MESS

1.0.0

Generated by Doxygen 1.8.11

Contents

1 M.E.S.S. - Matrix Equation Sparse Solver **2**

1.1 Welcome to M.E.S.S. 2

1.2 Features 2

1.3 Overview of the Functionality 2

1.4 Supported Matrix Equations 3

1.4.1 Algebraic Lyapunov equations 3

1.4.2 Continuous Time Algebraic Riccati equations 3

1.5 Installation 3

1.6 Contact 3

2 Tutorials **4**

3 Install Instructions **5**

4 Module Index **9**

4.1 Modules 9

5 Hierarchical Index **11**

5.1 Class Hierarchy 12

6 Data Structure Index **15**

6.1 Data Structures 15

7 File Index **19**

7.1 File List 19

8 Module Documentation **44**

8.1 General Type Definitions and Macros 44

8.1.1 Detailed Description 45

8.1.2 Macro Definition Documentation 45

8.1.3 Typedef Documentation 46

8.1.4 Enumeration Type Documentation 46

8.2 Matrix Operations 48

8.2.1 Detailed Description 49

8.3 Arithmetic Operations on Matrices 50

8.3.1 Detailed Description 52

8.3.2 Function Documentation 52

8.4 Arithmetic Operations on Matrix Columns 79

8.4.1 Detailed Description 79

8.4.2 Function Documentation 80

8.5 Computation of Matrix Norms 91

8.5.1 Detailed Description 92

8.5.2 Function Documentation 92

8.6 Allocate, Convert and Access to Matrices 109

8.6.1 Detailed Description 112

8.6.2 Function Documentation 112

8.7 Input/Output Operations with Matrices 155

8.7.1 Detailed Description 155

8.7.2 Function Documentation 155

8.8 Various Other Operations on Matrices 163

8.8.1 Detailed Description 164

8.8.2 Enumeration Type Documentation 164

8.8.3 Function Documentation 164

8.9 Data Structures, Types and Macros for Handling Matrices 177

8.9.1 Detailed Description 178

8.9.2 Macro Definition Documentation 178

8.9.3 Typedef Documentation . . . 180
8.9.4 Enumeration Type Documentation . . . 181
8.9.5 Function Documentation . . . 181
8.10 Generic matrix-vector Products . . . 183
8.10.1 Detailed Description . . . 183
8.10.2 Typedef Documentation . . . 184
8.10.3 Function Documentation . . . 184
8.11 Vector Operations . . . 187
8.11.1 Detailed Description . . . 187
8.12 Data Structures and Related Functions . . . 188
8.12.1 Detailed Description . . . 189
8.12.2 Typedef Documentation . . . 189
8.12.3 Function Documentation . . . 189
8.13 Arithmetic Operations with Vectors . . . 197
8.13.1 Detailed Description . . . 198
8.13.2 Macro Definition Documentation . . . 198
8.13.3 Function Documentation . . . 198
8.14 Various Other Operations with Vectors . . . 206
8.14.1 Detailed Description . . . 208
8.14.2 Function Documentation . . . 208
8.15 Direct Solvers for Linear Systems . . . 234
8.15.1 Detailed Description . . . 235
8.16 Generic Interface to Direct Solvers . . . 236
8.16.1 Detailed Description . . . 237
8.16.2 Macro Definition Documentation . . . 237
8.16.3 Typedef Documentation . . . 238
8.16.4 Function Documentation . . . 238
8.17 Direct Solvers for Linear Systems . . . 247
8.17.1 Detailed Description . . . 248
8.17.2 Enumeration Type Documentation . . . 248

8.17.3 Function Documentation . . . 248
8.18 Triangular Solvers . . . 265
8.18.1 Detailed Description . . . 266
8.18.2 Function Documentation . . . 266
8.19 Direct Solvers for Matrix Equations . . . 288
8.19.1 Detailed Description . . . 288
8.19.2 Function Documentation . . . 288
8.20 Matrix Decompositions . . . 297
8.20.1 Detailed Description . . . 297
8.20.2 Function Documentation . . . 297
8.21 Residual Computations . . . 300
8.21.1 Detailed Description . . . 301
8.21.2 Function Documentation . . . 301
8.22 Various Other Functions related to Direct Solvers. . . . 314
8.22.1 Detailed Description . . . 314
8.22.2 Typedef Documentation . . . 315
8.22.3 Function Documentation . . . 315
8.23 Solvers for Sets of Linear Systems . . . 317
8.23.1 Detailed Description . . . 317
8.24 Generic Interface to Multisolvers . . . 318
8.24.1 Detailed Description . . . 318
8.24.2 Typedef Documentation . . . 319
8.24.3 Enumeration Type Documentation . . . 319
8.24.4 Function Documentation . . . 319
8.25 Multisolvers . . . 322
8.25.1 Detailed Description . . . 322
8.25.2 Function Documentation . . . 322
8.26 Iterative Solvers for Linear Systems . . . 329
8.26.1 Detailed Description . . . 329
8.27 Types and Datastructures . . . 330

8.27.1 Detailed Description . . . 331
8.27.2 Macro Definition Documentation . . . 331
8.27.3 Typedef Documentation . . . 331
8.27.4 Function Documentation . . . 331
8.28 Solvers . . . 334
8.28.1 Detailed Description . . . 334
8.28.2 Function Documentation . . . 334
8.29 Preconditioners . . . 344
8.29.1 Detailed Description . . . 344
8.29.2 Function Documentation . . . 344
8.30 Dynamical Systems . . . 350
8.30.1 Detailed Description . . . 350
8.31 Interface for handling Dynamical Systems . . . 351
8.31.1 Detailed Description . . . 352
8.31.2 Macro Definition Documentation . . . 353
8.31.3 Typedef Documentation . . . 353
8.31.4 Function Documentation . . . 353
8.32 Model Order Reduction . . . 371
8.32.1 Detailed Description . . . 371
8.33 Balanced Truncation . . . 372
8.33.1 Detailed Description . . . 373
8.33.2 Macro Definition Documentation . . . 374
8.33.3 Typedef Documentation . . . 374
8.33.4 Function Documentation . . . 375
8.34 H2 Model Order Reduction . . . 392
8.34.1 Detailed Description . . . 394
8.34.2 Macro Definition Documentation . . . 395
8.34.3 Typedef Documentation . . . 396
8.34.4 Function Documentation . . . 396
8.35 Eigenvalue and Related Problems . . . 424

8.35.1 Detailed Description . . . 424
8.36 Dense Eigenvalue Solvers . . . 425
8.36.1 Detailed Description . . . 426
8.36.2 Function Documentation . . . 426
8.37 Sparse Eigenvalue Solvers . . . 442
8.37.1 Detailed Description . . . 443
8.37.2 Macro Definition Documentation . . . 444
8.37.3 Enumeration Type Documentation . . . 444
8.37.4 Function Documentation . . . 445
8.38 Low-Rank Methods for Matrix Equations . . . 464
8.38.1 Detailed Description . . . 464
8.39 Generic Functions and Data Structures . . . 465
8.39.1 Detailed Description . . . 466
8.39.2 Typedef Documentation . . . 467
8.39.3 Enumeration Type Documentation . . . 467
8.39.4 Function Documentation . . . 468
8.40 The ADI and the ADI-NM Algorithm . . . 483
8.40.1 Detailed Description . . . 484
8.40.2 Function Documentation . . . 484
8.41 Equation Generators . . . 502
8.41.1 Detailed Description . . . 503
8.41.2 Function Documentation . . . 504
8.42 Define own Matrix Generators . . . 534
8.42.1 Detailed Description . . . 535
8.42.2 Function Documentation . . . 535
8.43 Additional Matrix Equation Solvers . . . 553
8.43.1 Detailed Description . . . 554
8.43.2 Enumeration Type Documentation . . . 554
8.43.3 Function Documentation . . . 555
8.44 Miscellaneous Functions . . . 569

8.44.1 Detailed Description . . . 570
8.44.2 Function Documentation . . . 570
8.45 Easy Frontends to Main Functions . . . 585
8.45.1 Detailed Description . . . 585
8.45.2 Function Documentation . . . 585
8.46 Graph Algorithms . . . 595
8.46.1 Detailed Description . . . 595
8.46.2 Function Documentation . . . 595
8.47 Matrix Generators . . . 598
8.47.1 Detailed Description . . . 598
8.48 FDM Matrix Generators . . . 599
8.48.1 Detailed Description . . . 599
8.48.2 Typedef Documentation . . . 600
8.48.3 Function Documentation . . . 600
8.49 Error Handling . . . 604
8.49.1 Detailed Description . . . 604
8.50 Error Codes and Handling . . . 605
8.50.1 Detailed Description . . . 606
8.50.2 Macro Definition Documentation . . . 606
8.50.3 Function Documentation . . . 608
8.50.4 Variable Documentation . . . 609
8.51 Error Macros . . . 610
8.51.1 Detailed Description . . . 610
8.51.2 Macro Definition Documentation . . . 610
8.52 Interfaces to other Programming Languages and Libraries . . . 612
8.52.1 Detailed Description . . . 612
8.53 Interface to CSparse . . . 613
8.53.1 Detailed Description . . . 613
8.53.2 Function Documentation . . . 613
8.54 Interface to Cholmod . . . 619

8.54.1 Detailed Description . . . 619
8.54.2 Function Documentation . . . 619
8.55 Interface to Python . . . 620
8.55.1 Detailed Description . . . 622
8.55.2 Macro Definition Documentation . . . 623
8.55.3 Function Documentation . . . 625
8.55.4 Variable Documentation . . . 655
8.56 Plotting Interface . . . 656
8.56.1 Detailed Description . . . 656
8.57 X11 Function Plotter . . . 657
8.57.1 Detailed Description . . . 658
8.57.2 Typedef Documentation . . . 658
8.57.3 Function Documentation . . . 658
8.58 Plot Export . . . 663
8.58.1 Detailed Description . . . 664
8.58.2 Typedef Documentation . . . 665
8.58.3 Enumeration Type Documentation . . . 665
8.58.4 Function Documentation . . . 665
8.59 Miscellaneous Helper Functions . . . 672
8.59.1 Detailed Description . . . 675
8.59.2 Macro Definition Documentation . . . 675
8.59.3 Typedef Documentation . . . 675
8.59.4 Function Documentation . . . 675
8.60 Hashtable . . . 690
8.61 A Compact Loader Pool Implementation . . . 691
8.61.1 Detailed Description . . . 692
8.61.2 Typedef Documentation . . . 692
8.61.3 Function Documentation . . . 692
8.62 Time Measurement . . . 696
8.62.1 Detailed Description . . . 696

8.62.2 Typedef Documentation . . . 697
8.62.3 Function Documentation . . . 697
8.63 Various Macros . . . 700
8.63.1 Detailed Description . . . 702
8.63.2 Macro Definition Documentation . . . 703
8.64 Software Testing . . . 712
8.64.1 Detailed Description . . . 713
8.65 Basic Matrix and Vector Operations . . . 714
8.65.1 Detailed Description . . . 716
8.66 Direct Solvers for Linear Systems . . . 718
8.66.1 Detailed Description . . . 719
8.67 Eigenvalue and Related Problems . . . 720
8.67.1 Detailed Description . . . 721
8.68 Input/Output Operations . . . 722
8.68.1 Detailed Description . . . 722
8.69 Iterative Solvers for Linear Systems . . . 723
8.69.1 Detailed Description . . . 723
8.70 Low-Rank Methods for Matrix Equations . . . 724
8.70.1 Detailed Description . . . 725
8.71 Storage Formats . . . 726
8.71.1 Detailed Description . . . 726
8.72 Miscellaneous Functions and Functions Related to Vectors . . . 727
8.72.1 Detailed Description . . . 727
8.73 Interfaces_matlab . . . 728
8.73.1 Detailed Description . . . 730
8.73.2 Macro Definition Documentation . . . 730
8.73.3 Function Documentation . . . 730

9 Data Structure Documentation **764**

9.1 __glyap_dae1 Struct Reference . . . 764

9.1.1 Detailed Description . . . 765

9.1.2 Field Documentation . . . 765

9.2 __glyap_dae2 Struct Reference . . . 766

9.2.1 Detailed Description . . . 768

9.2.2 Field Documentation . . . 768

9.3 __glyap_so1 Struct Reference . . . 769

9.3.1 Detailed Description . . . 771

9.3.2 Field Documentation . . . 771

9.4 __glyap_so2 Struct Reference . . . 772

9.4.1 Detailed Description . . . 773

9.4.2 Field Documentation . . . 773

9.5 __ilu0_data Struct Reference . . . 774

9.5.1 Detailed Description . . . 774

9.5.2 Field Documentation . . . 775

9.6 __iluk Struct Reference . . . 775

9.6.1 Field Documentation . . . 776

9.7 __lyap Struct Reference . . . 776

9.7.1 Detailed Description . . . 777

9.7.2 Field Documentation . . . 777

9.8 __matlab_equation Struct Reference . . . 778

9.8.1 Field Documentation . . . 779

9.9 __mess_lrcfadi_step Struct Reference . . . 780

9.9.1 Detailed Description . . . 781

9.9.2 Field Documentation . . . 781

9.10 __mess_plotdata Struct Reference . . . 782

9.10.1 Detailed Description . . . 782

9.10.2 Field Documentation . . . 782

9.11 __mess_plotExportData Struct Reference . . . 783

9.11.1 Detailed Description . . . 783
9.11.2 Field Documentation . . . 784
9.12 __mess_plotter Struct Reference . . . 784
9.12.1 Detailed Description . . . 786
9.12.2 Field Documentation . . . 786
9.13 __mess_plotterScript Struct Reference . . . 788
9.13.1 Detailed Description . . . 788
9.13.2 Field Documentation . . . 789
9.14 __mess_solver_options Struct Reference . . . 789
9.14.1 Detailed Description . . . 790
9.14.2 Field Documentation . . . 790
9.15 __mess_solver_status Struct Reference . . . 791
9.15.1 Detailed Description . . . 791
9.15.2 Field Documentation . . . 791
9.16 __presordata Struct Reference . . . 792
9.16.1 Field Documentation . . . 793
9.17 __stable_lyap Struct Reference . . . 793
9.17.1 Field Documentation . . . 794
9.18 _equations_t Struct Reference . . . 794
9.18.1 Detailed Description . . . 795
9.18.2 Field Documentation . . . 795
9.19 _glyap3_st Struct Reference . . . 795
9.19.1 Detailed Description . . . 796
9.19.2 Field Documentation . . . 796
9.20 _mess_call Struct Reference . . . 796
9.20.1 Field Documentation . . . 797
9.21 _scalepoints Struct Reference . . . 797
9.21.1 Detailed Description . . . 797
9.21.2 Field Documentation . . . 797
9.22 _sylv_solver_d_st Struct Reference . . . 798

9.22.1 Detailed Description . . . 799
9.22.2 Field Documentation . . . 799
9.23 _sylv_solver_sd_st Struct Reference . . . 800
9.23.1 Detailed Description . . . 801
9.23.2 Field Documentation . . . 801
9.24 mess_equation_st::AINV_st Struct Reference . . . 802
9.24.1 Detailed Description . . . 802
9.24.2 Field Documentation . . . 802
9.25 mess_equation_st::ApEINV_st Struct Reference . . . 803
9.25.1 Detailed Description . . . 803
9.25.2 Field Documentation . . . 803
9.26 mess_equation_st::ApEX_st Struct Reference . . . 804
9.26.1 Detailed Description . . . 804
9.26.2 Field Documentation . . . 804
9.27 mess_equation_st::AX_st Struct Reference . . . 805
9.27.1 Detailed Description . . . 805
9.27.2 Field Documentation . . . 805
9.28 banded_lapack Struct Reference . . . 806
9.28.1 Field Documentation . . . 807
9.29 bicgstab_solver Struct Reference . . . 807
9.29.1 Field Documentation . . . 808
9.30 CCSRFmt Struct Reference . . . 808
9.30.1 Field Documentation . . . 809
9.31 cholesky_solver Struct Reference . . . 809
9.31.1 Field Documentation . . . 809
9.32 csparse_solver Struct Reference . . . 810
9.32.1 Detailed Description . . . 810
9.32.2 Field Documentation . . . 811
9.33 csparse_solver_complex Struct Reference . . . 811
9.33.1 Detailed Description . . . 811

9.33.2 Field Documentation . . . 811
9.34 mess_equation_st::EINV_st Struct Reference . . . 812
9.34.1 Detailed Description . . . 812
9.34.2 Field Documentation . . . 812
9.35 mess_equation_st::EX_st Struct Reference . . . 813
9.35.1 Detailed Description . . . 813
9.35.2 Field Documentation . . . 813
9.36 hessenberg_solver Struct Reference . . . 814
9.36.1 Detailed Description . . . 814
9.36.2 Field Documentation . . . 814
9.37 int_list Struct Reference . . . 815
9.37.1 Detailed Description . . . 815
9.37.2 Field Documentation . . . 816
9.38 jobdata_st Struct Reference . . . 817
9.38.1 Field Documentation . . . 818
9.39 lapack_solver Struct Reference . . . 819
9.39.1 Detailed Description . . . 819
9.39.2 Field Documentation . . . 819
9.40 lapackqr_solver Struct Reference . . . 820
9.40.1 Detailed Description . . . 821
9.40.2 Field Documentation . . . 821
9.41 mat_diff Struct Reference . . . 822
9.41.1 Detailed Description . . . 822
9.41.2 Field Documentation . . . 822
9.42 mess_bt_options_st Struct Reference . . . 823
9.42.1 Detailed Description . . . 823
9.42.2 Field Documentation . . . 824
9.43 mess_bt_status_st Struct Reference . . . 824
9.43.1 Detailed Description . . . 826
9.43.2 Field Documentation . . . 826

9.44 MESS_datatype_t Class Reference . . . 827
9.44.1 Constructor & Destructor Documentation . . . 828
9.44.2 Member Function Documentation . . . 828
9.44.3 Field Documentation . . . 828
9.45 MESS_direct Class Reference . . . 829
9.45.1 Constructor & Destructor Documentation . . . 830
9.45.2 Member Function Documentation . . . 831
9.45.3 Field Documentation . . . 832
9.46 MESS_direct_cholpackage_t Class Reference . . . 832
9.46.1 Constructor & Destructor Documentation . . . 833
9.46.2 Member Function Documentation . . . 833
9.46.3 Field Documentation . . . 834
9.47 mess_direct_levelset_st Struct Reference . . . 834
9.47.1 Field Documentation . . . 835
9.48 MESS_direct_lupackage_t Class Reference . . . 835
9.48.1 Constructor & Destructor Documentation . . . 836
9.48.2 Member Function Documentation . . . 836
9.48.3 Field Documentation . . . 836
9.49 mess_direct_st Struct Reference . . . 837
9.49.1 Detailed Description . . . 838
9.49.2 Field Documentation . . . 838
9.50 mess_dynsys_st Struct Reference . . . 840
9.50.1 Detailed Description . . . 841
9.50.2 Field Documentation . . . 841
9.51 mess_eigen_arpack_options_t Struct Reference . . . 842
9.51.1 Detailed Description . . . 843
9.51.2 Field Documentation . . . 843
9.52 MESS_equation Class Reference . . . 844
9.52.1 Constructor & Destructor Documentation . . . 845
9.52.2 Member Function Documentation . . . 846

9.52.3 Field Documentation . . . 847
9.53 mess_equation_st Struct Reference . . . 847
9.53.1 Detailed Description . . . 848
9.53.2 Field Documentation . . . 848
9.54 MESS_equation_t Class Reference . . . 850
9.54.1 Constructor & Destructor Documentation . . . 851
9.54.2 Member Function Documentation . . . 851
9.54.3 Field Documentation . . . 851
9.55 mess_freelist_st Struct Reference . . . 852
9.55.1 Detailed Description . . . 852
9.55.2 Field Documentation . . . 853
9.56 mess_h2_options_st Struct Reference . . . 854
9.56.1 Detailed Description . . . 854
9.56.2 Field Documentation . . . 854
9.57 mess_h2_status_st Struct Reference . . . 855
9.57.1 Detailed Description . . . 856
9.57.2 Field Documentation . . . 856
9.58 mess_krylov_arnoldi_st Struct Reference . . . 857
9.58.1 Detailed Description . . . 858
9.58.2 Field Documentation . . . 858
9.59 MESS_matrix Class Reference . . . 859
9.59.1 Constructor & Destructor Documentation . . . 861
9.59.2 Member Function Documentation . . . 863
9.59.3 Field Documentation . . . 869
9.60 mess_matrix_file_st Struct Reference . . . 869
9.60.1 Detailed Description . . . 870
9.60.2 Field Documentation . . . 870
9.61 mess_matrix_st Struct Reference . . . 871
9.61.1 Detailed Description . . . 872
9.61.2 Field Documentation . . . 872

9.62 MESS_memusage_t Class Reference . . . 873
9.62.1 Constructor & Destructor Documentation . . . 874
9.62.2 Member Function Documentation . . . 874
9.62.3 Field Documentation . . . 875
9.63 MESS_multidirect Class Reference . . . 875
9.63.1 Constructor & Destructor Documentation . . . 877
9.63.2 Member Function Documentation . . . 878
9.63.3 Field Documentation . . . 879
9.64 mess_multidirect_st Struct Reference . . . 879
9.64.1 Detailed Description . . . 880
9.64.2 Field Documentation . . . 880
9.65 MESS_multidirect_t Class Reference . . . 882
9.65.1 Constructor & Destructor Documentation . . . 883
9.65.2 Member Function Documentation . . . 883
9.65.3 Field Documentation . . . 884
9.66 mess_mvpcall_st Struct Reference . . . 884
9.66.1 Detailed Description . . . 885
9.66.2 Field Documentation . . . 885
9.67 MESS_operation_t Class Reference . . . 885
9.67.1 Constructor & Destructor Documentation . . . 887
9.67.2 Member Function Documentation . . . 887
9.67.3 Field Documentation . . . 887
9.68 MESS_options Class Reference . . . 887
9.68.1 Constructor & Destructor Documentation . . . 889
9.68.2 Member Function Documentation . . . 890
9.68.3 Field Documentation . . . 892
9.69 mess_options_st Struct Reference . . . 892
9.69.1 Detailed Description . . . 894
9.69.2 Field Documentation . . . 894
9.70 MESS_parameter_t Class Reference . . . 897

9.70.1 Constructor & Destructor Documentation . . . 898
9.70.2 Member Function Documentation . . . 898
9.70.3 Field Documentation . . . 898
9.71 mess_precond_st Struct Reference . . . 899
9.71.1 Detailed Description . . . 899
9.71.2 Field Documentation . . . 899
9.72 MESS_residual_t Class Reference . . . 900
9.72.1 Constructor & Destructor Documentation . . . 901
9.72.2 Member Function Documentation . . . 901
9.72.3 Field Documentation . . . 901
9.73 MESS_status Class Reference . . . 902
9.73.1 Constructor & Destructor Documentation . . . 903
9.73.2 Member Function Documentation . . . 904
9.73.3 Field Documentation . . . 905
9.74 mess_status_st Struct Reference . . . 905
9.74.1 Field Documentation . . . 907
9.75 MESS_storage_t Class Reference . . . 908
9.75.1 Constructor & Destructor Documentation . . . 909
9.75.2 Member Function Documentation . . . 909
9.75.3 Field Documentation . . . 910
9.76 mess_threadpool_st Struct Reference . . . 910
9.76.1 Detailed Description . . . 911
9.76.2 Field Documentation . . . 911
9.77 mess_threadpooljob_st Struct Reference . . . 912
9.77.1 Detailed Description . . . 912
9.77.2 Field Documentation . . . 913
9.78 mess_timer_st Struct Reference . . . 913
9.78.1 Detailed Description . . . 913
9.78.2 Field Documentation . . . 914
9.79 MESS_vector Class Reference . . . 914

9.79.1 Constructor & Destructor Documentation . . . 917
9.79.2 Member Function Documentation . . . 918
9.79.3 Field Documentation . . . 921
9.80 mess_vector_st Struct Reference . . . 921
9.80.1 Detailed Description . . . 922
9.80.2 Field Documentation . . . 922
9.81 mklpardiso_solver Struct Reference . . . 923
9.81.1 Field Documentation . . . 924
9.82 module_state Struct Reference . . . 924
9.82.1 Field Documentation . . . 925
9.83 multi_umfpack Struct Reference . . . 925
9.83.1 Field Documentation . . . 925
9.84 multilu Struct Reference . . . 926
9.84.1 Field Documentation . . . 927
9.85 mvpdata Struct Reference . . . 929
9.85.1 Field Documentation . . . 929
9.86 mvpdatag Struct Reference . . . 930
9.86.1 Field Documentation . . . 931
9.87 newlu_data Struct Reference . . . 932
9.87.1 Field Documentation . . . 933
9.88 norm2data Struct Reference . . . 933
9.88.1 Field Documentation . . . 934
9.89 norminv2data Struct Reference . . . 934
9.89.1 Field Documentation . . . 935
9.90 pair_mess_int_t2 Struct Reference . . . 935
9.90.1 Detailed Description . . . 935
9.90.2 Field Documentation . . . 935
9.91 pair_mess_int_t2_complex Struct Reference . . . 935
9.91.1 Detailed Description . . . 936
9.91.2 Field Documentation . . . 936

9.92 pair_mess_int_t2_double Struct Reference 936
9.92.1 Detailed Description 936
9.92.2 Field Documentation 937
9.93 pair_mess_int_t_complex Struct Reference 937
9.93.1 Detailed Description 937
9.93.2 Field Documentation 937
9.94 pair_mess_int_t_double Struct Reference 938
9.94.1 Detailed Description 938
9.94.2 Field Documentation 938
9.95 parameter_mvp_call Struct Reference 939
9.95.1 Field Documentation 939
9.96 rc2data Struct Reference 940
9.96.1 Field Documentation 940
9.97 res2_data_st Struct Reference 941
9.97.1 Field Documentation 942
9.98 res2nm_data_st Struct Reference 943
9.98.1 Field Documentation 944
9.99 th_done_entry Struct Reference 944
9.99.1 Field Documentation 945
9.100 umfpack_control_solver Struct Reference 945
9.100.1 Field Documentation 946
9.101 VWjob Struct Reference 946
9.101.1 Field Documentation 947

10 File Documentation 947

10.1 __my_module__my_sub.c File Reference . . . 947

10.1.1 Function Documentation . . . 948

10.2 __my_module__my_sub.c File Reference . . . 948

10.2.1 Function Documentation . . . 948

10.3 __my_module_MOD_my_sub.c File Reference . . . 948

10.3.1 Function Documentation . . . 948

10.4 __my_module_MOD_my_sub.c File Reference . . . 948

10.4.1 Function Documentation . . . 948

10.5 __my_module_NMOD_my_sub.c File Reference . . . 948

10.5.1 Function Documentation . . . 948

10.6 __my_module_NMOD_my_sub.c File Reference . . . 948

10.6.1 Function Documentation . . . 949

10.7 __mymodule__mysub.c File Reference . . . 949

10.7.1 Function Documentation . . . 949

10.8 __mymodule__mysub.c File Reference . . . 949

10.8.1 Function Documentation . . . 949

10.9 __mymodule_MOD_mysub.c File Reference . . . 949

10.9.1 Function Documentation . . . 949

10.10 __mymodule_MOD_mysub.c File Reference . . . 949

10.10.1 Function Documentation . . . 949

10.11 __mymodule_NMOD_mysub.c File Reference . . . 949

10.11.1 Function Documentation . . . 950

10.12 __mymodule_NMOD_mysub.c File Reference . . . 950

10.12.1 Function Documentation . . . 950

10.13 add.c File Reference . . . 950

10.13.1 Detailed Description . . . 951

10.13.2 Function Documentation . . . 951

10.14 addcol.c File Reference . . . 959

10.14.1 Detailed Description . . . 960

10.15 alloc.c File Reference . . . 960
10.15.1 Detailed Description . . . 960
10.15.2 Macro Definition Documentation . . . 960
10.15.3 Function Documentation . . . 960
10.15.4 Variable Documentation . . . 961
10.16 alloc.c File Reference . . . 961
10.16.1 Detailed Description . . . 961
10.17 amd.c File Reference . . . 961
10.17.1 Detailed Description . . . 962
10.17.2 Macro Definition Documentation . . . 962
10.18 any.c File Reference . . . 962
10.18.1 Detailed Description . . . 962
10.19 any.c File Reference . . . 962
10.19.1 Detailed Description . . . 963
10.20 arnoldi.c File Reference . . . 963
10.20.1 Detailed Description . . . 963
10.20.2 Function Documentation . . . 964
10.21 arnoldi_template.c File Reference . . . 967
10.21.1 Detailed Description . . . 968
10.21.2 Macro Definition Documentation . . . 968
10.21.3 Typedef Documentation . . . 968
10.21.4 Function Documentation . . . 968
10.22 arpack.c File Reference . . . 974
10.22.1 Detailed Description . . . 975
10.22.2 Macro Definition Documentation . . . 975
10.22.3 Function Documentation . . . 975
10.23 axpy.c File Reference . . . 977
10.23.1 Detailed Description . . . 978
10.23.2 Function Documentation . . . 978
10.23.3 Variable Documentation . . . 978

10.24 backslash.c File Reference . . . 978
10.24.1 Detailed Description . . . 979
10.25 banded.c File Reference . . . 979
10.25.1 Detailed Description . . . 979
10.25.2 Macro Definition Documentation . . . 980
10.25.3 Function Documentation . . . 980
10.26 bandwidth.c File Reference . . . 985
10.26.1 Detailed Description . . . 985
10.27 ber_sgn_fac.c File Reference . . . 985
10.27.1 Detailed Description . . . 985
10.28 bicgstab.c File Reference . . . 985
10.28.1 Detailed Description . . . 986
10.28.2 Macro Definition Documentation . . . 986
10.28.3 Function Documentation . . . 986
10.29 bicgstab.c File Reference . . . 988
10.29.1 Detailed Description . . . 988
10.30 biorth.c File Reference . . . 989
10.30.1 Detailed Description . . . 989
10.31 blas_defs.h File Reference . . . 989
10.31.1 Detailed Description . . . 994
10.31.2 Typedef Documentation . . . 994
10.31.3 Function Documentation . . . 994
10.31.4 Variable Documentation . . . 998
10.32 blockpart.c File Reference . . . 1005
10.32.1 Detailed Description . . . 1005
10.32.2 Function Documentation . . . 1005
10.33 call_macro.h File Reference . . . 1006
10.33.1 Macro Definition Documentation . . . 1006
10.34 callback_equation.c File Reference . . . 1006
10.34.1 Detailed Description . . . 1007

10.34.2 Macro Definition Documentation . . . 1007
10.34.3 Function Documentation . . . 1009
10.35 care.c File Reference . . . 1010
10.35.1 Detailed Description . . . 1011
10.36 care.c File Reference . . . 1011
10.37 care.c File Reference . . . 1011
10.37.1 Detailed Description . . . 1011
10.38 cat.c File Reference . . . 1011
10.38.1 Detailed Description . . . 1012
10.38.2 Function Documentation . . . 1012
10.39 cat.c File Reference . . . 1020
10.39.1 Detailed Description . . . 1020
10.39.2 Variable Documentation . . . 1020
10.40 category.dox File Reference . . . 1020
10.41 cg.c File Reference . . . 1020
10.41.1 Detailed Description . . . 1020
10.42 check_add.c File Reference . . . 1021
10.42.1 Detailed Description . . . 1021
10.42.2 Macro Definition Documentation . . . 1021
10.42.3 Function Documentation . . . 1021
10.43 check_addcol2p.c File Reference . . . 1022
10.43.1 Detailed Description . . . 1022
10.43.2 Macro Definition Documentation . . . 1022
10.43.3 Function Documentation . . . 1022
10.44 check_any.c File Reference . . . 1023
10.44.1 Detailed Description . . . 1023
10.44.2 Macro Definition Documentation . . . 1024
10.44.3 Function Documentation . . . 1024
10.45 check_arnoldi_template.c File Reference . . . 1025
10.45.1 Detailed Description . . . 1025

10.45.2 Macro Definition Documentation . . . 1025
10.45.3 Function Documentation . . . 1025
10.46 check_arnoldi_template_nrm.c File Reference . . . 1026
10.46.1 Detailed Description . . . 1027
10.46.2 Function Documentation . . . 1027
10.47 check_arpack.c File Reference . . . 1028
10.47.1 Detailed Description . . . 1028
10.47.2 Function Documentation . . . 1028
10.48 check_arpack_lanczos.c File Reference . . . 1031
10.48.1 Detailed Description . . . 1032
10.48.2 Function Documentation . . . 1032
10.49 check_backslash.c File Reference . . . 1035
10.49.1 Detailed Description . . . 1036
10.49.2 Macro Definition Documentation . . . 1036
10.49.3 Function Documentation . . . 1037
10.50 check_bandwidth.c File Reference . . . 1037
10.50.1 Detailed Description . . . 1037
10.50.2 Function Documentation . . . 1038
10.51 check_ber_sign_fac.c File Reference . . . 1038
10.51.1 Detailed Description . . . 1039
10.51.2 Function Documentation . . . 1039
10.52 check_biorth.c File Reference . . . 1040
10.52.1 Detailed Description . . . 1041
10.52.2 Function Documentation . . . 1041
10.53 check_cat.c File Reference . . . 1042
10.53.1 Detailed Description . . . 1043
10.53.2 Macro Definition Documentation . . . 1043
10.53.3 Function Documentation . . . 1043
10.53.4 Variable Documentation . . . 1045
10.54 check_cholmod.c File Reference . . . 1047

10.54.1 Detailed Description . . . 1047
10.54.2 Macro Definition Documentation . . . 1047
10.54.3 Function Documentation . . . 1048
10.55 check_cimport.c File Reference . . . 1049
10.55.1 Detailed Description . . . 1050
10.55.2 Function Documentation . . . 1050
10.56 check_cimport_c2d.c File Reference . . . 1051
10.56.1 Detailed Description . . . 1051
10.56.2 Function Documentation . . . 1051
10.57 check_colaxpy2.c File Reference . . . 1053
10.57.1 Detailed Description . . . 1053
10.57.2 Function Documentation . . . 1053
10.58 check_coldot.c File Reference . . . 1053
10.58.1 Detailed Description . . . 1054
10.58.2 Macro Definition Documentation . . . 1054
10.58.3 Function Documentation . . . 1054
10.58.4 Variable Documentation . . . 1055
10.59 check_coldotc.c File Reference . . . 1055
10.59.1 Detailed Description . . . 1056
10.59.2 Function Documentation . . . 1056
10.59.3 Variable Documentation . . . 1056
10.60 check_coldotcE.c File Reference . . . 1056
10.60.1 Detailed Description . . . 1057
10.60.2 Macro Definition Documentation . . . 1057
10.60.3 Function Documentation . . . 1057
10.61 check_coldotE.c File Reference . . . 1058
10.61.1 Detailed Description . . . 1058
10.61.2 Macro Definition Documentation . . . 1059
10.61.3 Function Documentation . . . 1059
10.62 check_colnorm.c File Reference . . . 1059

10.62.1 Detailed Description 1060
10.62.2 Macro Definition Documentation 1060
10.62.3 Function Documentation 1060
10.62.4 Variable Documentation 1061
10.63check_colscale.c File Reference 1061
10.63.1 Detailed Description 1062
10.63.2 Macro Definition Documentation 1062
10.63.3 Function Documentation 1062
10.63.4 Variable Documentation 1063
10.64check_colsub.c File Reference 1063
10.64.1 Detailed Description 1064
10.64.2 Macro Definition Documentation 1064
10.64.3 Function Documentation 1064
10.65check_colvecaxpy.c File Reference 1065
10.65.1 Detailed Description 1066
10.65.2 Macro Definition Documentation 1066
10.65.3 Function Documentation 1066
10.66check_colvecdot.c File Reference 1067
10.66.1 Detailed Description 1068
10.66.2 Function Documentation 1068
10.66.3 Variable Documentation 1069
10.67check_colvecdotc.c File Reference 1069
10.67.1 Detailed Description 1069
10.67.2 Function Documentation 1070
10.67.3 Variable Documentation 1070
10.68check_condest.c File Reference 1071
10.68.1 Detailed Description 1071
10.68.2 Macro Definition Documentation 1071
10.68.3 Function Documentation 1071
10.68.4 Variable Documentation 1072

10.69 check_convert.c File Reference . . . 1073
10.69.1 Detailed Description . . . 1073
10.69.2 Function Documentation . . . 1073
10.70 check_dae2_handles_adi.c File Reference . . . 1075
10.70.1 Detailed Description . . . 1075
10.70.2 Macro Definition Documentation . . . 1076
10.70.3 Function Documentation . . . 1076
10.71 check_dae2_handles_para.c File Reference . . . 1077
10.71.1 Detailed Description . . . 1078
10.71.2 Macro Definition Documentation . . . 1078
10.71.3 Function Documentation . . . 1078
10.72 check_decomp.c File Reference . . . 1079
10.72.1 Detailed Description . . . 1080
10.72.2 Function Documentation . . . 1080
10.73 check_dense_nm.c File Reference . . . 1081
10.73.1 Detailed Description . . . 1081
10.73.2 Function Documentation . . . 1081
10.74 check_dense_nm_tridiag.c File Reference . . . 1082
10.74.1 Detailed Description . . . 1083
10.74.2 Function Documentation . . . 1083
10.75 check_det.c File Reference . . . 1083
10.75.1 Detailed Description . . . 1084
10.75.2 Function Documentation . . . 1084
10.76 check_dgeqp3.c File Reference . . . 1085
10.76.1 Detailed Description . . . 1085
10.76.2 Function Documentation . . . 1085
10.77 check_diag.c File Reference . . . 1086
10.77.1 Detailed Description . . . 1087
10.77.2 Macro Definition Documentation . . . 1087
10.77.3 Function Documentation . . . 1087

10.77.4 Variable Documentation . . . 1088
10.78 check_direct_care.c File Reference . . . 1088
10.78.1 Detailed Description . . . 1089
10.78.2 Function Documentation . . . 1089
10.79 check_dynorm2.c File Reference . . . 1089
10.79.1 Detailed Description . . . 1089
10.79.2 Function Documentation . . . 1089
10.80 check_dynorm2t.c File Reference . . . 1091
10.80.1 Detailed Description . . . 1091
10.80.2 Function Documentation . . . 1091
10.81 check_dynorm_indefinite2.c File Reference . . . 1092
10.81.1 Detailed Description . . . 1092
10.81.2 Function Documentation . . . 1092
10.82 check_dynorm_indefinitef.c File Reference . . . 1094
10.82.1 Detailed Description . . . 1095
10.82.2 Function Documentation . . . 1095
10.83 check_dynormf.c File Reference . . . 1096
10.83.1 Detailed Description . . . 1096
10.83.2 Function Documentation . . . 1096
10.84 check_dynormft.c File Reference . . . 1097
10.84.1 Detailed Description . . . 1097
10.84.2 Function Documentation . . . 1097
10.85 check_eig.c File Reference . . . 1098
10.85.1 Detailed Description . . . 1099
10.85.2 Function Documentation . . . 1099
10.86 check_eig2.c File Reference . . . 1101
10.86.1 Detailed Description . . . 1102
10.86.2 Function Documentation . . . 1102
10.87 check_eig_complex.c File Reference . . . 1104
10.87.1 Detailed Description . . . 1105

10.87.2 Function Documentation . . . 1105
10.88 check_eig_hessenberg_schur.c File Reference . . . 1107
10.88.1 Detailed Description . . . 1108
10.88.2 Function Documentation . . . 1108
10.89 check_eig_hessenberg_schur_largest_ev.c File Reference . . . 1109
10.89.1 Detailed Description . . . 1109
10.89.2 Function Documentation . . . 1109
10.90 check_eliminate_zeros.c File Reference . . . 1110
10.90.1 Detailed Description . . . 1111
10.90.2 Function Documentation . . . 1111
10.91 check_eps.c File Reference . . . 1112
10.91.1 Detailed Description . . . 1113
10.91.2 Function Documentation . . . 1113
10.92 check_fputs_block.c File Reference . . . 1113
10.92.1 Detailed Description . . . 1113
10.92.2 Function Documentation . . . 1114
10.93 check_freelist.c File Reference . . . 1114
10.93.1 Detailed Description . . . 1114
10.93.2 Function Documentation . . . 1114
10.94 check_fro_inn.c File Reference . . . 1115
10.94.1 Detailed Description . . . 1116
10.94.2 Function Documentation . . . 1116
10.94.3 Variable Documentation . . . 1116
10.95 check_gaxpy.c File Reference . . . 1116
10.95.1 Detailed Description . . . 1117
10.95.2 Macro Definition Documentation . . . 1117
10.95.3 Function Documentation . . . 1117
10.96 check_getelement.c File Reference . . . 1118
10.96.1 Detailed Description . . . 1119
10.96.2 Function Documentation . . . 1119

10.97 check_getLU.c File Reference . . . 1120
10.97.1 Detailed Description . . . 1120
10.97.2 Function Documentation . . . 1120
10.98 check_gsign.c File Reference . . . 1123
10.98.1 Detailed Description . . . 1124
10.98.2 Function Documentation . . . 1124
10.99 check_hessenberg.c File Reference . . . 1125
10.99.1 Detailed Description . . . 1126
10.99.2 Function Documentation . . . 1126
10.100 check_itsolver.c File Reference . . . 1127
10.100.1 Detailed Description . . . 1127
10.100.2 Function Documentation . . . 1127
10.101 check_kron.c File Reference . . . 1129
10.101.1 Detailed Description . . . 1130
10.101.2 Macro Definition Documentation . . . 1130
10.101.3 Function Documentation . . . 1130
10.102 check_lapackqr.c File Reference . . . 1132
10.102.1 Detailed Description . . . 1132
10.102.2 Function Documentation . . . 1133
10.103 check_lift.c File Reference . . . 1136
10.103.1 Detailed Description . . . 1137
10.103.2 Macro Definition Documentation . . . 1137
10.103.3 Function Documentation . . . 1138
10.104 check_linesearch.c File Reference . . . 1138
10.104.1 Detailed Description . . . 1138
10.104.2 Macro Definition Documentation . . . 1139
10.104.3 Function Documentation . . . 1139
10.105 check_lradi_dae1.c File Reference . . . 1140
10.105.1 Detailed Description . . . 1140
10.105.2 Function Documentation . . . 1140

10.106 check_lradi_dae2.c File Reference . . . 1141
10.106.1 Detailed Description . . . 1142
10.106.2 Function Documentation . . . 1142
10.107 check_lradi_glyap.c File Reference . . . 1143
10.107.1 Detailed Description . . . 1143
10.107.2 Function Documentation . . . 1143
10.108 check_lradi_so.c File Reference . . . 1144
10.108.1 Detailed Description . . . 1145
10.108.2 Function Documentation . . . 1145
10.109 check_lradi_unsym.c File Reference . . . 1146
10.109.1 Detailed Description . . . 1146
10.109.2 Function Documentation . . . 1146
10.110 check_lrnm_dae1.c File Reference . . . 1148
10.110.1 Detailed Description . . . 1149
10.110.2 Function Documentation . . . 1149
10.111 check_lrnm_dae2.c File Reference . . . 1150
10.111.1 Detailed Description . . . 1151
10.111.2 Function Documentation . . . 1151
10.112 check_lrnm_g.c File Reference . . . 1152
10.112.1 Detailed Description . . . 1152
10.112.2 Function Documentation . . . 1152
10.113 check_lrnm_so.c File Reference . . . 1153
10.113.1 Detailed Description . . . 1154
10.113.2 Function Documentation . . . 1154
10.114 check_lrnm_unsym.c File Reference . . . 1155
10.114.1 Detailed Description . . . 1155
10.114.2 Function Documentation . . . 1155
10.115 check_lyap_sign_fac.c File Reference . . . 1157
10.115.1 Detailed Description . . . 1158
10.115.2 Function Documentation . . . 1158

10.116 check_matrix_biorth.c File Reference . . . 1159
10.116.1 Detailed Description . . . 1160
10.116.2 Macro Definition Documentation . . . 1160
10.116.3 Function Documentation . . . 1160
10.117 check_matrix_colaxpy.c File Reference . . . 1161
10.117.1 Detailed Description . . . 1161
10.117.2 Macro Definition Documentation . . . 1161
10.117.3 Function Documentation . . . 1162
10.118 check_matrix_gbiorth.c File Reference . . . 1162
10.118.1 Detailed Description . . . 1163
10.118.2 Macro Definition Documentation . . . 1163
10.118.3 Function Documentation . . . 1163
10.119 check_matrix_null.c File Reference . . . 1164
10.119.1 Detailed Description . . . 1164
10.119.2 Macro Definition Documentation . . . 1165
10.119.3 Function Documentation . . . 1165
10.120 check_matrix_sum.c File Reference . . . 1165
10.120.1 Detailed Description . . . 1166
10.120.2 Macro Definition Documentation . . . 1166
10.120.3 Function Documentation . . . 1166
10.121 check_matrixio.c File Reference . . . 1167
10.121.1 Detailed Description . . . 1168
10.121.2 Function Documentation . . . 1168
10.122 check_mgs.c File Reference . . . 1169
10.122.1 Detailed Description . . . 1169
10.122.2 Macro Definition Documentation . . . 1169
10.122.3 Function Documentation . . . 1169
10.123 check_mgs_inplace.c File Reference . . . 1170
10.123.1 Detailed Description . . . 1170
10.123.2 Macro Definition Documentation . . . 1171

10.123.3 Function Documentation . . . 1171
10.124 check_mgsadd.c File Reference . . . 1171
10.124.1 Detailed Description . . . 1172
10.124.2 Macro Definition Documentation . . . 1172
10.124.3 Function Documentation . . . 1173
10.125 check_mul.c File Reference . . . 1174
10.125.1 Detailed Description . . . 1174
10.125.2 Macro Definition Documentation . . . 1175
10.125.3 Function Documentation . . . 1175
10.125.4 Variable Documentation . . . 1176
10.126 check_mul_small.c File Reference . . . 1180
10.126.1 Detailed Description . . . 1181
10.126.2 Macro Definition Documentation . . . 1181
10.126.3 Function Documentation . . . 1181
10.126.4 Variable Documentation . . . 1183
10.127 check_mul_tall.c File Reference . . . 1185
10.127.1 Detailed Description . . . 1187
10.127.2 Macro Definition Documentation . . . 1187
10.127.3 Function Documentation . . . 1187
10.127.4 Variable Documentation . . . 1189
10.128 check_mulnormf2.c File Reference . . . 1191
10.128.1 Detailed Description . . . 1192
10.128.2 Function Documentation . . . 1192
10.129 check_mvp.c File Reference . . . 1192
10.129.1 Detailed Description . . . 1193
10.129.2 Function Documentation . . . 1193
10.130 check_mvpcall_matrix.c File Reference . . . 1194
10.130.1 Detailed Description . . . 1194
10.130.2 Function Documentation . . . 1194
10.131 check_norm.c File Reference . . . 1195

10.131.1 Detailed Description . . . 1196
10.131.2 Macro Definition Documentation . . . 1196
10.131.3 Function Documentation . . . 1196
10.131.4 Variable Documentation . . . 1197
10.132 check_one_value.c File Reference . . . 1198
10.132.1 Detailed Description . . . 1198
10.132.2 Macro Definition Documentation . . . 1198
10.132.3 Function Documentation . . . 1198
10.133 check_orth.c File Reference . . . 1199
10.133.1 Detailed Description . . . 1200
10.133.2 Macro Definition Documentation . . . 1200
10.133.3 Function Documentation . . . 1200
10.134 check_perm.c File Reference . . . 1201
10.134.1 Detailed Description . . . 1201
10.134.2 Macro Definition Documentation . . . 1202
10.134.3 Function Documentation . . . 1202
10.135 check_pinv.c File Reference . . . 1203
10.135.1 Detailed Description . . . 1204
10.135.2 Function Documentation . . . 1204
10.136 check_precond.c File Reference . . . 1208
10.136.1 Detailed Description . . . 1209
10.136.2 Function Documentation . . . 1209
10.137 check_proj_sym.c File Reference . . . 1211
10.137.1 Detailed Description . . . 1212
10.137.2 Macro Definition Documentation . . . 1212
10.137.3 Function Documentation . . . 1212
10.137.4 Variable Documentation . . . 1213
10.138 check_rank.c File Reference . . . 1213
10.138.1 Detailed Description . . . 1214
10.138.2 Function Documentation . . . 1214

10.139 check_real_imag_part.c File Reference . . . 1215
10.139.1 Detailed Description . . . 1215
10.139.2 Function Documentation . . . 1215
10.140 check_rowsub.c File Reference . . . 1215
10.140.1 Detailed Description . . . 1216
10.140.2 Macro Definition Documentation . . . 1216
10.140.3 Function Documentation . . . 1216
10.141 check_scalem.c File Reference . . . 1217
10.141.1 Detailed Description . . . 1217
10.141.2 Function Documentation . . . 1218
10.142 check_schur.c File Reference . . . 1218
10.142.1 Detailed Description . . . 1219
10.142.2 Function Documentation . . . 1219
10.143 check_set_get_col_row.c File Reference . . . 1220
10.143.1 Detailed Description . . . 1220
10.143.2 Function Documentation . . . 1220
10.144 check_setelement.c File Reference . . . 1222
10.144.1 Detailed Description . . . 1222
10.144.2 Function Documentation . . . 1222
10.145 check_sign.c File Reference . . . 1223
10.145.1 Detailed Description . . . 1224
10.145.2 Function Documentation . . . 1224
10.146 check_so1_handles.c File Reference . . . 1225
10.146.1 Detailed Description . . . 1226
10.146.2 Macro Definition Documentation . . . 1226
10.146.3 Function Documentation . . . 1226
10.147 check_so2_handles.c File Reference . . . 1227
10.147.1 Detailed Description . . . 1228
10.147.2 Macro Definition Documentation . . . 1228
10.147.3 Function Documentation . . . 1228

10.148 check_sub.c File Reference . . . 1229
10.148.1 Detailed Description . . . 1230
10.148.2 Macro Definition Documentation . . . 1230
10.148.3 Function Documentation . . . 1230
10.149 check_svd.c File Reference . . . 1231
10.149.1 Detailed Description . . . 1232
10.149.2 Function Documentation . . . 1232
10.150 check_svd_complex.c File Reference . . . 1233
10.150.1 Detailed Description . . . 1233
10.150.2 Function Documentation . . . 1233
10.151 check_svd_econ.c File Reference . . . 1234
10.151.1 Detailed Description . . . 1235
10.151.2 Function Documentation . . . 1235
10.152 check_svd_econ_complex.c File Reference . . . 1236
10.152.1 Detailed Description . . . 1237
10.152.2 Function Documentation . . . 1237
10.153 check_trace.c File Reference . . . 1238
10.153.1 Detailed Description . . . 1238
10.153.2 Macro Definition Documentation . . . 1238
10.153.3 Function Documentation . . . 1239
10.153.4 Variable Documentation . . . 1239
10.154 check_transpose.c File Reference . . . 1239
10.154.1 Detailed Description . . . 1240
10.154.2 Macro Definition Documentation . . . 1240
10.154.3 Function Documentation . . . 1240
10.154.4 Variable Documentation . . . 1241
10.155 check_tridiag.c File Reference . . . 1241
10.155.1 Detailed Description . . . 1242
10.155.2 Function Documentation . . . 1242
10.156 check_trisolve.c File Reference . . . 1242

10.156.1 Detailed Description . . . 1243
10.156.2 Macro Definition Documentation . . . 1243
10.156.3 Function Documentation . . . 1244
10.157 check_vector_perm.c File Reference . . . 1245
10.157.1 Detailed Description . . . 1246
10.157.2 Function Documentation . . . 1246
10.158 check_vectorio.c File Reference . . . 1249
10.158.1 Detailed Description . . . 1250
10.158.2 Function Documentation . . . 1250
10.159 cholesky.c File Reference . . . 1250
10.159.1 Detailed Description . . . 1251
10.159.2 Function Documentation . . . 1251
10.160 cholfactor.c File Reference . . . 1254
10.160.1 Detailed Description . . . 1254
10.161 cholmod.c File Reference . . . 1254
10.161.1 Detailed Description . . . 1254
10.162 cholmod_chol.c File Reference . . . 1254
10.162.1 Detailed Description . . . 1254
10.163 cimport.c File Reference . . . 1255
10.163.1 Detailed Description . . . 1255
10.164 CMakeCache.txt File Reference . . . 1255
10.164.1 Variable Documentation . . . 1256
10.165 CMakeCache.txt File Reference . . . 1256
10.165.1 Variable Documentation . . . 1256
10.166 CMakeCache.txt File Reference . . . 1256
10.167 CMakeCache.txt File Reference . . . 1256
10.167.1 Variable Documentation . . . 1257
10.168 CMakeCCompilerId.c File Reference . . . 1257
10.168.1 Macro Definition Documentation . . . 1257
10.168.2 Function Documentation . . . 1258

10.168.3 Variable Documentation . . . 1258
10.169 CMakeCXXCompilerId.cpp File Reference . . . 1258
10.169.1 Macro Definition Documentation . . . 1259
10.169.2 Function Documentation . . . 1260
10.169.3 Variable Documentation . . . 1260
10.170 CMakeFortranCompilerId.F File Reference . . . 1260
10.170.1 Macro Definition Documentation . . . 1261
10.170.2 Function/Subroutine Documentation . . . 1261
10.171 CMakeRuleHashes.txt File Reference . . . 1261
10.172 col_compress.c File Reference . . . 1261
10.172.1 Detailed Description . . . 1262
10.173 colops.c File Reference . . . 1262
10.173.1 Detailed Description . . . 1262
10.174 colrowsums.c File Reference . . . 1262
10.174.1 Detailed Description . . . 1263
10.175 condest.c File Reference . . . 1263
10.175.1 Detailed Description . . . 1263
10.175.2 Macro Definition Documentation . . . 1263
10.176 config.h File Reference . . . 1263
10.176.1 Macro Definition Documentation . . . 1265
10.177 convert.c File Reference . . . 1267
10.177.1 Detailed Description . . . 1268
10.177.2 Function Documentation . . . 1268
10.178 convert.c File Reference . . . 1274
10.178.1 Detailed Description . . . 1275
10.179 copy.c File Reference . . . 1275
10.179.1 Detailed Description . . . 1275
10.179.2 Variable Documentation . . . 1275
10.180 csc.c File Reference . . . 1275
10.180.1 Detailed Description . . . 1275

10.180.2 Function Documentation . . . 1275
10.181 csparse.c File Reference . . . 1278
10.181.1 Detailed Description . . . 1279
10.181.2 Macro Definition Documentation . . . 1279
10.181.3 Function Documentation . . . 1279
10.182 csparse_chol.c File Reference . . . 1296
10.182.1 Detailed Description . . . 1298
10.182.2 Macro Definition Documentation . . . 1298
10.182.3 Function Documentation . . . 1298
10.183 csr.c File Reference . . . 1310
10.183.1 Detailed Description . . . 1311
10.183.2 Function Documentation . . . 1311
10.184 cuthill.c File Reference . . . 1313
10.184.1 Macro Definition Documentation . . . 1314
10.184.2 Function Documentation . . . 1315
10.185 dae1.c File Reference . . . 1320
10.185.1 Detailed Description . . . 1320
10.186 dae1.c File Reference . . . 1320
10.186.1 Detailed Description . . . 1320
10.187 dae2.c File Reference . . . 1320
10.187.1 Detailed Description . . . 1320
10.188 dae2.c File Reference . . . 1320
10.188.1 Detailed Description . . . 1321
10.189 DartConfiguration.tcl File Reference . . . 1321
10.190 DCPX_T.c File Reference . . . 1321
10.190.1 Macro Definition Documentation . . . 1321
10.190.2 Function Documentation . . . 1321
10.190.3 Variable Documentation . . . 1321
10.191 decomp.c File Reference . . . 1322
10.191.1 Detailed Description . . . 1322

10.192 default_rom_chooser.c File Reference . . . 1322
10.192.1 Detailed Description . . . 1322
10.193 dense.c File Reference . . . 1322
10.193.1 Detailed Description . . . 1322
10.193.2 Function Documentation . . . 1322
10.194 dense_nm.c File Reference . . . 1325
10.194.1 Detailed Description . . . 1325
10.194.2 Macro Definition Documentation . . . 1325
10.195 dense_nm_gmpare.c File Reference . . . 1325
10.195.1 Detailed Description . . . 1326
10.196 dfs.c File Reference . . . 1326
10.196.1 Detailed Description . . . 1326
10.196.2 Macro Definition Documentation . . . 1326
10.197 dgeqp3.c File Reference . . . 1326
10.197.1 Detailed Description . . . 1327
10.198 diag.c File Reference . . . 1327
10.198.1 Detailed Description . . . 1327
10.199 diag.c File Reference . . . 1327
10.199.1 Detailed Description . . . 1327
10.199.2 Function Documentation . . . 1327
10.200 direct.c File Reference . . . 1330
10.200.1 Detailed Description . . . 1330
10.200.2 Variable Documentation . . . 1331
10.201 direct.h File Reference . . . 1331
10.201.1 Detailed Description . . . 1336
10.202 direct_chol_select.c File Reference . . . 1336
10.202.1 Detailed Description . . . 1336
10.203 direct_select.c File Reference . . . 1336
10.203.1 Detailed Description . . . 1337
10.204 doxy_main.c File Reference . . . 1337

10.205 dupl.c File Reference . . . 1337
10.205.1 Detailed Description . . . 1337
10.206 dynorm.c File Reference . . . 1337
10.206.1 Detailed Description . . . 1337
10.207 dynsys.h File Reference . . . 1338
10.207.1 Detailed Description . . . 1339
10.208 dynsysinit.c File Reference . . . 1339
10.208.1 Detailed Description . . . 1340
10.209 easyfrontend.h File Reference . . . 1340
10.209.1 Detailed Description . . . 1340
10.210 eig.c File Reference . . . 1340
10.210.1 Detailed Description . . . 1341
10.211 eig_hessenberg.c File Reference . . . 1341
10.211.1 Detailed Description . . . 1341
10.212 eigenvalue.h File Reference . . . 1341
10.212.1 Detailed Description . . . 1343
10.213 enum_conversions.c File Reference . . . 1343
10.213.1 Detailed Description . . . 1343
10.213.2 Macro Definition Documentation . . . 1344
10.214 eps.c File Reference . . . 1344
10.214.1 Function Documentation . . . 1344
10.215 eps.c File Reference . . . 1345
10.215.1 Detailed Description . . . 1345
10.216 equal.c File Reference . . . 1345
10.216.1 Detailed Description . . . 1345
10.216.2 Function Documentation . . . 1345
10.217 equation.c File Reference . . . 1347
10.217.1 Detailed Description . . . 1347
10.218 equation_apply.c File Reference . . . 1347
10.218.1 Detailed Description . . . 1348

10.219 equation_glyap.c File Reference . . . 1349
10.219.1 Detailed Description . . . 1349
10.219.2 Function Documentation . . . 1350
10.220 equation_glyap_dae1.c File Reference . . . 1356
10.220.1 Detailed Description . . . 1357
10.220.2 Function Documentation . . . 1357
10.221 equation_glyap_dae2.c File Reference . . . 1365
10.221.1 Detailed Description . . . 1367
10.221.2 Function Documentation . . . 1367
10.222 equation_glyap_so1.c File Reference . . . 1382
10.222.1 Detailed Description . . . 1382
10.222.2 Function Documentation . . . 1382
10.223 equation_glyap_so2.c File Reference . . . 1392
10.223.1 Detailed Description . . . 1393
10.223.2 Function Documentation . . . 1393
10.224 equation_lyap.c File Reference . . . 1403
10.224.1 Detailed Description . . . 1404
10.224.2 Function Documentation . . . 1404
10.225 equation_stable.c File Reference . . . 1413
10.225.1 Detailed Description . . . 1414
10.225.2 Macro Definition Documentation . . . 1414
10.225.3 Function Documentation . . . 1414
10.226 equations.c File Reference . . . 1417
10.226.1 Detailed Description . . . 1418
10.226.2 Macro Definition Documentation . . . 1419
10.226.3 Typedef Documentation . . . 1420
10.226.4 Function Documentation . . . 1420
10.226.5 Variable Documentation . . . 1422
10.227 error.c File Reference . . . 1423
10.227.1 Detailed Description . . . 1423

10.228 error_macro.h File Reference . . . 1423
10.228.1 Detailed Description . . . 1423
10.228.2 Macro Definition Documentation . . . 1424
10.229 errors.h File Reference . . . 1424
10.229.1 Detailed Description . . . 1425
10.230 eye.c File Reference . . . 1425
10.230.1 Detailed Description . . . 1425
10.230.2 Macro Definition Documentation . . . 1425
10.231 fdm_matrix.c File Reference . . . 1425
10.231.1 Detailed Description . . . 1426
10.231.2 Function Documentation . . . 1426
10.232 feature_tests.c File Reference . . . 1426
10.232.1 Function Documentation . . . 1426
10.232.2 Variable Documentation . . . 1426
10.233 feature_tests.cxx File Reference . . . 1426
10.233.1 Function Documentation . . . 1426
10.233.2 Variable Documentation . . . 1426
10.234 fill.c File Reference . . . 1426
10.234.1 Detailed Description . . . 1427
10.235 filter.c File Reference . . . 1427
10.235.1 Detailed Description . . . 1427
10.235.2 Function Documentation . . . 1427
10.236 filter2D.C File Reference . . . 1427
10.237 fortran_translate.h File Reference . . . 1427
10.237.1 Macro Definition Documentation . . . 1427
10.238 freelist.c File Reference . . . 1427
10.238.1 Detailed Description . . . 1428
10.238.2 Macro Definition Documentation . . . 1428
10.239 freelist.h File Reference . . . 1428
10.239.1 Detailed Description . . . 1429

10.240 galerkin.c File Reference . . . 1429
10.240.1 Detailed Description . . . 1429
10.240.2 Macro Definition Documentation . . . 1430
10.241 gaxpy.c File Reference . . . 1430
10.241.1 Detailed Description . . . 1430
10.242 getcol.c File Reference . . . 1430
10.242.1 Detailed Description . . . 1430
10.243 getelement.c File Reference . . . 1430
10.243.1 Detailed Description . . . 1430
10.244 getrow.c File Reference . . . 1431
10.244.1 Detailed Description . . . 1431
10.245 getset.c File Reference . . . 1431
10.245.1 Detailed Description . . . 1431
10.246 gettypes.c File Reference . . . 1431
10.246.1 Detailed Description . . . 1432
10.246.2 Macro Definition Documentation . . . 1432
10.246.3 Function Documentation . . . 1433
10.247 gglrsrm.c File Reference . . . 1433
10.247.1 Detailed Description . . . 1433
10.248 glyap.c File Reference . . . 1433
10.248.1 Detailed Description . . . 1434
10.249 glyap3.c File Reference . . . 1434
10.249.1 Detailed Description . . . 1435
10.249.2 Macro Definition Documentation . . . 1436
10.249.3 Typedef Documentation . . . 1436
10.249.4 Enumeration Type Documentation . . . 1436
10.249.5 Function Documentation . . . 1436
10.250 glyap_test.c File Reference . . . 1447
10.250.1 Detailed Description . . . 1447
10.250.2 Function Documentation . . . 1447

10.251 gmres.c File Reference . . . 1448
10.251.1 Detailed Description . . . 1449
10.252 gslrsrm.c File Reference . . . 1449
10.252.1 Detailed Description . . . 1449
10.253 gstein.c File Reference . . . 1449
10.253.1 Detailed Description . . . 1449
10.254 gstein_test.c File Reference . . . 1449
10.254.1 Detailed Description . . . 1450
10.254.2 Function Documentation . . . 1450
10.255 h2.h File Reference . . . 1451
10.255.1 Detailed Description . . . 1453
10.256 h2_norm.c File Reference . . . 1453
10.256.1 Detailed Description . . . 1453
10.256.2 Macro Definition Documentation . . . 1453
10.257 h2error.c File Reference . . . 1453
10.257.1 Detailed Description . . . 1454
10.257.2 Macro Definition Documentation . . . 1454
10.258 hessenberg_lu.c File Reference . . . 1454
10.258.1 Detailed Description . . . 1455
10.258.2 Typedef Documentation . . . 1455
10.258.3 Function Documentation . . . 1455
10.259 hlu0.c File Reference . . . 1459
10.259.1 Detailed Description . . . 1459
10.259.2 Typedef Documentation . . . 1460
10.259.3 Function Documentation . . . 1460
10.260 hluk.c File Reference . . . 1461
10.260.1 Detailed Description . . . 1461
10.260.2 Macro Definition Documentation . . . 1462
10.260.3 Typedef Documentation . . . 1462
10.260.4 Function Documentation . . . 1462

10.261 init.c File Reference . . . 1463
10.261.1 Detailed Description . . . 1464
10.262 init.c File Reference . . . 1464
10.262.1 Detailed Description . . . 1464
10.262.2 Function Documentation . . . 1464
10.262.3 Variable Documentation . . . 1464
10.263 init.c File Reference . . . 1464
10.263.1 Detailed Description . . . 1465
10.264 INSTALL.md File Reference . . . 1465
10.265 interface_cholmod.h File Reference . . . 1465
10.265.1 Detailed Description . . . 1465
10.266 interface_csparse.h File Reference . . . 1465
10.267 interface_matlab.c File Reference . . . 1466
10.267.1 Detailed Description . . . 1466
10.267.2 Function Documentation . . . 1466
10.267.3 Variable Documentation . . . 1467
10.268 interface_matlab.h File Reference . . . 1467
10.268.1 Detailed Description . . . 1469
10.268.2 Macro Definition Documentation . . . 1469
10.269 inverse.c File Reference . . . 1469
10.269.1 Detailed Description . . . 1469
10.269.2 Function Documentation . . . 1470
10.270 irka_biorth.c File Reference . . . 1474
10.270.1 Detailed Description . . . 1474
10.270.2 Macro Definition Documentation . . . 1474
10.271 irka_common.c File Reference . . . 1474
10.271.1 Detailed Description . . . 1475
10.272 irka_para_init.c File Reference . . . 1475
10.272.1 Detailed Description . . . 1476
10.272.2 Function Documentation . . . 1476

10.273 jacobi.c File Reference . . . 1476
10.273.1 Detailed Description . . . 1476
10.274 kron.c File Reference . . . 1476
10.274.1 Detailed Description . . . 1476
10.275 lanczos_template_nrm.c File Reference . . . 1477
10.275.1 Detailed Description . . . 1477
10.276 lapack.c File Reference . . . 1477
10.276.1 Detailed Description . . . 1478
10.276.2 Function Documentation . . . 1478
10.277 lapack_qr.c File Reference . . . 1484
10.277.1 Detailed Description . . . 1485
10.277.2 Function Documentation . . . 1485
10.278 lft.c File Reference . . . 1502
10.278.1 Detailed Description . . . 1503
10.279 link.txt File Reference . . . 1503
10.280 link.txt File Reference . . . 1503
10.281 link.txt File Reference . . . 1503
10.282 link.txt File Reference . . . 1503
10.283 link.txt File Reference . . . 1503
10.284 link.txt File Reference . . . 1503
10.285 link.txt File Reference . . . 1503
10.286 link.txt File Reference . . . 1503
10.287 link.txt File Reference . . . 1503
10.287.1 Variable Documentation . . . 1503
10.288 link.txt File Reference . . . 1503
10.288.1 Variable Documentation . . . 1504
10.289 link.txt File Reference . . . 1504
10.289.1 Variable Documentation . . . 1504
10.290 link.txt File Reference . . . 1504
10.290.1 Variable Documentation . . . 1505

10.291 link.txt File Reference . . . 1505
10.291.1 Variable Documentation . . . 1505
10.292 link.txt File Reference . . . 1505
10.292.1 Variable Documentation . . . 1506
10.293 link.txt File Reference . . . 1506
10.293.1 Variable Documentation . . . 1506
10.294 link.txt File Reference . . . 1506
10.294.1 Variable Documentation . . . 1507
10.295 link.txt File Reference . . . 1507
10.295.1 Variable Documentation . . . 1507
10.296 link.txt File Reference . . . 1507
10.296.1 Variable Documentation . . . 1508
10.297 link.txt File Reference . . . 1508
10.297.1 Variable Documentation . . . 1508
10.298 link.txt File Reference . . . 1508
10.298.1 Variable Documentation . . . 1508
10.299 link.txt File Reference . . . 1508
10.299.1 Variable Documentation . . . 1509
10.300 link.txt File Reference . . . 1509
10.300.1 Variable Documentation . . . 1509
10.301 link.txt File Reference . . . 1509
10.301.1 Variable Documentation . . . 1509
10.302 link.txt File Reference . . . 1509
10.302.1 Variable Documentation . . . 1510
10.303 link.txt File Reference . . . 1510
10.303.1 Variable Documentation . . . 1510
10.304 link.txt File Reference . . . 1510
10.304.1 Variable Documentation . . . 1511
10.305 link.txt File Reference . . . 1511
10.305.1 Variable Documentation . . . 1511

10.306 nk.txt File Reference 1511
10.306.1 Variable Documentation 1512
10.307 nk.txt File Reference 1512
10.307.1 Variable Documentation 1512
10.308 nk.txt File Reference 1512
10.308.1 Variable Documentation 1513
10.309 nk.txt File Reference 1513
10.309.1 Variable Documentation 1513
10.310 nk.txt File Reference 1513
10.310.1 Variable Documentation 1513
10.311 nk.txt File Reference 1513
10.311.1 Variable Documentation 1514
10.312 nk.txt File Reference 1514
10.312.1 Variable Documentation 1514
10.313 nk.txt File Reference 1514
10.313.1 Variable Documentation 1515
10.314 nk.txt File Reference 1515
10.314.1 Variable Documentation 1515
10.315 nk.txt File Reference 1515
10.315.1 Variable Documentation 1516
10.316 nk.txt File Reference 1516
10.316.1 Variable Documentation 1516
10.317 nk.txt File Reference 1516
10.317.1 Variable Documentation 1516
10.318 nk.txt File Reference 1516
10.318.1 Variable Documentation 1517
10.319 nk.txt File Reference 1517
10.319.1 Variable Documentation 1517
10.320 nk.txt File Reference 1517
10.320.1 Variable Documentation 1518

10.321 link.txt File Reference . . . 1518
10.321.1 Variable Documentation . . . 1518
10.322 link.txt File Reference . . . 1518
10.322.1 Variable Documentation . . . 1519
10.323 link.txt File Reference . . . 1519
10.323.1 Variable Documentation . . . 1519
10.324 link.txt File Reference . . . 1519
10.324.1 Variable Documentation . . . 1520
10.325 link.txt File Reference . . . 1520
10.325.1 Variable Documentation . . . 1520
10.326 link.txt File Reference . . . 1520
10.326.1 Variable Documentation . . . 1521
10.327 link.txt File Reference . . . 1521
10.327.1 Variable Documentation . . . 1521
10.328 link.txt File Reference . . . 1521
10.328.1 Variable Documentation . . . 1522
10.329 link.txt File Reference . . . 1522
10.329.1 Variable Documentation . . . 1522
10.330 link.txt File Reference . . . 1522
10.330.1 Variable Documentation . . . 1523
10.331 link.txt File Reference . . . 1523
10.331.1 Variable Documentation . . . 1523
10.332 link.txt File Reference . . . 1523
10.332.1 Variable Documentation . . . 1524
10.333 link.txt File Reference . . . 1524
10.333.1 Variable Documentation . . . 1524
10.334 link.txt File Reference . . . 1524
10.334.1 Variable Documentation . . . 1525
10.335 link.txt File Reference . . . 1525
10.335.1 Variable Documentation . . . 1525

10.336 link.txt File Reference . . . 1525
10.336.1 Variable Documentation . . . 1526
10.337 link.txt File Reference . . . 1526
10.337.1 Variable Documentation . . . 1526
10.338 link.txt File Reference . . . 1526
10.338.1 Variable Documentation . . . 1527
10.339 link.txt File Reference . . . 1527
10.339.1 Variable Documentation . . . 1527
10.340 link.txt File Reference . . . 1527
10.340.1 Variable Documentation . . . 1528
10.341 link.txt File Reference . . . 1528
10.341.1 Variable Documentation . . . 1528
10.342 link.txt File Reference . . . 1528
10.342.1 Variable Documentation . . . 1529
10.343 link.txt File Reference . . . 1529
10.343.1 Variable Documentation . . . 1529
10.344 link.txt File Reference . . . 1529
10.344.1 Variable Documentation . . . 1530
10.345 link.txt File Reference . . . 1530
10.345.1 Variable Documentation . . . 1530
10.346 link.txt File Reference . . . 1530
10.346.1 Variable Documentation . . . 1531
10.347 link.txt File Reference . . . 1531
10.347.1 Variable Documentation . . . 1531
10.348 link.txt File Reference . . . 1531
10.348.1 Variable Documentation . . . 1532
10.349 link.txt File Reference . . . 1532
10.349.1 Variable Documentation . . . 1532
10.350 link.txt File Reference . . . 1532
10.350.1 Variable Documentation . . . 1533

10.351 link.txt File Reference . . . 1533
10.351.1 Variable Documentation . . . 1533
10.352 link.txt File Reference . . . 1533
10.352.1 Variable Documentation . . . 1534
10.353 link.txt File Reference . . . 1534
10.353.1 Variable Documentation . . . 1534
10.354 link.txt File Reference . . . 1534
10.354.1 Variable Documentation . . . 1535
10.355 link.txt File Reference . . . 1535
10.355.1 Variable Documentation . . . 1535
10.356 link.txt File Reference . . . 1535
10.356.1 Variable Documentation . . . 1536
10.357 link.txt File Reference . . . 1536
10.357.1 Variable Documentation . . . 1536
10.358 link.txt File Reference . . . 1536
10.358.1 Variable Documentation . . . 1537
10.359 link.txt File Reference . . . 1537
10.359.1 Variable Documentation . . . 1537
10.360 link.txt File Reference . . . 1537
10.360.1 Variable Documentation . . . 1538
10.361 link.txt File Reference . . . 1538
10.361.1 Variable Documentation . . . 1538
10.362 link.txt File Reference . . . 1538
10.362.1 Variable Documentation . . . 1539
10.363 link.txt File Reference . . . 1539
10.363.1 Variable Documentation . . . 1539
10.364 link.txt File Reference . . . 1539
10.364.1 Variable Documentation . . . 1540
10.365 link.txt File Reference . . . 1540
10.365.1 Variable Documentation . . . 1540

10.366 nk.txt File Reference 1540
10.366.1 Variable Documentation 1541
10.367 nk.txt File Reference 1541
10.367.1 Variable Documentation 1541
10.368 nk.txt File Reference 1541
10.368.1 Variable Documentation 1542
10.369 nk.txt File Reference 1542
10.369.1 Variable Documentation 1542
10.370 nk.txt File Reference 1542
10.370.1 Variable Documentation 1543
10.371 nk.txt File Reference 1543
10.371.1 Variable Documentation 1543
10.372 nk.txt File Reference 1543
10.372.1 Variable Documentation 1544
10.373 nk.txt File Reference 1544
10.373.1 Variable Documentation 1544
10.374 nk.txt File Reference 1544
10.374.1 Variable Documentation 1545
10.375 nk.txt File Reference 1545
10.375.1 Variable Documentation 1545
10.376 nk.txt File Reference 1545
10.376.1 Variable Documentation 1546
10.377 nk.txt File Reference 1546
10.377.1 Variable Documentation 1546
10.378 nk.txt File Reference 1546
10.378.1 Variable Documentation 1547
10.379 nk.txt File Reference 1547
10.379.1 Variable Documentation 1547
10.380 nk.txt File Reference 1547
10.380.1 Variable Documentation 1548

10.381 link.txt File Reference . . . 1548
10.381.1 Variable Documentation . . . 1548
10.382 link.txt File Reference . . . 1548
10.382.1 Variable Documentation . . . 1549
10.383 link.txt File Reference . . . 1549
10.383.1 Variable Documentation . . . 1549
10.384 link.txt File Reference . . . 1549
10.384.1 Variable Documentation . . . 1550
10.385 link.txt File Reference . . . 1550
10.385.1 Variable Documentation . . . 1550
10.386 link.txt File Reference . . . 1550
10.386.1 Variable Documentation . . . 1551
10.387 link.txt File Reference . . . 1551
10.387.1 Variable Documentation . . . 1551
10.388 link.txt File Reference . . . 1551
10.388.1 Variable Documentation . . . 1552
10.389 link.txt File Reference . . . 1552
10.389.1 Variable Documentation . . . 1552
10.390 link.txt File Reference . . . 1552
10.390.1 Variable Documentation . . . 1553
10.391 link.txt File Reference . . . 1553
10.391.1 Variable Documentation . . . 1553
10.392 link.txt File Reference . . . 1553
10.392.1 Variable Documentation . . . 1554
10.393 link.txt File Reference . . . 1554
10.393.1 Variable Documentation . . . 1554
10.394 link.txt File Reference . . . 1554
10.394.1 Variable Documentation . . . 1555
10.395 link.txt File Reference . . . 1555
10.395.1 Variable Documentation . . . 1555

10.396 link.txt File Reference . . . 1555
10.396.1 Variable Documentation . . . 1555
10.397 link.txt File Reference . . . 1555
10.397.1 Variable Documentation . . . 1556
10.398 link.txt File Reference . . . 1556
10.398.1 Variable Documentation . . . 1556
10.399 link.txt File Reference . . . 1556
10.399.1 Variable Documentation . . . 1556
10.400 link.txt File Reference . . . 1556
10.400.1 Variable Documentation . . . 1557
10.401 link.txt File Reference . . . 1557
10.401.1 Variable Documentation . . . 1557
10.402 link.txt File Reference . . . 1557
10.402.1 Variable Documentation . . . 1558
10.403 link.txt File Reference . . . 1558
10.403.1 Variable Documentation . . . 1558
10.404 link.txt File Reference . . . 1558
10.404.1 Variable Documentation . . . 1559
10.405 link.txt File Reference . . . 1559
10.405.1 Variable Documentation . . . 1559
10.406 link.txt File Reference . . . 1559
10.406.1 Variable Documentation . . . 1559
10.407 link.txt File Reference . . . 1559
10.407.1 Variable Documentation . . . 1560
10.408 link.txt File Reference . . . 1560
10.408.1 Variable Documentation . . . 1560
10.409 link.txt File Reference . . . 1560
10.409.1 Variable Documentation . . . 1561
10.410 link.txt File Reference . . . 1561
10.410.1 Variable Documentation . . . 1561

10.411 link.txt File Reference 1561
10.411.1 Variable Documentation 1562
10.412 link.txt File Reference 1562
10.412.1 Variable Documentation 1562
10.413 link.txt File Reference 1562
10.413.1 Variable Documentation 1563
10.414 link.txt File Reference 1563
10.414.1 Variable Documentation 1563
10.415 link.txt File Reference 1563
10.415.1 Variable Documentation 1564
10.416 link.txt File Reference 1564
10.416.1 Variable Documentation 1564
10.417 link.txt File Reference 1564
10.417.1 Variable Documentation 1565
10.418 link.txt File Reference 1565
10.418.1 Variable Documentation 1565
10.419 link.txt File Reference 1565
10.419.1 Variable Documentation 1565
10.420 link.txt File Reference 1565
10.420.1 Variable Documentation 1566
10.421 link.txt File Reference 1566
10.421.1 Variable Documentation 1566
10.422 link.txt File Reference 1566
10.422.1 Variable Documentation 1567
10.423 link.txt File Reference 1567
10.423.1 Variable Documentation 1567
10.424 link.txt File Reference 1567
10.424.1 Variable Documentation 1568
10.425 link.txt File Reference 1568
10.425.1 Variable Documentation 1568

10.426 nk.txt File Reference . . . 1568
10.426.1 Variable Documentation . . . 1569
10.427 nk.txt File Reference . . . 1569
10.427.1 Variable Documentation . . . 1569
10.428 nk.txt File Reference . . . 1569
10.428.1 Variable Documentation . . . 1570
10.429 nk.txt File Reference . . . 1570
10.429.1 Variable Documentation . . . 1570
10.430 nk.txt File Reference . . . 1570
10.430.1 Variable Documentation . . . 1570
10.431 nk.txt File Reference . . . 1570
10.431.1 Variable Documentation . . . 1571
10.432 nk.txt File Reference . . . 1571
10.432.1 Variable Documentation . . . 1571
10.433 nk.txt File Reference . . . 1571
10.433.1 Variable Documentation . . . 1571
10.434 nk.txt File Reference . . . 1571
10.434.1 Variable Documentation . . . 1572
10.435 nk.txt File Reference . . . 1572
10.435.1 Variable Documentation . . . 1572
10.436 nk.txt File Reference . . . 1572
10.436.1 Variable Documentation . . . 1573
10.437 nk.txt File Reference . . . 1573
10.437.1 Variable Documentation . . . 1573
10.438 nk.txt File Reference . . . 1573
10.438.1 Variable Documentation . . . 1574
10.439 nk.txt File Reference . . . 1574
10.439.1 Variable Documentation . . . 1574
10.440 nsolve.c File Reference . . . 1574
10.440.1 Detailed Description . . . 1574

10.440.2 Function Documentation . . . 1575
10.441 linsolve_multi.c File Reference . . . 1580
10.441.1 Detailed Description . . . 1581
10.441.2 Function Documentation . . . 1581
10.442 lnsolvem.c File Reference . . . 1586
10.442.1 Detailed Description . . . 1587
10.442.2 Function Documentation . . . 1587
10.443 lnsolvem_multi.c File Reference . . . 1592
10.443.1 Detailed Description . . . 1593
10.443.2 Function Documentation . . . 1593
10.444 lnspace.c File Reference . . . 1596
10.444.1 Detailed Description . . . 1597
10.445 lradi.c File Reference . . . 1597
10.445.1 Detailed Description . . . 1597
10.445.2 Macro Definition Documentation . . . 1597
10.446 lradi.c File Reference . . . 1597
10.446.1 Detailed Description . . . 1598
10.447 lrcf_adi.h File Reference . . . 1598
10.447.1 Detailed Description . . . 1604
10.448 lnm.c File Reference . . . 1604
10.448.1 Detailed Description . . . 1604
10.448.2 Macro Definition Documentation . . . 1604
10.448.3 Function Documentation . . . 1604
10.449 lnm.c File Reference . . . 1605
10.449.1 Detailed Description . . . 1605
10.450 lnm_linesearch.c File Reference . . . 1606
10.450.1 Detailed Description . . . 1606
10.450.2 Function Documentation . . . 1606
10.451 lrsrm.c File Reference . . . 1606
10.451.1 Detailed Description . . . 1606

10.452 srm_somor.c File Reference . . . 1607
10.452.1 Detailed Description . . . 1607
10.453 srm_somor_so.c File Reference . . . 1607
10.453.1 Detailed Description . . . 1607
10.454 u.c File Reference . . . 1607
10.454.1 Detailed Description . . . 1608
10.454.2 Macro Definition Documentation . . . 1608
10.454.3 Function Documentation . . . 1608
10.455 vap.c File Reference . . . 1608
10.455.1 Detailed Description . . . 1608
10.456 vap.c File Reference . . . 1608
10.456.1 Detailed Description . . . 1608
10.457 vap_res2.c File Reference . . . 1608
10.457.1 Detailed Description . . . 1609
10.458 vap_sgn_fac.c File Reference . . . 1609
10.458.1 Detailed Description . . . 1609
10.459 malloc.c File Reference . . . 1609
10.459.1 Detailed Description . . . 1610
10.459.2 Typedef Documentation . . . 1610
10.459.3 Variable Documentation . . . 1610
10.460 map.c File Reference . . . 1610
10.460.1 Detailed Description . . . 1612
10.460.2 Macro Definition Documentation . . . 1612
10.460.3 Function Documentation . . . 1613
10.461 map.c File Reference . . . 1614
10.461.1 Detailed Description . . . 1616
10.461.2 Macro Definition Documentation . . . 1616
10.461.3 Function Documentation . . . 1617
10.462 matrix.c File Reference . . . 1619
10.462.1 Detailed Description . . . 1620

10.462.2 Macro Definition Documentation . . . 1620
10.463 matrix.h File Reference . . . 1620
10.463.1 Detailed Description . . . 1629
10.464 matrix_print.c File Reference . . . 1629
10.464.1 Detailed Description . . . 1629
10.465 memsize.c File Reference . . . 1630
10.465.1 Detailed Description . . . 1630
10.466 mess.h File Reference . . . 1630
10.466.1 Detailed Description . . . 1630
10.467 mess_call.c File Reference . . . 1631
10.467.1 Detailed Description . . . 1631
10.467.2 Typedef Documentation . . . 1631
10.467.3 Function Documentation . . . 1631
10.467.4 Variable Documentation . . . 1632
10.468 mess_classname_from_mexmess.c File Reference . . . 1632
10.468.1 Detailed Description . . . 1632
10.469 MESS_direct.cc File Reference . . . 1632
10.469.1 Function Documentation . . . 1633
10.470 MESS_direct.h File Reference . . . 1633
10.471 MESS_direct_install.cc File Reference . . . 1633
10.472 MESS_enum.cc File Reference . . . 1633
10.472.1 Macro Definition Documentation . . . 1634
10.472.2 Function Documentation . . . 1635
10.473 mess_enum_converter.c File Reference . . . 1636
10.473.1 Detailed Description . . . 1637
10.473.2 Macro Definition Documentation . . . 1637
10.474 MESS_enums.cc File Reference . . . 1637
10.474.1 Macro Definition Documentation . . . 1639
10.474.2 Function Documentation . . . 1639
10.475 MESS_enums.h File Reference . . . 1641

10.475.1 Macro Definition Documentation . . . 1642
10.475.2 Function Documentation . . . 1642
10.476 MESS_equation.cc File Reference . . . 1643
10.476.1 Function Documentation . . . 1643
10.477 MESS_equation.h File Reference . . . 1643
10.478 mess_equation_from_mexmess.c File Reference . . . 1643
10.478.1 Detailed Description . . . 1644
10.478.2 Typedef Documentation . . . 1644
10.478.3 Variable Documentation . . . 1644
10.479 MESS_equation_install.cc File Reference . . . 1645
10.480 MESS_error.h File Reference . . . 1645
10.480.1 Macro Definition Documentation . . . 1645
10.481 MESS_HAVE_PTRDIFF_T.c File Reference . . . 1645
10.481.1 Macro Definition Documentation . . . 1645
10.481.2 Function Documentation . . . 1645
10.481.3 Variable Documentation . . . 1645
10.482 MESS_HAVE_SIZE_T.c File Reference . . . 1646
10.482.1 Macro Definition Documentation . . . 1646
10.482.2 Function Documentation . . . 1646
10.482.3 Variable Documentation . . . 1646
10.483 mess_matrix.c File Reference . . . 1646
10.483.1 Detailed Description . . . 1647
10.484 MESS_matrix.cc File Reference . . . 1647
10.484.1 Macro Definition Documentation . . . 1647
10.484.2 Function Documentation . . . 1647
10.485 MESS_matrix.cc File Reference . . . 1647
10.485.1 Macro Definition Documentation . . . 1648
10.485.2 Function Documentation . . . 1648
10.486 MESS_matrix.h File Reference . . . 1648
10.486.1 Macro Definition Documentation . . . 1651

10.486.2 Function Documentation . . . 1652
10.487 MESS_matrix_install.cc File Reference . . . 1662
10.487.1 Macro Definition Documentation . . . 1665
10.487.2 Function Documentation . . . 1665
10.488 MESS_matrix_ops_mm_generator.h File Reference . . . 1672
10.488.1 Macro Definition Documentation . . . 1673
10.489 MESS_matrix_ops_mm_mm.cc File Reference . . . 1673
10.489.1 Function Documentation . . . 1673
10.490 MESS_matrix_ops_mm_ocm.cc File Reference . . . 1679
10.490.1 Function Documentation . . . 1679
10.491 MESS_matrix_ops_mm_om.cc File Reference . . . 1679
10.491.1 Function Documentation . . . 1679
10.492 MESS_matrix_ops_mm_os.cc File Reference . . . 1679
10.492.1 Function Documentation . . . 1680
10.493 MESS_matrix_ops_mm_osc.cc File Reference . . . 1680
10.493.1 Function Documentation . . . 1681
10.494 MESS_matrix_ops_mm_oscm.cc File Reference . . . 1681
10.494.1 Function Documentation . . . 1681
10.495 MESS_matrix_ops_mm_osm.cc File Reference . . . 1681
10.495.1 Function Documentation . . . 1682
10.496 MESS_misc.cc File Reference . . . 1682
10.496.1 Function Documentation . . . 1682
10.497 MESS_multidirect.cc File Reference . . . 1684
10.497.1 Function Documentation . . . 1684
10.498 MESS_multidirect.h File Reference . . . 1684
10.499 MESS_multidirect_install.cc File Reference . . . 1684
10.500 mess_options.c File Reference . . . 1684
10.500.1 Detailed Description . . . 1685
10.501 MESS_options.cc File Reference . . . 1685
10.501.1 Function Documentation . . . 1685

10.501.2 Variable Documentation . . . 1685
10.502 MESS_options.cc File Reference . . . 1685
10.502.1 Function Documentation . . . 1686
10.503 MESS_options.h File Reference . . . 1686
10.504 MESS_options_install.cc File Reference . . . 1686
10.505 mess_set_errorlevel.c File Reference . . . 1686
10.505.1 Detailed Description . . . 1687
10.506 mess_status.c File Reference . . . 1687
10.506.1 Detailed Description . . . 1687
10.506.2 Function Documentation . . . 1687
10.507 MESS_status.cc File Reference . . . 1687
10.507.1 Function Documentation . . . 1687
10.508 MESS_status.cc File Reference . . . 1687
10.508.1 Function Documentation . . . 1688
10.509 MESS_status.h File Reference . . . 1688
10.510 MESS_status_install.cc File Reference . . . 1688
10.511 MESS_to_oct.cc File Reference . . . 1688
10.511.1 Function Documentation . . . 1689
10.512 mess_vector.c File Reference . . . 1689
10.512.1 Detailed Description . . . 1689
10.513 MESS_vector.cc File Reference . . . 1689
10.513.1 Macro Definition Documentation . . . 1689
10.513.2 Function Documentation . . . 1689
10.514 MESS_vector.cc File Reference . . . 1689
10.514.1 Macro Definition Documentation . . . 1690
10.514.2 Function Documentation . . . 1690
10.515 MESS_vector.h File Reference . . . 1690
10.515.1 Macro Definition Documentation . . . 1692
10.515.2 Function Documentation . . . 1692
10.516 MESS_vector_install.cc File Reference . . . 1700

10.516.1 Macro Definition Documentation . . . 1701
10.516.2 Function Documentation . . . 1702
10.517 MESS_vector_ops_mm_mv.cc File Reference . . . 1707
10.517.1 Function Documentation . . . 1707
10.518 MESS_vector_ops_mv_generator.h File Reference . . . 1710
10.518.1 Macro Definition Documentation . . . 1710
10.519 MESS_vector_ops_mv_mv.cc File Reference . . . 1711
10.519.1 Function Documentation . . . 1711
10.520 MESS_vector_ops_mv_os.cc File Reference . . . 1714
10.520.1 Function Documentation . . . 1714
10.521 MESS_vector_ops_mv_osc.cc File Reference . . . 1715
10.521.1 Function Documentation . . . 1715
10.522 MESS_vector_ops_ocm_mv.cc File Reference . . . 1716
10.522.1 Function Documentation . . . 1716
10.523 MESS_vector_ops_om_mv.cc File Reference . . . 1716
10.523.1 Function Documentation . . . 1717
10.524 MESS_vector_ops_oscm_mv.cc File Reference . . . 1717
10.524.1 Function Documentation . . . 1717
10.525 MESS_vector_ops_osm_mv.cc File Reference . . . 1717
10.525.1 Function Documentation . . . 1717
10.526 mess_version.c File Reference . . . 1717
10.526.1 Detailed Description . . . 1718
10.526.2 Macro Definition Documentation . . . 1718
10.527 mex_mess_care.c File Reference . . . 1718
10.527.1 Detailed Description . . . 1718
10.528 mex_mess_dense_nm_gmpare.c File Reference . . . 1718
10.528.1 Detailed Description . . . 1719
10.529 mex_mess_glyap.c File Reference . . . 1719
10.529.1 Detailed Description . . . 1719
10.530 mex_mess_gstein.c File Reference . . . 1719

10.530.1 Detailed Description . . . 1719
10.531 mex_mess_lradi.c File Reference . . . 1719
10.531.1 Detailed Description . . . 1719
10.532 mex_mess_lrnm.c File Reference . . . 1719
10.532.1 Detailed Description . . . 1720
10.533 mex_mess_lyap.c File Reference . . . 1720
10.533.1 Detailed Description . . . 1720
10.534 mex_mess_sylvester_dense.c File Reference . . . 1720
10.535 mex_mess_sylvester_sparsedense.c File Reference . . . 1720
10.535.1 Detailed Description . . . 1720
10.536 mex_test_matrix.c File Reference . . . 1720
10.536.1 Detailed Description . . . 1720
10.537 mex_test_options.c File Reference . . . 1721
10.537.1 Detailed Description . . . 1721
10.538 mex_test_status.c File Reference . . . 1721
10.538.1 Detailed Description . . . 1721
10.539 mex_test_vector.c File Reference . . . 1721
10.539.1 Detailed Description . . . 1721
10.540 mgs.c File Reference . . . 1721
10.540.1 Detailed Description . . . 1722
10.541 mgsadd.c File Reference . . . 1722
10.541.1 Detailed Description . . . 1722
10.542 minmax.c File Reference . . . 1722
10.542.1 Detailed Description . . . 1722
10.543 misc.c File Reference . . . 1722
10.543.1 Function Documentation . . . 1723
10.544 misc.c File Reference . . . 1723
10.544.1 Detailed Description . . . 1724
10.544.2 Macro Definition Documentation . . . 1724
10.545 misc.h File Reference . . . 1725

10.545.1 Detailed Description . . . 1730
10.546 mklpardiso.c File Reference . . . 1731
10.546.1 Detailed Description . . . 1731
10.546.2 Typedef Documentation . . . 1731
10.546.3 Function Documentation . . . 1731
10.547 mult.c File Reference . . . 1740
10.547.1 Detailed Description . . . 1740
10.547.2 Function Documentation . . . 1740
10.548 multidirect.c File Reference . . . 1741
10.548.1 Detailed Description . . . 1742
10.548.2 Macro Definition Documentation . . . 1742
10.548.3 Variable Documentation . . . 1742
10.549 multidirect_select.c File Reference . . . 1742
10.549.1 Detailed Description . . . 1742
10.550 multilu_adavanced.c File Reference . . . 1743
10.550.1 Detailed Description . . . 1744
10.550.2 Macro Definition Documentation . . . 1744
10.550.3 Typedef Documentation . . . 1744
10.550.4 Function Documentation . . . 1744
10.551 multilu_umfpack.c File Reference . . . 1755
10.551.1 Detailed Description . . . 1756
10.551.2 Macro Definition Documentation . . . 1756
10.551.3 Function Documentation . . . 1756
10.552 nvp.c File Reference . . . 1762
10.552.1 Detailed Description . . . 1762
10.553 nvpcall.c File Reference . . . 1762
10.553.1 Detailed Description . . . 1763
10.553.2 Function Documentation . . . 1763
10.554 nxIsSubClass.c File Reference . . . 1765
10.554.1 Detailed Description . . . 1765

10.555 MY_MODULE_mp_MY_SUB-UPPER.c File Reference . . . 1765
10.555.1 Function Documentation . . . 1766
10.556 MY_MODULE_mp_MY_SUB-UPPER.c File Reference . . . 1766
10.556.1 Function Documentation . . . 1766
10.557 my_module_MP_my_sub.c File Reference . . . 1766
10.557.1 Function Documentation . . . 1766
10.558 my_module_MP_my_sub.c File Reference . . . 1766
10.558.1 Function Documentation . . . 1766
10.559 my_module_mp_my_sub_.c File Reference . . . 1766
10.559.1 Function Documentation . . . 1766
10.560 my_module_mp_my_sub_.c File Reference . . . 1766
10.560.1 Function Documentation . . . 1767
10.561 my_module_my_sub_.c File Reference . . . 1767
10.561.1 Function Documentation . . . 1767
10.562 my_module_my_sub_.c File Reference . . . 1767
10.562.1 Function Documentation . . . 1767
10.563 my_moduleSmy_sub.c File Reference . . . 1767
10.563.1 Function Documentation . . . 1767
10.564 my_moduleSmy_sub.c File Reference . . . 1767
10.564.1 Function Documentation . . . 1767
10.565 MY_SUB-UPPER.c File Reference . . . 1767
10.565.1 Function Documentation . . . 1768
10.566 MY_SUB-UPPER.c File Reference . . . 1768
10.566.1 Function Documentation . . . 1768
10.567 my_sub.c File Reference . . . 1768
10.567.1 Function Documentation . . . 1768
10.568 my_sub.c File Reference . . . 1768
10.568.1 Function Documentation . . . 1768
10.569 my_sub_.c File Reference . . . 1768
10.569.1 Function Documentation . . . 1768

10.570my_sub_.c File Reference . . . 1768
10.570.1Function Documentation . . . 1769
10.571my_sub__.c File Reference . . . 1769
10.571.1Function Documentation . . . 1769
10.572my_sub__.c File Reference . . . 1769
10.572.1Function Documentation . . . 1769
10.573MYMODULE_mp_MYSUB-UPPER.c File Reference . . . 1769
10.573.1Function Documentation . . . 1769
10.574MYMODULE_mp_MYSUB-UPPER.c File Reference . . . 1769
10.574.1Function Documentation . . . 1769
10.575mymodule_MP_mysub.c File Reference . . . 1769
10.575.1Function Documentation . . . 1770
10.576mymodule_MP_mysub.c File Reference . . . 1770
10.576.1Function Documentation . . . 1770
10.577mymodule_mp_mysub_.c File Reference . . . 1770
10.577.1Function Documentation . . . 1770
10.578mymodule_mp_mysub_.c File Reference . . . 1770
10.578.1Function Documentation . . . 1770
10.579mymodule_mysub_.c File Reference . . . 1770
10.579.1Function Documentation . . . 1770
10.580mymodule_mysub_.c File Reference . . . 1770
10.580.1Function Documentation . . . 1771
10.581mymoduleSmysub.c File Reference . . . 1771
10.581.1Function Documentation . . . 1771
10.582mymoduleSmysub.c File Reference . . . 1771
10.582.1Function Documentation . . . 1771
10.583MYSUB-UPPER.c File Reference . . . 1771
10.583.1Function Documentation . . . 1771
10.584MYSUB-UPPER.c File Reference . . . 1771
10.584.1Function Documentation . . . 1771

10.585 mysub.c File Reference . . . 1771
10.585.1 Function Documentation . . . 1772
10.586 mysub.c File Reference . . . 1772
10.586.1 Function Documentation . . . 1772
10.587 mysub_.c File Reference . . . 1772
10.587.1 Function Documentation . . . 1772
10.588 mysub_.c File Reference . . . 1772
10.588.1 Function Documentation . . . 1772
10.589 newlu.c File Reference . . . 1772
10.589.1 Detailed Description . . . 1773
10.589.2 Macro Definition Documentation . . . 1773
10.589.3 Function Documentation . . . 1773
10.590 norm.c File Reference . . . 1784
10.590.1 Detailed Description . . . 1784
10.590.2 Function Documentation . . . 1785
10.591 norm.c File Reference . . . 1786
10.591.1 Detailed Description . . . 1786
10.591.2 Variable Documentation . . . 1786
10.592 offsets.c File Reference . . . 1786
10.592.1 Detailed Description . . . 1787
10.592.2 Macro Definition Documentation . . . 1787
10.593 ones.c File Reference . . . 1787
10.593.1 Detailed Description . . . 1788
10.594 ones.c File Reference . . . 1788
10.594.1 Detailed Description . . . 1788
10.595 openmp.c File Reference . . . 1788
10.595.1 Detailed Description . . . 1788
10.595.2 Macro Definition Documentation . . . 1788
10.596 options.c File Reference . . . 1788
10.596.1 Detailed Description . . . 1789

10.597 options.c File Reference . . . 1789
10.597.1 Detailed Description . . . 1789
10.598 options.c File Reference . . . 1789
10.598.1 Detailed Description . . . 1790
10.598.2 Macro Definition Documentation . . . 1790
10.599 orth.c File Reference . . . 1790
10.599.1 Detailed Description . . . 1790
10.600 output.txt File Reference . . . 1791
10.601 para.c File Reference . . . 1791
10.601.1 Detailed Description . . . 1791
10.601.2 Function Documentation . . . 1791
10.602 perm.c File Reference . . . 1792
10.602.1 Detailed Description . . . 1793
10.602.2 Function Documentation . . . 1793
10.603 perm.c File Reference . . . 1796
10.603.1 Detailed Description . . . 1797
10.603.2 Function Documentation . . . 1797
10.604 pinv.c File Reference . . . 1799
10.604.1 Detailed Description . . . 1799
10.605 plot_scriptExporter.c File Reference . . . 1799
10.605.1 Detailed Description . . . 1800
10.605.2 Function Documentation . . . 1800
10.606 plot_scriptExporter.h File Reference . . . 1801
10.606.1 Detailed Description . . . 1802
10.607 plot_x11.h File Reference . . . 1802
10.607.1 Detailed Description . . . 1803
10.608 precond.c File Reference . . . 1803
10.608.1 Detailed Description . . . 1803
10.609 print.c File Reference . . . 1803
10.609.1 Variable Documentation . . . 1803

10.610 profile.txt File Reference . . . 1803
10.611 proj_sym.c File Reference . . . 1803
10.611.1 Detailed Description . . . 1804
10.612 project.c File Reference . . . 1804
10.612.1 Detailed Description . . . 1804
10.613 pymess.c File Reference . . . 1804
10.613.1 Detailed Description . . . 1805
10.614 pymess.h File Reference . . . 1805
10.615 rail_371_c60.C File Reference . . . 1807
10.616 rand.c File Reference . . . 1807
10.616.1 Detailed Description . . . 1808
10.616.2 Macro Definition Documentation . . . 1808
10.616.3 Function Documentation . . . 1808
10.617 rand.c File Reference . . . 1808
10.617.1 Detailed Description . . . 1809
10.617.2 Function Documentation . . . 1809
10.618 rank.c File Reference . . . 1809
10.618.1 Detailed Description . . . 1809
10.619 rc2norm.c File Reference . . . 1809
10.619.1 Detailed Description . . . 1810
10.619.2 Macro Definition Documentation . . . 1810
10.619.3 Function Documentation . . . 1810
10.620 rcFnorm.c File Reference . . . 1811
10.620.1 Detailed Description . . . 1811
10.621 read.c File Reference . . . 1811
10.621.1 Detailed Description . . . 1812
10.621.2 Macro Definition Documentation . . . 1812
10.621.3 Function Documentation . . . 1813
10.622 real_complex.c File Reference . . . 1818
10.622.1 Detailed Description . . . 1819

10.622.2 Function Documentation . . . 1819
10.623 realimag.c File Reference . . . 1819
10.623.1 Detailed Description . . . 1819
10.624 reimplemented.c File Reference . . . 1820
10.624.1 Detailed Description . . . 1820
10.625 res2.c File Reference . . . 1820
10.625.1 Detailed Description . . . 1820
10.626 res2.c File Reference . . . 1820
10.626.1 Detailed Description . . . 1821
10.626.2 Macro Definition Documentation . . . 1821
10.626.3 Typedef Documentation . . . 1822
10.626.4 Function Documentation . . . 1822
10.627 res2_template.c File Reference . . . 1830
10.627.1 Detailed Description . . . 1831
10.627.2 Macro Definition Documentation . . . 1831
10.627.3 Typedef Documentation . . . 1831
10.627.4 Function Documentation . . . 1831
10.628 scal.c File Reference . . . 1841
10.628.1 Detailed Description . . . 1842
10.629 scalars.txt File Reference . . . 1842
10.630 scale.c File Reference . . . 1842
10.630.1 Detailed Description . . . 1842
10.630.2 Variable Documentation . . . 1842
10.631 schur.c File Reference . . . 1842
10.631.1 Detailed Description . . . 1843
10.631.2 Macro Definition Documentation . . . 1843
10.631.3 Function Documentation . . . 1843
10.632 schur_to_evd.c File Reference . . . 1843
10.632.1 Detailed Description . . . 1843
10.633 select.c File Reference . . . 1844

10.633.1 Detailed Description . . . 1844
10.634 setelement.c File Reference . . . 1844
10.634.1 Detailed Description . . . 1844
10.634.2 Function Documentation . . . 1844
10.635 shellsort.c File Reference . . . 1845
10.635.1 Detailed Description . . . 1845
10.635.2 Function Documentation . . . 1845
10.636 sign.c File Reference . . . 1846
10.636.1 Detailed Description . . . 1846
10.636.2 Function Documentation . . . 1846
10.637 solo1.c File Reference . . . 1847
10.637.1 Detailed Description . . . 1847
10.638 solo1.c File Reference . . . 1847
10.638.1 Detailed Description . . . 1847
10.639 solo2.c File Reference . . . 1847
10.639.1 Detailed Description . . . 1847
10.640 solo2.c File Reference . . . 1847
10.640.1 Detailed Description . . . 1848
10.641 solver.h File Reference . . . 1848
10.641.1 Detailed Description . . . 1849
10.642 solvername.h File Reference . . . 1850
10.642.1 Detailed Description . . . 1850
10.642.2 Macro Definition Documentation . . . 1850
10.643 sor.c File Reference . . . 1850
10.643.1 Detailed Description . . . 1850
10.644 sor.c File Reference . . . 1850
10.644.1 Detailed Description . . . 1851
10.644.2 Function Documentation . . . 1851
10.645 sort.c File Reference . . . 1852
10.645.1 Detailed Description . . . 1853

10.645.2 Function Documentation . . . 1853
10.646 source.txt File Reference . . . 1854
10.647 source.txt File Reference . . . 1854
10.647.1 Function Documentation . . . 1855
10.647.2 Variable Documentation . . . 1855
10.648 source.txt File Reference . . . 1855
10.649 source.txt File Reference . . . 1855
10.650 source.txt File Reference . . . 1855
10.651 source.txt File Reference . . . 1855
10.652 source.txt File Reference . . . 1855
10.653 source.txt File Reference . . . 1855
10.653.1 Function Documentation . . . 1856
10.653.2 Variable Documentation . . . 1856
10.654 source.txt File Reference . . . 1856
10.654.1 Typedef Documentation . . . 1857
10.654.2 Variable Documentation . . . 1857
10.655 splsolve.c File Reference . . . 1857
10.655.1 Detailed Description . . . 1857
10.656 spy.c File Reference . . . 1857
10.656.1 Detailed Description . . . 1857
10.657 stable.c File Reference . . . 1858
10.657.1 Detailed Description . . . 1858
10.658 status.c File Reference . . . 1858
10.658.1 Detailed Description . . . 1858
10.659 status.c File Reference . . . 1858
10.659.1 Detailed Description . . . 1859
10.660 status.c File Reference . . . 1859
10.660.1 Detailed Description . . . 1859
10.661 status.c File Reference . . . 1859
10.661.1 Detailed Description . . . 1859

10.662 statusopt.c File Reference . . . 1860
10.662.1 Detailed Description . . . 1860
10.663 std.c File Reference . . . 1860
10.663.1 Detailed Description . . . 1860
10.664 std.c File Reference . . . 1860
10.664.1 Detailed Description . . . 1861
10.665 stein_res2.c File Reference . . . 1861
10.665.1 Detailed Description . . . 1861
10.666 sub.c File Reference . . . 1861
10.666.1 Detailed Description . . . 1861
10.667 superlu.c File Reference . . . 1861
10.667.1 Detailed Description . . . 1861
10.668 superlu_refinement_bug.c File Reference . . . 1862
10.668.1 Function Documentation . . . 1862
10.669 superlu_refinement_bug_43.c File Reference . . . 1862
10.669.1 Function Documentation . . . 1862
10.670 superlu_refinement_bug_50.c File Reference . . . 1862
10.670.1 Function Documentation . . . 1862
10.671 suppress_cppcheck.txt File Reference . . . 1862
10.672 svd.c File Reference . . . 1862
10.672.1 Detailed Description . . . 1862
10.673 sylv_res2.c File Reference . . . 1863
10.673.1 Detailed Description . . . 1863
10.674 sylvester_dense.c File Reference . . . 1863
10.674.1 Detailed Description . . . 1864
10.674.2 Typedef Documentation . . . 1864
10.674.3 Function Documentation . . . 1864
10.675 sylvester_dense.c File Reference . . . 1871
10.675.1 Detailed Description . . . 1871
10.675.2 Function Documentation . . . 1872

10.676 sylvester_sparsedense.c File Reference . . . 1873
10.676.1 Detailed Description . . . 1874
10.676.2 Typedef Documentation . . . 1874
10.676.3 Enumeration Type Documentation . . . 1874
10.676.4 Function Documentation . . . 1874
10.677 sylvester_sparsedense.c File Reference . . . 1880
10.677.1 Detailed Description . . . 1880
10.678 sylvester_sparsedense.c File Reference . . . 1880
10.678.1 Detailed Description . . . 1880
10.678.2 Function Documentation . . . 1881
10.679 TargetDirectories.txt File Reference . . . 1882
10.680 TargetDirectories.txt File Reference . . . 1882
10.681 TargetDirectories.txt File Reference . . . 1882
10.682 TargetDirectories.txt File Reference . . . 1882
10.683 test.c File Reference . . . 1882
10.683.1 Detailed Description . . . 1882
10.684 testRestrict.c File Reference . . . 1883
10.684.1 Function Documentation . . . 1883
10.685 threadpool.c File Reference . . . 1883
10.685.1 Detailed Description . . . 1884
10.685.2 Function Documentation . . . 1884
10.686 threadpool.h File Reference . . . 1884
10.686.1 Detailed Description . . . 1885
10.687 time.c File Reference . . . 1885
10.687.1 Detailed Description . . . 1885
10.687.2 Macro Definition Documentation . . . 1885
10.688 trace.c File Reference . . . 1885
10.688.1 Detailed Description . . . 1886
10.689 transfer.c File Reference . . . 1886
10.689.1 Detailed Description . . . 1886

10.689.2 Function Documentation . . . 1886
10.690 transpose.c File Reference . . . 1886
10.690.1 Detailed Description . . . 1887
10.691 tridiag.c File Reference . . . 1887
10.691.1 Detailed Description . . . 1887
10.692 trisolve.c File Reference . . . 1887
10.692.1 Detailed Description . . . 1889
10.692.2 Function Documentation . . . 1889
10.693 triul.c File Reference . . . 1890
10.693.1 Detailed Description . . . 1890
10.694 tsia.c File Reference . . . 1890
10.694.1 Detailed Description . . . 1890
10.694.2 Macro Definition Documentation . . . 1891
10.695 tsiag.c File Reference . . . 1891
10.695.1 Detailed Description . . . 1891
10.695.2 Macro Definition Documentation . . . 1891
10.696 tutorial_backslash.c File Reference . . . 1891
10.696.1 Detailed Description . . . 1891
10.697 tutorial_bt.c File Reference . . . 1892
10.697.1 Detailed Description . . . 1892
10.698 tutorial_bt_gglrsrm.c File Reference . . . 1896
10.698.1 Detailed Description . . . 1896
10.699 tutorial_bt_gslrsrm.c File Reference . . . 1898
10.699.1 Detailed Description . . . 1898
10.700 tutorial_bt_lrsrm.c File Reference . . . 1900
10.700.1 Detailed Description . . . 1900
10.701 tutorial_bt_lrsrm_somor.c File Reference . . . 1902
10.701.1 Detailed Description . . . 1902
10.702 tutorial_bt_lrsrm_somor_so.c File Reference . . . 1904
10.702.1 Detailed Description . . . 1904

10.703 Tutorial_care.c File Reference . . . 1906
10.703.1 Detailed Description . . . 1906
10.704 Tutorial_direct_solvers.c File Reference . . . 1908
10.704.1 Detailed Description . . . 1908
10.705 Tutorial_eig.c File Reference . . . 1912
10.705.1 Detailed Description . . . 1912
10.706 Tutorial_eigs.c File Reference . . . 1914
10.706.1 Detailed Description . . . 1914
10.707 Tutorial_gen_fdm_cd.c File Reference . . . 1915
10.707.1 Detailed Description . . . 1915
10.708 Tutorial_gen_laplace.c File Reference . . . 1916
10.708.1 Detailed Description . . . 1916
10.709 Tutorial_gschur.c File Reference . . . 1917
10.709.1 Detailed Description . . . 1917
10.710 Tutorial_h2norm.c File Reference . . . 1918
10.710.1 Detailed Description . . . 1918
10.711 Tutorial_irka_biorth.c File Reference . . . 1919
10.711.1 Detailed Description . . . 1919
10.712 Tutorial_it_unsym.c File Reference . . . 1921
10.712.1 Detailed Description . . . 1921
10.713 Tutorial_lradi.c File Reference . . . 1925
10.713.1 Detailed Description . . . 1925
10.714 Tutorial_lrnm-g.c File Reference . . . 1928
10.714.1 Detailed Description . . . 1928
10.715 Tutorial_lrnm.c File Reference . . . 1930
10.715.1 Detailed Description . . . 1930
10.716 Tutorial_lyap.c File Reference . . . 1933
10.716.1 Detailed Description . . . 1933
10.717 Tutorial_matrix_operations.c File Reference . . . 1935
10.717.1 Detailed Description . . . 1935

10.718 tutorial_norms.c File Reference . . . 1938
10.718.1 Detailed Description . . . 1938
10.719 tutorial_plot.c File Reference . . . 1939
10.719.1 Detailed Description . . . 1939
10.720 tutorial_qr.c File Reference . . . 1941
10.720.1 Detailed Description . . . 1941
10.721 tutorial_schur.c File Reference . . . 1943
10.721.1 Detailed Description . . . 1943
10.722 tutorial_sign.c File Reference . . . 1944
10.722.1 Detailed Description . . . 1944
10.723 tutorial_spy.c File Reference . . . 1945
10.723.1 Detailed Description . . . 1945
10.724 tutorial_sylvester.c File Reference . . . 1947
10.724.1 Detailed Description . . . 1947
10.725 tutorial_tsia.c File Reference . . . 1950
10.725.1 Detailed Description . . . 1950
10.726 tutorial_vector_read.c File Reference . . . 1950
10.726.1 Detailed Description . . . 1950
10.727 tutorials.dox File Reference . . . 1951
10.728 woside_mgs.c File Reference . . . 1951
10.728.1 Detailed Description . . . 1951
10.728.2 Macro Definition Documentation . . . 1951
10.729 umfpack.c File Reference . . . 1951
10.729.1 Detailed Description . . . 1952
10.729.2 Macro Definition Documentation . . . 1952
10.730 umfpack_control.c File Reference . . . 1952
10.730.1 Detailed Description . . . 1954
10.730.2 Macro Definition Documentation . . . 1954
10.730.3 Typedef Documentation . . . 1955
10.730.4 Function Documentation . . . 1955

10.731 vector.c File Reference . . . 1966
10.731.1 Detailed Description . . . 1966
10.731.2 Macro Definition Documentation . . . 1967
10.732 vector.h File Reference . . . 1967
10.732.1 Detailed Description . . . 1971
10.733 vector_print.c File Reference . . . 1971
10.733.1 Detailed Description . . . 1971
10.734 vector_read.c File Reference . . . 1971
10.734.1 Detailed Description . . . 1972
10.734.2 Macro Definition Documentation . . . 1972
10.734.3 Function Documentation . . . 1972
10.735 VerifyFortran.h File Reference . . . 1974
10.735.1 Macro Definition Documentation . . . 1974
10.736 wachspress.c File Reference . . . 1974
10.736.1 Detailed Description . . . 1974
10.736.2 Function Documentation . . . 1975
10.737 write.c File Reference . . . 1976
10.737.1 Detailed Description . . . 1976
10.737.2 Macro Definition Documentation . . . 1976
10.738 811init.c File Reference . . . 1976
10.738.1 Detailed Description . . . 1977
10.738.2 Typedef Documentation . . . 1977
10.738.3 Function Documentation . . . 1977
10.739 zerbla.c File Reference . . . 1981
10.739.1 Detailed Description . . . 1981
10.739.2 Function Documentation . . . 1981

Bibliography **1983**

Index **1985**

1 M.E.S.S. - Matrix Equation Sparse Solver

1.1 Welcome to M.E.S.S.

M.E.S.S., the Matrix Equation Sparse Solver library, is the successor to the Lyapack Toolbox for MATLAB. It is available as a MATLAB toolbox as well as a C library. It is intended for the solution of symmetric linear anyd quadratic, differential and algebraic matrix equations with real, large, and sparse (or sparse + low-rank) coefficient matrices in the linear terms, low-rank quadratic and constant terms, and linearow-rank solutions. The C version provides a large set of auxillary subroutines for sparse matrix computations and efficient usage of modern multicore workstations.

1.2 Features

- Solving large scale (generalized) Lyapunov and Riccati Equations.
- Model order reduction for 1st and 2nd order systems via Balanced Truncation.
- $\mathcal{H}_2$ model order reduction via the IRKA and TSIA algorithms.
- Tools for LQR problems.
- A uniform inferface to handle sparse and dense matrices.
- Shared memory parallel algorithms using OpenMP and Pthreads.
- A solver for special sparse-dense Sylvester equations.
- A large set of direct and iterative solvers for linear systems.
- Various helper subroutines for thread pools, configuration files, interfaces to Python.

1.3 Overview of the Functionality

The libary is organized in different functional groups:

- General Type Definitions and Macros
- Matrix Operations
- Vector Operations
- Direct Solvers for Linear Systems
- Solvers for Sets of Linear Systems
- Iterative Solvers for Linear Systems
- Dynamical Systems
- Eigenvalue and Related Problems
- Low-Rank Methods for Matrix Equations
- Easy Frontends to Main Functions
- Graph Algorithms
- Matrix Generators
- Error Handling

- Interfaces to other Programming Languages and Libraries
- Plotting Interface
- Miscellaneous Helper Functions
- Tutorials
- Software Testing

1.4 Supported Matrix Equations

1.4.1 Algebraic Lyapunov equations

$$AXE^T + EXA^T + BB^T = 0$$
$$A^TXE + E^TXA + C^TC = 0$$

These equations are solved via the low-rank ($X = ZZ^T$, or $X = LDL^T$) alternating directions implicit iteration (LR-ADI) using the function mess_lradi. For the case where E, A are real sparse matrices we also provide a lyap-like function mess_lyap and mess_glyap.

1.4.2 Continuous Time Algebraic Riccati equations

$$AXE^T + EXA^T - EXBB^TXE^T + C^TC = 0$$
$$A^TXE + E^TXA - E^TXC^TCXE + BB^T = 0$$

For these standard control and filter equations we provide several solvers. Similar to mess_lyap above we also provide a quick-start interface mess_care using a similar syntax as care. For experts and for use with the non-standard equations, we implemented a low-rank Newton ADI method with linesearch in mess_lrnm.

1.5 Installation

For installation details please look at the Installation page.

1.6 Contact

The main website of **M.E.S.S.** is located at http://www.mpi-magdeburg.mpg.de/projects/mess.

File	Description

2 Tutorials

In this section we have tutorials for the following applications:

File	Description
tutorial_matrix_operations.c	show basic algebraic matrix operations
tutorial_direct_solvers.c	show how to solve a linear system of equations
tutorial_lyap.c	using mess_lyap to solve a (generalized) Lyapunov Equation
tutorial_care.c	using mess_care to solve a (generalized) Riccati Equation
tutorial_lradi.c	using mess_lradi to solve a (generalized) Lyapunov Equation
tutorial_lrnm.c	using mess_lrnm to solve a (generalized) Riccati Equation
tutorial_bt.c	using mess_bt_lrsrm to reduce a LTI-System
tutorial_plot.c	using mess_plotExport instance to generate and export plots
tutorial_spy.c	using mess_matrix_spy to plot the sparsity pattern of a matrix
tutorial_eig.c	compute and plot eigenvalues and singular values of a dense matrix
tutorial_sign.c	using mess_eigen_sign to compute the sign function of a matrix
tutorial_sylvester.c	Solving standard or generalized sparse-dense Sylvester equations.

Further examples for balanced trunctation:

File	Description
tutorial_bt_gglrsrm.c	reduce a first order generalized LTI system using low-rank square root balanced trunction
tutorial_bt_gslrsrm.c	reduce a first order generalized LTI to a first order LTI system using low-rank square root balanced trunction
tutorial_bt_lrsrm.c	reduce a first order LTI to a first order LTI system using low-rank square root balanced trunction
tutorial_bt_lrsrm_somor.c	reduce a second order LTI to a first order LTI system using low-rank square root balanced trunction
tutorial_bt_lrsrm_somor_so.c	reduce a second order LTI to a second order LTI system using low-rank square root balanced trunction

Further examples for eigenvalue computations:

File	Description
tutorial_eig.c	show dense eigenvalue computations using mess_eigen_eig
tutorial_eigs.c	show sparse eigenvalue computations using mess_eigen_eigs and ARPACK interface
tutorial_gschur.c	show generalized schur computation using mess_eigen_gschur mess_eigen_gschur_complex
tutorial_schur.c	using mess_eigen_schur and mess_eigen_schur_complex to compute schur decomposition

Further examples for miscallenous examples:

File	Description
tutorial_vector_read.c	generate a random vector and read and write operations
tutorial_norms.c	compute different matrix norms
tutorial_it_unsym.c	usage of iterative solvers
tutorial_gen_fdm_cd.c	convection diffusion matrix generation
tutorial_gen_laplace.c	discrete laplace operator matrix generation
tutorial_h2norm.c	compute the $\mathcal{H}_2$ Norm
tutorial_irka_biorth.c	$\mathcal{H}_2$ model order reduction via irka algorithm
tutorial_tsia.c	$\mathcal{H}_2$ model order reduction via tsia algorithm
tutorial_backslash.c	solve a linear system using mess_matrix_backslash
tutorial_lrnm-g.c	solve a algebraic riccati equation and show stepfunction application
tutorial_qr.c	solve a linear system using mess_direct_create_lapack_qr

3 Install Instructions

Build dependencies

C-M.E.S.S. uses the `CMake` build system to configure the source code. In contrast to the GNU Autotools this is a relatively new tool, but it is used by many, even large, projects like KDE or MySQL. Unfortunately you need to install CMake before you are able to build C-M.E.S.S.

C-M.E.S.S. requires a C and Fortran Compiler supporting at least OpenMP 2.5

- g++ + gcc + gfortran >= 4.8
- icc+ifort >= Composer XE 2013
- clang >= 3.9

The key library dependencies are:

- LAPACK > 3.2.0
- SuiteSparse > 3.7.0

Optionally C-M.E.S.S. depends on:

- Python >= 2.7 with NumPy (>= 1.6) and SciPy (>= 0.9.0)
- libmatio >= 1.5.0
- Doxygen
- ARPACK
- Serveral compression libraries like zlib, bz2, lzma

Preparation on Systems with Package-Managers

The dependencies can be installed as well using the package management:

On Debian(>=7.0)/Ubuntu(>=14.04):

```
sudo apt-get install build-essential cmake gfortran libblas-dev liblapack-dev \
     libsuitesparse-dev zlib1g-dev  libbz2-dev liblzma-dev libarpack2-dev libmatio-dev \
     doxygen python3-dev python-dev python-scipy python3-scipy
```

If the cmake package installs an old (<3.0.2) version of cmake check if the `cmake3` is available.

On Redhat/Fedora/CentOS or Scientific Linux:

```
sudo yum install gcc-gfortran cmake blas-devel lapack-devel suitesparse-devel zlib-devel
     bzip2-devel rpm-build lzma-devel arpack-devel python-devel numpy scipy
```

If the cmake package installs an old (<3.0.2) version of cmake check if the `cmake3` is available.

On FreeBSD:

```
portinstall gcc cmake blas lapack suitesparse
```

or via the pkg system

```
pkg install gcc cmake blas lapack suitesparse
```

On MacOSX:

```
brew install cmake gcc suite-sparse
```

You have to specify the compilers on MacOSX, e.g.

```
CC=gcc-7 CXX=g++-7 FC=gfortan-7 cmake ../ <OPTIONS>
```

For an optimal performance we recommend to install an optimized and threaded BLAS library like Intel MKL, Open↩ BLAS or ATLAS.

Build and install MESS

If you have installed or build all dependencies, you can build the MESS library. We recommend an out-of-source build to keep the sources clean.

```
mkdir mess-build
cd mess-build
```

From inside this build directory you can now run CMake

```
cmake ../ <OPTIONS>
```

and build and install MESS using

```
make
make install
```

Options for CMake

Options that influences the behavior of the configuration process are:

Parameter	Description
-DCMAKE_INSTALL_PREFIX=/path/	Install path for make install. Default /usr/local/
-DMESS64=ON/OFF	Turn on 64-bit integers in MESS This requires all external libraries to work with 64bit integers too. Default: OFF
-DDEBUG=ON/OFF	Enables the debug build. Optimizations are turned off, verbose output of the library, debug symbols are included. Default: OFF
-DCOVERAGE=ON/OFF	Enable the code coverage, only useful with the DEBUG option and the automatic tests. Default: OFF
-DTESTS=ON/OFF	Build software tests. Default: ON
-DLARGETESTS=ON/OFF	Build long running software tests. Default: OFF
-DHUGETESTS=ON/OFF	Build an extensiv test. Try to test all possible combination of LRADI/L↩ RNM options. Default: OFF
-DPYTHON=ON/OFF	Try to build the Py-M.E.S.S. interface. It searches for all available Python interpreters and if the support NumPy and SciPY. Default: OFF
-DPYTHON_VERSION=x.y	Search only for the given Python version x.y instead of all.
-DMATIO=ON/OFF	Enables MATIO support to read form Matlab files. If MATIO is compiled with HDF5 support, MESS can load Matlab v7.3 files. Default: OFF
-DDOC=ON/OFF	Enable the `make doc` target to build the documentation. Default: OFF.
-DDOCPDF=ON/OFF	Enable the `make doc` target to build the documentation as PDF. Default: OFF.
-DMATHJAX=ON/OFF	Enable MATHJAX for documentation. Default: OFF.
-DSUPERLU=ON/OFF	Enable SuperLU support. Default OFF.
-DSUPERLU_ROOT=/path/	Give CMake a path, where to find your SuperLU build directory.
-DARPACK=ON/OFF	Enable Arpack support. Default OFF.
-DLD_MULTIPLICITY=NUMBER	Set Leading Dimension of Dense Matrices to a multiple of NUMBER. Default 1.
-DSUITESPARSE=/path/	Give CMake a path, to your SuiteSparse build directory.
-DBLAS_LIBRARIES=lib	Specify a BLAS library.
-DLAPACK_LIBRARIES=lib	Specify a LAPACK library.
-DMATLAB=ON/OFF	Enable Mex-M.E.S.S. Interface
-DMATLABROOT=/path/	Give CMake a path, to your Matlab installation path

If CMake detects the wrong compilers you can specify them by setting the *CC*, *CXX* and *FC* environment variables. In this way the Intel Compiler suite can be enabled using

```
CC=icc CXX=icpc FC=ifort cmake ../ <OPTIONS>
```

Build the Documentation

In order to build the documentation you have to install `doxygen` before you run the configure step. If `-DDOC=ON` is given as an argument to `CMake` the *make doc* target is enabled. The documentation is build in `doc/html` by calling

```
make doc
```

Caused by a problem in Doxygen the bibliography is only integrated in the HTML page if the installed Doxygen version is higher than 1.8.3.

Run the Software-Tets

If the tests are enabled you can verify your configuration after the build process using

```
make test
```

The coverage can be enabled using the `COVERAGE` option and needs `gcov` and `lcov` to be installed on the system. The code coverage is done using

```
make coverage
```

If valgrind was found during the configuration, you can run all tests with valgrind using

```
make memtest
```

Py-M.E.S.S. Interface and Sphinx Documentation

In order to build the Py-M.E.S.S. Documentation you need `sphinx`, `numpydoc`, `matplotlib` and a working `Py-M.E.S.S.` installation. You can build the `Py-M.E.S.S.` by setting `PYTHON=ON`. `Py-M.E.S.S.` needs `numpy` and `scipy`. After a successfull CMake configuration type

```
make
```

and change into the directory `python` and using the `setup.py` files to install the Python interface

```
cd python/python_X.Y
python setup.py build && sudo python setup.py install
```

For a local installation create a directory `mkdir ~/pymess_install` compile `C-M.E.S.S.` as described above and execute the following commands

```
cd python/python_X.Y
python setup.py build && python setup.py install --prefix=~/pymess_install
```

Do not forget to adapt the `PYTHONPATH` variable

```
export PYTHONPATH=/home/daniels/pymess_install/lib/python2.7/site-packages:$PYTHONPATH
```

If `CMake` found all necessary tools for the Py-M.E.S.S. documentation a special target for your `Python` executable is activated to generate the documentation:

```
make pymess-python2.7-doc-html
make pymess-python2.7-doc-pdf
```

Mex-M.E.S.S. Interface

To generate Mex-M.E.S.S the Matlab interface to C-M.E.S.S. set `MATLAB=ON` and give your Matlab installation directory via `MATLABROOT=/path/`. Make sure that you use compilers which are compatible with your Matlab installation

```
CC=gcc-4.9 CXX=g++-4.9 FC=gfortran-4.9 cmake ../cmess -DMATLAB=ON -DMATLABROOT=/path/
```

If the mex-compiler complains abot your gcc version configure your Matlab mex setup properly.

Start Matlab and change to the C-M.E.S.S. `<build>/matlab` directory. Execute `mess_path.m` script to add Mex-M.E.S.S. to the matlab path. Afterwards you can test the Mex-M.E.S.S. interface using

```
run_tests()
run_examples()
```

function calls.

Debian Packages

C-M.E.S.S. comes with all scripts required to build a debian packages. Therefore, use

```
dpkg-buildpackage -us -uc
```

to create the `libmess`, `libmess-dev`, and `libmess-doc` package for your current system.

4 Module Index

4.1 Modules

Here is a list of all modules:

General Type Definitions and Macros 44

Matrix Operations 48

- **Arithmetic Operations on Matrices** 50
- **Arithmetic Operations on Matrix Columns** 79
- **Computation of Matrix Norms** 91
- **Allocate, Convert and Access to Matrices** 109
- **Input/Output Operations with Matrices** 155
- **Various Other Operations on Matrices** 163
- **Data Structures, Types and Macros for Handling Matrices** 177
- **Generic matrix-vector Products** 183

Vector Operations 187

Data Structures and Related Functions 188
Arithmetic Operations with Vectors 197
Various Other Operations with Vectors 206
Direct Solvers for Linear Systems 234
Generic Interface to Direct Solvers 236
Direct Solvers for Linear Systems 247
Triangular Solvers 265
Direct Solvers for Matrix Equations 288
Matrix Decompositions 297
Residual Computations 300
Various Other Functions related to Direct Solvers. 314
Solvers for Sets of Linear Systems 317
Generic Interface to Multisolvers 318
Multisolvers 322
Iterative Solvers for Linear Systems 329
Types and Datastructures 330
Solvers 334
Preconditioners 344
Dynamical Systems 350
Interface for handling Dynamical Systems 351
Model Order Reduction 371
Balanced Truncation 372
H2 Model Order Reduction 392
Eigenvalue and Related Problems 424
Dense Eigenvalue Solvers 425
Sparse Eigenvalue Solvers 442
Low-Rank Methods for Matrix Equations 464
Generic Functions and Data Structures 465
The ADI and the ADI-NM Algorithm 483
Equation Generators 502
Define own Matrix Generators 534
Additional Matrix Equation Solvers 553

Miscellaneous Functions 569
Easy Frontends to Main Functions 585
Graph Algorithms 595
Matrix Generators 598
FDM Matrix Generators 599
Error Handling 604
Error Codes and Handling 605
Error Macros 610
Interfaces to other Programming Languages and Libraries 612
Interface to CSparse 613
Interface to Cholmod 619
Interface to Python 620
Plotting Interface 656
X11 Function Plotter 657
Plot Export 663
Miscellaneous Helper Functions 672
Hashtable 690
A Compact Loader Pool Implementation 691
Time Measurement 696
Various Macros 700
Software Testing 712
Basic Matrix and Vector Operations 714
Direct Solvers for Linear Systems 718
Eigenvalue and Related Problems 720
Input/Output Operations 722
Iterative Solvers for Linear Systems 723
Low-Rank Methods for Matrix Equations 724
Storage Formats 726
Miscellaneous Functions and Functions Related to Vectors 727
Interfaces_matlab 728

5 Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

__glyap_dae1 764

__glyap_dae2 766

__glyap_so1 769

__glyap_so2 772

__ilu0_data 774

__iluk 775

__lyap 776

__matlab_equation 778

__mess_lrcfadi_step 780

__mess_plotdata 782

__mess_plotExportData 783

__mess_plotter 784

__mess_plotterScript 788

__mess_solver_options 789

__mess_solver_status 791

__presordata 792

__stable_lyap 793

_equations_t 794

_glyap3_st 795

_mess_call 796

_scalepoints 797

_sylv_solver_d_st 798

_sylv_solver_sd_st 800

mess_equation_st::AINV_st 802

mess_equation_st::ApEINV_st 803

mess_equation_st::ApEX_st 804

mess_equation_st::AX_st 805

banded_lapack 806

bicgstab_solver 807

CCSRFmt 808

cholesky_solver 809

csparse_solver 810

csparse_solver_complex 811

mess_equation_st::EINV_st 812

mess_equation_st::EX_st 813

hessenberg_solver 814

int_list 815

jobdata_st 817

lapack_solver 819

lapackqr_solver 820

mat_diff 822

mess_bt_options_st 823

mess_bt_status_st 824

mess_direct_levelset_st 834

mess_direct_st 837

mess_dynsys_st 840

mess_eigen_arpack_options_t 842

mess_equation_st 847

mess_freelist_st 852

mess_h2_options_st 854

mess_h2_status_st 855

mess_krylov_arnoldi_st 857

mess_matrix_file_st 869

mess_matrix_st 871

mess_multidirect_st 879

mess_mvpcall_st 884

mess_options_st 892

mess_precond_st 899

mess_status_st 905

mess_threadpool_st 910

mess_threadpooljob_st 912

mess_timer_st 913

mess_vector_st 921

mklpardiso_solver 923

module_state 924

multi_umfpack 925

multilu 926

mvpdata 929

mvpdatag 930

newlu_data 932

norm2data 933

norminv2data 934
octave_base_value

- **MESS_datatype_t** 827
- **MESS_direct** 829
- **MESS_direct_cholpackage_t** 832
- **MESS_direct_lupackage_t** 835
- **MESS_equation** 844
- **MESS_equation_t** 850
- **MESS_matrix** 859
- **MESS_memusage_t** 873
- **MESS_multidirect** 875
- **MESS_multidirect_t** 882
- **MESS_operation_t** 885
- **MESS_options** 887
- **MESS_parameter_t** 897
- **MESS_residual_t** 900
- **MESS_status** 902
- **MESS_storage_t** 908
- **MESS_vector** 914

pair_mess_int_t2 935

pair_mess_int_t2_complex 935

pair_mess_int_t2_double 936

pair_mess_int_t_complex 937

pair_mess_int_t_double 938

parameter_mvp_call 939

rc2data 940

res2_data_st 941

res2nm_data_st 943

th_done_entry 944

umfpack_control_solver 945

VWjob 946

6 Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

__glyap_dae1
Contains matrices, solvers and additional data for a DAE 1 function handles ??

__glyap_dae2
Contains matrices, solvers and additional data for a DAE 2 function handles ??

__glyap_so1
Contains matrices, solvers and additional data for Second Order System function handles ??

__glyap_so2
Contains matrices, solvers and additional data for Second Order System function handles ??

__ilu0_data
Internal structure for $ILU(0)$ preconditioner. Internal structure for $ILU(0)$ preconditioner ??

__iluk ??

__lyap
Contains matrices, solvers and additional data for a generalized Lyapunov Equation ??

__matlab_equation ??

__mess_lrcfadi_step
Structure to pass information to the step function inside the LRCF iterations ??

__mess_plotdata
Structure for one data element in a plot ??

__mess_plotExportData
Structure for one data element in a plot ??

__mess_plotter
Structure to hold a complete plotter ??

__mess_plotterScript
Structure to hold a complete plotter ??

__mess_solver_options
Representation of options for iterative solvers ??

__mess_solver_status
Representation of status information of an iterative solvers ??

__presordata ??

__stable_lyap ??

_equations_t
Structure for all `Py-M.E.S.S.` to `M.E.S.S.` converters. Structure for all `Py-M.E.S.S.` to `M.E.S.S.` converters ??

_glyap3_st ??

_mess_call ??

_scalepoints ??

_sylv_solver_d_st ??

_sylv_solver_sd_st
Internal structure for sparse-dense Sylvester equation solver ??

mess_equation_st::AINV_st
Structure defining $op(A)Y = X$??

mess_equation_st::ApEINV_st
Structure defining $op(A + pE)Y = X$??

mess_equation_st::ApEX_st
Structure defining $Y = op(A + \alpha pE)X$??

mess_equation_st::AX_st
Structure defining $Y = op(A)X$??

banded_lapack ??

bicgstab_solver ??

CCSRFmt ??

cholesky_solver
Internal structure for the `LAPACK` based solver ??

csparse_solver ??

csparse_solver_complex ??

mess_equation_st::EINV_st
Structure defining $op(E)Y = X$??

mess_equation_st::EX_st
Structure defining $Y = op(E)X$??

hessenberg_solver
Internal structure for the `LAPACK` based solver ??

int_list ??

jobdata_st ??

lapack_solver
Internal structure for the LAPACK based solver ??

lapackqr_solver
Internal data structure. Internal data structure for QR/LQ decompostitions ??

mat_diff
Helper structure for mat_diff_mvp ??

mess_bt_options_st
Options structure for balanced truncation ??

mess_bt_status_st
Status structure for balanced truncation ??

MESS_datatype_t ??

MESS_direct ??

MESS_direct_cholpackage_t ??

mess_direct_levelset_st ??

MESS_direct_lupackage_t ??

mess_direct_st
Representation of a generic direct solver ??

mess_dynsys_st
Representation of a dynamical system ??

mess_eigen_arpack_options_t
Options structure for ARPACK ??

MESS_equation ??

mess_equation_st
Structure defining an equation as a set of function pointers ??

MESS_equation_t ??

mess_freelist_st
Structure containing a minimal memory manage in order to avoid memory leaks in the Python and MEX-M.E.S.S. interface ??

mess_h2_options_st
Structure to store $\mathcal{H}_2$ options ??

mess_h2_status_st
Structure to store $\mathcal{H}_2$ status ??

mess_krylov_arnoldi_st ??

MESS_matrix ??

mess_matrix_file_st
Type definition and structure for mess_matrix files ??

mess_matrix_st
The mess_matrix structure to handle dense and sparse matrices in a common context ??

MESS_memusage_t ??

MESS_multidirect ??

mess_multidirect_st
Representation of a generic direct solver ??

MESS_multidirect_t ??

mess_mvpcall_st
Represent a generic matrix-vector-product ??

MESS_operation_t ??

MESS_options ??

mess_options_st
Options for the LRCF-ADI algorithm family ??

MESS_parameter_t ??

mess_precond_st
Representation of a preconditioner ??

MESS_residual_t ??

MESS_status ??

mess_status_st
Structure/object to save the final status of a LRCF algorithm ??

MESS_storage_t ??

mess_threadpool_st
Representation of a thread pool ??

mess_threadpooljob_st
Representation of a thread pool job ??

mess_timer_st
Representation of a timer ??

MESS_vector ??

mess_vector_st
Representation of a real or complex vector ??

mklpardiso_solver ??

module_state ??

multi_umfpack ??

multilu
Internal structure to store a multilu ??

mvpdata ??

mvpdatag ??

newlu_data ??

norm2data ??

norminv2data ??

pair_mess_int_t2 ??

pair_mess_int_t2_complex ??

pair_mess_int_t2_double ??

pair_mess_int_t_complex ??

pair_mess_int_t_double ??

parameter_mvp_call ??

rc2data ??

res2_data_st ??

res2nm_data_st ??

th_done_entry ??

umfpack_control_solver ??

VWjob ??

7 File Index

7.1 File List

Here is a list of all files with brief descriptions:

symbols/__my_module__my_sub.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/__my_module__my_sub.c ??

symbols/__my_module_MOD_my_sub.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/__my_module_MOD_my_sub.c ??

symbols/__my_module_NMOD_my_sub.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/__my_module_NMOD_my_sub.c ??

symbols/__mymodule__mysub.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/__mymodule__mysub.c ??

symbols/__mymodule_MOD_mysub.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/__mymodule_MOD_mysub.c ??

symbols/__mymodule_NMOD_mysub.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/__mymodule_NMOD_mysub.c ??

add.c
Add two matrices ??

addcol.c
Add new columns to a matrix ??

matrix/alloc.c
Allocate and resize matrices ??

vector/alloc.c
Allocation function of mess_vector structures ??

amd.c
Compute AMD reordering of a matrix ??

matrix/any.c
Any function clone from MATLAB ??

vector/any.c
Anyfunction of mess_vector structures ??

arnoldi.c
Arnoldi iteration ??

arnoldi_template.c
Arnoldi iteration ??

arpack.c
Interface to ARPACK ??

axpy.c
Axpy function of mess_vector structures ??

backslash.c
Implementation of a MATLAB backslash like function call ??

banded.c
Interface to LAPACK or UMFPACK based banded solver ??

bandwidth.c
Determine the bandwidth of a matrix ??

ber_sgn_fac.c
Solve a Algebraic Bernoulli equation using the sign function method ??

direct/singlesolver/bicgstab.c
Wrapper to use bicgstab as single solver ??

itsolver/bicgstab.c
Solve $Ax = b$ with BiCGStab ??

biorth.c
Biorthonormalize two matrices ??

blas_defs.h
Fortran name mangling for LAPACK and BLAS ??

blockpart.c
Detect blocks inside a matrix ??

call_macro.h
Function call wrapper for test functions ??

callback_equation.c ??

lib/direct/care.c
Solve the algebraic Riccati Equation ??

lib/easyfrontend/care.c
Easy to use Riccati solver interface ??

python/c_interface/functions/care.c
Interface to mess_care ??

matrix/cat.c
Concatenate up to four matrices ??

vector/cat.c
Cat and Resize functions of mess_vector structures ??

cg.c
Conjugate gradients itsolver ??

check_add.c
Check matrix addition ??

check_addcol2p.c
Check the mess_matrix_addcols2p function ??

check_any.c
Check function mess_matrix_any ??

check_arnoldi_template.c
Check the mess_eigen_arnoldi_template function ??

check_arnoldi_template_nrm.c
Check the mess_eigen_arnoldi_template_nrm function ??

check_arpack.c
Check the solution of the eigenvalue problem using the Arnoldi process from ARPACK ??

check_arpack_lanczos.c
Check the solution of the eigenvalue problem using the Lanczos process from ARPACK ??

check_backslash.c
Check the solution of a system of linear equations ??

check_bandwidth.c
Check mess_matrix_bandwidth function ??

check_ber_sign_fac.c
Check the computation of the solution of Bernoulli equations using matrix sign function and its residual ??

check_biorth.c
Check the construction of a biorthonormal basis for two matrices ??

check_cat.c
Check the concatenation of two matrices ??

check_cholmod.c
Check cholmod.c functions ??

check_cimport.c
Check the creation of different storage type matrices from given arrays ??

check_cimport_c2d.c
Check the creation of a dense mess_matrix from a C 2 dimensional array ??

check_colaxpy2.c
Check the mess_matrix_colaxpy2 function ??

check_coldot.c
Check the column dot product ??

check_coldotc.c
Check the column dot product of a complex matrix ??

check_coldotcE.c
Check the mess_matrix_coldotcE function ??

check_coldotE.c
Check the mess_matrix_coldotE function ??

check_colnorm.c
Check the column norm ??

check_colscale.c
Check the column scaling function of a matrix ??

check_colsub.c
Check the column submatrix computation ??

check_colvecaxpy.c
Check the axpy scalar vector product ??

check_colvecdot.c
Check the column vector product ??

check_colvecdotc.c
Check the complex column vector product ??

check_condest.c
Check the condition number estimation ??

check_convert.c
Check the conversion subroutine ??

check_dae2_handles_adi.c
Check function handles for Hessenberg index 2 DAE systems, which are necessary for mess↩_lrcfadi_adi ??

check_dae2_handles_para.c
Check function handles for Hessenberg index 2 DAE systems, which are necessary for mess↩_lrcfadi_parameter ??

check_decomp.c
Check the mess_matrix_decomp function ??

check_dense_nm.c
Check the dense Newton Method for dense algebraic Riccati equations ??

check_dense_nm_tridiag.c
Check the solution of a generalized Riccati Equation in transposed and nontransposed version using mess_dense_nm_gmpare ??

check_det.c
Check the det and detc functions of the solver ??

check_dgeqp3.c
Check the mess_direct_dgeqp3 function ??

check_diag.c
Check the output of diagonal elements of a matrix ??

check_direct_care.c
Check the mess_direct_care and mess_direct_care_res2 function ??

check_dynorm2.c
Check the 2-norm computation of a matrix ??

check_dynorm2t.c
Check the 2-norm computation of a transposed matrix ??

check_dynorm_indefinite2.c
Check the mess_matrix_indefinite_dynorm2 function ??

check_dynorm_indefinitef.c
Check the mess_matrix_indefinite_dynormf and mess_matrix_indefinite_dynormf2 functions ??

check_dynormf.c
Check the Frobenius-norm computation of a matrix ??

check_dynormft.c
Check the Frobenius-norm computation of a transposed matrix ??

check_eig.c
Check the solution of the eigenvalue problem using xGEEV from LAPACK ??

check_eig2.c
Check the solution of the eigenvalue problem using xGEEV from LAPACK (complex version) ??

check_eig_complex.c
Check the solution of the eigenvalue problem using xGEEV from LAPACK (complex version) ??

check_eig_hessenberg_schur.c
Check the mess_eigen_hessenberg_schur, mess_eigen_schur_to_evd ??

check_eig_hessenberg_schur_largest_ev.c
Check the mess_eigen_hessenberg_abs_largest ??

check_eliminate_zeros.c
Check the mess_matrix_eliminate_zeros function ??

check_eps.c
Check the machine epsilon calculation ??

check_fputs_block.c
Check writing of large blocks ??

check_freelist.c
Check the mess_freelist_t structure ??

check_fro_inn.c
Check the frobenius inner product ??

check_gaxpy.c
Check the matrix-vector product with an update of a vector ??

check_getelement.c
Check if a matrix element is returned correctly ??

check_getLU.c
Check the getL and getU functions of the solver ??

check_gsign.c
Check the mess_eigen_gsign function ??

check_hessenberg.c
Check the mess_direct_create_hessenberg_lu solver ??

check_itsolver.c
Check different iterative solvers ??

check_kron.c
Check the Kronecker product computation ??

check_lapackqr.c
Check functions of mess_direct_create_lapack_qr for over and underdetermined systems ??

check_lift.c
Check the mess_matrix_lift and mess_vector_lift function ??

check_linesearch.c
Check the linesearch function defined in lrnm_linesearch.c ??

check_lradi_dae1.c
Check the solution of a Lyapunov Equation using the LRCF-ADI method and DAE 1 function handles ??

check_lradi_dae2.c
Check the solution of a Lyapunov Equation using the LRCF-ADI method and DAE 2 function handles ??

check_lradi_glyap.c
Check the solution of a Lyapunov Equation using the LRCF-ADI method ??

check_lradi_so.c
Check the solution of a Lyapunov Equation arising from a second order system ??

check_lradi_unsym.c
Check the solution of standard Lyapunov Equations using the LRCF-ADI method ??

check_lrnm_dae1.c
Check the solution of a Lyapunov Equation using the LRCF-ADI method and DAE 1 function handles ??

check_lrnm_dae2.c
Check the solution of generalized Riccati Equation arising from a Hessenberg index 2 DAE system using LRCF-NM ??

check_lrnm_g.c
Check the solution of a generalized Riccati Equation in transposed and nontransposed version using LRCF-NM ??

check_lrnm_so.c
Check the solution of a Riccati Equation arising from a second order system ??

check_lrnm_unsym.c
Check the solution of Riccati Equations using LRCF-NM ??

check_lyap_sign_fac.c
Check the computation of the solution of Lyapunov Equations using matrix sign function and its residual ??

check_matrix_biorth.c
Check the biorthonormal basis construction ??

check_matrix_colaxpy.c
Check the column update of a matrix using an axpy-operation ??

check_matrix_gbiorth.c
Check the biorthonormal basis construction with respect to a weighted scalar product ??

check_matrix_null.c
Check the orthogonal basis construction of the nullspace ??

check_matrix_sum.c
Check the column and row sum computation of a matrix ??

check_matrixio.c
Check IO operations on mess_matrix instances ??

check_mgs.c
Check the ortohogonal basis construction using the modified Gram-Schmidt process ??

check_mgs_inplace.c
Check the ortohogonal basis construction using the modified Gram-Schmidt process (inplace version) ??

check_mgsadd.c
Check the orthogonal basis construction using the modified Gram-Schmidt process (column adding version) ??

check_mul.c
Check the multiplication of two matrices ??

check_mul_small.c
Check the multiplication of two matrices (version with small matrices) ??

check_mul_tall.c
Check the multiplication of two matrices (version with tall matrices) ??

check_mulnormf2.c
Check the Frobenius-norm computation of a matrix ??

check_mvp.c
Check matrix-vector product computation ??

check_mvpcall_matrix.c
Check the mess_mvpcall_matrix function ??

check_norm.c
Check norm functions ??

check_one_value.c
Check the mess_matrix_one_value and mess_matrix_one_valuec functions ??

check_orth.c
Check the orthogonal basis construction using SVD ??

check_perm.c
Check the permutation function for matrices ??

check_pinv.c
Check the computation of the pseudoinverse of a matrix ??

check_precond.c
Check different preconditioner ??

check_proj_sym.c
Check the proj_sym function ??

check_rank.c
Check rank estimation ??

check_real_imag_part.c
Check the mess_matrix_realpart mess_matrix_imagpart and mess_matrix_complex_from_parts functions ??

check_rowsub.c
Check the row submatrix computation ??

check_scalem.c
Check the mess_matrix_rowscalem and mess_matrix_colscalem ??

check_schur.c
Check the mess_eigen_schur and mess_eigen_schur_complex functions ??

check_set_get_col_row.c
Check the overwriting of a matrix column ??

check_setelement.c
Check the overwriting of a matrix element ??

check_sign.c
Check the mess_eigen_sign function ??

check_so1_handles.c
Check function handles for second order systems (so1) ??

check_so2_handles.c
Check function handles for second order systems (so2) ??

check_sub.c
Check the sub matrix computation ??

check_svd.c
Check the singular value decomposition ??

check_svd_complex.c
Check the singular value decomposition (complex version) ??

check_svd_econ.c
Check the economy size singular value decomposition ??

check_svd_econ_complex.c
Check the economy size singular value decomposition (complex version) ??

check_trace.c
Check the trace computation ??

check_transpose.c
Check the transpose function ??

check_tridiag.c
Check mess_matrix_tridiag function ??

check_trisolve.c
Check the functions in trisolve.c ??

check_vector_perm.c
Check the permutation function for vectors ??

check_vectorio.c
Check IO operations on mess_vector instances ??

cholesky.c
Cholesky decomposition based on DPOTRF ??

cholfactor.c
Compute a Cholesky factor $X = ZZ^T$ of a matrix ??

cholmod.c
Import and export vectors and dense/csc matrices from `M.E.S.S.` to CHOLMOD ??

cholmod_chol.c
Interface for Tim Davis CHOLMOD (Cholesky Decomposition) ??

cimport.c
Import matrices from `C` and `Fortran` arrays ??

CMakeCCompilerId.c ??

CMakeCXXCompilerId.cpp ??

CMakeFortranCompilerId.F ??

col_compress.c
Column compression for the ADI process ??

colops.c
Perform operations directly on the columns of a matrix ??

colrowsums.c
Compute the column or row sum of a matrix ??

condest.c
Condition number estimaton ??

config.h ??

matrix/convert.c
Uniform matrix converting interface ??

vector/convert.c
Convert functions of mess_vector structures ??

copy.c
Copy functions of mess_vector structures ??

csc.c
Kernels for $y \leftarrow Ax + y$ in MESS_CSC storage ??

csparse.c
Interface for Tim Davis `CXSparse` ??

csparse_chol.c
Interface for Tim Davis `CXSparse` (Cholesky decomposition) ??

csr.c
Kernels for $y \leftarrow Ax + y$ in MESS_CSR storage ??

cuthill.c
Cuthill MC Kee Reordering ??

matlab/c_interface/conv/eqn/dae1.c
Equation conversion for dae1 function handles ??

python/c_interface/conv/eqn/dae1.c
Equation conversion for dae1 function handles ??

matlab/c_interface/conv/eqn/dae2.c
Equation conversion for so1 function handles ??

python/c_interface/conv/eqn/dae2.c
Equation conversion for dae2 function handles ??

DartConfiguration.tcl ??

DCPX_T.c ??

decomp.c
Decompose a Matrix in symmetric (hermitian) and skewsymmetric (skewhermitian) part ??

default_rom_chooser.c
Default adaptiv ROM dimension chooser ??

dense.c
Kernels for $y \leftarrow Ax + y$ in MESS_DENSE storage ??

dense_nm.c
Various implementations of the dense Newton method for the algebraic Riccati Equation ??

dense_nm_gmpare.c
Interface to mess_dense_nm_gmpare ??

dfs.c
Depth-first-search on a graph and reachability set ??

dgeqp3.c
Interface for `DGEQP3` ??

matrix/diag.c
Operations on the diagonal elements of a matrix ??

precond/diag.c
Diagonal/Jacobi preconditioner ??

direct.c
The ultimate direct solver interface ??

direct.h
Direct linear systems solvers and single-pattern multi-value LU ??

direct_chol_select.c
Interface to mess_direct_chol_select ??

direct_select.c
Interface to mess_direct_lu_select ??

doxy_main.c ??

dupl.c
Remove duplicate entries and zero entries from sparse storage schemes ??

dynorm.c
Norm computation of a dyadic matrix product ??

dynsys.h
Basic functions for dynamical systems ??

dynsysinit.c
Initialize dynamical systems ??

easyfrontend.h
Easy frontends for several functions ??

eig.c
Wrapper around various eigenvalue computations from LAPACK ??

eig_hessenberg.c
Eigenvalue computations for upper hessenberg matrices from LAPACK ??

eigenvalue.h
Interface for Eigenvalue computation ??

enum_conversions.c ??

lib/misc/eps.c
Compute machine precision ??

python/c_interface/functions/eps.c
Interface to mess_eps ??

equal.c
Operations on the pattern of matrices and norm of the difference of matrices ??

equation.c
Manage the mess_equation structure ??

equation_apply.c
Wrapper for mess_equation function pointers ??

equation_glyap.c
Build an equation object (generalized Lyapunov or Riccati Equation) ??

equation_glyap_dae1.c
Function handles for Index 1 DAE System ??

equation_glyap_dae2.c
Function handles for Hessenberg Index 2 DAE System ??

equation_glyap_so1.c
Build an equation object (generalized Lyapunov or Riccati Equation) out of a second order system ??

equation_glyap_so2.c
Build an equation object (generalized Lyapunov or Riccati Equation) out of a second order system ??

equation_lyap.c
Build an equation object (Lyapunov or Riccati Equation) ??

equation_stable.c
Stabilize a given Lyapunov Equation with matrices U and V, i.e. build an operator $(A - UV^T)$??

equations.c
Handle different equation types for `Python` interface ??

error.c
Error level handling ??

error_macro.h
Interface to error macros ??

errors.h
Interface to error codes ??

eye.c
Generate identity matrices ??

fdm_matrix.c
Generate simple FDM matrices on the unit square ??

feature_tests.c ??

feature_tests.cxx ??

fill.c
Fill up symmetry in coordinate matrices ??

filter.c
Filter functions of mess_vector structures ??

filter2D.C ??

fortran_translate.h ??

freelist.c
Memory manager ??

freelist.h
A structure and manage allocation and free of `M.E.S.S.` structures ??

galerkin.c
Galerkin projection for the mess_lrnm method ??

gaxpy.c
Compute the matrix-vector product $y \leftarrow y + op(A)x$??

getcol.c
Extract or set a column in a matrix ??

getelement.c
Read access to matrix elements ??

getrow.c
Extract or set a row out of a matrix ??

getset.c
Get/set of mess_vector structures ??

gettypes.c
Get human readable string from type identifiers ??

gglrsrm.c
GS-LR-SRM balanced truncation ??

glyap.c
Interface to mess_glyap ??

glyap3.c
Generate a solver for dense Lyapunov and Stein equations based on GLYAP3 ??

glyap_test.c
Check the solution of generalized Lyapunov Equations ??

gmres.c
Solve a linear equation with GMRES ??

gslrsrm.c
GS-LR-SRM balanced truncation ??

gstein.c
Interface to mess_gstein ??

gstein_test.c
Check the solution of generalized Stein Equations ??

h2.h
All functions for $\mathcal{H}_2$ model order reduction and balanced truncation ??

h2_norm.c
Compute the $\mathcal{H}_2$-norm of a LTI system ??

h2error.c
Compute the $\mathcal{H}_2$-error between two dynamical systems ??

hessenberg_lu.c
Generate a dense LAPACK based solver for upper Hessenberg Matrices ??

ilu0.c
Generate an $ILU(0)$ preconditioner ??

iluk.c
Generate an $ILU(k)$ preconditioner based on [27] ??

matrix/init.c
Basic management of mess_matrix structures ??

misc/init.c
Initialize M.E.S.S. ??

vector/init.c
Init function of mess_vector structures ??

interface_cholmod.h
Interface of conversion from a mess_matrix or a mess_vector to a CHOLMOD datastructure (and vice versa) ??

interface_csparse.h
Interface to CSparse. mess.h and cs.h from CSparse should be included in the program already ??

interface_matlab.c ??

interface_matlab.h
Interface to MATLAB ??

inverse.c
Check the computation of the inverse of a matrix ??

irka_biorth.c
IRKA algorithm (biorthonormal version) ??

irka_common.c
Common functions for IRKA ??

irka_para_init.c
Initialize parameters for IRKA ??

jacobi.c
Solve a linear equation with Jacobi iteration ??

kron.c
Kronecker product function of mess_vector structures ??

lanczos_template_nrm.c
The Lanczos iteration for 2-norm computation (real symmetric case) ??

lapack.c
Generate a dense LAPACK based LU-solver ??

lapack_qr.c
Generate a LAPACK based solver based on QR decomposition ??

lift.c
Add a lower trailing block zeros to a matrix ??

linsolve.c
Check the solution of a linear system of equations ??

linsolve_multi.c
Check the solution of a shifted linear system of equations ??

linsolvem.c
Check the solution of a linear system of equations (matrix version) ??

linsolvem_multi.c
Check the solution of a shifted linear system of equations (matrix version) ??

linspace.c
Linspace functions of mess_vector structures ??

lib/lrcf_adi/lradi.c
The template formulation of the LRCF-ADI ??

python/c_interface/functions/lradi.c
Interface to mess_lradi ??

lrcf_adi.h
LRCF-ADI related functions and data structures ??

lib/lrcf_adi/lrnm.c
Implementation of the Newton method ??

python/c_interface/functions/lrnm.c
Interface to mess_lrnm ??

lrnm_linesearch.c
Implementation of the linesearch function for the Newton method ??

lrsrm.c
LR-SRM balanced truncation ??

lrsrm_somor.c
LR-SRM balanced truncation ??

lrsrm_somor_so.c
LR-SRM balanced truncation for second order LTI systems to second order LTI systems ??

lu.c
Reuse kernel for sparse LU decomposition ??

lib/easyfrontend/lyap.c
Easy to use Lyapunov solver interface ??

python/c_interface/functions/lyap.c
Interface to mess_lyap ??

lyap_res2.c
Compute the 2-norm residual of a Lyapunov equation ??

lyap_sgn_fac.c
Solve a Lyapunov Equation using the sign function method ??

malloc.c
Alloc functions for C–M.E.S.S. ??

matrix/map.c
Map a scalar function to a matrix ??

vector/map.c
Map functions of mess_vector structures ??

matrix.c
Convert a Python matrix to a mess_matrix or vice-versa ??

matrix.h
Interface to handle matrices ??

matrix_print.c
Print matrices and underlying information to the standard output ??

memsize.c
Memory size function of mess_vector structures ??

mess.h
Main M.E.S.S. include file ??

mess_call.c ??

mess_classname_from_mexmess.c ??

MESS_direct.cc ??

MESS_direct.h ??

MESS_direct_install.cc ??

MESS_enum.cc ??

mess_enum_converter.c ??

MESS_enums.cc ??

MESS_enums.h ??

MESS_equation.cc ??

MESS_equation.h ??

mess_equation_from_mexmess.c ??

MESS_equation_install.cc ??

MESS_error.h ??

MESS_HAVE_PTRDIFF_T.c ??

MESS_HAVE_SIZE_T.c ??

mess_matrix.c ??

cpp_interface/MESS_matrix/MESS_matrix.cc ??

octmess/src/MESS_matrix.cc ??

MESS_matrix.h ??

MESS_matrix_install.cc ??

MESS_matrix_ops_mm_generator.h ??

MESS_matrix_ops_mm_mm.cc ??

MESS_matrix_ops_mm_ocm.cc ??

MESS_matrix_ops_mm_om.cc ??

MESS_matrix_ops_mm_os.cc ??

MESS_matrix_ops_mm_osc.cc ??

MESS_matrix_ops_mm_oscm.cc ??

MESS_matrix_ops_mm_osm.cc ??

MESS_misc.cc ??

MESS_multidirect.cc ??

MESS_multidirect.h ??

MESS_multidirect_install.cc ??

mess_options.c ??

cpp_interface/MESS_options/MESS_options.cc ??

octmess/src/MESS_options.cc ??

MESS_options.h ??

MESS_options_install.cc ??

mess_set_errorlevel.c
Interface to mess_set_errorlevel ??

mess_status.c ??

cpp_interface/MESS_status/MESS_status.cc ??

octmess/src/MESS_status.cc ??

MESS_status.h ??

MESS_status_install.cc ??

MESS_to_oct.cc ??

mess_vector.c ??

cpp_interface/MESS_vector/MESS_vector.cc ??

octmess/src/MESS_vector.cc ??

MESS_vector.h ??

MESS_vector_install.cc ??

MESS_vector_ops_mm_mv.cc ??

MESS_vector_ops_mv_generator.h ??

MESS_vector_ops_mv_mv.cc ??

MESS_vector_ops_mv_os.cc ??

MESS_vector_ops_mv_osc.cc ??

MESS_vector_ops_ocm_mv.cc ??

MESS_vector_ops_om_mv.cc ??

MESS_vector_ops_oscm_mv.cc ??

MESS_vector_ops_osm_mv.cc ??

mess_version.c
Interface to mess_version ??

mex_mess_care.c ??

mex_mess_dense_nm_gmpare.c ??

mex_mess_glyap.c ??

mex_mess_gstein.c ??

mex_mess_lradi.c ??

mex_mess_lrnm.c ??

mex_mess_lyap.c ??

mex_mess_sylvester_dense.c ??

mex_mess_sylvester_sparsedense.c ??

mex_test_matrix.c ??

mex_test_options.c ??

mex_test_status.c ??

mex_test_vector.c ??

mgs.c
The modified Gram-Schmidt algorithm ??

mgsadd.c
Add columns to an orthogonal matrix ??

minmax.c
MinMax functions of mess_vector structures ??

lib/misc/misc.c
Miscellaneous functions (print memory, round value) ??

matlab/c_interface/misc/misc.c ??

misc.h
Miscallenous functions ??

mklpardiso.c
Interface to MKL-PARDISO ??

mult.c
Matrix-matrix multipication ??

multidirect.c
Generic interface for the multi-LU / multisolvers ??

multidirect_select.c
Interface to mess_multidirect_select ??

multilu_adavanced.c
Multisolver implementation with mess lu ??

multilu_umfpack.c
UMFPACK multisolver interface ??

mvp.c
Compute the matrix-vector product $y = Ax$??

mvpcall.c
Handling generic matrix-vector products ??

mxIsSubClass.c ??

symbols/MY_MODULE_mp_MY_SUB-UPPER.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/MY_MODULE_mp_MY_SUB-UPPER.c ??

symbols/my_module_MP_my_sub.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/my_module_MP_my_sub.c ??

symbols/my_module_mp_my_sub_.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/my_module_mp_my_sub_.c ??

symbols/my_module_my_sub_.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/my_module_my_sub_.c ??

symbols/my_moduleSmy_sub.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/my_moduleSmy_sub.c ??

symbols/MY_SUB-UPPER.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/MY_SUB-UPPER.c ??

symbols/my_sub.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/my_sub.c ??

symbols/my_sub_.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/my_sub_.c ??

symbols/my_sub__.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/my_sub__.c ??

symbols/MYMODULE_mp_MYSUB-UPPER.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/MYMODULE_mp_MYSUB-UPPER.c ??

symbols/mymodule_MP_mysub.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/mymodule_MP_mysub.c ??

symbols/mymodule_mp_mysub_.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/mymodule_mp_mysub_.c ??

symbols/mymodule_mysub_.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/mymodule_mysub_.c ??

symbols/mymoduleSmysub.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/mymoduleSmysub.c ??

symbols/MYSUB-UPPER.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/MYSUB-UPPER.c ??

symbols/mysub.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/mysub.c ??

symbols/mysub_.c ??

VerifyC/CMakeFiles/FortranCInterface/symbols/mysub_.c ??

newlu.c
Easy left looking sparse LU ??

matrix/norm.c
Compute various norms of a matrix ??

vector/norm.c
Norm functions of mess_vector structures ??

offsets.c
Return offsets of structures defined in M.E.S.S. ??

matrix/ones.c
Generate a matrix containing one number over and over again ??

vector/ones.c
Fill functions of mess_vector structures ??

openmp.c
OpenMP utilities ??

lib/dynsys/bt/options.c
Options for balance truncation ??

lib/lrcf_adi/options.c
LRCF-ADI options ??

python/c_interface/conv/options.c
Convert a Python options class to mess_options and vice-versa ??

orth.c
Orthogonalize a matrix ??

para.c
Parameter compuation for ADI processes ??

matrix/perm.c
Different matrix permutation operations ??

vector/perm.c
Permutation functions of mess_vector structures ??

pinv.c
Compute Pseudoinverse of a dense matrix ??

plot_scriptExporter.c
A small plot script creator ??

plot_scriptExporter.h
Interface definition of the Gnuplot module ??

plot_x11.h
Interface definition of the plotting module ??

precond.c
Preconditioner interface ??

print.c ??

proj_sym.c
Projection of a matrix on symmetric matrix space ??

project.c
Project a system ((G)LTI, second order) to another system ((G)LTI, second order) ??

pymess.c
Functions which should be called from `Python` ??

pymess.h
Interface to `Python` ??

rail_371_c60.C ??

matrix/rand.c
Generate different random matrices ??

vector/rand.c
Rand functions of mess_vector structures ??

rank.c
Estimate the rank of a matrix ??

rc2norm.c
Compute the 2-norm difference between two factored solutions of a matrix equation ??

rcFnorm.c
Frobenius-norm difference between factored solutions ??

read.c
Read matrices from files ??

real_complex.c
Realpart Imagpart functions of mess_vector structures ??

realimag.c
Handle real and imaginary parts of complex matrices ??

reimplemented.c
Check if instance is reimplemented ??

direct/res2/res2.c
2-norm residual of $r = Ax - b$ and similar equations ??

lrcf_adi/norms/res2.c
2 Norm residual computation for the Lyapunov and the Riccati operator ??

res2_template.c
New LRCF-ADI res2 implementation for the eqn structure ??

scal.c
Scale a matrix ??

scale.c
Scale functions of mess_vector structures ??

schur.c
Compute the Schur decomposition of a matrix ??

schur_to_evd.c
Computes Spectral Decomposition from Schur decomposition ??

select.c
Select reordering by name ??

setelement.c
Set an element in a matrix ??

shellsort.c
Compact shellsort implementation for matrix data structures ??

sign.c
Implementation of a basic sign function iteration ??

matlab/c_interface/conv/eqn/so1.c
Equation conversion for so1 function handles ??

python/c_interface/conv/eqn/so1.c
Equation conversion for so1 function handles ??

matlab/c_interface/conv/eqn/so2.c
Equation conversion for so1 function handles ??

python/c_interface/conv/eqn/so2.c
Equation conversion for so2 function handles ??

solver.h
Interface to all iterative linear system solvers ??

solvername.h
Structure of solvernames ??

itsolver/sor.c
Solve $Ax = b$ with SOR and similar algorithms ??

precond/sor.c
Generate a SOR based preconditioner ??

sort.c
Sort the data structures inside a matrix ??

splsolve.c
Efficient solution with sparse lower trinagular matrices ??

spy.c
Generate spy plots of matrices ??

stable.c
Check if a dynamical system is stable or not ??

lib/dynsys/bt/status.c
Work with the mess_bt_status objects ??

lib/itsolver/status.c
Status / options object for iterative solvers ??

lib/lrcf_adi/status.c
LRCF-ADI status ??

python/c_interface/conv/status.c
Convert a mess_status structure to a python status object ??

statusopt.c
Status and options for $\mathcal{H}_2$ things ??

matlab/c_interface/conv/eqn/std.c
Equation conversion for dae1 function handles ??

python/c_interface/conv/eqn/std.c
Equation conversion for standard lyapunov riccati equation ??

stein_res2.c
Compute the 2-norm residual of a Stein equation ??

sub.c
Get a submatrix from a matrix ??

superlu.c
Interface to SuperLU ??

superlu_refinement_bug.c ??

superlu_refinement_bug_43.c ??

superlu_refinement_bug_50.c ??

svd.c
Compute the singular value decompositon (SVD) ??

sylv_res2.c
Residual computations for dense Sylvester equations ??

lib/direct/singlesolver/sylvester_dense.c
Generate a solver for dense Sylvester eqautions based on LAPACK `dtrsyl` / `ztrsyl` / `dtgsyl` / `ztgsyl` ??

tests/direct/sylvester_dense.c
Check the solution of dense Sylvester equations ??

lib/direct/singlesolver/sylvester_sparsedense.c
Generate a solver for a sparse-dense Sylvester equation with a small second matrix ??

python/c_interface/functions/sylvester_sparsedense.c
Interface to mess_direct_create_sylvester_sparsedense ??

tests/direct/sylvester_sparsedense.c
Check the solution of a sparse-dense Sylvester equations with a small second matrix ??

test.c
Test functions which can be called from Python ??

TestRestrict.c ??

threadpool.c
Basic functions for a loader pool ??

threadpool.h
Interface to handle a pool of threads ??

time.c
Functions for timers and time measuring ??

trace.c
Compute the trace of a matrix ??

transfer.c
Evalute the transfer function ??

transpose.c
Transpose a matrix ??

tridiag.c
Generate tridiagonal matrices ??

trisolve.c
Solving with triangular systems ??

triul.c
Upper triangular part of Matrix Operations ??

tsia.c
TSIA algorithm ??

tsiag.c
Generalized TSIA Algorithm ??

tutorial_backslash.c
Demostrate how to solve a linear system using a \-operator ??

tutorial_bt.c ??

tutorial_bt_gglrsrm.c
Demonstrate how to reduce a first order GLTI system using a generalized low-rank square root balanced truncation (BT-GGLRSRM) ??

tutorial_bt_gslrsrm.c
Demonstrate how to reduce a first order GLTI system to a first order LTI system using a generalized low-rank square root balanced truncation (BT-GSLRSRM) ??

tutorial_bt_lrsrm.c
Demonstrate how to reduce a first order LTI system using a low-rank square root balanced truncation (BT-LRSRM) ??

tutorial_bt_lrsrm_somor.c
Demonstrate how to reduce a second order LTI system to a first order LTI system using a low-rank square root balanced truncation (BT-LRSRM-SOMOR) ??

tutorial_bt_lrsrm_somor_so.c
Demonstrate how to reduce a second order LTI system to another second order LTI system using a low-rank square root balanced truncation (BT-LRSRM-SOMOR-SO) ??

tutorial_care.c
Tutorial for solving a (generalized) Riccati Equation using mess_care solver ??

tutorial_direct_solvers.c
Tutorial about linear equation systems ??

tutorial_eig.c
Tutorial for demonstrating dense eigenvalue and svd decomposition using mess_eigen_eig and mess_eigen_svd ??

tutorial_eigs.c
Demonstrate how to approximate some eigenvalues of a matrix ??

tutorial_gen_fdm_cd.c
Demonstrate how to discretize a convection-diffusion operator on the unit square ??

tutorial_gen_laplace.c
Demonstrate how to discretize a convection-diffusion operator on the unit square ??

tutorial_gschur.c
Demonstrate how to compute the generalized Schur decomposition of a matrix pair (A, B) ??

tutorial_h2norm.c
Demonstrate how to compute the $\mathcal{H}_2$-norm of a LTI system ??

tutorial_irka_biorth.c
Demonstrate the IRKA algorithm ??

tutorial_it_unsym.c
Demonstrate how to use various iterative solvers for unsymmetric systems of linear equations ??

tutorial_lradi.c
Tutorial for solving a (generalized) Lyapunov Equation using mess_lradi solver ??

tutorial_lrnm-g.c
Demonstrate how to solve a generalized algebraic Riccati Equation using LRCF-ADI-NM ??

tutorial_lrnm.c
Tutorial for solving a (generalized) Riccati Equation using mess_lrnm solver ??

tutorial_lyap.c
Tutorial for solving a (generalized) Lyapunov Equation using mess_lyap solver ??

tutorial_matrix_operations.c
Tutorial on matrix operations ??

tutorial_norms.c
Demonstrate how to compute different matrix norms with different storage formats ??

tutorial_plot.c
Tutorial for the plotting interface and plot the sine and cosine function ??

tutorial_qr.c
Demonstrate how to use a QR solver ??

tutorial_schur.c
Demonstrate how to compute the (complex) Schur decomposition of a matrix ??

tutorial_sign.c
Tutorial for demonstrating how to compute the sign of a matrix using mess_eigen_sign ??

tutorial_spy.c
Tutorial for demonstrating the mess_matrix_spy function ??

tutorial_sylvester.c
Demonstrate how to solve a sparse-dense Sylvester equation ??

tutorial_tsia.c
Demonstrate how to reduce a LTI system using the TSIA algorithm ??

tutorial_vector_read.c
Demonstrate a simple input/ouput example for vectors (generate random vectors, read, write, print) ??

twoside_mgs.c
Skew projection ??

umfpack.c
Interface to UMFPACK ??

umfpack_control.c
Interface to UMFPACK ??

vector.c
Convert a Python matrix/array to a C matrix/array or vice-versa ??

vector.h
Interface to handle vectors ??

vector_print.c
Print mess_vector instance and underlying information to the standard output ??

vector_read.c
Read a vector from a matrix market file ??

VerifyFortran.h ??

wachspress.c
Auxiliary routines for Wachspress parameter computation ??

write.c
Write matrices to files ??

x11init.c
A small plotter interface ??

xerbla.c
New error handler for LAPACK ??

8 Module Documentation

8.1 General Type Definitions and Macros

This section contains all generic macros and type definitions.

Macros

- #define DEPRECATED __attribute__ ((deprecated))
- #define MESS_PRINTF_INT "%d"

  String to print a mess_int_t inside printf.
- #define MESS_PRINTF_INT2D "%2d"

  String to print a mess_int_t with 2 digits inside printf.
- #define MESS_INTEGER MESS_REAL

  Define for compatibility with the MatrixMarket file format.
- #define MESS_IS_REAL(m) ((*m).data_type == MESS_REAL)

  Check if a mess_matrix, mess_vector or mess_direct object is real.
- #define MESS_IS_INTEGER(m) ((*m).data_type == MESS_REAL)

  Check if a mess_matrix, mess_vector or mess_direct object is real.
- #define MESS_IS_COMPLEX(m) ((*m).data_type == MESS_COMPLEX)

  Check if a mess_matrix, mess_vector or mess_direct object is complex.

Typedefs

- typedef double _Complex mess_double_cpx_t

  Default double complex of ***M.E.S.S.***.

- typedef float _Complex mess_float_cpx_t

  Default float complex of ***M.E.S.S.***.

- typedef int mess_int_t

  Default integer of ***M.E.S.S.***.

- typedef unsigned short MESS_STORETYPE

  Type definition for the data type in ***M.E.S.S.***.

- typedef unsigned short MESS_SYMMETRY

  Type definition for the data type in ***M.E.S.S.***.

Enumerations

8.1.1 Detailed Description

One of the important ones are the mess_int_t type definition. Per default this point to a normal `int` integer. If **M.E.S.S.** is configured for 64 Bit integers, i.e. `MESS64` is set to during the configure step, the mess_int_t type points to `long` or `int64_t`.

Another important data type is the mess_operation_t enumeration. It defines how routines like mess_matrix_mvp or mess_matrix_multiply operate .

8.1.2 Macro Definition Documentation

8.1.2.1 #define DEPRECATED __attribute__ ((deprecated))

8.1.2.2 #define MESS_INTEGER MESS_REAL

The MESS_INTEGER define is only included for compatibility reasons with the `MatrixMarket` file format.
It should not be used in programs.

See also

mess_datatype_t

8.1.2.3 #define MESS_IS_COMPLEX(m) ((*m).data_type == MESS_COMPLEX)

The MESS_IS_COMPLEX macro is used to check easily for a complex mess_matrix, mess_vector or mess_direct.

8.1.2.4 #define MESS_IS_INTEGER(m) ((*m).data_type == MESS_REAL)

The MESS_IS_INTEGER is a wrapper around MESS_IS_REAL.
It exists for compatibility reasons with the `MatrixMarket` file format.

8.1.2.5 #define MESS_IS_REAL(m) ((*m).data_type == MESS_REAL)

The MESS_IS_REAL macro is used to check easily for a real mess_matrix, mess_vector or mess_direct.

8.1.2.6 #define MESS_PRINTF_INT "%d"

The MESS_PRINTF_INT is used to display a mess_int_t inside `printf` and similar functions.

See also

mess_int_t

8.1.2.7 #define MESS_PRINTF_INT2D "%2d"

The MESS_PRINTF_INT2D is used to display a mess_int_t inside `printf` and similar functions with 2 digits.

See also

mess_int_t

8.1.3 Typedef Documentation

8.1.3.1 typedef double _Complex mess_double_cpx_t

The mess_double_cpx_t typedef is used to avoid incompatability problems when including **M.E.S.S.** from C++. C++ treats complex is a template.

8.1.3.2 typedef float _Complex mess_float_cpx_t

The mess_float_cpx_t typedef is used to avoid incompatability problems when including **M.E.S.S.** from C++. C++ treats complex is a template.

8.1.3.3 typedef int mess_int_t

The mess_int_t typedef is used to make **M.E.S.S.** independent from the size of the operating systems integer. It allows to change between 32 and 64 integers without rewriting code. Only the return values of function are always 32bit integers.
Per default the mess_int_t is equal to `int`.
If **M.E.S.S.** is configured with `MESS64=ON` than it is equal to `int64_t`.

8.1.3.4 typedef unsigned short MESS_STORETYPE

The MESS_STORETYPE is used to represent the storage type flag inside mess_matrix and mess_vector.

8.1.3.5 typedef unsigned short MESS_SYMMETRY

The MESS_SYMMETRY is used to represent the symmetry type flag inside mess_matrix.

8.1.4 Enumeration Type Documentation

8.1.4.1 enum mess_datatype_t

The mess_datatype_t enumeration defines the possible data types inside **M.E.S.S.**
The mess_datatype_t is used inside mess_matrix, mess_vector and mess_direct.

Enumerator

MESS_REAL The values of the object are real.

MESS_COMPLEX The values of the object are complex.

8.1.4.2 enum mess_norm_t

The mess_norm_t enumeration is used to represent the type of norm.

Enumerator

MESS_2_NORM The 2-norm.

MESS_FROBENIUS_NORM The Frobenius-norm.

MESS_1_NORM The 1-norm, maximum absolute column sum norm.

MESS_INF_NORM The inf-norm, maximum absolute row sum norm.

8.1.4.3 enum mess_operation_t

The mess_operation_t enumeration is used to define how a subroutine acts on a mess_matrix.

Enumerator

MESS_OP_NONE mess_matrix object is applied non-transposed.

MESS_OP_TRANSPOSE mess_matrix object is applied transposed.

MESS_OP_HERMITIAN mess_matrix object is applied hermitian transposed.

8.1.4.4 enum mess_print_format_t

The mess_print_format_t enumeration is used to control the output of print functions.

Enumerator

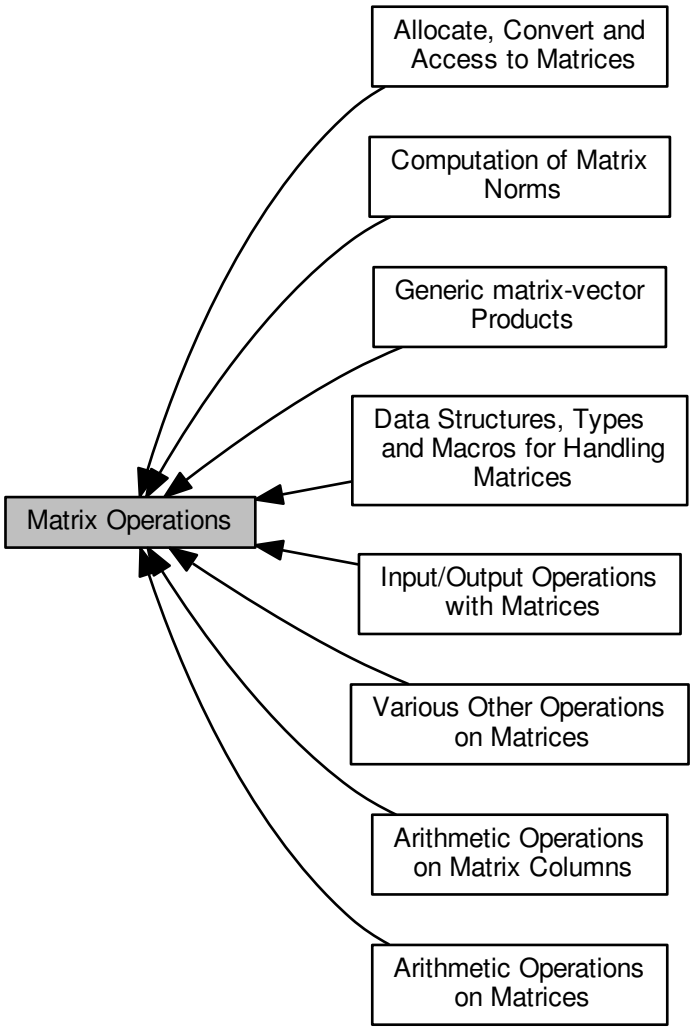
MESS_PRINT_FORMAT_SHORT Short scientific notation with 6 digits after decimal point.

MESS_PRINT_FORMAT_LONG Long scientific notation with 15 digits after decimal point.

8.2 Matrix Operations

This section contains all matrix related operations.

Collaboration diagram for Matrix Operations:



Modules

- Arithmetic Operations on Matrices

  This category contains all basic arithmetic operations with matrices.
- Arithmetic Operations on Matrix Columns

  This category contains various arithmetic operations on matrix columns.
- Computation of Matrix Norms

  This category collects all matrix norm computations and related functions like the condition number or its estimate.
- Allocate, Convert and Access to Matrices

This category contains all subroutines which act on the matrix storage type, data structures and data management.

- Input/Output Operations with Matrices

  This category contains all functions to read, write and print matrices.

- Various Other Operations on Matrices

  This category contains all matrix related functions that fit not well in one of the other categories.

- Data Structures, Types and Macros for Handling Matrices

  This category contains all matrix related data structures, macros and type definitions.

- Generic matrix-vector Products

  Representation of a generic matrix-vector Product.

8.2.1 Detailed Description

This includes the management of the mess_matrix data structure, modifying the structure of the mess_matrix and arithmetic operations. Additionally it includes an input/output interface to read and write `MatrixMarket` files.

Matrices are represented as an instance of a mess_matrix object. Before they can be used the first time they must be initialized using mess_matrix_init and destroyed afterwards using mess_matrix_clear.

```
mess_matrix A;
mess_matrix_init(&A);
// do some stuff with A
mess_matrix_clear(&A);
```

A mess_matrix can handle real and complex matrices and the following storage formats:

- Dense (Fortran) storage (MESS_DENSE)
- Compressed Sparse Row storage (MESS_CSR)
- Compressed Sparse Column storage (MESS_CSC)
- Coordinate storage (MESS_COORD)

Most functions can handle all of them or if not convert them automatically to the desired format. If such an automatic convertation is done a warning is displayed to the standard error ouput. Functions that need a internal copy of the matrix copy them to the right format as well without such a warning. This can cause memory problems when you pass a dense matrix to the SVD or a eigenvalue solver. You can convert matrices between all formats using the mess_matrix_convert function.

Matrices can be created by loading them from a `MatrixMarket` file using mess_matrix_read or mess_matrix↩_read_formated. Otherwise they can be created out of the corresponding data structures using mess_matrix_↩dense_from_farray, mess_matrix_dense_from_carray, mess_matrix_csr, mess_matrix_csc or mess_matrix_coord. A not recommended element-wise access is possible using mess_matrix_getelement, mess_matrix_setelement and mess_matrix_setelement_complex. Otherwise you can access the elements in mess_matrix directly but keep in mind that this structure can change sometimes.

Beside normal matrix operations like the matrix-vector product, there exists many operation that work on matrix columns. They are available in the Arithmetic Operations on Matrix Columns section.

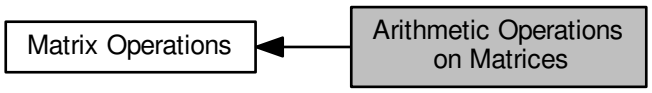
See also

Arithmetic Operations on Matrices
Arithmetic Operations on Matrix Columns
Computation of Matrix Norms
matrix_data
Input/Output Operations with Matrices
Various Other Operations on Matrices
Data Structures, Types and Macros for Handling Matrices
Matrix Generators

8.3 Arithmetic Operations on Matrices

This category contains all basic arithmetic operations with matrices.

Collaboration diagram for Arithmetic Operations on Matrices:



Functions

- int mess_matrix_ctranspose (mess_matrix A, mess_matrix AH)

  Transpose a matrix.
- int mess_matrix_transpose (mess_matrix A, mess_matrix AT)

  Transpose a matrix.
- int mess_matrix_xtranspose (mess_matrix A, mess_matrix AT, int hermitian)

  Transpose a matrix.
- int mess_matrix_conj (mess_matrix A)

  Conjugate a matrix.
- int mess_matrix_proj_sym (mess_matrix matrix)

  Get projection of a matrix on symmetric matrix space.
- int mess_matrix_rank (mess_matrix A, mess_int_t *rank)

  Estimate the rank of a matrix.
- int mess_matrix_orth (mess_matrix A, mess_matrix Q)

  Compute an orthogonal basis for the range of a matrix.
- int mess_matrix_null (mess_matrix A, mess_matrix Z)

  Compute an orthogonal basis for the nullspace of a matrix.
- int mess_matrix_mgs (mess_matrix A, mess_matrix Q, mess_matrix R)

  Perform a modified Gram-Schmidt orthogonalization.
- int mess_matrix_mgs_inplace (mess_matrix Q)

  Perform a modified Gram-Schmidt orthogonalization inplace on a matrix.
- int mess_matrix_mgs_add (mess_matrix Z, mess_matrix V, mess_matrix E)

  Use the modified Gram-Schmidt orthogonalization to add new columns to an orthogonal matrix.
- int mess_matrix_mvp (mess_operation_t op, mess_matrix A, mess_vector x, mess_vector y)

  Compute a matrix-vector product.
- int mess_matrix_gaxpy (mess_operation_t op, mess_matrix A, mess_vector x, mess_vector y)

  Compute a matrix-vector product with update of the output.
- int mess_matrix_colsums (mess_matrix A, mess_vector x)

  Compute the column sums of a matrix.
- int mess_matrix_rowsums (mess_matrix A, mess_vector x)

  Compute the row sums of a matrix.
- int mess_matrix_add (double alpha, mess_matrix A, double beta, mess_matrix B)

  Add two real matrices.

- int mess_matrix_addc (mess_double_cpx_t alpha, mess_matrix A, mess_double_cpx_t beta, mess_matrix B)

  Add two matrices.
- int mess_matrix_multiply (mess_operation_t opA, mess_matrix A, mess_operation_t opB, mess_matrix B, mess_matrix C)
- int mess_matrix_scale (double alpha, mess_matrix mat)

  Scale a real matrix by a scalar.
- int mess_matrix_scalec (mess_double_cpx_t alpha, mess_matrix mat)

  Scale a complex matrix by a scalar.
- int mess_matrix_fro_inn (mess_operation_t op, mess_matrix A, mess_matrix B, void ∗fro)

  Compute frobenius inner product of two dense matrices.
- int mess_matrix_trace (mess_matrix A, double ∗tr)

  Compute the trace of a real matrix.
- int mess_matrix_tracec (mess_matrix A, mess_double_cpx_t ∗tr)

  Compute the trace of a complex matrix.
- int mess_matrix_biorth (mess_matrix Vin, mess_matrix Win, mess_matrix Vout, mess_matrix Wout)

  Construct two biorthonormal matrices from two given matrices.
- int mess_matrix_gbiorth (mess_matrix E, mess_matrix Vin, mess_matrix Win, mess_matrix Vout, mess_matrix Wout)

  Construct two biorthonormal matrices from two given matrices with respect to a weighted scalar product.
- int mess_matrix_proj_mgs (mess_vector in, mess_matrix Q, mess_matrix QT, mess_int_t colcount, mess_vector out, double ∗nrmout)

  Skew projection using Q and QT.
- int mess_matrix_proj_gmgs (mess_vector in, mess_matrix Q, mess_matrix QT, mess_matrix E, char ∗Eflag, mess_int_t colcount, mess_vector out, double ∗nrmout)

  Generalized skew projection using Q and QT.
- int mess_matrix_map (mess_matrix mat, double(∗f_real)(double), mess_double_cpx_t(∗f_cpx)(mess_double_cpx_t), int full)

  Map a scalar function to a matrix.
- int mess_matrix_map_abs (mess_matrix mat, int full)
- int mess_matrix_map_acos (mess_matrix mat, int full)
- int mess_matrix_map_acosh (mess_matrix mat, int full)
- int mess_matrix_map_arg (mess_matrix mat, int full)
- int mess_matrix_map_asin (mess_matrix mat, int full)
- int mess_matrix_map_asinh (mess_matrix mat, int full)
- int mess_matrix_map_atan (mess_matrix mat, int full)
- int mess_matrix_map_atanh (mess_matrix mat, int full)
- int mess_matrix_map_ceil (mess_matrix mat, int full)
- int mess_matrix_map_conj (mess_matrix mat, int full)
- int mess_matrix_map_cos (mess_matrix mat, int full)
- int mess_matrix_map_cosh (mess_matrix mat, int full)
- int mess_matrix_map_exp (mess_matrix mat, int full)
- int mess_matrix_map_expm1 (mess_matrix mat, int full)
- int mess_matrix_map_floor (mess_matrix mat, int full)
- int mess_matrix_map_log (mess_matrix mat, int full)
- int mess_matrix_map_not (mess_matrix mat, int full)
- int mess_matrix_map_round (mess_matrix mat, int full)
- int mess_matrix_map_sin (mess_matrix mat, int full)
- int mess_matrix_map_sinh (mess_matrix mat, int full)
- int mess_matrix_map_sqrt (mess_matrix mat, int full)
- int mess_matrix_map_tan (mess_matrix mat, int full)
- int mess_matrix_map_tanh (mess_matrix mat, int full)
- int mess_matrix_map_isfinite (mess_matrix mat, int full)

- int mess_matrix_map_isinf (mess_matrix mat, int full)
- int mess_matrix_map_isnan (mess_matrix mat, int full)
- int mess_matrix_any (mess_matrix mat, mess_int_t(∗f_real)(double), mess_int_t(∗f_cpx)(mess_double_↩cpx_t), mess_int_t dim, mess_vector anyvec)

  Checks a matrix rowwise/columnwise if a given predicate is true.

8.3.1 Detailed Description

It includes matrix-matrix products, matrix-vector products, addition of two matrices, scaling, orthogonal bases for the matrix columns and similar things. Operations on matrix columns are available in the Arithmetic Operations on Matrix Columns category. The computation of norms and related things are available in the Computation of Matrix Norms category. Most routines are able to work with sparse and dense data. For real and complex data sometimes exists two variants of a function. In the case of dense matrices we use BLAS (`NetlibBLAS`, `OpenBLAS`, `Intel MKL`, `Atlas`, `FlexiBLAS`) operations as much as possible to get the best performance out of them.

More complex operations like solving linear equations or eigenvalue problems can be found in the corresponding categories.

See also

Arithmetic Operations on Matrix Columns
Computation of Matrix Norms

8.3.2 Function Documentation

8.3.2.1 int mess_matrix_add (double *alpha*, mess_matrix *A*, double *beta*, mess_matrix *B*)

Parameters

in	*alpha*	input coefficient in front of A
in	*A*	input matrix A
in	*beta*	input coefficient in front of B
in,out	*B*	input/output matrix B

Returns

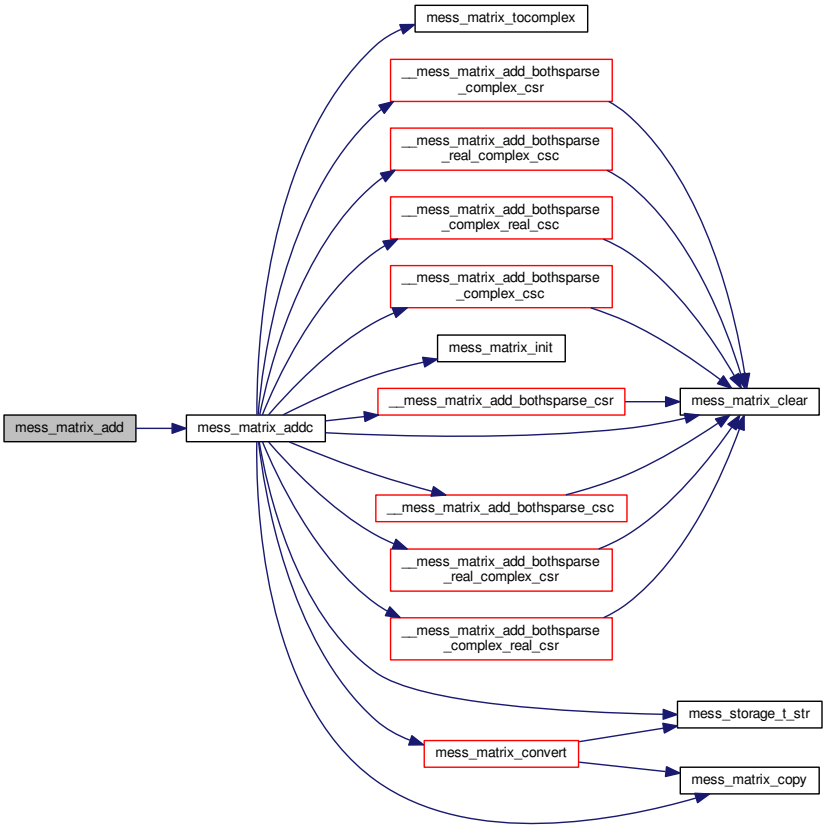
zero on success or a non-zero error value otherwise

The mess_matrix_add function add α times a mess_matrix A to β times a matrix B and store it in B.

$$B \leftarrow \alpha A + \beta B$$

If there are new zero elements they have to stay in the pattern for compatibility. The matrix must be sorted internally. This can be achieved using mess_matrix_sort. This function is only a wrapper around mess_matrix_addc, which provides an interface for real and complex matrices.

Here is the call graph for this function:



8.3.2.2 int mess_matrix_addc (mess_double_cpx_t *alpha*, mess_matrix *A*, mess_double_cpx_t *beta*, mess_matrix *B*)

Parameters

in	*alpha*	input coefficient in front of A
in	*A*	input matrix A
in	*beta*	input coefficient in front of B
in,out	*B*	input/output matrix B

Returns

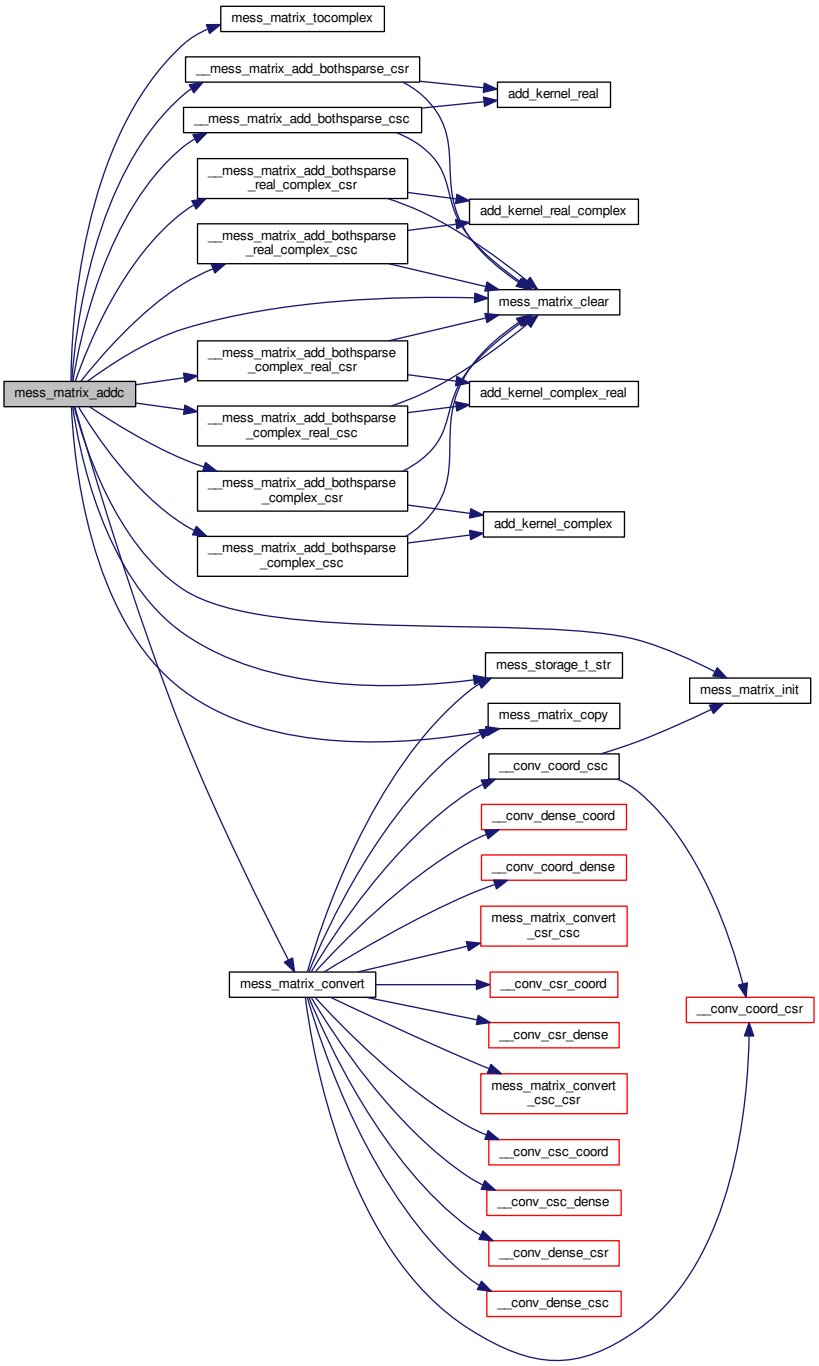
zero on success or a non-zero error value otherwise

The mess_matrix_addc function add α times a matrix A to β times a matrix B and store it in B.

$$B \leftarrow \alpha A + \beta B$$

The matrices can be real or complex. Depending on the data, the right internal algorithm is chosen. If there are new zero elements in the matrix, they will be kept due to produces the correct of the sum. If the matrices are sparse, they must have sorted internal data structures. If this is not the case, the result will be wrong. The internal data structure can be sorted using mess_matrix_sort before.

Here is the call graph for this function:



8.3.2.3 int mess_matrix_any (mess_matrix *mat*, mess_int_t(∗)(double) *f_real*, mess_int_t(∗)(mess_double_cpx_t) *f_cpx*, mess_int_t *dim*, mess_vector *anyvec*)

Parameters

in,out	*mat*	input/output matrix mat
in	*f_real*	input scalar real function to apply to each entry of mat with mess_int_t return value

Parameters

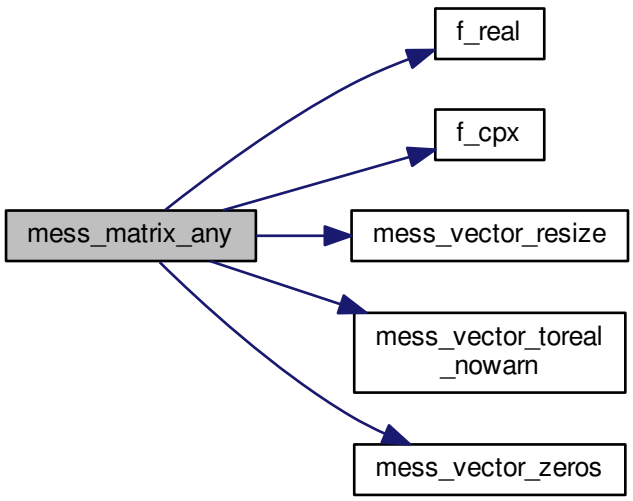
in	*f_cpx*	input scalar complex function to apply to each entry of mat with mess_int_t return value
in	*dim*	input 0 for rowise fashion 1 for columnwise fashion.
in,out	*anyvec*	input/output mess_vector

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_any function tests entries of mat with f_real or f_cpx in a rowwise/columnwise fashion.
If dim is equal to 0 rowwise fashion is used.
If dim is equal to 1 columnwise fashion is used.
If mat is MESS_REAL f_real is used.
If mat is MESS_COMPLEX f_cpx is used.
f_real and f_cpx are scalar functions with real/complex argument and mess_int_t return value.
If f_real / f_cpx applied to the entry $mat_{i,j}$ returns a nonzero value 1 is written to the j -th entry of anyvec.
Independent of the datatype of mat anyvec is in every case a MESS_REAL mess_vector.
If dim is equal to 0 the number of entries of anyvec is equal to the number of rows of mat.
If dim is equal to 1 the number of entries of anyvec is equal to the number of cols of mat.
The mess_matrix_any function works in a similiar fashion like the any function of MATLAB.

Here is the call graph for this function:



8.3.2.4 int mess_matrix_biorth (mess_matrix *Vin*, mess_matrix *Win*, mess_matrix *Vout*, mess_matrix *Wout*)

Parameters

in	*Vin*	input matrix V_{in}
in	*Win*	input matrix W_{in}
out	*Vout*	output matrix V_{out}
out	*Wout*	output matrix W_{out}

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_biorth function constructs a biorthonormal basis

$$W_{out}^T V_{out} = I$$

from input matrices V_{in} and W_{in}.

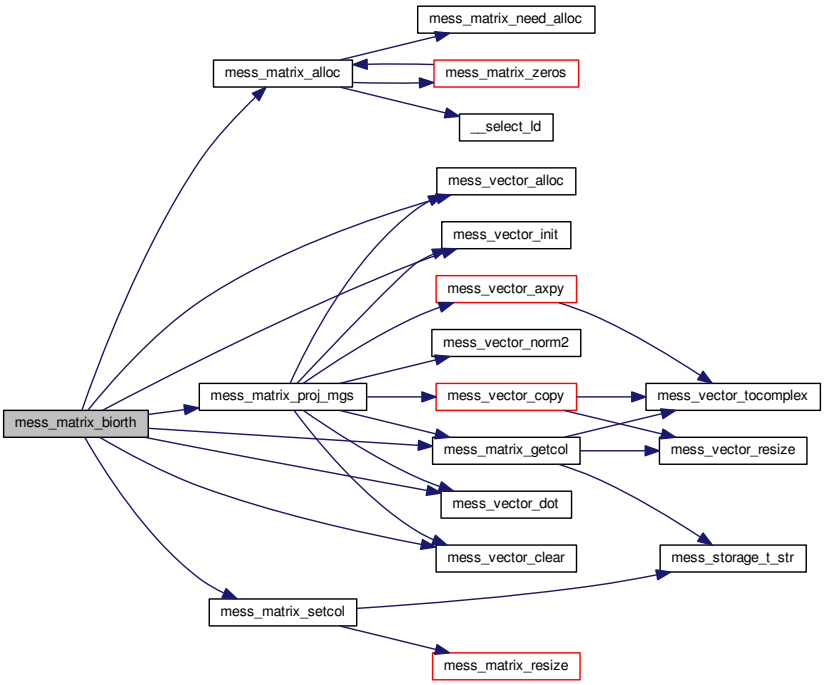
See also

mess_matrix_gbiorth

Attention

The function only works for real and dense matrices.

Here is the call graph for this function:



8.3.2.5 int mess_matrix_colsums (mess_matrix *A*, mess_vector *x*)

Parameters

in	*A*	input matrix A
out	*x*	output vector x with the column sums of A

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_colsums function computes the sum of each column of a matrix by evaluating

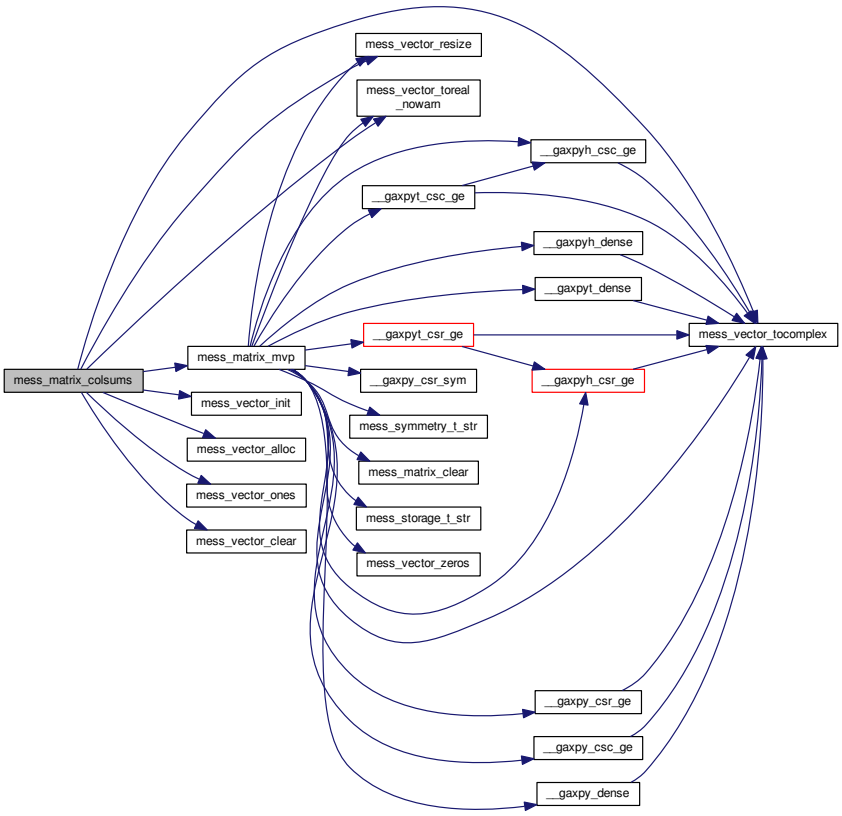
$$x = A^T e$$

where $e = [1, ..., 1]^T$.

See also

mess_matrix_rowsums

Here is the call graph for this function:



8.3.2.6 int mess_matrix_conj (mess_matrix *A*)

Parameters

in,out	*A*	input/output matrix A to conjugate

Returns

zero on success or a non zero error code otherwise

The mess_matrix_conj function congujates a complex matrix, i.e. it computes

$$A \leftarrow \bar{A}.$$

8.3.2.7 int mess_matrix_ctranspose (mess_matrix *A*, mess_matrix *AH*)

Parameters

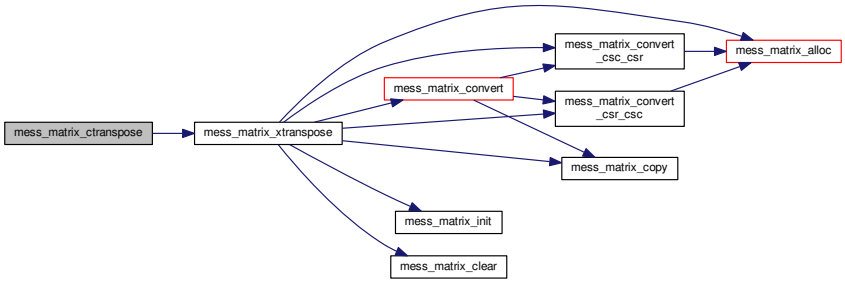
in	*A*	input matrix A
out	*AH*	output hermitian transposed matrix A^H

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_ctranspose function creates the hermitian transpose of a given matrix.
It supports all internal matrix storage formats

Here is the call graph for this function:



8.3.2.8 int mess_matrix_fro_inn (mess_operation_t *op*, mess_matrix *A*, mess_matrix *B*, void * *fro*)

Parameters

in	*op*	input operation for A (transposed / hermitian / none)
in	*A*	dense input matrix
in	*B*	dense input matrix
out	*fro*	output pointer to resulting frobenius inner product

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_fro_inn function computes the frobenius inner product of two dense matrices:

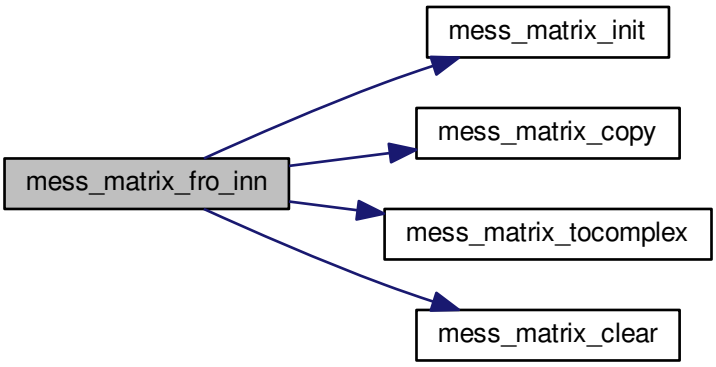
$$fro = \langle op(A), B \rangle_F = \sum_{i,j} \overline{op(A)_{i,j}} B_{i,j}$$

The output argument is a void pointer. If both input matrices were real, it has to be casted to (double $*$), otherwise to (mess_double_cpx_t $*$).

Attention

This function completely relies on dense computations. It returns an error if it is called with sparse matrices.

Here is the call graph for this function:



8.3.2.9 int mess_matrix_gaxpy (mess_operation_t *op*, mess_matrix *A*, mess_vector *x*, mess_vector *y*)

Parameters

in	*op*	input operation on matrix
in	*A*	input matrix A
in	*x*	input right hand side vector x
out	*y*	output vector y

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_gaxpy function computes the matrix-vector product with an update of the output, i.e.

$$y \leftarrow y + op(A)x$$

where $op(A)$ can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$
MESS_OP_HERMITIAN	$op(A) = A^H$

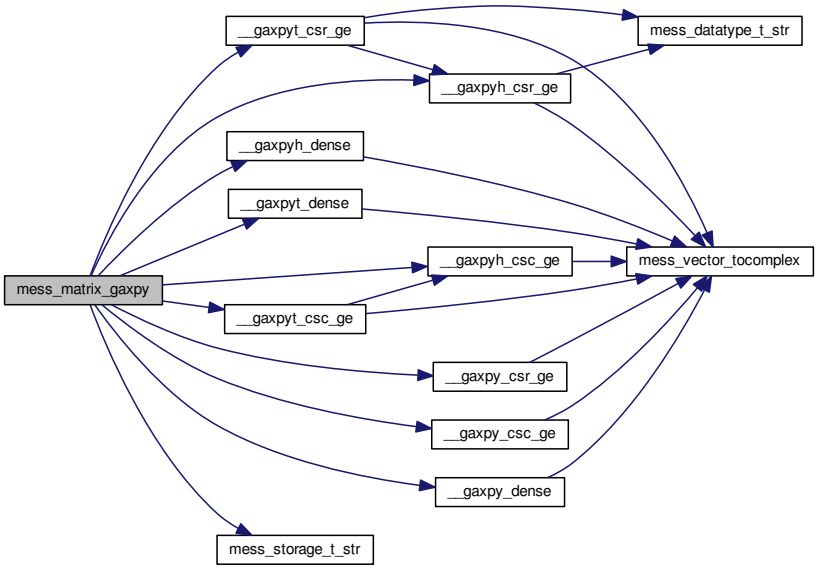
Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_mvp

Here is the call graph for this function:



8.3.2.10 int mess_matrix_gbiorth (mess_matrix *E*, mess_matrix *Vin*, mess_matrix *Win*, mess_matrix *Vout*, mess_matrix *Wout*)

Parameters

in	*E*	input matrix E symmetric positive definite
in	*Vin*	input matrix V_{in}
in	*Win*	input matrix W_{in}
out	*Vout*	output matrix V_{out}
out	*Wout*	output matrix W_{out}

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_gbiorth function constructs a biorthonormal basis

$$W_{out}^T E V_{out} = I$$

from the input matrices V_{in} and W_{in}.
E need to be a symmetric positive definite matrix defining a proper scalar product.

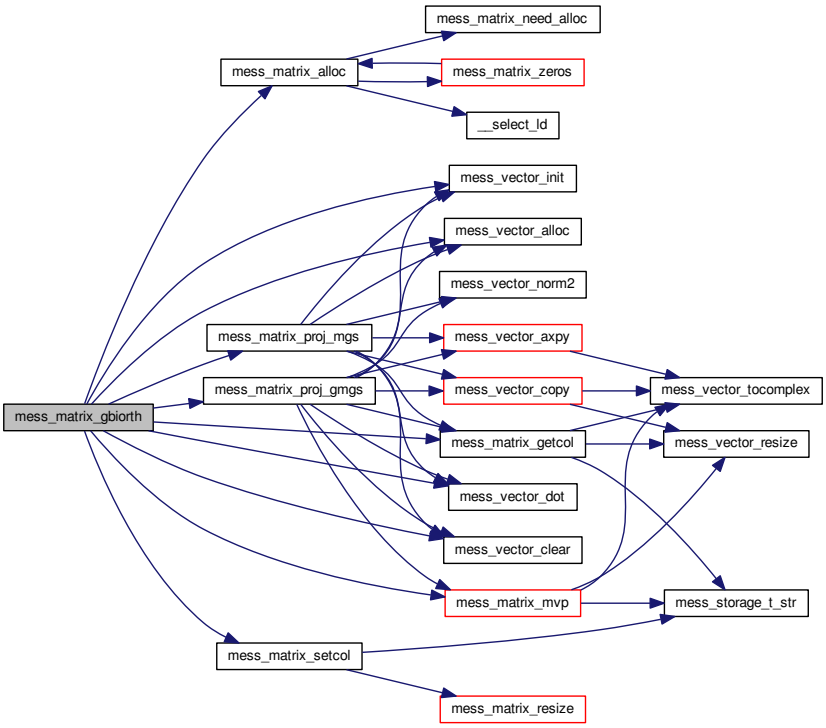
See also

mess_matrix_biorth

Attention

The function only works for real and dense matrices.

Here is the call graph for this function:



8.3.2.11 int mess_matrix_map (mess_matrix *mat*, double(∗)(double) *f_real*, mess_double_cpx_t(∗)(mess_double↩_cpx_t) *f_cpx*, int *full*)

Parameters

in,out	*mat*	input/output matrix mat
in	*f_real*	input scalar real function to apply to each entry of mat
in	*f_cpx*	input scalar complex function to apply to each entry of mat
in	*full*	input scalar if nonzero apply functions to each entry of mat, otherwise only to nonzero entries of mat

Returns

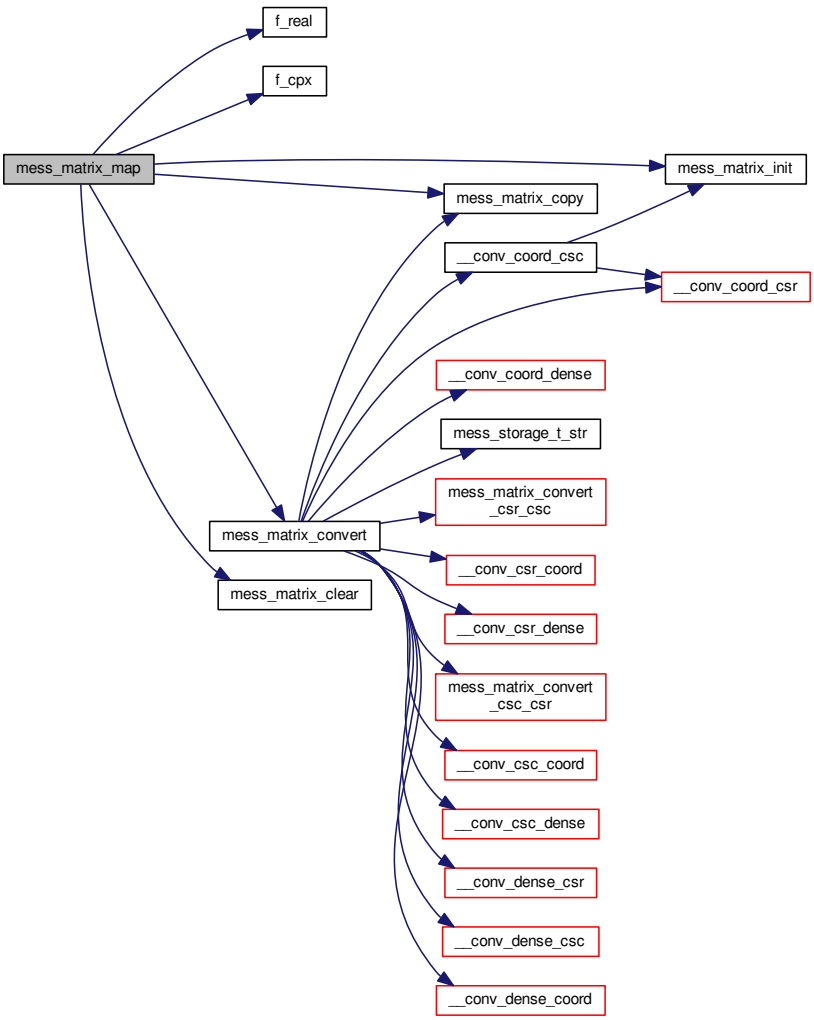
zero on success or a non-zero error value otherwise

The mess_matrix_map function is a higher order function, which applies a scalar real `f_real` or a scalar complex `f_cpx` function to a each entry of a matrix mat. Depending on the data type of mat we perform the following operation:

data type	operation
MESS_REAL	$mat(i,j) \leftarrow f_{real}(mat(i,j))$
MESS_COMPLEX	$mat(i,j) \leftarrow f_{cpx}(mat(i,j))$

If `full` is zero then the functions `f_real` and `f_cpx` are only applied to the nonzero entries of `mat`. If `full` is nonzero the the function `f_real` and `f_cpx` are applied to all entries of `mat`. If `mat` is a sparse matrix mess_matrix_map may convert the matrix to a dense one due to the possible fill-in.

Here is the call graph for this function:



8.3.2.12 int mess_matrix_map_abs (mess_matrix *mat*, int *full*)

8.3.2.13 int mess_matrix_map_acos (mess_matrix *mat*, int *full*)

8.3.2.14 int mess_matrix_map_acosh (mess_matrix *mat*, int *full*)

8.3.2.15 int mess_matrix_map_arg (mess_matrix *mat*, int *full*)

8.3.2.16 int mess_matrix_map_asin (mess_matrix *mat*, int *full*)

8.3.2.17 int mess_matrix_map_asinh (mess_matrix *mat*, int *full*)

8.3.2.18 int mess_matrix_map_atan (mess_matrix *mat*, int *full*)

8.3.2.19 int mess_matrix_map_atanh (mess_matrix *mat*, int *full*)

8.3.2.20 int mess_matrix_map_ceil (mess_matrix *mat*, int *full*)

8.3.2.21 int mess_matrix_map_conj (mess_matrix *mat*, int *full*)

8.3.2.22 int mess_matrix_map_cos (mess_matrix *mat*, int *full*)

8.3.2.23 int mess_matrix_map_cosh (mess_matrix *mat*, int *full*)

8.3.2.24 int mess_matrix_map_exp (mess_matrix *mat*, int *full*)

8.3.2.25 int mess_matrix_map_expm1 (mess_matrix *mat*, int *full*)

8.3.2.26 int mess_matrix_map_floor (mess_matrix *mat*, int *full*)

8.3.2.27 int mess_matrix_map_isfinite (mess_matrix *mat*, int *full*)

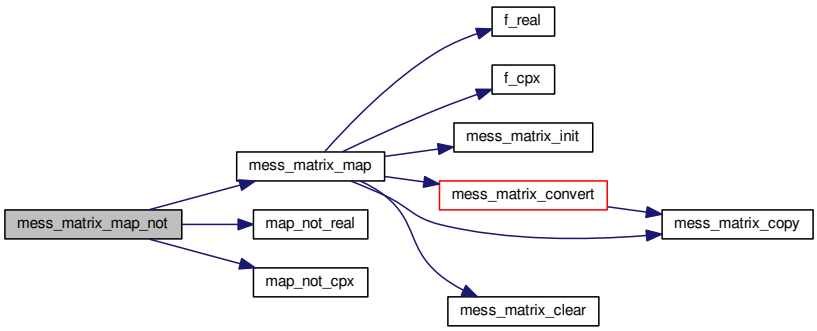
8.3.2.28 int mess_matrix_map_isinf (mess_matrix *mat*, int *full*)

8.3.2.29 int mess_matrix_map_isnan (mess_matrix *mat*, int *full*)

8.3.2.30 int mess_matrix_map_log (mess_matrix *mat*, int *full*)

8.3.2.31 int mess_matrix_map_not (mess_matrix *mat*, int *full*)

Here is the call graph for this function:



8.3.2.32 int mess_matrix_map_round (mess_matrix *mat*, int *full*)

8.3.2.33 int mess_matrix_map_sin (mess_matrix *mat*, int *full*)

8.3.2.34 int mess_matrix_map_sinh (mess_matrix *mat*, int *full*)

8.3.2.35 int mess_matrix_map_sqrt (mess_matrix *mat*, int *full*)

8.3.2.36 int mess_matrix_map_tan (mess_matrix *mat*, int *full*)

8.3.2.37 int mess_matrix_map_tanh (mess_matrix *mat*, int *full*)

8.3.2.38 int mess_matrix_mgs (mess_matrix *A*, mess_matrix *Q*, mess_matrix *R*)

Parameters

in	*A*	input matrix A
out	*Q*	output orthogonal matrix Q
out	*R*	output upper triangular coefficient matrix R (set to NULL if not wanted)

Returns

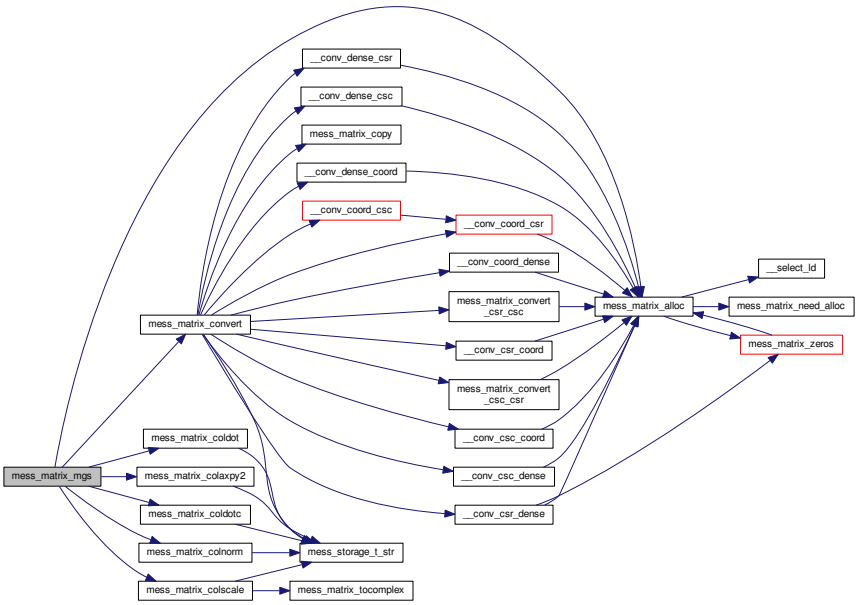
zero on success or a non-zero error value otherwise

The mess_matrix_mgs function computes an orthonormal basis for A using the modified Gram-Schmidt process.
If R is not equal to NULL the coefficients are computed, too.
Sparse matrices are converted to dense ones immediately.

See also

mess_matrix_mgs_inplace
mess_matrix_mgs_add
mess_matrix_orth

Here is the call graph for this function:



8.3.2.39 int mess_matrix_mgs_add (mess_matrix *Z*, mess_matrix *V*, mess_matrix *E*)

Parameters

in,out	*Z*	input/output orthogonal matrix Z
in	*V*	input new columns to add V
in	*E*	input mass matrix E, NULL if not given

Returns

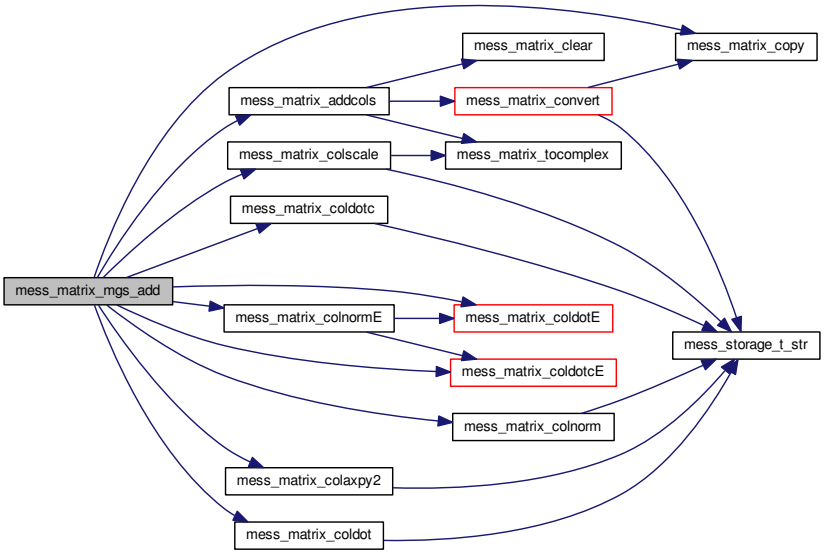
zero on success or a non-zero error value otherwise

The mess_matrix_mgs_add function computes an orthonormal basis for $\begin{bmatrix} Z & V \end{bmatrix}$ with the modified Gram-Schmidt process under the assumption that Z is already orthogonal.
If E is given, we use the E inner product. In this case E need to be symmetric positive definite. The result is written to Z.

See also

mess_matrix_mgs
mess_matrix_mgs_inplace
mess_matrix_orth

Here is the call graph for this function:



8.3.2.40 int mess_matrix_mgs_inplace (mess_matrix *Q*)

Parameters

in,out	*Q*	on input: matrix to orthonormalize on output: orthonormal matrix

Returns

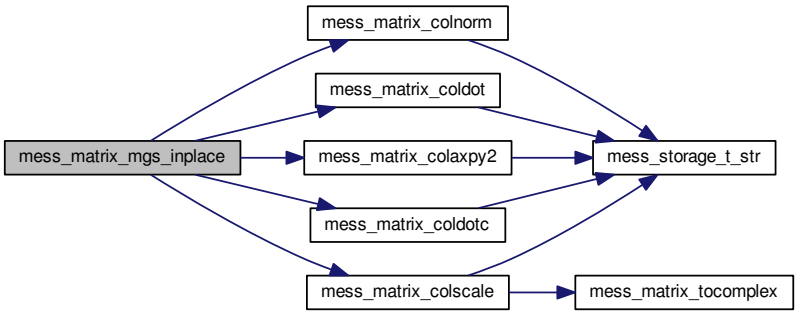
zero on success or a non-zero error value otherwise

The mess_matrix_mgs_inplace function computes an orthonormal basis for Q using the modified Gram-Schmidt process. All operations are inplace on Q and need no extra memory.

See also

mess_matrix_mgs
mess_matrix_mgs_add
mess_matrix_orth

Here is the call graph for this function:



8.3.2.41 int mess_matrix_multiply (mess_operation_t *opA*, mess_matrix *A*, mess_operation_t *opB*, mess_matrix *B*, mess_matrix *C*)

8.3.2.42 int mess_matrix_mvp (mess_operation_t *op*, mess_matrix *A*, mess_vector *x*, mess_vector *y*)

Parameters

in	*op*	input operation applied to A
in	*A*	input matrix A
in	*x*	input right hand side vector x
out	*y*	output vector y

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_mvp function computes the matrix-vector product, i.e. $y = op(A)x$ where $op(A)$ can be

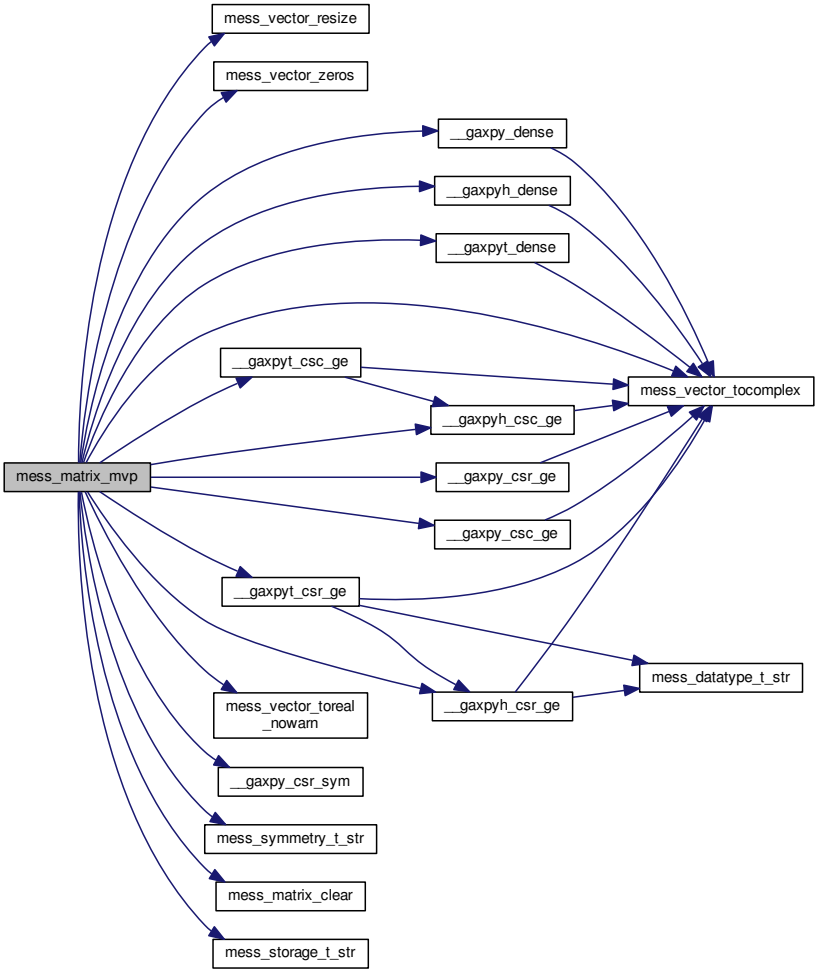
Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$
MESS_OP_HERMITIAN	$op(A) = A^H$

The function chooses automatically the right kernel for the multiplication depending on the storage type of the matrix. Currently only general matrices without any symmetric structure property are supported.

See also

mess_matrix_gaxpy

Here is the call graph for this function:



8.3.2.43 int mess_matrix_null (mess_matrix *A*, mess_matrix *Z*)

Parameters

in	*A*	input matrix A
out	*Z*	output Z orthogonal basis for the nullspace of A

Returns

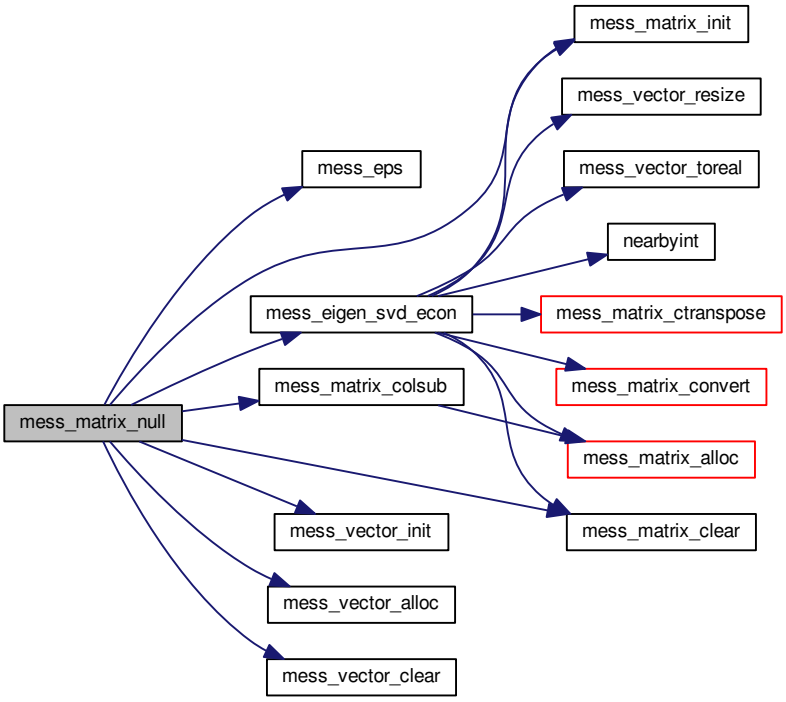
zero on success or a non zero error code otherwise

The mess_matrix_null function computes an orthogonal basis for the nullspace of a matrix using the SVD.

See also

mess_matrix_rank
mess_matrix_orth

Here is the call graph for this function:



8.3.2.44 int mess_matrix_orth (mess_matrix *A*, mess_matrix *Q*)

Parameters

in	*A*	input matrix A
out	*Q*	output Q orthogonal basis for the range of A

Returns

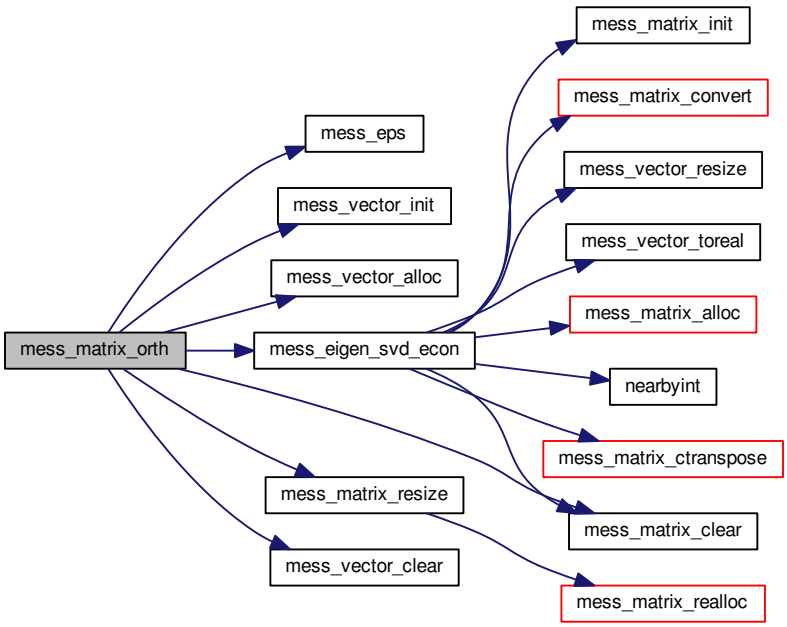
zero on success or a non zero error code otherwise

The mess_matrix_orth function computes an orthogonal basis for the range of a matrix using the SVD.

See also

mess_matrix_rank
mess_matrix_null

Here is the call graph for this function:



8.3.2.45 int mess_matrix_proj_gmgs (mess_vector *in*, mess_matrix *Q*, mess_matrix *QT*, mess_matrix *E*, char * *Eflag*, mess_int_t *colcount*, mess_vector *out*, double * *nrmout*)

Parameters

in	*in*	input vector
in	*Q*	input first subspace to project on
in	*QT*	input second subspace to project on
in	*E*	input mass matrix
in	*Eflag*	input operation flag for E
out	*out*	output vector
in	*colcount*	input column counter for Q and QT (number of columns to use from Q and QT)
out	*nrmout*	norm of the output vector

Returns

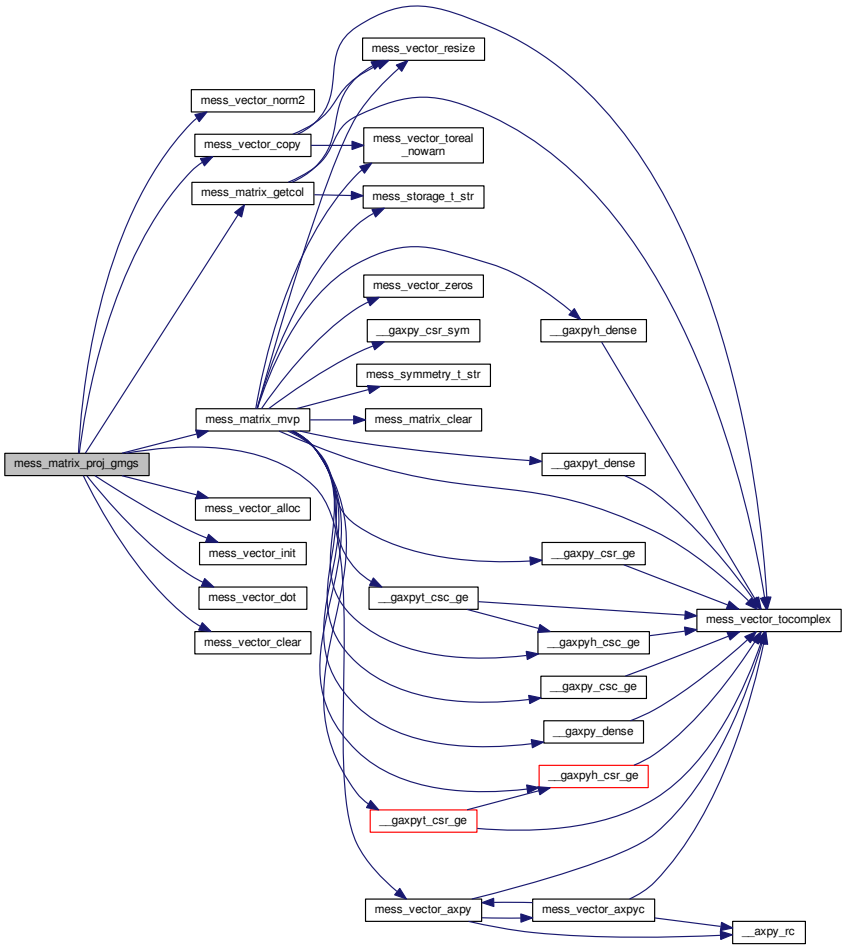
zero on success or a non-zero error value otherwise

The mess_matrix_proj_mgs function computes the skew projection

$$out = \Pi_i^{colcount}(I - Q_i op(E) QT_i^T)u$$

where Q_i is the i-th column of Q and QT_i is the i-th column of QT.
The `Eflag` argument specifies whether $op(E) = E$ or $op(E) = E^T$ is used.

Here is the call graph for this function:



8.3.2.46 int mess_matrix_proj_mgs (mess_vector *in*, mess_matrix *Q*, mess_matrix *QT*, mess_int_t *colcount*, mess_vector *out*, double * *nrmout*)

Parameters

in	*in*	input vector
in	*Q*	input first subspace to project on
in	*QT*	input second subspace to project on
out	*out*	output vector
in	*colcount*	input column counter for Q and QT (number of columns to use from Q and QT)
out	*nrmout*	norm of the output vector

Returns

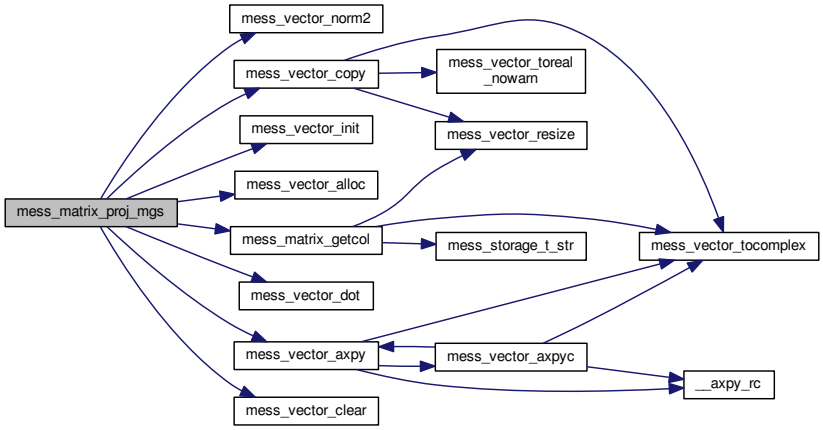
zero on success or a non-zero error value otherwise

The mess_matrix_proj_mgs function computes the skew projection

$$out = \Pi_i^{colcount}(I - Q_i QT_i^T)u$$

where Q_i is the i-th column of Q and QT_i is the i-th column of QT.

Here is the call graph for this function:



8.3.2.47 int mess_matrix_proj_sym (mess_matrix *matrix*)

Parameters

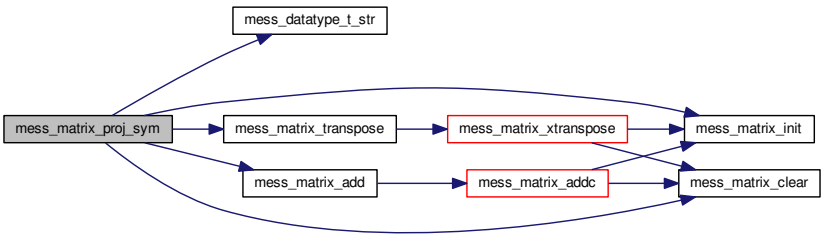
in,out	*matrix*	input matrix

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_proj_sym function projects the given matrix on the space of symmetric matrices by computing $A \mapsto \mathcal{P}(A) := frac12(A + A^T)$, where $(P) : (R)^{n \times n} \to (S)^{n \times n}$.

Here is the call graph for this function:



8.3.2.48 int mess_matrix_rank (mess_matrix *A*, mess_int_t * *rank*)

Parameters

in	*A*	input matrix A
out	*rank*	output estimated rank of A

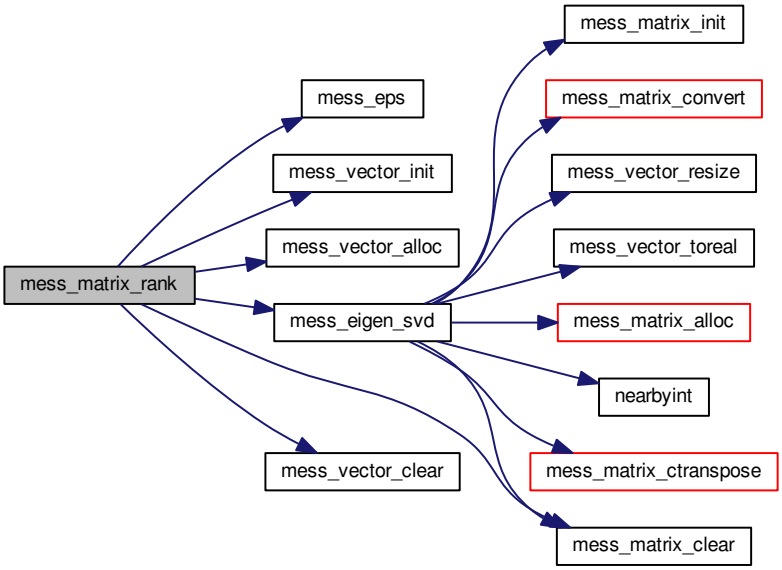
Returns

zero on success or a non zero error code otherwise

The mess_matrix_rank function estimates the rank of a matrix. It uses a similar algorithm like it is done by MATLAB.

```
S=SVD(A)
tol = max(size(A))*eps*max(S)
rank = sum ( s>tol)
```

Here is the call graph for this function:



8.3.2.49 int mess_matrix_rowsums (mess_matrix *A*, mess_vector *x*)

Parameters

in	*A*	input matrix A
out	*x*	output vector x with the row sums of A

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_rowsums function computes the sum of each row of a matrix by evaluating

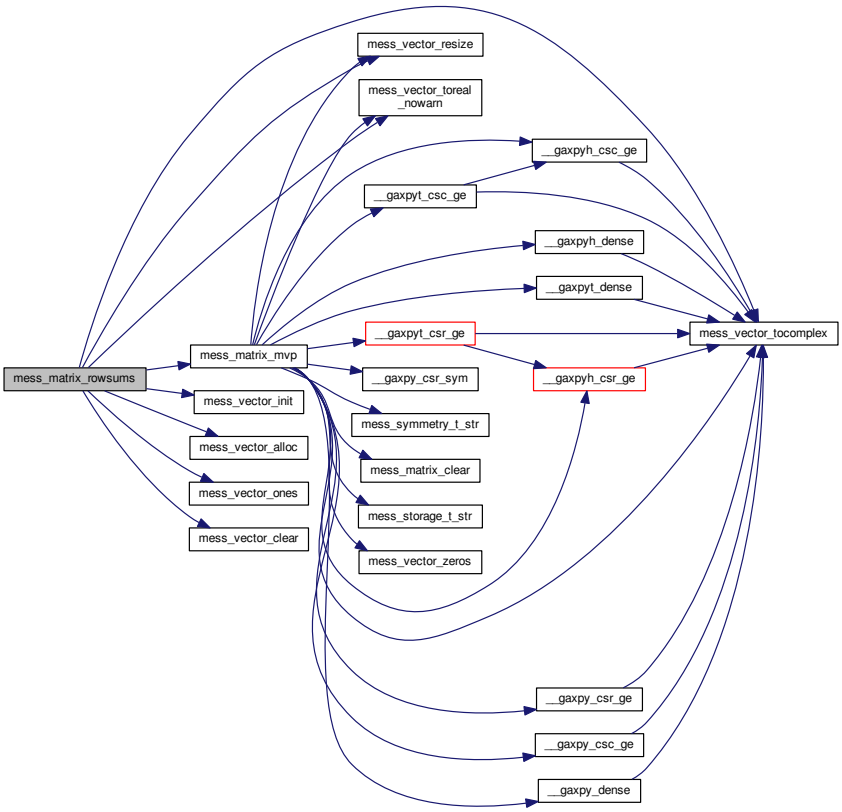
$$x = Ae$$

where $e = [1, ..., 1]^T$.

See also

mess_matrix_colsums

Here is the call graph for this function:



8.3.2.50 int mess_matrix_scale (double *alpha*, mess_matrix *A*)

Parameters

in	*alpha*	input scaling factor α
in,out	*A*	matrix to scale A

Returns

zero on success or a non zero error code otherwise

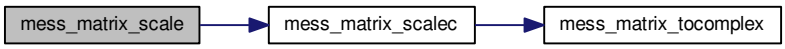
The mess_matrix_scale function scales a real matrix by $alpha$:

$$A \leftarrow \alpha A.$$

See also

mess_matrix_scalec

Here is the call graph for this function:



8.3.2.51 int mess_matrix_scalec (mess_double_cpx_t *alpha*, mess_matrix *A*)

Parameters

in	*alpha*	input scaling factor
in,out	*A*	matrix to scale

Returns

zero on success or a non zero error code otherwise

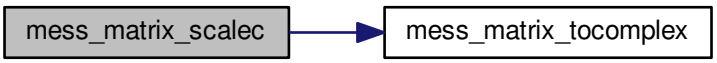
The mess_matrix_scalec function scales a complex matrix by $alpha$:

$$A \leftarrow \alpha A.$$

See also

mess_matrix_scale

Here is the call graph for this function:



8.3.2.52 int mess_matrix_trace (mess_matrix *A*, double * *tr*)

Parameters

in	*A*	input matrix A
out	*tr*	output trace of A

Returns

zero on success or a non zero error code otherwise

The mess_matrix_trace function computes the trace of a real matrix.

Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_tracec

Here is the call graph for this function:

8.3.2.53 int mess_matrix_tracec (mess_matrix *A*, mess_double_cpx_t * *tr*)

Parameters

in	*A*	input matrix A
out	*tr*	output trace A

Returns

zero on success or a non zero error code otherwise

The mess_matrix_tracec function computes the trace of a complex matrix.

Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_trace

Here is the call graph for this function:

8.3.2.54 int mess_matrix_transpose (mess_matrix *A*, mess_matrix *AT*)

Parameters

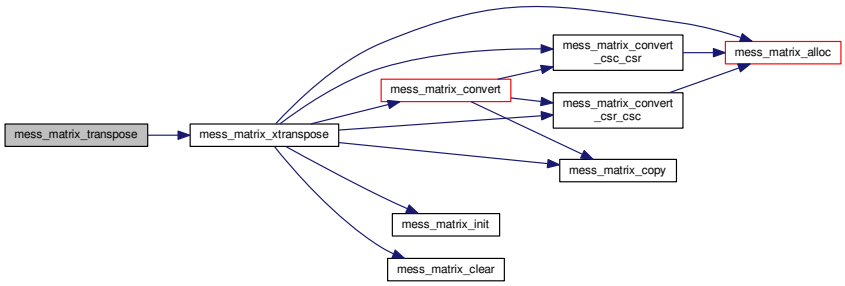
in	*A*	input matrix A
out	*AT*	output transposed matrix A^T

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_transpose function creates the transpose of a given matrix.
It supports all internal matrix storage formats.

Here is the call graph for this function:



8.3.2.55 int mess_matrix_xtranspose (mess_matrix *A*, mess_matrix *AT*, int *hermitian*)

Parameters

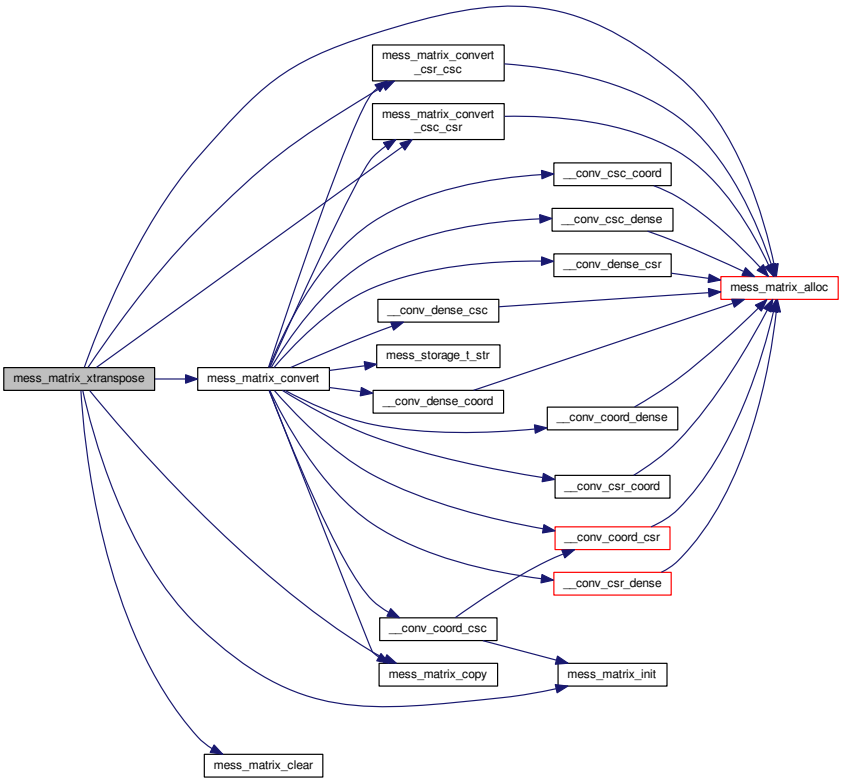
in	*A*	input matrix A
out	*AT*	output transposed matrix A^T (A^H)
in	*hermitian*	input if true then hermitian transpose is computed

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_xtranspose function creates the transpose/hermitian transpose of a given matrix.
It supports all internal matrix storage formats.

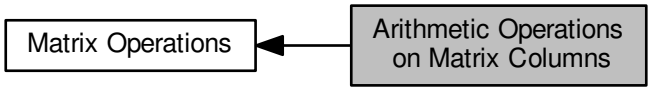
Here is the call graph for this function:



8.4 Arithmetic Operations on Matrix Columns

This category contains various arithmetic operations on matrix columns.

Collaboration diagram for Arithmetic Operations on Matrix Columns:



Functions

- int mess_matrix_colnorm (mess_matrix Q, mess_int_t col, double *norm)

  Compute the euclidean norm of a matrix column.
- int mess_matrix_colscale (mess_matrix Q, mess_int_t col, mess_double_cpx_t scale)

  Scale a column of a matrix.
- int mess_matrix_coldot (mess_matrix Q, mess_int_t col1, mess_int_t col2, double *dot)

  Compute the dot product between two columns of a real matrix.
- int mess_matrix_coldotc (mess_matrix Q, mess_int_t col1, mess_int_t col2, mess_double_cpx_t *dot)

  Compute the dot product between two columns of a complex matrix.
- int mess_matrix_colaxpy (mess_double_cpx_t alpha, mess_vector vect, mess_int_t col, mess_matrix Q)

  Perform an axpy-update on a column of a matrix.
- int mess_matrix_colaxpy2 (mess_matrix Q, mess_double_cpx_t alpha, mess_int_t colc, mess_int_t col1)

  Update a column of a matrix by α times another column.
- int mess_matrix_colvecdot (mess_matrix Q, mess_int_t col, mess_vector v, double *dot)

  Compute the dot product between a matrix column and a vector (real).
- int mess_matrix_colvecdotc (mess_matrix Q, mess_int_t col, mess_vector v, mess_double_cpx_t *dot)

  Computes the dot product between a matrix column and a vector (complex).
- int mess_matrix_colvecaxpy (mess_double_cpx_t alpha, mess_int_t col, mess_matrix Q, mess_vector v)

  Update a vector using a column of a matrix.
- int mess_matrix_coldotE (mess_matrix Q, mess_matrix E, mess_int_t col1, mess_int_t col2, double *dot)

  Compute the weighted dot product of two matrix columns (real).
- int mess_matrix_coldotcE (mess_matrix Q, mess_matrix E, mess_int_t col1, mess_int_t col2, mess_double↩_cpx_t *dot)

  Compute the weighted dot product of two matrix columns (complex).
- int mess_matrix_colnormE (mess_matrix Q, mess_matrix E, mess_int_t col, double *dot)

  Compute the E norm of a matrix column.

8.4.1 Detailed Description

These are for example the norm of a column, scalar product of two columns, a gaxpy update of a column and other similar operations. Most of them are only efficient on dense matrices but will work on sparse ones too. Most of them use corresponding level-1 BLAS (`NetlibBLAS`, `OpenBLAS`, `Intel MKL`, `Atlas`) operations if it is possible.

See also

Arithmetic Operations on Matrices
Computation of Matrix Norms

8.4.2 Function Documentation

8.4.2.1 int mess_matrix_colaxpy (mess_double_cpx_t *alpha*, mess_vector *v*, mess_int_t *col*, mess_matrix *Q*)

Parameters

in	*alpha*	input coefficient α
in	*v*	input vector v to add
in	*col*	input index col of the column
in,out	*Q*	input/output matrix Q

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_colaxpy function updates a column of matrix using an axpy-operation

$$Q(:, col) \leftarrow \alpha \cdot v + Q(:, col)$$

.

Attention

This function expects a dense matrix.

See also

mess_matrix_colvecaxpy
mess_matrix_colaxpy2

Here is the call graph for this function:

8.4.2.2 int mess_matrix_colaxpy2 (mess_matrix *Q*, mess_double_cpx_t *alpha*, mess_int_t *colc*, mess_int_t *col1*)

Parameters

in,out	*Q*	input/output matrix Q
in	*alpha*	input coefficient α
in	*colc*	input index col_c of the column to add
in	*col1*	input index col_1 of the column to update

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_colaxpy2 function updates a column col_1 of Q with an axpy-operation with a second column col_c. It computes

$$Q(:, col_1) \leftarrow Q(:, col_1) + \alpha \cdot Q(:, col_c).$$

Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_colaxpy
mess_matrix_colvecaxpy

Here is the call graph for this function:

8.4.2.3 int mess_matrix_coldot (mess_matrix *Q*, mess_int_t *col1*, mess_int_t *col2*, double * *dot*)

Parameters

in	*Q*	input matrix Q
in	*col1*	input index col_1 of first column
in	*col2*	input index col_2 of second column
out	*dot*	output dot product

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_coldot function computes the dot product between two columns of a real matrix:

$$dot = Q(:, col_1)^T Q(:, col_2).$$

Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_coldotc
mess_matrix_coldotE
mess_matrix_coldotcE
mess_matrix_colvecdot
mess_matrix_colvecdotc

Here is the call graph for this function:

8.4.2.4 int mess_matrix_coldotc (mess_matrix *Q*, mess_int_t *col1*, mess_int_t *col2*, mess_double_cpx_t * *dot*)

Parameters

in	*Q*	input matrix Q
in	*col1*	input index col_1 of first column
in	*col2*	input index col_2 of second column
out	*dot*	output dot product

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_coldotc function computes the dot product between two columns of a complex matrix:

$$dot = Q(:, col_1)^H Q(:, col_2).$$

Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_coldot
mess_matrix_coldotE
mess_matrix_coldotcE
mess_matrix_colvecdot
mess_matrix_colvecdotc

Here is the call graph for this function:

8.4.2.5 int mess_matrix_coldotcE (mess_matrix *Q*, mess_matrix *E*, mess_int_t *col1*, mess_int_t *col2*, mess_double_cpx_t * *dot*)

Parameters

in	*Q*	input complex matrix Q
in	*E*	input symmetric positive definite weight matrix E
in	*col1*	input index col_1 of the first column
in	*col2*	input index col_2 of the second column
out	*dot*	output dot product

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_coldotcE function computes

$$dot = Q(:, col_1)^H * E * Q(:, col_2)$$

for complex matrices.

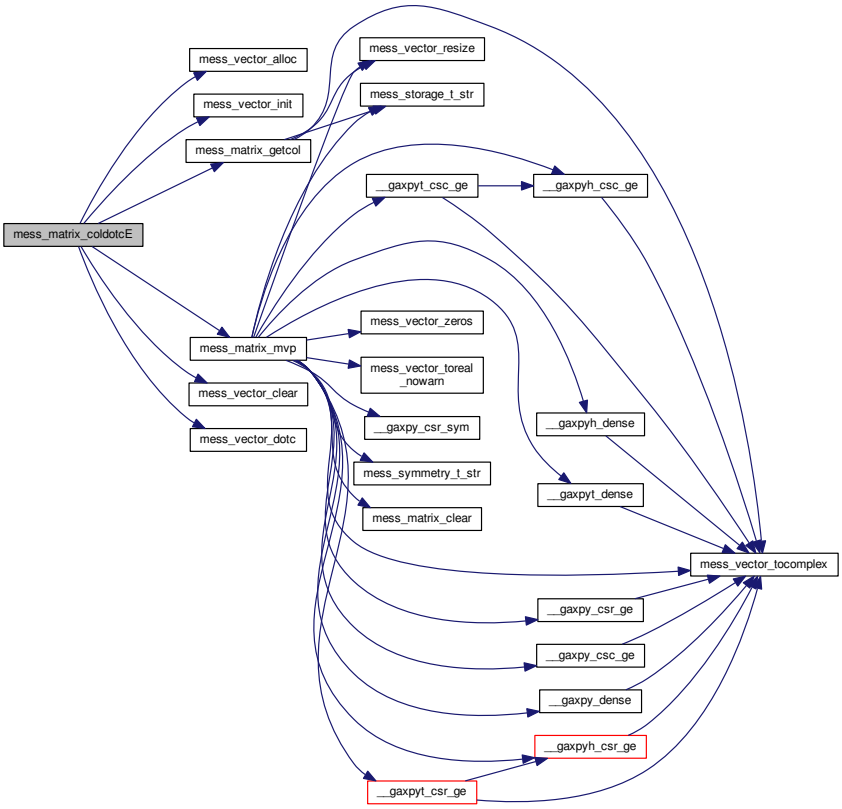
Attention

E need to be symmetric positive definite.

See also

mess_matrix_coldot
mess_matrix_coldotc
mess_matrix_coldotE

Here is the call graph for this function:



8.4.2.6 int mess_matrix_coldotE (mess_matrix *Q*, mess_matrix *E*, mess_int_t *col1*, mess_int_t *col2*, double * *dot*)

Parameters

in	*Q*	input real matrix Q
in	*E*	input symmetric positive definite weight matrix E
in	*col1*	input index col_1 of the first column
in	*col2*	input index col_2 of the second column
out	*dot*	output dot product

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_coldotE function computes

$$dot = Q(:, col_1)^T * E * Q(:, col_2)$$

for real matrices.

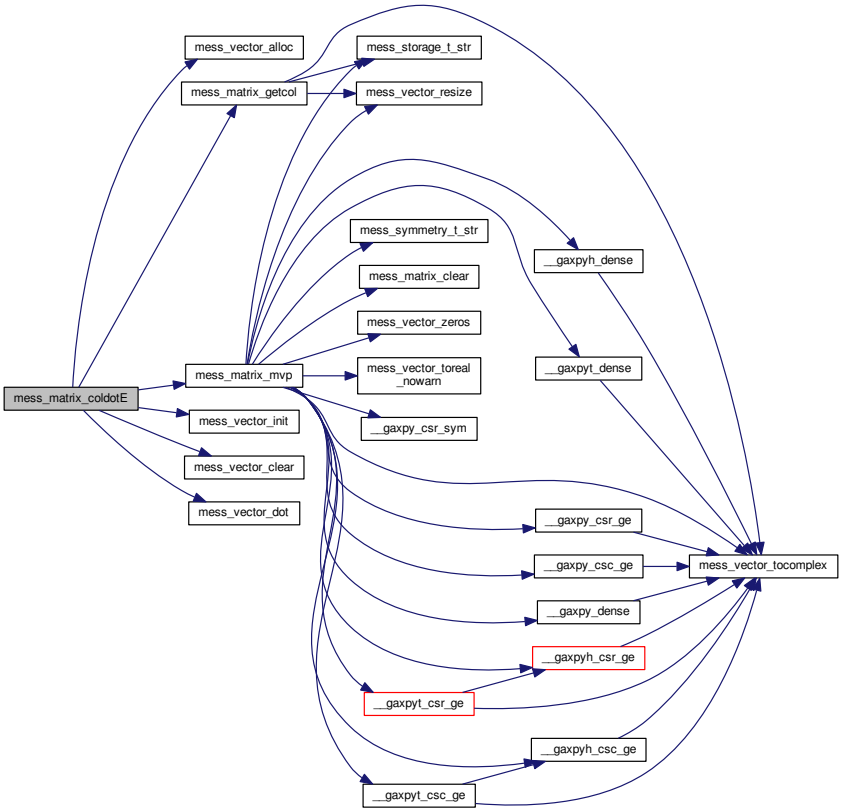
Attention

E need to be symmetric positive definite.

See also

mess_matrix_coldot
mess_matrix_coldotc
mess_matrix_coldotcE

Here is the call graph for this function:



8.4.2.7 int mess_matrix_colnorm (mess_matrix *Q*, mess_int_t *col*, double * *norm*)

Parameters

in	*Q*	input matrix Q
in	*col*	input index col of the column
out	*norm*	output euclidean norm the desired column of Q

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_colnorm function computes

$$norm = \|Q(:, col)\|_2.$$

Attention

This function does not yet support the MESS_COORD storage type.

Here is the call graph for this function:

8.4.2.8 int mess_matrix_colnormE (mess_matrix *Q*, mess_matrix *E*, mess_int_t *col*, double * *norm*)

Parameters

in	*Q*	input matrix Q
in	*E*	input symmetric positive definite weight matrix E
in	*col*	input index col of the column
out	*norm*	output norm

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_colnormE function computes

$$norm = \|Q(:, col)\|_E = \sqrt{Q(:, col)^T E Q(:, col)}$$

or

$$norm = \|Q(:, col)\|_E = \sqrt{Q(:, col)^H E Q(:, col)}$$

for real or complex matrix Q and real matrix E.

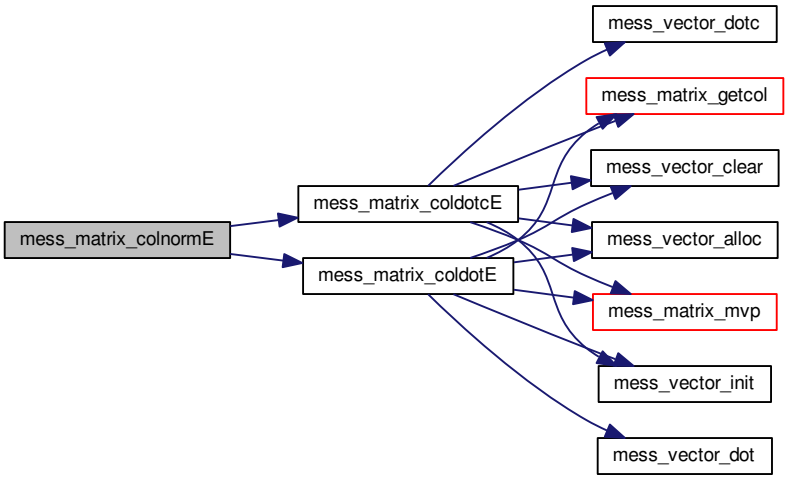
Attention

E needs to be symmetric positive definite.

See also

mess_matrix_colnorm

Here is the call graph for this function:



8.4.2.9 int mess_matrix_colscale (mess_matrix *Q*, mess_int_t *col*, mess_double_cpx_t *scale*)

Parameters

in,out	*Q*	input/output matrix Q
in	*col*	input index col of the column
in	*scale*	input factor $scale$ to scale

Returns

zero on success or a non-zero error value otherwise

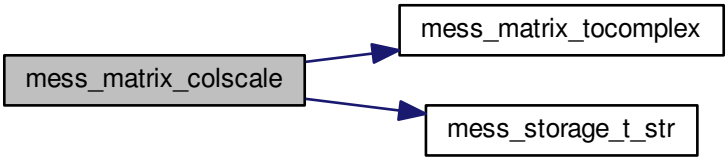
Attention

This function does not yet support the MESS_COORD storage type.

The mess_matrix_colscale function scales a column of a matrix:

$$Q(:, col) \leftarrow scale \cdot Q(:, col).$$

Here is the call graph for this function:



8.4.2.10 int mess_matrix_colvecaxpy (mess_double_cpx_t *alpha*, mess_int_t *col*, mess_matrix *Q*, mess_vector *v*)

Parameters

in	*alpha*	input coefficient α
in	*col*	input index col of the column
in	*Q*	input matrix Q
in,out	*v*	input/output vector to update v

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_colvecaxpy function computes

$$v \leftarrow v + \alpha \cdot Q(:, col).$$

The operation with exchanged roles is implemented in mess_matrix_colaxpy.
In case of real Q and real v the imaginary part of α will be ignored. If you want to avoid this effect convert v to complex vector before computation.
In case of complex Q and real v, v will be a complex vector after computation.

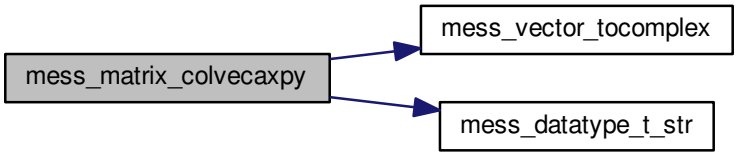
Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_colaxpy
mess_matrix_colaxpy2

Here is the call graph for this function:



8.4.2.11 int mess_matrix_colvecdot (mess_matrix *Q*, mess_int_t *col*, mess_vector *v*, double * *dot*)

Parameters

in	*Q*	input the matrix Q
in	*col*	input index col of the column
in	*v*	input vector v
out	*dot*	output dot product

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_colvecdot function computes

$$dot = Q(:, col)^T v$$

for real data.

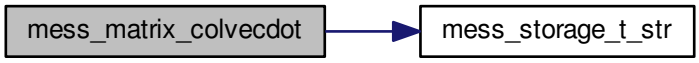
Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_coldot
mess_matrix_coldotc
mess_matrix_coldotE
mess_matrix_coldotcE
mess_matrix_colvecdotc

Here is the call graph for this function:



8.4.2.12 int mess_matrix_colvecdotc (mess_matrix *Q*, mess_int_t *col*, mess_vector *v*, mess_double_cpx_t * *dot*)

Parameters

in	*Q*	input the matrix Q
in	*col*	input index col of the column
in	*v*	input vector v
out	*dot*	output dot product

The mess_matrix_colvecdotc function computes

$$dot = Q(:, col)^H v$$

for complex valued data.

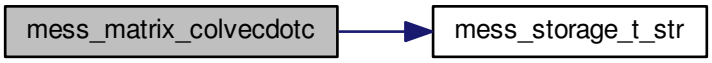
Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_coldot
mess_matrix_coldotc
mess_matrix_coldotE
mess_matrix_coldotcE
mess_matrix_colvecdot

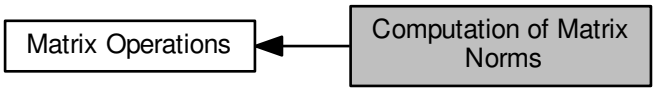
Here is the call graph for this function:



8.5 Computation of Matrix Norms

This category collects all matrix norm computations and related functions like the condition number or its estimate.

Collaboration diagram for Computation of Matrix Norms:



Functions

- int mess_matrix_norminvest (mess_direct decomp, double *nrm)

  Estimate 1-norm of the inverse of a real square matrix.
- int mess_matrix_condest (mess_matrix A, double *nrm)

  Compute a lower bound for the 1-norm condition number of a real square matrix.
- int mess_matrix_norm1 (mess_matrix A, double *nrm)

  Compute the 1-norm of a matrix.
- int mess_matrix_norm2 (mess_matrix A, double *nrm)

  Compute the 2-norm of a matrix.
- int mess_matrix_norm2inv (mess_matrix A, double *nrm)

  Compute the 2-norm of the inverse of a matrix.
- int mess_matrix_normf (mess_matrix A, double *nrm)

  Compute the Frobenius-norm of a matrix.
- int mess_matrix_normf2 (mess_matrix A, double *nrm)

  Compute the square of the Frobenius-norm of a matrix.
- int mess_matrix_norminf (mess_matrix A, double *nrm)

  Compute the ∞-norm of a matrix.
- int mess_matrix_norm (mess_matrix A, mess_norm_t nrm_t, double *nrm)

  Compute the norm of a matrix.
- int mess_matrix_diffnorm (mess_matrix A, mess_matrix B, double *nrm)

  Compute the 2-norm of the difference of two matrices.
- int mess_matrix_diffnormf (mess_matrix A, mess_matrix B, double *nrm)

  Compute the Frobenius norm of the difference of two matrices.
- int mess_matrix_dynorm2 (mess_matrix A, double *nrm)

  Compute the 2-norm of $A^H A$ or AA^H.
- int mess_matrix_dynormf (mess_matrix A, double *nrm)

  Compute the Frobenius-norm of $A^H A$ or AA^H.
- int mess_matrix_dynormf2 (mess_matrix A, double *nrm)

  Compute the squared Frobenius-norm of $A^H A$ or AA^H.
- int mess_matrix_indefinite_dynorm2 (mess_matrix W, mess_matrix K, double *nrm)

  Compute the 2-norm of $WW^T - KK^T$, $WW^H - KK^T$, $WW^T - KK^H$ or $WW^H - KK^H$.
- int mess_matrix_indefinite_dynormf (mess_matrix W, mess_matrix K, double *nrm)

  Compute the Frobenius-norm of $WW^T - KK^T$, $WW^H - KK^T$, $WW^T - KK^H$ or $WW^H - KK^H$.
- int mess_matrix_indefinite_dynormf2 (mess_matrix W, mess_matrix K, double *nrm)

  Compute the square of the Frobenius-norm of $WW^T - KK^T$, $WW^H - KK^T$, $WW^T - KK^H$ or $WW^H - KK^H$.
- int mess_matrix_mulnormf2 (mess_operation_t op1, mess_matrix A, mess_operation_t op2, mess_matrix B, double *nrm)

  Compute the square of the Frobenius-norm of a $op1(A) * op2(B)$.

8.5.1 Detailed Description

Additionally there exists the computation of the two norm of a low-rank factor without computing the large matrix. Depending on the storage type direct methods or iterative ones will be used for the 2-norm computation.

The computation of matrix-valued residuals is available in Residual Computations or Low-Rank Methods for Matrix Equations.

See also

Dense Eigenvalue Solvers
Sparse Eigenvalue Solvers

8.5.2 Function Documentation

8.5.2.1 int mess_matrix_condest (mess_matrix *A*, double * *nrm*)

Parameters

in	*A*	input real square matrix A
out	*nrm*	output estimated 1-norm condition, if A is not invertible `nrm` is `INFINITY`

Returns

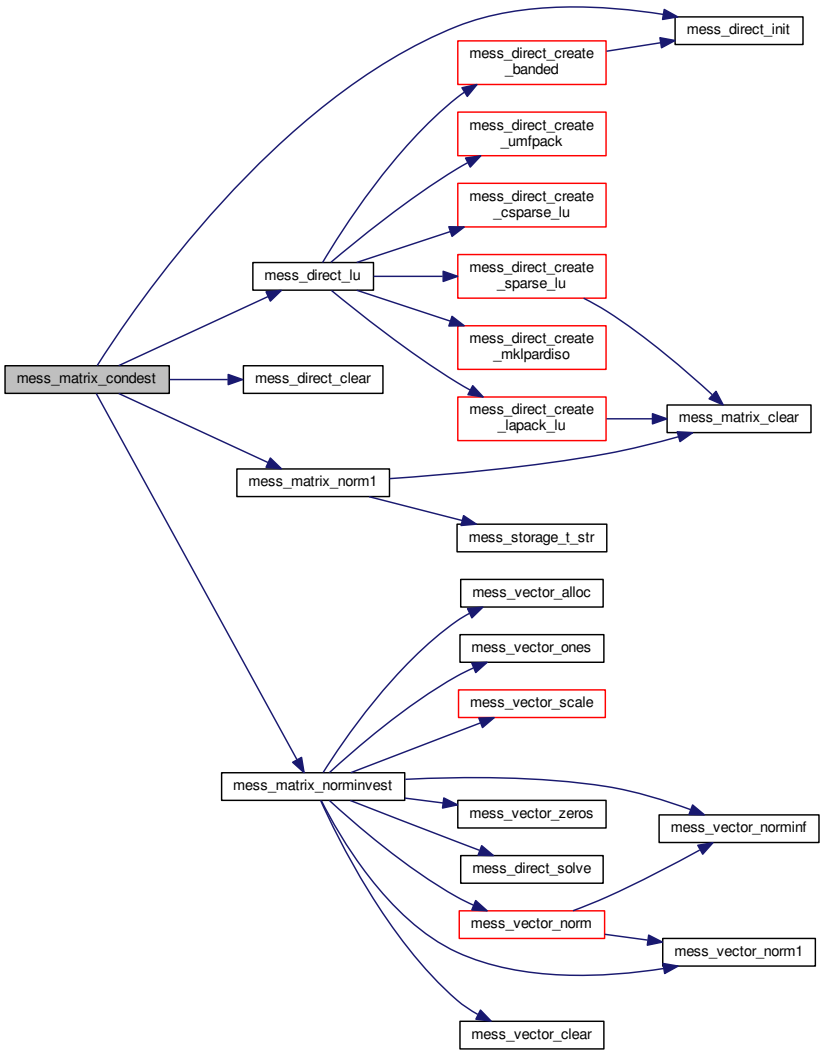
zero on success or a non zero error code

The mess_matrix_condest function estimates the 1-norm condition number of a real square matrix [13].

See also

mess_matrix_norminvest

Here is the call graph for this function:



8.5.2.2 int mess_matrix_diffnorm (mess_matrix *A*, mess_matrix *B*, double * *nrm*)

Parameters

in	*A*	input matrix A
in	*B*	input matrix B
out	*nrm*	output 2-norm of the difference of matrix A and matrix B

Returns

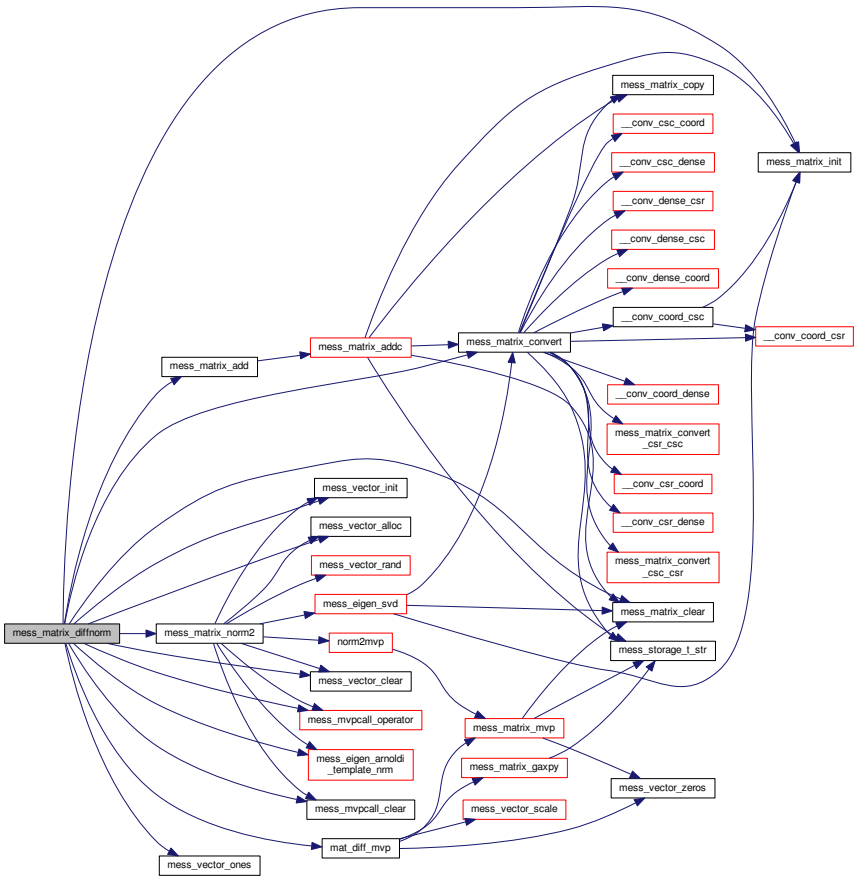
zero on success or a non-zero error value otherwise

The mess_matrix_diffnorm function computes the 2-norm of the difference of two matrices

$$nrm = ||A - B||_2.$$

It uses an Arnoldi process to estimate the largest eigenvalue. If both matrices are MESS_DENSE the difference matrix is computed and a normal eigensolver is used.

Here is the call graph for this function:



8.5.2.3 int mess_matrix_diffnormf (mess_matrix *A*, mess_matrix *B*, double * *nrm*)

Parameters

in	*A*	input matrix A
in	*B*	input matrix B
out	*nrm*	output Frobenius norm of the difference of matrix A and matrix B

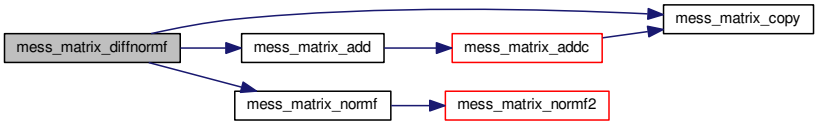
Returns

zero on success or a non-zero error value otherwise

The mess_matrix_diffnormf function computes the Frobenius norm of the difference of two matrices

$$nrm = ||A - B||_F.$$

Here is the call graph for this function:



8.5.2.4 int mess_matrix_dynorm2 (mess_matrix *A*, double * *nrm*)

Parameters

in	*A*	input matrix A
out	*nrm*	output 2-norm

Returns

zero on success or a non zero error code

The mess_matrix_dynorm2 function computes the 2-norm of $A^H A$ or AA^H by using the equality:

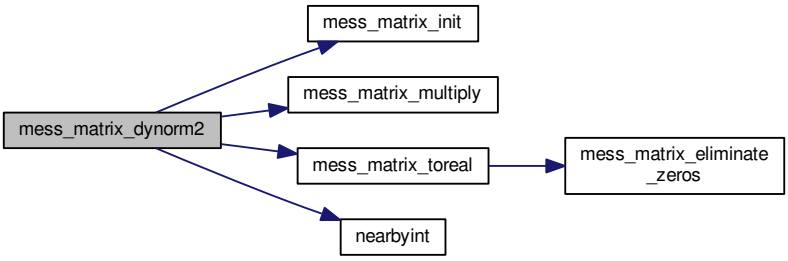
$$\|A^H A\|_2 = \|AA^H\|_2$$

and exploiting the smaller sized product.

See also

mess_matrix_dynormf

Here is the call graph for this function:



8.5.2.5 int mess_matrix_dynormf (mess_matrix *A*, double * *nrm*)

Parameters

in	*A*	input matrix A
out	*nrm*	output Frobenius-norm

Returns

zero on success or a non zero error code

The mess_matrix_dynormf function computes the Frobenius-norm of $A^H A$ or AA^H by using the equality:

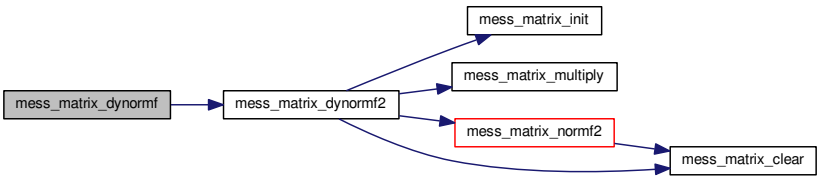
$$\|A^H A\|_F = \|AA^H\|_F$$

and exploiting the smaller sized product.

See also

mess_matrix_dynorm2
mess_matrix_dynormf2

Here is the call graph for this function:



8.5.2.6 int mess_matrix_dynormf2 (mess_matrix *A*, double * *nrm*)

Parameters

in	*A*	input matrix A
out	*nrm*	output Frobenius-norm

Returns

zero on success or a non zero error code

The mess_matrix_dynormf2 function computes the squared Frobenius-norm of $A^H A$ or AA^H by using the equality:

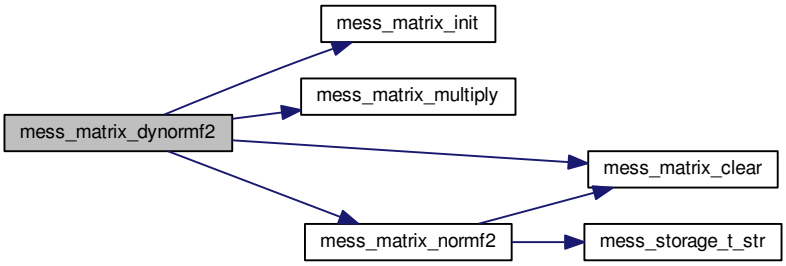
$$\|A^H A\|_F^2 = \|AA^H\|_F^2$$

and exploiting the smaller sized product.

See also

mess_matrix_dynorm2
mess_matrix_dynormf

Here is the call graph for this function:



8.5.2.7 int mess_matrix_indefinite_dynorm2 (mess_matrix *W*, mess_matrix *K*, double * *nrm*)

Parameters

in	*W*	input matrix W
in	*K*	input matrix K
out	*nrm*	output Frobenius-norm

Returns

zero on success or a non zero error code

The mess_matrix_indefinite_dynorm2 function computes the 2-norm of $WW^T - KK^T$, $WW^H - KK^T$, $WW^T - KK^H$ or $WW^H - KK^H$ by using the following facts:

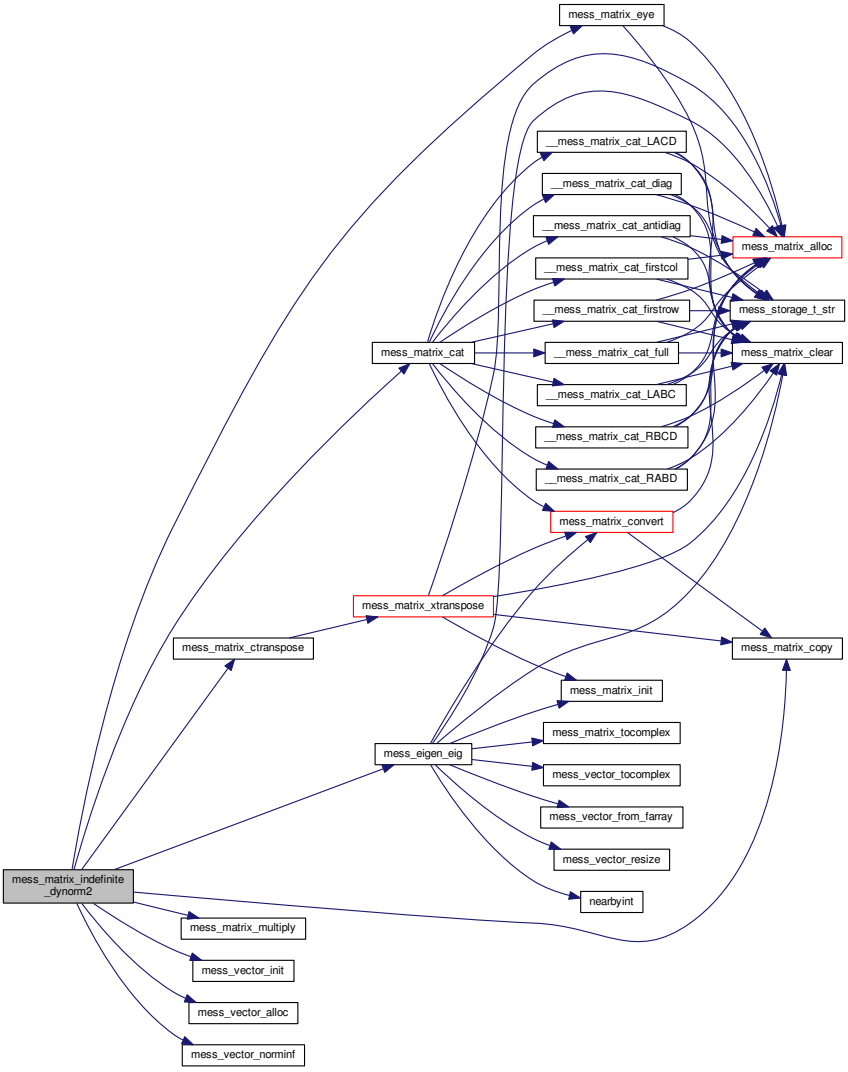
$$\begin{aligned} WW^T - KK^T &= \begin{bmatrix} W & K \end{bmatrix} \begin{bmatrix} I_m & 0 \\ 0 & -I_p \end{bmatrix} \begin{bmatrix} W^T \\ K^T \end{bmatrix} \\ &:= UDU^T \\ \sigma(UDU^T) \setminus \{0\} &= \sigma(U^TUD) \setminus \{0\} \\ \|WW^T - KK^T\|_2 &= max\{|\lambda| \mid \lambda \in \sigma(WW^T - KK)\} = max\{|\lambda| \mid \lambda \in \sigma(U^TUD)\} \end{aligned}$$

Analogue for the case $(\cdot)^H$ and the mixed case $(\cdot)^T, (\cdot)^H$. We also assume that the number of rows is smaller than the number of columns for W and K. For a real/complex matrix $W/\ K$ the operation $(\cdot)^T/\ (\cdot)^H$ is performed.

See also

mess_matrix_dynorm2
mess_matrix_indefinite_dynorm2
mess_matrix_indefinite_dynormf

Here is the call graph for this function:



8.5.2.8 int mess_matrix_indefinite_dynormf (mess_matrix *W*, mess_matrix *K*, double * *nrm*)

Parameters

in	*W*	input matrix W
in	*K*	input matrix K
out	*nrm*	output Frobenius-norm

Returns

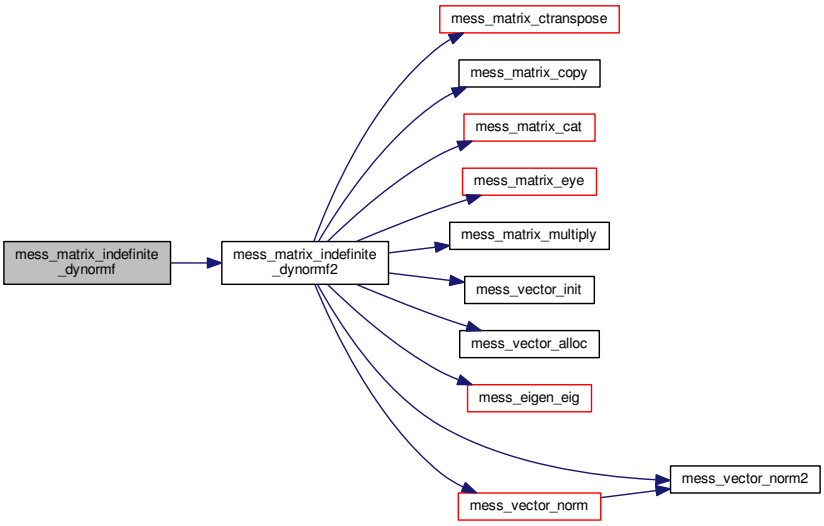
zero on success or a non zero error code

The mess_matrix_indefinite_dynormf function computes the square of the Frobenius-norm of $WW^T - KK^T$, $WW^H - KK^T$, $WW^T - KK^H$ or $WW^H - KK^H$. It directly calls mess_matrix_indefinite_dynormf2 and takes the root.

See also

mess_matrix_dynorm2
mess_matrix_indefinite_dynorm2
mess_matrix_indefinite_dynormf

Here is the call graph for this function:



8.5.2.9 int mess_matrix_indefinite_dynormf2 (mess_matrix *W*, mess_matrix *K*, double * *nrm*)

Parameters

in	*W*	input matrix W
in	*K*	input matrix K
out	*nrm*	output Frobenius-norm

Returns

zero on success or a non zero error code

The mess_matrix_indefinite_dynormf2 function computes the square of the Frobenius-norm of $WW^T - KK^T$, $WW^H - KK^T$, $WW^T - KK^H$ or $WW^H - KK^H$ by using the following facts:

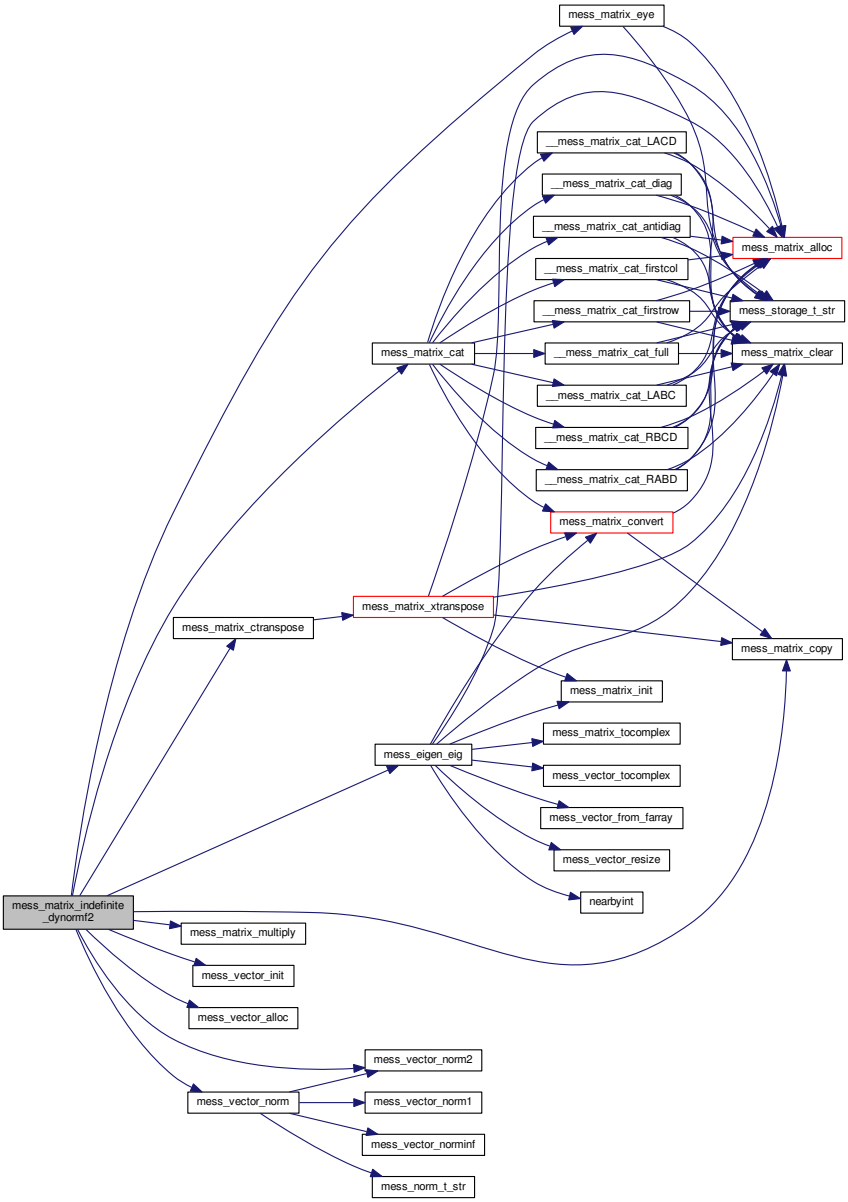
$$\begin{aligned} WW^T - KK^T &= \begin{bmatrix} W & K \end{bmatrix} \begin{bmatrix} I_m & 0 \\ 0 & -I_p \end{bmatrix} \begin{bmatrix} W^T \\ K^T \end{bmatrix} \\ &:= UDU^T \\ \sigma(UDU^T) \setminus \{0\} &= \sigma(U^TUD) \setminus \{0\} \\ \|WW^T - KK^T\|_F^2 &= \sum_{\lambda_i \in \sigma(UDU^T)} \lambda_i^2 = \sum_{\lambda_i \in \sigma(U^TUD)} \lambda_i^2 \end{aligned}$$

Analogue for the case $(\cdot)^H$ and the mixed case $(\cdot)^T, (\cdot)^H$. We also assume that the number of rows is smaller than the number of columns for W and K. For a real/complex matrix $W/\ K$ the operation $(\cdot)^T/\ (\cdot)^H$ is performed.

See also

mess_matrix_dynorm2
mess_matrix_indefinite_dynorm2
mess_matrix_indefinite_dynormf

Here is the call graph for this function:



8.5.2.10 int mess_matrix_mulnormf2 (mess_operation_t *op1*, mess_matrix *A*, mess_operation_t *op2*, mess_matrix *B*, double * *nrm*)

Parameters

in	*op1*	operation of A

Parameters

in	*A*	input matrix A
in	*op2*	operation of B
in	*B*	input matrix B
out	*nrm*	output squared Frobenius-norm

Returns

zero on success or a non zero error code otherwise

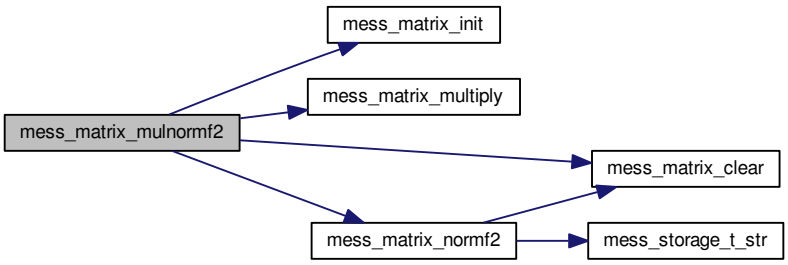
The mess_matrix_mulnormf2 function calculates:

$$nrm = \|op(A) * op(B)\|_F^2.$$

See also

mess_matrix_norm1
mess_matrix_norm2
mess_matrix_norminf

Here is the call graph for this function:



8.5.2.11 int mess_matrix_norm (mess_matrix *A*, mess_norm_t *nrm_t*, double * *nrm*)

Parameters

in	*A*	input matrix A
in	*nrm*↩ _t	input mess_norm_t the desired type of norm
out	*nrm*	output $\|A\|$

Returns

zero on success or a non zero error code otherwise

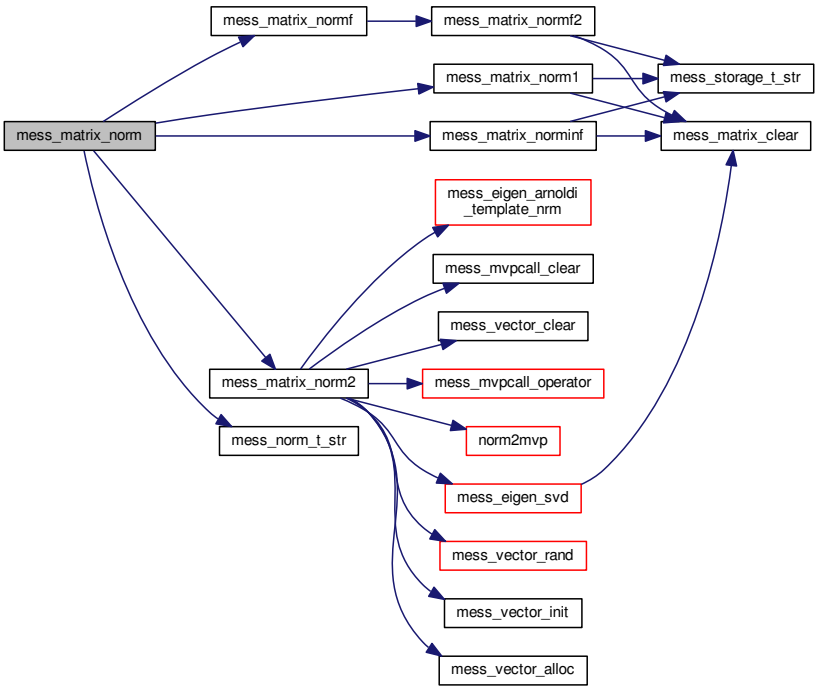
The mess_matrix_norm function calculates the norm of a matrix. Supported norms are:

MESS_2_NORM, MESS_FROBENIUS_NORM, MESS_1_NORM, MESS_INF_NORM.

See also

mess_matrix_norm1
mess_matrix_norm2
mess_matrix_normf
mess_matrix_norminf

Here is the call graph for this function:



8.5.2.12 int mess_matrix_norm1 (mess_matrix *A*, double * *nrm*)

Parameters

in	*A*	input matrix A
out	*nrm*	output 1-norm

Returns

zero on success or a non zero error code otherwise

The mess_matrix_norm1 function calculates the 1-norm of a matrix:

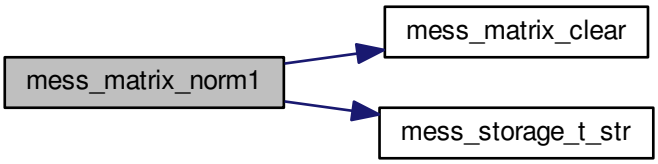
$$nrm = \|A\|_1.$$

It is also know as the column-sum-norm.

See also

mess_matrix_norminf
mess_matrix_norm2
mess_matrix_normf

Here is the call graph for this function:



8.5.2.13 int mess_matrix_norm2 (mess_matrix *A*, double * *nrm*)

Parameters

in	*A*	input matrix A
out	*nrm*	output 2-norm

Returns

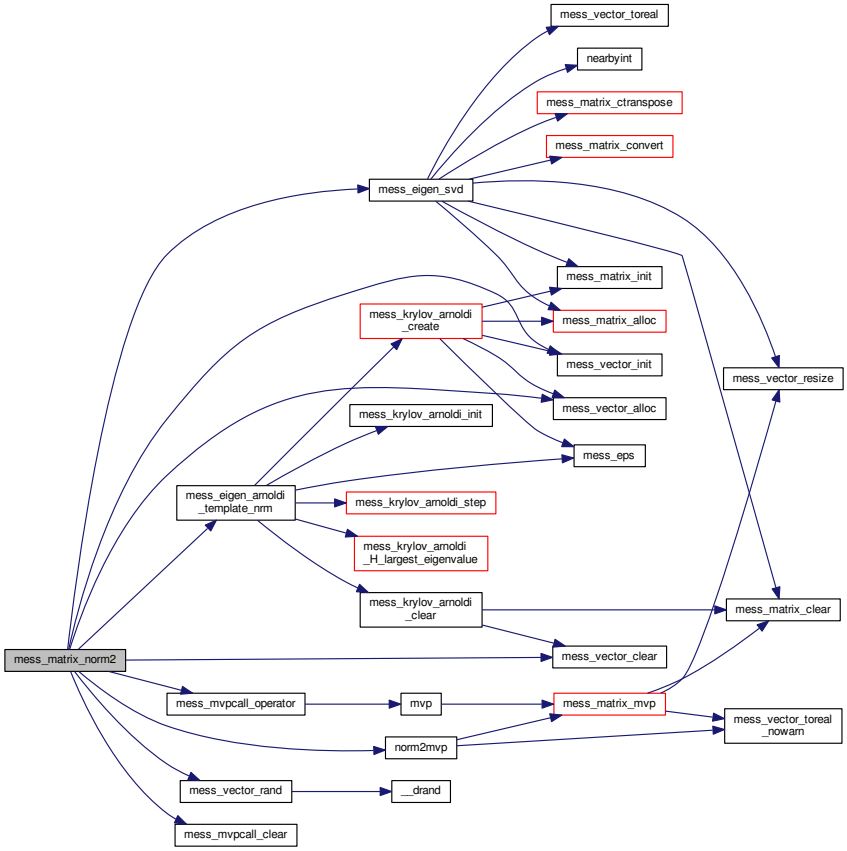
zero on success or a non zero error code otherwise

The mess_matrix_norm2 function calculates the 2-norm of a matrix, i.e. the largest eigenvalue of $A^H A$. Depending on the storage format of the matrix the SVD or a iterative scheme will be used.
Even if the matrix is dense and larger than 100 rows or columns the iterative scheme is used, too.

See also

mess_matrix_norm1
mess_matrix_normf
mess_matrix_norminf
mess_eigen_svd
mess_eigen_arnoldi_template_nrm

Here is the call graph for this function:



8.5.2.14 int mess_matrix_norm2inv (mess_matrix *A*, double ∗ *nrm*)

Parameters

in	*A*	input matrix A
out	*nrm*	output 2-norm

Returns

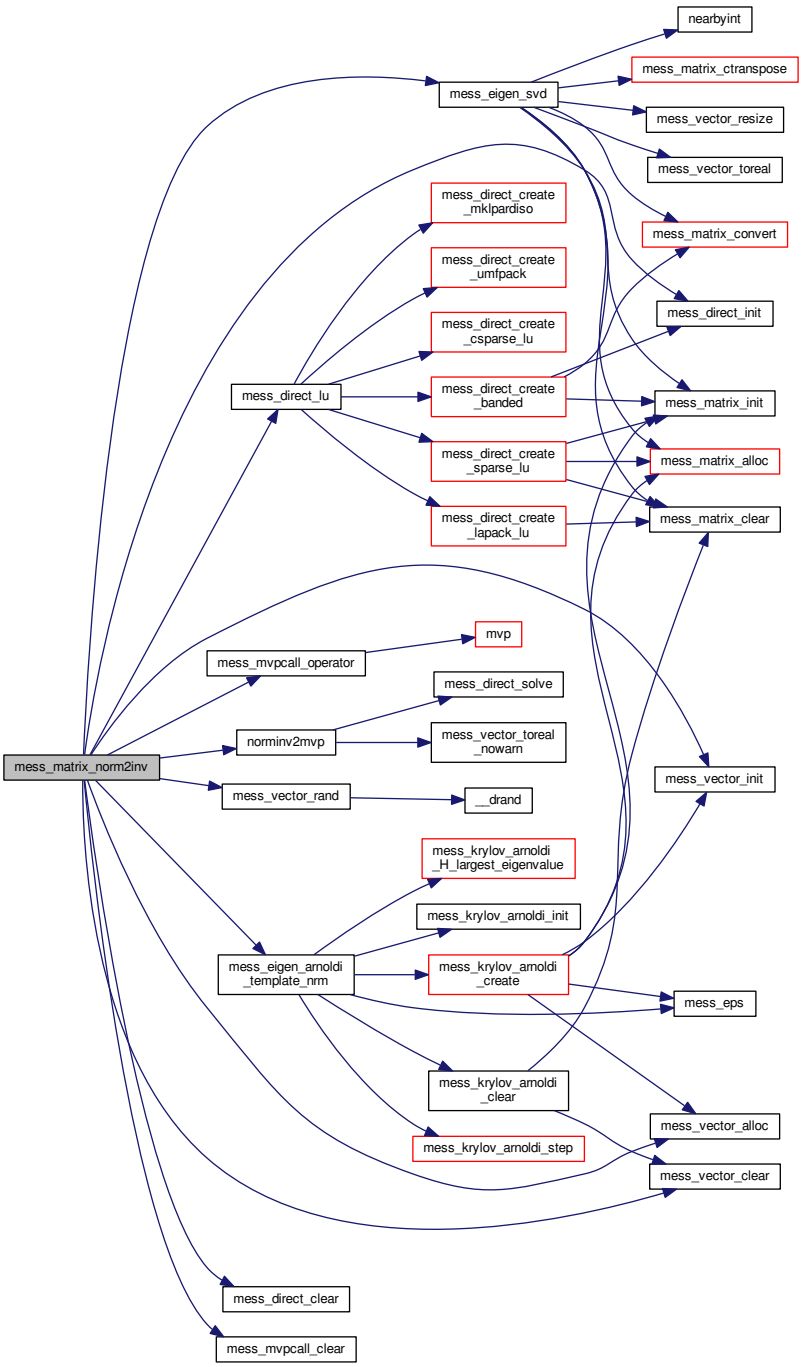
zero on success or a non zero error code otherwise

The mess_matrix_norm2inv function computes the 2-norm of A^{-1}.
If the matrix is small SVD is used to compute the norm. Otherwise a LU decomposition of the matrix is computed and the Arnoldi process is used to estimate norm of the inverse.

See also

mess_matrix_norm2
mess_eigen_svd
mess_eigen_arnoldi_template_nrm
mess_direct_lu

Here is the call graph for this function:



8.5.2.15 int mess_matrix_normf (mess_matrix *A*, double * *nrm*)

Parameters

in	*A*	input matrix A
out	*nrm*	output Frobenius-norm

The mess_matrix_normf function calculates the Frobenius-norm of a matrix:

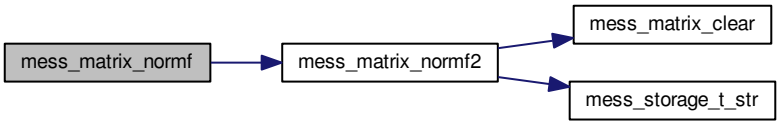
$$nrm = \|A\|_F.$$

It uses the mess_matrix_normf2 function internally.

See also

mess_matrix_normf2
mess_matrix_norm1
mess_matrix_norm2
mess_matrix_norminf

Here is the call graph for this function:



8.5.2.16 int mess_matrix_normf2 (mess_matrix *A*, double * *nrm*)

Parameters

in	*A*	input matrix A
out	*nrm*	output squared Frobenius-norm

Returns

zero on success or a non zero error code otherwise

The mess_matrix_normf2 function calculates the square of Frobenius-norm of a matrix:

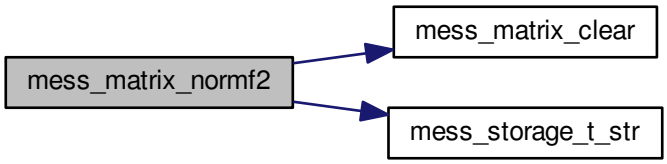
$$nrm = \sum_{i,j=1}^{m,n} |A_{i,j}|^2 = \|A\|_F^2.$$

It is used by mess_matrix_normf to compute the Frobenius-norm.

See also

mess_matrix_norm1
mess_matrix_norm2
mess_matrix_norminf

Here is the call graph for this function:



8.5.2.17 int mess_matrix_norminf (mess_matrix *A*, double ∗ *nrm*)

Parameters

in	*A*	input matrix A
out	*nrm*	output ∞-norm

Returns

zero on success or a non zero error code otherwise

The mess_matrix_norminf function calculates the ∞-norm of a matrix:

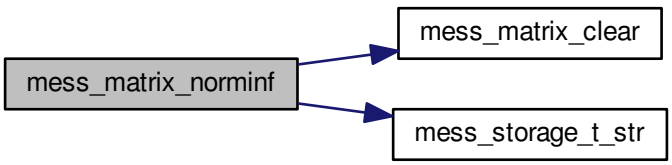
$$nrm = \|A\|_\infty.$$

It is also known as row-sum-norm.

See also

mess_matrix_norm1
mess_matrix_norm2
mess_matrix_normf

Here is the call graph for this function:



8.5.2.18 int mess_matrix_norminvest (mess_direct *decomp*, double * *nrm*)

Parameters

in	*decomp*	input contains decomposition of real matrix A
out	*nrm*	output estimated 1-norm condition of A^{-1}

Returns

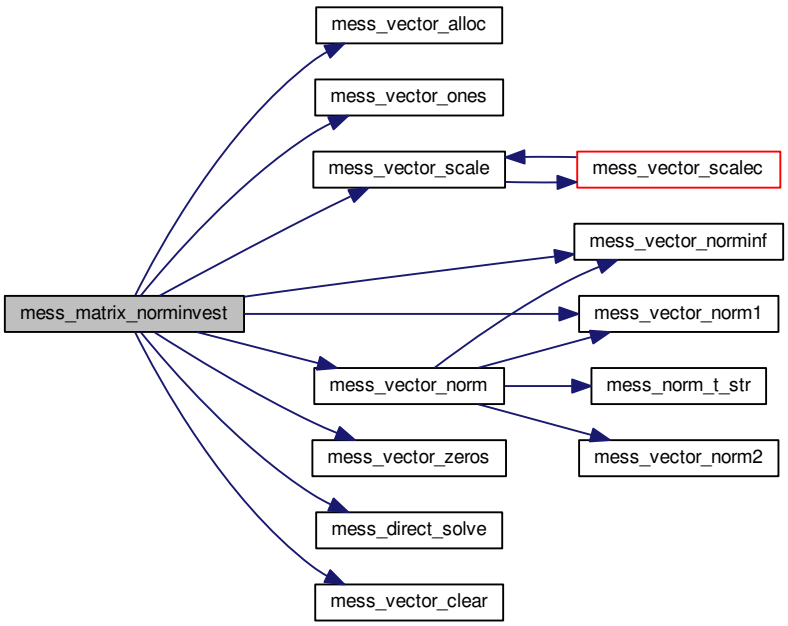
zero on success or a non zero error code

The mess_matrix_norminvest function estimates the 1-norm of the inverse of a real square matrix [13].

See also

mess_matrix_condest

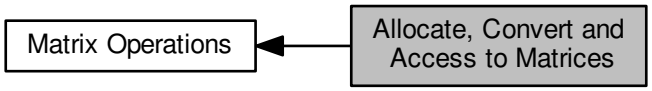
Here is the call graph for this function:



8.6 Allocate, Convert and Access to Matrices

This category contains all subroutines which act on the matrix storage type, data structures and data management.

Collaboration diagram for Allocate, Convert and Access to Matrices:



Functions

- int mess_matrix_alloc (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t nnz, mess_storage_t storetype, mess_datatype_t datatype)

  Allocate a sparse or dense $(m \times n)$ matrix.
- void mess_matrix_alloc_setld (mess_int_t ld_m)
- int mess_matrix_realloc (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t nnz)

  Reallocate or resize a $(m \times n)$ matrix.
- int mess_matrix_resize (mess_matrix matrix, mess_int_t rows, mess_int_t cols)

  Wrapper for mess_matrix_realloc.
- int mess_matrix_reset (mess_matrix matrix)

  Resets a mess_matrix object.
- int mess_matrix_convert (mess_matrix input, mess_matrix output, mess_storage_t outtype)

  Convert the storage type of a matrix (universal converter interface).
- int mess_matrix_sort (mess_matrix mat)

  Sort the internal structure of a sparse matrix.
- int mess_matrix_cat (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix D, mess_storage_t otype, mess_matrix O)

  Concatenate up to four matrices.
- int mess_matrix_rowsub (mess_matrix input, mess_int_t srow, mess_int_t erow, mess_matrix out)

  Copy a row-block out of a matrix.
- int mess_matrix_colsub (mess_matrix input, mess_int_t scol, mess_int_t ecol, mess_matrix out)

  Copy a column block out of a matrix.
- int mess_matrix_sub (mess_matrix in, mess_int_t rowS, mess_int_t rowE, mess_int_t colS, mess_int_t colE, mess_matrix out)

  Copy a sub matrix out of a matrix.
- int mess_matrix_lift (mess_matrix in, mess_int_t n, mess_matrix out)

  Add lower trailing block of n rows to matrix.
- int mess_matrix_realpart (mess_matrix matrix, mess_matrix realpart)

  Get the real part of a matrix.
- int mess_matrix_imagpart (mess_matrix matrix, mess_matrix imagpart)

  Get the imaginary part of a matrix.
- int mess_matrix_complex_from_parts (mess_matrix xr, mess_matrix xc, mess_matrix x)

  Create a complex matrix from real and imaginary parts.

- int mess_matrix_getelement (mess_matrix matrix, mess_int_t row, mess_int_t col, double *val, mess_double_cpx_t *valc)

  Get an element from a matrix.
- int mess_matrix_setelement_complex (mess_matrix matrix, mess_int_t row, mess_int_t col, mess_double_cpx_t value)

  Set a complex entry in a matrix.
- int mess_matrix_setelement (mess_matrix matrix, mess_int_t row, mess_int_t col, double value)

  Set a real entry in a matrix.
- int mess_matrix_convert_csr_csc (mess_matrix inmatrix, mess_matrix outmatrix)

  Convert a Compressed Sparse Row matrix to a Compressed Sparse Column matrix.
- int mess_matrix_convert_csc_csr (mess_matrix inmatrix, mess_matrix outmatrix)

  Convert a Compressed Sparse Column matrix to a Compressed Sparse Row matrix.
- int mess_matrix_symfillup (mess_matrix mat)

  Fill up the symmetry in a coordinate matrix.
- int mess_matrix_dense_from_farray (mess_matrix mat, mess_int_t rows, mess_int_t cols, mess_int_t ld, double *realv, mess_double_cpx_t *complexv)

  Create a MESS_DENSE mess_matrix from a `Fortran` array.
- int mess_matrix_dense_from_carray (mess_matrix mat, mess_int_t rows, mess_int_t cols, double **realv, mess_double_cpx_t **complexv)

  Create a MESS_DENSE mess_matrix from a `C` 2-dimensional array.
- int mess_matrix_csr (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t *rowptr, mess_int_t *colptr, double *values, mess_double_cpx_t *values_cpx)

  Create a MESS_CSR mess_matrix from given arrays.
- int mess_matrix_csc (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t *rowptr, mess_int_t *colptr, double *values, mess_double_cpx_t *values_cpx)

  Create a MESS_CSC mess_matrix from given arrays.
- int mess_matrix_coord (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t nnz, mess_int_t onebased, mess_int_t *rowptr, mess_int_t *colptr, double *values, mess_double_cpx_t *values_cpx)

  Create a MESS_COORD mess_matrix from given arrays.
- int mess_matrix_eye (mess_matrix A, mess_int_t rows, mess_int_t cols, mess_storage_t storetype)

  Generate a real identity matrix.
- int mess_matrix_eyec (mess_matrix mat, mess_int_t rows, mess_int_t cols, mess_storage_t store)

  Generate a complex identity matrix.
- int mess_matrix_tridiag (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_storage_t store_t, mess_datatype_t data_t, mess_double_cpx_t lower, mess_double_cpx_t diag, mess_double_cpx_t upper)

  Create a tridiagonal matrix.
- int mess_matrix_rand_init (mess_int_t *seed)

  Init random number generator with given seed.
- int mess_matrix_rand (mess_matrix mat, mess_int_t rows, mess_int_t cols, mess_storage_t storetype, mess_datatype_t dt, double p)

  Wrapper around mess_matrix_rand_csr, mess_matrix_rand_csc, mess_matrix_rand_dense and mess_matrix_rand_coord.
- int mess_matrix_rand_dense (mess_matrix mat, mess_int_t rows, mess_int_t cols, mess_datatype_t dt)

  Generate a MESS_REAL MESS_COMPLEX random dense matrix.
- int mess_matrix_rand_dense_uniform (mess_matrix mat, mess_int_t *seed, mess_int_t rows, mess_int_t cols)

  Generate a random MESS_REAL dense matrix with a uniform value distribution $(-1, 1)$.
- int mess_matrix_rand_csr (mess_matrix mat, mess_int_t rows, mess_int_t cols, double p, mess_datatype_t dt)

  Generate a MESS_REAL/MESS_COMPLEX random MESS_CSR matrix.
- int mess_matrix_rand_csc (mess_matrix mat, mess_int_t rows, mess_int_t cols, double p, mess_datatype_t dt)

Generate a MESS_REAL/MESS_COMPLEX random MESS_CSC matrix.

- int mess_matrix_rand_coord (mess_matrix mat, mess_int_t rows, mess_int_t cols, double p, mess_datatype_t dt)

  Generate a MESS_REAL/MESS_COMPLEX random COORD matrix.
- int mess_matrix_dupl (mess_matrix mat)

  Detect and sum up duplicate entries in a sparse matrix.
- int mess_matrix_eliminate_zeros (mess_matrix mat)

  Remove zero entries in a sparse matrix.
- int mess_matrix_zeros (mess_matrix matrix)

  Set all entries in a matrix to zero.
- int mess_matrix_ones (mess_matrix matrix)

  Set all entries in a matrix to one.
- int mess_matrix_one_value (mess_matrix matrix, double value)

  Set all entries of a matrix to one value.
- int mess_matrix_one_valuec (mess_matrix matrix, mess_double_cpx_t value)

  Set all entries of a matrix to one value (complex version).
- int mess_matrix_need_alloc (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t nnz, mess_storage_t storetype, mess_datatype_t datatype)

  Check if a new allocation of a matrix is necessary.
- int mess_matrix_copy (mess_matrix in, mess_matrix out)

  Copy a matrix to another one.
- int mess_matrix_getcol (mess_matrix matrix, mess_int_t col, mess_vector c)

  Get a column from a matrix.
- int mess_matrix_getrow (mess_matrix matrix, mess_int_t row, mess_vector r)

  Get a row from a matrix.
- int mess_matrix_setcol (mess_matrix matrix, mess_int_t col, mess_vector colv)

  Set a column in a matrix.
- int mess_matrix_triu (mess_matrix mat, mess_int_t k)

  Upper triangular part of mess_matrix.
- int mess_matrix_tril (mess_matrix mat, mess_int_t k)

  Lower triangular part of mess_matrix.
- int mess_matrix_rowscalem (mess_vector r, mess_matrix mat)

  Scale rows of a matrix.
- int mess_matrix_colscalem (mess_vector c, mess_matrix mat)

  Scale cols of a matrix.
- int mess_matrix_setrow (mess_matrix matrix, mess_int_t row, mess_vector rowv)

  Set a row in a matrix.
- int mess_matrix_diag (mess_matrix matrix, mess_vector d)

  Get the main diagonal elements of from matrix.
- int mess_matrix_diag_from_vector (mess_vector v, mess_matrix diag)

  Create a diagonal mess_matrix from mess_vector.
- int mess_matrix_diagpos (mess_matrix matrix, mess_int_t *pos)

  Find the position of the main diagonal elements.
- int mess_matrix_addcols (mess_matrix matrix, mess_matrix toadd)

  Add columns to an existing matrix.
- int mess_matrix_addcols2p (mess_matrix Z, double s1, mess_matrix V1, double s2, mess_matrix V2)

  Add the sum of two column matrices to a real matrix.
- int mess_matrix_addcols1 (mess_matrix Z, double s1, mess_matrix V1)

  Add a scaled column to a real matrix.
- int mess_matrix_toreal (mess_matrix m)

  Change the data type of a matrix to real.

- int mess_matrix_tocomplex (mess_matrix m)

  Change the data type of a matrix to complex.
- int mess_matrix_totype (mess_matrix mat, mess_datatype_t dt)

  Convert a matrix to a specified data type.
- int mess_matrix_view (mess_matrix original, mess_int_t srow, mess_int_t scol, mess_int_t rows, mess_int_t cols, mess_matrix view)
- int mess_matrix_bandwidth (mess_matrix matrix, mess_int_t *kl, mess_int_t *ku)

  Determine the bandwidth of a matrix.

8.6.1 Detailed Description

It also includes routine to generate typical matrices like the identity or random ones. Beside that concatenation and access to submatrices are part of this subsection too.

If you want to import matrices from other libraries which provide direct access to MESS_CSR, MESS_CSC, MESS_COORD or MESS_DENSE storage the functions mess_matrix_csr, mess_matrix_csc, mess_matrix_coord, mess_matrix_dense_from_carray, and mess_matrix_dense_from_farray are available. Otherwise the mess_matrix_alloc function provides an interface to allocate the different supported matrix formats.

See also

Data Structures, Types and Macros for Handling Matrices

8.6.2 Function Documentation

8.6.2.1 int mess_matrix_addcols (mess_matrix *matrix*, mess_matrix *toadd*)

Parameters

in,out	*matrix*	input/output matrix where the new columns should be added
in	*toadd*	input matrix with the columns to be added

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_addcols function adds columns to an existing matrix.
It works like the

```
Z = [Z V];
```

statement in MATLAB.

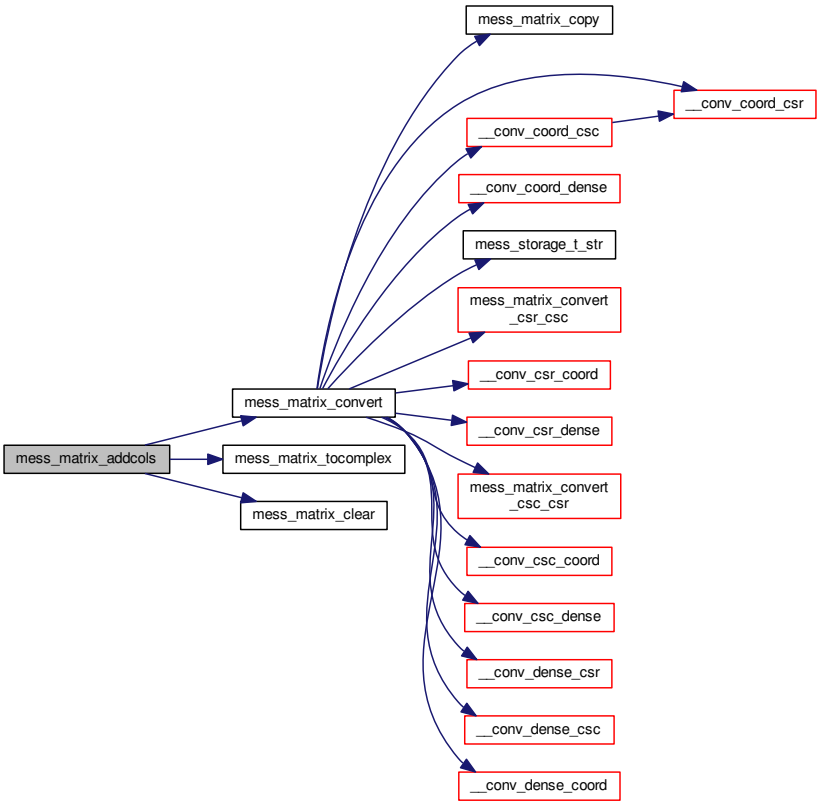
Attention

The function currently only supports dense matrices.

See also

mess_matrix_mgs_add
mess_matrix_addcols2p
mess_matrix_addcols1

Here is the call graph for this function:



8.6.2.2 int mess_matrix_addcols1 (mess_matrix *Z*, double *s1*, mess_matrix *V1*)

Parameters

in,out	*Z*	input/output matrix where to add the column
in	*s1*	input scaling factor V1
in	*V1*	input matrix V1

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_addcols1 function forms

```
Z = [Z , s1*V1]
```

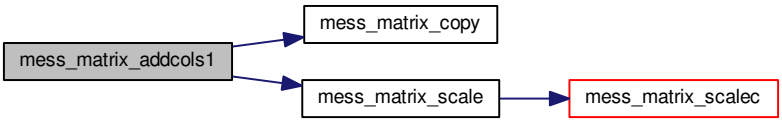
Attention

The function currently only supports real dense matrices.

See also

mess_matrix_addcols2p
mess_matrix_addcols

Here is the call graph for this function:



8.6.2.3 int mess_matrix_addcols2p (mess_matrix *Z*, double *s1*, mess_matrix *V1*, double *s2*, mess_matrix *V2*)

Parameters

in,out	*Z*	input/output matrix where to add the sum
in	*s1*	input scaling factor for V1
in	*V1*	input matrix V1
in	*s2*	input scaling factor for V2
in	*V2*	input matrix V2

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_addcols2p function forms

```
Z = [Z , s1*V1+s2*V2]
```

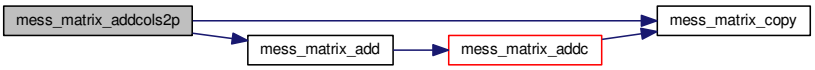
Attention

The function currently only supports real dense matrices.

See also

mess_matrix_addcols2p
mess_matrix_addcols1

Here is the call graph for this function:



8.6.2.4 int mess_matrix_alloc (mess_matrix *matrix*, mess_int_t *rows*, mess_int_t *cols*, mess_int_t *nnz*, mess_storage_t *storetype*, mess_datatype_t *datatype*)

Parameters

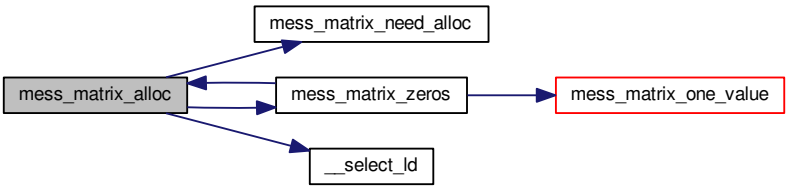
out	*matrix*	output matrix to allocate
in	*rows*	input number of rows
in	*cols*	input number of cols
in	*nnz*	input number of nonzero elements (irrelevant for dense matrices)
in	*storetype*	input type of the storage format
in	*datatype*	input data type of the matrix

Returns

zero on success or a non zero error code otherwise

The mess_matrix_alloc function allocates a matrix.
It is possible to allocate dense, compressed sparse row, compressed sparse column and coordinate matrices.
This function checks if the matrix is already allocated and has the right size. If this is the case the matrix is only set to zero. Otherwise all data structures are freed and allocated correctly.
If **M.E.S.S.** is configured in debug mode the allocations are counted and printed to stderr.
In case you allocate a dense matrix the leading dimension is set to the number of rows. This can change in the future.

Here is the call graph for this function:



8.6.2.5 void mess_matrix_alloc_setld (mess_int_t *ld_m*)

8.6.2.6 int mess_matrix_bandwidth (mess_matrix *matrix*, mess_int_t * *kl*, mess_int_t * *ku*)

Parameters

in	*matrix*	input matrix
out	*kl*	output bandwidth below the diagonal
out	*ku*	output bandwidth above the diagonal

Returns

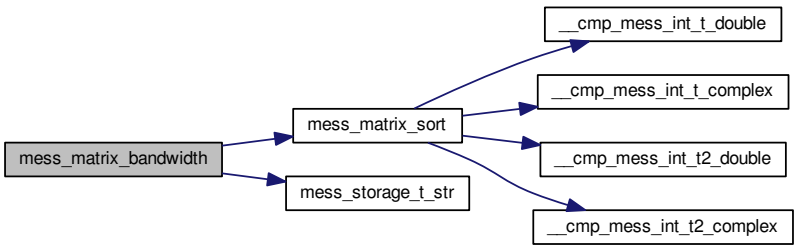
zero on success or a non zero error code otherwise

The mess_matrix_bandwidth function determines the bandwidth of a matrix. The bandwidth is returned as bandwidth below and above the diagonal like it is necessary for the banded matrix subroutines in LAPACK. If the input matrix is dense both bandwidths are set to the number of rows -1 or respectively the number of columns -1.

Attention

This function does not yet support the MESS_COORD storage type.

Here is the call graph for this function:



8.6.2.7 int mess_matrix_cat (mess_matrix *A*, mess_matrix *B*, mess_matrix *C*, mess_matrix *D*, mess_storage_t *otype*, mess_matrix *O*)

Parameters

in	*A*	input matrix A
in	*B*	input matrix B
in	*C*	input matrix C
in	*D*	input matrix D
in	*otype*	input output storage type
out	*O*	output matrix O

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_cat function concatenates 4 matrices like:

$$O = \begin{bmatrix} A & B \\ C & D \end{bmatrix}.$$

If a block does not exists it must be set to NULL.
For example:

```
mess_matrix_cat(NULL,B,C,NULL,MESS_CSR,O);
```

performs

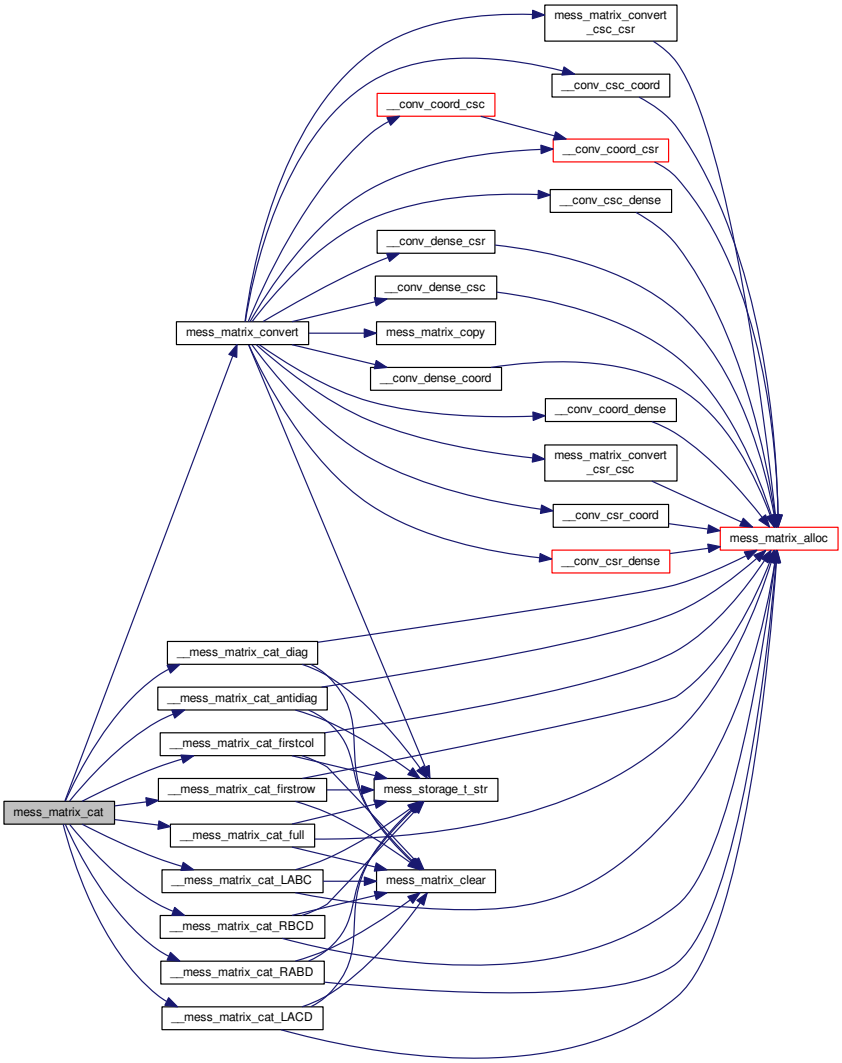
$$0 = \begin{bmatrix} 0 & B \\ C & 0 \end{bmatrix}$$

and in addition the output has MESS_CSR storage type.

Attention

This function does not yet support the MESS_COORD storage type as output type.

Here is the call graph for this function:



8.6.2.8 int mess_matrix_colscalem (mess_vector *c*, mess_matrix *A*)

Parameters

in	*c*	input mess_vector with scaling vectors
in,out	*A*	matrix to scale

Returns

zero on success or a non zero error code otherwise

Author

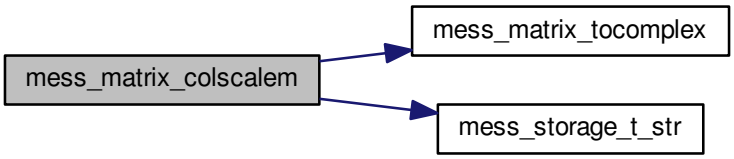
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

The mess_matrix_colscalem function scales $i-th$ col of a matrix by the $i-th$ entry of c. mess_matrix::cols should be equal to mess_vector::dim.

See also

mess_matrix_scale
mess_matrix_rowscalem

Here is the call graph for this function:



8.6.2.9 int mess_matrix_colsub (mess_matrix *input*, mess_int_t *scol*, mess_int_t *ecol*, mess_matrix *out*)

Parameters

in	*input*	input matrix
in	*scol*	input start column of the block
in	*ecol*	input end column of the block
out	*out*	output matrix

Returns

zero on success or a non zero error code otherwise

The mess_matrix_colsub function gets a column submatrix out of a given matrix.
It is equal to MATLAB

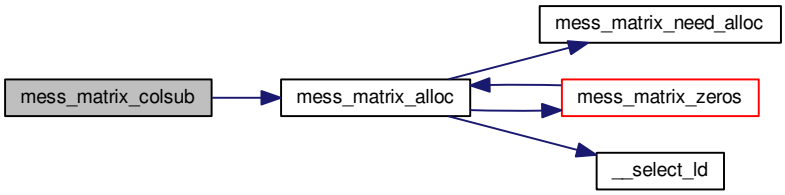
```
out=input(:,scol:ecol)
```

.

Attention

This function does not yet support the MESS_COORD storage type.

Here is the call graph for this function:



8.6.2.10 int mess_matrix_complex_from_parts (mess_matrix *xr*, mess_matrix *xc*, mess_matrix *x*)

Parameters

in	*xr*	input real part xr
in	*xc*	input complex part xc
out	*x*	output complex matrix x

Returns

zero on success or a non zero error code otherwise

The mess_matrix_complex_from_parts function creates a complex matrix from separate real and imaginary parts. Mathematically this means

$$x = xr + xc \cdot i.$$

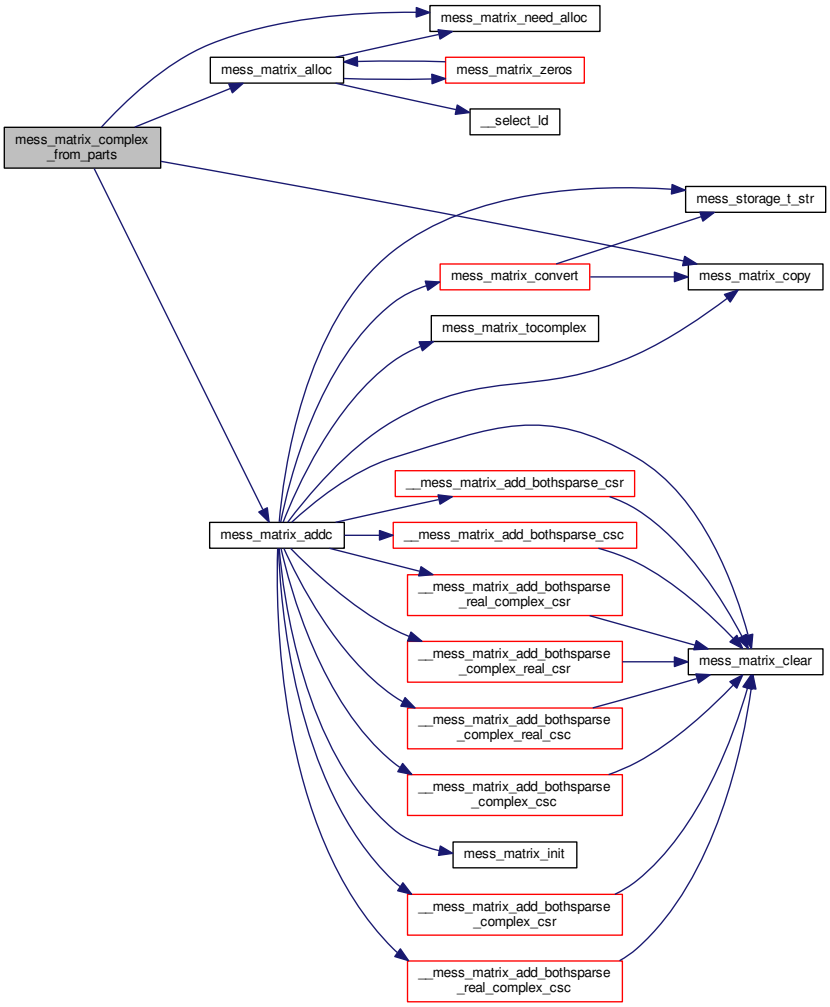
Attention

If the matrices xr and xc are sparse they need to have the same pattern structure otherwise the result will be completely wrong.

See also

mess_matrix_addc
mess_matrix_scale

Here is the call graph for this function:



8.6.2.11 int mess_matrix_convert (mess_matrix *input*, mess_matrix *output*, mess_storage_t *outtype*)

Parameters

in	*input*	input matrix
out	*output*	output matrix
in	*outtype*	input outputtype

Returns

zero on success or a non zero error code otherwise

The mess_matrix_convert functions converts a given matrix to an other storage scheme.
It supports dense and sparse matrices in various formats. If the input matrix has the same storage format than the desired one, it is copied instead of being converted. If a matrix is converted to dense it will have the default leading dimension of mess_matrix_alloc. The support for user supplied matrices is integrated but not used in **M.E.S.S.** at the moment.

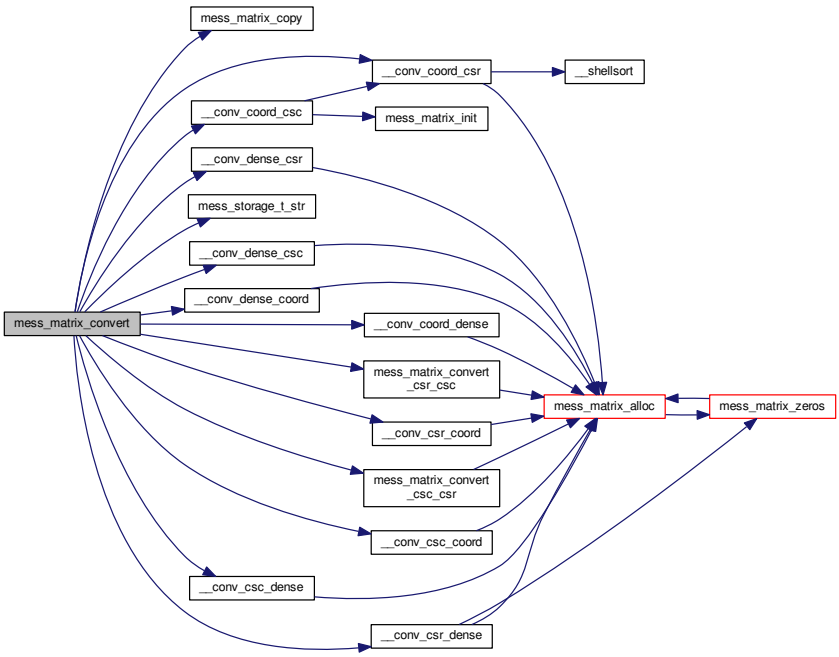
Supported converters:

- CSR → CSC
- CSR → DENSE
- CSR → COORD
- CSC → CSR
- CSC → DENSE
- CSC → COORD
- COORD → CSR
- COORD → CSC
- COORD → DENSE
- DENSE → CSR
- DENSE → CSC
- DENSE → COORD

See also

mess_matrix_alloc
mess_matrix_read
mess_matrix_read_formated

Here is the call graph for this function:



8.6.2.12 int mess_matrix_convert_csc_csr (mess_matrix *inmatrix*, mess_matrix *outmatrix*)

Parameters

in	*inmatrix*	input matrix
out	*outmatrix*	output matrix

Returns

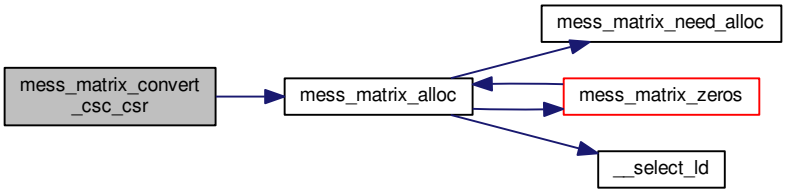
zero on success or a non zero error code otherwise

The mess_matrix_convert_csc_csr converts a Compressed Sparse Column matrix to a Compressed Sparse Row matrix.

See also

mess_matrix_convert_csr_csc
mess_matrix_convert

Here is the call graph for this function:



8.6.2.13 int mess_matrix_convert_csr_csc (mess_matrix *inmatrix*, mess_matrix *outmatrix*)

Parameters

in	*inmatrix*	input matrix
out	*outmatrix*	output matrix

Returns

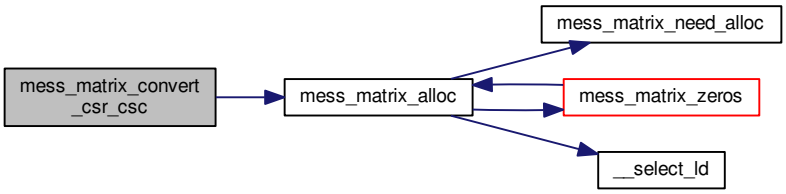
zero on success or a non zero error code otherwise

The mess_matrix_convert_csr_csc converts a Compressed Sparse Row matrix to a Compressed Sparse Column matrix.

See also

mess_matrix_convert_csc_csr
mess_matrix_convert

Here is the call graph for this function:



8.6.2.14 int mess_matrix_coord (mess_matrix *matrix*, mess_int_t *rows*, mess_int_t *cols*, mess_int_t *nnz*, mess_int_t *onebased*, mess_int_t * *rowptr*, mess_int_t * *colptr*, double * *values*, mess_double_cpx_t * *values_cpx*)

Parameters

out	*matrix*	output matrix which should be created from the given data
in	*rows*	input number of rows
in	*cols*	input number of columns
in	*nnz*	input number of non zero elements
in	*onebased*	input flag to identify one based indexing
in	*rowptr*	input array containing row pointers
in	*colptr*	input array containing column pointers
in	*values*	input array containing real values
in	*values_cpx*	input array containing complex values

Returns

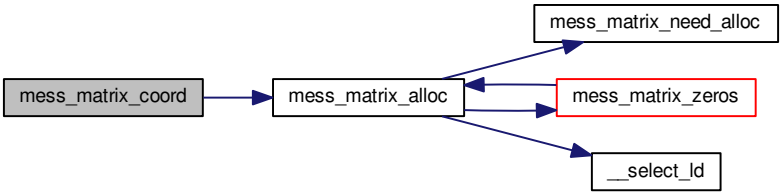
zero on success or a non zero error code

The mess_matrix_coord function creates a MESS_COORD mess_matrix out of the given arrays.
If the `values` array is `NULL`, the `values_cpx` array is used and the generated matrix is complex. If both value pointer are `NULL` an error is returned. If both are not `NULL` the complex data is used.

See also

mess_matrix_csr
mess_matrix_csc
mess_matrix_dense_from_farray
mess_matrix_dense_from_carray

Here is the call graph for this function:



8.6.2.15 int mess_matrix_copy (mess_matrix *in*, mess_matrix *out*)

Parameters

in	*in*	input source matrix
out	*out*	destination matrix

Returns

zero on success or a non zero error code otherwise

The mess_matrix_copy function copies a matrix from in to out.
If a view of a matrix is copied, the leading dimension changes to the default of mess_matrix_alloc.

8.6.2.16 int mess_matrix_csc (mess_matrix *matrix*, mess_int_t *rows*, mess_int_t *cols*, mess_int_t * *rowptr*, mess_int_t * *colptr*, double * *values*, mess_double_cpx_t * *values_cpx*)

Parameters

out	*matrix*	output matrix which should be created from the given data
in	*rows*	input number of rows
in	*cols*	input number of columns
in	*rowptr*	input array containing row pointers
in	*colptr*	input array containing column pointers
in	*values*	input array containing real values
in	*values_cpx*	input array containing complex values

Returns

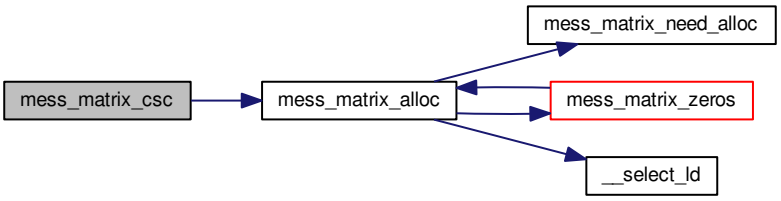
zero on success or a non zero error code

The mess_matrix_csc function creates a MESS_CSC Matrix Operations out of the given arrays.
Depending on the first element in the column pointer array `colptr` it detects if zero or one based indexing is used.
The last entry of the column pointer array `colptr`[cols] contains the number of non zero elements of the matrix.
If the `values` array is `NULL`, the `values_cpx` array is used and the generated matrix is complex. If both value pointer are `NULL` an error is returned. If both are not `NULL` the complex data is used.

See also

mess_matrix_csr
mess_matrix_coord
mess_matrix_dense_from_farray
mess_matrix_dense_from_carray

Here is the call graph for this function:



8.6.2.17 int mess_matrix_csr (mess_matrix *matrix*, mess_int_t *rows*, mess_int_t *cols*, mess_int_t * *rowptr*, mess_int_t * *colptr*, double * *values*, mess_double_cpx_t * *values_cpx*)

Parameters

out	*matrix*	output matrix which should be created from the given data
in	*rows*	input number of rows
in	*cols*	input number of columns
in	*rowptr*	input array containing row pointers
in	*colptr*	input array containing column pointers
in	*values*	input array containing real values
in	*values_cpx*	input array containing complex values

Returns

zero on success or a non zero error code

The mess_matrix_csr function creates a MESS_CSR mess_matrix out of the given arrays.
Depending on the first element in the row pointer array `rowptr` it detects if zero or one based indexing is used.
The last entry of the row pointer array `rowptr`[rows] contains the number of non zero elements of the matrix. If the `values` array is `NULL`, the `values_cpx` array is used and the generated matrix is complex. If both value pointer are `NULL` an error is returned. If both are not `NULL` the complex data is used.

See also

mess_matrix_csc
mess_matrix_coord
mess_matrix_dense_from_farray
mess_matrix_dense_from_carray

Here is the call graph for this function:

8.6.2.18 int mess_matrix_dense_from_carray (mess_matrix *mat*, mess_int_t *rows*, mess_int_t *cols*, double ** *realv*, mess_double_cpx_t ** *complexv*)

Parameters

out	*mat*	output matrix to copy the data to
in	*rows*	input number of rows of the matrix
in	*cols*	input number of columns of the matrix
in	*realv*	input 2 D-array with real values
in	*complexv*	input 2 D-array with complex values

Returns

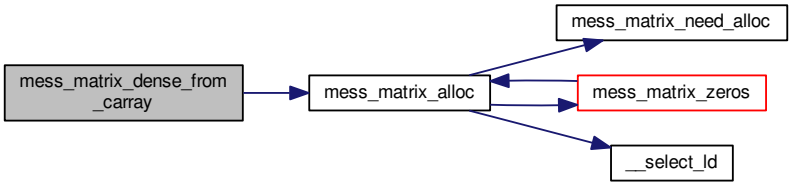
zero on success or a non zero error code otherwise

The mess_matrix_dense_from_carray function copies a C 2-dimensional array(row-major, double** pointer) matrix to a mess_matrix in MESS_DENSE format.
Depending whether `realv` or `complexv` is given a real matrix or a complex matrix is created. If both are given the complex data is used.
The leading dimension of the mess_matrix is determined by mess_matrix_alloc.

See also

mess_matrix_csc
mess_matrix_csr
mess_matrix_coord
mess_matrix_dense_from_farray

Here is the call graph for this function:



8.6.2.19 int mess_matrix_dense_from_farray (mess_matrix *mat,* mess_int_t *rows,* mess_int_t *cols,* mess_int_t *ld,* double * *realv,* mess_double_cpx_t * *complexv*)

Parameters

out	*mat*	output matrix to copy the data to
in	*rows*	input number of rows of the matrix
in	*cols*	input number of columns of the matrix
in	*ld*	input leading dimension of the matrix
in	*realv*	input array with real values
in	*complexv*	input array with complex values

Returns

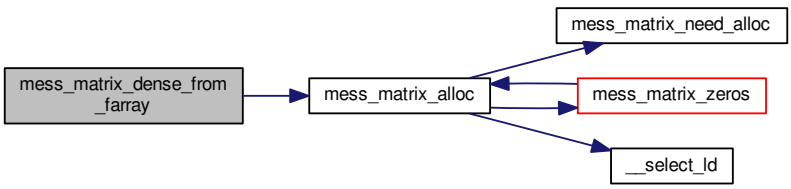
zero on success or a non zero error code otherwise

The mess_matrix_dense_from_farray function copies a `Fortran-like` (column-major) matrix to a mess_matrix in MESS_DENSE format.
If the provided leading dimension is smaller than the number of rows, the number of rows is used. Depending whether `realv` or `complexv` is given a real matrix or a complex matrix is created. If both are given the complex data is used.
The leading dimension of the mess_matrix is determined by mess_matrix_alloc.

See also

mess_matrix_csr
mess_matrix_coord
mess_matrix_dense_from_carray

Here is the call graph for this function:



8.6.2.20 int mess_matrix_diag (mess_matrix *matrix*, mess_vector *d*)

Parameters

in	*matrix*	input matrix
out	*d*	output vector with the main diagonal elements

Returns

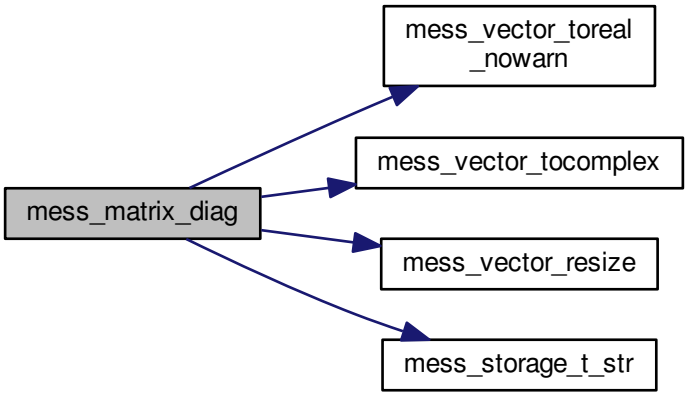
zero on success or a non-zero error value otherwise

The mess_matrix_diag function extracts the main diagonal elements of a matrix and copy them to d. If d has not the right size it is resized.

Attention

This function does not yet support the MESS_COORD storage type.

Here is the call graph for this function:



8.6.2.21 int mess_matrix_diag_from_vector (mess_vector *v*, mess_matrix *diag*)

Parameters

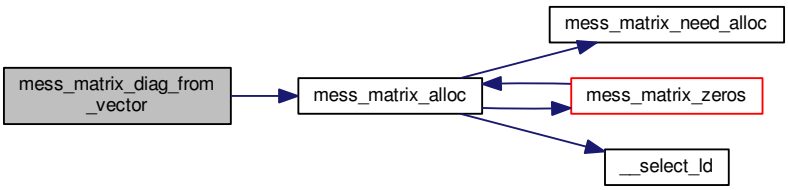
in	*v*	input vector with the main diagonal elements
in	*diag*	output matrix diagonal matrix with entries from v

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_diag_from_vector function creates a diagonal matrix with enties from v. A MESS_DENSE matrix is created and the datatype is determined by the datatype of v.

Here is the call graph for this function:



8.6.2.22 int mess_matrix_diagpos (mess_matrix *matrix*, mess_int_t * *pos*)

Parameters

in	*matrix*	input matrix
out	*pos*	output array containing the postion of the diagonal elements, at least `MIN(rows,cols)` elements long

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_diagpos function determines the position of the main diagonal elements in the values array. If an element does not exist $pos[e]$ is set to -1.

Attention

This function does not yet support the MESS_COORD storage type.

Here is the call graph for this function:

8.6.2.23 int mess_matrix_dupl (mess_matrix *mat*)

Parameters

in,out	*mat*	matrix from which the duplicate entries should be removed

Returns

0 on success or a non zero error code otherwise.

The mess_matrix_dupl function detects and sums up duplicate entries in CSR/CSC matrices, that means if there exists two or more entries in one row or column which have the same column or row indicies.
The function assumes that the data structures are sorted. If this is not fulfilled the matrix can be sorted using mess_matrix_sort before.

See also

mess_matrix_sort

8.6.2.24 int mess_matrix_eliminate_zeros (mess_matrix *mat*)

Parameters

in,out	*mat*	matrix from which the zero entries should be removed

Returns

0 on success or a non zero error code otherwise.

The mess_matrix_eliminate_zeros function detects and removes zero entries in CSR/CSC/COORD matrices.
Please note that a dense matrix is immediately returned.

8.6.2.25 int mess_matrix_eye (mess_matrix *matrix*, mess_int_t *rows*, mess_int_t *cols*, mess_storage_t *store*)

Parameters

out	*matrix*	output matrix
in	*rows*	input number of rows
in	*cols*	input number of columns
in	*store*	input storage type of the matrix

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_eye function generates a real identity matrix of size $(rows \times cols)$.

See also

mess_matrix_eyec

Here is the call graph for this function:

8.6.2.26 int mess_matrix_eyec (mess_matrix *matrix*, mess_int_t *rows*, mess_int_t *cols*, mess_storage_t *store*)

Parameters

out	*matrix*	output matrix
in	*rows*	input number of rows
in	*cols*	input number of columns
in	*store*	input storage type of the matrix

Returns

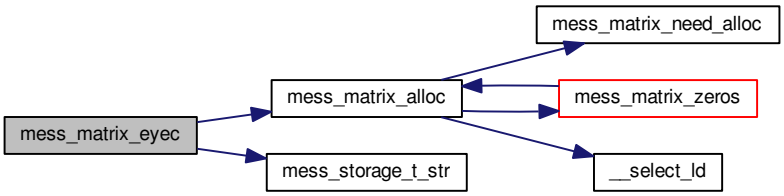
zero on success or a non-zero error value otherwise

The mess_matrix_eyec function generates a complex identity matrix of size $(rows \times cols)$.

See also

mess_matrix_eye

Here is the call graph for this function:



8.6.2.27 int mess_matrix_getcol (mess_matrix *matrix*, mess_int_t *col*, mess_vector *c*)

Parameters

in	*matrix*	input matrix
in	*col*	input number of the column to get
out	*c*	output vector containing col

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_getcol function gets a column from a matrix. The MATLAB equivalent is

```
c=A(:,col)
```

If you want to extract a column from a complex matrix the output vector is converted to complex, In the case of a MESS_CSR matrix this operation might be inefficient due to access limitations. The indexing is zero based.

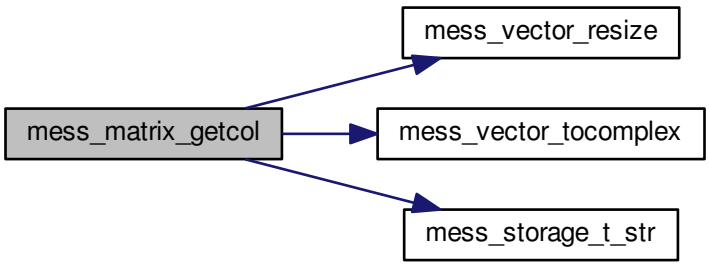
Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_setcol
mess_matrix_getrow

Here is the call graph for this function:



8.6.2.28 int mess_matrix_getelement (mess_matrix *matrix*, mess_int_t *row*, mess_int_t *col*, double * *val*, mess_double_cpx_t * *valc*)

Parameters

in	*matrix*	input matrix
in	*row*	input the row index
in	*col*	input the column index
out	*val*	output the real value, NULL if not need
out	*valc*	output the complex value if need, NULL if not wanted

Returns

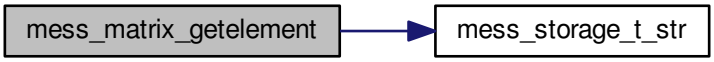
zero on success or a non zero error code otherwise

The mess_matrix_getelement function gets an element from a matrix.
Please do not use it for traversing the whole matrix because this getting slow especially on sparse matrices.
If the matrix is real and both `val` and `valc` pointer are given, the (`row`, `col`) entry of the matrix is stored to both.
If the matrix is complex and only the `val` argument is given, the real part of the entry is returned. If both `val` and `valc` are given and the matrix is complex the `valc` argument returns the complex entry and the `val` argument only the real part.

See also

mess_matrix_setelement_complex
mess_matrix_setelement

Here is the call graph for this function:



8.6.2.29 int mess_matrix_getrow (mess_matrix *matrix*, mess_int_t *row*, mess_vector *r*)

Parameters

in	*matrix*	input matrix
in	*row*	input number of the row to get
out	*r*	output vector containing row

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_getrow function gets a the `row` from a matrix. The `MATLAB` equivalent is

```
r=A(row,:)
```

.

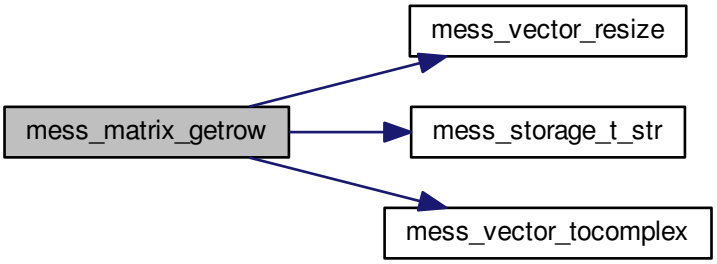
Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_setrow
mess_matrix_getcol

Here is the call graph for this function:



8.6.2.30 int mess_matrix_imagpart (mess_matrix *matrix*, mess_matrix *imagpart*)

Parameters

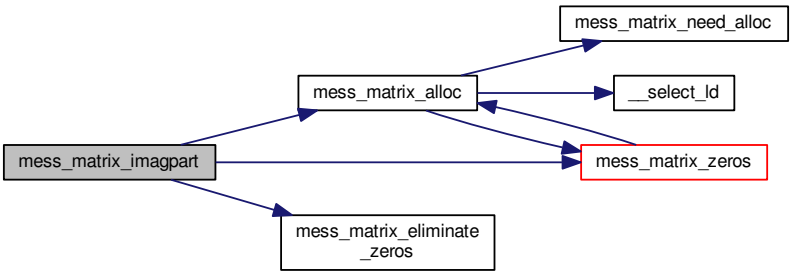
in	*matrix*	input matrix
out	*imagpart*	matrix with the imaginary part

Returns

zero on success or a non zero error code otherwise

The mess_matrix_imagpart function copies the imaginary part of a given matrix to the output matrix. If the input is real the output is a zero matrix.

Here is the call graph for this function:



8.6.2.31 int mess_matrix_lift (mess_matrix *in*, mess_int_t *n*, mess_matrix *out*)

Parameters

in	*in*	input matrix
in	*n*	input positive number of rows of zeros to add
out	*out*	matrix

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_lift function adds a trailing block of `n` rows zeros at the lower of mess_matrix `in`. The result is in mess_matrix `out`.

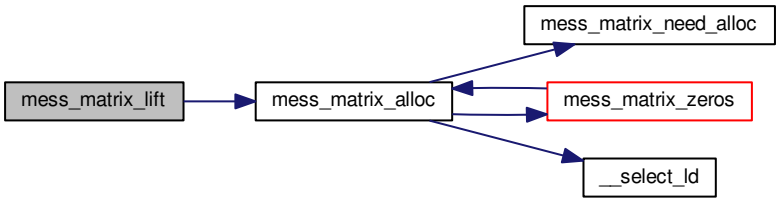
Attention

The function only works for dense matrices.

See also

mess_matrix_cat
mess_matrix_zeros

Here is the call graph for this function:



8.6.2.32 int mess_matrix_need_alloc (mess_matrix *matrix*, mess_int_t *rows*, mess_int_t *cols*, mess_int_t *nnz*, mess_storage_t *storetype*, mess_datatype_t *datatype*)

Parameters

in	*matrix*	input matrix to check
in	*rows*	input number of rows
in	*cols*	input number of cols
in	*nnz*	input number of nonzero elements (irrelevant for dense matrices)
in	*storetype*	input type of the storage format
in	*datatype*	input data type of the matrix

Returns

1 if the allocation is necessary, 0 otherwise.

The mess_matrix_need_alloc function checks if a new allocation of a matrix is necessary or not. That means the function checks if the matrix has already the internal structure that the corresponding mess_matrix_alloc call will return.
It can be used to skip an allocation if the matrix was already allocated previously.
It is used inside mess_matrix_alloc to reduce the number of allocations.

See also

mess_matrix_alloc

8.6.2.33 int mess_matrix_one_value (mess_matrix *matrix*, double *value*)

Parameters

in,out	*matrix*	input/output matrix to be overwritten
in	*value*	input value to be written to all matrix entries

Returns

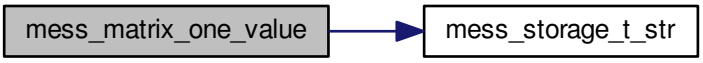
zero on success or a non zero error code otherwise

The mess_matrix_one_value function sets all entries in a given matrix `matrix` to a given value `value`.
The matrix needs to be preallocated. If you want to set a complex value use mess_matrix_one_valuec instead.

See also

mess_matrix_one_valuec

Here is the call graph for this function:



8.6.2.34 int mess_matrix_one_valuec (mess_matrix *matrix*, mess_double_cpx_t *value*)

Parameters

in,out	*matrix*	matrix to be overwritten
in	*value*	input value to be written to all matrix entries

Returns

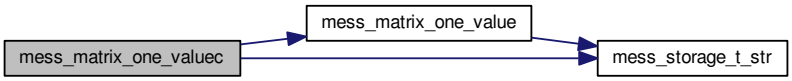
zero on success or a non zero error code otherwise

The mess_matrix_one_valuec function sets all entries in a given matrix $matrix$ to the given value $value$.
The matrix $matrix$ needs to be preallocated. The value is complex in this function. If the matrix is real only the real part of the value is set.
If you want to avoid this, convert the matrix to a complex one before.
For real matrices and a real value use mess_matrix_one_value instead.

See also

mess_matrix_one_value

Here is the call graph for this function:



8.6.2.35 int mess_matrix_ones (mess_matrix *matrix*)

Parameters

in,out	*matrix*	matrix to be set to one

Returns

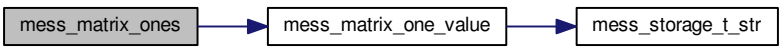
zero on success or a non zero error code otherwise

The mess_matrix_ones function sets all entries in a matrix to one.
The matrix needs to be preallocated.
It relies on mess_matrix_one_value.

See also

mess_matrix_one_value

Here is the call graph for this function:



8.6.2.36 int mess_matrix_rand (mess_matrix *mat*, mess_int_t *rows*, mess_int_t *cols*, mess_storage_t *storetype*, mess_datatype_t *dt*, double *p*)

Parameters

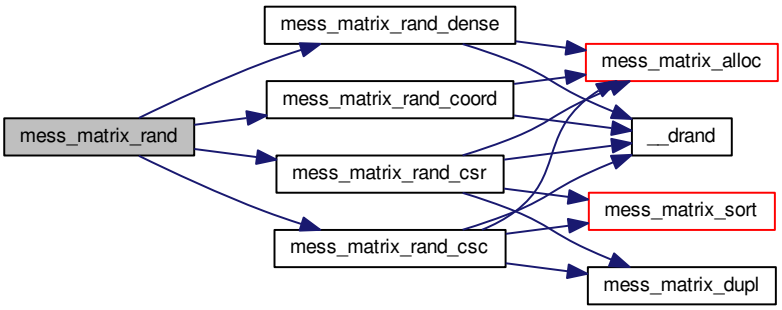
out	*mat*	output random matrix
in	*rows*	input number of rows
in	*cols*	input number of cols
in	*storetype*	input storage type of the output matrix
in	*p*	input fillrate for the rows or columns of a sparse matrix
in	*dt*	input the desired datatype

Returns

zero on success or a non zero error code otherwise

The mess_matrix_rand function calls depending on the `storetype` and `dt` argument the corresponding random function for sparse or dense, MESS_REAL or MESS_COMPLEX matrices.
The `p` parameter is ignored in the case of a dense matrix.

Here is the call graph for this function:



8.6.2.37 int mess_matrix_rand_coord (mess_matrix *mat*, mess_int_t *rows*, mess_int_t *cols*, double *p*, mess_datatype_t *dt*)

Parameters

out	*mat*	output random matrix
in	*rows*	input rows of the matrix
in	*cols*	input cols of the matrix
in	*p*	input percentage of non zero elements
in	*dt*	input the desired datatype

Returns

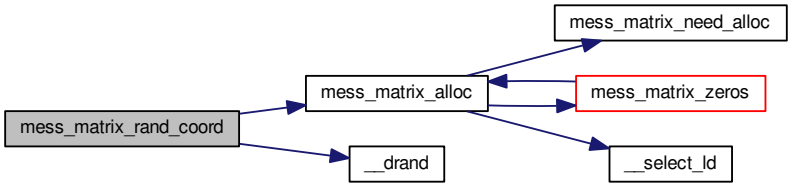
zero on success or a non zero error code otherwise

The mess_matrix_rand_coord function generates a MESS_REAL or MESS_COMPLEX random Compressed Sparse Row matrix with approximately $p \cdot cols$ entries per row.

See also

mess_matrix_rand_dense
mess_matrix_rand_csr
mess_matrix_rand_csc

Here is the call graph for this function:



8.6.2.38 int mess_matrix_rand_csc (mess_matrix *mat*, mess_int_t *rows*, mess_int_t *cols*, double *p*, mess_datatype_t *dt*)

Parameters

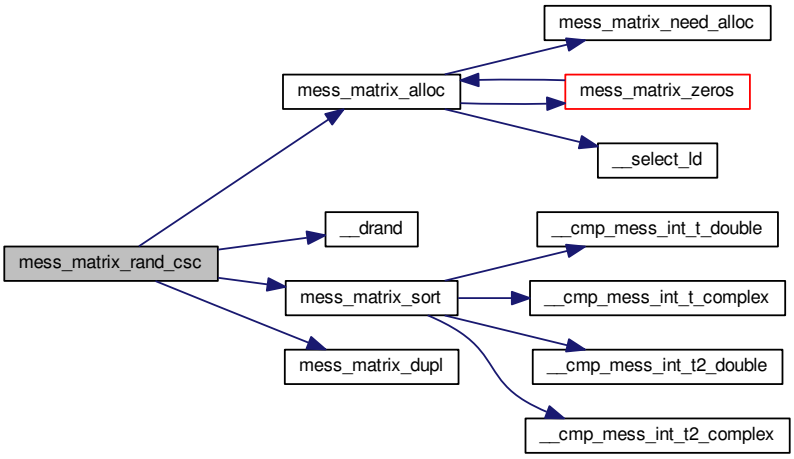
out	*mat*	output random matrix
in	*rows*	input rows of the matrix
in	*cols*	input cols of the matrix
in	*p*	input percentage of non zero elements
in	*dt*	input the desired datatype

Returns

zero on success or a non zero error code otherwise

The mess_matrix_rand_csc function generates a MESS_REAL or MESS_COMPLEX random MESS_CSC matrix with approximately $p \cdot rows$ entries per column.

Here is the call graph for this function:



8.6.2.39 int mess_matrix_rand_csc (mess_matrix *mat*, mess_int_t *rows*, mess_int_t *cols*, double *p*, mess_datatype_t *dt*)

Parameters

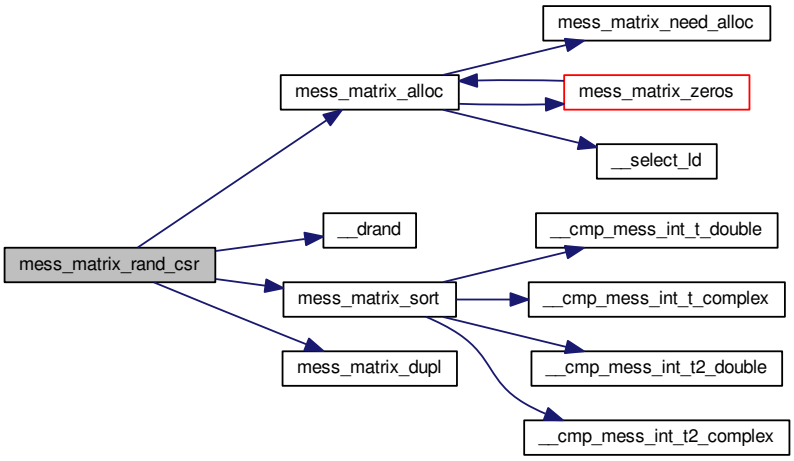
out	*mat*	output random matrix
in	*rows*	input rows of the matrix
in	*cols*	input cols of the matrix
in	*p*	input percentage of non zero elements
in	*dt*	input the desired datatype

Returns

zero on success or a non zero error code otherwise

The mess_matrix_rand_csr function generates a MESS_REAL or MESS_COMPLEX random MESS_CSR matrix with approximately $p \cdot cols$ entries per row.

Here is the call graph for this function:



8.6.2.40 int mess_matrix_rand_dense (mess_matrix *mat*, mess_int_t *rows*, mess_int_t *cols*, mess_datatype_t *dt*)

Parameters

out	*mat*	output random matrix
in	*rows*	input number of rows
in	*cols*	input number of cols
in	*dt*	input desired dataype

Returns

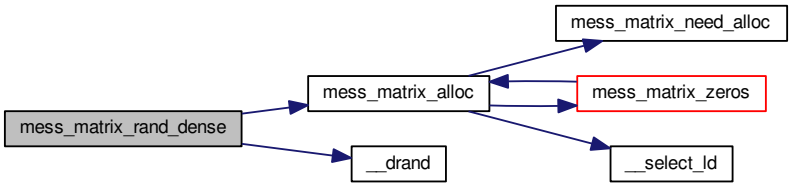
zero on success or a non zero error code otherwise

The mess_matrix_rand_dense function generates a MESS_REAL or MESS_COMPLEX random dense matrix mat.
The matrix is allocated by mess_matrix_alloc and gets the leading dimension from it.
The matrix is filled with values generated by `rand` from `libc`.

See also

mess_matrix_rand_dense_uniform

Here is the call graph for this function:



8.6.2.41 int mess_matrix_rand_dense_uniform (mess_matrix *mat*, mess_int_t * *seed*, mess_int_t *rows*, mess_int_t *cols*)

Parameters

out	*mat*	output random matrix
in,out	*seed*	input/output initial seed for the random number generator
in	*rows*	input number of rows
in	*cols*	input number of cols

Returns

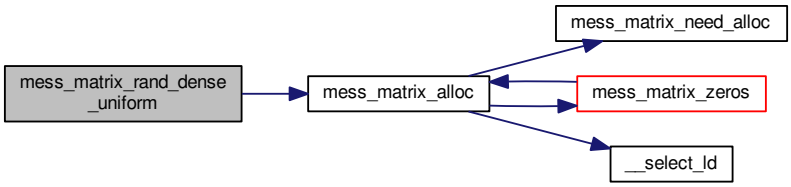
zero on success or a non zero error code otherwise

The mess_matrix_rand_dense_uniform function generates a random dense matrix mat with a uniform value distribution between $(-1, 1)$ generated by LAPACK DLANRV.
The $seed$ parameter is an integer array of length 4. The value of $seed[3]$ need to be an odd value, if not it will be adjusted. The $seed$ is updated during the call of DLANRV.

See also

mess_matrix_rand_dense

Here is the call graph for this function:



8.6.2.42 int mess_matrix_rand_init (mess_int_t * *seed*)

Parameters

in	*seed*	pointer to mess_int_t for given seed, if NULL current calender time is used.

Returns

zero in every case

The mess_matrix_rand_init initializes the rand function from libc for generating random numbers.
If seed points to NULL the current calender time is used for initialization of rand function. This function is usefull in combination with:

See also

mess_matrix_rand
mess_matrix_rand_dense
mess_matrix_rand_csc
mess_matrix_rand_csr
mess_matrix_rand_coord

8.6.2.43 int mess_matrix_realloc (mess_matrix *matrix*, mess_int_t *rows*, mess_int_t *cols*, mess_int_t *nnz*)

Parameters

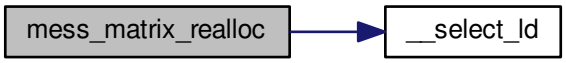
in,out	*matrix*	input/output matrix to be resized
in	*rows*	input new number of rows
in	*cols*	input new number of cols
in	*nnz*	input new number of nonzero elements

Returns

zero on success or a non zero error code otherwise

The mess_matrix_realloc function reallocates data structures for a matrix.
In case of a dense matrix the matrix is resized correctly in sense of cutting of the left upper $(1 : rows, 1 : cols)$ block.
If the matrix is a compressed sparse matrix only the values array and the corresponding index array is reallocated. Changing the dimension is not possible yet. This might change in the future.
In case of coordinate matrices it acts like the dense one and ignores the nnz argument if the new matrix is smaller than the original one.

Here is the call graph for this function:



8.6.2.44 int mess_matrix_realpart (mess_matrix *matrix*, mess_matrix *realpart*)

Parameters

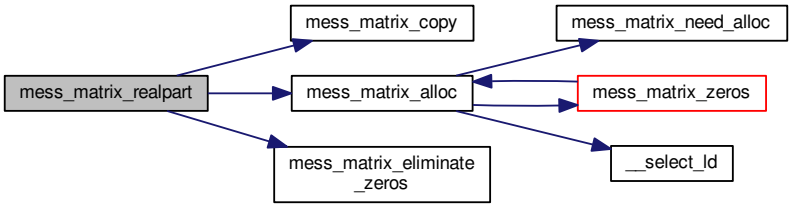
in	*matrix*	input matrix
out	*realpart*	matrix with the real part

Returns

zero on success or a non zero error code otherwise

The mess_matrix_realpart function copies the real part of a given matrix to the output matrix.
If the input is real the matrix is copied.

Here is the call graph for this function:



8.6.2.45 int mess_matrix_reset (mess_matrix *matrix*)

Parameters

in,out	*matrix*	input/output matrix object to reset

Returns

zeros on success or a non zero error value.

The mess_matrix_reset function resets a mess_matrix object that it behaves like a newly initialized one. In contrast to mess_matrix_clear it does only free the internal data and not the surrounding structure.

8.6.2.46 int mess_matrix_resize (mess_matrix *matrix*, mess_int_t *rows*, mess_int_t *cols*)

Parameters

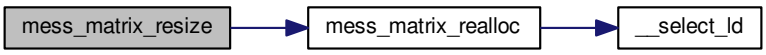
in,out	*matrix*	input/output matrix to be resized
in	*rows*	input new number of rows
in	*cols*	input new number of cols

Returns

zero on success or a non zero error code otherwise

The mess_matrix_resize function is only a wrapper for mess_matrix_realloc to get a mathematical correct name for it. The name realloc tends to be too computer science specific.
If the matrix is a sparse one, the number of non zeros entries is untouched. Otherwise it behaves exactly like the mess_matrix_realloc function and returns the typical error codes.

Here is the call graph for this function:



8.6.2.47 int mess_matrix_rowscalem (mess_vector *r*, mess_matrix *A*)

Parameters

in	*r*	input mess_vector with scaling vectors
in,out	*A*	matrix to scale

Returns

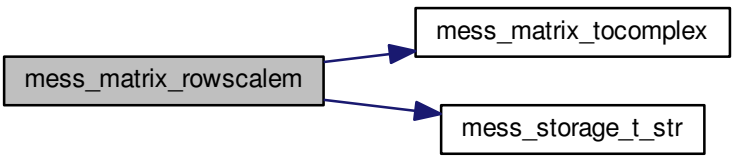
zero on success or a non zero error code otherwise

The mess_matrix_rowscalem function scales $i-th$ row of a matrix by the $i-th$ entry of r. mess_matrix::rows should be equal to mess_vector::dim.

See also

mess_matrix_scale
mess_matrix_colscalem

Here is the call graph for this function:



8.6.2.48 int mess_matrix_rowsub (mess_matrix *input*, mess_int_t *srow*, mess_int_t *erow*, mess_matrix *out*)

Parameters

in	*input*	input matrix
in	*srow*	input start row of the block
in	*erow*	input end row of the block
out	*out*	output matrix

Returns

zero on success or a non zero error code otherwise

The mess_matrix_rowsub function gets a row submatrix out of a given matrix.
It is equal to MATLAB

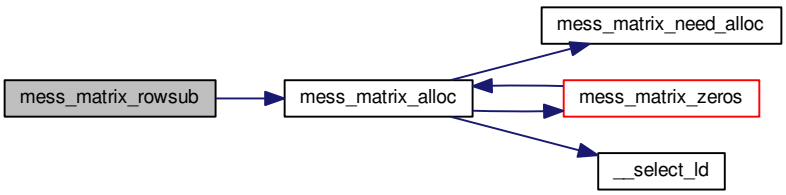
```
out=input(srow:erow,:)
```

.

Attention

This function does not yet support the MESS_COORD storage type.

Here is the call graph for this function:



8.6.2.49 int mess_matrix_setcol (mess_matrix *matrix*, mess_int_t *col*, mess_vector *colv*)

Parameters

in,out	*matrix*	input/output matrix where to set the column
in	*col*	input number of the column to set
in	*colv*	input vector to fill the column

Returns

zero on success or a non zero error code otherwise

The mess_matrix_setcol function overwrites a column of a matrix with a given vector. In the case that you want to copy a complex column in a real matrix, the complex information is lost and only the real part is copied.

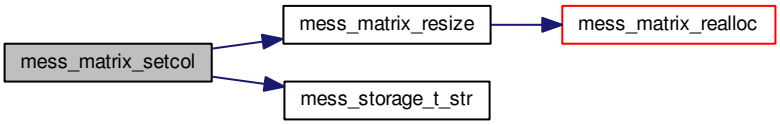
Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_getcol
mess_matrix_setrow

Here is the call graph for this function:



8.6.2.50 int mess_matrix_setelement (mess_matrix *matrix*, mess_int_t *row*, mess_int_t *col*, double *value*)

Parameters

in,out	*matrix*	matrix to set a value
in	*row*	input row index
in	*col*	input column index
in	*value*	input value to set at (row, col)

Returns

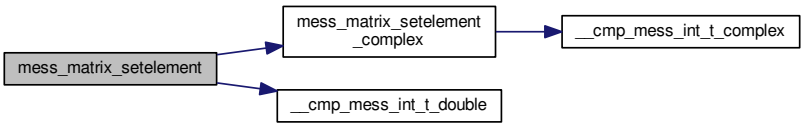
zero on success or a non zero error code otherwise

The mess_matrix_setelement function sets a value in a matrix.
This function should not be used to assemble a sparse matrix because of performance problems.

See also

mess_matrix_setelement_complex
mess_matrix_getelement

Here is the call graph for this function:



8.6.2.51 int mess_matrix_setelement_complex (mess_matrix *matrix*, mess_int_t *row*, mess_int_t *col*, mess_double_cpx_t *value*)

Parameters

in,out	*matrix*	matrix to set a value
in	*row*	input row index
in	*col*	input column index
in	*value*	input value to set at (row, col)

Returns

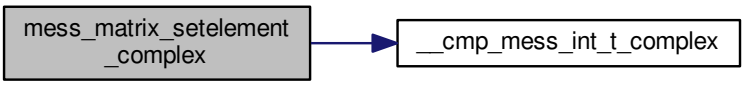
zero on success or a non zero error code otherwise

The mess_matrix_setelement_complex function sets a value in a matrix.
This function should not be used to assemble a sparse matrix because of performance problems.
The input need to be a complex matrix. Otherwise it has to be converted to a complex one before.

See also

mess_matrix_setelement
mess_matrix_getelement

Here is the call graph for this function:



8.6.2.52 int mess_matrix_setrow (mess_matrix *matrix*, mess_int_t *row*, mess_vector *rowv*)

Parameters

in,out	*matrix*	input/output matrix where to set the column
in	*row*	input number of the column to set
in	*rowv*	input vector to fill the column

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_setrow function overwrites a row of a matrix with a given vector.

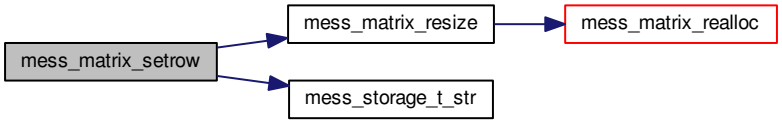
Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_setcol
mess_matrix_getrow

Here is the call graph for this function:



8.6.2.53 int mess_matrix_sort (mess_matrix *mat*)

Parameters

in,out	*mat*	matrix

Returns

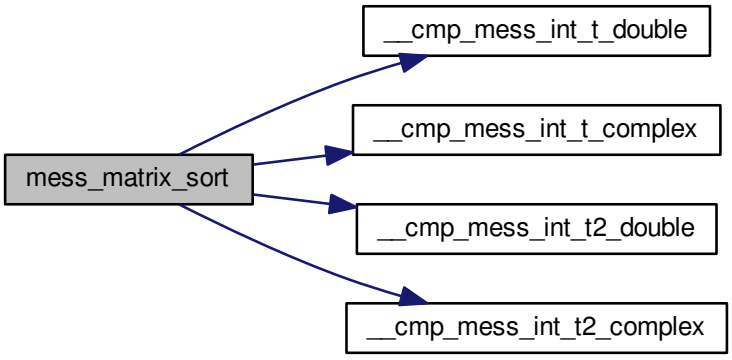
zero on success or a non-zero error value otherwise

The mess_matrix_sort function sorts the internal structure of a sparse matrix.
It uses the qsort function provided by libc.
Sorting means that the row or column indices in a compressed sparse matrix appear in ascending order. This is necessary for many algorithms to work correctly or more efficient. In case of a matrix-vector product it helps to get a better data locality.

See also

mess_matrix_dupl

Here is the call graph for this function:



8.6.2.54 int mess_matrix_sub (mess_matrix *in*, mess_int_t *rowS*, mess_int_t *rowE*, mess_int_t *colS*, mess_int_t *colE*, mess_matrix *out*)

Parameters

in	*in*	input matrix
in	*rowS*	input starting row
in	*rowE*	input end row
in	*colS*	input stating column
in	*colE*	input end column
out	*out*	ouput block of the matrix

Returns

zero on success or a non zero error code otherwise

The mess_matrix_sub function gets a sub matrix out of the matrix in.
It works like

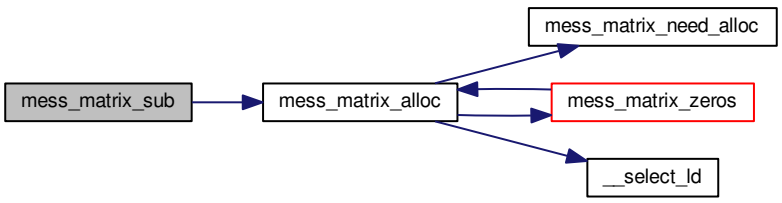
```
out = in (rowS:rowE,colS:colE)
```

in MATLAB .

Attention

This function does not yet support the MESS_COORD storage type.

Here is the call graph for this function:



8.6.2.55 int mess_matrix_symfillup (mess_matrix *mat*)

Parameters

in,out	*mat*	matrix to fill up

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_symfillup function fills up the symmetry in a given symmetric matrix. It does not check if the not stored triangle exists.
It only works if the matrix is stored as coordinate one and the SYMMETRY flag is set in the data structure.
It is mostly called from the mess_matrix_read functions to output a general matrices without any stored symmetry.

See also

mess_matrix_read

8.6.2.56 int mess_matrix_tocomplex (mess_matrix *m*)

Parameters

in,out	*m*	matrix to be changed to complex

Returns

zero on success or a non zero error code otherwise

The mess_matrix_tocomplex function converts an arbitrary matrix to a complex matrix.
If the matrix is already complex nothing will change.
If the matrix is real the whole values array will be copied to a complex array.
If the matrix is a view on to an other matrix, the parent matrix will be converted to a complex one.

8.6.2.57 int mess_matrix_toreal (mess_matrix *m*)

Parameters

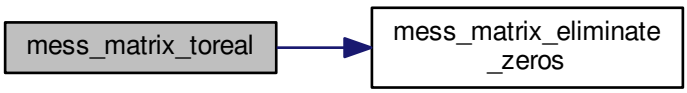
in,out	*m*	matrix to be changed to real

Returns

zero on success or a non zero error code otherwise

The mess_matrix_toreal function converts an arbitrary matrix to a real matrix.
If the matrix is already real nothing will change.
If the matrix is complex all imaginary parts are cut of.
If the matrix is a view on to an other matrix, the parent matrix will be converted to a complex one.

Here is the call graph for this function:



8.6.2.58 int mess_matrix_totype (mess_matrix *mat*, mess_datatype_t *dt*)

Parameters

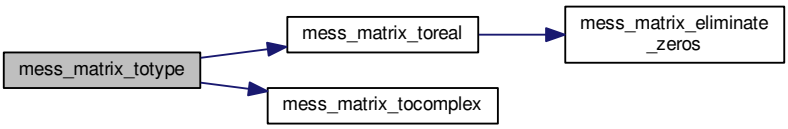
in,out	*mat*	matrix to be converted
in	*dt*	input desired output type

Returns

zero on success or a non zero error code otherwise

The mess_matrix_totype function converts a matrix to a specified data type.
It is a wrapper around mess_matrix_toreal and mess_matrix_tocomplex.

Here is the call graph for this function:



8.6.2.59 int mess_matrix_tridiag (mess_matrix *matrix*, mess_int_t *rows*, mess_int_t *cols*, mess_storage_t *store_t*, mess_datatype_t *data_t*, mess_double_cpx_t *lower*, mess_double_cpx_t *diag*, mess_double_cpx_t *upper*)

Parameters

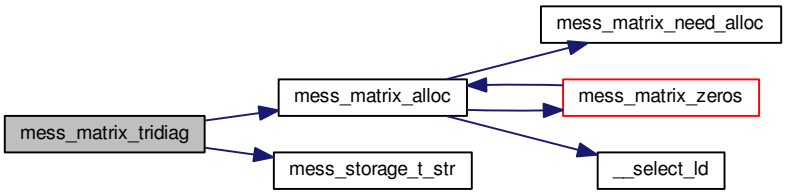
in,out	*matrix*	input/output matrix to be overwritten
in	*rows*	input number of rows
in	*cols*	input number of columns
in	*store↩ _t*	input desired storage type
in	*data↩ _t*	input desired data type
in	*lower*	input value for lower diagonal
in	*diag*	input value for main diagonal
in	*upper*	input value for upper diagonal

Returns

zero on success or a non zero error code otherwise

The mess_matrix_tridiag function creates a tridigonal `matrix` using the values `lower`, `diag`, `upper`.
If a real matrix is wanted then the imaginary part of `lower`, `diag`, `upper` is ignored.

Here is the call graph for this function:



8.6.2.60 int mess_matrix_tril (mess_matrix *mat*, mess_int_t *k*)

Parameters

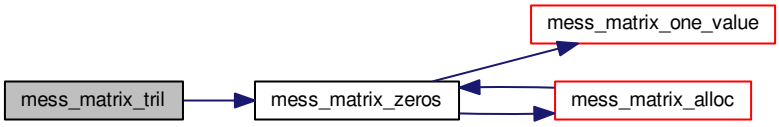
in,out	*mat*	input/output MESS_DENSE mess_matrix.
in	*k*	input scalar specifies the k-th diagonal of `mat`.

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_tril function sets everything above the `k-th` diagonal to zero. Only MESS_DENSE matrices are supported. If `k` is zero everything above the main diagonal is set to zero. if `k` is positive/negative everything above the `k` -th upper/lower main diagonal is set to zero.

Here is the call graph for this function:



8.6.2.61 int mess_matrix_triu (mess_matrix *mat*, mess_int_t *k*)

Parameters

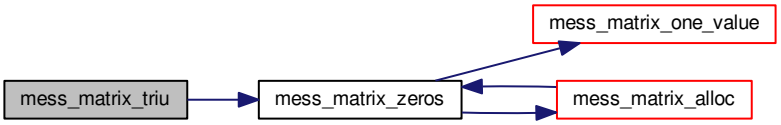
in,out	*mat*	input/output MESS_DENSE mess_matrix.
in	*k*	input scalar specifies the k-th diagonal of `mat`.

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_triu function sets everything below the `k-th` diagonal to zero. Only MESS_DENSE matrices are supported. If `k` is zero everything below the main diagonal is set to zero. if `k` is positive/negative everything below the `k` -th upper/lower main diagonal is set to zero.

Here is the call graph for this function:



8.6.2.62 int mess_matrix_view (mess_matrix *original*, mess_int_t *srow*, mess_int_t *scol*, mess_int_t *rows*, mess_int_t *cols*, mess_matrix *view*)

8.6.2.63 int mess_matrix_zeros (mess_matrix *matrix*)

Parameters

`in,out`	*matrix*	matrix to be set to zero

Returns

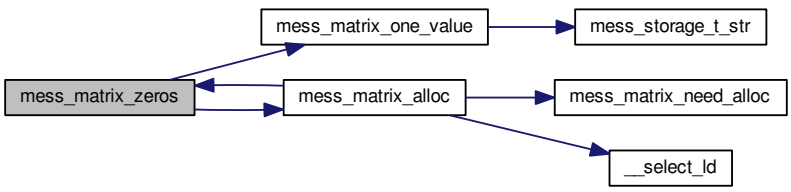
zero on success or a non zero error code otherwise

The mess_matrix_zeros function sets all entries in a matrix to zero.
The matrix needs to be preallocated.
It relies on mess_matrix_one_value.

See also

mess_matrix_one_value

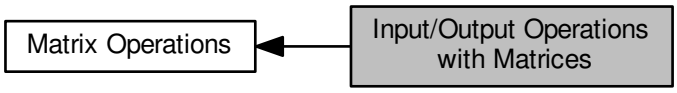
Here is the call graph for this function:



8.7 Input/Output Operations with Matrices

This category contains all functions to read, write and print matrices.

Collaboration diagram for Input/Output Operations with Matrices:



Functions

- int mess_matrix_read (const char *filename, mess_matrix matrix)

  Read a matrix from a file into a mess_matrix structure.
- int mess_matrix_write (const char *filename, mess_matrix matrix)

  Write a mess_matrix to a `MatrixMarket` *file.*
- int mess_matrix_read_formated (const char *filename, mess_matrix matrix, mess_storage_t format)

  Read a matrix from a file and convert it to the desired format.
- int mess_matrix_display (mess_matrix matrix)

  Display a mess_matrix to stdout.
- int mess_matrix_print (mess_matrix matrix)

  Print a mess_matrix to stdout.
- int mess_matrix_printshort (mess_matrix matrix)

  Print the first values of a matrix to stdout.
- int mess_matrix_printdata (mess_matrix mat)

  Print underlying data structures of a matrix to stdout.
- int mess_matrix_printinfo (mess_matrix mat)

  Print basic information of a matrix to the stdout.

8.7.1 Detailed Description

The default file format for matrices is the `MatrixMarket` file format. It can be read using mess_matrix_read or mess_matrix_read_formated. If you want to save a matrix as a `MatrixMarket` file you can use mess_matrix↩_write. If `zlib`, `bzip2` or `lzma` are found on the system you can read gz-, bz2-, or xz-compressed `Matrix↩Market` files too. If you write a file which ends with .gz it will be compressed using `gzip`. The same works with bz2 and xz.

The mess_matrix_print function and their variants can be used to print matrices to the screen.

See also

Various Other Operations with Vectors

8.7.2 Function Documentation

8.7.2.1 int mess_matrix_display (mess_matrix *matrix*)

Parameters

in	*matrix*	input the matrix to display

Returns

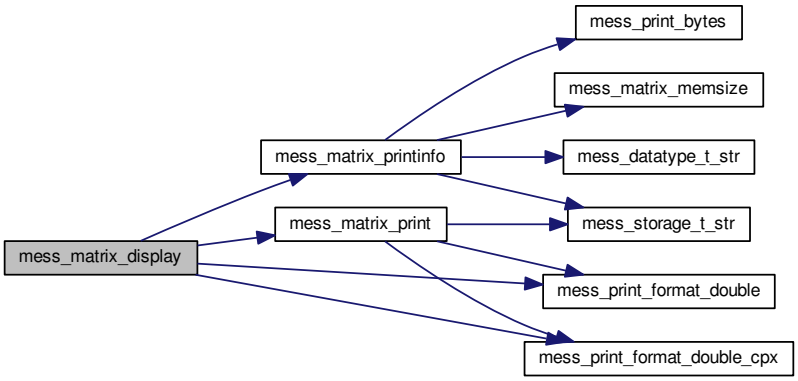
zero on success or a non zero error code otherwise

The mess_matrix_display function prints a complete matrix to the standard output.
In contrast to mess_matrix_print the output is in a rectangular and better readable form. The function uses the `csc_io_get_term_width` function to retrieve the width of the terminal. In case of a sparse matrix mess_matrix_print is called. We use zero-based counting for indices.

See also

mess_matrix_printshort
mess_matrix_printinfo
mess_matrix_printdata

Here is the call graph for this function:



8.7.2.2 int mess_matrix_print (mess_matrix *matrix*)

Parameters

in	*matrix*	input the matrix to print out

Returns

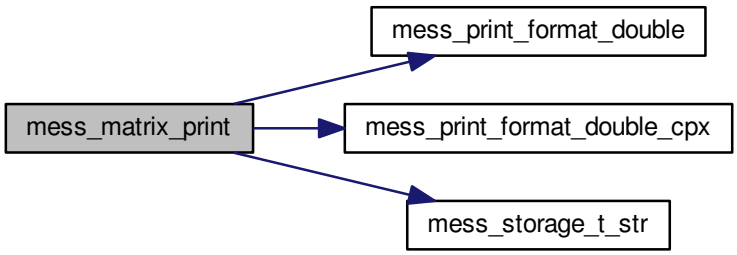
zero on success or a non zero error code otherwise

The mess_matrix_print function prints a complete matrix to the standard output.
The output format is almost the same as in MATLAB .
In the case of sparse matrices only the non zero entries are printed. We use zero-based counting for indices.

See also

mess_matrix_printshort
mess_matrix_printinfo
mess_matrix_printdata
mess_matrix_display

Here is the call graph for this function:



8.7.2.3 int mess_matrix_printdata (mess_matrix *mat*)

Parameters

in	*mat*	input the matrix to print

Returns

zero on success, a non zero error code otherwise

The mess_matrix_printdata prints underlying data structures of a matrix to the standard output. Depending on the storage type the printed information differs. We use zero-based counting for indices.

See also

mess_matrix_printshort
mess_matrix_printinfo
mess_matrix_print
mess_matrix_display

8.7.2.4 int mess_matrix_printinfo (mess_matrix *matrix*)

Parameters

in	*matrix*	input the matrix to print

Returns

zero on success, a non zero error code otherwise

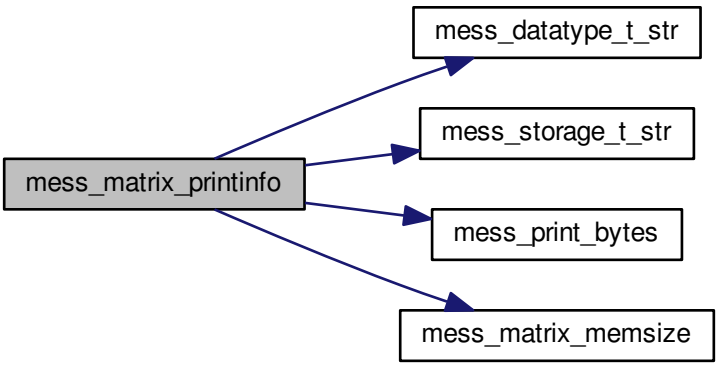
The mess_matrix_printinfo function prints basic information of a matrix to the standard output. These are

- size,
- number of non zero elements,
- leading dimension,
- data type,
- ... .

See also

mess_matrix_printshort
mess_matrix_print
mess_matrix_printdata
mess_matrix_display

Here is the call graph for this function:



8.7.2.5 int mess_matrix_printshort (**mess_matrix** *matrix*)

Parameters

in	*matrix*	input the matrix to print

Returns

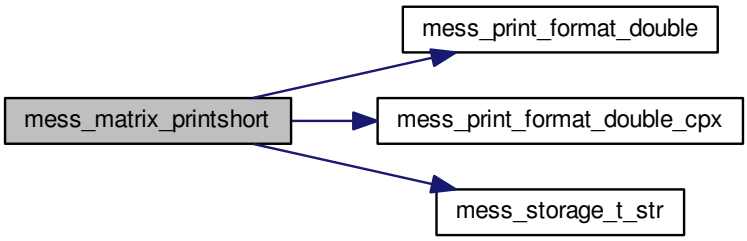
zero on success, a non zero error code otherwise

The mess_matrix_printshort function prints the first 40 values of a matrix to the standard output.
The output format is almost the same as in MATLAB .
In case of sparse matrices only the non zero entries are printed to the screen. We use zero-based counting for indices.

See also

mess_matrix_print
mess_matrix_printinfo
mess_matrix_printdata
mess_matrix_display

Here is the call graph for this function:



8.7.2.6 int mess_matrix_read (const char * *filename*, **mess_matrix** *matrix*)

Parameters

in	*filename*	input file to be read
out	*matrix*	structure storing all information

Returns

zero on success or a non zero error code otherwise

The mess_matrix_read function reads a matrix from a file.
The supported file formats are MatrixMarket, assembled Harwell-Boeing and MATLAB (if **M.E.S.S.** is compiled with MAT File I/O support).
If **M.E.S.S.** supports zlib and bzip2, files can be compressed as well and it is automatically detected if it is a compressed file or not.
In the case of a MATLAB file, the filename need to be formed like

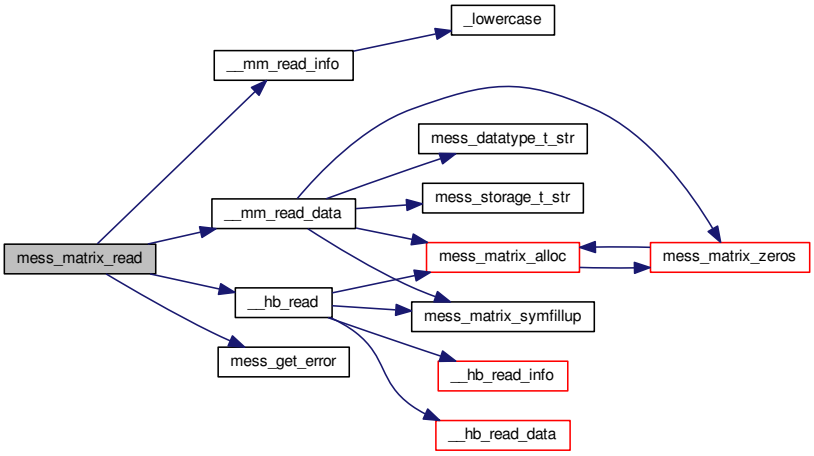
FILENAME:VARIABLENAME

in order to select the variable to read.
If the matrix is sparse, the output will be a coordinate matrix.
Dense matrices are read directly into a dense one.
If a matrix has a symmetry property the missing parts are filled up and the matrix becomes a general full stored one.
If a matrix should be read to a given storage format use mess_matrix_read_formated instead.

See also

mess_matrix_write
mess_matrix_symfillup

Here is the call graph for this function:



8.7.2.7 int mess_matrix_read_formated (const char * *filename*, mess_matrix *matrix*, mess_storage_t *format*)

Parameters

in	*filename*	input file to read
out	*matrix*	matrix to save the read data
in	*format*	input desired format

Returns

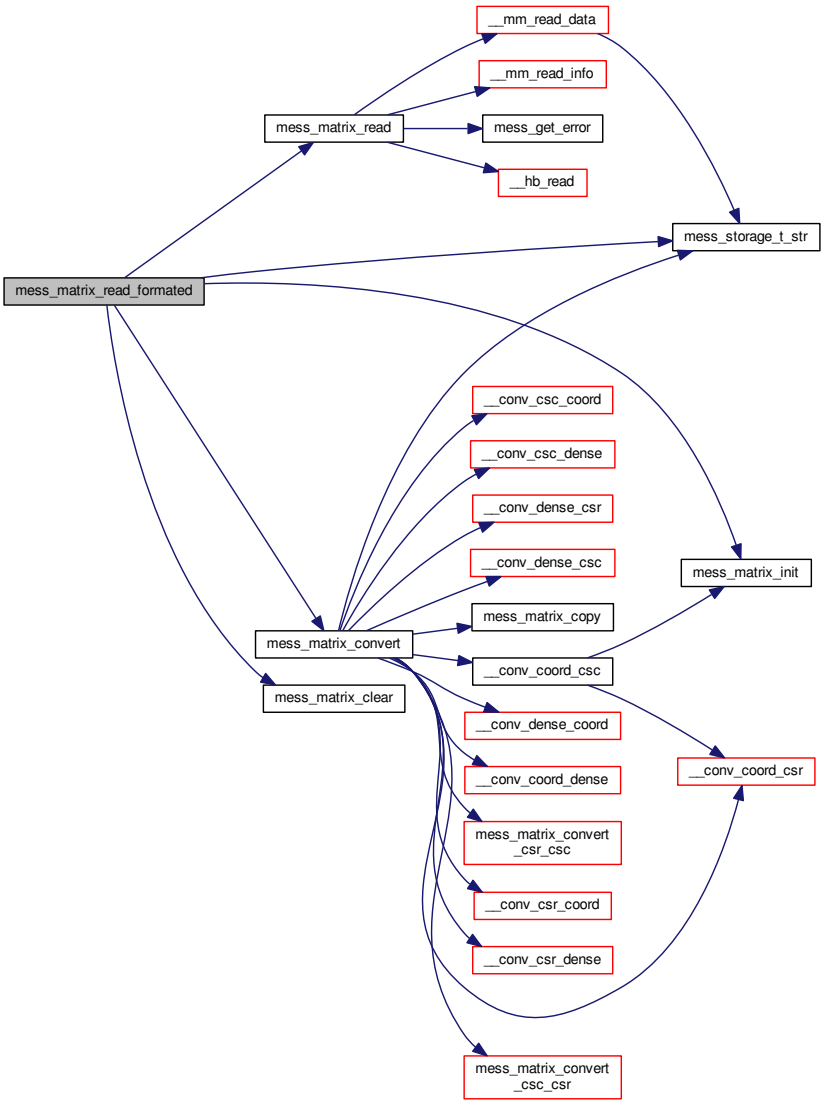
zero on success or a non-zero error value otherwise

The mess_matrix_read_formated function reads a matrix from a file and converts it to the desired format.
It works like mess_matrix_read and mess_matrix_convert together.

See also

mess_matrix_convert
mess_matrix_read

Here is the call graph for this function:



8.7.2.8 int mess_matrix_write (const char * *filename*, mess_matrix *matrix*)

Parameters

in	*filename*	input filename to write matrix
in	*matrix*	input mess_matrix to save to file

Returns

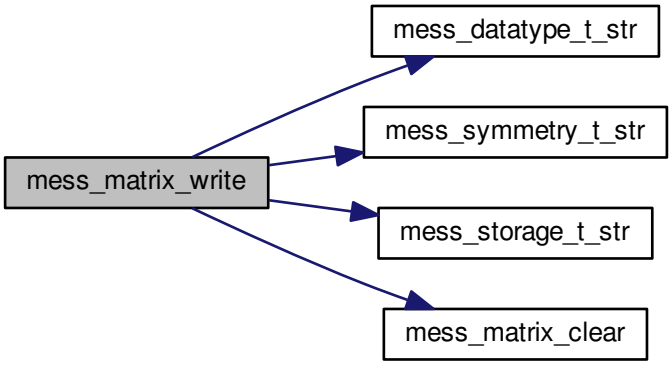
zero on success or a non zero error code

The mess_matrix_write function writes a mess_matrix to a `MatrixMarket` file.
If the file is sparse, the `MatrixMarket` file is in coordinate storage otherwise it is in dense storage.
If **M.E.S.S.** is configured to use `zlib`, `bzip2` or `lzma` you can write compressed files by adding .gz, .bz2 or .xz as file extension.

See also

mess_matrix_read

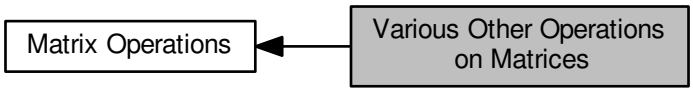
Here is the call graph for this function:



8.8 Various Other Operations on Matrices

This category contains all matrix related functions that fit not well in one of the other categories.

Collaboration diagram for Various Other Operations on Matrices:



Enumerations

Functions

- mess_int_t mess_matrix_memsize (mess_matrix matrix)

  Return the size of a matrix in bytes.
- mess_int_t mess_matrix_memsize_nnz (mess_int_t rows, mess_int_t cols, mess_int_t nnz, mess_int_t type, mess_int_t dtype)

  Determine memory size of a matrix from parameters.
- int mess_matrix_spy (mess_matrix matrix, const char *filename, int width, int height)

  Generate a bitmap file from the non zero structure of a matrix.
- int mess_matrix_reorder (mess_reorder_t reorder, mess_matrix matrix, mess_int_t *p, mess_int_t *q)

  Select and compute a matrix reordering.
- int mess_matrix_reorder_amd (mess_matrix matrix, mess_int_t *p)

  Compute `AMD` *reordering of a matrix.*
- int mess_matrix_reorder_colamd (mess_matrix matrix, mess_int_t *p)

  Compute the `COLAMD` *reordering of a matrix.*
- int mess_matrix_reorder_rcm (mess_matrix A, mess_int_t *perm)

  Compute the Reverse Cuthill-McKee reordering of a matrix.
- int mess_matrix_perm (mess_matrix matrix, mess_int_t *p, mess_int_t *q)

  Permute a general matrix.
- int mess_matrix_permcopy (mess_matrix matrix, mess_int_t *p, mess_int_t *q, mess_matrix out)

  Copy and permute a matrix.
- int mess_matrix_colperm (mess_matrix A, mess_int_t *perm)

  Permute the columns of a matrix inplace.
- int mess_matrix_colpermcopy (mess_matrix A, mess_int_t *perm, mess_matrix B)

  Permute the columns of a matrix while copying.
- int mess_matrix_joinpatterns (mess_matrix A, mess_matrix B)

  Join the pattern of two sparse matrices.
- int mess_matrix_equalpattern (mess_matrix A, mess_matrix B)

  Check if two matrices have the same pattern.
- int mess_vector_tomatrix (mess_vector v, mess_matrix mat)

  Convert a vector to a matrix.
- int mess_vector_frommatrix (mess_matrix mat, mess_vector v)

Convert a single column matrix to a vector.

- int mess_matrix_equal (mess_matrix mat1, mess_matrix mat2)

  Check if two matrices are equal on data structure level.
- int mess_matrix_equal_verbose (mess_matrix mat1, mess_matrix mat2)

  Check if two matrices are equal on data structure level (with verbose output).
- int mess_matrix_decomp (mess_matrix A, mess_matrix Asym, mess_matrix Askewsym)

  Decompose a matrix in a symmetric (hermitian) and skewsymmetric (skewhermitian) part.

8.8.1 Detailed Description

These are for example reorderings like mess_matrix_reorder_amd or mess_matrix_reorder_colamd or permutations.

A special function is mess_matrix_spy which produces a spy plot of a matrix as a bmp file.

See also

Various Other Operations with Vectors

8.8.2 Enumeration Type Documentation

8.8.2.1 enum mess_reorder_t

The mess_reorder_t enumeration is used to select different matrix reorderings. Mostly they are used to reduce the fill-in in direct sparse solvers or to increase the data-locality in matrix-vector products.

See also

mess_matrix_reorder
mess_matrix_reorder_amd
mess_matrix_reorder_colamd
mess_matrix_reorder_rcm

Enumerator

MESS_REORDER_NONE Select the natural (identity) reordering

MESS_REORDER_RCM Select the Reverse Cuthill McKee (RCM) reordering.

MESS_REORDER_AMD Select the AMD reordering from `SuiteSparse`.

MESS_REORDER_COLAMD Select the COLAMD reordering from `SuiteSparse`.

8.8.3 Function Documentation

8.8.3.1 int mess_matrix_colperm (mess_matrix *A*, mess_int_t * *perm*)

Parameters

in,out	*A*	matrix to permute
in	*perm*	input column permutation

Returns

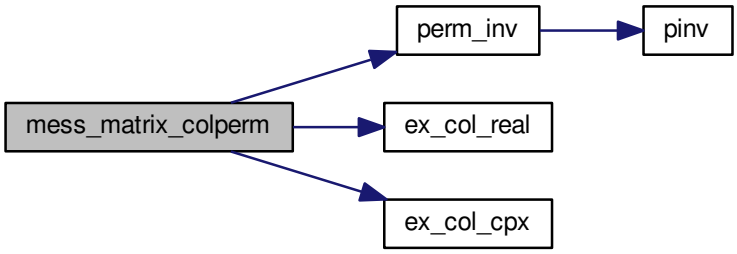
zero on success or a non zero error code otherwise

The mess_matrix_colperm function permutes the columns of a matrix A by the given permutation $perm$, where $perm[i] = j$ means the $i - th$ column of A will be moved to the j-th column in the permuted A.
The algorithm works inplace.
If OpenMP is available the permuation process will be done in parallel if the permutation allows it.

See also

mess_matrix_colpermcopy

Here is the call graph for this function:



8.8.3.2 int mess_matrix_colpermcopy (mess_matrix *A*, mess_int_t * *perm*, mess_matrix *B*)

Parameters

in	*A*	input matrix to permute
in	*perm*	input column permutation
out	*B*	permuted matrix

Returns

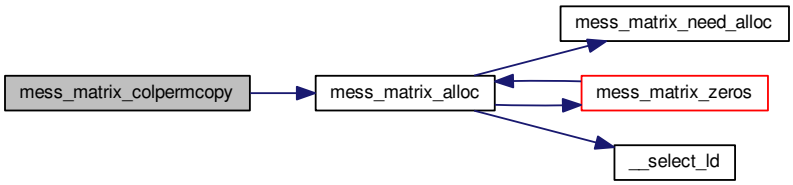
zero on success or a non zero error code otherwise

The mess_matrix_colpermcopy function permutes the columns of a matrix A by a given permutation $perm$ and stores the permuted matrix in B, where $perm[i] = j$ means the i-th column of A will be moved to the j-th column in the permuted matrix B.

See also

mess_matrix_colperm

Here is the call graph for this function:



8.8.3.3 int mess_matrix_decomp (mess_matrix *A*, mess_matrix *Asym*, mess_matrix *Askewsym*)

Parameters

in	*A*	input matrix
out	*Asym*	output symmetric (hermitian) part of A (NULL if not wanted)
out	*Askewsym*	output skewsymmetric (skewhermitian) oart of A (NULL if not wanted)

Returns

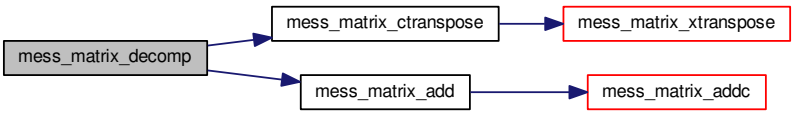
zero on success or a non-zero error value otherwise

The mess_matrix_decomp function decomposes a matrix A into

$$\begin{aligned} A_{sym} &\leftarrow \frac{1}{2}(A + A^H) \\ A_{skewsym} &\leftarrow \frac{1}{2}(A - A^H) \end{aligned}.$$

Note that Asym and Aksewsym have the same storage type as A.

Here is the call graph for this function:



8.8.3.4 int mess_matrix_equal (mess_matrix *mat1*, mess_matrix *mat2*)

Parameters

in	*mat1*	input first matrix
in	*mat2*	input second matrix

Returns

zero if the matrices are not equal or an error occured, a non zero value otherwise

The mess_matrix_equal function checks if two matrices are equal on data structure level.
If an error occurs this is treated as non equal matrices.
If you compare sparse matrices please ensure that they are sorted using mess_matrix_sort before.
If matrices have different storage type they are non equal, too. The leading dimension of dense matrices is ignored.

8.8.3.5 int mess_matrix_equal_verbose (mess_matrix *mat1*, mess_matrix *mat2*)

Parameters

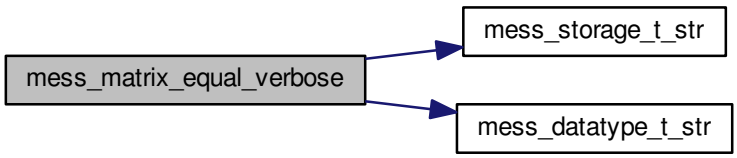
in	*mat1*	input first matrix
in	*mat2*	input second matrix

Returns

zero if the matrices are not equal or an error occured, a non zero value otherwise

The mess_matrix_equal_verbose function checks if two matrices are equal on data structure level.
If an error occurs this is treated as non equal matrices.
If you compare sparse matrices please ensure that they are sorted using mess_matrix_sort before. If matrices have different storage type they are non equal, too. The leading dimension of dense matrices is ignored.
In contrast to mess_matrix_equal this function produces some additional output to identify where the matrices differ.

Here is the call graph for this function:



8.8.3.6 int mess_matrix_equalpattern (mess_matrix *A*, mess_matrix *B*)

Parameters

in	*A*	input matrix A
in	*B*	input matrix B

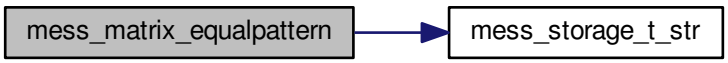
Returns

1 if the pattern are equal, 0 if the pattern differs and all values larger than one are error codes.

The mess_matrix_equalpattern function checks if the matrices A and B have the same pattern.
If they have the same pattern `1` will be returned.
If the pattern is not the same `0` will be returned.
All values bigger than `1` are error codes.

The input matrices A and B have to be sorted internally. That means the colptr of a MESS_CSR or the rowptr of a MESS_CSC mess_matrix must be ordered in the same way. If the matrices A or B are sparse they need to be sorted using mess_matrix_sort before. If both are sparse the need to have the same storage format.

Here is the call graph for this function:



8.8.3.7 int mess_matrix_joinpatterns (mess_matrix *A*, mess_matrix *B*)

Parameters

`in,out`	*A*	input/output matrix A
`in,out`	*B*	input/output matrix B

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_joinpatterns function joins the pattern of two sparse matrices A and B. At exit $\mathcal{P}(A) = \mathcal{P}(B)$. This is done by computing

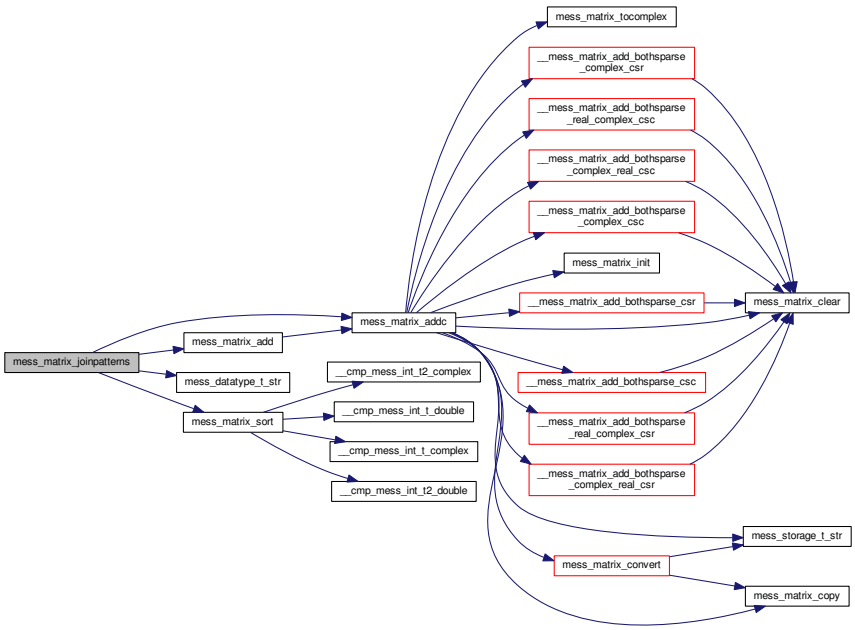
$$A \leftarrow A + 0 \cdot B$$

and

$$B \leftarrow B + 0 \cdot A.$$

using mess_matrix_add.

Here is the call graph for this function:



8.8.3.8 mess_int_t mess_matrix_memsize (mess_matrix *matrix*)

Parameters

in	*matrix*	input matrix

Returns

approximate size of the input matrix in bytes.

The mess_matrix_memsize function determines the size of a matrix in bytes.
If the matrix is a view the returned size is 0, because the effectively the matrix needs no additional memory.

See also

mess_matrix_memsize_nnz

8.8.3.9 mess_int_t mess_matrix_memsize_nnz (mess_int_t *rows*, mess_int_t *cols*, mess_int_t *nnz*, mess_int_t *type*, mess_int_t *dtype*)

Parameters

in	*rows*	input number of rows
in	*cols*	input number of cols
in	*nnz*	input number of nonzero elements
in	*type*	input storage type
in	*dtype*	input data type of the matrix

Returns

approximate size of the desired matrix

The mess_matrix_memsize_nnz function determines the size of a matrix in byte from given parameters.

See also

mess_matrix_memsize

8.8.3.10 int mess_matrix_perm (mess_matrix *matrix*, mess_int_t * *p*, mess_int_t * *q*)

Parameters

in,out	*matrix*	matrix to permute
in	*p*	input row permutation of length $matrix \to rows$
in	*q*	input column permutation of length $matrix \to cols$

Returns

zero on success or a non zero error code otherwise

The mess_matrix_perm function permutes a matrix in rows and column space.
If p or q is NULL, it is assumed that it would be the identity permutation.
The underlying algorithms do not work in-place at the moment.

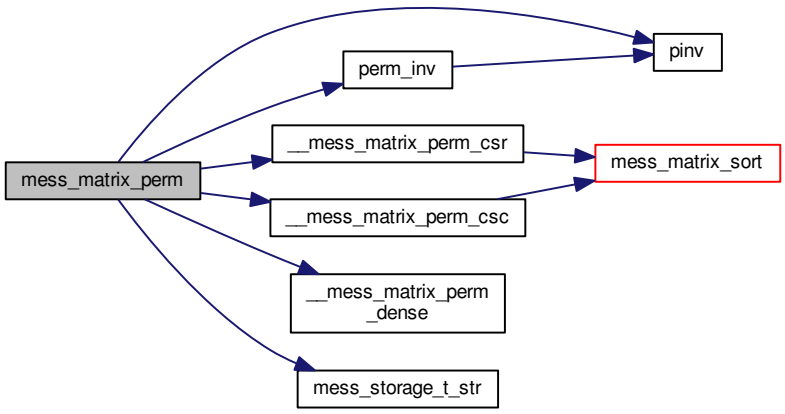
Attention

This function does not yet support the MESS_COORD storage type.

See also

mess_matrix_permcopy

Here is the call graph for this function:



8.8.3.11 int mess_matrix_permcopy (mess_matrix *matrix*, mess_int_t * *p*, mess_int_t * *q*, mess_matrix *out*)

Parameters

in	*matrix*	input matrix
in	*p*	input row permutation of length $matrix \to rows$
in	*q*	input column permutation of length $matrix \to cols$
out	*out*	permuted matrix

Returns

zero on success or a non zero error code otherwise

The mess_matrix_permcopy function copies and permutes a matrix.
It is similar to mess_matrix_perm.
If a permutation vector is set NULL, the identity is assumed.

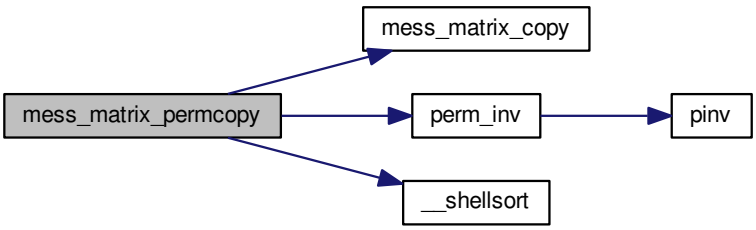
Attention

This subroutine currently only works for Compressed Sparse Rows storage.

See also

mess_matrix_perm

Here is the call graph for this function:



8.8.3.12 int mess_matrix_reorder (mess_reorder_t *reorder*, mess_matrix *A*, mess_int_t * *p*, mess_int_t * *q*)

Parameters

in	*reorder*	input mess_reorder_t type of the reordering
in	*A*	input matrix A to reorder
out	*p*	output row permutation vector of length rows of A
out	*q*	output column permutation vector of length columns of A

Returns

zero on success or a non-zero error value otherwise

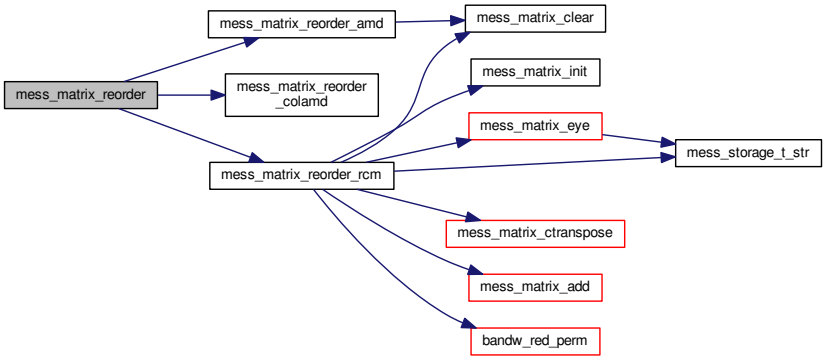
The mess_matrix_reorder function selects and computes a reordering given by the value of the reorder argument. The possible reorderings are listed in the mess_reorder_t enumeration. If the input matrix A is dense, the identity permutation is returned.

Both output vectors p and q need to be preallocated to the right size before.

See also

mess_matrix_reorder_amd
mess_matrix_reorder_colamd
mess_matrix_reorder_rcm

Here is the call graph for this function:



8.8.3.13 int mess_matrix_reorder_amd (mess_matrix *A*, mess_int_t * *p*)

Parameters

in	*A*	input matrix A
out	*p*	output permutation array of length rows of A

Returns

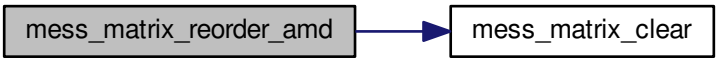
zero on success or a non-zero error value otherwise

The mess_matrix_reorder_amd function reorders a matrix with approximate minimum degree (AMD) ordering. It uses the AMD reordering subroutines from SuiteSparse.

See also

mess_matrix_reorder_rcm
mess_matrix_reorder_colamd
mess_matrix_reorder

Here is the call graph for this function:



8.8.3.14 int mess_matrix_reorder_colamd (mess_matrix *A*, mess_int_t * *p*)

Parameters

in	*A*	input matrix A
out	*p*	input/output permutation of length columns of A

Returns

zero on success or a non-zero error value otherwise

The mess_matrix_reorder_colamd function computes the column approximate minimum degree (COLAMD) reordering of a given matrix.
It uses the COLAMD package from SuiteSparse.

See also

mess_matrix_reorder_rcm
mess_matrix_reorder_amd
mess_matrix_reorder

8.8.3.15 int mess_matrix_reorder_rcm (mess_matrix *A*, mess_int_t * *perm*)

Parameters

in	*A*	input matrix A
out	*perm*	output permutation containing the Reverse Cuthill-McKee reordering

Returns

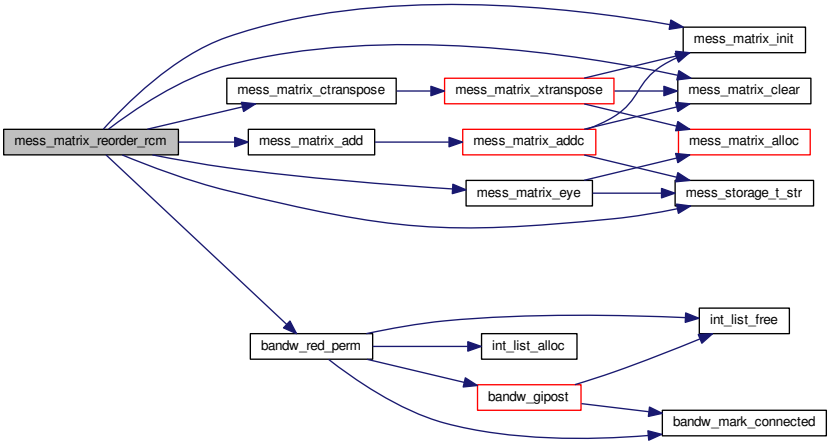
zero on success or a non-zero error value otherwise

The mess_matrix_reorder_rcm function calls the Gibbs-Poole-Stockmeyer modification of the Cuthill-McKee algorithm to produce a bandwith reducing permutaion $perm$ for a given adjacency list $adel$ of a sparse matrix.

See also

mess_matrix_reorder_amd
mess_matrix_reorder_colamd
mess_matrix_reorder

Here is the call graph for this function:



8.8.3.16 int mess_matrix_spy (mess_matrix *matrix*, const char * *filename*, int *width*, int *height*)

Parameters

in	*matrix*	input matrix to visualize
in	*filename*	input filename of output bitmap
in	*width*	input width of output bitmap
in	*height*	input height of output bitmap

Returns

zero on success or a non zero error code.

The function mess_matrix_spy produces a bitmap of the non zero structure of a matrix.
The width and height argument are changed if the dimension of the matrix do not fit good in the bitmap.

Here is the call graph for this function:

8.8.3.17 int mess_vector_frommatrix (mess_matrix *mat*, mess_vector *v*)

Parameters

in	*mat*	input single column input matrix
out	*v*	output vector with the elements from the matrix column

Returns

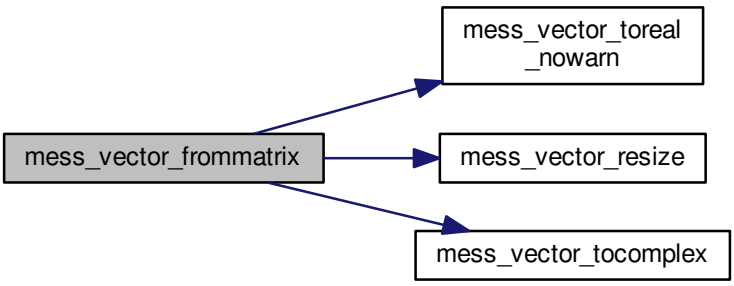
zero on success or a non zero error code otherwise

The mess_vector_frommatrix function converts a single column matrix into a vector.

See also

mess_vector_tomatrix

Here is the call graph for this function:



8.8.3.18 int mess_vector_tomatrix (mess_vector *v*, mess_matrix *mat*)

Parameters

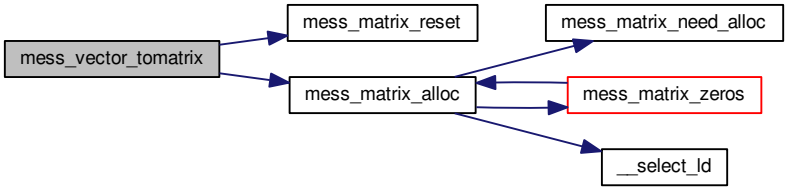
in	*v*	input vector
out	*mat*	output matrix

Returns

zero on success or a non-zero error value otherwise

The mess_vector_tomatrix converts a mess_vector structure to a mess_matrix structure with one column.

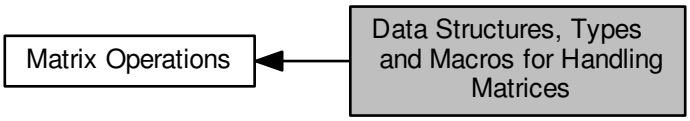
Here is the call graph for this function:



8.9 Data Structures, Types and Macros for Handling Matrices

This category contains all matrix related data structures, macros and type definitions.

Collaboration diagram for Data Structures, Types and Macros for Handling Matrices:



Data Structures

- struct mess_matrix_st

  The mess_matrix structure to handle dense and sparse matrices in a common context.
- struct mess_matrix_file_st

  Type definition and structure for mess_matrix files.

Macros

- #define MESS_IS_CSR(c) ((∗c).store_type == MESS_CSR)

  Macro to check if a matrix is stored in MESS_CSR storage.
- #define MESS_IS_CSC(c) ((∗c).store_type == MESS_CSC)

  Macro to check if a matrix is stored in MESS_CSC storage.
- #define MESS_IS_DENSE(c) ((∗c).store_type == MESS_DENSE)

  Macro to check if a matrix is stored in MESS_DENSE storage.
- #define MESS_IS_COORD(c) ((∗c).store_type == MESS_COORD)

  Macro to check if a matrix is stored in MESS_COORD storage.
- #define MESS_IS_SPARSE(c) (MESS_IS_CSR(c) || MESS_IS_CSC(c) || MESS_IS_COORD(c))

  Macro to check if a matrix is stored in some sparse storage.
- #define MESS_IS_GENERAL(m) ((∗m).symmetry == MESS_GENERAL)

  Check if a matrix is a general one, that means it has no special symmetry.
- #define MESS_IS_UNSYMMETRIC(m) ((∗m).symmetry == MESS_GENERAL)

  Check if a matrix is a general one.
- #define MESS_IS_SYMMETRIC(m) ((∗m).symmetry == MESS_SYMMETRIC)

  Check if a matrix is stored symmetric.
- #define MESS_IS_SKEWSYMMETRIC(m) ((∗m).symmetry == MESS_SKEWSYMMETRIC)

  Check if a matrix is stored skew-symmetric.
- #define MESS_IS_HERMITIAN(m) ((∗m).symmetry == MESS_HERMITIAN)

  Check if a matrix is stored symmetric.
- #define MESS_MATRIX_FILE_MAGIC {0x63,0x6D,0x65,0x73,0x73,0x4D}
- #define MESS_MATRIX_FILE_MAGIC_SET(m) {m[0]=0x63; m[1]=0x6D; m[2]=0x65; m[3]=0x73 ; m[4]=0x73 ; m[5]=0x4D; }
- #define MESS_MATRIX_FILE_MAGIC_CHECK(m) ((m[0] == 0x63)&&(m[1]==0x6D) &&(m[2]==0x65)&&(m[3]==0x73)&&(m[4 D))
- #define MESS_MATRIX_FILE_VERSION 0x01
- #define MESS_MATRIX_CHECKFORMAT(input, output, status, fmt)

  Check if a matrix is stored in the correct storage format and converts it if necessary.
- #define MESS_MATRIX_RESET(matrix) do { mess_matrix_reset((matrix)); } while(0)

  Resets a mess_matrix object.

Typedefs

- typedef struct mess_matrix_st mess_matrix_st

  The mess_matrix structure to handle dense and sparse matrices in a common context.
- typedef struct mess_matrix_st ∗ mess_matrix

  Type definition to make mess_matrix_st easier to use.
- typedef struct mess_matrix_file_st mess_matrix_file

  Type definition and structure for mess_matrix files.

Enumerations

Functions

- int mess_matrix_clear (mess_matrix ∗matrix)

  Remove a mess_matrix object from memory.
- int mess_matrix_init (mess_matrix ∗matrix)

  Initialize the mess_matrix structure.

8.9.1 Detailed Description

Additionally it contains all directly related macros. The definition of the data_type (mess_datatype_t) and the operation (mess_operation_t) for mess_matrix_mvp, mess_matrix_gaxpy and mess_matrix_multiply can be found in General Type Definitions and Macros.

8.9.2 Macro Definition Documentation

8.9.2.1 #define MESS_IS_COORD(c) ((∗c).store_type == MESS_COORD)

The MESS_IS_COORD checks if a matrix is stored in the MESS_COORD storage.

8.9.2.2 #define MESS_IS_CSC(c) ((∗c).store_type == MESS_CSC)

The MESS_IS_CSC checks if a matrix is stored in the MESS_CSC storage.

8.9.2.3 #define MESS_IS_CSR(c) ((∗c).store_type == MESS_CSR)

The MESS_IS_CSR checks if a matrix is stored in the MESS_CSR storage.

8.9.2.4 #define MESS_IS_DENSE(c) ((∗c).store_type == MESS_DENSE)

The MESS_IS_DENSE checks if a matrix is stored in the MESS_DENSE storage.

8.9.2.5 #define MESS_IS_GENERAL(m) ((∗m).symmetry == MESS_GENERAL)

The MESS_IS_GENERAL macro checks if the matrix has no special symmetric structure.

8.9.2.6 #define MESS_IS_HERMITIAN(m) ((∗m).symmetry == MESS_HERMITIAN)

The MESS_IS_HERMITIAN macro checks if a matrix is stored hermitian symmetric. That means the algorithms have to take care about implicit hermitian symmetry.
At the moment this is not used in **M.E.S.S.**.

8.9.2.7 #define MESS_IS_SKEWSYMMETRIC(m) ((∗m).symmetry == MESS_SKEWSYMMETRIC)

The MESS_IS_SKEWSYMMETRIC macro checks if a matrix is stored skew symmetric. That means the algorithms have to take care about implicit skew symmetry.
At the moment this is not used in **M.E.S.S.**.

8.9.2.8 #define MESS_IS_SPARSE(*c*) (MESS_IS_CSR(c) || MESS_IS_CSC(c) || MESS_IS_COORD(c))

The MESS_IS_SPARSE checks if a matrix is stored in some sparse format. It is the combination of MESS_IS_CSR, MESS_IS_CSC and MESS_IS_COORD.

8.9.2.9 #define MESS_IS_SYMMETRIC(m) ((∗m).symmetry == MESS_SYMMETRIC)

The MESS_IS_SYMMETRIC macro checks if a matrix is stored symmetric. That means the algorithms have to take care about implicit symmetry.
At the moment this is not used in **M.E.S.S.**.

8.9.2.10 #define MESS_IS_UNSYMMETRIC(m) ((∗m).symmetry == MESS_GENERAL)

The MESS_IS_UNSYMMETRIC macro is used as a wrapper for MESS_IS_GENERAL.

8.9.2.11 #define MESS_MATRIX_CHECKFORMAT(*input*, *output*, *status*, *fmt*)

Value:

```
{                                                                    \
    if ( (input)->store_type == (fmt)){
                                    \
        (output) = (input);
                                    \
        (status) = -1;
                                    \
    } else {
                                    \
        mess_matrix_init(&output);
                                                        \
        MSG_WARN("convert matrix \"%s\" from %s to %s\n", #input,
      mess_storage_t_str((input)->store_type), mess_storage_t_str(fmt));    \
        (status) = mess_matrix_convert((input),(output), (fmt));
                                                    \
        FUNCTION_FAILURE_HANDLE((status), ((status)>0),
      mess_matrix_convert);                                                  \
    }
                                    \
}
```

Parameters

in	*input*	Matrix object which storage formart should be checked
out	*output*	Matrix object with the desired storage format
out	*status*	Status of the conversion
in	*fmt*	input Desired storage format

The MESS_MATRIX_CHECKFORMAT macro checks if a given input matrix has the desired storage format fmt. If the matrix is already in the correct format the output object will be set to the input object and stat will be -1. If the input is not of the desired format output will be a newly created matrix object and contains a copy of the input matrix in the desired storage format. If the conversion was successful stat contains 0 otherwise and positive error code. If stat is 0 the matrix output needs to be freed using mess_matrix_clear.

For example if one want to guarantee that the matrix `A` is in MESS_CSR storage:

```
int status;
mess_matrix helper;
MESS_MATRIX_CHECKFORMAT(A, helper, status, MESS_CSR);
... Work with helper ...
if ( status == 0 ) {
mess_matrix_clear(&helper);
}
```

8.9.2.12 #define MESS_MATRIX_FILE_MAGIC {0x63,0x6D,0x65,0x73,0x73,0x4D}

Definition of a mess_matrix binary file.

8.9.2.13 #define MESS_MATRIX_FILE_MAGIC_CHECK(m) ((m[0] == 0x63)&&(m[1]==0x6D) &&(m[2]==0x65)&&(m[3]==0x73)&&(m[4]==0x73)&&(m[5]==0x4D))

Macro to check if a file is a valid mess_matrix binary file.

8.9.2.14 #define MESS_MATRIX_FILE_MAGIC_SET(m) {m[0]=0x63; m[1]=0x6D; m[2]=0x65; m[3]=0x73 ; m[4]=0x73 ; m[5]=0x4D; }

Macro to set a file to a valid mess_matrix binary file.

8.9.2.15 #define MESS_MATRIX_FILE_VERSION 0x01

Definition of the binary format version

8.9.2.16 #define MESS_MATRIX_RESET(*matrix*) do { mess_matrix_reset((matrix)); } while(0)

Parameters

in,out	*matrix*	Matrix object to reset

The MESS_MATRIX_RESET is a wrapper around mess_matrix_reset .

8.9.3 Typedef Documentation

8.9.3.1 typedef struct mess_matrix_st* mess_matrix

This type definition creates a pointer out of the mess_matrix_st structure.
This pointer is the data type which should be used for the matrices. Most functions take a mess_matrix object directly. Only mess_matrix_init and mess_matrix_clear need a pointer to a mess_matrix, that means a double pointer to a mess_matrix_st structure.

See also

mess_matrix_init
mess_matrix_clear

8.9.3.2 typedef struct mess_matrix_file_st mess_matrix_file

The mess_matrix_file type definition and structure defines a mess_matrix file.

8.9.3.3 typedef struct mess_matrix_st mess_matrix_st

This structure represents a matrix which is stored as compressed row, compressed column or dense storage. The used storage stored matrix is determined by the store_type property. In case of dense storage the matrix is stored in the Fortran format. Otherwise a 0 based indirect addressing format like MESS_CSR, MESS_CSC or MESS_C↩OORD is used.

The data_type flag indicates the data type of the matrix. It is set to one of the values of the mess_datatype_↩t enumeration. If the matrix is real the values array is used to store the data. In case of a complex matrix the values_cpx array is used. The unused one of those two arrays should be set to `NULL`.

See also

mess_matrix_convert

Attention

The symmetry flag is supposed to indicate whether the matrix is symmetric or not. Unfortunately we do not use this at the moment. This is the reason why the symmetry is always set to MESS_GENERAL even if you read a symmetric matrix from a file.

8.9.4 Enumeration Type Documentation

8.9.4.1 enum mess_storage_t

The mess_storage_t enumeration is used to represent the storage type of a mess_matrix object.
It is used for the store_type filed inside mess_matrix_st.

Enumerator

MESS_UNKNOWN Matrix storage type is not set yet.

MESS_CSR Matrix is stored in Compressed Sparse Row (MESS_CSR) storage.

MESS_CSC Matrix is stored in Compressed Sparse Column (MESS_CSC) storage.

MESS_DENSE Matrix is stored in dense Fortran storage (MESS_DENSE).

MESS_COORD Matrix is storage in sparse coordinate storage (MESS_COORD)

8.9.4.2 enum mess_symmetry_t

The mess_symmetry_t enumeration is used to represent the symmetry of a matrix. It is used for the symmetry flag inside the mess_matrix_st structure.
At the moment only MESS_GENERAL is used inside **`M.E.S.S.`**. The other ones exist for further extensions.

Enumerator

MESS_GENERAL The matrix is a general one without any symmetry.

MESS_SYMMETRIC The matrix is symmetric.

MESS_SKEWSYMMETRIC The matrix is skew-symmetric

MESS_HERMITIAN The matrix is hermitian symmetric.

8.9.5 Function Documentation

8.9.5.1 int mess_matrix_clear (mess_matrix * *matrix*)

Parameters

in,out	*matrix*	pointer to the mess_matrix structure

Returns

always zero (if **M.E.S.S.** is compiled using the DEBUG option then it returns an error if the matrix is already freed)

The mess_matrix_clear function removes the mess_matrix and cleans up.
If the matrix is a view on to another one only the helper data is freed. The data is freed when the parent matrix is freed.
In contrast to other functions the mess_matrix is passed as a pointer instead of a direct passing.

8.9.5.2 int mess_matrix_init (**mess_matrix** ∗ *matrix*)

Parameters

in,out	*matrix*	a pointer to a mess_matrix object

Returns

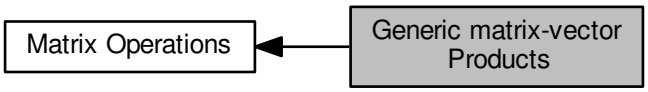
zero on success or a non zero error code otherwise

The mess_matrix_init function creates a new mess_matrix object.
It allocates the basic data structures and sets all values to 0.
The array for the matrix data is allocated. This have to be done separately by mess_matrix_alloc.
In contrast to other functions the mess_matrix is passed as a pointer instead of a direct passing.

8.10 Generic matrix-vector Products

Representation of a generic matrix-vector Product.

Collaboration diagram for Generic matrix-vector Products:



Data Structures

- struct mess_mvpcall_st

  Represent a generic matrix-vector-product.

Typedefs

- typedef struct mess_mvpcall_st ∗ mess_mvpcall

  Represent a generic matrix-vector-product.

Functions

- int mess_mvpcall_matrix (mess_mvpcall ∗mvpcall, mess_operation_t op, mess_matrix A)

  Generate a mess_mvpcall object for a simple matrix.
- int mess_mvpcall_operator (mess_mvpcall ∗mvpcall, mess_int_t dim, mess_datatype_t data_type, int(∗mvp)(void ∗data, mess_operation_t op, mess_vector x, mess_vector y), void ∗data)

  Generate a mess_mvpcall object for a complex operator.
- int mess_mvpcall_apply (mess_mvpcall mvpcall, mess_operation_t op, mess_vector x, mess_vector y)

  Apply a generic matrix-vector product.
- int mess_mvpcall_clear (mess_mvpcall ∗mvpcall)

  Clean a mess_mvpcall object .

8.10.1 Detailed Description

Additionally it contains a clean up of a mess_mvpcall object and a function applying a given operation to a given matrix.

8.10.2 Typedef Documentation

8.10.2.1 typedef struct mess_mvpcall_st * mess_mvpcall

The mess_mvpcall object represents a generic matrix-vector product $y = Ax$ where A is not longer restricted to be a simple matrix.
This is necessary to use for example iterative solvers or iterative eigenvalue computations on more complex objects than a simple matrix.

See also

Sparse Eigenvalue Solvers
Solvers

8.10.3 Function Documentation

8.10.3.1 int mess_mvpcall_apply (mess_mvpcall *mvpcall*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*)

Parameters

in	*mvpcall*	input mess_mvpcall object defining the matrix-vector product
in	*op*	input operation applied to the matrix
in	*x*	input right hand side vector
out	*y*	vector

Returns

zero on success or a non zero error code otherwise

The mess_mvpcall_apply function computes

$$y = op(A)x$$

where A is represented by a mess_mvpcall object $mvpcall$.

8.10.3.2 int mess_mvpcall_clear (mess_mvpcall * *mvpcall*)

Parameters

in	*mvpcall*	input mess_mvpcall object to clear

Returns

zero on success or a non zero error code otherwise

The mess_mvpcall_clear function clears a mess_mvpcall object and sets it to `NULL`.

8.10.3.3 int mess_mvpcall_matrix (mess_mvpcall * *mvpcall*, mess_operation_t *op*, mess_matrix *A*)

Parameters

out	*mvpcall*	generated mess_mvpcall object based on the given matrix
in	*op*	input operation applied to A
in	*A*	input matrix of the matrix-vector product

Returns

zero on success or a non zero error code otherwise

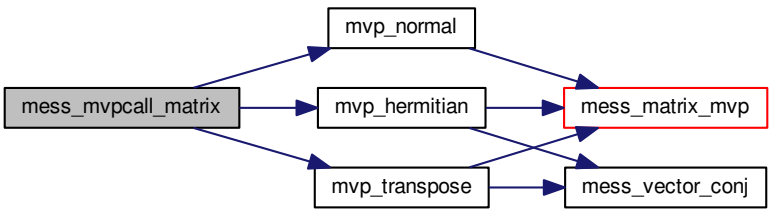
The mess_mvpcall_matrix function generates a mess_mvpcall object which represents

$$mvpcall = op(A)$$

where A is a mess_matrix and op can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$
MESS_OP_HERMITIAN	$op(A) = A^H$

Here is the call graph for this function:



8.10.3.4 int mess_mvpcall_operator (mess_mvpcall ∗ *mvpcall*, mess_int_t *dim*, mess_datatype_t *data_type*, int(∗)(void ∗data, mess_operation_t op, mess_vector x, mess_vector y) *mvp*, void ∗ *data*)

Parameters

out	*mvpcall*	generated mess_mvpcall object based on the given function handle
in	*dim*	input dimension of the represented operator
in	*data_type*	input data type of the represented operator
in	*mvp*	input function pointer to the matrix-vector product function
in	*data*	input pointer to additional data for the matrix-vector product

Returns

zero on success or a non zero error code otherwise

The mess_mvpcall_operator function generates a mess_mvpcall object which represents

$$y = Ax$$

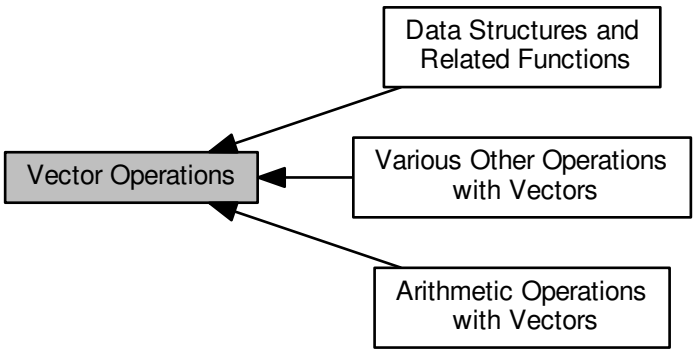
where A is a complex operator represented by a function pointer mvp.

Here is the call graph for this function:

8.11 Vector Operations

This section contains all vector related operations.

Collaboration diagram for Vector Operations:



Modules

- Data Structures and Related Functions

  This category contains all vector related data structures, type definitions and functions.
- Arithmetic Operations with Vectors

  This category contains all basic arithmetic operations with vectors.
- Various Other Operations with Vectors

  This category contains all vector related functions that fit not well in one of the other categories.

8.11.1 Detailed Description

Vectors are representes as instances of mess_vector and stored in Fortran storage format. You have to initialize a mess_vector instance with mess_vector_init and destroy it afterwards with mess_vector_clear.

```
mess_vector v;
mess_int_t rows = 10;
mess_vector_init(&v,rows, MESS_REAL);
//perform operations with v
mess_vector_clear(&v);
```

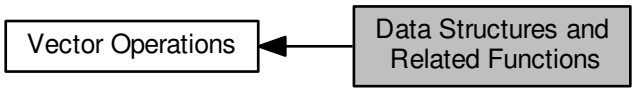
You can find more informations in the following sections:

- Data Structures and Related Functions
- Arithmetic Operations with Vectors
- Various Other Operations with Vectors

8.12 Data Structures and Related Functions

This category contains all vector related data structures, type definitions and functions.

Collaboration diagram for Data Structures and Related Functions:



Data Structures

- struct mess_vector_st

  Representation of a real or complex vector.

Typedefs

- typedef struct mess_vector_st mess_vector_st

  Representation of a real or complex vector.
- typedef mess_vector_st ∗ mess_vector

  Type definition of a real or complex vector.

Functions

- int mess_vector_init (mess_vector ∗vect)

  Initialize the mess_vector structure.
- int mess_vector_alloc (mess_vector vect, mess_int_t dim, mess_datatype_t dtype)

  Allocate a MESS_REAL or MESS_COMPLEX the mess_vector structure.
- int mess_vector_reset (mess_vector vect)

  Resets a mess_vector object.
- int mess_vector_clear (mess_vector ∗vect)

  Clean up the mess_vector structure to use it again.
- int mess_vector_copy (mess_vector x, mess_vector y)

  Copy mess_vector structure x to a mess_vector y structure.
- int mess_vector_copy_tocomplex (mess_vector in, mess_vector out)

  Copy a mess_vector strucutre into a complex mess_vector structure.
- int mess_vector_tocomplex (mess_vector v)

  Convert a vector to complex one.
- int mess_vector_toreal (mess_vector v)

  Convert a vector to real one.
- int mess_vector_toreal_nowarn (mess_vector v)

  Convert a vector to real values (without any warnings).
- int mess_vector_totype (mess_vector v, mess_datatype_t dt)

Convert a vector to the given datatype.

- int mess_vector_from_farray (mess_vector v, mess_int_t dim, double ∗vals, mess_double_cpx_t ∗vals_cpx)

  Create a mess_vector from a `Fortran` array.
- int mess_vector_from_lapack (mess_vector v, mess_int_t dim, double ∗vals_re, double ∗vals_im)

  Create a mess_vector from two `Fortran` array.
- int mess_vector_realpart (mess_vector in, mess_vector out)

  Get the real part of a vector.
- int mess_vector_imagpart (mess_vector in, mess_vector out)

  Get the imaginary part of a vector.
- int mess_vector_complex_from_parts (mess_vector xr, mess_vector xc, mess_vector x)

  Generate a complex vector from real and imaginary part.

8.12.1 Detailed Description

It includes functions to

- convert and copy a vector in various versions
- generate a complex vector from real and imaginary part
- get the real or imaginary part of a complex vector
- initialize and clean up a vector

and data structures and type definitions to implement a vector.

8.12.2 Typedef Documentation

8.12.2.1 typedef mess_vector_st∗ mess_vector

The mess_vector type definition defines a mess_vector in order to use it easier.

8.12.2.2 typedef struct mess_vector_st mess_vector_st

The mess_vector_st structure holds all information about a vector.

8.12.3 Function Documentation

8.12.3.1 int mess_vector_alloc (mess_vector *vect*, mess_int_t *dim*, mess_datatype_t *dtype*)

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

`in,out`	*vect*	input/output allocated mess_vector instance
`in`	*dim*	input dimension of the new vector
`in`	*dtype*	input data type of the new vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_alloc function allocates a vector `vect` with length `dim` and mess_datatype_t `dtype`.
If no memory is left it returns MESS_ERROR_MEMORY.

8.12.3.2 int mess_vector_clear (mess_vector * *vect*)

Parameters

`in,out`	*vect*	input/outout pointer to the mess_vector structure

Returns

always zero

The mess_vector_clear function removes a vector from memory and cleans up the mess_vector structure.

8.12.3.3 int mess_vector_complex_from_parts (mess_vector *xr*, mess_vector *xc*, mess_vector *x*)

Parameters

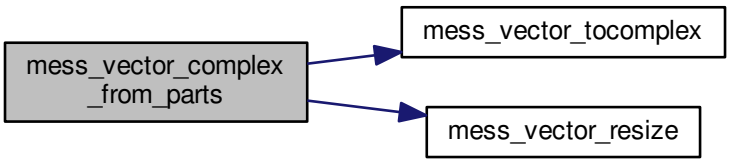
`in`	*xr*	input real part
`in`	*xc*	input imaginary part
`out`	*x*	output vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_complex_from_parts function creates a complex vector from real and imaginary part.

Here is the call graph for this function:



8.12.3.4 int mess_vector_copy (mess_vector *x*, mess_vector *y*)

Parameters

in	*x*	input vector
in,out	*y*	input/output vector

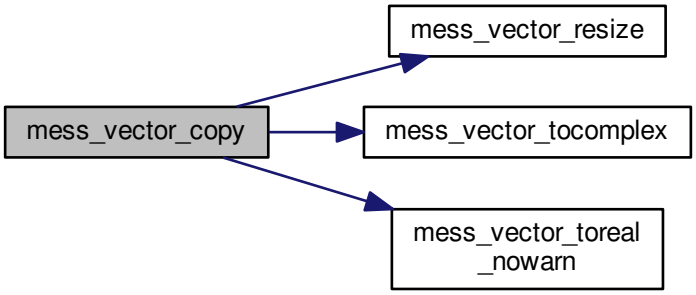
Returns

zero on success or a non-zero error value

The mess_vector_copy function copies a mess_vector structure x to a mess_vector structure y. If the size of y is not correct it will be resized. The data type of y is changed to the data type of x. In case of errors the return values will be:

- MESS_ERROR_NULLPOINTER - x or y points to NULL
- MESS_ERROR_DATATYPE - x is not real or complex

Here is the call graph for this function:



8.12.3.5 int mess_vector_copy_tocomplex (mess_vector *in*, mess_vector *out*)

Parameters

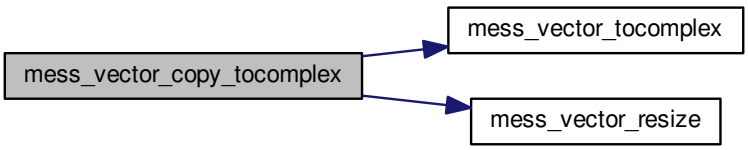
in	*in*	input vector
in,out	*out*	input/output vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_copy_tocomplex function copies a mess_vector structure into a complex mess_vector structure, e.g. it does not change the data type if you copy a real vector into a complex one like mess_vector_copy it do.

Here is the call graph for this function:



8.12.3.6 int mess_vector_from_farray (mess_vector *v*, mess_int_t *dim*, double * *vals*, mess_double_cpx_t * *vals_cpx*)

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

out	*v*	output vector to copy the data to
in	*dim*	dimension of the vector
in	*vals*	input array with real values
in	*vals_cpx*	input array with complex values

Returns

zero on success or a non zero error code otherwise

The mess_vector_from_farray function copies a `vals` or `val_cpx` array to a mess_vector. Depending whether `vals` or `vals_cpx` is given a real mess_vector or a complex mess_vector is created.

See also

mess_matrix_dense_from_carray
mess_matrix_dense_from_farray

8.12.3.7 int mess_vector_from_lapack (mess_vector *v*, mess_int_t *dim*, double * *vals_re*, double * *vals_im*)

Parameters

out	*v*	output vector to copy the data to
in	*dim*	dimension of the vector
in	*vals_re*	input array with real part values, `NULL` if not wanted
in	*vals_im*	input array with imaginary part values, `NULL` if not wanted

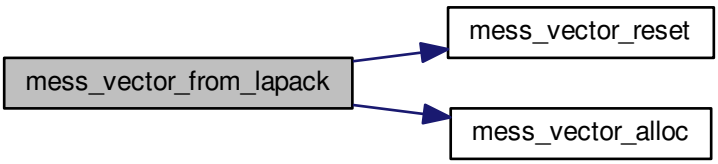
Returns

zero on success or a non zero error code otherwise

The mess_vector_from_lapack function copies a `vals_re` and `val_im` array to a mess_vector. If `vals_im` points to `NULL` or all entries are zero `v` will be a real vector. The function is mostly used when results from `LAPACK` are splited in real an imaginary part and should be copied to a mess_vector.

If `vals_re` and `vals_im` point to `NULL` an to `NULL` an error is returned.

Here is the call graph for this function:



8.12.3.8 int mess_vector_imagpart (mess_vector *in*, mess_vector *out*)

Parameters

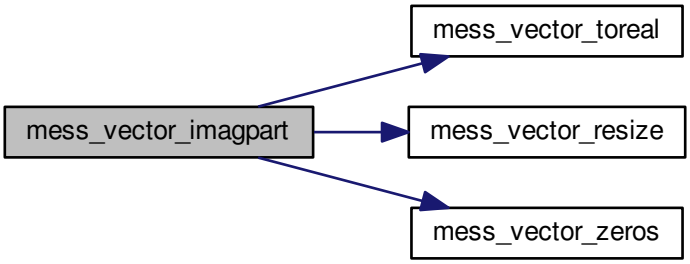
`in`	*in*	input vector
`in,out`	*out*	output vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_imagpart function gets the imaginary part of a vector.

Here is the call graph for this function:



8.12.3.9 int mess_vector_init (mess_vector * *v*)

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

in,out	*v*	a pointer to a mess_vector object

Returns

zero on success or a non zero error code otherwise

The mess_vector_init function creates a new mess_vector object.
It allocates the basic data structures and sets all values to 0.

8.12.3.10 int mess_vector_realpart (mess_vector *in*, mess_vector *out*)

Parameters

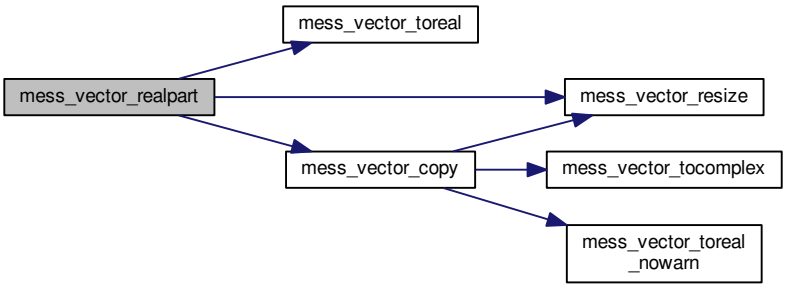
in	*in*	input vector
in,out	*out*	output vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_realpart function gets the real part of a vector.

Here is the call graph for this function:



8.12.3.11 int mess_vector_reset (mess_vector *v*)

Parameters

in,out	*v*	input/output vector object to reset

Returns

zeros on success or a non zero error value.

The mess_vector_reset function resets a mess_vector object that it behaves like a newly initialized one. In contrast to mess_vector_clear it does only free the internal data and not the surrounding structure.

8.12.3.12 int mess_vector_tocomplex (mess_vector *v*)

Parameters

in,out	*v*	input/output vector to convert

Returns

zero on success or a non-zero error value otherwise

The mess_vector_tocomplex function converts an arbitrary vector to a complex vector.

See also

mess_vector_toreal
mess_vector_toreal_nowarn
mess_vector_totype

8.12.3.13 int mess_vector_toreal (mess_vector *v*)

Parameters

in,out	*v*	input/output vector to convert

Returns

zero on success or a non-zero error value otherwise

The mess_vector_toreal function converts an arbitrary vector to a real vector. Imaginary parts are ignored.

See also

mess_vector_tocomplex
mess_vector_toreal_nowarn
mess_vector_totype

8.12.3.14 int mess_vector_toreal_nowarn (mess_vector *v*)

Parameters

in,out	*v*	input/output vector to convert

Returns

zero on success or a non-zero error value otherwise

The mess_vector_toreal_nowarn function converts an arbitrary vector to a real vector. Imaginary parts are ignored and no warnings are displayed.

See also

mess_vector_tocomplex
mess_vector_toreal
mess_vector_totype

8.12.3.15 int mess_vector_totype (mess_vector *v*, mess_datatype_t *dt*)

Parameters

in,out	*v*	input/output given vector
in	*dt*	input desired datatype

Returns

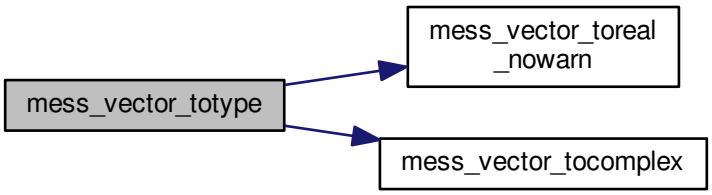
zero on success or a non-zero error value otherwise

The mess_vector_totype function converts a vector to a given datatype.

See also

mess_vector_tocomplex
mess_vector_toreal
mess_vector_toreal_nowarn

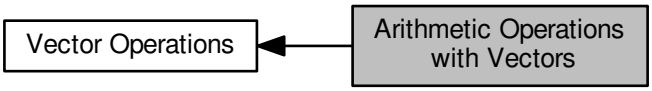
Here is the call graph for this function:



8.13 Arithmetic Operations with Vectors

This category contains all basic arithmetic operations with vectors.

Collaboration diagram for Arithmetic Operations with Vectors:



Macros

- #define mess_vector_scalee(val, vec) if (MESS_IS_REAL(vec)) { mess_vector_scale((val),(vec)); } else { mess_vector_scalec((val), (vec));}

  Macro to scale a vector.

Functions

- int mess_vector_axpy (double a, mess_vector x, mess_vector y)

  Compute $y \leftarrow ax + y$ (real).
- int mess_vector_norm2 (mess_vector x, double ∗nrm)

  Compute the 2-norm of a vector.
- int mess_vector_norm1 (mess_vector x, double ∗nrm)

  Compute the 1-norm of a vector.
- int mess_vector_norminf (mess_vector x, double ∗nrm)

  Compute the ∞-norm of a vector.
- int mess_vector_norm (mess_vector x, mess_norm_t nrm_t, double ∗nrm)

  Compute the norm of a vector.
- int mess_vector_scale (double alpha, mess_vector x)

  Scale a vector with a real scalar.
- int mess_vector_dot (mess_vector x, mess_vector y, double ∗dot)

  Compute the dot product of two vectors.
- int mess_vector_minvalue (mess_vector v, double ∗min)

  Get the minimal value in a vector.
- int mess_vector_maxvalue (mess_vector v, double ∗max)

  Get the maximum value in a vector.
- int mess_vector_scalec (mess_double_cpx_t alpha, mess_vector x)

  Scale a vector with a complex scalar.
- int mess_vector_dotc (mess_vector x, mess_vector y, mess_double_cpx_t ∗dot)

  Compute the dot product of two vectors (complex).
- int mess_vector_axpyc (mess_double_cpx_t a, mess_vector x, mess_vector y)

  Compute $y \leftarrow ax + y$ (complex).
- int mess_vector_dotu (mess_vector x, mess_vector y, mess_double_cpx_t ∗dot)

  Compute the dot product of two vectors (complex).

- int mess_vector_diffnorm (mess_vector x1, mess_vector x2, double *diff)

  Compute $\|x_1 - x_2\|_2$.
- int mess_vector_diffnorminf (mess_vector x1, mess_vector x2, double *diff)

  Compute $\|x_1 - x_2\|_\infty$.
- int mess_vector_max (mess_vector v, double *maxval, mess_int_t *maxind)

  Return the largest entry of a vector.
- int mess_vector_conj (mess_vector vector)

  Complex conjugatation of a vector.

8.13.1 Detailed Description

It includes

- an axpy update of a vector
- real and complex vector products
- scaling
- the computation of different norms
- the computation of the minimum and maximum value of a vector.

Additionally there is a macro to scale a real or complex vector.

8.13.2 Macro Definition Documentation

8.13.2.1 #define mess_vector_scalee(*val, vec*) if (MESS_IS_REAL(vec)) { mess_vector_scale((val),(vec)); } else { mess_vector_scalec((val), (vec));}

The mess_vector_scalee macro scales a real or complex vector with a value.

8.13.3 Function Documentation

8.13.3.1 int mess_vector_axpy (double *a*, mess_vector *x*, mess_vector *y*)

Parameters

in	*a*	input scaling scalar a
in	*x*	input vector x
in,out	*y*	input/output vector y

Returns

zero on success or a non-zero error value otherwise

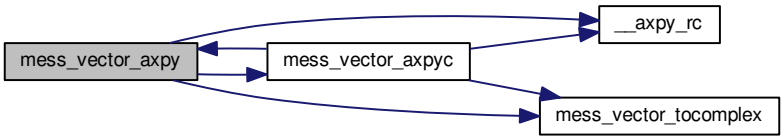
The mess_vector_axpy function adds a times a vector x to y and store it in y

$$y \leftarrow a \cdot x + y.$$

See also

mess_vector_axpyc

Here is the call graph for this function:



8.13.3.2 int mess_vector_axpyc (mess_double_cpx_t *a*, mess_vector *x*, mess_vector *y*)

Parameters

in	*a*	input scalar (double or mess_double_cpx_t) a
in	*x*	input vector
in,out	*y*	input/output vector

Returns

zero on success or a non-zero error value otherwise

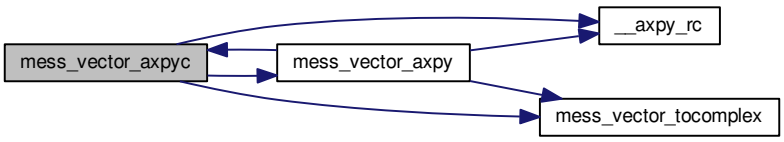
The mess_vector_axpyc function adds a times a vector x to y and store it in y:

$$y \leftarrow a \cdot x + y.$$

See also

mess_vector_axpy

Here is the call graph for this function:



8.13.3.3 int mess_vector_conj (mess_vector *vector*)

Parameters

in,out	*vector*	input/output vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_conj function performs a complex conjugation on a vector.

8.13.3.4 int mess_vector_diffnorm (mess_vector *x1*, mess_vector *x2*, double * *diff*)

Parameters

in	*x1*	input vector x_1
in	*x2*	input vector x_2
out	*diff*	output difference

Returns

zero on success or a non-zero error value otherwise

The mess_vector_diffnorm function computes

$$diff = \|x_1 - x_2\|_2.$$

8.13.3.5 int mess_vector_diffnorminf (mess_vector *x1*, mess_vector *x2*, double * *diff*)

Parameters

in	*x1*	input vector x_1
in	*x2*	input vector two
out	*diff*	output difference

Returns

zero on success or a non-zero error value otherwise

The mess_vector_diffnorminf function computes

$$diff = \|x_1 - x_2\|_\infty.$$

8.13.3.6 int mess_vector_dot (mess_vector *x*, mess_vector *y*, double * *dot*)

Parameters

in	*x*	input vector x
in	*y*	input vector y
out	*dot*	output dot product

Returns

zero on success or a non-zero error value otherwise

The mess_vector_dot function computes the dot product of a real vector x and a vector y, i.e.

$$dot = x^T y.$$

8.13.3.7 int mess_vector_dotc (mess_vector *x*, mess_vector *y*, mess_double_cpx_t * *dot*)

Parameters

in	*x*	input vector x
in	*y*	input vector y
out	*dot*	output dot product

Returns

zero on success or a non-zero error value otherwise

The mess_vector_dotc function computes the dot product of two vectors, i.e.

$$dot = x^H y.$$

8.13.3.8 int mess_vector_dotu (mess_vector *x*, mess_vector *y*, mess_double_cpx_t * *dot*)

Parameters

in	*x*	input vector x
in	*y*	input vector y
out	*dot*	output dot product

Returns

zero on success or a non-zero error value otherwise

The mess_vector_dotu function computes the dot product of two vectors, i.e.

$$dot = x^T y.$$

8.13.3.9 int mess_vector_max (mess_vector *v*, double * *maxval*, mess_int_t * *maxind*)

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

in	*v*	input vector
out	*maxval*	output the largest absolute value
out	*maxind*	output index of the maxval element

Returns

zero on success or a non-zero error value otherwise

The mess_vector_max function returns the largest absolute value and its position in the vector.

8.13.3.10 int mess_vector_maxvalue (mess_vector *v*, double * *max*)

Parameters

in	*v*	input vector
out	*max*	output maximum value

Returns

zero on success or a non-zero error value otherwise

The mess_vector_max function gets the maximum value of a vector.
In case of a complex vector we only use the real part.

8.13.3.11 int mess_vector_minvalue (mess_vector *v*, double * *min*)

Parameters

in	*v*	input vector
out	*min*	output minimum value

Returns

zero on success or a non-zero error value otherwise

The mess_vector_minvalue function gets the minimum value of a vector.
In case of a complex vector we only use the real part.

8.13.3.12 int mess_vector_norm (mess_vector *x*, mess_norm_t *nrm_t*, double * *nrm*)

Parameters

in	*x*	input vector x
in	*nrm↩ _t*	input mess_norm_t the desired type of norm
out	*nrm*	output $\|x\|$

Returns

zero on success or a non zero error code otherwise

The mess_vector_norm function calculates the norm of a mess_vector Supported norms are:

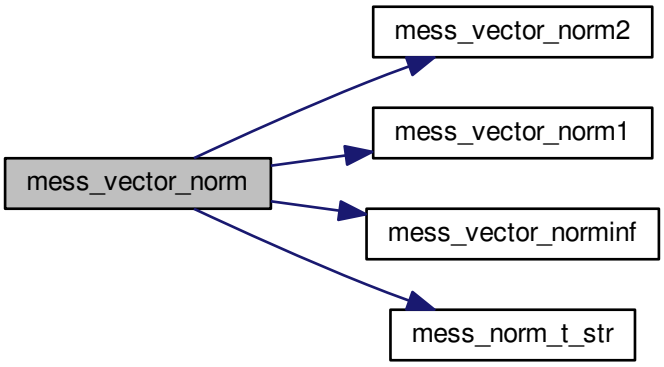
MESS_2_NORM, MESS_FROBENIUS_NORM, MESS_1_NORM, MESS_INF_NORM.

Here MESS_FROBENIUS_NORM and MESS_2_NORM is treated as the same.

See also

mess_vector_norm2
mess_vector_norminf
mess_vector_norm1

Here is the call graph for this function:



8.13.3.13 int mess_vector_norm1 (mess_vector *x*, double * *nrm*)

Parameters

in	*x*	input vector
out	*nrm*	output computed 1-norm of x

Returns

zero on success or a non-zero error value otherwise

The mess_vector_norm1 function computes the 1-norm of a vector x

$$nrm = \|x\|_1.$$

8.13.3.14 int mess_vector_norm2 (mess_vector *x*, double * *nrm*)

Parameters

in	*x*	input vector
out	*nrm*	output computed 2-norm of x

Returns

zero on success or a non-zero error value otherwise

The mess_vector_norm2 function computes the 2-norm of a vector x

$$nrm = \|x\|_2.$$

See also

mess_vector_norm1
mess_vector_norminf

8.13.3.15 int mess_vector_norminf (mess_vector *x*, double * *nrm*)

Parameters

in	*x*	input vector
out	*nrm*	output computed ∞-norm

Returns

zero on success or a non-zero error value otherwise

The mess_vector_norminf function computes the ∞-norm of a vector x

$$nrm = \|x\|_\infty.$$

8.13.3.16 int mess_vector_scale (double *a*, mess_vector *x*)

Parameters

in	*a*	input scalar a
in,out	*x*	input/output vector

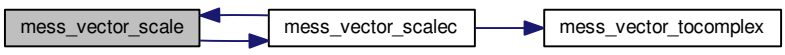
Returns

zero on success or a non-zero error value otherwise

The mess_vector_scale function scales a vector x with a scalar a such that

$$x \leftarrow a \cdot x.$$

Here is the call graph for this function:



8.13.3.17 int mess_vector_scalec (mess_double_cpx_t *a*, mess_vector *x*)

Parameters

in	*a*	input scalar a
in	*x*	input/output vector

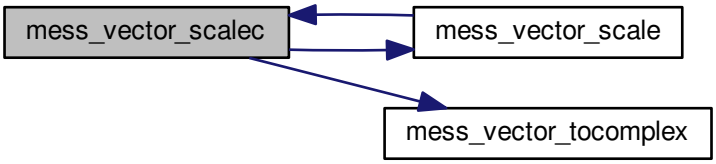
Returns

zero on success or a non-zero error value otherwise

The mess_vector_scalec function scales a vector x with a complex scalar a such that

$$x \leftarrow a \cdot x.$$

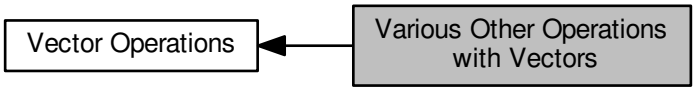
Here is the call graph for this function:



8.14 Various Other Operations with Vectors

This category contains all vector related functions that fit not well in one of the other categories.

Collaboration diagram for Various Other Operations with Vectors:



Functions

- int mess_vector_print (mess_vector vect)

  Print out a mess_vector structure.
- int mess_vector_printshort (mess_vector vect)

  Print out a mess_vector structure in a short form.
- int mess_vector_write (const char ∗filename, mess_vector vect)

  Write a vector to a matrix market vector file.
- int mess_vector_read (const char ∗filename, mess_vector vect)

  Read a vector from a matrix market vector file.
- int mess_vector_printinfo (mess_vector v)

  Print dimension and datatype of a mess_vector structure on stdout.
- int mess_vector_resize (mess_vector x, mess_int_t dim)

  Resize a vector.
- int mess_vector_zeros (mess_vector v)

  Fill a vector with zeros.
- int mess_vector_ones (mess_vector v)

  Fill a vector with ones.
- int mess_vector_rand (mess_vector v)

  Fill a vector with random numbers.
- int mess_vector_get (mess_vector v, mess_int_t i, mess_double_cpx_t ∗val)

  Get entry `i` of a mess_vector instance.
- int mess_vector_rand_init (mess_int_t ∗seed)

  Init random number generator with given seed.
- int mess_vector_iperm (mess_vector in, mess_int_t ∗iperm, mess_vector out)

  Inverse permutation of a vector.
- int mess_vector_perm (mess_vector in, mess_int_t ∗perm, mess_vector out)

  Permute of a vector.
- int mess_vector_perm_inplace (mess_vector v, mess_int_t ∗p)

  Permute a vector inplace.
- int mess_vector_iperm_inplace (mess_vector v, mess_int_t ∗q)

  Permute a vector inplace.
- int mess_vector_iperm_split (mess_vector in, mess_int_t ∗iperm, mess_vector reout, mess_vector imout)

  Inverse permutation of a vector and spilts real and imaginary part.

- int mess_vector_iperm_combine (mess_vector rein, mess_vector imin, mess_int_t *iperm, mess_vector out)

  Inverse permutation of a vector and spilts real and imaginary part.
- int mess_vector_perm_split (mess_vector in, mess_int_t *perm, mess_vector reout, mess_vector imout)

  Permutate a vector and split in real and imaginary part.
- int mess_vector_perm_combine (mess_vector rein, mess_vector imin, mess_int_t *perm, mess_vector out)

  Inverse permutation of a vector and combination of real and imaginary part.
- double * mess_vector_valuereal (mess_vector vec, mess_int_t index)

  Get a real value from a vector.
- mess_double_cpx_t * mess_vector_valuecomplex (mess_vector vec, mess_int_t index)

  Get a complex value from a vector.
- int mess_vector_sort (mess_vector vect)

  Sort the components of a vector in ascending order.
- int mess_vector_sort_realpart (mess_vector vect)

  Sort the components of a vector by real part in ascending order.
- int mess_vector_sort_imagpart (mess_vector vect)

  Sort the components of a vector by imaginary part in ascending order.
- int mess_vector_logspace10 (mess_vector vect, double a, double b, mess_int_t nsample)

  Create a $\log_{10}$ space vector.
- int mess_vector_logspacee (mess_vector vect, double a, double b, mess_int_t nsample)

  Create a $\log_{\mathrm{e}}$ space.
- int mess_vector_logspace2 (mess_vector vect, double a, double b, mess_int_t nsample)

  Create a $\log_2$ space.
- int mess_vector_linspace (mess_vector vect, double a, double b, mess_int_t nsample)

  Create a linspace vector.
- int mess_vector_cat (mess_vector x1, mess_vector x2, mess_vector x)

  Concatenate two vectors.
- int mess_vector_split (mess_vector input, mess_int_t n, mess_vector x1, mess_vector x2)

  Split a vector after the n-th entry.
- int mess_vector_lift (mess_vector in, mess_int_t n, mess_vector out)

  Add n zeros to vector.
- int mess_vector_kron (mess_vector in1, mess_vector in2, mess_vector out)

  Compute Kronecker product for two vectors.
- int mess_vector_filter_stable (mess_vector in)

  Filter values with positive realpart from vector.
- int mess_vector_filter (mess_vector in, int(*filter_real)(const double *), int(*filter_complex)(const mess_double_cpx_t *))

  Filter values from vector.
- int mess_vector_convert_if_real (mess_vector v)

  Convert complex vector to real vector if imaginary part of every entry is small.
- int mess_vector_map (mess_vector v, double(*f_real)(double), mess_double_cpx_t(*f_cpx)(mess_double_cpx_t))

  Map a scalar function to a vector.
- int mess_vector_map_abs (mess_vector)
- int mess_vector_map_acos (mess_vector)
- int mess_vector_map_acosh (mess_vector)
- int mess_vector_map_arg (mess_vector)
- int mess_vector_map_asin (mess_vector)
- int mess_vector_map_asinh (mess_vector)
- int mess_vector_map_atan (mess_vector)
- int mess_vector_map_atanh (mess_vector)
- int mess_vector_map_ceil (mess_vector)
- int mess_vector_map_conj (mess_vector)

- int mess_vector_map_cos (mess_vector)
- int mess_vector_map_cosh (mess_vector)
- int mess_vector_map_exp (mess_vector)
- int mess_vector_map_expm1 (mess_vector)
- int mess_vector_map_floor (mess_vector)
- int mess_vector_map_log (mess_vector)
- int mess_vector_map_not (mess_vector)
- int mess_vector_map_round (mess_vector)
- int mess_vector_map_sin (mess_vector)
- int mess_vector_map_sinh (mess_vector)
- int mess_vector_map_sqrt (mess_vector)
- int mess_vector_map_tan (mess_vector)
- int mess_vector_map_tanh (mess_vector)
- int mess_vector_map_isfinite (mess_vector)
- int mess_vector_map_isinf (mess_vector)
- int mess_vector_map_isnan (mess_vector)
- int mess_vector_any (mess_vector mat, mess_int_t(∗f_real)(double), mess_int_t(∗f_cpx)(mess_double_cpx_t), mess_int_t ∗anyval)

  Checks a mess_vector if a given predicate is true.
- mess_int_t mess_vector_memsize (mess_vector v)

  Return the size of a mess_vector in bytes.

8.14.1 Detailed Description

It includes functions to

- concatenate two vectors
- create a linspace or a logspace to different bases
- generate a vector filled with zeros, ones or random numbers
- get a value at a certain position
- permute a vector in various versions
- read, write and print a vector
- split a vector at a certain position.

8.14.2 Function Documentation

8.14.2.1 int mess_vector_any (mess_vector *v*, mess_int_t(∗)(double) *f_real*, mess_int_t(∗)(mess_double_cpx_t) *f_cpx*, mess_int_t ∗ *anyval*)

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

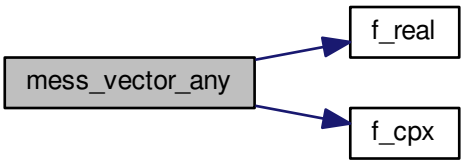
in,out	*v*	input/output mess_vector v
in	*f_real*	input scalar real function to apply to each entry of v with mess_int_t return value
in	*f_cpx*	input scalar complex function to apply to each entry of v with mess_int_t return value
in,out	*anyval*	input/output pointer to a mess_int_t

Returns

zero on success or a non-zero error value otherwise

The mess_vector_any function tests entries of `v` with `f_real` or `f_cpx`.
If `v` is MESS_REAL `f_real` is used.
If `v` is MESS_COMPLEX `f_cpx` is used.
`f_real` and `f_cpx` are scalar functions with real/complex argument and mess_int_t return value.
If `f_real` / `f_cpx` applied to any entry v_i returns a nonzero value `1` `anyval` is set to `1` and the mess_vector any terminates. If `f_real` / `f_cpx` returns `0` for every entry v_i mess_vector_any set `anyval` to `0` and terminates.
The mess_vector_any function works in a similiar fashion like the `any` function of `MATLAB`.

Here is the call graph for this function:



8.14.2.2 int mess_vector_cat (mess_vector *x1*, mess_vector *x2*, mess_vector *x*)

Parameters

`in`	*x1*	input vector x_1
`in`	*x2*	input vector x_2
`out`	*x*	output vector

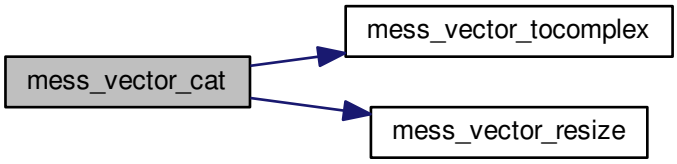
Returns

zero on success or a non-zero error value otherwise

The mess_vector_cat function concatenates two vectors, i.e.

$$x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}.$$

Here is the call graph for this function:



8.14.2.3 int mess_vector_convert_if_real (mess_vector *v*)

Parameters

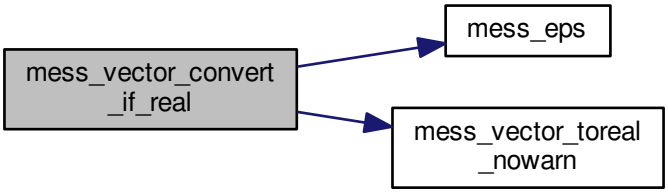
in,out	*v*	input/output vector v

Returns

zero on success or a non-zero error value otherwise

The mess_vector_convert_if_real converts a complex vector to a real one if the imaginary part of all entries are small. More precise if $\frac{|v_i - \mathrm{Re}(v_i)|}{|\mathrm{Re}(v_i)|} \leq eps \; \forall i$ then we perform $v \leftarrow \mathrm{Re}(v)$.

Here is the call graph for this function:



8.14.2.4 int mess_vector_filter (mess_vector *in*, int(∗)(const double ∗) *filter_real*, int(∗)(const mess_double_cpx_t ∗) *filter_complex*)

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

in,out	*in*	input/output vector
in	*filter_real*	function
in	*filter_complex*	function

Returns

zero on success or a non zero error value otherwise

The mess_vector_filter returns a vector constructed from values fullfilling condition given by the filter function pointer `filter_real` and `filter_complex`.

Here is the call graph for this function:

8.14.2.5 int mess_vector_filter_stable (mess_vector *in*)

Parameters

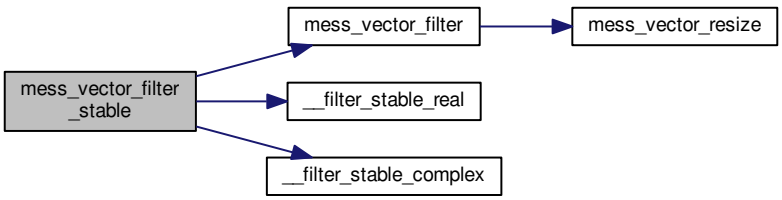
`in,out`	*in*	input vector

Returns

zero on success or a non zero error value otherwise

The mess_vector_filter_stable deletes all values with nonnegative realpart from vector and return the resized one.

Here is the call graph for this function:



8.14.2.6 int mess_vector_get (mess_vector *v*, mess_int_t *i*, mess_double_cpx_t * *val*)

Parameters

`in`	*v*	input mess_vector instance
`in`	*i*	input index of entry
`out`	*val*	output the `i` -th entry of `v`.

Returns

zero on success or a non-zero error value otherwise

The mess_vector_get function gets the `i` -th entry of `v`. If the index `i` is out of range or `v` points to `NULL` an error is returned. We assume zero based indexing.

8.14.2.7 int mess_vector_iperm (mess_vector *in*, mess_int_t * *iperm*, mess_vector *out*)

Parameters

in	*in*	input vector
in	*iperm*	input permutation
in,out	*out*	input/output vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_iperm function fullfills an inverse permutation, i.e.

$$out[perm[i]] = in[i].$$

See also

mess_vector_perm
mess_vector_perm_inplace
mess_vector_perm_combine
mess_vector_perm_split
mess_vector_iperm_inplace
mess_vector_iperm_combine
mess_vector_iperm_split

Here is the call graph for this function:

8.14.2.8 int mess_vector_iperm_combine (mess_vector *rein*, mess_vector *imin*, mess_int_t * *iperm*, mess_vector *out*)

Parameters

in	*rein*	input real part of vector
in	*imin*	input imaginary part of vector
in	*iperm*	input permutation
out	*out*	output vector

Returns

zero on success or a non-zero error value otherwise

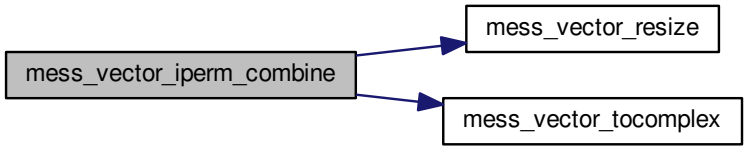
The mess_vector_iperm_combine function fullfills an inverse permutation to real and imaginary part and combines these to a vector, i.e.

$$out[perm[i]] = \mathrm{Re}(in[i]) + \mathrm{Im}(in[i]) * I.$$

See also

mess_vector_perm
mess_vector_perm_inplace
mess_vector_perm_combine
mess_vector_perm_split
mess_vector_iperm
mess_vector_iperm_inplace
mess_vector_iperm_split

Here is the call graph for this function:



8.14.2.9 int mess_vector_iperm_inplace (mess_vector *v*, mess_int_t * *perm*)

Parameters

in,out	*v*	input/output vector
in	*perm*	input permutation

Returns

zero on success or a non-zero error value otherwise

The mess_vector_iperm_inplace function permutes a vector nearly in place using a modified version of `dlapmt` from `LAPACK`.

$$in[perm[i]] \leftarrow in[i].$$

See also

mess_vector_perm
mess_vector_perm_inplace
mess_vector_perm_combine
mess_vector_perm_split
mess_vector_iperm
mess_vector_iperm_combine
mess_vector_iperm_split

8.14.2.10 int mess_vector_iperm_split (mess_vector *in*, mess_int_t * *iperm*, mess_vector *reout*, mess_vector *imout*)

Parameters

in	*in*	input vector in
in	*iperm*	input permutation
out	*reout*	output real part of vector in
out	*imout*	output imaginary part of vector in

Returns

zero on success or a non-zero error value otherwise

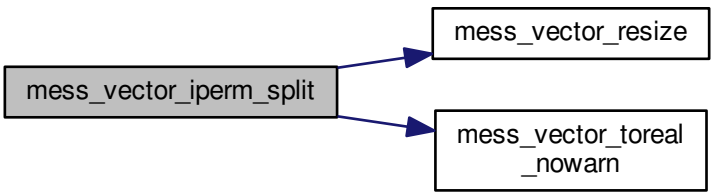
The mess_vector_iperm_split functions fullfills an inverse permutation to a vector and splits it in real and imaginary part, i.e.

$$\mathrm{Re}(out[perm[i]]) + \mathrm{Im}(out[perm[i]]) * I = in[i].$$

See also

mess_vector_perm
mess_vector_perm_inplace
mess_vector_perm_combine
mess_vector_perm_split
mess_vector_iperm
mess_vector_iperm_inplace
mess_vector_iperm_combine

Here is the call graph for this function:



8.14.2.11 int mess_vector_kron (mess_vector *in1*, mess_vector *in2*, mess_vector *out*)

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

in	*in1*	input vector in_1
in	*in2*	input vector in_2
out	*out*	output vector

Returns

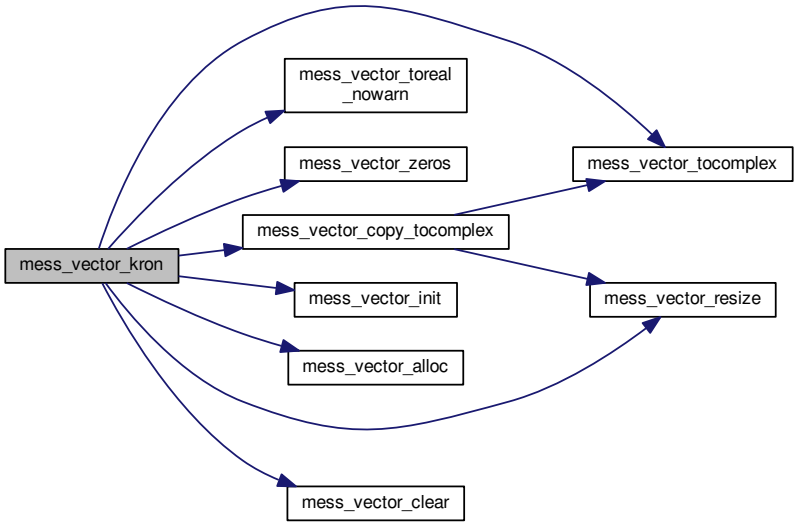
zero on success or a non-zero error value otherwise

The mess_vector_kron preforms a Kronecker product for in_1 and in_2, i.e.

$$\begin{aligned} out &= \begin{bmatrix} in_{1_1} \\ \vdots \\ in_{1_n} \end{bmatrix} \otimes \begin{bmatrix} in_{2_1} & \dots & in_{2_m} \end{bmatrix} \\ &= \begin{bmatrix} in_{1_1} in_2 \\ \vdots \\ in_{1_n} in_2 \end{bmatrix} \end{aligned}$$

The result is a vector out.

Here is the call graph for this function:



8.14.2.12 int mess_vector_lift (mess_vector *in*, mess_int_t *n*, mess_vector *out*)

Parameters

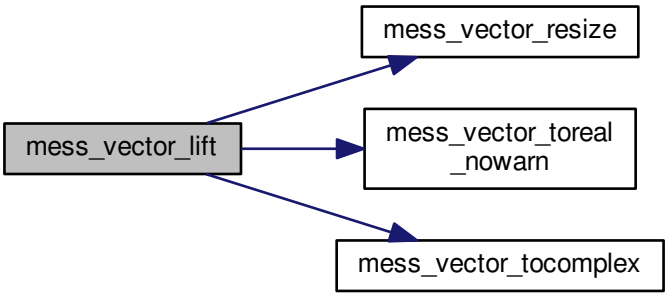
in	*in*	input vector
in	*n*	input positive number of zeros to add
out	*out*	output vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_lift function adds n zeros at the end of the vector in.
The result is in vector out.

Here is the call graph for this function:



8.14.2.13 int mess_vector_linspace (mess_vector *vect*, double *a*, double *b*, mess_int_t *nsample*)

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

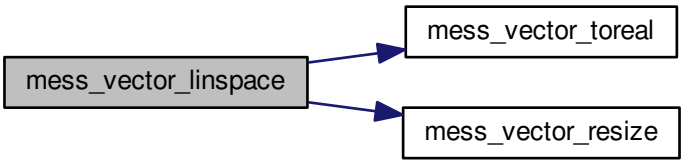
out	*vect*	output vector
in	*a*	input lower bound
in	*b*	input upper bound
in	*nsample*	input number of samples

Returns

zero on success or a non-zero error value otherwise

The mess_vector_linspace function creates a linearly spaced vector.

Here is the call graph for this function:



8.14.2.14 int mess_vector_logspace10 (mess_vector *vect*, double *a*, double *b*, mess_int_t *nsample*)

Parameters

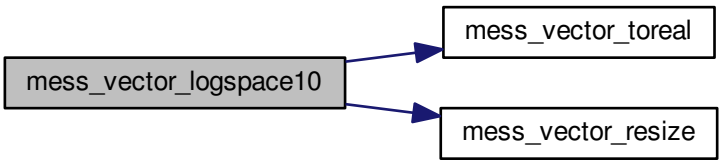
out	*vect*	output vector
in	*a*	input lower bound for the exponential
in	*b*	input upper bound for the exponential
in	*nsample*	input number of samples

Returns

zero on success or a non-zero error value otherwise

The mess_vector_logspace10 function creates a logarithmically spaced vector to base 10.

Here is the call graph for this function:



8.14.2.15 int mess_vector_logspace2 (mess_vector *vect*, double *a*, double *b*, mess_int_t *nsample*)

Parameters

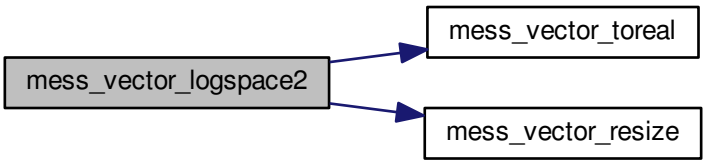
out	*vect*	output vector
in	*a*	input lower bound for the exponential
in	*b*	input upper bound for the exponential
in	*nsample*	input number of samples

Returns

zero on success or a non-zero error value otherwise

The mess_vector_logspacee function creates a logarithmically spaced vector to base 2.

Here is the call graph for this function:



8.14.2.16 int mess_vector_logspacee (mess_vector *vect*, double *a*, double *b*, mess_int_t *nsample*)

Parameters

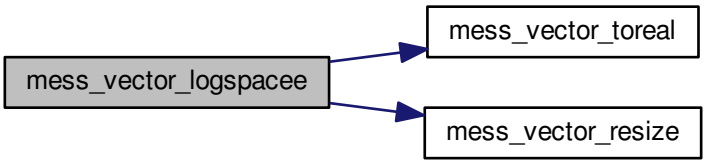
out	*vect*	output vector
in	*a*	input lower bound for the exponential
in	*b*	input upper bound for the exponential
in	*nsample*	input number of samples

Returns

zero on success or a non-zero error value otherwise

The mess_vector_logspacee function creates a logarithmically spaced vector to base e.

Here is the call graph for this function:



8.14.2.17 int mess_vector_map (mess_vector *v*, double(∗)(double) *f_real*, mess_double_cpx_t(∗)(mess_double_↩ cpx_t) *f_cpx*)

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

in,out	*v*	input/output vector v
in	*f_real*	input scalar real function to apply to each entry of v
in	*f_cpx*	input scalar complex function to apply to each entry of v

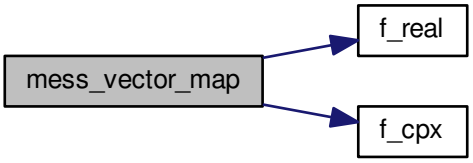
Returns

zero on success or a non-zero error value otherwise

The mess_vector_map function is a higher order function, which applies a scalar real `f_real` or a scalar complex `f_cpx` function to a each entry of a vector v. Depending on the data type of v we perform the following operation:

data type	operation
MESS_REAL	$v(i) \leftarrow f_{real}(v(i))$
MESS_COMPLEX	$v(i) \leftarrow f_{cpx}(v(i))$

Here is the call graph for this function:



8.14.2.18 int mess_vector_map_abs (mess_vector)

8.14.2.19 int mess_vector_map_acos (mess_vector)

8.14.2.20 int mess_vector_map_acosh (mess_vector)

8.14.2.21 int mess_vector_map_arg (mess_vector)

8.14.2.22 int mess_vector_map_asin (mess_vector)

8.14.2.23 int mess_vector_map_asinh (mess_vector)

8.14.2.24 int mess_vector_map_atan (mess_vector)

8.14.2.25 int mess_vector_map_atanh (mess_vector)

8.14.2.26 int mess_vector_map_ceil (mess_vector)

8.14.2.27 int mess_vector_map_conj (mess_vector)

8.14.2.28 int mess_vector_map_cos (mess_vector)

8.14.2.29 int mess_vector_map_cosh (mess_vector)

8.14.2.30 int mess_vector_map_exp (mess_vector)

8.14.2.31 int mess_vector_map_expm1 (mess_vector)

8.14.2.32 int mess_vector_map_floor (mess_vector)

8.14.2.33 int mess_vector_map_isfinite (mess_vector)

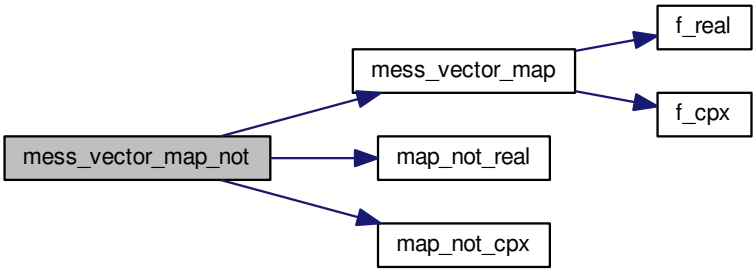
8.14.2.34 int mess_vector_map_isinf (mess_vector)

8.14.2.35 int mess_vector_map_isnan (mess_vector)

8.14.2.36 int mess_vector_map_log (mess_vector)

8.14.2.37 int mess_vector_map_not (mess_vector)

Here is the call graph for this function:



8.14.2.38 int mess_vector_map_round (mess_vector)

8.14.2.39 int mess_vector_map_sin (mess_vector)

8.14.2.40 int mess_vector_map_sinh (mess_vector)

8.14.2.41 int mess_vector_map_sqrt (mess_vector)

8.14.2.42 int mess_vector_map_tan (mess_vector)

8.14.2.43 int mess_vector_map_tanh (mess_vector)

8.14.2.44 mess_int_t mess_vector_memsize (mess_vector *v*)

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

in	*v*	input vector

Returns

approximate size of the `v` in bytes.

The mess_vector_memsize function determines the size of `v` in bytes.

See also

mess_matrix_memsize

8.14.2.45 int mess_vector_ones (mess_vector *v*)

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

in,out	*v*	input/output vector v

Returns

zero on success or a non-zero error value otherwise

The mess_vector_ones function fills a vector v with one entries.

8.14.2.46 int mess_vector_perm (mess_vector *in*, mess_int_t * *perm*, mess_vector *out*)

Parameters

in	*in*	input vector
in	*perm*	input permutation
in,out	*out*	input/output vector

The mess_vector_perm function permutes a vector, such that

$$out[i] = in[perm[i]].$$

See also

mess_vector_perm_inplace
mess_vector_perm_combine
mess_vector_perm_split
mess_vector_iperm
mess_vector_iperm_inplace
mess_vector_iperm_combine
mess_vector_iperm_split

Here is the call graph for this function:

8.14.2.47 int mess_vector_perm_combine (mess_vector *rein*, mess_vector *imin*, mess_int_t * *perm*, mess_vector *out*)

Parameters

in	*rein*	input real part of vector
in	*imin*	input imaginary part of vector
in	*perm*	input permutation
out	*out*	output vector

Returns

zero on success or a non-zero error value otherwise

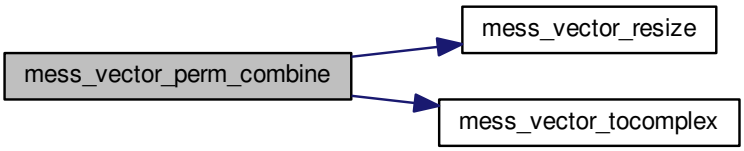
The mess_vector_perm_combine function fullfills an inverse permutation of the real part rein and the imaginary part imin of a vector and combines these, i.e.

$$out[i] = \mathrm{Re}(rein[perm[i]]) + \mathrm{Im}(imin[perm[i]]) * I.$$

See also

mess_vector_perm
mess_vector_perm_inplace
mess_vector_perm_split
mess_vector_iperm
mess_vector_iperm_inplace
mess_vector_iperm_combine
mess_vector_iperm_split

Here is the call graph for this function:



8.14.2.48 int mess_vector_perm_inplace (mess_vector *v*, mess_int_t * *perm*)

Parameters

in,out	*v*	input/output vector
in	*perm*	input permutation

Returns

zero on success or a non-zero error value otherwise

The mess_vector_perm_inplace function permutes a vector nearly in place using a modified version `dlapmt` from LAPACK.

$$in[i] \leftarrow in[perm[i]].$$

See also

mess_vector_perm
mess_vector_perm_combine
mess_vector_perm_split
mess_vector_iperm
mess_vector_iperm_inplace
mess_vector_iperm_combine
mess_vector_iperm_split

8.14.2.49 int mess_vector_perm_split (mess_vector *in*, mess_int_t * *perm*, mess_vector *reout*, mess_vector *imout*)

Parameters

in	*in*	input vector in
in	*perm*	input permutation
out	*reout*	output real part of permuted vector in
out	*imout*	output imaginary part of pemuted vector in

Returns

zero on success or a non-zero error value otherwise

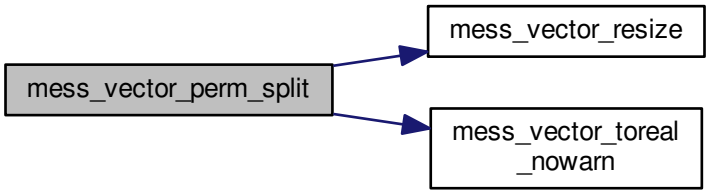
The mess_vector_perm_split function permutes a vector and splits this vector in real and imaginary part, i.e.

$$\mathrm{Re}(out[i]) + \mathrm{Im}(out[i])) * I = in[perm[i]].$$

See also

mess_vector_perm
mess_vector_perm_inplace
mess_vector_perm_combine
mess_vector_iperm
mess_vector_iperm_inplace
mess_vector_iperm_combine
mess_vector_iperm_split

Here is the call graph for this function:



8.14.2.50 int mess_vector_print (mess_vector *vect*)

Parameters

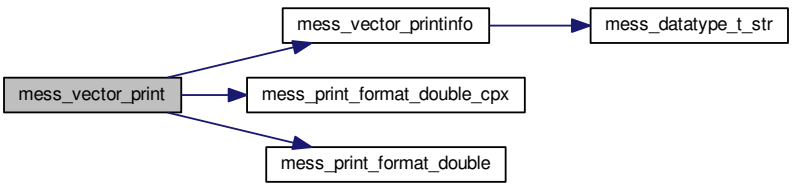
in	*vect*	input vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_print function prints all components of a mess_vector structure on stdout.
If the given vector points to `NULL` a MESS_ERROR_NULLPOINTER is returned.
If the data type is unknown MESS_ERROR_DATATYPE is returned. We use zero based indexing.

Here is the call graph for this function:



8.14.2.51 int mess_vector_printinfo (mess_vector *v*)

Parameters

in	*v*	input vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_printinfo function prints dimension and datatype of a mess_vector structure on stdout.

Here is the call graph for this function:

8.14.2.52 int mess_vector_printshort (mess_vector *vect*)

Parameters

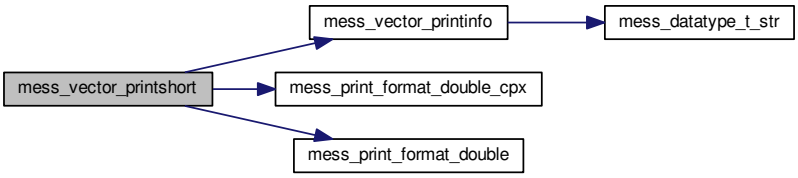
in	*vect*	input vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_printshort function is the same like mess_vector_print except of that only the first 20 components are printed. We use zero based indexing.

Here is the call graph for this function:



8.14.2.53 int mess_vector_rand (mess_vector *v*)

Parameters

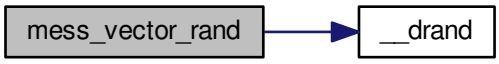
in,out	*v*	input/ouput vector v

Returns

zero on success or a non-zero error value otherwise

The mess_vector_rand function fills a vector v with random entries.

Here is the call graph for this function:



8.14.2.54 int mess_vector_rand_init (mess_int_t * *seed*)

Parameters

in	*seed*	pointer to mess_int_t for given seed, if `NULL` current calender time is used.

Returns

zero in every case

The mess_vector_rand_init initializes the `rand` function from `libc` for generating random numbers.
If `seed` points to `NULL` the current calender time is used for initialization of `rand` function. This function is usefull in combination with: This is a wrapper around mess_matrix_rand_init.

See also

mess_vector_rand

Here is the call graph for this function:

8.14.2.55 int mess_vector_read (const char * *filename*, mess_vector *vect*)

Parameters

in	*filename*	input filename of matrix market vector file
in,out	*vect*	vector storing read data

Returns

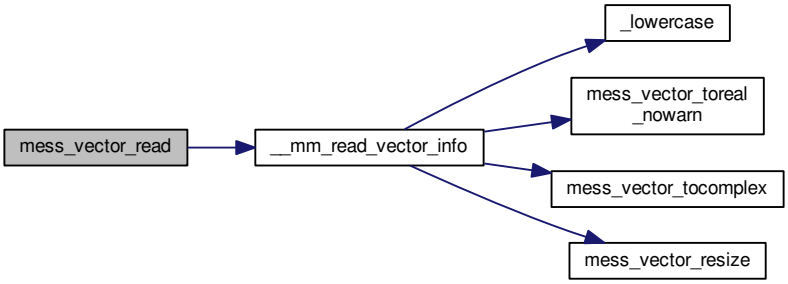
zero on success or a non zero error code

The mess_vector_read function reads a vector from a file.
The file need to be in `MatrixMarket` vector format, that means the header need to be

"%%MatrikMarket vector..."

and the dimension is only one value.

Here is the call graph for this function:



8.14.2.56 int mess_vector_resize (mess_vector *v*, mess_int_t *dim*)

Parameters

in,out	*v*	input/outout vector to resize
in	*dim*	input new dimension of the vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_resize function resizes a given vector v to dimension dim.
If the new dimension is smaller the $v \rightarrow dim$ elements after dim are lost.
The following errors can be returned:

Error	Description
MESS_ERROR_DIMENSION	the new dimension is less or equal to zero

Error	Description
MESS_ERROR_NULLPOINTER	the input points to NULL
MESS_ERROR_MEMORY	reallocating memory caused an error
MESS_ERROR_NOCOMPLEX	no support for complex numbers available
MESS_ERROR_DATATYPE	the data type in v is wrong

8.14.2.57 int mess_vector_sort (mess_vector *vect*)

Parameters

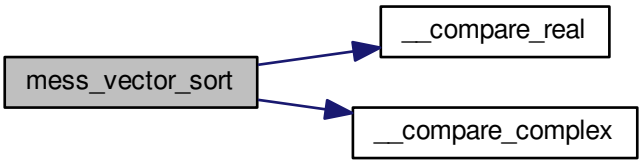
in,out	*vect*	input/output vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_sort function sorts the values of a vector in ascending order.

Here is the call graph for this function:



8.14.2.58 int mess_vector_sort_imagpart (mess_vector *vect*)

Parameters

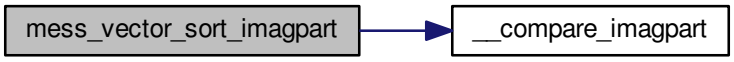
in,out	*vect*	input/output vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_sort_imagpart function sorts the values of a vector by imaginary part in ascending order.

Here is the call graph for this function:



8.14.2.59 int mess_vector_sort_realpart (mess_vector *vect*)

Parameters

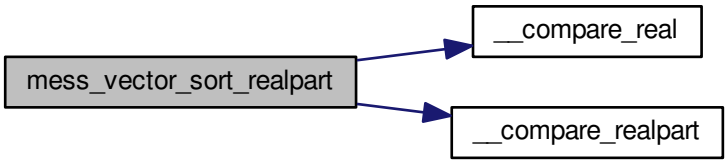
in,out	*vect*	input/output vector

Returns

zero on success or a non-zero error value otherwise

The mess_vector_sort_realpart function sorts the values of a vector by real part in ascending order.

Here is the call graph for this function:



8.14.2.60 int mess_vector_split (mess_vector *input*, mess_int_t *n*, mess_vector *x1*, mess_vector *x2*)

Parameters

in	*input*	input vector
in	*n*	input size of the first output n
out	*x1*	first output x_1
out	*x2*	second output x_2

Returns

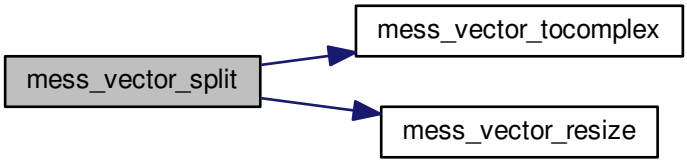
zero on success or a non-zero error value otherwise

The mess_vector_split function splits a vector into two vectors. The split is done after the n-th element. In this way $dim(x_1) = n$ and

$$\begin{aligned} x_1 &= input(0 : n-1), \\ x_2 &= input(n : end). \end{aligned}$$

If x_1 points to NULL, x_1 is neglected and $x_2 = input(n : end)$.
If x_2 points to NULL, x_2 is neglected and $x_1 = input(0 : n-1)$.

Here is the call graph for this function:



8.14.2.61 mess_double_cpx_t∗ mess_vector_valuecomplex (mess_vector *vec*, mess_int_t *index*)

Parameters

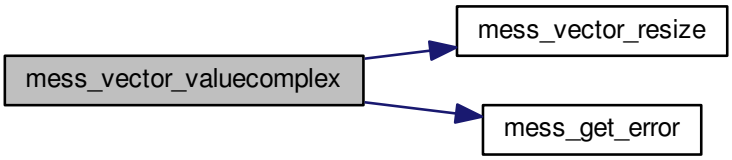
in	*vec*	input vector
in	*index*	input index of the value

Returns

pointer value at the index or a non-zero error value

The mess_vector_valuecomplex function gets a pointer to value of the given index. If the vector is not long enough it is resized.

Here is the call graph for this function:



8.14.2.62 double∗ mess_vector_valuereal (mess_vector *vec*, mess_int_t *index*)

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

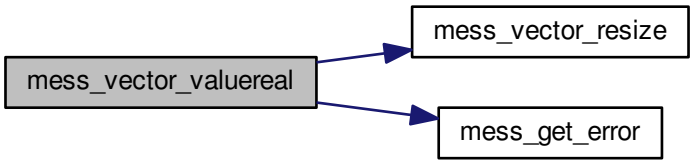
in	*vec*	input vector
in	*index*	input index of the value

Returns

pointer value at the index or a non-zero error value

The mess_vector_valuereal function gets a pointer to value of the given index. If the vector is not long enough it is resized.

Here is the call graph for this function:



8.14.2.63 int mess_vector_write (const char * *filename*, mess_vector *vect*)

Parameters

in	*filename*	input filename where to store the matrix
in,out	*vect*	mess_vector to write

Returns

zero on success or a non zero error code

The mess_vector_write function writes a vector to a file.
The output is similar to a matrix market file but contains only a one dimensional array.
As consequence the header is

"%%MatrixMarket vector ..."

and the stored dimension is only one value.

Here is the call graph for this function:

8.14.2.64 int mess_vector_zeros (mess_vector *v*)

Parameters

in,out	*v*	input/output vector v

Returns

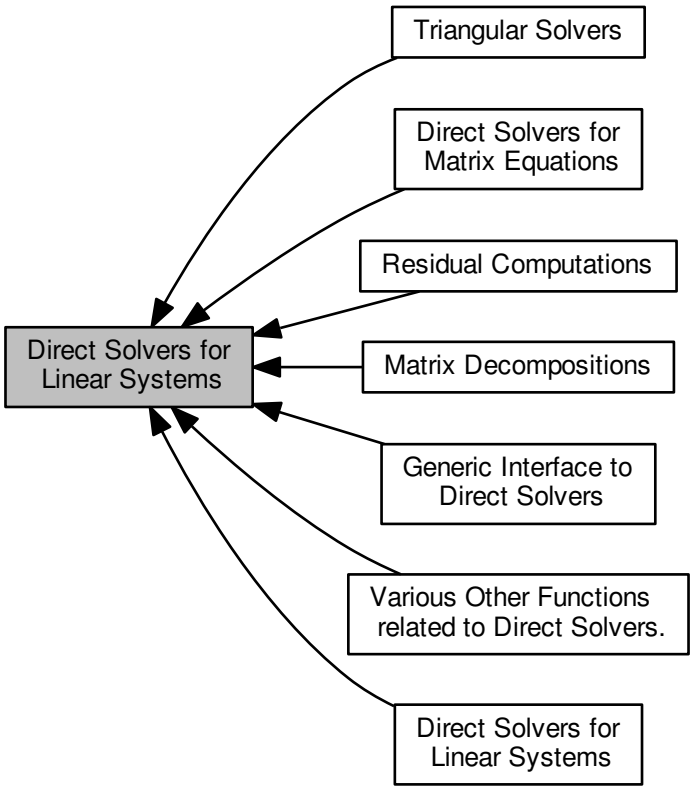
zero on success or a non-zero error value otherwise

The mess_vector_zeros function fills a vector v with zeros.

8.15 Direct Solvers for Linear Systems

This group contains various direct solvers for linear systems and other useful subroutines from this area.

Collaboration diagram for Direct Solvers for Linear Systems:



Modules

- Generic Interface to Direct Solvers

  This group describes the interface to the direct solvers.
- Direct Solvers for Linear Systems

  This group contains solvers for normal linear systems.
- Triangular Solvers

  This module contains functions to solve different triangular systems.
- Direct Solvers for Matrix Equations

  This group contains direct solver for matrix Equations like the Lyapunov Equation.
- Matrix Decompositions

  This group contains matrix decompositions which are strongly related to direct solvers.
- Residual Computations

  This group contains various dense two norm residual computations.
- Various Other Functions related to Direct Solvers.

  This group contains various other function related to the direct solution of linear systems. Mostly used internally.

8.15.1 Detailed Description

The mess_direct object and the corresponding function represent various direct solvers in **M.E.S.S.**. Depending on the configuration solvers from LAPACK and SuiteSparse are integrated. The basic operations to deal with a solver object is described in Generic Interface to Direct Solvers. Beside solvers for

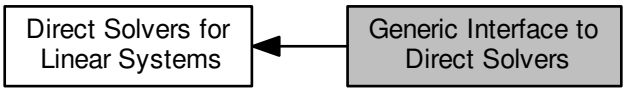
$$Ax = b$$

there exist direct solvers for Lyapunov Equations and other some matrix decompositions, too.

8.16 Generic Interface to Direct Solvers

This group describes the interface to the direct solvers.

Collaboration diagram for Generic Interface to Direct Solvers:



Data Structures

- struct mess_direct_st

  Representation of a generic direct solver.

Macros

- #define MESS_HAVE_SOLVE(solver) ((solver)->solve != NULL)
- #define MESS_HAVE_SOLVET(solver) ((solver)->solvet != NULL)
- #define MESS_HAVE_SOLVEH(solver) ((solver)->solveh != NULL)
- #define MESS_HAVE_SOLVEM(solver) ((solver)->solvem != NULL)
- #define MESS_HAVE_SOLVEMT(solver) ((solver)->solvemt != NULL)
- #define MESS_HAVE_SOLVEMH(solver) ((solver)->solvemh != NULL)
- #define MESS_HAVE_GETL(solver) ((solver)->getL != NULL)
- #define MESS_HAVE_GETU(solver) ((solver)->getU != NULL)
- #define MESS_HAVE_GETPERMP(solver) ((solver)->getpermp != NULL)
- #define MESS_HAVE_GETPERMQ(solver) ((solver)->getpermq != NULL)
- #define MESS_HAVE_GETSCALEROW(solver) ((solver)->getscalerow != NULL)
- #define MESS_HAVE_GETSCALECOL(solver) ((solver)->getscalecol != NULL)
- #define MESS_HAVE_DET(solver) ((solver)->det != NULL)
- #define MESS_HAVE_DETC(solver) ((solver)->det != NULL)
- #define MESS_HAVE_INVERSE(solver) ((solver)->inverse != NULL)

Typedefs

- typedef struct mess_direct_st * mess_direct

  Type definition to hide the pointer on the mess_direct_st structure.

Functions

- int mess_direct_init (mess_direct *direct)

  Initialize a mess_direct object.
- int mess_direct_clear (mess_direct *direct)

  Clear a mess_direct object.
- int mess_direct_solve (mess_operation_t op, mess_direct solver, mess_vector b, mess_vector x)

  Solve $op(A)x = b$, where b and x are vectors.
- int mess_direct_solvem (mess_operation_t op, mess_direct solver, mess_matrix b, mess_matrix x)

  Solve $op(A)X = B$, where B and X are matrices.
- int mess_direct_determinant (mess_direct solver, double *m, double *e)

  Calculate the determinant of a matrix.
- int mess_direct_determinantc (mess_direct solver, double *mr, double *mi, double *e)

  Calculate the determinant of a complex matrix.
- int mess_direct_getL (mess_direct solver, mess_matrix L)

  Retrieve the left matrix of a decomposition.
- int mess_direct_getU (mess_direct solver, mess_matrix U)

  Retrieve the right matrix of a decomposition.
- int mess_direct_getpermp (mess_direct solver, mess_int_t *p)

  Get the row permutation of a decomposition.
- int mess_direct_getpermq (mess_direct solver, mess_int_t *q)

  Get the column permutation of a decomposition.
- int mess_direct_getscalerow (mess_direct solver, mess_vector r)

  Get the row scaling factors of a decomposition.
- int mess_direct_getscalecol (mess_direct solver, mess_vector c)

  Get the col scaling factors of a decomposition.
- int mess_direct_inverse (mess_direct solver, mess_matrix inv)

  Compute the inverse of a matrix.
- int mess_matrix_backslash (mess_operation_t op, mess_matrix A, mess_vector b, mess_vector x)

  Solve $op(A)x = b$ with b and x vectors.
- int mess_matrix_backslashm (mess_operation_t op, mess_matrix A, mess_matrix b, mess_matrix x)

  Solve $op(A)x = b$ with b and x matrices.
- int mess_matrix_pinv (mess_matrix A, mess_matrix Pinv)

  Compute the pseudoinverse of a dense matrix A.

8.16.1 Detailed Description

The generic interface provides an unified interface for various direct solvers. Independently from the underlying solver the linear system could solved in the same way every time. The interface dynamically wraps the correct solve function for the selected solver.

8.16.2 Macro Definition Documentation

8.16.2.1 #define MESS_HAVE_DET(*solver*) ((solver)->det != NULL)

Check if the solver provide a det function.

8.16.2.2 #define MESS_HAVE_DETC(*solver*) ((solver)->det != NULL)

Check if the solver provide a detc function.

8.16.2.3 #define MESS_HAVE_GETL(*solver*) ((solver)->getL != NULL)

Check if the solver provide a getL function.

8.16.2.4 #define MESS_HAVE_GETPERMP(*solver*) ((solver)->getpermp != NULL)

Check if the solver provide a getpermp function.

8.16.2.5 #define MESS_HAVE_GETPERMQ(*solver*) ((solver)->getpermq != NULL)

Check if the solver provide a getpermq function.

8.16.2.6 #define MESS_HAVE_GETSCALECOL(*solver*) ((solver)->getscalecol != NULL)

Check if the solver provide a getscalecol function.

8.16.2.7 #define MESS_HAVE_GETSCALEROW(*solver*) ((solver)->getscalerow != NULL)

Check if the solver provide a getscalerow function.

8.16.2.8 #define MESS_HAVE_GETU(*solver*) ((solver)->getU != NULL)

Check if the solver provide a getU function.

8.16.2.9 #define MESS_HAVE_INVERSE(*solver*) ((solver)->inverse != NULL)

Check if the solver provide a inverse function.

8.16.2.10 #define MESS_HAVE_SOLVE(*solver*) ((solver)->solve != NULL)

Check if the solver provide a solve function.

8.16.2.11 #define MESS_HAVE_SOLVEH(*solver*) ((solver)->solveh != NULL)

Check if the solver provide a solveh function.

8.16.2.12 #define MESS_HAVE_SOLVEM(*solver*) ((solver)->solvem != NULL)

Check if the solver provide a solvem function.

8.16.2.13 #define MESS_HAVE_SOLVEMH(*solver*) ((solver)->solvemh != NULL)

Check if the solver provide a solvemh function.

8.16.2.14 #define MESS_HAVE_SOLVEMT(*solver*) ((solver)->solvemt != NULL)

Check if the solver provide a solvemt function.

8.16.2.15 #define MESS_HAVE_SOLVET(*solver*) ((solver)->solvet != NULL)

Check if the solver provide a solvet function.

8.16.3 Typedef Documentation

8.16.3.1 typedef struct mess_direct_st* mess_direct

The mess_direct type definition hides one pointer on the mess_direct_st structure.
This type definition is the way how the direct solver interface should be used.

8.16.4 Function Documentation

8.16.4.1 int mess_direct_clear (mess_direct * *direct*)

Parameters

in,out	*direct*	pointer to the mess_direct object

Returns

always zero.

The mess_direct_clear function clears a mess_direct object and calls the clear function of the solver.

8.16.4.2 int mess_direct_determinant (mess_direct *solver*, double * *m*, double * *e*)

Parameters

in	*solver*	input solver build up from the matrix
out	*m*	output mantissa of determinant
out	*e*	output exponent of determinant

Returns

zero on success or a non-zero error value

The mess_direct_determinant function computes the determinant of a matrix. It returns an error if the underlying solver does not provide a det function. Unfortunately most of the solvers does not have such a function. This function should be used for real matrices. To avoid overflow problems the determinant is returned as $det=m*2^e$.

8.16.4.3 int mess_direct_determinantc (mess_direct *solver*, double * *mr*, double * *mi*, double * *e*)

Parameters

in	*solver*	input solver build up from the matrix
out	*mr*	output mantissa for real part of determinant
out	*mi*	output mantissa for imaginary part of determinant
out	*e*	output exponent for real part and imaginary part of determinant

Returns

zero on success or a non-zero error value

The mess_direct_determinantc function computes the determinant of a matrix. It returns an error if the underlying solver does not provide a det function. Unfortunately most of the solvers does not have such a function. This function works for complex matrices. To avoid overflow problems the determinant is returned as $det=m_r*2^e+m_i*2^e*I$.

8.16.4.4 int mess_direct_getL (mess_direct *solver*, mess_matrix *L*)

Parameters

in	*solver*	input solver
out	*L*	output left matrix matrix L

Returns

zero on success or a non-zero error value

The mess_direct_getL function gets the left matrix of a decomposition. It returns an error if the corresponding function is not available for the solver.

8.16.4.5 int mess_direct_getpermp (mess_direct *solver*, mess_int_t * *p*)

Parameters

in	*solver*	input solver
out	*p*	output row permutation

Returns

zero on success or a non-zero error value

The mess_direct_getpermp function gets the row permutation of a decomposition. The array p must be preallocated.

8.16.4.6 int mess_direct_getpermq (mess_direct *solver*, mess_int_t * *q*)

Parameters

in	*solver*	input solver
out	*q*	output column permutation

Returns

zero on success or a non-zero error value

The mess_direct_getpermq function gets the column permutation of a decomposition. The array q must be preallocated.

8.16.4.7 int mess_direct_getscalecol (mess_direct *solver*, mess_vector *c*)

Parameters

in	*solver*	input solver
out	*c*	output col scaling factors

Returns

zero on success or a non-zero error value

The mess_direct_getscalecol function gets the col scaling factors of a decomposition.

8.16.4.8 int mess_direct_getscalerow (mess_direct *solver*, mess_vector *r*)

Parameters

in	*solver*	input solver
out	*r*	output row scaling factors

Returns

zero on success or a non-zero error value

The mess_direct_getscalerow function gets the row scaling factors of a decomposition.

8.16.4.9 int mess_direct_getU (mess_direct *solver*, mess_matrix *U*)

Parameters

in	*solver*	input solver
out	*U*	output right matrix matrix L

Returns

zero on success or a non-zero error value

The mess_direct_getU function gets the right matrix of a decomposition. It returns an error if the corresponding function is not available for the solver.

8.16.4.10 int mess_direct_init (mess_direct * *direct*)

Parameters

in,out	*direct*	pointer to the mess_direct object

Returns

zero on success or a non-zero error value

The mess_direct_init function initializes a mess_direct object.

8.16.4.11 int mess_direct_inverse (mess_direct *solver*, mess_matrix *inv*)

Parameters

in	*solver*	input solver for the matrix
out	*inv*	inverse of the matrix

Returns

zero on success or a non-zero error value

The mess_direct_inverse function computes the inverse of a matrix. Depending on the solver this is a very time consuming operation.

8.16.4.12 int mess_direct_solve (mess_operation_t *op*, mess_direct *solver*, mess_vector *b*, mess_vector *x*)

Parameters

in	*op*	input operation applied to A
in	*solver*	input solver to use
in	*b*	input right-hand-side matrix of the equation
out	*x*	solution matrix of the equation

Returns

zero on success or a non-zero error value

The mess_direct_solve function solves $op(A)x = b$. The operation op can be

- MESS_OP_NONE ($op(A) = A$)
- MESS_OP_TRANSPOSE ($op(A) = A^T$)
- MESS_OP_HERMITIAN ($op(A) = A^H$).

The function calls the corresponding function of internal solver. If the solver is real then MESS_OP_TRANSPOSE and MESS_OP_HERMITIAN are the same. If a solver does not provide a corresponding internal solver function for vector right-hand-sides or the desired operation an error is returned.

8.16.4.13 int mess_direct_solvem (mess_operation_t *op*, mess_direct *solver*, mess_matrix *B*, mess_matrix *X*)

Parameters

in	*op*	input operation applied to A
in	*solver*	input solver to use
in	*B*	input right-hand-side matrix of the equation
out	*X*	solution matrix of the equation

Returns

zero on success or a non-zero error value

The mess_direct_solvem function solves $op(A)X = B$. The operation op can be

- MESS_OP_NONE ($op(A) = A$)
- MESS_OP_TRANSPOSE ($op(A) = A^T$)
- MESS_OP_HERMITIAN ($op(A) = A^H$).

The function calls the corresponding function of internal solver. If the solver is real then MESS_OP_TRANSPOSE and MESS_OP_HERMITIAN are the same. If a solver does not provide a corresponding internal solver function for multiple right hand sides or the desired operation an error is returned.

8.16.4.14 int mess_matrix_backslash (mess_operation_t *op*, mess_matrix *A*, mess_vector *b*, mess_vector *x*)

Parameters

in	*op*	input operation on A
in	*A*	input system matrix A
in	*b*	input right hand side b
out	*x*	output solution

Returns

zero on success or a non zero error code otherwise

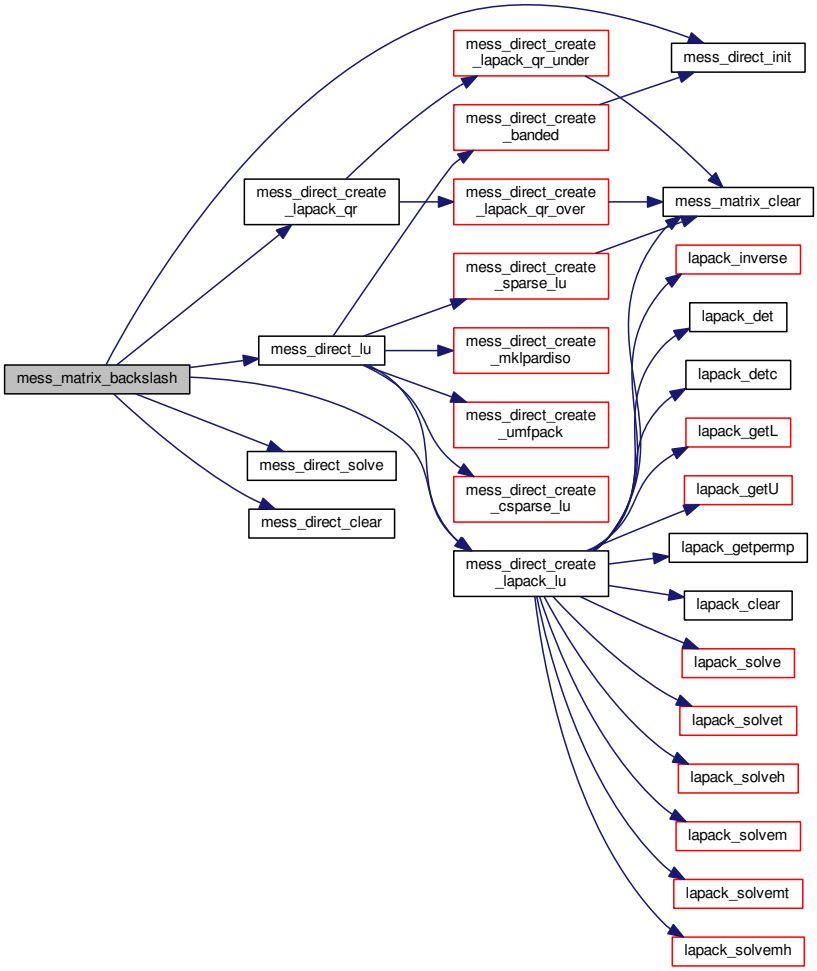
The mess_matrix_backslash function solves the system $op(A)x = b$.
The solver (sparse, dense) is detected automatically.
If you plan to use the matrix A more than one time to solve system please use a mess_direct instance.
The operation op can be one of

- MESS_OP_NONE, that means $op(A) = A$
- MESS_OP_TRANSPOSE, that means $op(A) = A^T$
- MESS_OP_HERMITIAN, that means $op(A) = A^H$.

Attention

This function only works if b and x are vectors. If the right hand side is a matrix please use mess_matrix_backslashm.

Here is the call graph for this function:



8.16.4.15 int mess_matrix_backslashm (mess_operation_t *op*, mess_matrix *A*, mess_matrix *b*, mess_matrix *x*)

Parameters

in	*op*	input operation on A
in	*A*	input system matrix A
in	*b*	input right hand side b
out	*x*	output solution x

Returns

zero on success or a non zero error code otherwise

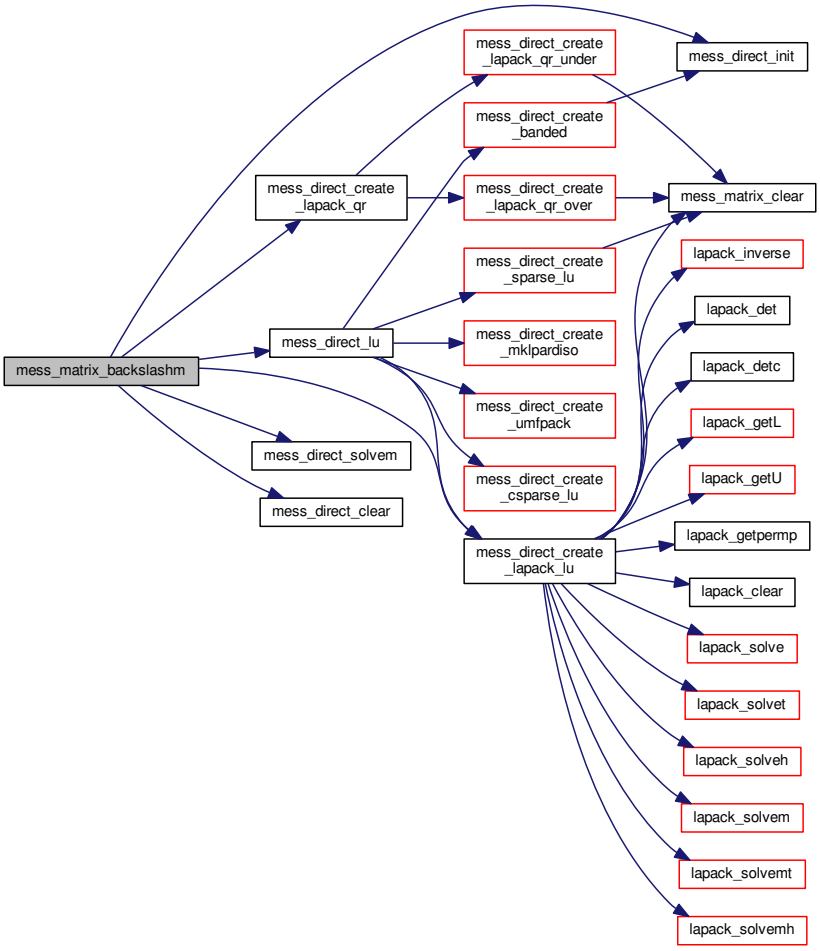
The mess_matrix_backslashm function solves the system $op(A)x = b$.
The solver (sparse, dense) is detected automatically.
If you plan to use the matrix A more than one time to solve system please use a mess_direct instance.
The operation op can be one of

- MESS_OP_NONE, that means $op(A) = A$
- MESS_OP_TRANSPOSE, that means $op(A) = A^T$
- MESS_OP_HERMITIAN, that means $op(A) = A^H$.

Attention

This function only works if b and x are matrices. If the right hand side is a vector please use mess_matrix_backslash.

Here is the call graph for this function:



8.16.4.16 int mess_matrix_pinv (mess_matrix *A*, mess_matrix *Pinv*)

Parameters

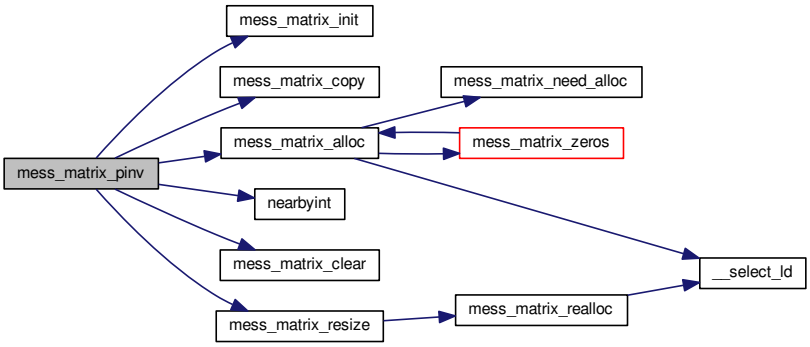
in	*A*	input A
out	*Pinv*	output P_{inv} pseudoinverse of A

The mess_matrix_pinv computes the pseudoinverse of a dense matrix using LAPACK `dgels` or `zgels` methods.

See also

mess_direct_inverse
mess_eigen_svd

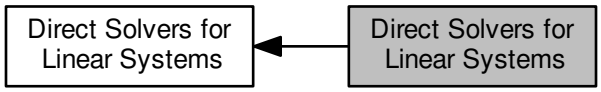
Here is the call graph for this function:



8.17 Direct Solvers for Linear Systems

This group contains solvers for normal linear systems.

Collaboration diagram for Direct Solvers for Linear Systems:



Enumerations

Functions

- int mess_direct_lu (mess_matrix matrix, mess_direct direct)

  Compute a LU decomposition of a matrix.
- int mess_direct_lu_select (mess_direct_lupackage_t lupackage)

  Change the default behaviour of mess_direct_lu.
- int mess_direct_chol (mess_matrix matrix, mess_direct direct)

  Compute an Chol decomposition of a matrix.
- int mess_direct_chol_select (mess_direct_cholpackage_t cholpackage)

  Change the default behaviour of mess_direct_chol.
- int mess_direct_create_umfpack (mess_matrix matrix, mess_direct direct)

  Generate a direct linear solver for standard linear systems $Ax = b$ with `UMFPACK`
- int mess_direct_create_umfpack_control (mess_matrix matrix, mess_direct direct, double *Control)

  Generate a direct linear solver for standard linear systems $Ax = b$ with `UMFPACK`.
- int mess_direct_create_csparse_lu (mess_matrix matrix, mess_direct direct)

  Generate a CXSparse LU solver.
- int mess_direct_create_csparse_cholesky (mess_matrix matrix, mess_direct direct)

  Generate a CXSparse Cholesky solver.
- int mess_direct_create_banded (mess_matrix matrix, mess_direct direct)

  Generate a direct linear banded solver for standard linear systems $Ax = b$ with LAPACK.
- int mess_direct_create_bicgstab (mess_matrix matrix, mess_direct solver)

  Generate a bicgstab solver in order to use it as direct solver.
- int mess_direct_create_lapack_lu (mess_matrix matrix, mess_direct solver)

  Generate a dense `LAPACK` *based LU-solver.*
- int mess_direct_create_cholesky (mess_matrix matrix, mess_direct solver)

  Generate a dense `LAPACK` *based Cholesky solver.*
- int mess_direct_create_lapack_qr (mess_matrix matrix, mess_direct solver)

  Construct a solver based on QR or LQ decomposition from `LAPACK`.
- int mess_direct_create_sparse_lu (mess_matrix A, mess_direct sol)

  Compute a sparse left-looking LU decomposition.
- int mess_direct_create_hessenberg_lu (mess_matrix matrix, mess_direct solver)

  Generate a dense `LAPACK` *based LU-solver for upper hessenberg matrices.*

8.17.1 Detailed Description

This module provides direct solvers, which act as backend for mess_direct. Various solvers like LU, Cholesky or QR decompositions are available. An automatic selection system depending on the installed features and the given matrix is available, too. This can be used via mess_direct_lu and mess_direct_chol. Additionally there are enumeration types in this category to select the behaviour of a Cholesky and LU decomposition.

8.17.2 Enumeration Type Documentation

8.17.2.1 enum mess_direct_cholpackage_t

The mess_direct_cholpackage_t type is an enumeration to select the behavior of mess_direct_chol.
The value MESS_DIRECT_DEFAULT_CHOLESKY selects the default behavior. That means depending on the mess_storage_t type of the mess_matrix a suitable and available solver will be selected. All other enumeration items fix it to the desired solver.

Enumerator

MESS_DIRECT_DEFAULT_CHOLESKY Select the automatic solver detection.

MESS_DIRECT_LAPACK_CHOLESKY Select the `LAPACK` dense solver.

MESS_DIRECT_CSPARSE_CHOLESKY Select the `CSparse` sparse solver.

MESS_DIRECT_CHOLMOD_CHOLESKY Select the `CHOLMOD` sparse solver.

8.17.2.2 enum mess_direct_lupackage_t

The mess_direct_lupackage_t type is an enumeration to select the behavior of mess_direct_lu.
The value MESS_DIRECT_DEFAULT_LU selects the default behavior. That means depending of the storage type of the mess_matrix object a suitable and available solver will be selected. All other enumeration items fix it to the desired solver.

Enumerator

MESS_DIRECT_DEFAULT_LU Select the automatic solver detection.

MESS_DIRECT_SPARSE_LU Select the internal sparse solver.

MESS_DIRECT_LAPACK_LU Select the `LAPACK` dense solver.

MESS_DIRECT_UMFPACK_LU Select the `UMFPACK` sparse solver.

MESS_DIRECT_SUPERLU_LU Select the `SuperLU` sparse solver.

MESS_DIRECT_CSPARSE_LU Select the `CSparse` sparse solver.

MESS_DIRECT_BANDED_LU Select the banded sparse solver.

MESS_DIRECT_MKLPARDISO_LU Select the mess_direct_create_mklpardiso solver.

8.17.3 Function Documentation

8.17.3.1 int mess_direct_chol (mess_matrix *matrix*, mess_direct *solver*)

Parameters

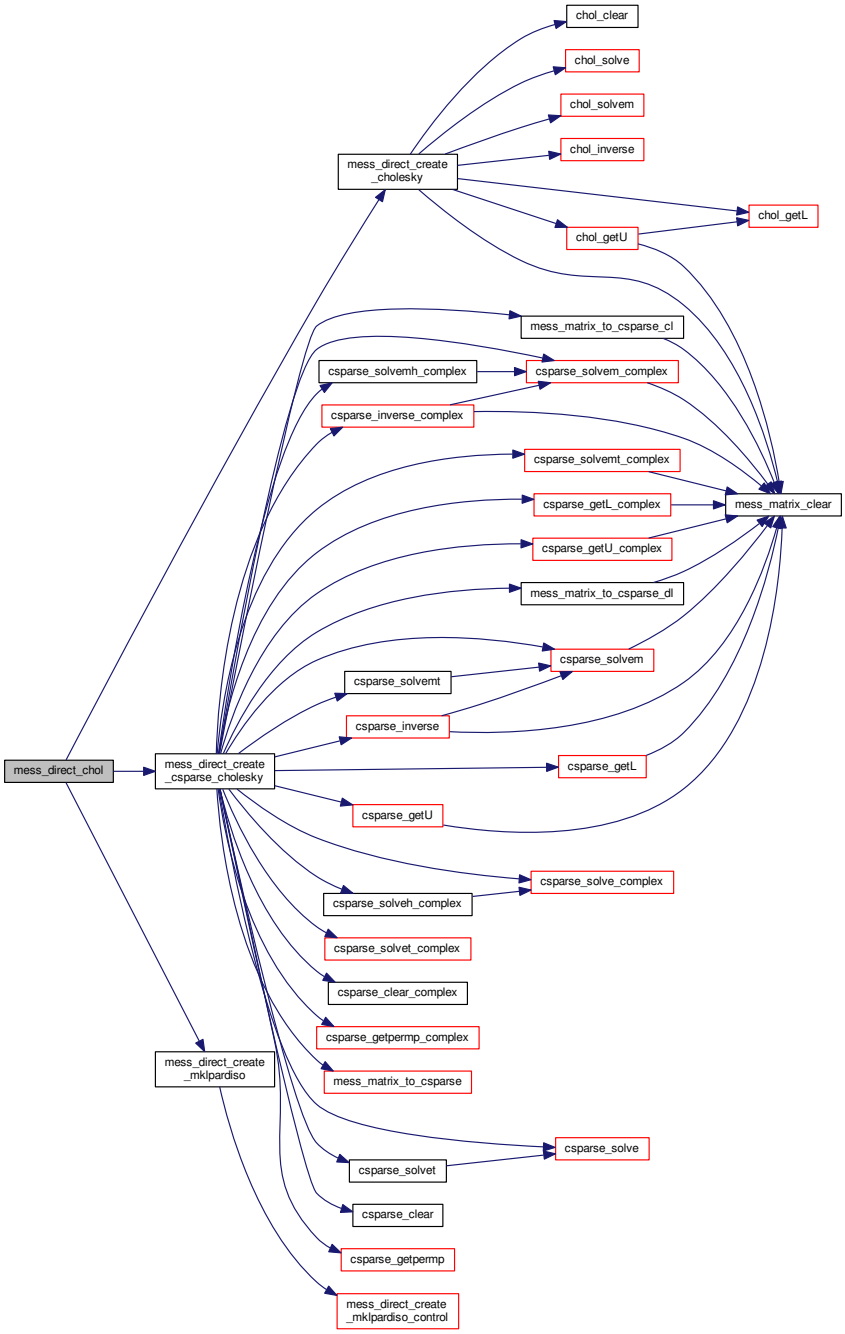
in	*matrix*	input matrix which should be decomposed
out	*solver*	solver which contains a Cholesky decomposition of the input matrix

Returns

zero on success or a non-zero error value

The mess_direct_chol function computes the Cholesky decomposition of a given matrix. In the default case it selects the solver which works best for the matrix. This means in case of a sparse matrix it tries to use `CSparse` or in case of a dense matrix it uses `LAPACK`. For the sparse case are fallback situations implemented, such that it works even if `CSparse` is not available. This default selecting behaviour can be changed using mess_direct_chol_select.

Here is the call graph for this function:



8.17.3.2 int mess_direct_chol_select (mess_direct_cholpackage_t *chol*)

Parameters

in	*chol*	input type of the Cholesky decomposition used by mess_direct_chol

Returns

zero on success or a non-zero error value

The mess_direct_chol_select function selects the solver package which should be used for the mess_direct_chol function. It overrides the automatic selection of the solver. If the automatic selection should be enabled again, use MESS_DIRECT_DEFAULT_CHOLESKY as mess_direct_cholpackage_t.

8.17.3.3 int mess_direct_create_banded (mess_matrix *matrix*, mess_direct *solver*)

Parameters

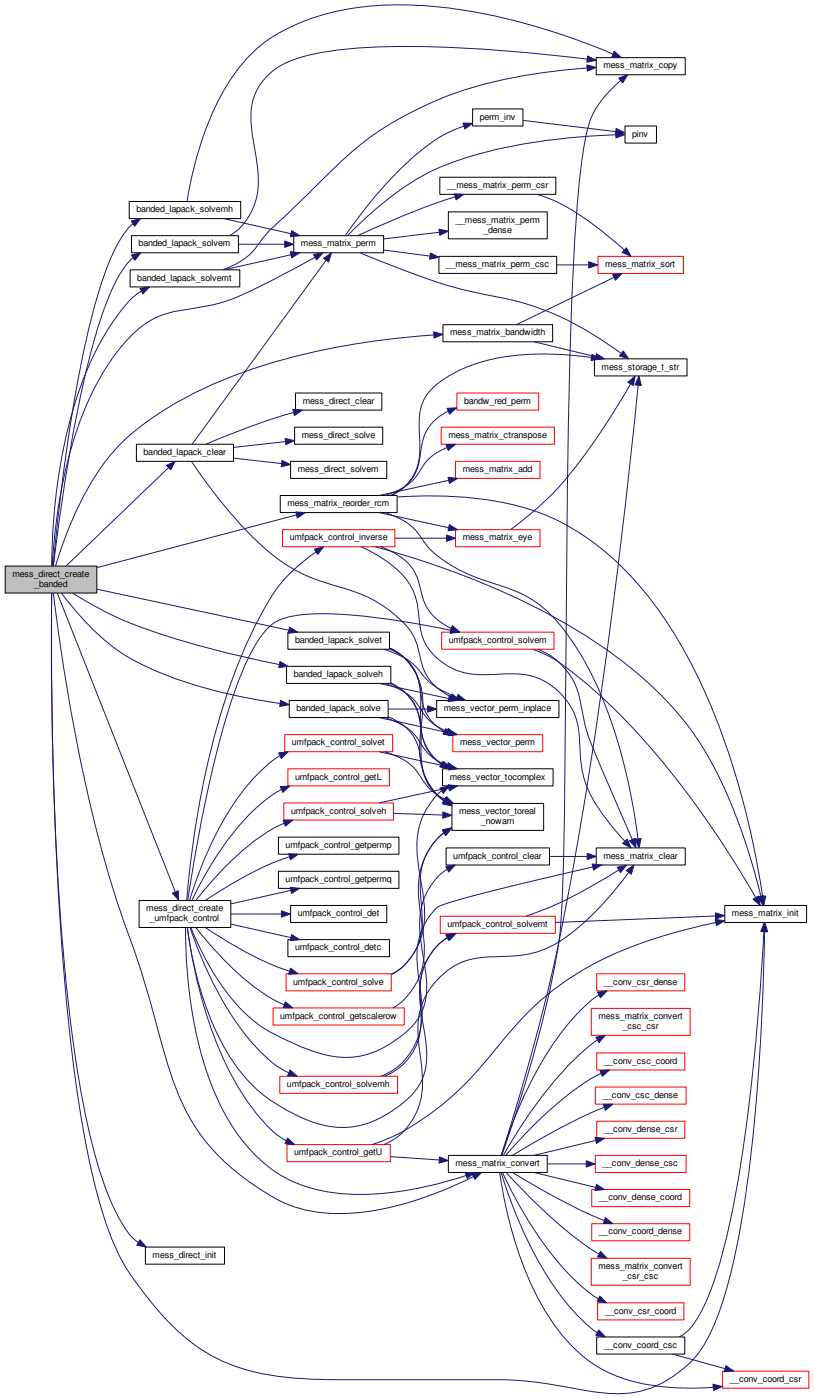
in	*matrix*	matrix to decompose
out	*solver*	output solver

Returns

zero on success or a non zero error value otherwise

The mess_direct_create_banded function uses bandwidth reducing reordering and LAPACK banded solvers or UMFPACK solver for solving linear systems. If bandwidth after reordering to large we switch to UMFPACK based solver (if available).

Here is the call graph for this function:



8.17.3.4 int mess_direct_create_bicgstab (mess_matrix *matrix*, mess_direct *solver*)

Parameters

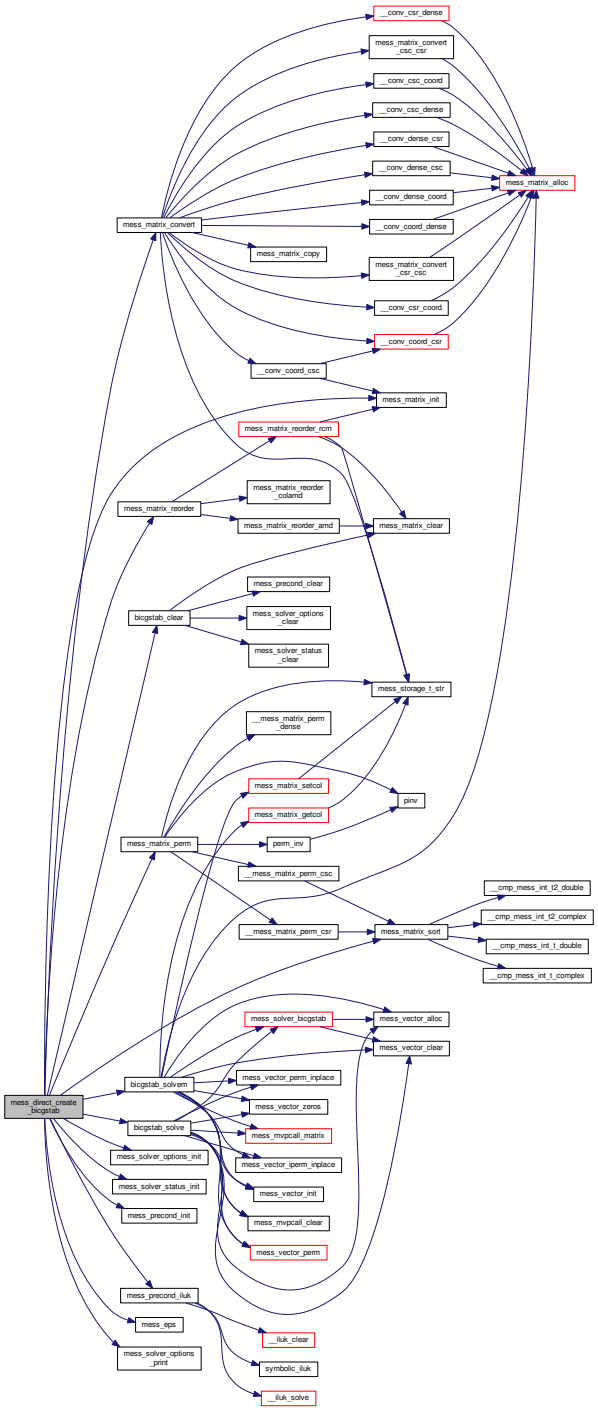
in	*matrix*	input System matrix
out	*solver*	generated solver based on BiCGSTAB

Returns

zero on success or a non-zero error value otherwise

The mess_direct_create_bicgstab function generates a pseudo direct solver.
It relies internally on BiCGSTAB and wraps it in a way that it can be used like a direct solver. It reorders the matrix using the AMD reordering and computes a $ILU(3)$ preconditioner with AMD reordering as default. The maximum number of iterations is at least 100 in real case or 300 in complex case.

Here is the call graph for this function:



8.17.3.5 int mess_direct_create_cholesky (mess_matrix *matrix*, mess_direct *solver*)

Parameters

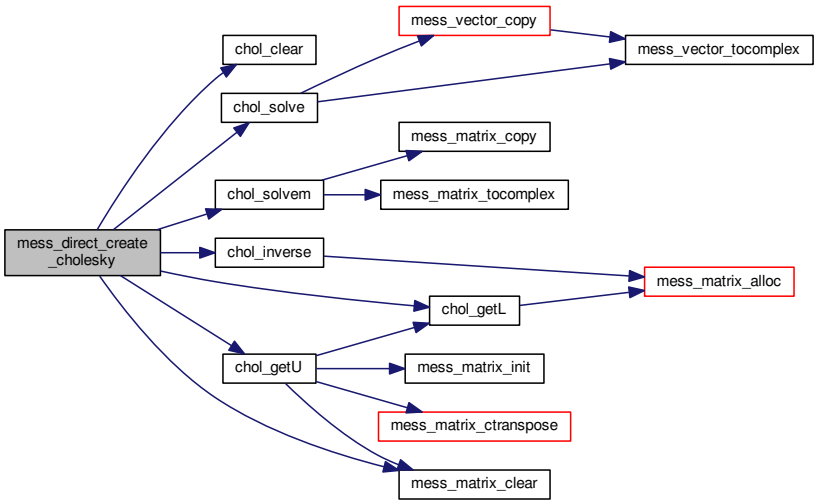
in	*matrix*	symmetric positive definite input matrix
out	*solver*	generated solver

Returns

zero on success or a non-zero error value otherwise

The mess_direct_create_cholesky function generates a Cholesky solver for dense matrices based on LAPACK. It only works for symmetric positive definite matrices. If the matrix does not fulfill this property an error is thrown.

Here is the call graph for this function:



8.17.3.6 int mess_direct_create_csparse_cholesky (mess_matrix *matrix*, mess_direct *solver*)

Parameters

in	*matrix*	input system matrix
in,out	*solver*	output solver

Returns

zero on success or a non zero error code otherwise

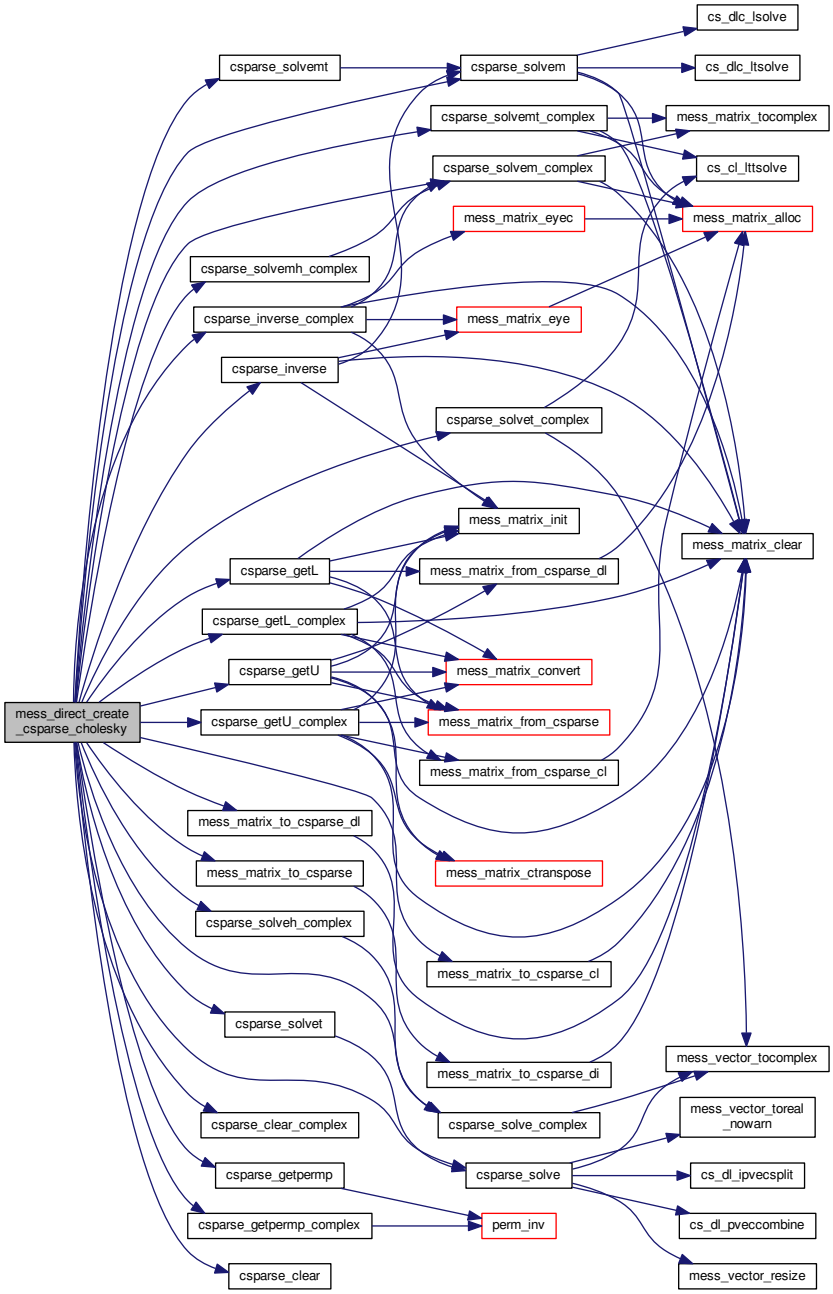
The mess_direct_create_csparse_cholesky function generates a LU-factorization based solver.
It uses CSparse[1] to reorder and factorize the system matrix.
The reordering can be modified by using following runtime configurations:

- mess.csparse.order, 0=no reordering, $1, 2$= different AMD reordering, default 2

- mess.csparse.tol, tolerance in pivoting, default 1.0

[13].

Here is the call graph for this function:



8.17.3.7 int mess_direct_create_csparse_lu (mess_matrix *matrix*, mess_direct *solver*)

Parameters

in	*matrix*	input system matrix
out	*solver*	output solver

Returns

zero on success or a non-zero error value otherwise

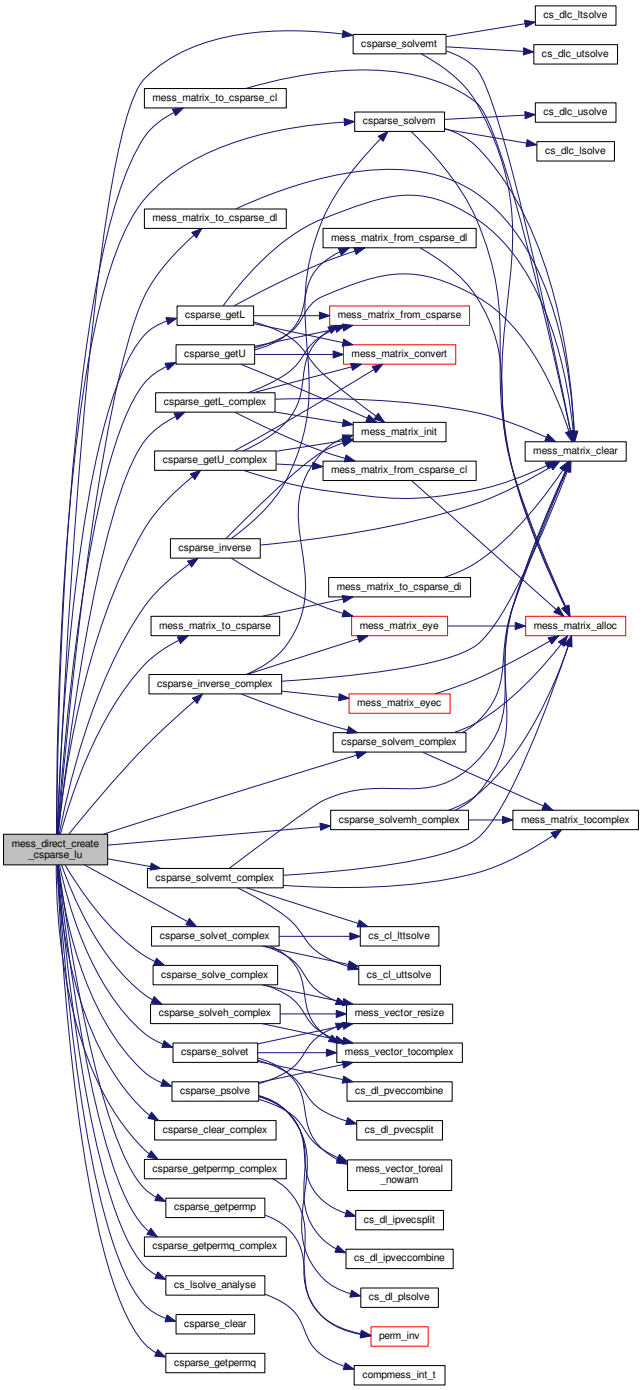
The mess_direct_create_csparse_lu function generates a LU-factorization based solver.
It uses `CSparse`[1] to reorder and factorize the system matrix. The reordering can be modified by using the following runtime configurations.

- mess.csparse.order, 0=no reordering, $1, 2$ = different AMD reordering, default 2

- mess.csparse.tol, tolerance in pivoting, default 1.0

[13].

Here is the call graph for this function:



8.17.3.8 int mess_direct_create_hessenberg_lu (mess_matrix *matrix*, mess_direct *solver*)

Parameters

in	*matrix*	input upper hessenberg matrix to decompose
out	*solver*	output generated solver

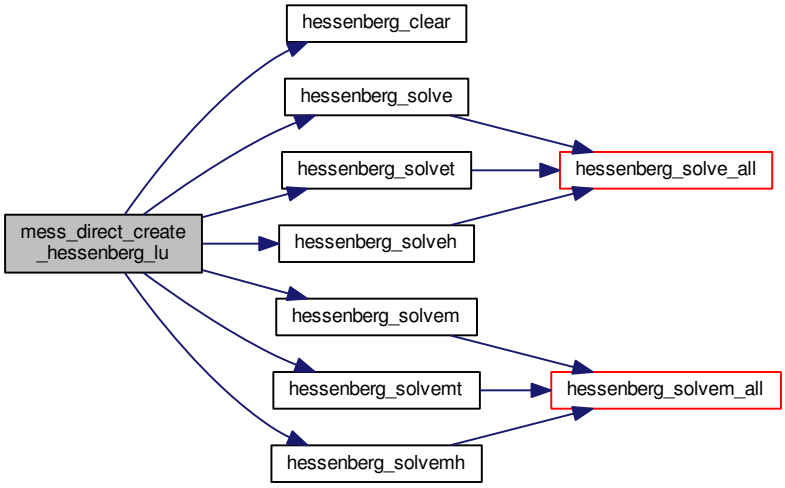
Returns

zero on success or a non-zero error value otherwise

The mess_direct_create_hessenberg_lu function generates a LU solver for dense upper hessenberg matrices based on LAPACK. It is not checked that `matrix` is actually upper Hessenberg,

See [15] Chapter 4.3.4

Here is the call graph for this function:



8.17.3.9 int mess_direct_create_lapack_lu (mess_matrix *matrix*, mess_direct *solver*)

Parameters

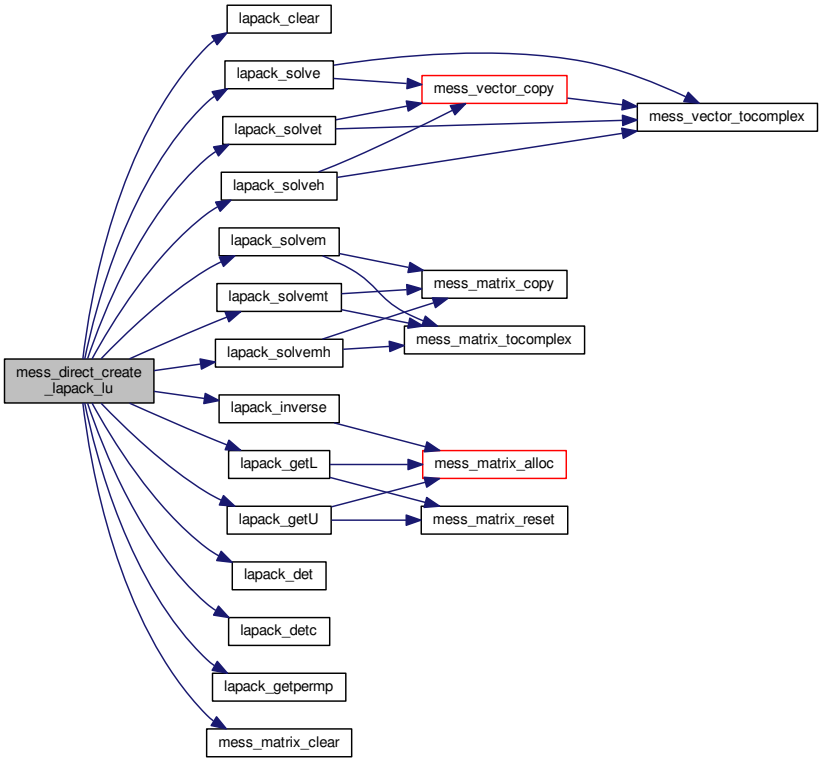
in	*matrix*	input matrix to decompose
out	*solver*	generated solver

Returns

zero on success or a non-zero error value otherwise

The mess_direct_create_lapack_lu function generates a LU solver for dense matrices based on LAPACK.
It uses DGETRF/ZGETRF to compute the LU decomposition.

Here is the call graph for this function:



8.17.3.10 int mess_direct_create_lapack_qr (mess_matrix *matrix*, mess_direct *solver*)

Parameters

in	*matrix*	input system matrix
in,out	*solver*	generated solver

Returns

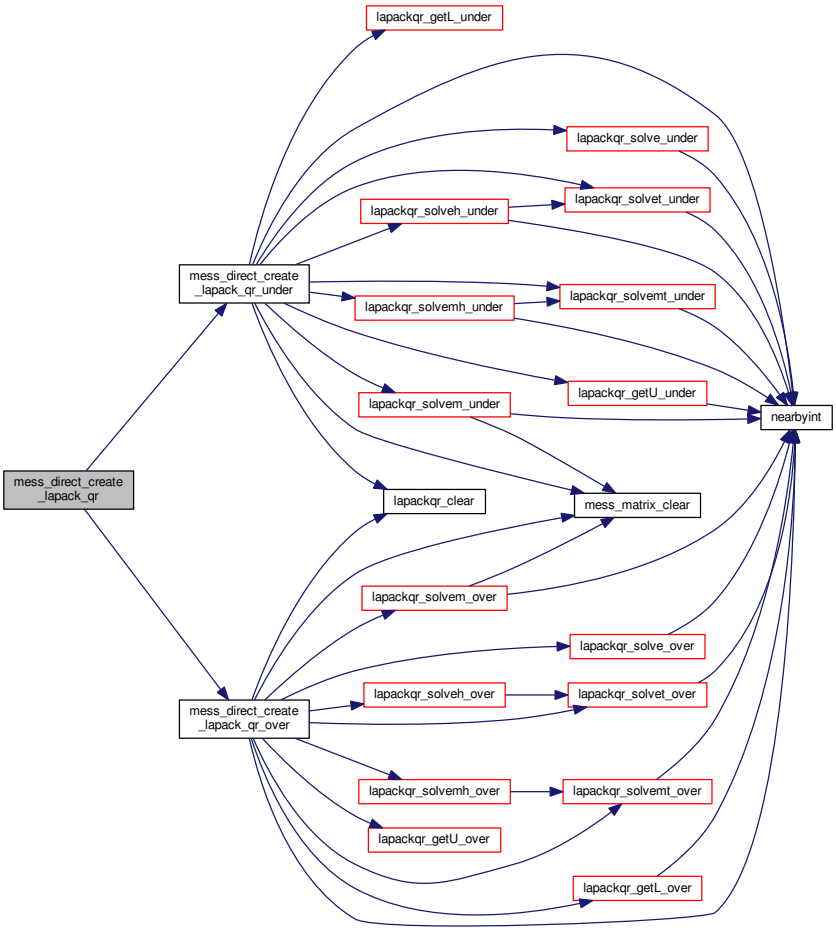
zero on success or a non zero error code otherwise

The mess_direct_create_lapack_qr function generates a solver based on the QR or LQ decomposition from LA↩PACK. If the number of rows is greater equals the number of cols, we perform a QR decomposition, otherwise we perform a LQ decomposition. In both cases it is necessary, that matrix has full rank.

Attention

Please keep in mind, that a sparse matrix is converted to a dense one.

Here is the call graph for this function:



8.17.3.11 int mess_direct_create_sparse_lu (mess_matrix *A*, mess_direct *sol*)

Parameters

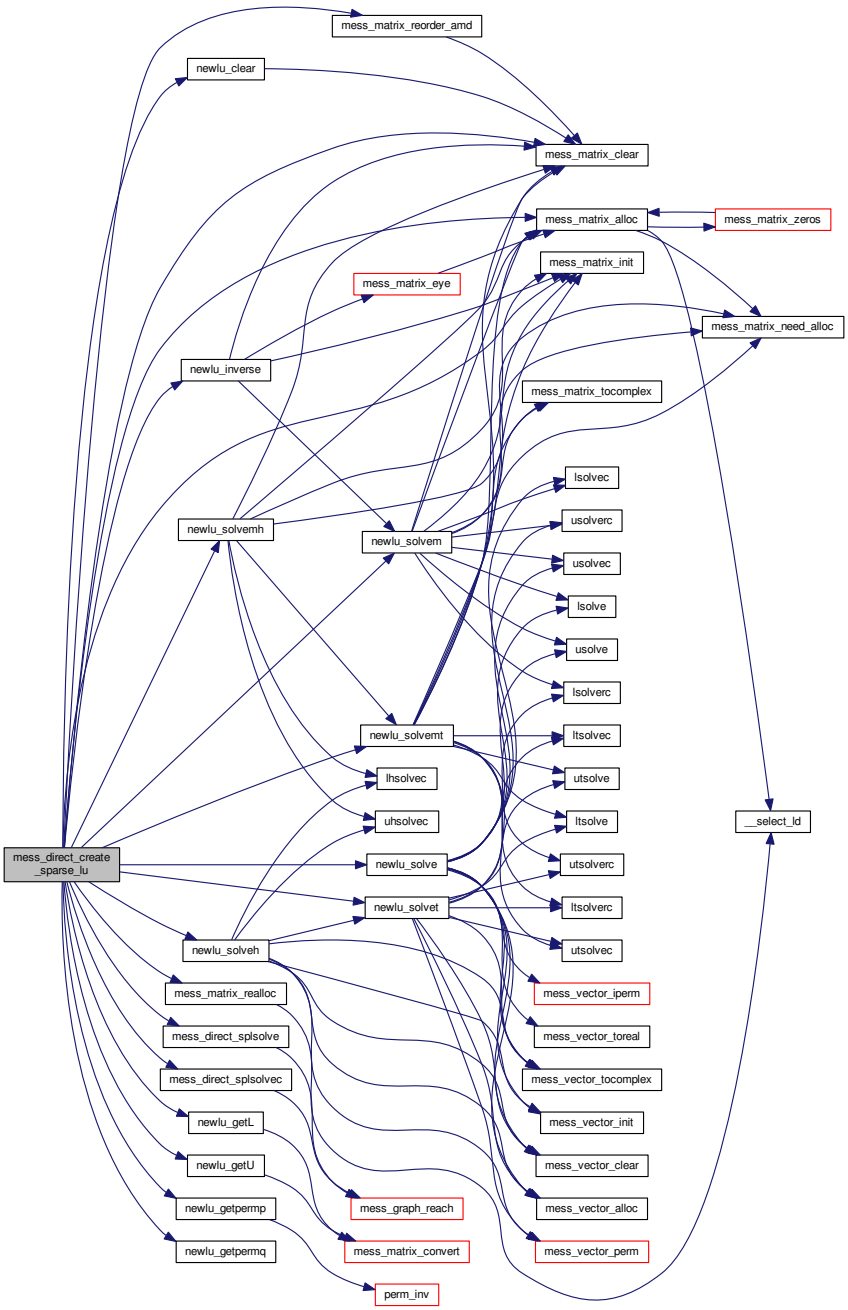
in	*A*	input matrix to factorize
out	*sol*	solver object containing the left-looking sparse LU decomposition

Returns

zero on success or a non zero error code otherwise

The mess_direct_create_sparse_lu function computes a left looking LU decomposition of a sparse matrix.

Here is the call graph for this function:



8.17.3.12 int mess_direct_create_umfpack (mess_matrix *matrix*, mess_direct *solver*)

Parameters

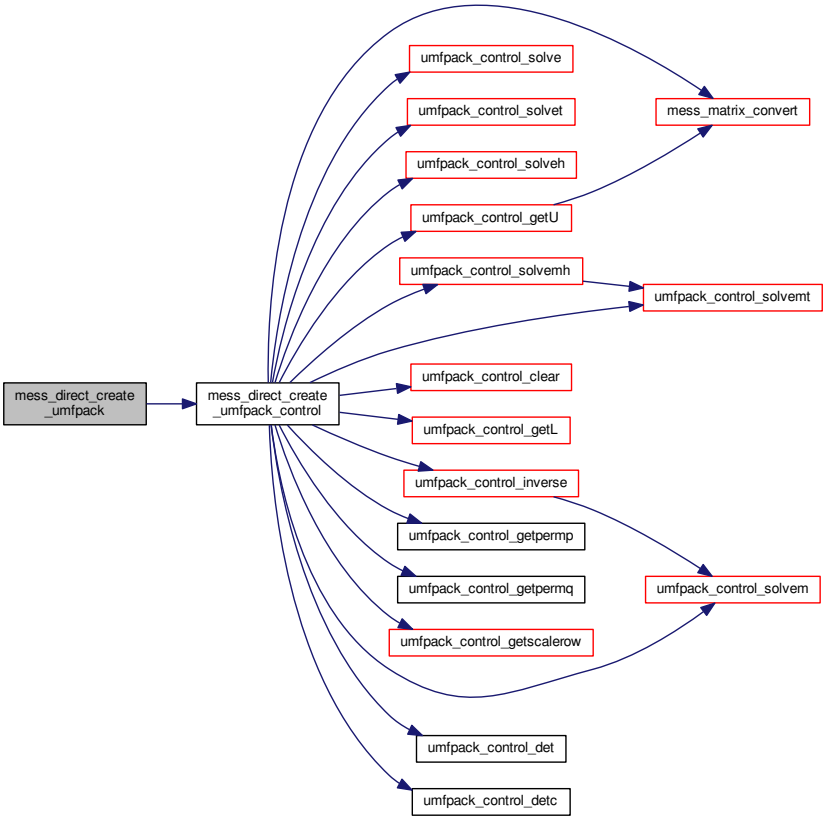
in	*matrix*	matrix to decompose
out	*solver*	output solver

Returns

zero on success or a non zero error value otherwise

The mess_direct_create_umfpack function factorizes a matrix with UMFPACK with iterative refinement and provides a solver for linear systems.

Here is the call graph for this function:



8.17.3.13 int mess_direct_create_umfpack_control (mess_matrix *matrix*, mess_direct *solver*, double * *Control*)

Parameters

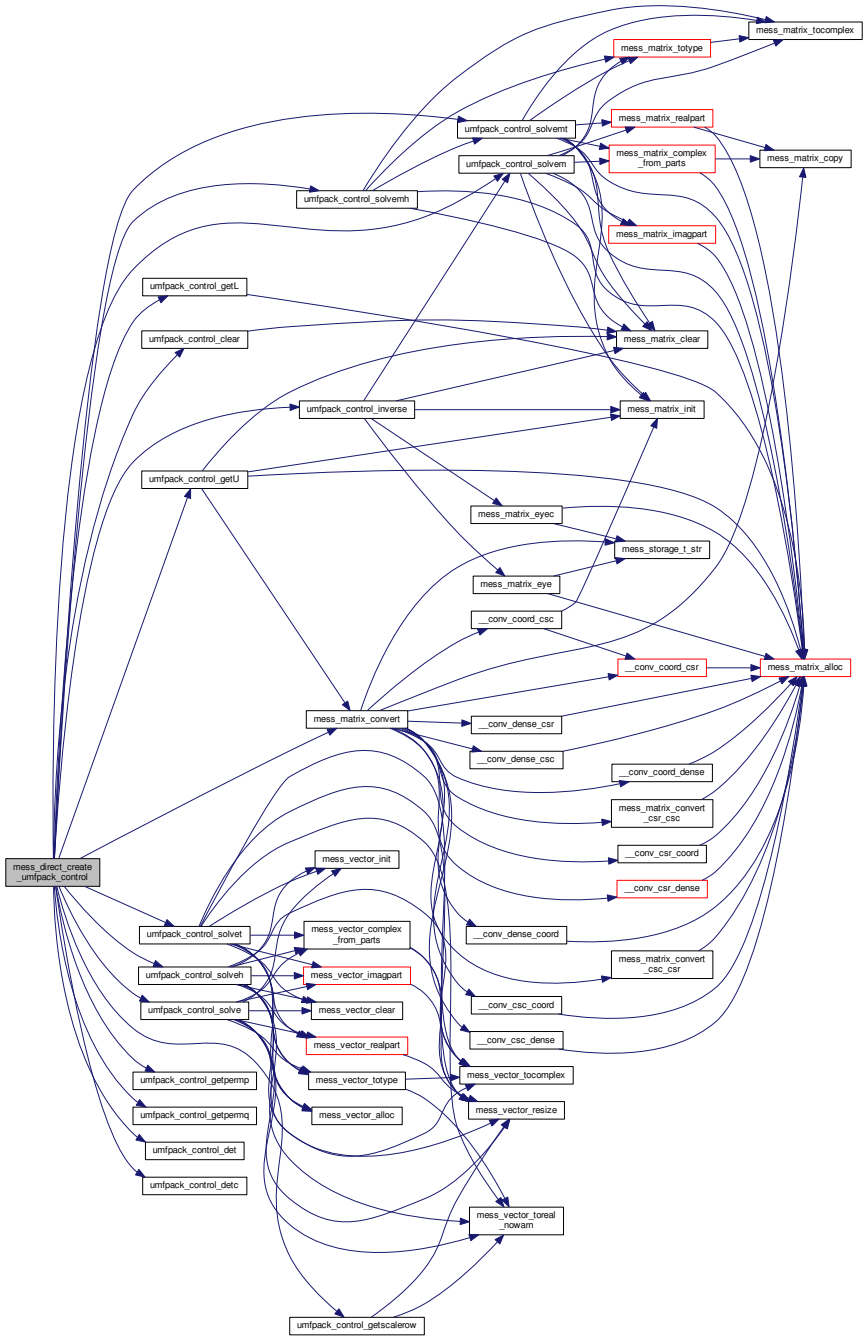
in	*matrix*	matrix to decompose
in,out	*solver*	direct solver
in	*Control*	umfpack control array

Returns

zero on success or a non zero error value otherwise

The mess_direct_create_umfpack_control function factorizes a matrix with UMFPACK. The Control argument provides the ability to control UMFPACK solver options.

Here is the call graph for this function:



8.17.3.14 int mess_direct_lu (mess_matrix *matrix*, mess_direct *solver*)

Parameters

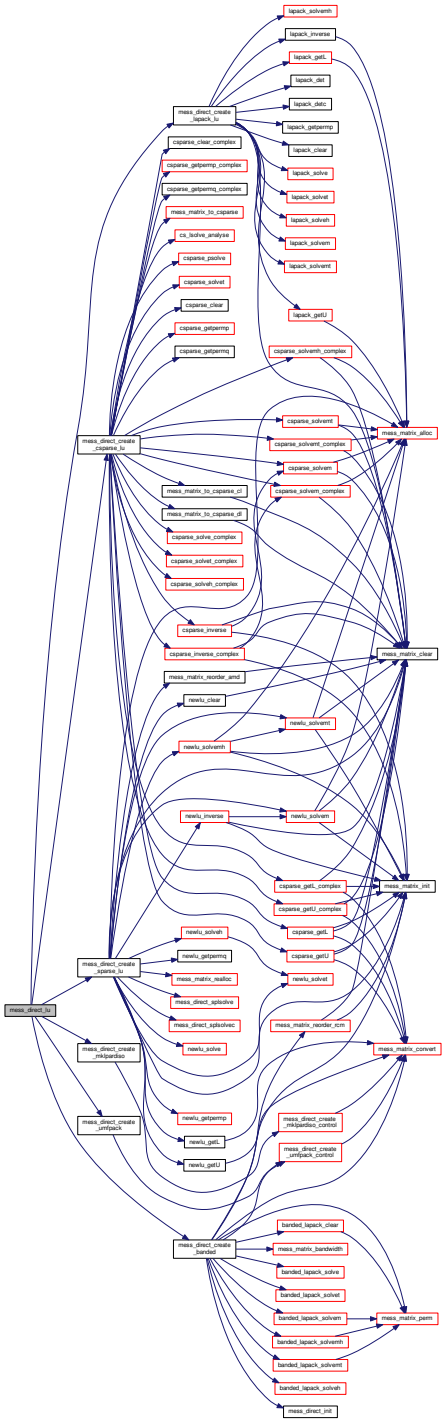
in	*matrix*	input matrix which should be decomposed
out	*solver*	solver which contains a LU decomposition of the input matrix

Returns

zero on success or a non-zero error value

The mess_direct_lu function computes the LU decomposition of a given matrix. In the default case it selects the solver which works best for the matrix. This means in case of a sparse matrix it tries to use UMFPACK or in case of a dense matrix it uses LAPACK. For the sparse case are fallback situations implemented, such that it works even if UMFPACK is not available. This default selecting behaviour can be changed using mess_direct_lu_select.

Here is the call graph for this function:



8.17.3.15 int mess_direct_lu_select (mess_direct_lupackage_t *lu*)

Parameters

in	*lu*	input type of the LU decomposition used by mess_direct_lu

Returns

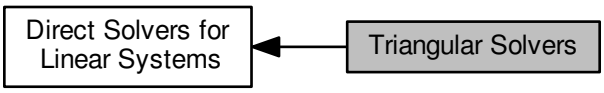
zero on success or a non-zero error value

The mess_direct_lu_select function selects the solver package which should be used for the mess_direct_lu function. It overrides the automatic selection of the solver. If the automatic selection should be enabled again, use MESS_DIRECT_DEFAULT_LU as mess_direct_lupackage_t.

8.18 Triangular Solvers

This module contains functions to solve different triangular systems.

Collaboration diagram for Triangular Solvers:



Functions

- int mess_solver_usolve (mess_matrix U, mess_vector y)

  Solve $Ux = b$ (user interface).
- int mess_solver_usolvem (mess_matrix U, mess_matrix Y)

  Solve $UX = B$ (user interface).
- int mess_solver_usolve_kernelcsr_real (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, double *y)

  Solve $Ux = b$ (CSR, kernel only, real).
- int mess_solver_usolve_kernelcsr_real_complex (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $Ux = b$ (CSR, kernel only, real U, complex y).
- int mess_solver_usolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $Ux = b$ (CSR, kernel only, complex).
- int mess_solver_lsolve (mess_matrix L, mess_vector y)

  Solve $Lx = b$ (user interface).
- int mess_solver_lsolvem (mess_matrix L, mess_matrix Y)

  Solve $LX = B$ (user interface).
- int mess_solver_lsolve_kernelcsr_real (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, double *y)

  Solve $Lx = b$ (CSR, kernel only, real).
- int mess_solver_lsolve_kernelcsr_real_complex (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $Lx = b$ (CSR, kernel only, real L, complex b).
- int mess_solver_lsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $Lx = b$ (CSR, kernel only, complex).
- int mess_solver_lusolvem_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *lval, mess_int_t *lcolptr, mess_int_t *lrowptr, mess_double_cpx_t *uval, mess_int_t *ucolptr, mess_int_t *urowptr, mess_int_t *p, mess_int_t *q, mess_matrix b, mess_matrix x)

  LU solver for complex CSR stored L and U.
- int mess_solver_ltsolve (mess_matrix L, mess_vector y)

  Solve $L^T x = b$ (user interface).
- int mess_solver_ltsolvem (mess_matrix L, mess_matrix Y)

Solve $L^T X = B$ (user interface).

- int mess_solver_lhsolve (mess_matrix L, mess_vector y)

  Solve $L^H x = b$ (user interface).
- int mess_solver_lhsolvem (mess_matrix L, mess_matrix Y)

  Solve $L^H X = B$ (user interface).
- int mess_solver_ltsolve_kernelcsr_real (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, double *y)

  Solve $L^T x = b$ (CSR, kernel only, real).
- int mess_solver_ltsolve_kernelcsr_real_complex (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $L^T x = b$ (CSR, kernel only, real L, complex y).
- int mess_solver_utsolve_kernelcsr_real (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, double *x)

  Solve $U^T x = b$ (CSR, kernel only, real).
- int mess_solver_utsolve_kernelcsr_real_complex (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *x)

  Solve $U^T x = b$ (CSR, kernel only, real U, complex y).
- int mess_solver_ltsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $L^T x = b$ (CSR, kernel only, complex).
- int mess_solver_lhsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $L^H x = b$ (CSR, kernel only, complex).
- int mess_solver_utsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *x)

  Solve $U^T x = b$ (CSR, kernel only, complex).
- int mess_solver_uhsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *x)

  Solve $U^H x = b$ (CSR, kernel only, complex).
- int mess_solver_utsolve (mess_matrix U, mess_vector y)

  Solve $U^T x = b$ (user interface).
- int mess_solver_uhsolve (mess_matrix U, mess_vector y)

  Solve $U^H x = b$ (user interface).
- int mess_solver_utsolvem (mess_matrix U, mess_matrix y)

  Solve $U^T X = B$ (user interface).
- int mess_solver_uhsolvem (mess_matrix U, mess_matrix y)

  Solve $U^H X = B$ (user interface).

8.18.1 Detailed Description

The solution of triangular linear systems is a basic building block for the solution of generic linear systems. This section provides different solver for different shapes and data types of triangular systems.

8.18.2 Function Documentation

8.18.2.1 int mess_solver_lhsolve (mess_matrix *L*, mess_vector *y*)

Parameters

in	*L*	input lower triangular matrix
in,out	*y*	on input: right hand side b
		on output: solution x

Returns

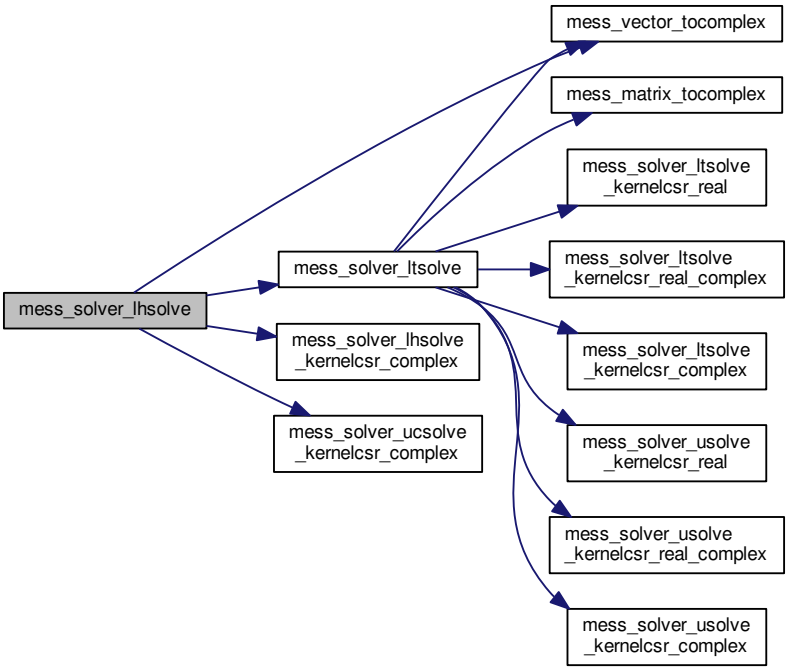
zero on success or a non zero error code otherwise

The mess_solver_lhsolve function solves the system

$$L^H x = b.$$

It is the user interface for mess_solver_ltsolve_kernelcsr_real and mess_solver_lhsolve_kernelcsr_complex.

Here is the call graph for this function:



8.18.2.2 int mess_solver_lhsolve_kernelcsr_complex (mess_int_t *dim*, mess_double_cpx_t * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_double_cpx_t * *y*)

Parameters

in	*dim*	input row dimension of L
in	*values*	input values of L
in	*colptr*	input column pointer for L
in	*rowptr*	input row pointer for L
in,out	*y*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_lhsolve_kernelcsr_complex function solves the system

$$L^H x = b$$

where L is a complex lower triangular matrix stored in Compressed Row Storage. The transposition is done implicitly.

Attention

Internal use only.

8.18.2.3 int mess_solver_lhsolvem (mess_matrix *L*, mess_matrix *Y*)

Parameters

in	*L*	input lower triangular matrix
in,out	*Y*	on input: right hand side matrix B on output: solution matrix X

Returns

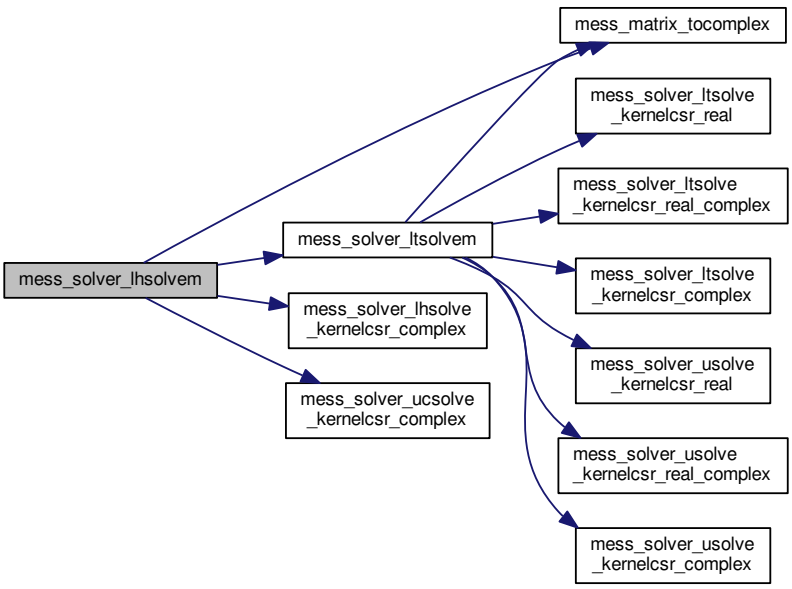
zero on success, a non zero error code otherwise

The mess_solver_lhsolvem function solves the system

$$L^H X = B.$$

It is the user interface for mess_solver_ltsolve_kernelcsr_real and mess_solver_lhsolve_kernelcsr_complex.

Here is the call graph for this function:



8.18.2.4 int mess_solver_lsolve (mess_matrix *L*, mess_vector *y*)

Parameters

in	*L*	input lower triangular matrix
in,out	*y*	on input: right hand side b on output: solution x

Returns

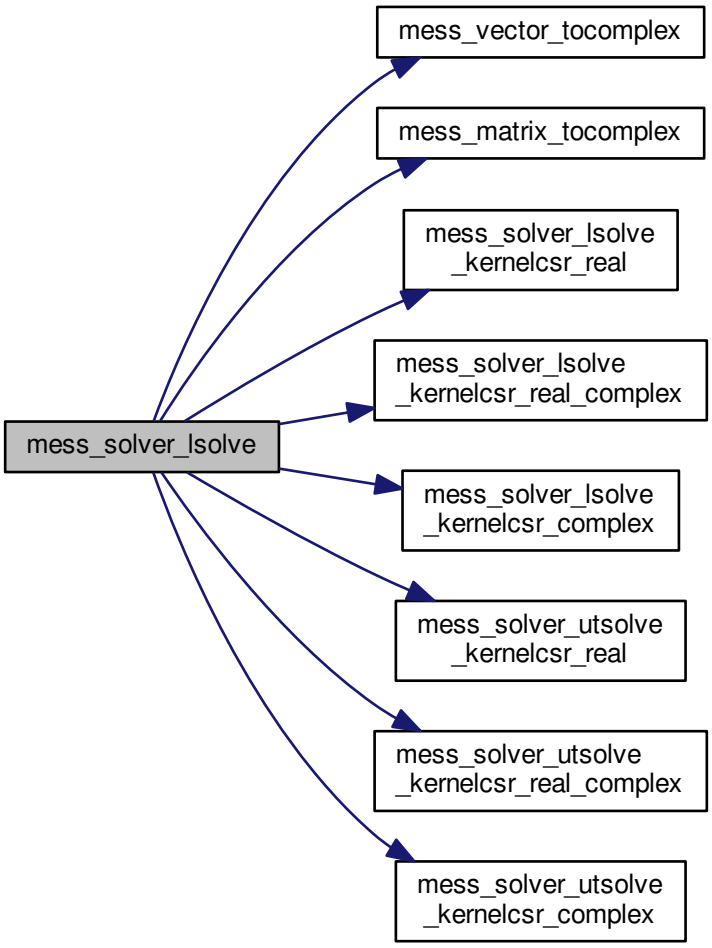
zero on success or a non zero error code otherwise

The mess_solver_lsolve function solves the system

$$Lx = b.$$

It is the user interface for mess_solver_lsolve_kernelcsr_real and mess_solver_lsolve_kernelcsr_complex.

Here is the call graph for this function:



8.18.2.5 int mess_solver_lsolve_kernelcsr_complex (mess_int_t *dim*, mess_double_cpx_t * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_double_cpx_t * *y*)

Parameters

in	*dim*	input row dimension of L
in	*values*	input values of L
in	*colptr*	input column pointer for L
in	*rowptr*	input row pointer for L
in,out	*y*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_lsolve_kernelcsr_complex function solves the system

$$Lx = b$$

where L is a complex lower triangular matrix stored in Compressed Row Storage.

Attention

Internal use only.

8.18.2.6 int mess_solver_lsolve_kernelcsr_real (mess_int_t *dim*, double * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, double * *y*)

Parameters

in	*dim*	input row dimension of L
in	*values*	input values of L
in	*colptr*	input column pointer for L
in	*rowptr*	input row pointer for L
in,out	*y*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_lsolve_kernelcsr_real function solves the system

$$Lx = b$$

where L is a real lower triangular matrix stored in Compressed Row Storage.

Attention

Internal use only.

8.18.2.7 int mess_solver_lsolve_kernelcsr_real_complex (mess_int_t *dim*, double * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_double_cpx_t * *y*)

Parameters

in	*dim*	input row dimension of L
in	*values*	input values of L
in	*colptr*	input column pointer for L
in	*rowptr*	input row pointer for L
in,out	*y*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_lsolve_kernelcsr_real_complex function solves the system

$$Lx = b$$

where L is a real lower triangular matrix stored in Compressed Row Storage.

Attention

Internal use only.

8.18.2.8 int mess_solver_lsolvem (mess_matrix *L*, mess_matrix *Y*)

Parameters

in	*L*	input lower triangular matrix
in,out	*Y*	on input: right hand side matrix B on output: solution matrix X

Returns

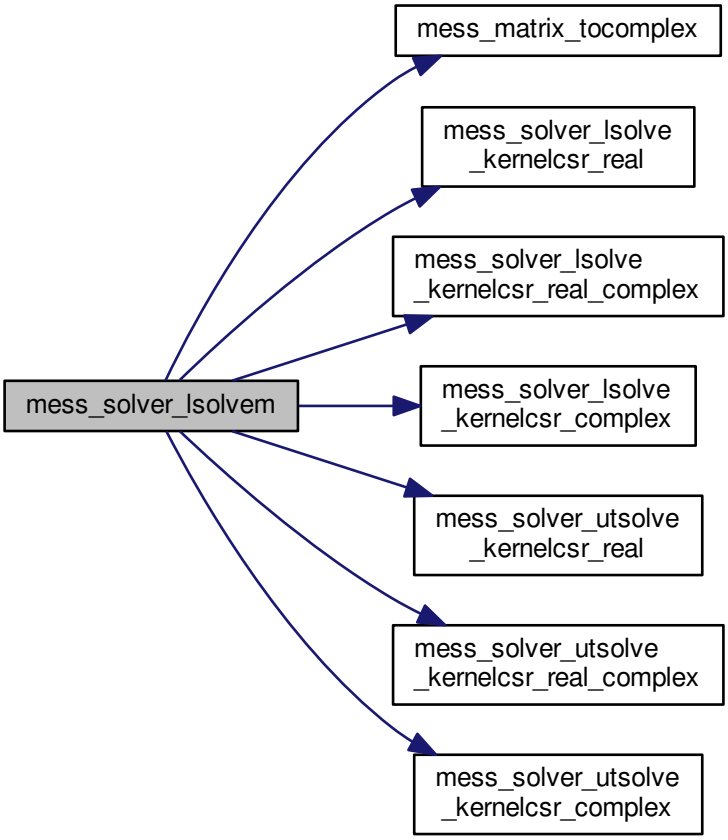
zero on success or a non zero error code otherwise

The mess_solver_lsolvem function solves the system

$$LX = B.$$

It is the user interface for mess_solver_lsolve_kernelcsr_real and mess_solver_lsolve_kernelcsr_complex.

Here is the call graph for this function:



8.18.2.9 int mess_solver_ltsolve (mess_matrix *L*, mess_vector *y*)

Parameters

in	*L*	input lower triangular matrix
in,out	*y*	on input: right hand side b on output: solution x

Returns

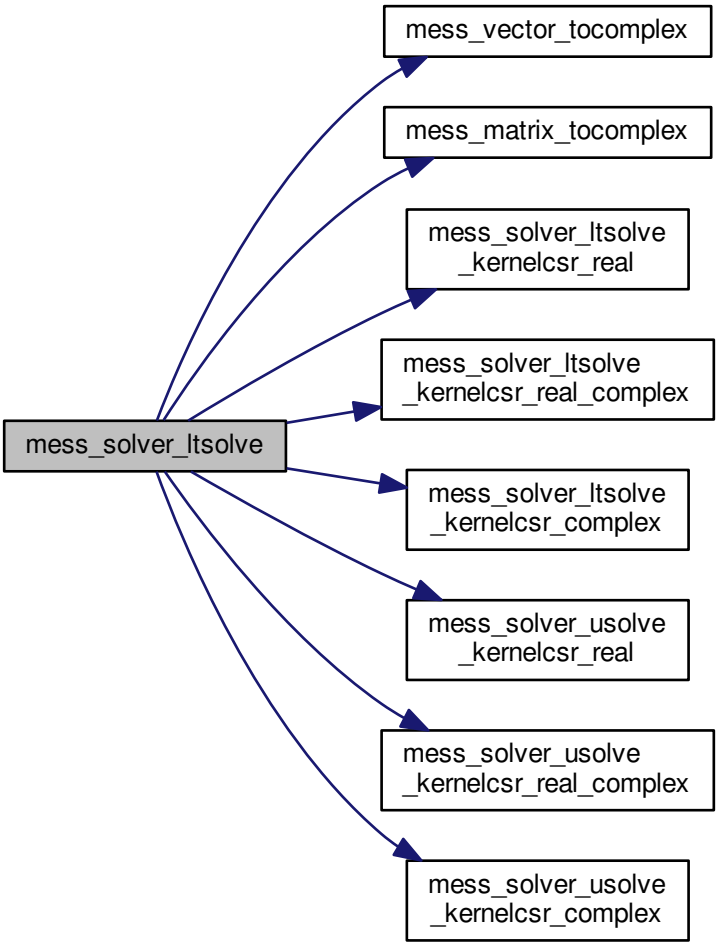
zero on success or a non zero error code otherwise

The mess_solver_ltsolve function solves the system

$$L^T x = b.$$

It is the user interface for mess_solver_ltsolve_kernelcsr_real and mess_solver_ltsolve_kernelcsr_complex.

Here is the call graph for this function:



8.18.2.10 int mess_solver_ltsolve_kernelcsr_complex (mess_int_t *dim*, mess_double_cpx_t * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_double_cpx_t * *y*)

Parameters

in	*dim*	input row dimension of L
in	*values*	input values of L
in	*colptr*	input column pointer for L
in	*rowptr*	input row pointer for L
in,out	*y*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_ltsolve_kernelcsr_complex function solves the system

$$L^T x = b$$

where L is a complex lower triangular matrix stored in Compressed Row Storage. The transposition is done implicitly.

Attention

Internal use only.

8.18.2.11 int mess_solver_ltsolve_kernelcsr_real (mess_int_t *dim*, double * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, double * *y*)

Parameters

in	*dim*	input row dimension of L
in	*values*	input values of L
in	*colptr*	input column pointer for L
in	*rowptr*	input row pointer for L
in,out	*y*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_ltsolve_kernelcsr_real function solves the system

$$L^T x = b$$

where L is a real lower triangular matrix stored in Compressed Row Storage. The transposition is done implicitly.

Attention

Internal use only.

8.18.2.12 int mess_solver_ltsolve_kernelcsr_real_complex (mess_int_t *dim*, double * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_double_cpx_t * *y*)

Parameters

in	*dim*	input row dimension of L
in	*values*	input values of L
in	*colptr*	input column pointer for L
in	*rowptr*	input row pointer for L
in,out	*y*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_ltsolve_kernelcsr_real_complex function solves the system

$$L^T x = b$$

where L is a real lower triangular matrix stored in Compressed Row Storage. The transposition is done implicitly.

Attention

Internal use only.

8.18.2.13 int mess_solver_ltsolvem (mess_matrix *L*, mess_matrix *Y*)

Parameters

in	*L*	input lower triangular matrix
in,out	*Y*	on input: right hand side matrix B on output: solution matrix X

Returns

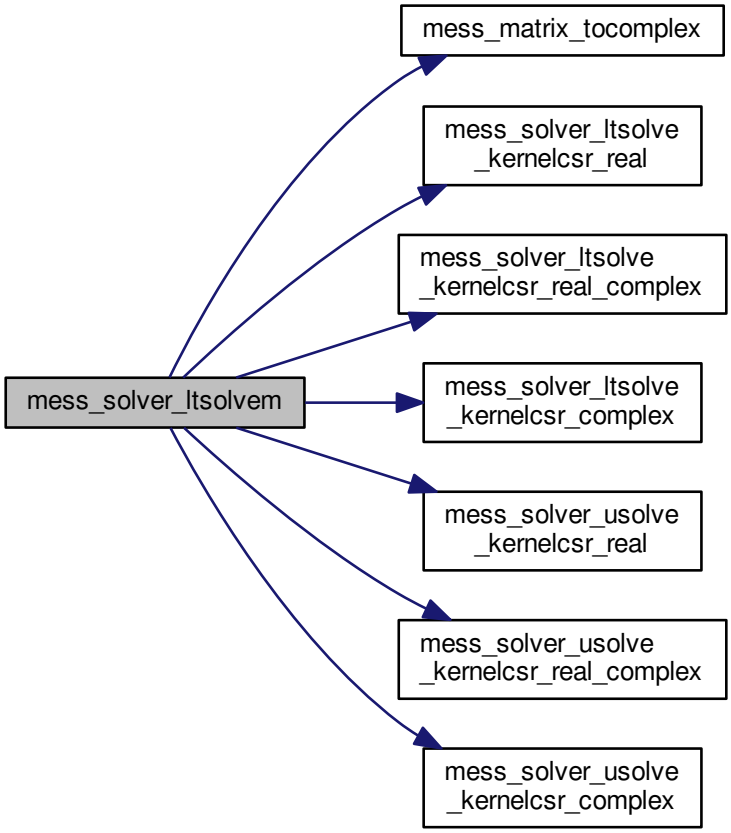
zero on success, a non zero error code otherwise

The mess_solver_ltsolvem function solves the system

$$L^T X = B.$$

It is the user interface for mess_solver_ltsolve_kernelcsr_real and mess_solver_ltsolve_kernelcsr_complex.

Here is the call graph for this function:



8.18.2.14 int mess_solver_lusolvem_kernelcsr_complex (mess_int_t *dim*, mess_double_cpx_t * *lval*, mess_int_t * *lcolptr*, mess_int_t * *lrowptr*, mess_double_cpx_t * *uval*, mess_int_t * *ucolptr*, mess_int_t * *urowptr*, mess_int_t * *p*, mess_int_t * *q*, mess_matrix *b*, mess_matrix *x*)

Parameters

in	*dim*	input dimension
in	*lval*	input values of L
in	*lcolptr*	input column pointer for L
in	*lrowptr*	input row pointer for L
in	*uval*	input values of U
in	*ucolptr*	input column pointer for U
in	*urowptr*	input row pointer for U
in	*p*	input row permuation
in	*q*	input column permutaion
in	*b*	input right hand side
out	*x*	solution

Returns

zero on success or a non zero error code otherwise

The mess_solver_lusolvem_kernelcsr_complex function solves

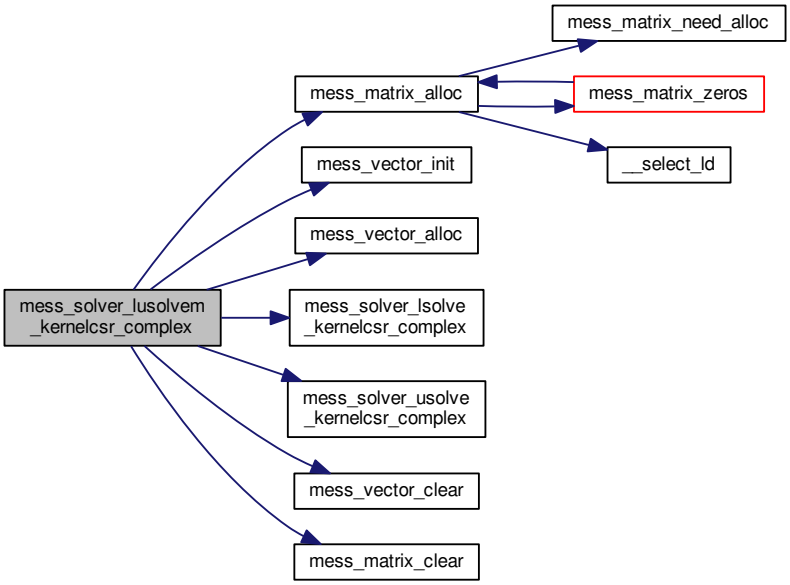
$$LUx = b$$

for B and X matrices. L and U are given as vectors.

Attention

Internal use only.

Here is the call graph for this function:



8.18.2.15 int mess_solver_uhsolve (mess_matrix *U*, mess_vector *y*)

Parameters

in	*U*	input upper triangular matrix
in,out	*y*	on input: right hand side b on output: solution x

Returns

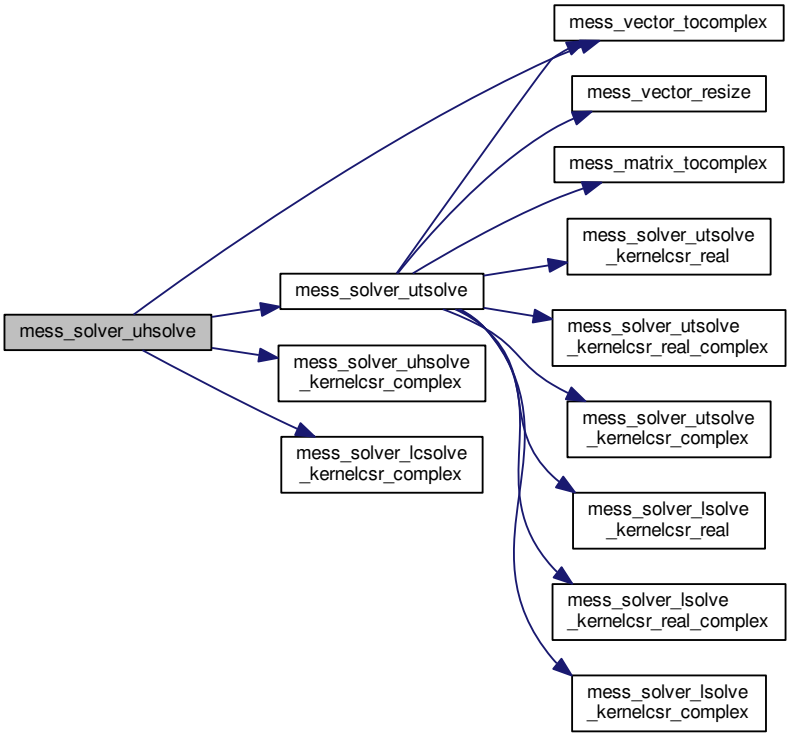
zero on success or a non-zero error value otherwise

The mess_solver_uhsolve function solves the system

$$U^H x = b$$

It is the user interface for mess_solver_uhsolve_kernelcsr_complex and mess_solver_utsolve_kernelcsr_real.

Here is the call graph for this function:



8.18.2.16 int mess_solver_uhsolve_kernelcsr_complex (mess_int_t *dim*, mess_double_cpx_t * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_double_cpx_t * *x*)

Parameters

in	*dim*	input row dimension of U
in	*values*	input values of U
in	*colptr*	input column pointer for U
in	*rowptr*	input row pointer for U
in,out	*x*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_uhsolve_kernelcsr_complex function solves the system

$$U^H x = b$$

where U is a complex upper triangular matrix stored in Compressed Row Storage. The transposition is done implicitly.

Attention

Internal use only.

8.18.2.17 int mess_solver_uhsolvem (mess_matrix *U*, mess_matrix *Y*)

Parameters

in	*U*	input upper triangular matrix
in,out	*Y*	on input: right hand side matrix B on output: solution matrix X

Returns

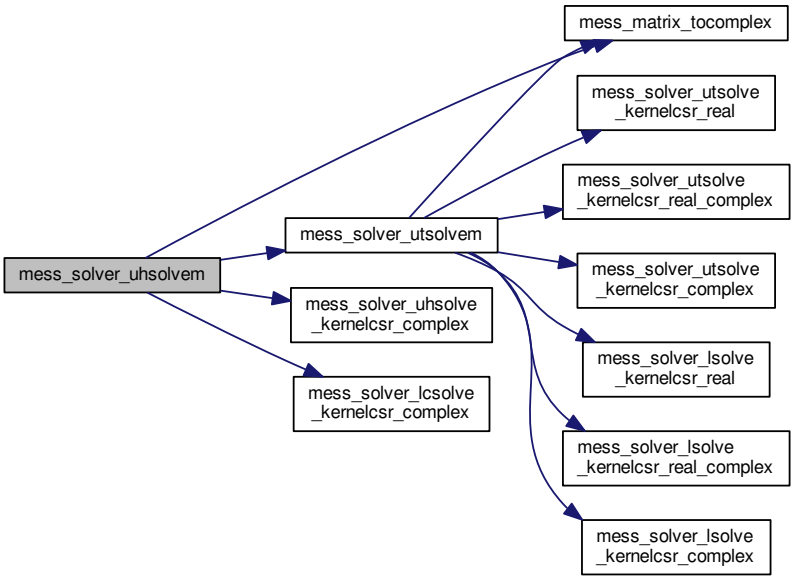
zero on success or a non zero error code otherwise

The mess_solver_uhsolvem function solves the system

$$U^H X = B.$$

It is the user interface for mess_solver_utsolve_kernelcsr_real and mess_solver_uhsolve_kernelcsr_complex.

Here is the call graph for this function:



8.18.2.18 int mess_solver_usolve (mess_matrix *U*, mess_vector *y*)

Parameters

in	*U*	input upper triangular matrix
in,out	*y*	on input: right hand side b on output: solution x

Returns

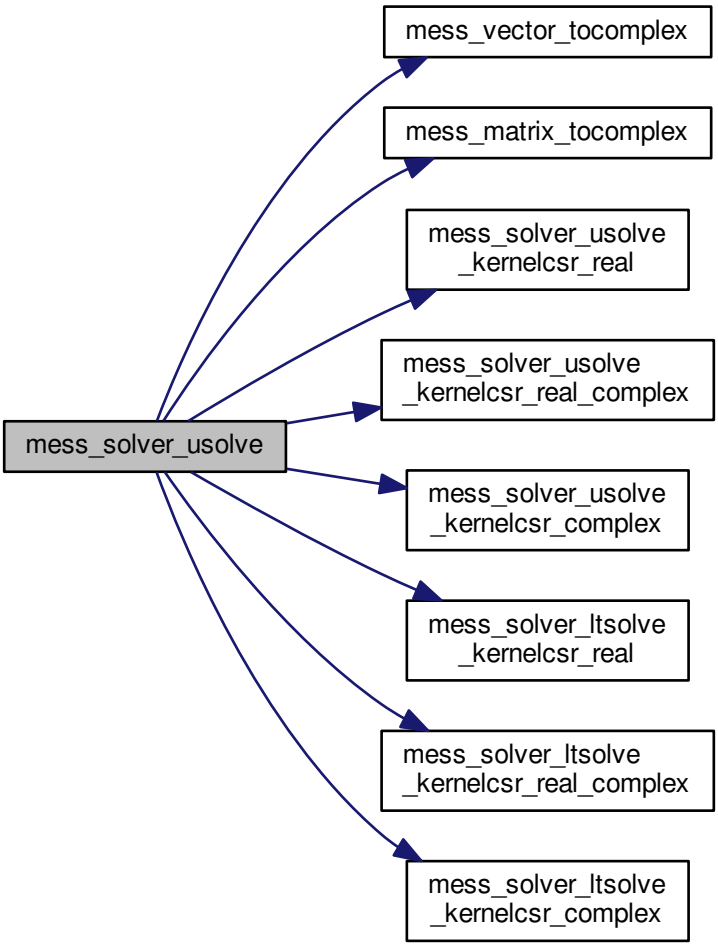
zero on success or a non zero error code otherwise

The mess_solver_usolve function solves the system

$$Ux = b.$$

It is the user interface for mess_solver_usolve_kernelcsr_real and mess_solver_usolve_kernelcsr_complex.

Here is the call graph for this function:



8.18.2.19 int mess_solver_usolve_kernelcsr_complex (mess_int_t *dim*, mess_double_cpx_t * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_double_cpx_t * *y*)

Parameters

in	*dim*	input row dimension of U
in	*values*	input values of U

Parameters

in	*colptr*	input column pointer for U
in	*rowptr*	input row pointer for U
in,out	*y*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_usolve_kernelcsr_complex function solves the system

$$Ux = b$$

where U is a real lower triangular matrix stored in Compressed Row Storage.

Attention

Internal use only.

8.18.2.20 int mess_solver_usolve_kernelcsr_real (mess_int_t *dim*, double * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, double * *y*)

Parameters

in	*dim*	input row dimension of U
in	*values*	input values of U
in	*colptr*	input column pointer for U
in	*rowptr*	input row pointer for U
in,out	*y*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_usolve_kernelcsr_real function solves the system

$$Ux = b$$

where U is a real upper triangular matrix stored in Compressed Row Storage.

Attention

Internal use only.

8.18.2.21 int mess_solver_usolve_kernelcsr_real_complex (mess_int_t *dim*, double * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_double_cpx_t * *y*)

Parameters

in	*dim*	input row dimension of U
in	*values*	input values of U
in	*colptr*	input column pointer for U
in	*rowptr*	input row pointer for U
in,out	*y*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_usolve_kernelcsr_real_complex function solves the system

$$Ux = b$$

where U is a real upper triangular matrix stored in Compressed Row Storage.

Attention

Internal use only.

8.18.2.22 int mess_solver_usolvem (mess_matrix *U*, mess_matrix *Y*)

Parameters

in	*U*	input upper triangular matrix
in,out	*Y*	on input: right hand side matrix B on output: solution matrix X

Returns

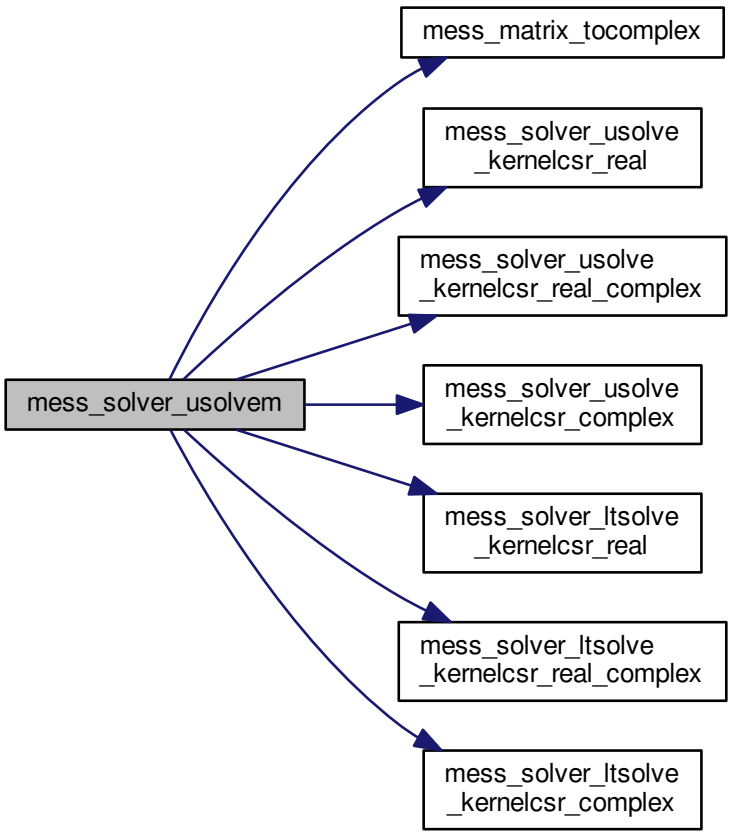
zero on success or a non zero error code otherwise

The mess_solver_usolvem function solves the system

$$UX = B.$$

It is the user interface for mess_solver_usolve_kernelcsr_real and mess_solver_usolve_kernelcsr_complex.

Here is the call graph for this function:



8.18.2.23 int mess_solver_utsolve (mess_matrix *U*, mess_vector *y*)

Parameters

in	*U*	input upper triangular matrix
in,out	*y*	on input: right hand side b on output: solution x

Returns

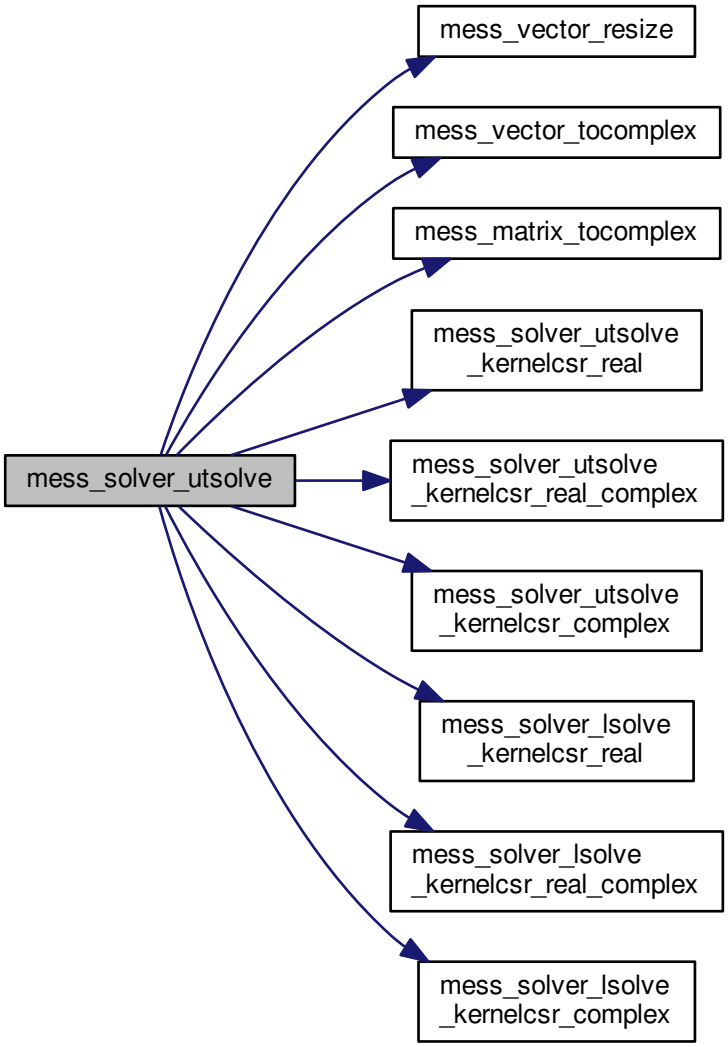
zero on success or a non-zero error value otherwise

The mess_solver_utsolve function solves the system

$$U^T x = b.$$

It is the user interface for mess_solver_utsolve_kernelcsr_real and mess_solver_utsolve_kernelcsr_complex.

Here is the call graph for this function:



8.18.2.24 int mess_solver_utsolve_kernelcsr_complex (mess_int_t *dim*, mess_double_cpx_t * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_double_cpx_t * *x*)

Parameters

in	*dim*	input row dimension of U
in	*values*	input values of U
in	*colptr*	input column pointer for U
in	*rowptr*	input row pointer for U
in,out	*x*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_utsolve_kernelcsr_complex function solves the system

$$U^T x = b$$

where U is a complex upper triangular matrix stored in Compressed Row Storage. The transposition is done implicitly.

Attention

Internal use only.

8.18.2.25 int mess_solver_utsolve_kernelcsr_real (mess_int_t *dim*, double * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, double * *x*)

Parameters

in	*dim*	input row dimension of U
in	*values*	input values of U
in	*colptr*	input column pointer for U
in	*rowptr*	input row pointer for U
in,out	*x*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_utsolve_kernelcsr_real function solves the system

$$U^T x = b$$

where U is a real upper triangular matrix stored in Compressed Row Storage. The transposition is done implicitly.

Attention

Internal use only.

8.18.2.26 int mess_solver_utsolve_kernelcsr_real_complex (mess_int_t *dim*, double * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_double_cpx_t * *x*)

Parameters

in	*dim*	input row dimension of U
in	*values*	input values of U
in	*colptr*	input column pointer for U
in	*rowptr*	input row pointer for U
in,out	*x*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_utsolve_kernelcsr_real_complex function solves the system

$$U^T x = b$$

where U is a real upper triangular matrix stored in Compressed Row Storage. The transposition is done implicitly.

Attention

Internal use only.

8.18.2.27 int mess_solver_utsolvem (mess_matrix *U*, mess_matrix *Y*)

Parameters

in	*U*	input upper triangular matrix
in,out	*Y*	on input: right hand side matrix B on output: solution matrix X

Returns

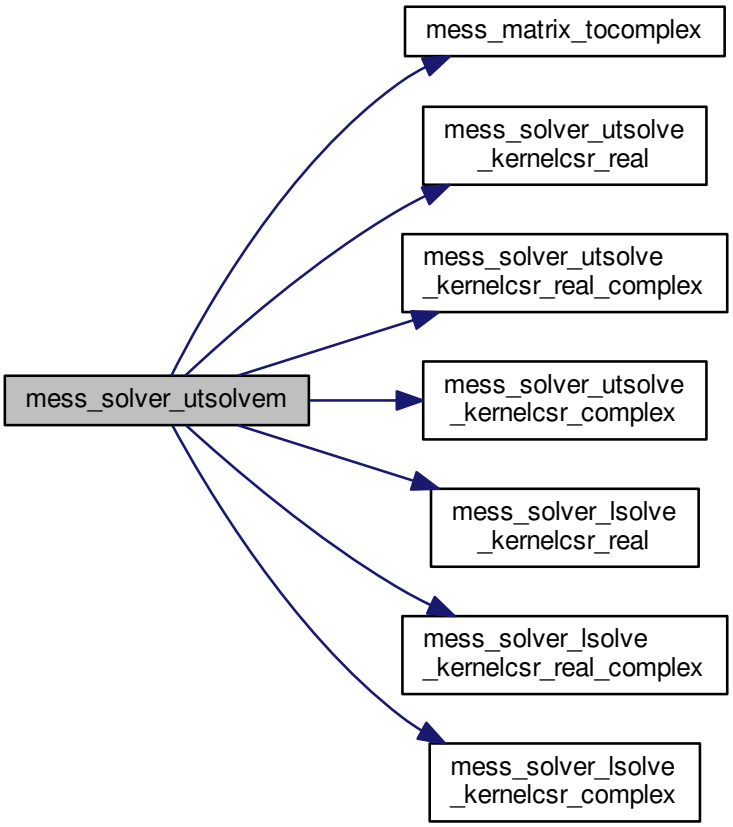
zero on success or a non zero error code otherwise

The mess_solver_utsolvem function solves the system

$$U^T X = B$$

It is the user interface for mess_solver_utsolve_kernelcsr_real and mess_solver_utsolve_kernelcsr_complex.

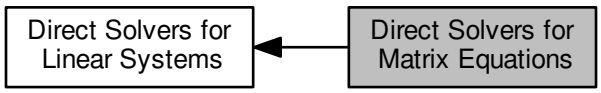
Here is the call graph for this function:



8.19 Direct Solvers for Matrix Equations

This group contains direct solver for matrix Equations like the Lyapunov Equation.

Collaboration diagram for Direct Solvers for Matrix Equations:



Functions

- int mess_direct_create_generalized_lyapunovchol (mess_matrix A, mess_matrix E, mess_direct solver)

  Generate a dense standard / generalized Lyapunov equation solver for the cholesky factor.
- int mess_direct_create_generalized_lyapunov (mess_matrix A, mess_matrix E, mess_direct solver)

  Generate a dense standard / generalized Lyapunov equation solver.
- int mess_direct_create_generalized_stein (mess_matrix A, mess_matrix E, mess_direct solver)

  Generate a dense standard / generalized Stein equation solver.
- int mess_direct_care (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X)

  Compute the solution of a (generalized) Riccati Equation using a dense newton method.
- int mess_direct_care_sign (mess_matrix A, mess_matrix Q, mess_matrix G, mess_matrix X)
- int mess_direct_create_sylvester_sparsedense (mess_matrix A, mess_matrix F, mess_matrix E, mess_matrix H, mess_direct solver)

  Generate a solver for a sparse-dense Sylvester equation with a small second matrix.
- int mess_direct_create_sylvester_dense (mess_matrix A, mess_matrix F, mess_matrix E, mess_matrix H, mess_direct solver)

  Generate a solver for a dense Sylvester equation using `LAPACK`.

8.19.1 Detailed Description

The solution of matrix equations is the main goal of the **M.E.S.S.** library.
In contrast to the normal linear solvers these solvers are mostly restricted to matrix-valued right hand sides. For this case the function mess_direct_solve will cause an error. Please use mess_direct_solvem instead.
Some of them like the solve for the Riccati Equation will not setup a mess_direct object because they cannot reuse their data and compute the solution directly.
Additionally to these functions are enumerations to select a preconditioner.

8.19.2 Function Documentation

8.19.2.1 int mess_direct_care (mess_matrix *A*, mess_matrix *E*, mess_matrix *Q*, mess_matrix *G*, mess_matrix *X*)

Parameters

in	*A*	input A matrix of the Riccati Equation
in	*E*	input E matrix of the Riccati Equation (NULL if E does not exist)
in	*Q*	input Q matrix of the Riccati Equation
in	*G*	input G matrix of the Riccati Equation
out	*X*	output X solution of the Riccati Equation

Returns

zero on success or a non-zero error value otherwise

The mess_direct_care function solves either

$$A^T X + XA - XGX + Q = 0$$

or

$$A^T XE + E^T XA - E^T XGXE + Q = 0$$

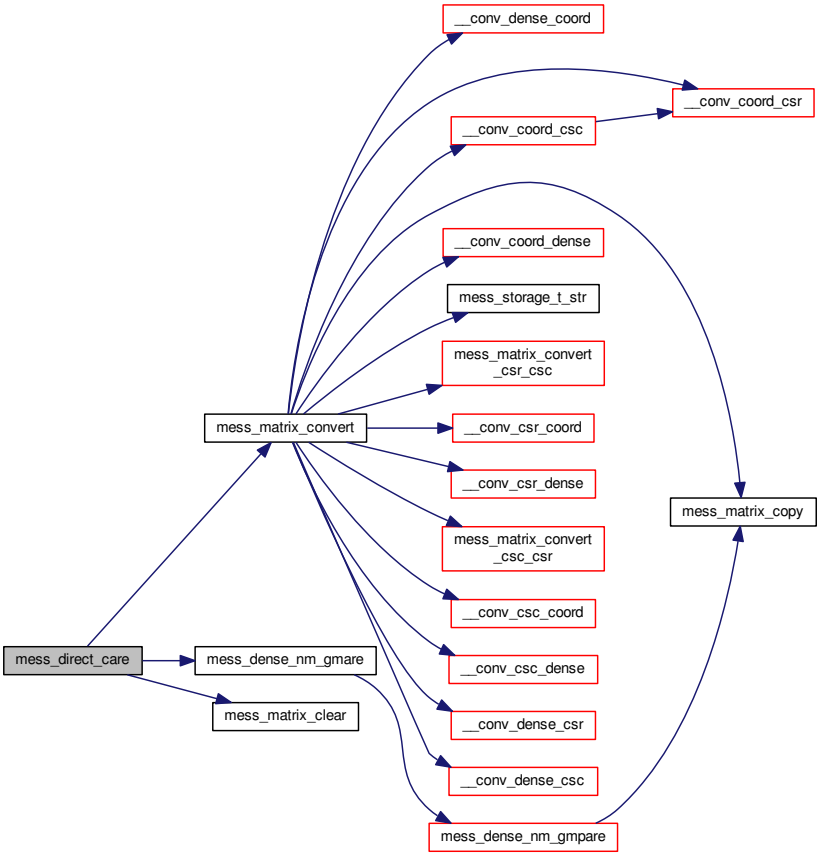
depending on the existence of the matrix E.
If the matrix object for E is set to `NULL` the standard Riccati Equation is solved, otherwise the generalized one is solved.

The dense Newton method from mess_dense_nm_gmare is used.

Attention

- If the inputs are sparse matrices, they are converted to dense ones. This might cause an out of memory.

Here is the call graph for this function:



8.19.2.2 int mess_direct_care_sign (mess_matrix *A*, mess_matrix *Q*, mess_matrix *G*, mess_matrix *X*)

8.19.2.3 int mess_direct_create_generalized_lyapunov (mess_matrix *A*, mess_matrix *E*, mess_direct *solver*)

Parameters

in	*A*	input real matrix A
in	*E*	input real matrix E, (NULL if not wanted)
out	*solver*	output solver

Returns

zero on success or a non-zero error value otherwise

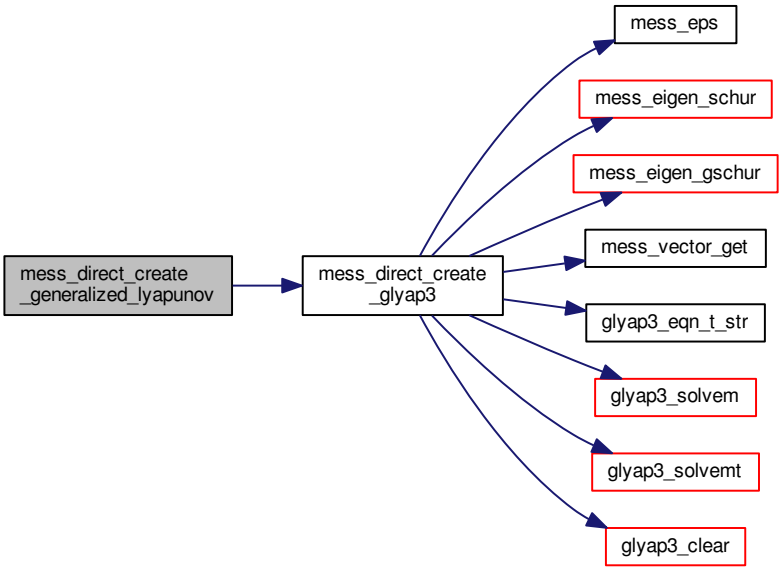
The mess_direct_create_generalized_lyapunov function generates a solver for dense standard / generalized Lyapunov equations based on the GLYAP3 solver.

The created solver can solve the generalized Lyapunov equations: $AXE^T + EXA^T + Y = 0$ and $A^TXE + E^TXA + Y = 0$. If E points to NULL the created solver can solve standard Lyapunov equations: $AX + XA^T + Y = 0$ and $A^TX + XA + Y = 0$.

Attention

In both cases the right hand side Y must be real and symmetric.

Here is the call graph for this function:



8.19.2.4 int mess_direct_create_generalized_lyapunovchol (mess_matrix *A*, mess_matrix *E*, mess_direct *solver*)

Parameters

in	*A*	input real matrix A
in	*E*	input real matrix E, (NULL if not wanted)
out	*solver*	output solver

Returns

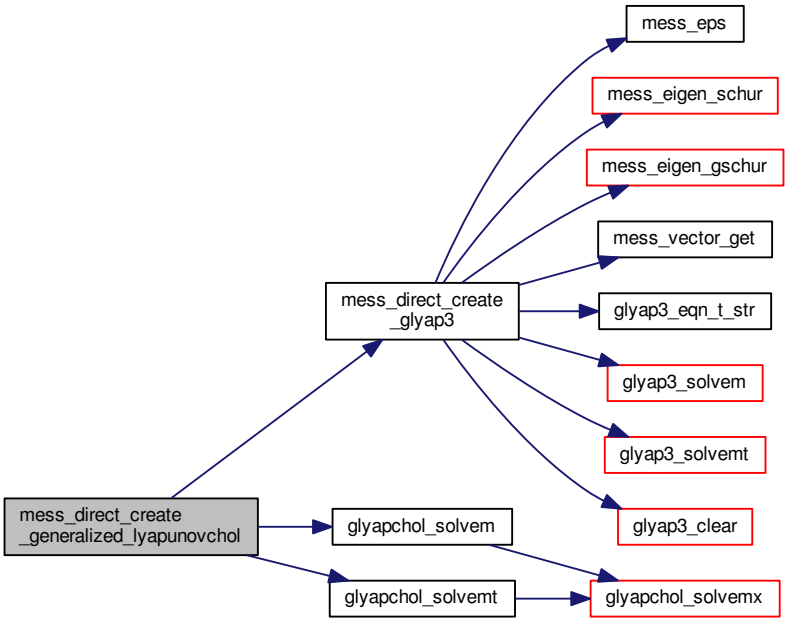
zero on success or a non-zero error value otherwise

The mess_direct_create_generalized_lyapunovchol function generates a solver for dense standard / generalized Lyapunov equations based on the GLYAP3 solver.
The created solver can solve the generalized Lyapunov equations: $AXE^T + EXA^T + Y = 0$ and $A^TXE + E^TXA + BB^T = 0$. If E points to NULL the created solver can solve standard Lyapunov equations: $AX + XA^T + Y = 0$ and $A^TX + XA + BB^T = 0$. The solution is always returned as a cholesky factorization $X \approx ZZ^T$.

Attention

In both cases the right hand side Y must be real and symmetric.

Here is the call graph for this function:



8.19.2.5 int mess_direct_create_generalized_stein (mess_matrix *A*, mess_matrix *E*, mess_direct *solver*)

Parameters

in	*A*	input real matrix A
in	*E*	input real matrix E, (NULL if not wanted)
out	*solver*	output solver

Returns

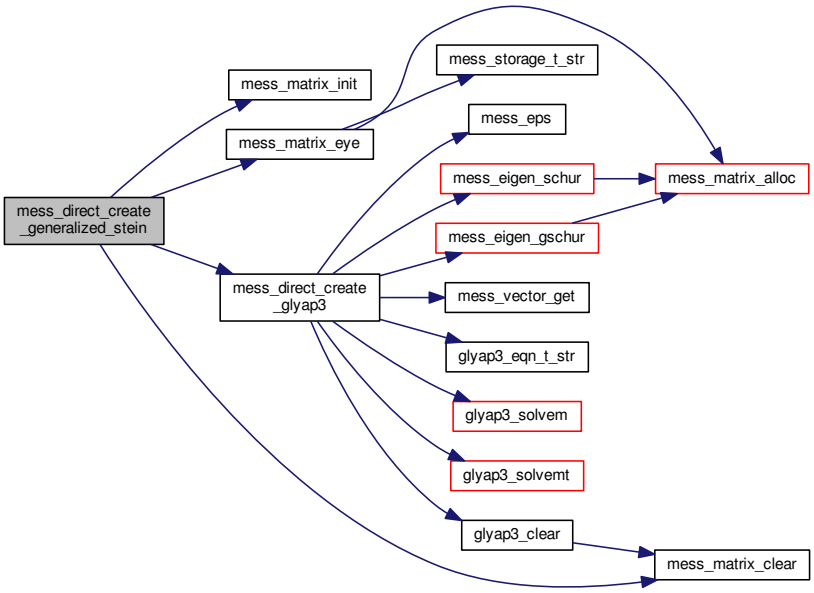
zero on success or a non-zero error value otherwise

The mess_direct_create_generalized_stein function generates a solver for dense standard / generalized Stein equations based on the GLYAP3 solver.
The created solver can solve the generalized Stein equations: $AXA^T - EXE^T + Y = 0$ and $A^TXA - E^TXE + Y = 0$. If E points to NULL the created solver can solve standard Stein equations: $AXA^T - X + Y = 0$ and $A^TXA - X + Y = 0$.

Note

GLYAP3 currently does not support standard Stein equations, therefore mess_direct_create_generalized_stein uses the generalized Stein solver GLYAP3 with $E = I_n$.

Here is the call graph for this function:



8.19.2.6 int mess_direct_create_sylvester_dense (mess_matrix *A*, mess_matrix *F*, mess_matrix *E*, mess_matrix *H*, mess_direct *solver*)

Parameters

in	*A*	input matrix
in	*F*	input matrix
in	*E*	input matrix
in	*H*	input matrix
out	*solver*	solver to create

Returns

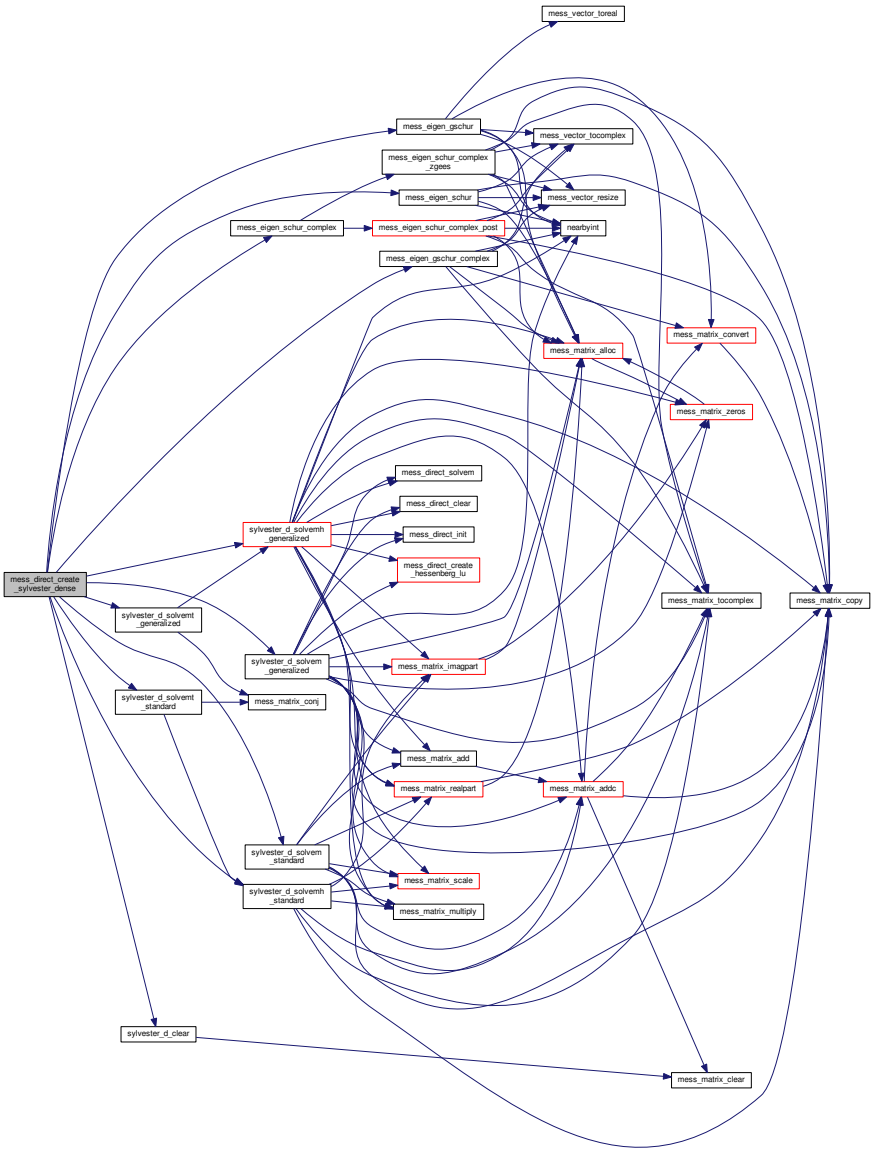
zero on success or a non-zero error value otherwise

The mess_direct_create_sylvester_dense function generates a Sylvester equation solver using LAPACK. It solves

- Standard: $AX + XH + M = 0$, $A^T X + XH^T + M = 0$, $A^H X + XH^H + M = 0$
- Generalized Sylvester: $AXF + EXH + M = 0$, $A^T XF^T + E^T XH^T + M = 0$, $A^H XF^H + E^H XH^H + M = 0$

with A, F, E, H small and dense matrices. If a standard Sylvester equation solver is wanted use NULL and NULL for the matrices E and F. F must be a regular matrix.

Here is the call graph for this function:



8.19.2.7 int mess_direct_create_sylvester_sparsedense (mess_matrix *A*, mess_matrix *F*, mess_matrix *E*, mess_matrix *H*, mess_direct *solver*)

Parameters

in	*A*	input first matrix
in	*F*	input generalized second matrix
in	*E*	input generalized matrix for the right term
in	*H*	input second matrix
out	*solver*	solver to create

Returns

zero on success or a non-zero error value otherwise

The mess_direct_create_sylvester_sparsedense function creates a solver for

- Standard: $AX + XH + M = 0$, $A^T X + XH^T + M = 0$, $A^H X + XH^H + M = 0$
- Half Generalized Sylvester: $AX + EXH + M = 0$, $A^T X + E^T XH^T + M = 0$, $A^H X + E^H XH^H + M = 0$
- Generalized Sylvester: $AXF + EXH + M = 0$, $A^T XF^T + E^T XH^T + M = 0$, $A^H XF^H + E^H XH^H + M = 0$

with A and E sparse and large and F and H are small and dense.

If it gets `NULL` pointers for F and E, it creates the solver for $AX + XH + M = 0$.

If only F is a `NULL` pointer, it creates the solver for $AX + EXH + M = 0$.

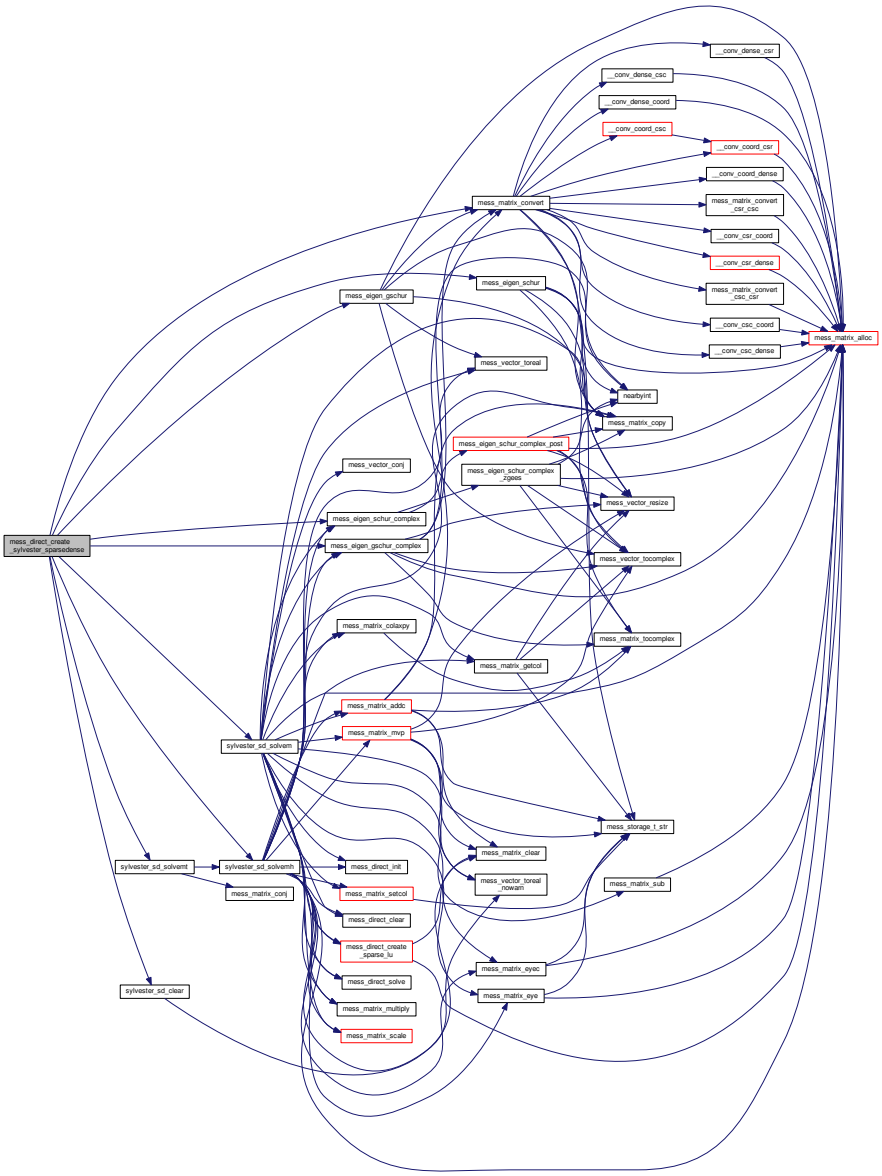
If none of the arguments are `NULL`, it creates the solver for $AXF + EXH + M = 0$.

See [14] and [8] for references.

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Robert Dykstra (dykstra@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

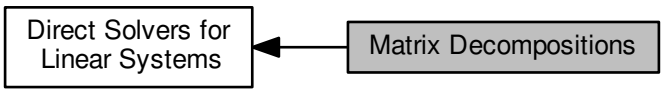
Here is the call graph for this function:



8.20 Matrix Decompositions

This group contains matrix decompositions which are strongly related to direct solvers.

Collaboration diagram for Matrix Decompositions:



Functions

- int mess_direct_dgeqp3 (mess_matrix A, mess_matrix Q, mess_matrix R, mess_int_t *perm)

  Interface for `LAPACK` *DGEQP3.*
- int mess_direct_cholfactor (mess_matrix ZZ, mess_matrix Z)

  Compute a Cholesky factor of a given matrix.

8.20.1 Detailed Description

The matrix decompositions mentioned in this section are strongly related to direct solvers but are used mostly outside a solver context. Currently this affects only the Column Pivoted QR decomposition.

The decompositions in this section does not rely on the Generic Interface to Direct Solvers because they are not used in the direct solver context.
This are for example the column pivoted QR decomposition or computing the Cholesky factor of a may be singluar matrix.

8.20.2 Function Documentation

8.20.2.1 int mess_direct_cholfactor (mess_matrix *ZZ*, mess_matrix *Z*)

Parameters

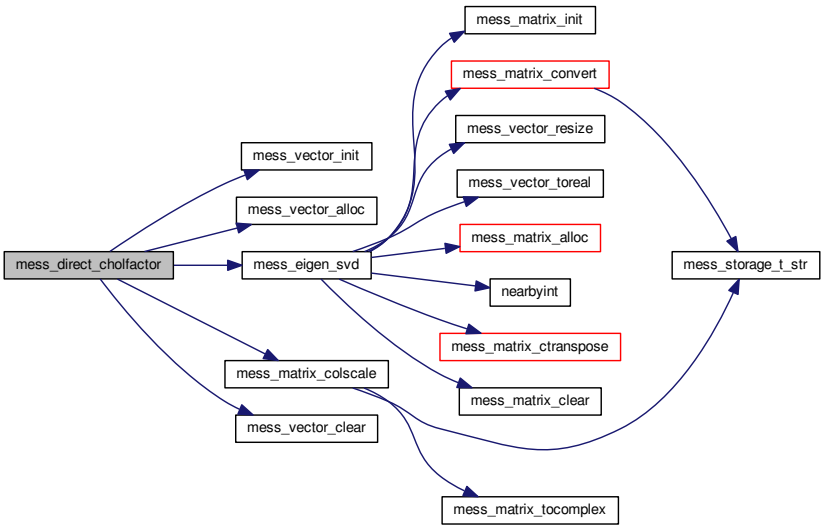
in	*ZZ*	input symmetric matrix
out	*Z*	output Cholesky factor such that $Z_{out}Z_{out}^T = ZZ$

Returns

zero on success or a non-zero error value otherwise

The mess_direct_cholfactor function computes a Cholesky factor of the matrix Z using the SVD.
It works for singular matrices, too.

Here is the call graph for this function:



8.20.2.2 int mess_direct_dgeqp3 (mess_matrix *A*, mess_matrix *Q*, mess_matrix *R*, mess_int_t * *perm*)

Parameters

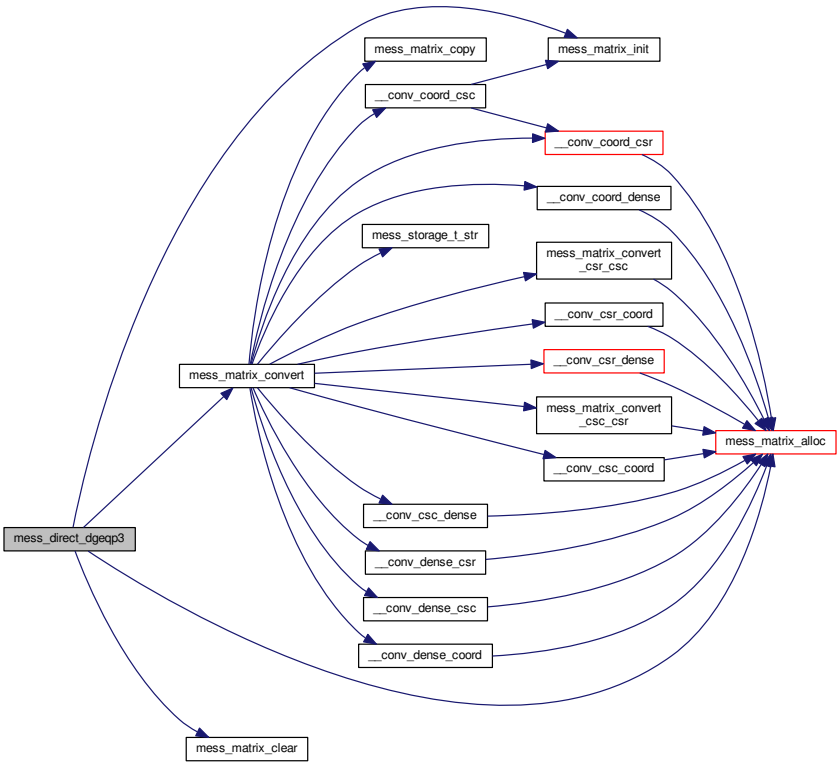
in	*A*	input matrix
out	*Q*	output orthogonal matrix
out	*R*	output upper triangular matrix
out	*perm*	output permutation

Returns

zero on success or a non zero error code otherwise

The mess_direct_dgeqp3 function provides a frontend to the DGEQP3 routine from LAPACK. If a parameter points to NULL, R is not computed.

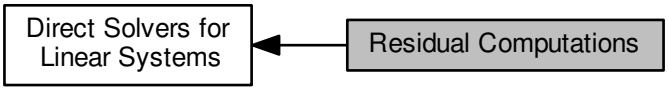
Here is the call graph for this function:



8.21 Residual Computations

This group contains various dense two norm residual computations.

Collaboration diagram for Residual Computations:



Functions

- int mess_direct_res2 (mess_operation_t op, mess_matrix A, mess_vector x, mess_vector b, double ∗resid, double ∗relresid)

  Compute the (relative) residual of $Ax = b$.
- int mess_direct_res2m (mess_operation_t op, mess_matrix A, mess_matrix x, mess_matrix b, double ∗resid, double ∗relresid)

  Compute the (relative) residual of $Ax = b$.
- int mess_direct_sylvester_res2 (mess_operation_t op, mess_matrix A, mess_matrix H, mess_matrix M, mess_matrix X, double ∗res)

  Compute the 2-norm residual of a standard Sylvester equation.
- int mess_direct_sylvestersg_res2 (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix H, mess_matrix M, mess_matrix X, double ∗res)

  Compute the 2-norm residual of a semi-generalized Sylvester equation.
- int mess_direct_generalized_sylvester_res2 (mess_operation_t op, mess_matrix A, mess_matrix F, mess↩_matrix E, mess_matrix H, mess_matrix M, mess_matrix X, double ∗res)

  Compute the 2-norm residual of a generalized Sylvester equation.
- int mess_direct_lyapunov_res2 (mess_operation_t op, mess_matrix A, mess_matrix M, mess_matrix X, double ∗res)

  Compute the 2-norm residual of a Lyapunov Equation (DENSE).
- int mess_direct_generalized_lyapunov_res2 (mess_operation_t op, mess_matrix A, mess_matrix E, mess↩_matrix M, mess_matrix X, double ∗res2)

  Compute the 2-norm residual of a generalized Lyapunov Equation.
- int mess_direct_stein_res2 (mess_operation_t op, mess_matrix A, mess_matrix M, mess_matrix X, double ∗res2)

  Compute the 2-norm residual of a Stein Equation (dense).
- int mess_direct_generalized_stein_res2 (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix M, mess_matrix X, double ∗res2)

  Compute the 2-norm residual of a generalized Stein Equation.
- int mess_direct_care_res2 (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X, double ∗res2)

  Compute the 2-norm residual of the algebraic (generalized) Riccati equation.

8.21.1 Detailed Description

The residual computations can be used to check if a given solver has computed a good solution to a given problem. All residuals are computed as dense vectors or matrices. It might not be a good idea to use them for large matrix equations.

This section provides residual computations for many linear matrix equations and ordinary linear systems.
Please keep in mind that all these operation work in dense arithmetics. That means in the case of large-scale matrix equation they might fail. In this case we refer to the techniques presented in the Low-Rank Methods for Matrix Equations section.

8.21.2 Function Documentation

8.21.2.1 int mess_direct_care_res2 (mess_matrix *A*, mess_matrix *E*, mess_matrix *Q*, mess_matrix *G*, mess_matrix *X*, double * *res2*)

Parameters

in	*A*	input matrix
in	*E*	input matrix of the Riccati Equation (NULL if E does not exist)
in	*Q*	input matrix
in	*G*	input matrix
in	*X*	input solution of the Riccati Equation
out	*res2*	output 2-norm residual

Returns

zero on success or a non zero error code otherwise

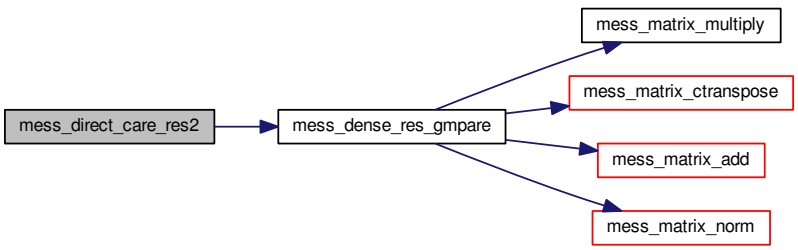
The mess_direct_care_res2 function computes the 2-norm residual of the dense Riccati Equation:

$$res2 = \|Q + A^T X + XA - XGX\|_2.$$

If E is not NULL then we compute the 2-norm residual of the generalized Riccati Equation

$$res2 = \|Q + A^T XE + E^T XA - E^T XGXE\|_2.$$

Here is the call graph for this function:



8.21.2.2 int mess_direct_generalized_lyapunov_res2 (mess_operation_t *op*, mess_matrix *A*, mess_matrix *E*, mess_matrix *M*, mess_matrix *X*, double * *res2*)

Parameters

in	*op*	input operation on matrices A and E
in	*A*	input matrix A
in	*E*	input matrix E
in	*M*	input right hand side
in	*X*	input solution
out	*res2*	output residual

Returns

zero on success or a non zero error code otherwise

The mess_direct_generalized_lyapunov_res2 function computes the 2-norm resiudal of a generalized Lyapunov Equation:

$$op(A)Xop(E)^T + op(E)Xop(A)^T + M = 0$$

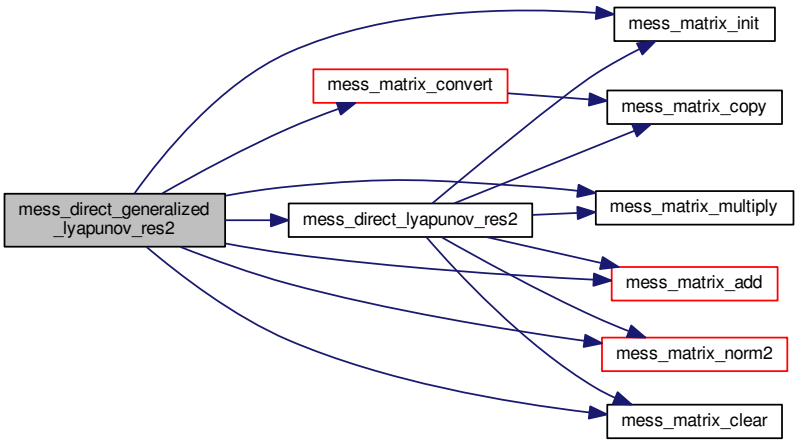
that means it computes:

$$res2 = \|op(A)Xop(E)^T + op(E)Xop(A)^T + M\|_2,$$

where op can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$

Here is the call graph for this function:



8.21.2.3 int mess_direct_generalized_stein_res2 (mess_operation_t *op*, mess_matrix *A*, mess_matrix *E*, mess_matrix *M*, mess_matrix *X*, double * *res2*)

Parameters

in	*op*	input operation on matrices A and E
in	*A*	input matrix A
in	*E*	input matrix E
in	*M*	input right hand side
in	*X*	input solution
out	*res2*	output residual

Returns

zero on success or a non zero error code otherwise

The mess_direct_generalized_stein_res2 function computes the 2-norm resiudal of a generalized Stein Equation:

$$op(A)Xop(A)^T - op(E)Xop(E)^T + M = 0$$

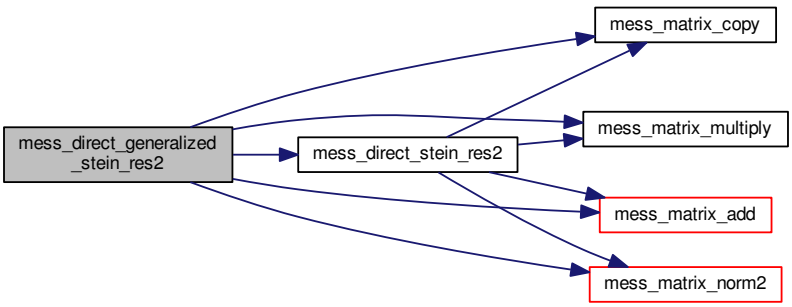
that means it computes:

$$res2 = \|op(A)Xop(A)^T - op(E)Xop(E)^T + M\|_2,$$

where op can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$

Here is the call graph for this function:



8.21.2.4 int mess_direct_generalized_sylvester_res2 (mess_operation_t *op*, mess_matrix *A*, mess_matrix *F*, mess_matrix *E*, mess_matrix *H*, mess_matrix *M*, mess_matrix *X*, double * *res*)

Parameters

in	*op*	input operation on the matrices A, F, E and H

Parameters

in	*A*	input first matrix
in	*F*	input second matrix
in	*E*	input third matrix
in	*H*	input fourth matrix
in	*M*	input right hand side matrix
in	*X*	input solution matrix
out	*res*	norm

Returns

zero on success or a non zero error code otherwise

The mess_direct_generalized_sylvester_res2 function computes the 2-Norm residual of the generalized Sylvester equation

$$R := op(A)Xop(F) + op(E)Xop(H) + M = 0$$

that means it computes

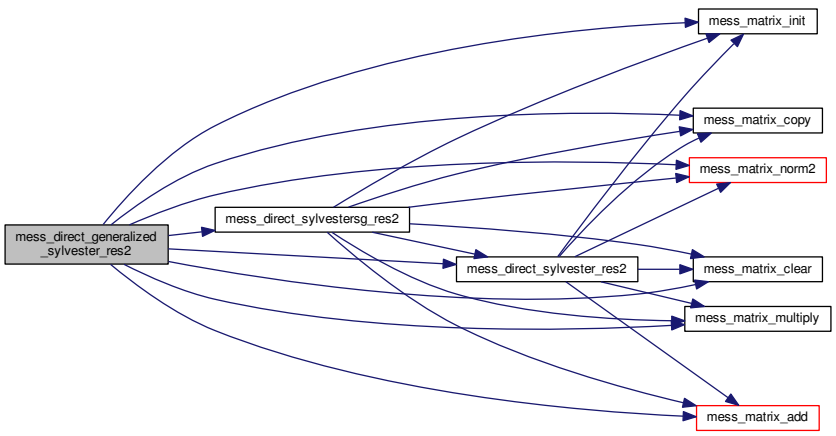
$$\|op(A)Xop(F) + op(E)Xop(H) + M\|_2,$$

where op can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$
MESS_OP_HERMITIAN	$op(A) = A^H$

It is implemented as dense computation. The matrix R is computed inside the function. If the input matrices are too large, this function will be time consuming.

Here is the call graph for this function:



8.21.2.5 int mess_direct_lyapunov_res2 (mess_operation_t *op*, mess_matrix *A*, mess_matrix *M*, mess_matrix *X*, double * *res*)

Parameters

in	*op*	input operation on matrix A
in	*A*	input system matrix of the Lyapunov equation
in	*M*	input right hand side of the Lyapunov Equation
in	*X*	input solution of the Lyapunov Equation
out	*res*	two-norm of the residual

Returns

zero on success or a non zero error code otherwise

The mess_direct_lyapunov_res2 function computes the 2-norm residual of a Lyapunov Equation:

$$op(A)X + Xop(A)^T + M = 0$$

that means it computes:

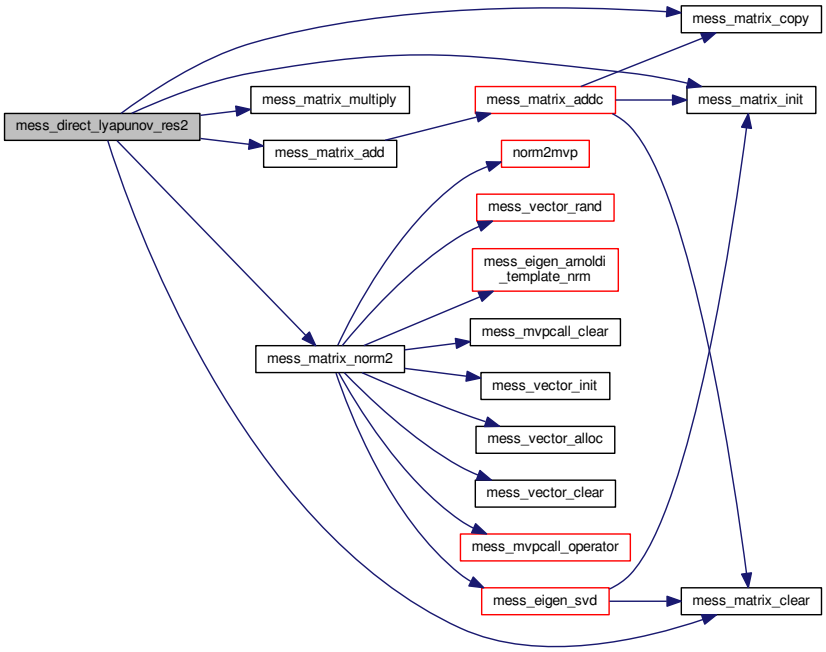
$$res = \|op(A)X + Xop(A)^T + M\|_2,$$

where op can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$

All computations are done in dense arithmetics. Please do not use this function for large-scale equations.

Here is the call graph for this function:



8.21.2.6 int mess_direct_res2 (mess_operation_t *op*, mess_matrix *A*, mess_vector *x*, mess_vector *b*, double * *resid*, double * *relresid*)

Parameters

in	*op*	input operation on matrix A
in	*A*	input system matrix A
in	*x*	input x vector
in	*b*	input b vector
out	*resid*	residual
out	*relresid*	relative residual

Returns

zero on success or a non zero error code otherwise

The mess_direct_res2 function computes the absolute and the relative residual of

$$op(A)x = b$$

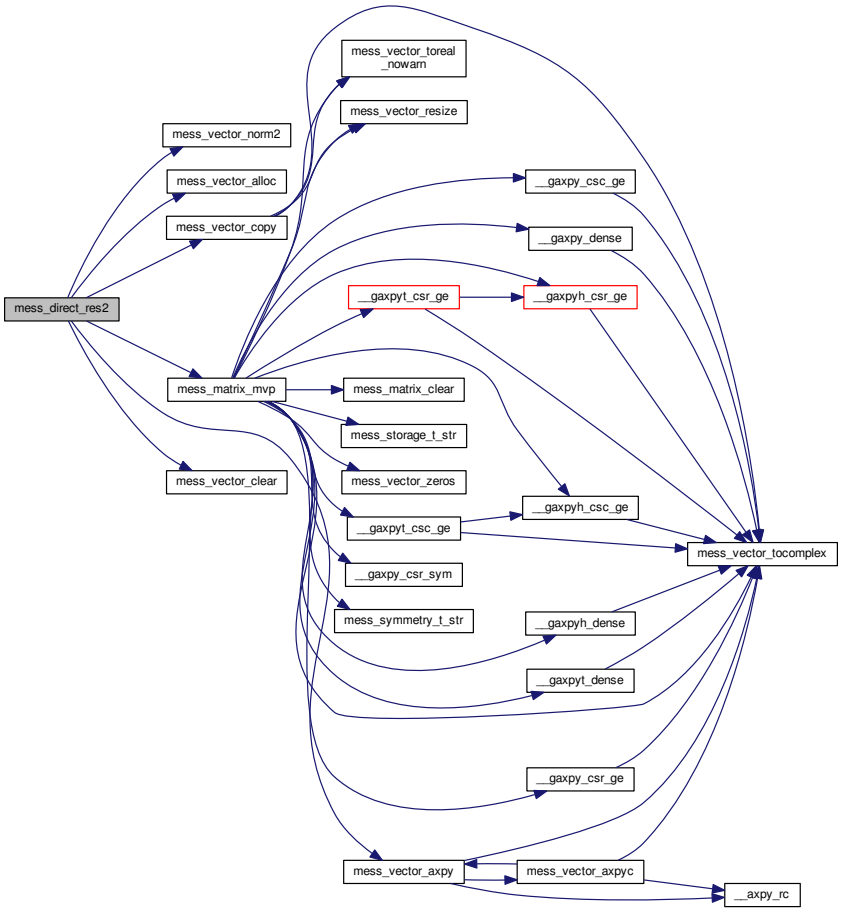
where op can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$
MESS_OP_HERMITIAN	$op(A) = A^H$

See also

mess_direct_res2_shifted
mess_direct_res2m

Here is the call graph for this function:



8.21.2.7 int mess_direct_res2m (mess_operation_t *op*, mess_matrix *A*, mess_matrix *x*, mess_matrix *b*, double * *resid*, double * *relresid*)

Parameters

in	*op*	input operation on matrix A
in	*A*	input system matrix A
in	*x*	input x matrix
in	*b*	input b matrix
out	*resid*	residual
out	*relresid*	relative residual

Returns

zero on success or a non zero error code otherwise

The mess_direct_res2m function computes the absolute and the relative residual of

$$op(A)x = b$$

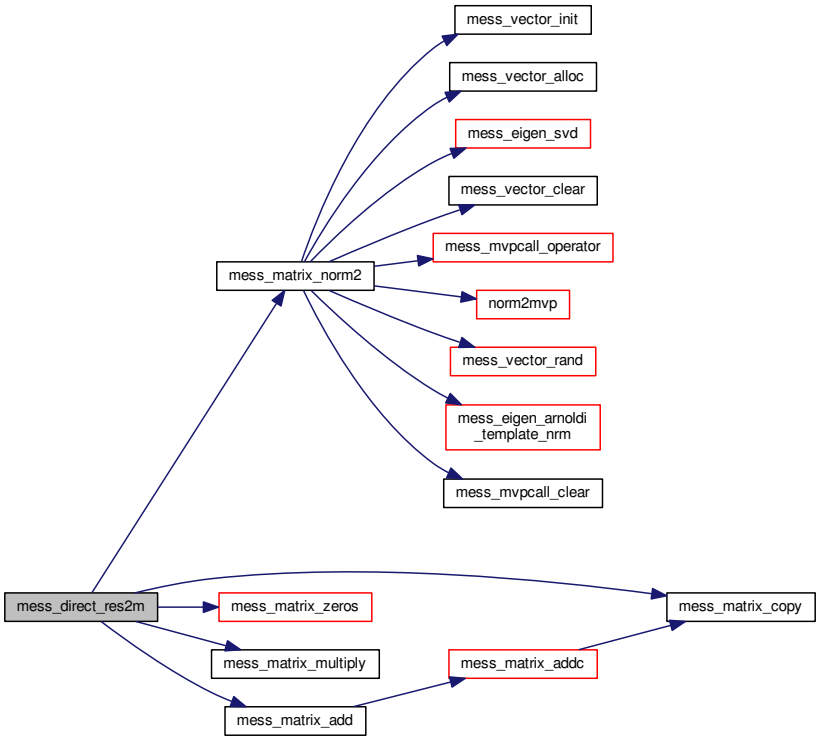
where op can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$
MESS_OP_HERMITIAN	$op(A) = A^H$

See also

mess_direct_res2_shifted
mess_direct_res2

Here is the call graph for this function:



8.21.2.8 int mess_direct_stein_res2 (mess_operation_t *op*, mess_matrix *A*, mess_matrix *M*, mess_matrix *X*, double * *res*)

Parameters

in	*op*	input operation on matrix A
in	*A*	input system matrix of the Stein equation
in	*M*	input right hand side of the Stein equation
in	*X*	input solution of the Stein equation
out	*res*	two-norm of the residual

Returns

zero on success or a non zero error code otherwise

The mess_direct_stein_res2 function computes the 2-norm residual of a Stein Equation:

$$op(A)Xop(A)^T - X + M = 0$$

that means it computes:

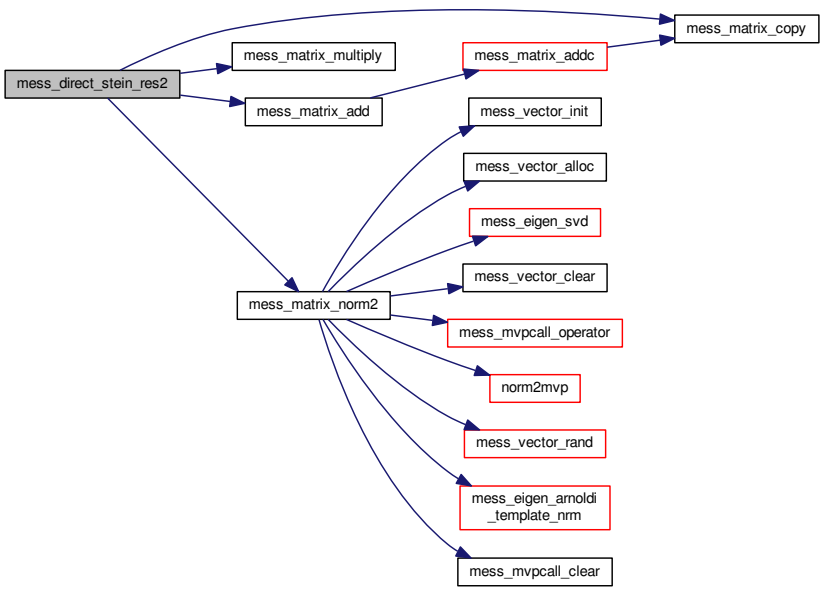
$$res = \|op(A)Xop(A)^T - X + M\|_2,$$

where op can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$

All computations are done in dense arithmetics. Please do not use this function for large-scale equations.

Here is the call graph for this function:



8.21.2.9 int mess_direct_sylvester_res2 (mess_operation_t *op*, mess_matrix *A*, mess_matrix *H*, mess_matrix *M*, mess_matrix *X*, double * *res*)

Parameters

in	*op*	input operation on matrices A and H
in	*A*	input first matrix
in	*H*	input second matrix
in	*M*	input right hand side matrix
in	*X*	input solution matrix
out	*res*	norm

Returns

zero on success or a non zero error code otherwise

The mess_direct_sylvester_res2 function computes the 2-Norm residual of the Sylvester equation

$$R := op(A)X + Xop(H) + M = 0$$

that means it computes

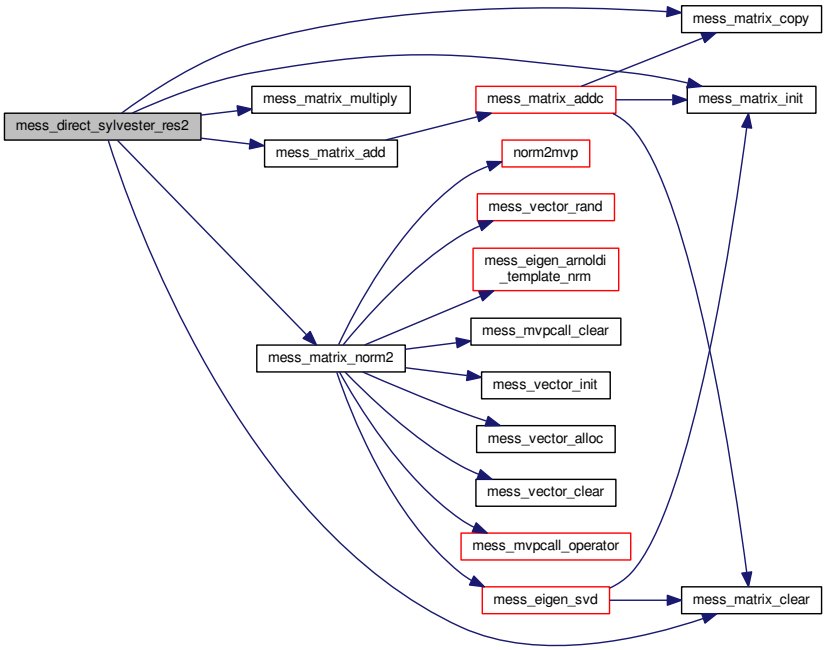
$$res = \|op(A)X + Xop(H) + M\|_2,$$

where op can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$
MESS_OP_HERMITIAN	$op(A) = A^H$

It is implemented as dense computation. The matrix R is computed inside the function. If the input matrices are too large, this function will be time consuming.

Here is the call graph for this function:



8.21.2.10 int mess_direct_sylvestersg_res2 (mess_operation_t *op*, mess_matrix *A*, mess_matrix *E*, mess_matrix *H*, mess_matrix *M*, mess_matrix *X*, double * *res*)

Parameters

in	*op*	input operation on the matrices A, E and H
in	*A*	input first matrix
in	*E*	input second matrix
in	*H*	input third matrix
in	*M*	input right hand side matrix
in	*X*	input solution matrix
out	*res*	norm

Returns

zero on success or a non zero error code otherwise

The mess_direct_sylvestersg_res2 function computes the 2-Norm residual of the semi-generalized Sylvester equation

$$R := op(A)X + op(E)Xop(H) + M = 0$$

that means it computes

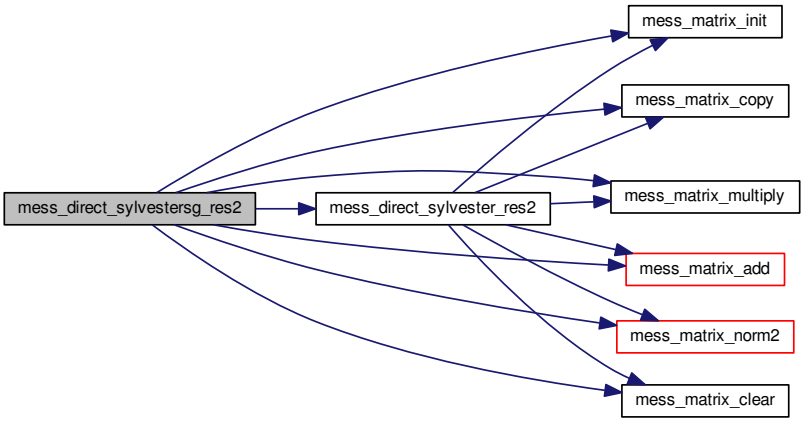
$$\|op(A)X + op(E)Xop(H) + M\|_2,$$

where op can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$
MESS_OP_HERMITIAN	$op(A) = A^H$

It is implemented as dense computation. The matrix R is computed inside the function. If the input matrices are too large, this function will be time consuming.

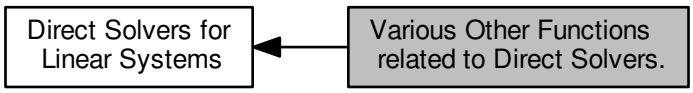
Here is the call graph for this function:



8.22 Various Other Functions related to Direct Solvers.

This group contains various other function related to the direct solution of linear systems. Mostly used internally.

Collaboration diagram for Various Other Functions related to Direct Solvers.:



Data Structures

- struct mess_direct_levelset_st

Typedefs

- typedef struct mess_direct_levelset_st mess_direct_levelset

Functions

- int mess_direct_splsolve (mess_matrix G, mess_matrix B, mess_int_t k, mess_int_t *top, mess_int_t *xi, double *x, const mess_int_t *pinv)

  Solve $x = G^{-1}B(:,k)$ with all sparse.
- int mess_direct_splsolvec (mess_matrix G, mess_matrix B, mess_int_t k, mess_int_t *top, mess_int_t *xi, mess_double_cpx_t *x, const mess_int_t *pinv)

  Solve $x = G^{-1}B(:,k)$ with all sparse with complex data.
- int mess_decomp_lureuse_kernelcsr (mess_int_t rows, mess_int_t cols, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_int_t *lcolptr, mess_int_t *lrowptr, mess_int_t *ucolptr, mess_int_t *urowptr, mess_int_t *p, mess_int_t *q, double *lvalues, double *uvalues)
- int mess_decomp_lureuse_kernelcsr_complex (mess_int_t rows, mess_int_t cols, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_int_t *lcolptr, mess_int_t *lrowptr, mess_int_t *ucolptr, mess_int_t *urowptr, mess_int_t *p, mess_int_t *q, mess_double_cpx_t *lvalues, mess_double_cpx_t *uvalues)
- int mess_decomp_lureuse_kernelcsr_flopcount (mess_int_t rows, mess_int_t cols, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_int_t *lcolptr, mess_int_t *lrowptr, mess_int_t *ucolptr, mess_int_t *urowptr, mess_int_t *p, mess_int_t *q, double *lvalues, double *uvalues, mess_int_t *flops)

8.22.1 Detailed Description

It includes the analysis of triangular matrices and the solve of

$$Ax = B(:,k).$$

8.22.2 Typedef Documentation

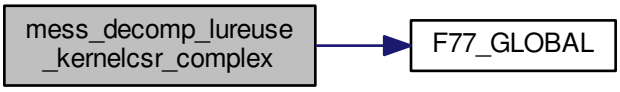
8.22.2.1 typedef struct mess_direct_levelset_st mess_direct_levelset

8.22.3 Function Documentation

8.22.3.1 int mess_decomp_lureuse_kernelcsr (mess_int_t *rows*, mess_int_t *cols*, double * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_int_t * *lcolptr*, mess_int_t * *lrowptr*, mess_int_t * *ucolptr*, mess_int_t * *urowptr*, mess_int_t * *p*, mess_int_t * *q*, double * *lvalues*, double * *uvalues*)

8.22.3.2 int mess_decomp_lureuse_kernelcsr_complex (mess_int_t *rows*, mess_int_t *cols*, mess_double_cpx_t * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_int_t * *lcolptr*, mess_int_t * *lrowptr*, mess_int_t * *ucolptr*, mess_int_t * *urowptr*, mess_int_t * *p*, mess_int_t * *q*, mess_double_cpx_t * *lvalues*, mess_double_cpx_t * *uvalues*)

Here is the call graph for this function:



8.22.3.3 int mess_decomp_lureuse_kernelcsr_flopcount (mess_int_t *rows*, mess_int_t *cols*, double * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_int_t * *lcolptr*, mess_int_t * *lrowptr*, mess_int_t * *ucolptr*, mess_int_t * *urowptr*, mess_int_t * *p*, mess_int_t * *q*, double * *lvalues*, double * *uvalues*, mess_int_t * *flops*)

8.22.3.4 int mess_direct_splsolve (mess_matrix *G*, mess_matrix *B*, mess_int_t *k*, mess_int_t * *top*, mess_int_t * *xi*, double * *x*, const mess_int_t * *pinv*)

Parameters

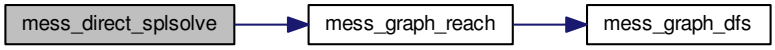
in	*G*	input lower triangular matrix
in	*pinv*	input inverse row permutation of G
in	*B*	input right hand side matrix
in	*k*	input number of the column in B
out	*x*	solution
out	*top*	top of the chi stack
out	*xi*	the reach(L,B(:,col)) stack

Returns

zero on success or a non zero error code otherwise

The mess_direct_splsolve fucntion solves $x = G^{-1}B(:,k)$.
Ported from `CSparse`. For more details look at [13].

Here is the call graph for this function:



8.22.3.5 int mess_direct_splsolvec (mess_matrix *G*, mess_matrix *B*, mess_int_t *k*, mess_int_t * *top*, mess_int_t * *xi*, mess_double_cpx_t * *x*, const mess_int_t * *pinv*)

Parameters

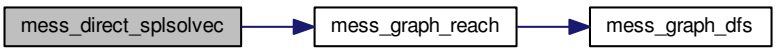
in	*G*	input lower triangular matrix
in	*pinv*	input inverse row permutation of G
in	*B*	input right hand side matrix
in	*k*	input number of the column in B
out	*x*	solution
out	*top*	top of the chi stack
out	*xi*	the reach(L,B(:,col)) stack

Returns

zero on success or a non zero error code otherwise

The mess_direct_splsolvec function solves $x = G^{-1}B(:,k)$ with all sparse with complex data. Ported from `CSparse`. For more details look at [13].

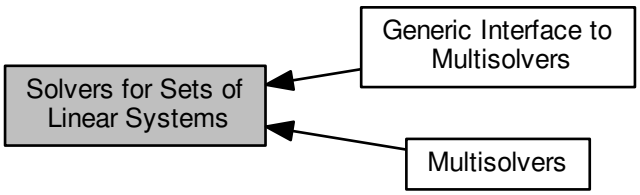
Here is the call graph for this function:



8.23 Solvers for Sets of Linear Systems

This group contains multi-direct solvers for a shifted linear systems and other useful subroutines from this area.

Collaboration diagram for Solvers for Sets of Linear Systems:



Modules

- Generic Interface to Multisolvers

  This category describes the interface to the multi-direct solvers.
- Multisolvers

  This category contains multi-direct solvers for a shifted linear systems.

8.23.1 Detailed Description

This group provides multi-direct solvers for a system

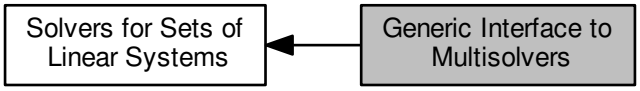
$$(sA + pE)x = b,$$

where s and p are a set of shiftparameters. The basic operations to deal with a solver object is described in Generic Interface to Multisolvers.

8.24 Generic Interface to Multisolvers

This category describes the interface to the multi-direct solvers.

Collaboration diagram for Generic Interface to Multisolvers:



Data Structures

- struct mess_multidirect_st

  Representation of a generic direct solver.

Typedefs

- typedef struct mess_multidirect_st $*$ mess_multidirect

  Type definition to hide one pointer of the mess_multidirect_st structure.

Enumerations

Functions

- int mess_multidirect_init (mess_multidirect $*$direct)

  Initialize a mess_multidirect object.
- int mess_multidirect_clear (mess_multidirect $*$direct)

  Clear a mess_multidirect object.
- int mess_multidirect_solve (mess_operation_t op, mess_multidirect solver, mess_int_t ind, mess_vector b, mess_vector x)

  Solve $op(A(i))x = b$.
- int mess_multidirect_solvem (mess_operation_t op, mess_multidirect solver, mess_int_t ind, mess_matrix b, mess_matrix x)

  Solve $op(A(i))X = B$.

8.24.1 Detailed Description

It includes functions to

- clean up and initialize mess_multidirect objects
- load and unload a system with an index
- return the data type or the memory usage of a multisolver
- select a multisolver from a given name
- solve linear systems
- wait until an index is loaded completely.

8.24.2 Typedef Documentation

8.24.2.1 typedef struct mess_multidirect_st* mess_multidirect

The mess_multidirect type definition hides one pointer to the mess_multidirect_st structure.
This is the common way how the mess_multidirect solver structure should be used. All function of the interface take the type definition instead of the pointer to the structure as input parameter.

8.24.3 Enumeration Type Documentation

8.24.3.1 enum mess_multidirect_t

The mess_multidirect_t enumeration contains the possible multi-solvers for

$$(o_i A + p_i E)x = b.$$

The enumeration can be used to select a multisolver via mess_multidirect_select or inside the ADI processes to select the shifted solver during the iteration. This selection is done in the mess_options structure.

Enumerator

MESS_MULTIDIRECT_SPARSE_LU Select the internal Single-Pattern Multi-Value multisolver.

See also

mess_multidirect_create_sparse_lu

MESS_MULTIDIRECT_UMFPACK_LU Select the `UMFPACK` based multissolver.

See also

mess_multidirect_create_umfpack

8.24.4 Function Documentation

8.24.4.1 int mess_multidirect_clear (mess_multidirect * *direct*)

Parameters

in	*direct*	input pointer to the object

Returns

always zero

The mess_multidirect_clear function clears a mess_multidirect object and call the clear function of the solver.

8.24.4.2 int mess_multidirect_init (mess_multidirect * *direct*)

Parameters

in,out	*direct*	pointer to the mess_multidirect object

Returns

always zero

The mess_multidirect_init function initializes a mess_multidirect object.
If no memory is left, MESS_ERROR_MEMORY is returned.

8.24.4.3 int mess_multidirect_solve (mess_operation_t *op*, mess_multidirect *solver*, mess_int_t *ind*, mess_vector *b*, mess_vector *x*)

Parameters

in	*op*	input operation applied to matrix
in	*solver*	input multisolver
in	*ind*	input index of the desired matrix
in	*b*	input right hand side vector
out	*x*	solution vector

Returns

zero on success or a non-zero error value otherwise

The mess_multidirect_solve function solves the system

$$op(A(ind))x = b$$

with x and b vectors, where op is a mess_operation_t, i.e. op can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$
MESS_OP_HERMITIAN	$op(A) = A^H$

8.24.4.4 int mess_multidirect_solvem (mess_operation_t *op*, mess_multidirect *solver*, mess_int_t *ind*, mess_matrix *b*, mess_matrix *x*)

Parameters

in	*op*	input operation applied to matrix
in	*solver*	input multisolver
in	*ind*	input index of the desired matrix
in	*b*	input right hand side matrix
out	*x*	solution matrix

Returns

zero on success or a non-zero error value otherwise

The mess_multidirect_solvem function solves the system

$$op(A(ind))X = B$$

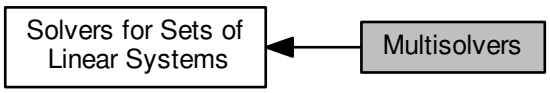
with X and B matrices, where op is a mess_operation_t, i.e. op can be

- MESS_OP_NONE ($op(A) = A$),
- MESS_OP_TRANSPOSE ($op(A) = A^T$),
- MESS_OP_HERMITIAN ($op(A) = A^H$).

8.25 Multisolvers

This category contains multi-direct solvers for a shifted linear systems.

Collaboration diagram for Multisolvers:



Functions

- int mess_multidirect_create_umfpack (mess_matrix matrix, mess_vector shiftsl, mess_vector shiftsr, mess_multidirect mlu, mess_direct base, mess_matrix shiftmatrix)

  Create a multisolver on base of UMFPACK.
- int mess_multidirect_create_sparse_lu (mess_matrix matrix, mess_vector shiftsl, mess_vector shiftsr, mess_multidirect mlu, mess_direct base, mess_matrix shiftmatrix)

  Create a single-pattern-multi-value LU decomposition of $(sA + pI)$ *or* $(sA + pE)$.
- int mess_multidirect_create (mess_matrix matrix, mess_vector shiftsl, mess_vector shiftsr, mess_multidirect mlu, mess_direct base, mess_matrix shiftmatrix)

  Create a multisolver for shifted linear systems $(sA + pI)$ *or* $(sA + pE)$.
- int mess_multidirect_select (mess_multidirect_t solver)

  Select the multisolver which is used by mess_multidirect_create.

8.25.1 Detailed Description

It includes functions creating multisolvers for a shifted system

$$(sA + pE)x = b$$

with the aid of different routines.
Additionally a Single-Pattern-Multi-Value LU decomposition of the shifted system can be computed.

8.25.2 Function Documentation

8.25.2.1 int mess_multidirect_create (mess_matrix *matrix*, mess_vector *shiftsl*, mess_vector *shiftsr*, mess_multidirect *mlu*, mess_direct *base*, mess_matrix *shiftmatrix*)

Parameters

in	*matrix*	input system matrix
in	*shiftsl*	input left shifts s, in front of A, NULL is not given
in	*shiftsr*	input right shifts p, in front of E
out	*mlu*	multidirect solver to create
in	*base*	input existing solver to get the pattern if it is possible
in	*shiftmatrix*	input matrix to shift if it points to NULL, the identity matrix is used

Returns

zero on success or a non-zero error value otherwise

The mess_multidirect_create function creates a solver for shifted linear systems. Depending on the solver choosen by mess_multidirect_select it call

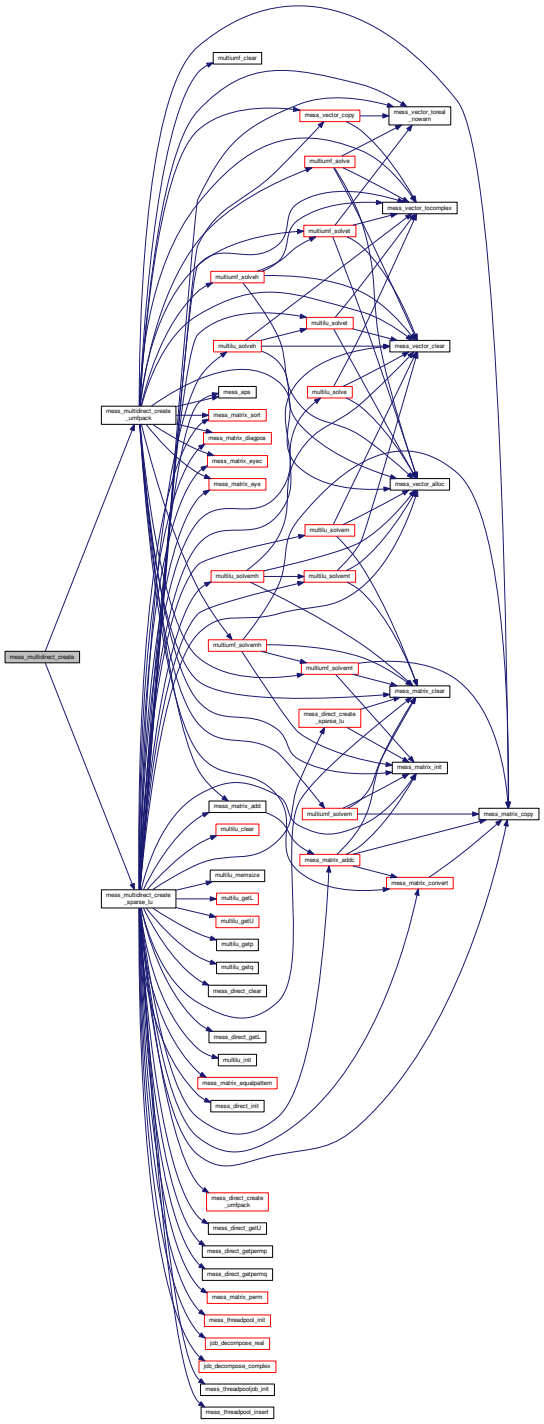
- mess_multidirect_create_sparse_lu,
- mess_multidirect_create_umfpack,

If you specify a base solver you have to ensure that the base solver is created from a matrix with the same pattern as matrix.
If you do not want to use a base solver set it to `NULL`.
If you want to shift with the identity matrix set `shiftmatrix` to `NULL`.
If left or right parameters are not needed set them to `NULL` as well.

See also

mess_multidirect_select

Here is the call graph for this function:



8.25.2.2 int mess_multidirect_create_sparse_lu (mess_matrix *matrix*, mess_vector *shiftsl*, mess_vector *shiftsr*, mess_multidirect *mlu*, mess_direct *base_input*, mess_matrix *shiftmatrix_input*)

Parameters

in	*matrix*	input system matrix
in	*shiftsl*	input left shifts s, in front of A, NULL is not given

Parameters

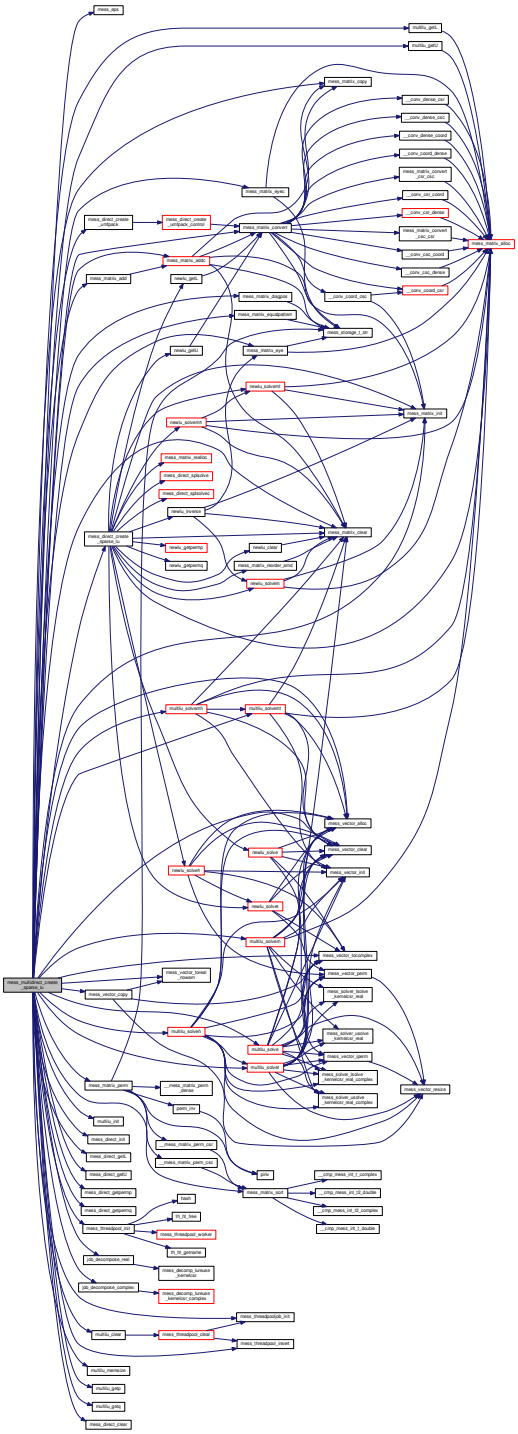
in	*shiftsr*	input right shifts p, in front of E
out	*mlu*	multidirect solver to create
in	*base_input*	existing solver to get the pattern if it is possible
in	*shiftmatrix_input*	matrix to shift if it points to `NULL`, the identity matrix is used

Returns

zero on success or a non-zero error value otherwise

The mess_multidirect_create_sparse_lu function create a single pattern multi values LU decomposition.
If you specify a base solver you have to ensure that the base solver is created from a matrix with the same pattern as matrix.
If you do not want to use a base solver set it to `NULL`.
If you want to shift with the identity matrix set `shiftmatrix_input` to `NULL`.

Here is the call graph for this function:



8.25.2.3 int mess_multidirect_create_umfpack (mess_matrix *matrix*, mess_vector *shiftsl*, mess_vector *shiftsr*, mess_multidirect *mlu*, mess_direct *base*, mess_matrix *shiftmatrix_input*)

Parameters

in	*matrix*	input system matrix
in	*shiftsl*	input left shifts s, in front of A, NULL is not given

Parameters

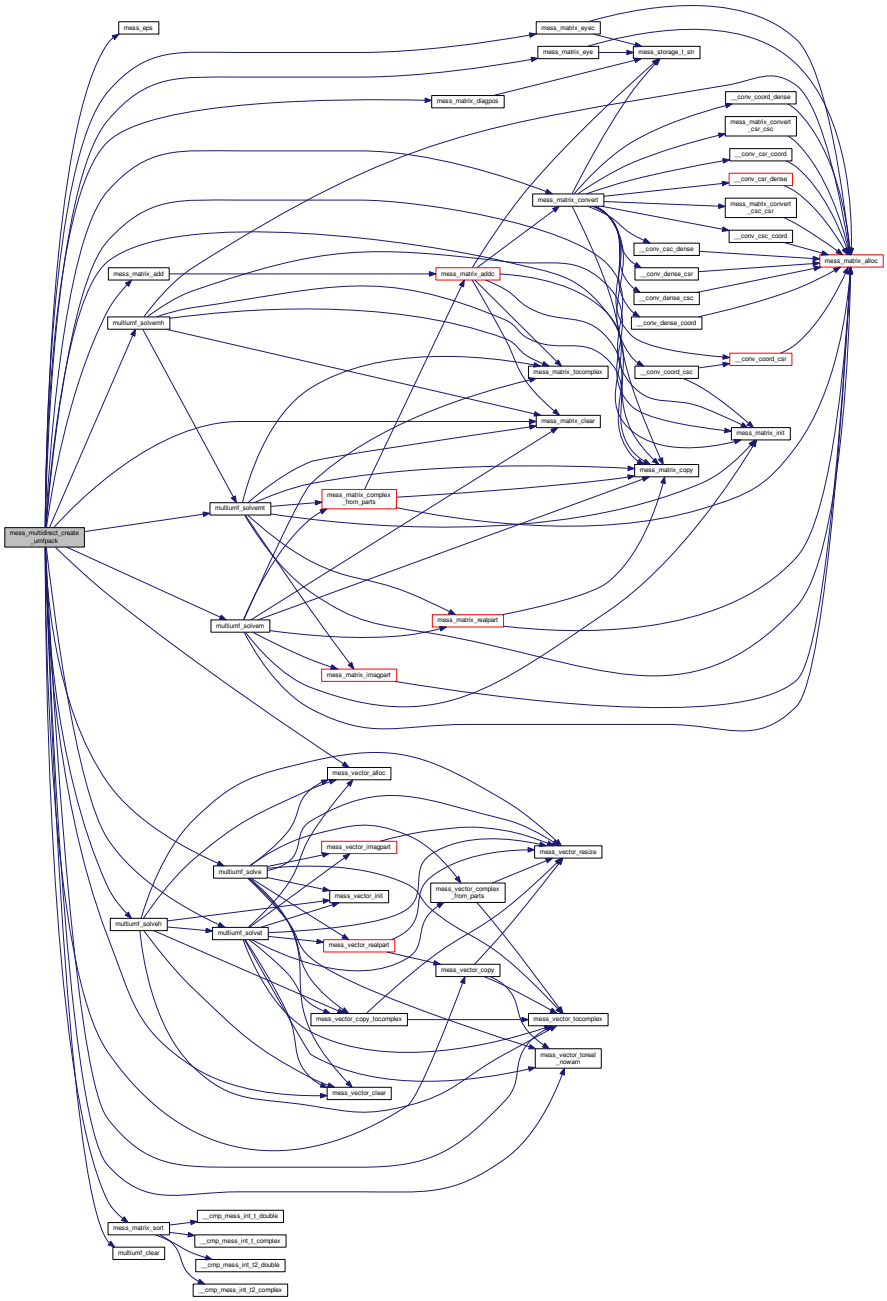
in	*shiftsr*	input right shifts p, in front of E
out	*mlu*	multidirect solver to create
in	*base*	input existing solver to get the pattern if it is possible
in	*shiftmatrix_input*	matrix to shift if it points to `NULL`, the identity matrix is used

Returns

zero on success or a non-zero error value otherwise

The mess_multidirect_create_umfpack function creates a multisolver from `UMFPACK`.

Here is the call graph for this function:



8.25.2.4 int mess_multidirect_select (mess_multidirect_t *solver*)

Parameters

in	*solver*	input Solver to select

Returns

zero on success or a non zero error code otherwise

The mess_multidirect_select function selects one of the following solver to be used as backend in mess_multidirect_create :

- mess_multidirect_create_sparse_lu,
- mess_multidirect_create_umfpack,

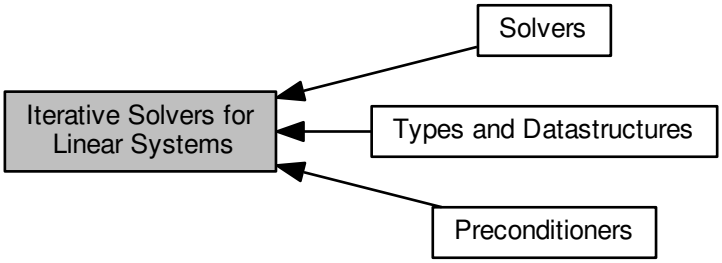
See also

mess_multidirect_create

8.26 Iterative Solvers for Linear Systems

This group contains preconditioner, iterative solvers for a system of linear equations and other useful subroutines from this area.

Collaboration diagram for Iterative Solvers for Linear Systems:



Modules

- Types and Datastructures

  This category contains all solver related data structures, functions and type definitions.
- Solvers

  This category contains iterative solvers for a system of linear equations and other useful subroutines.
- Preconditioners

  This category contains various preconditioner.

8.26.1 Detailed Description

A **linear iteration method** to solve a system of linear equations $Ax = b$ generally looks like

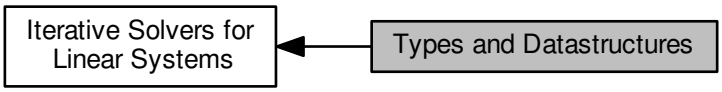
$$x^{k+1} = Mx^k + Nb,$$

where $M = (I - NA)$. Matrices M and N are specified in the different methods.
A **preconditioner** P of a matrix A is a matrix such that $P^{-1}A$ has a smaller condition number than A. It is useful in linear iteration methods since the rate of convergence for most linear iterative solvers increases as the condition number of a matrix decreases as a consequence of preconditioning.

8.27 Types and Datastructures

This category contains all solver related data structures, functions and type definitions.

Collaboration diagram for Types and Datastructures:



Data Structures

- struct __mess_solver_status

  Representation of status information of an iterative solvers.
- struct __mess_solver_options

  Representation of options for iterative solvers.
- struct mess_precond_st

  Representation of a preconditioner.

Macros

- #define MESS_PRECOND_DIAG 1

  Definition of a diagonal preconditioner.
- #define MESS_PRECOND_ILU0 2

  Definition of a $ILU(0)$ preconditioner.
- #define MESS_PRECOND_ILUK 3

  Definition of a $ILU(k)$ preconditioner.

Typedefs

- typedef struct __mess_solver_status $*$ mess_solver_status

  Type definition for status information of an iterative solvers.
- typedef struct __mess_solver_options $*$ mess_solver_options

  Type definition for options of an iterative solvers.
- typedef struct mess_precond_st $*$ mess_precond

  Type definition for a preconditioner.

Functions

- int mess_solver_options_init (mess_solver_options ∗opt)
  Initialize mess_solver_options.
- int mess_solver_options_print (mess_solver_options opt)
  Print a mess_solver_options object.
- int mess_solver_options_clear (mess_solver_options ∗opt)
  Clear a mess_solver_options object.
- int mess_solver_status_init (mess_solver_status ∗opt)
  Initialize a mess_solver_status object.
- int mess_solver_status_print (mess_solver_status opt)
  Print a mess_solver_status object to stdout.
- int mess_solver_status_clear (mess_solver_status ∗opt)
  Clear a mess_solver_status object.
- int mess_solver_status_clean (mess_solver_status stat)
  Reset values in mess_solver_status.

8.27.1 Detailed Description

It includes the structures solver status and options and the corresponding initialization, print and clean up functions. Additionally it contains the type definition of a preconditioner.

8.27.2 Macro Definition Documentation

8.27.2.1 #define MESS_PRECOND_DIAG 1

MESS_PRECOND_DIAG defines a diagonal preconditioner.

8.27.2.2 #define MESS_PRECOND_ILU0 2

MESS_PRECOND_ILU0 defines a $ILU(0)$ preconditioner.

8.27.2.3 #define MESS_PRECOND_ILUK 3

MESS_PRECOND_ILUK defines a $ILU(k)$ preconditioner.

8.27.3 Typedef Documentation

8.27.3.1 typedef struct mess_precond_st∗ mess_precond

The mess_precond type definition defines a CSS preconditioner.

8.27.3.2 typedef struct __mess_solver_options∗ mess_solver_options

The mess_solver_options type definition defines options for an iterative solver.

8.27.3.3 typedef struct __mess_solver_status∗ mess_solver_status

The mess_solver_status type definition defines the status information about an iterative solver.

8.27.4 Function Documentation

8.27.4.1 int mess_solver_options_clear (mess_solver_options ∗ *opt*)

Parameters

in	*opt*	input pointer to the mess_solver_options object

Returns

always zero

The mess_solver_options_clear function clears a mess_solver_options object.

8.27.4.2 int mess_solver_options_init (**mess_solver_options** ∗ *opt*)

Parameters

in,out	*opt*	pointer to the mess_solver_options object

Returns

always zero

The mess_solver_options_init function initializes a mess_solver_options object.

8.27.4.3 int mess_solver_options_print (**mess_solver_options** *opt*)

Parameters

in	*opt*	input input mess_solver_options objects

Returns

zero on success or a non-zero error value

The mess_solver_options_print function print a mess_solver_options to stdout.

8.27.4.4 int mess_solver_status_clean (**mess_solver_status** *stat*)

Parameters

in	*stat*	input mess_solver_status object

Returns

always zero

The mess_solver_status_clean function resets all values in mess_solver_status.

8.27.4.5 int mess_solver_status_clear (**mess_solver_status** ∗ *opt*)

Parameters

in	*opt*	input pointer to the mess_solver_status object

Returns

always zero

The mess_solver_status_clear function clears a mess_solver_status object.

8.27.4.6 int mess_solver_status_init (**mess_solver_status** ∗ *opt*)

Parameters

in,out	*opt*	pointer to the mess_solver_status object

Returns

always zero

The mess_solver_status_init function initializes a mess_solver_status object.

8.27.4.7 int mess_solver_status_print (**mess_solver_status** *stat*)

Parameters

in	*stat*	input mess_solver_status object

Returns

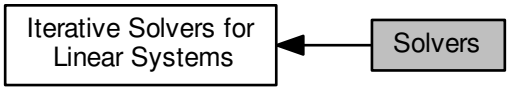
zero on success or a non-zero error value

The mess_solver_status_print function prints a mess_solver_status object to stdout.

8.28 Solvers

This category contains iterative solvers for a system of linear equations and other useful subroutines.

Collaboration diagram for Solvers:



Functions

- int mess_solver_gmres (mess_mvpcall matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with (preconditioned) generalized minimal residual algorithm.
- int mess_solver_gmres_restart (mess_mvpcall matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with restarted GMRES.
- int mess_solver_bicgstab (mess_mvpcall matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with BICGSTAB algorithm.
- int mess_solver_cg (mess_mvpcall matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations using conjugate gradients.
- int mess_solver_sor (mess_matrix matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with successive overrelaxation.
- int mess_solver_ssor (mess_matrix matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with symmetric successive overrelaxation).
- int mess_solver_gaussseidel (mess_matrix matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with the Gauss-Seidel method.
- int mess_solver_jacobi (mess_matrix matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with the Jacobi method.
- int mess_solver_jacobi_convergence (mess_matrix matrix, int *convergence)

  Check if the Jacobi iteration is guaranteed to converge.

8.28.1 Detailed Description

It includes various iterative solvers like BICGSTAB, CG, Gauss-Seidel, GMRES, Jacobi, (S)SOR for a linear system $Ax = b$ and a convergence check for the Jacobi iteration.

8.28.2 Function Documentation

8.28.2.1 int mess_solver_bicgstab (mess_mvpcall *matrix,* mess_precond *pre,* mess_vector *b,* mess_vector *x,* mess_solver_options *opt,* mess_solver_status *stat*)

Parameters

in	*matrix*	input maxtrix A of the equation
in	*pre*	input left preconditioner for the equation (if not used, set to NULL)
in	*b*	input right hand side
in,out	*x*	on input: initial guess, on output: approximated solution
in	*opt*	input object containing the parameters for the iteration, if it is set to NULL defaults are used
out	*stat*	output statitics about the convergence (if not used, set to NULL)

Returns

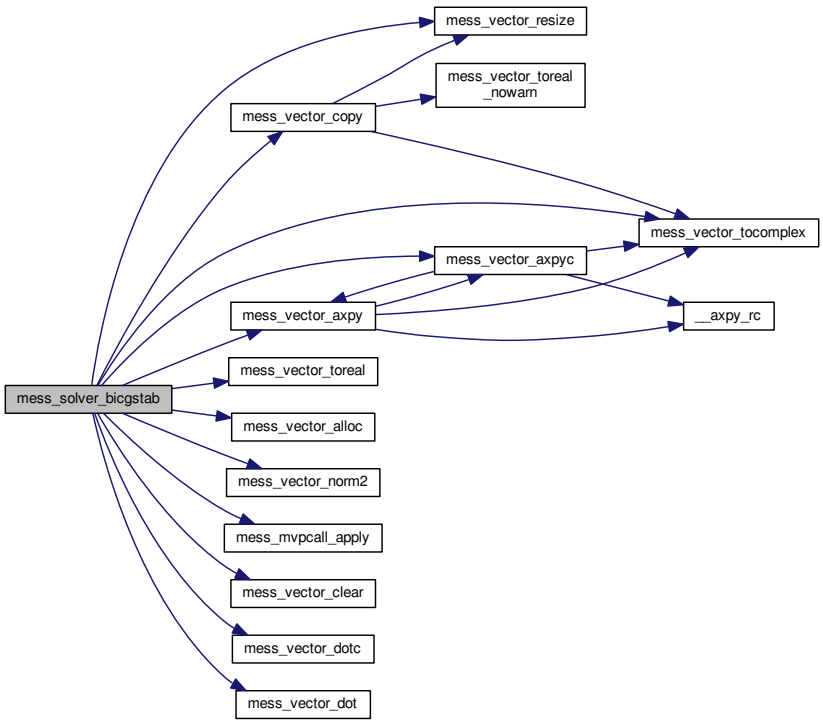
zero on success or a non-zero error value otherwise

The mess_solver_bicgstab function implements the BICGSTAB algorithm to solve

$$Ax = b.$$

It can be accelerated using a left preconditioner defined by pre. It works for real and complex data. Details about it are available in [22].

Here is the call graph for this function:



8.28.2.2 int mess_solver_cg (mess_mvpcall *matrix*, mess_precond *pre*, mess_vector *b*, mess_vector *x*, mess_solver_options *opt*, mess_solver_status *stat*)

Parameters

in	*matrix*	input matrix A of the equation
in	*pre*	input preconditioner
in	*b*	input right hand side
in,out	*x*	solution
in	*opt*	input options
out	*stat*	output status

Returns

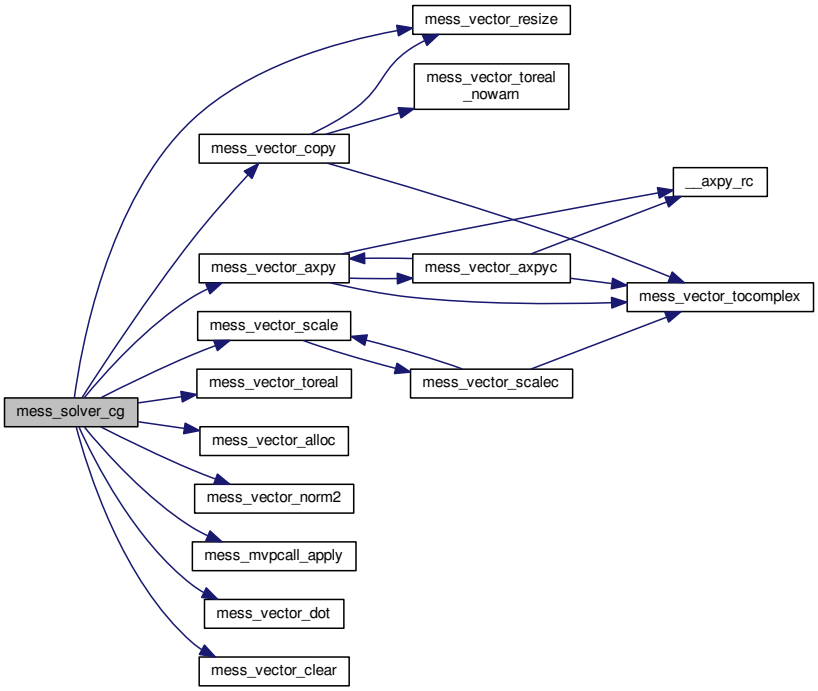
zero on success or a non-zero error value otherwise

The mess_solver_cg function solves a system of linear equations

$$Ax = b$$

using the conjugate gradients (CG) method. Therefore the matrix A need to be symmetric positive definite otherwise the method will fail.
Additionally a preconditioner can be used to accelerate the convergence. The preconditioner must not destroy symmetry and positive definiteness. A list of avialable preconditioners is listed in Preconditioners.

Here is the call graph for this function:



8.28.2.3 int mess_solver_gaussseidel (mess_matrix *matrix*, mess_precond *pre*, mess_vector *b*, mess_vector *x*, mess_solver_options *opt*, mess_solver_status *stat*)

Parameters

in	*matrix*	input matrix A of the equation
in	*pre*	input preconditioning placeholder to fit in the API of the other iterative solvers (ignored for this method)
in	*b*	input right hand side
in,out	*x*	on input: intial solution, on output: computed solution
in	*opt*	input options for the solver (details see mess_solver_options)
out	*stat*	statistics about the solution process (details see mess_solver_status)

Returns

zero on success or a non-zero error value otherwise

The mess_solver_gaussseidel function solves

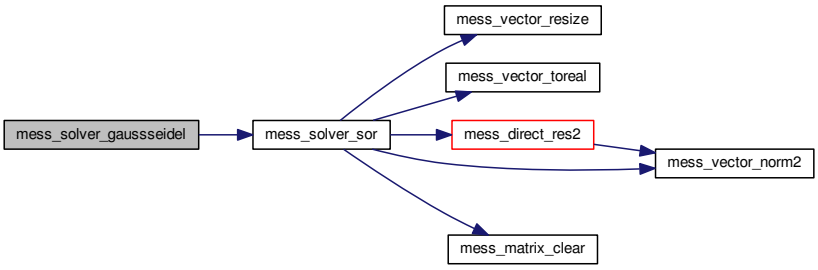
$$Ax = b$$

with Gauss-Seidel method.
It is only a wrapper around the mess_solver_sor function setting the overrelaxation parameter **omega** to 1.

See also

mess_solver_sor

Here is the call graph for this function:



8.28.2.4 int mess_solver_gmres (mess_mvpcall *matrix*, mess_precond *pre*, mess_vector *b*, mess_vector *x*, mess_solver_options *opt*, mess_solver_status *stat*)

Parameters

in	*matrix*	input maxtrix A of the equation
in	*pre*	input left preconditioner (if not used, set to NULL)
in	*b*	input right hand side
in,out	*x*	on input: initial guess, on output: approximated solution
in	*opt*	input object containing the parameters for the iteration, if it is set to NULL defaults are used
out	*stat*	output statitics about the convergence (if not used, set to NULL)

Returns

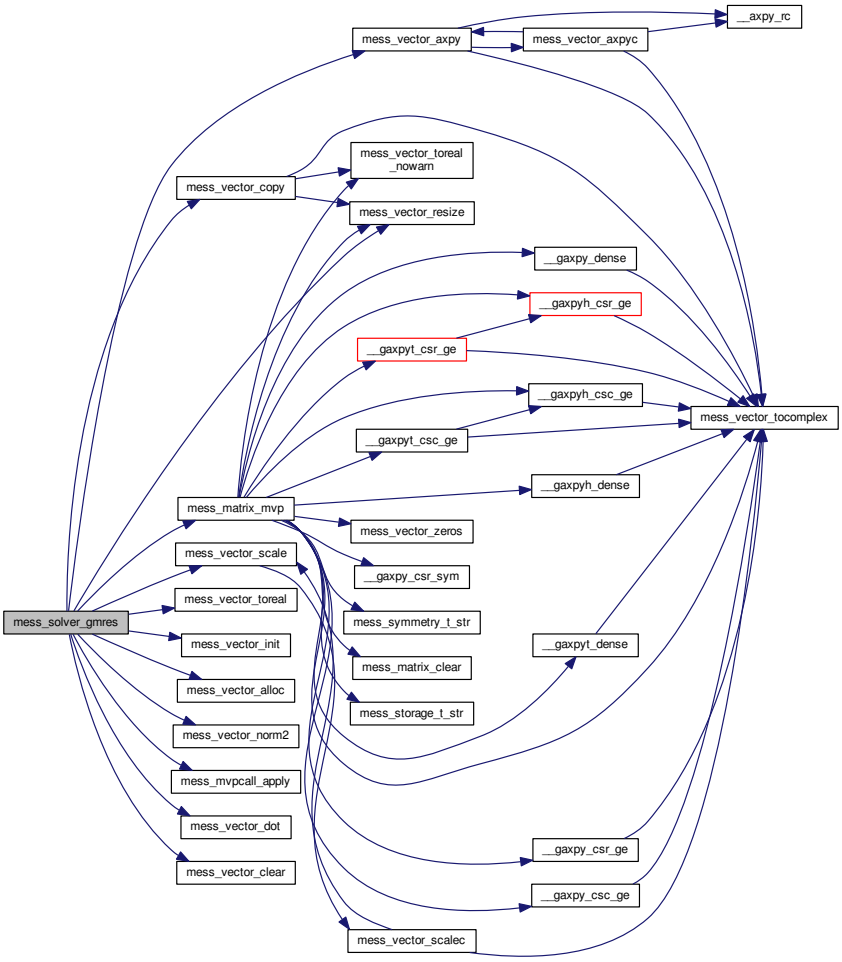
zero on success or a non-zero error value otherwise

The mess_solver_gmres function solves a system of linear equations with the (preconditioned) generalized minimal residual (GMRES) algorithm.
The GMRES-algorithm is a Krylov subspace method to solve linear systems with general matrices.
For symmetric or skew-symmetric matrices special matrix-vector products are used.
An initial solution need to be provided in the vector x which is used as x_0. The relative tolerance provided by `tol` and the dimension of the Krylov subspace provided by `maxit` are parameters for the algorithm. If the relative tolerance is reached before the complete subspace is spanned the algorihm terminates with a hint that no restart is needed. The function online provides one GMRES run.
To implement a complete restarted GMRES you can use the restart component of the CSS_Solver_Status structure to determine if you call the function with the solution of the last run as initial solution again.
A short version could be:

```
int restart = 0;
do {
mess_solver_gmres(matrix, NULL, b, x, opt, stat);
restart = stat->need_restart;
} while ( restart == 1);
```

The second parameter `pre` is the preconditioner. If pre is not null, the preconditioner is used.

Here is the call graph for this function:



8.28.2.5 int mess_solver_gmres_restart (mess_mvpcall *matrix*, mess_precond *pre*, mess_vector *b*, mess_vector *x*, mess_solver_options *opt*, mess_solver_status *stat*)

Parameters

in	*matrix*	input matrix A of the equation
in	*pre*	input input preconditioner
in	*b*	input right hand side
out	*x*	solution
in	*opt*	input options
out	*stat*	output status

Returns

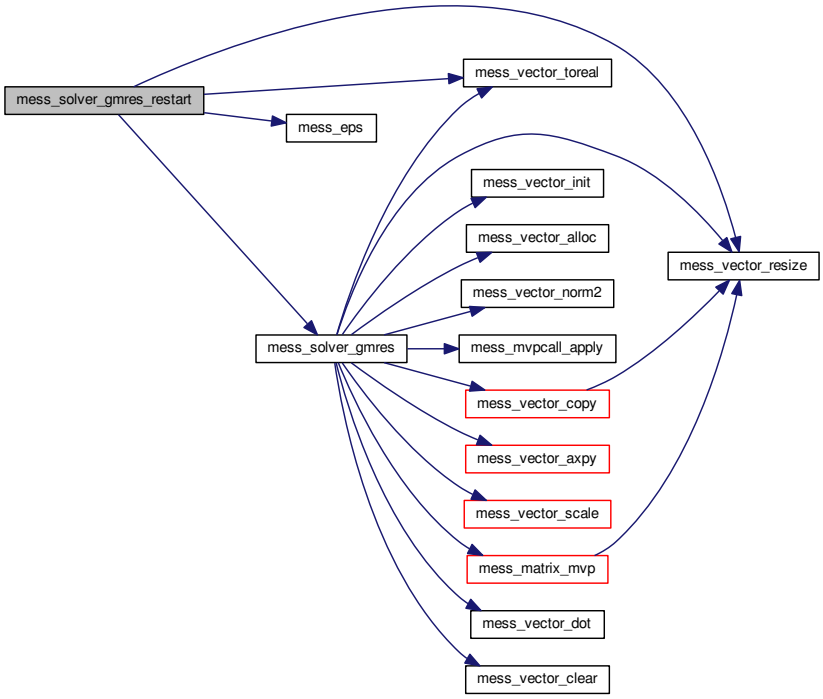
zero on success or a non-zero error value otherwise

The mess_solver_gmres_restart function solves

$$Ax = b$$

with restarted GMRES.

Here is the call graph for this function:



8.28.2.6 int mess_solver_jacobi (mess_matrix *matrix*, mess_precond *pre*, mess_vector *b*, mess_vector *x*, mess_solver_options *opt*, mess_solver_status *stat*)

Parameters

in	*matrix*	input matrix A of the equation
in	*pre*	input preconditioner (ignored for this method)
in	*b*	input right hand side
in,out	*x*	on input: intial solution, on output: computed solution
in	*opt*	input options for the solver (details see mess_solver_options)
out	*stat*	statistics about the solution process (details see mess_solver_status)

Returns

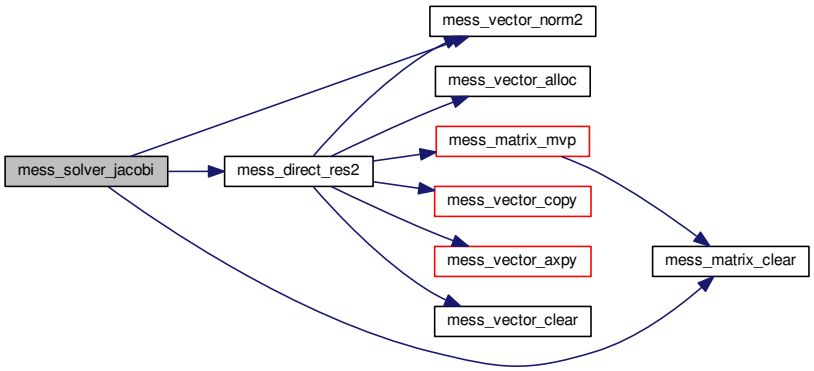
zero on success or a non-zero error value otherwise

The mess_solver_jacobi function implements the Jacobi method to solve a system of linear equations

$$Ax = b.$$

Details about the algorithm can be found in [22] .
Additionally the convergence can be checked before using the mess_solver_jacobi_convergence function.

Here is the call graph for this function:



8.28.2.7 int mess_solver_jacobi_convergence (mess_matrix *matrix*, int * *convergence*)

Parameters

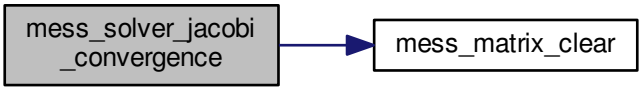
in	*matrix*	input matrix A of the equation
out	*convergence*	output parameter (true if the Jacobi iteration is guaranteed to converge)

Returns

zero on success or a non-zero error value otherwise

The mess_solver_jacobi_convergence function checks if the matrix is strictly diagonal dominant. If the matrix is strictly diagonal dominant the Jacobi iteration is guaranteed to converge. In this case convergence is set to true otherwise to false.
A proof of this convergence criteria can be found in [22] Theorem 4.11.

Here is the call graph for this function:



8.28.2.8 int mess_solver_sor (mess_matrix *matrix*, mess_precond *pre*, mess_vector *b*, mess_vector *x*, mess_solver_options *opt*, mess_solver_status *stat*)

Parameters

in	*matrix*	input matrix A of the equation
in	*pre*	input preconditioning placeholder to fit in the API of the other iterative solvers (ignored for this method)
in	*b*	input right hand side
in,out	*x*	on input: intial solution, on output: computed solution
in	*opt*	input options for the solver (details see mess_solver_options)
out	*stat*	statistics about the solution process (details see mess_solver_status)

Returns

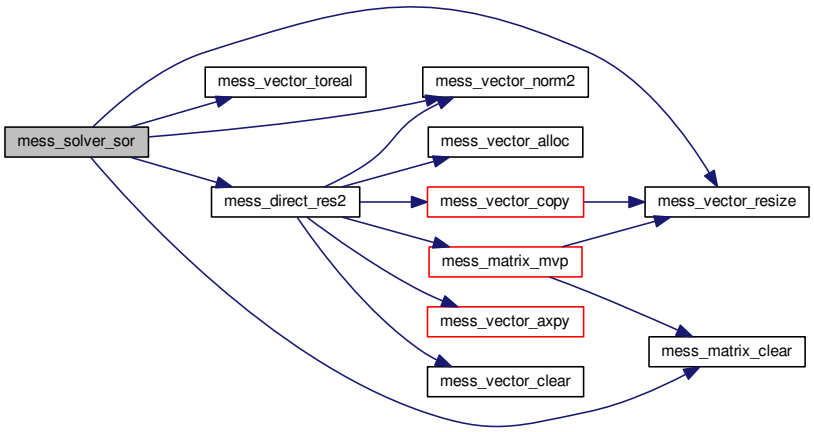
zero on success or a non-zero error value otherwise

The mess_solver_sor function solves a system of linear equations

$$Ax = b$$

with successive overrelaxation (SOR).
The overrelaxation can be set in the omega value in **opt**. If you use the stepdebug feature, the stepdebug function is only evaluated every 5th step. The preconditioner is not used.

Here is the call graph for this function:



8.28.2.9 int mess_solver_ssor (mess_matrix *matrix*, mess_precond *pre*, mess_vector *b*, mess_vector *x*, mess_solver_options *opt*, mess_solver_status *stat*)

Parameters

in	*matrix*	input matrix A of the equation
in	*pre*	input preconditioning placeholder to fit in the API of the other iterative solvers (ignored for this method)
in	*b*	input right hand side of the equation
in,out	*x*	on input: intial solution, on output: computed solution
in	*opt*	input options for the solver (details see mess_solver_options)
out	*stat*	statistics about the solution process (details see mess_solver_status)

Returns

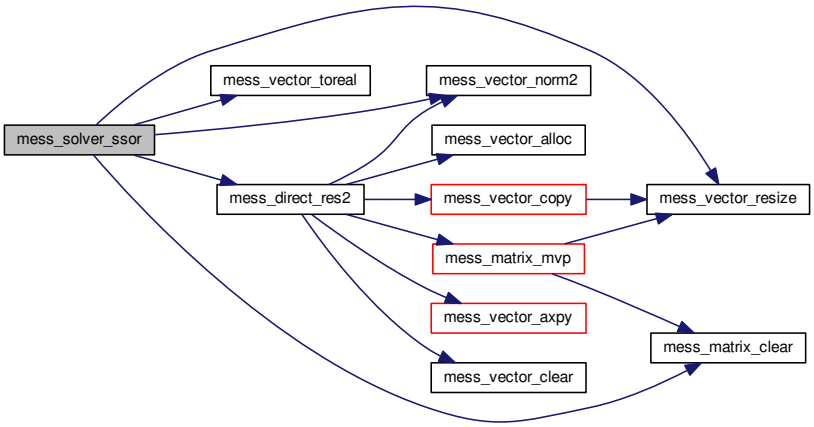
zero on success or a non-zero error value otherwise

The mess_solver_ssor function solves a system of linear equations

$$Ax = b$$

with symmetric successive overrelaxation (SSOR).
The overrelaxation can be set in the omega value in **opt**. If you use the stepdebug feature, the stepdebug function is only evaluated every 5th step. The preconditioner is not used.

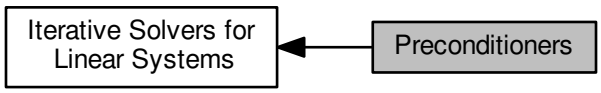
Here is the call graph for this function:



8.29 Preconditioners

This category contains various preconditioner.

Collaboration diagram for Preconditioners:



Functions

- int mess_precond_init (mess_precond *precond)

  Initialize a mess_precond object.
- int mess_precond_clear (mess_precond *precond)

  Clean up the mess_precond structure to use it again.
- int mess_precond_diag (mess_precond pre, mess_matrix mat)

  Setup a diagonal preconditioner.
- int mess_precond_ilu0 (mess_precond pre, mess_matrix matrix)

  Compute an $ILU(0)$ preconditioner.
- int mess_precond_iluk (mess_precond pre, mess_matrix matrix, mess_int_t leveloffill)

  Setup an $ILU(k)$ preconditioner from a matrix.
- int mess_precond_sor (mess_precond pre, mess_matrix matrix, mess_int_t it, double omega)

  Compute a SOR preconditioner.
- int mess_precond_ssor (mess_precond pre, mess_matrix matrix, mess_int_t it, double omega)

  Compute a SSOR preconditioner.

8.29.1 Detailed Description

It includes the initialization and clean up of the special preconditioner structure and contains different preconditioner like a diagonal, ILU or (S)SOR preconditioner.

8.29.2 Function Documentation

8.29.2.1 int mess_precond_clear (mess_precond * *precond*)

Parameters

in,out	*precond*	pointer to the mess_precond

Returns

always zero

The mess_precond_clear function removes the mess_precond object from memory and cleans up.

8.29.2.2 int mess_precond_diag (mess_precond *pre*, mess_matrix *mat*)

Parameters

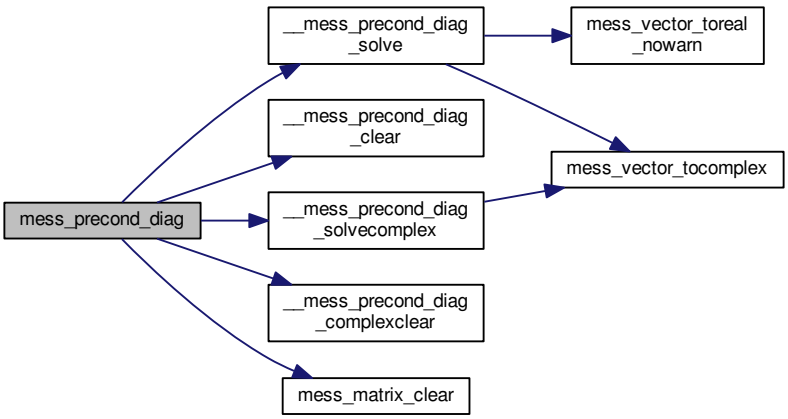
out	*pre*	generated preconditioner
in	*mat*	input matrix to compute the preconditioner for

Returns

zero on success or a non-zero error value otherwise

The mess_precond_diag function generates a diagonal preconditioner from a matrix mat.
If the matrix is not in MESS_CSR storage format it is converted to it internally. The original matrix is kept as it is in this case.

Here is the call graph for this function:



8.29.2.3 int mess_precond_ilu0 (mess_precond *pre*, mess_matrix *matrix*)

Parameters

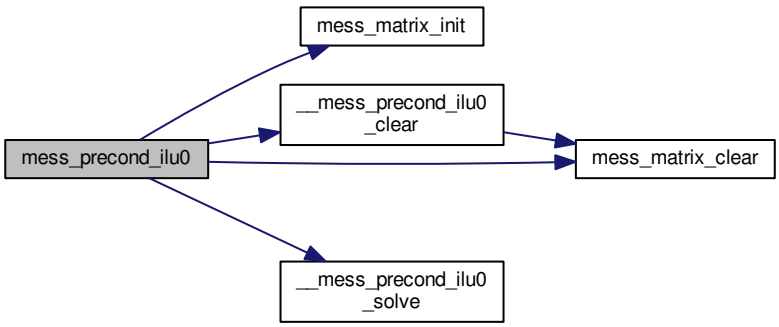
in	*matrix*	input matrix to compute the preconditioner for
out	*pre*	output generated ILU(0) preconditioner

Returns

zero on success or a non-zero error value otherwise

The mess_precond_ilu0 function computes an $ILU(0)$ preconditioner.

Here is the call graph for this function:



8.29.2.4 int mess_precond_iluk (mess_precond *pre*, mess_matrix *mat*, mess_int_t *lof*)

Parameters

out	*pre*	generated preconditioner
in	*mat*	input matrix to compute the preconditioner for
in	*lof*	input level of fill

Returns

zero on success or a non zero error code otherwise

The mess_precond_iluk function computes an $ILU(k)$ preconditioner for a real or complex matrix mat.
All entries with a level $> lof$ are dropped and set to 0. If $lof == 0$ it results in the classical incomplete LU decomposition.

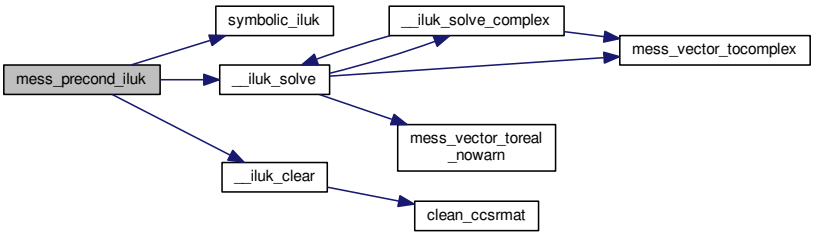
See also

mess_precond
mess_solver_bicgstab

Remarks

- The quality and the efficiency of the preconditioner can be improved by reordering the matrix before using the AMD reordering (mess_matrix_reorder_amd).
- All the diagonals of the input matrix must not be zero.

Here is the call graph for this function:



8.29.2.5 int mess_precond_init (mess_precond * *precond*)

Parameters

in,out	*precond*	pointer to a mess_precond object

Returns

always zero

The mess_precond_init function sets all values in mess_precond to zero and all pointers to `NULL`.

8.29.2.6 int mess_precond_sor (mess_precond *pre*, mess_matrix *matrix*, mess_int_t *it*, double *omega*)

Parameters

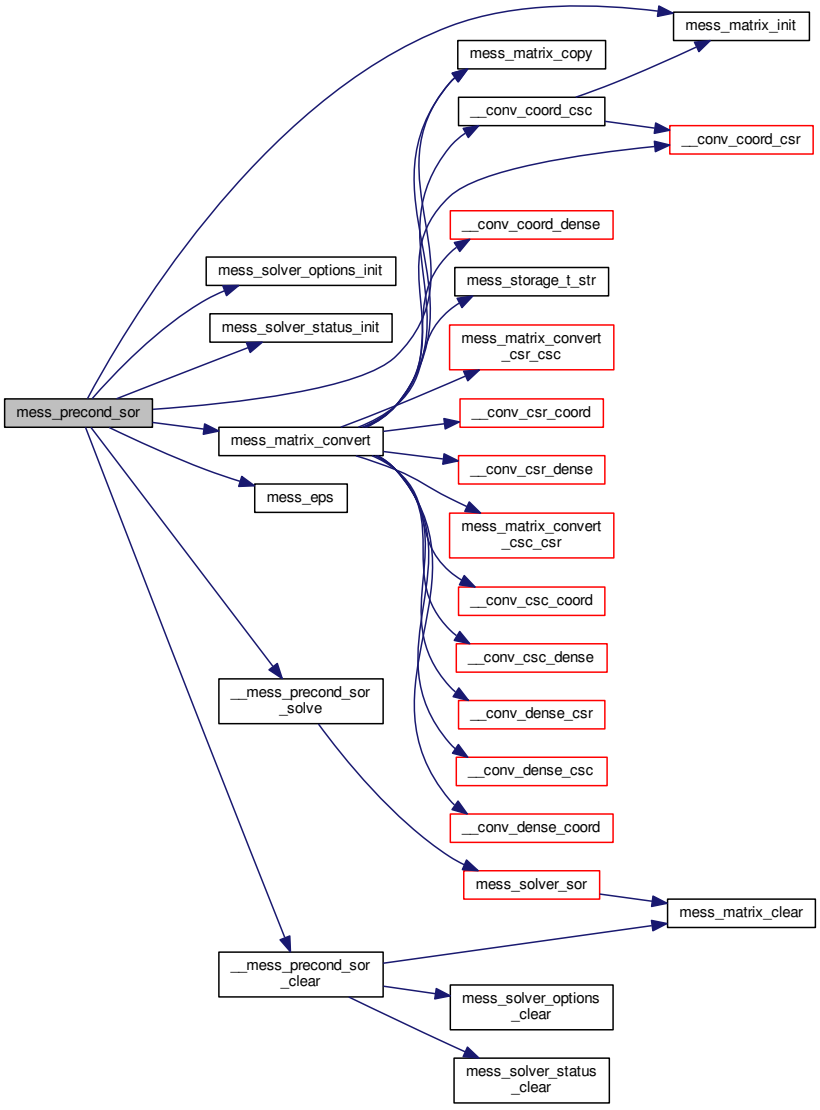
out	*pre*	generated preconditioner
in	*matrix*	input matrix to compute the preconditioner for
in	*it*	input number of steps
in	*omega*	input SOR parameter

Returns

zero on success or a non-zero error value otherwise

The mess_precond_sor function generates a SOR preconditioner on top of the mess_solver_sor function. If a symmetric preconditioner is wanted use mess_solver_ssor instead.

Here is the call graph for this function:



8.29.2.7 int mess_precond_ssor (mess_precond *pre*, mess_matrix *matrix*, mess_int_t *it*, double *omega*)

Parameters

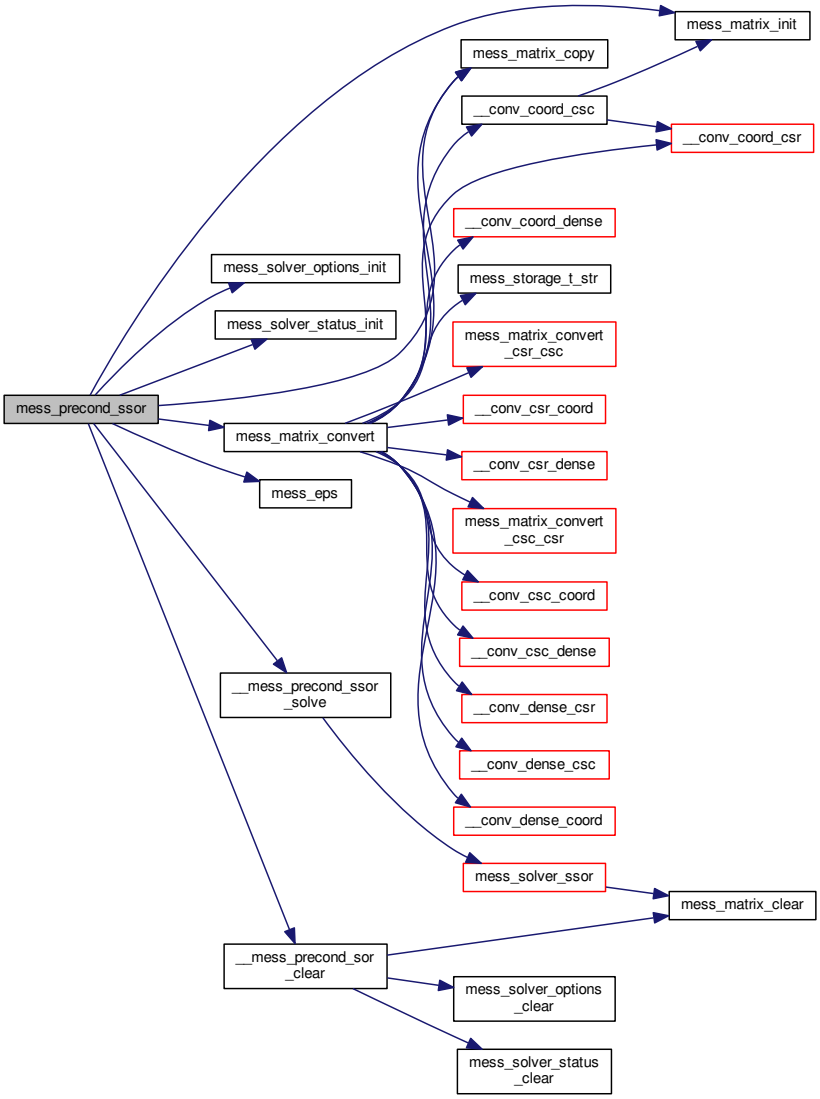
out	*pre*	generated preconditioner
in	*matrix*	input matrix to compute the preconditioner for
in	*it*	input number of steps
in	*omega*	input SSOR parameter

Returns

zero on success or a non-zero error value otherwise

The mess_precond_ssor function generates a SSOR preconditioner on top of the mess_solver_ssor function. The non symmetric case is implemented in mess_precond_sor.

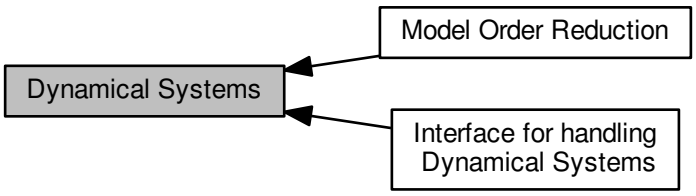
Here is the call graph for this function:



8.30 Dynamical Systems

This module covers all basic things for dynamical systems.

Collaboration diagram for Dynamical Systems:



Modules

- Interface for handling Dynamical Systems

  This category contains a structure, definitions and functions corresponding to first and second order systems.
- Model Order Reduction

  This section contains categories reducing the the order of a GLTI system in different ways.

8.30.1 Detailed Description

Dynamical systems are represented as a

- first order system

$$\begin{array}{rcll} E\dot{x}(t) & = & Ax(t) & +Bu(t) \quad , \\ y(t) & = & Cx(t) & \quad , \end{array}$$

  where $A \in \mathbb{R}^{n\times n}$ is a system matrix, $E \in \mathbb{R}^{n\times n}$ is a mass matrix, $B \in \mathbb{R}^{n\times m}$ is an input matrix and $C \in \mathbb{R}^{p\times n}$ is an output matrix of the first order system.

- second order system

$$\begin{array}{rcll} M\ddot{x}(t) + G\dot{x}(t) + \quad Kx(t) & = & Bu(t) & \quad , \\ y(t) & = & C_p x(t) & +C_v\dot{x}(t) \quad , \end{array}$$

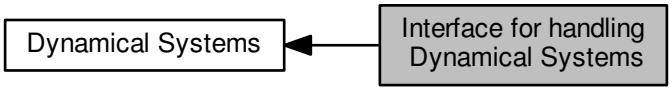
  where $M \in \mathbb{R}^{n\times n}$, $G \in \mathbb{R}^{n\times n}$, $K \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ is an input matrix, $C_p \in \mathbb{R}^{p\times n}$ is a position and $C_v \in \mathbb{R}^{p\times n}$ a velocity output matrix of the second order system.

The time dependent vectors $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ and $y(t) \in \mathbb{R}^p$ are usually state, input and output vector.

8.31 Interface for handling Dynamical Systems

This category contains a structure, definitions and functions corresponding to first and second order systems.

Collaboration diagram for Interface for handling Dynamical Systems:



Data Structures

- struct mess_dynsys_st

  Representation of a dynamical system.

Macros

- #define MESS_DYNSYS_LTI 0x01
- #define MESS_DYNSYS_GLTI 0x02
- #define MESS_DYNSYS_2ND 0x03
- #define MESS_IS_DYNSYS_LTI(x) ((∗x).type == MESS_DYNSYS_LTI)
- #define MESS_IS_DYNSYS_GLTI(x) ((∗x).type == MESS_DYNSYS_GLTI)
- #define MESS_IS_DYNSYS_2ND(x) ((∗x).type == MESS_DYNSYS_2ND)
- #define MESS_IS_DYNSYS_SISO(x) (((∗x).inputs==1) && ((∗x).outputs==1))

Typedefs

- typedef struct mess_dynsys_st ∗ mess_dynsys

  Type definition of easy usage of mess_dynsys_st.

Functions

- char ∗ mess_dynsysstr (unsigned short type)

  Return name of a dynamical system type.
- int mess_dynsys_init (mess_dynsys ∗sys)

  Initialize a mess_dynsys object.
- int mess_dynsys_clear (mess_dynsys ∗sys)

  Clean up a mess_dynsys object.
- int mess_dynsys_lti (mess_dynsys lti, mess_matrix A, mess_matrix B, mess_matrix C)

  Generate a LTI system from matrices A, B and C.
- int mess_dynsys_lti_copy (mess_dynsys lti, mess_matrix A, mess_matrix B, mess_matrix C)

  Generate a LTI system from A, B and C (copy version).
- int mess_dynsys_glti (mess_dynsys lti, mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix E)

Generate a generalized LTI system from matrices A, B, C and E.

- int mess_dynsys_glti_copy (mess_dynsys lti, mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix E)

  Generate a LTI system from A, B, C and E (copy version).
- int mess_dynsys_2nd (mess_dynsys sys, mess_matrix M, mess_matrix G, mess_matrix K, mess_matrix B, mess_matrix Cp, mess_matrix Cv)

  Generate a second order system from matrices M, G, K, B, Cp and Cv.
- int mess_dynsys_2nd_copy (mess_dynsys sys, mess_matrix M, mess_matrix G, mess_matrix K, mess_matrix B, mess_matrix Cp, mess_matrix Cv)

  Generate a second order system from matrices M, G, K, B, Cp and Cv (copy version).
- int mess_dynsys_2nd_to_1st (mess_dynsys sys2nd, mess_dynsys sys1st)

  Convert a second order system to a first order system.
- int mess_dynsys_printinfo (mess_dynsys sys)

  Print information about a dynamical system.
- int mess_dynsys_isstable (mess_dynsys sys, int ∗stable)

  Check if a dynamical system is stable.
- int mess_dynsys_evaltransfer (mess_dynsys sys, double aa, double bb, mess_int_t nsample, mess_vector omega, mess_vector G, mess_matrix ∗Gout)

  Evaluate a transfer function.
- int mess_dynsys_transfer_diff (mess_int_t nsample, mess_matrix ∗Go, mess_matrix ∗Gr, double ∗err2, double ∗rel2, double ∗errinf, double ∗relinf, mess_vector diffvec)

  Compute difference and relative difference between two transfer functions.
- int mess_dynsys_project_to_lti (mess_dynsys sys, mess_matrix V, mess_matrix W, mess_dynsys red)

  Project a dynamical system (LTI or GLTI) to a LTI system.
- int mess_dynsys_project_to_glti (mess_dynsys sys, mess_matrix V, mess_matrix W, mess_dynsys red)

  Project a GLTI system to a GLTI system.
- int mess_dynsys_project_2nd_to_1st (mess_dynsys sys2nd, mess_matrix V, mess_matrix W, mess_dynsys sys1st)

  Project a second order system to a first order system.
- int mess_dynsys_project_2nd_to_2nd (mess_dynsys sys, mess_matrix V, mess_matrix W, mess_dynsys red)

  Project a second order system to a second order system.
- int mess_matrix_isstableg (mess_matrix A, mess_matrix E, int ∗stable)

  Check if a matrix pair is stable.
- int mess_matrix_isstable (mess_matrix A, int ∗stable)

  Check if a matrix is stable.

8.31.1 Detailed Description

It includes a structure for dynamical systems and its corresponding type definition , definitions of (generalized) time invariant systems ((G)LTI) and second order systems.
Additionally there are functions to

- clean up, initialize and print mess_dynsys objects
- evaluate a transfer function
- project different systems
- return the name of the type
- test if a certain system is a (G)LTI system, a second order system or a single input single output (SISO) system

- test if a matrix or a system is stable.

(G)LTI and second order systems can be generated and second order systems can be converted to first order systems:

$$\begin{aligned} \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix} \begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 & I \\ -K & -G \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} &= \begin{bmatrix} 0 \\ B \end{bmatrix} u(t) \quad , \\ y(t) &= \begin{bmatrix} C_p & C_v \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} \quad . \end{aligned}$$

8.31.2 Macro Definition Documentation

8.31.2.1 #define MESS_DYNSYS_2ND 0x03

Define a second order system.

8.31.2.2 #define MESS_DYNSYS_GLTI 0x02

Define a generalized LTI system.

8.31.2.3 #define MESS_DYNSYS_LTI 0x01

Define a LTI system.

8.31.2.4 #define MESS_IS_DYNSYS_2ND(x) ((∗x).type == MESS_DYNSYS_2ND)

Check if a mess_dynsys object is a second order system.

8.31.2.5 #define MESS_IS_DYNSYS_GLTI(x) ((∗x).type == MESS_DYNSYS_GLTI)

Check if a mess_dynsys object is a GLTI system.

8.31.2.6 #define MESS_IS_DYNSYS_LTI(x) ((∗x).type == MESS_DYNSYS_LTI)

Check if a mess_dynsys object is a LTI system.

8.31.2.7 #define MESS_IS_DYNSYS_SISO(x) (((∗x).inputs==1) && ((∗x).outputs==1))

Check if a mess_dynsys object is a SISO system.

8.31.3 Typedef Documentation

8.31.3.1 typedef struct mess_dynsys_st∗ mess_dynsys

The mess_dynsys type definition of easy usage of mess_dynsys_st .

8.31.4 Function Documentation

8.31.4.1 int mess_dynsys_2nd (mess_dynsys *sys*, mess_matrix *M*, mess_matrix *G*, mess_matrix *K*, mess_matrix *B*, mess_matrix *Cp*, mess_matrix *Cv*)

Parameters

out	*sys*	mess_dynsys object
in	*M*	input 2nd derivative matrix
in	*G*	input 1st derivative matrix
in	*K*	input state space matrix
in	*B*	input matrix
in	*Cp*	input positon output matrix
in	*Cv*	input velocity output matrix

Returns

zero on success or a non-zero error value

The mess_dynsys_2nd function builds a mess_dynsys object for a second order system defined by M, G, K, B, C_p and C_v:

$$\begin{array}{rcl} M\ddot{x}(t) + G\dot{x}(t) + \quad Kx(t) & = & Bu(t) \\ y & = & C_p x(t) \quad + C_v \dot{x}(t) \end{array}.$$

The matrices are not copied to the object, only a pointer is used.
When the object is cleared, the matrices will be cleared too.

8.31.4.2 int mess_dynsys_2nd_copy (mess_dynsys *sys*, mess_matrix *M*, mess_matrix *G*, mess_matrix *K*, mess_matrix *B*, mess_matrix *Cp*, mess_matrix *Cv*)

Parameters

out	*sys*	mess_dynsys object
in	*M*	input second derivative matrix
in	*G*	input first derivative matrix
in	*K*	input state space matrix
in	*B*	input matrix
in	*Cp*	input position output matrix
in	*Cv*	input velocity output matrix

Returns

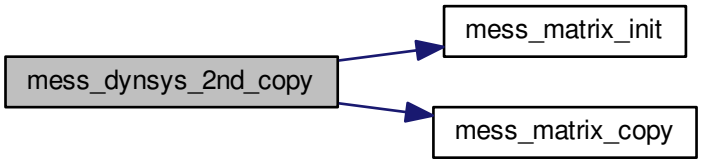
zero on success or a non-zero error value

The mess_dynsys_2nd_copy function builds a mess_dynsys object for a second order system defined by M, G, K, B, C_p and C_v:

$$\begin{array}{rcl} M\ddot{x}(t) + G\dot{x}(t) + \quad Kx(t) & = & Bu(t) \\ y(t) & = & C_p x(t) \quad + C_v \dot{x}(t) \end{array}.$$

The matrices are copied to the object.
It works like mess_dynsys_lti_copy or mess_dynsys_glti_copy.

Here is the call graph for this function:



8.31.4.3 int mess_dynsys_2nd_to_1st (mess_dynsys *sys2nd*, mess_dynsys *sys1st*)

Parameters

in	*sys2nd*	second order system (input)
out	*sys1st*	first order system (output)

Returns

zero on success or a non-zero error value

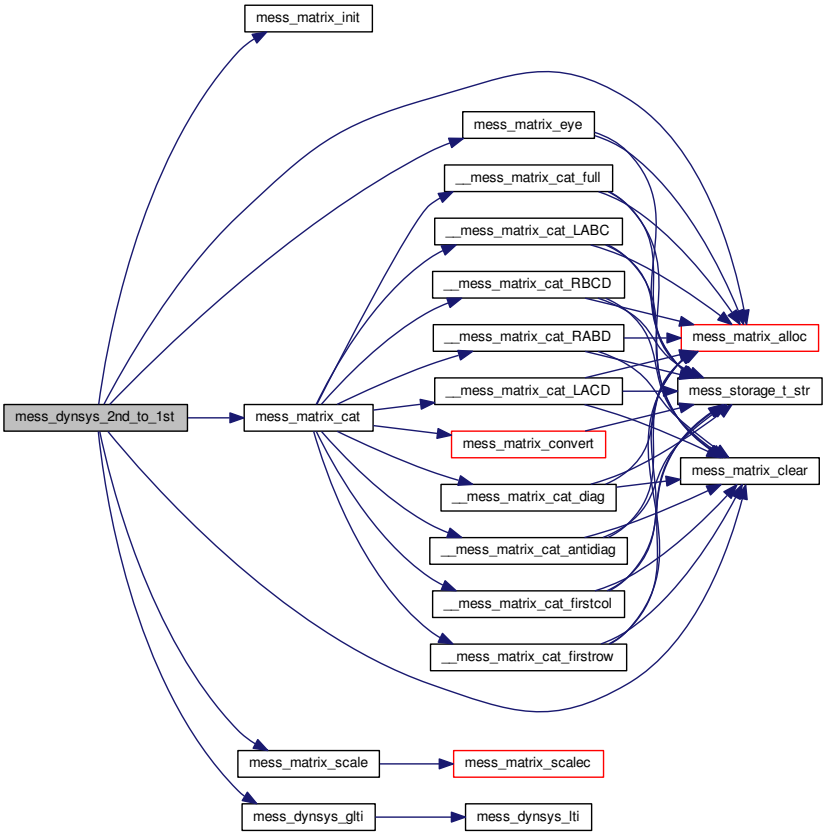
The mess_dynsys_2nd_to_1st function converts a second order system

$$\begin{array}{rcl} M\ddot{x}(t) + G\dot{x}(t) + \quad Kx(t) & = & Bu(t) \\ y & = & C_p x(t) \quad + C_v \dot{x}(t) \end{array} .$$

to a generalized state space first order system

$$\begin{array}{rcl} \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix} \begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} & = & \begin{bmatrix} 0 & I \\ -K & -G \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \\ x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ B \end{bmatrix} u(t) \\ y(t) & = & \begin{bmatrix} C_p & C_v \end{bmatrix} \end{array} .$$

Here is the call graph for this function:



8.31.4.4 int mess_dynsys_clear (mess_dynsys * *sys*)

Parameters

in	*sys*	input pointer to the object

Returns

always zero

The mess_dynsys_clear function cleans up a mess_dynsys object.

Here is the call graph for this function:

8.31.4.5 int mess_dynsys_evaltransfer (mess_dynsys *sys*, double *aa*, double *bb*, mess_int_t *nsample*, mess_vector *omega*, mess_vector *G*, mess_matrix * *Gout*)

Parameters

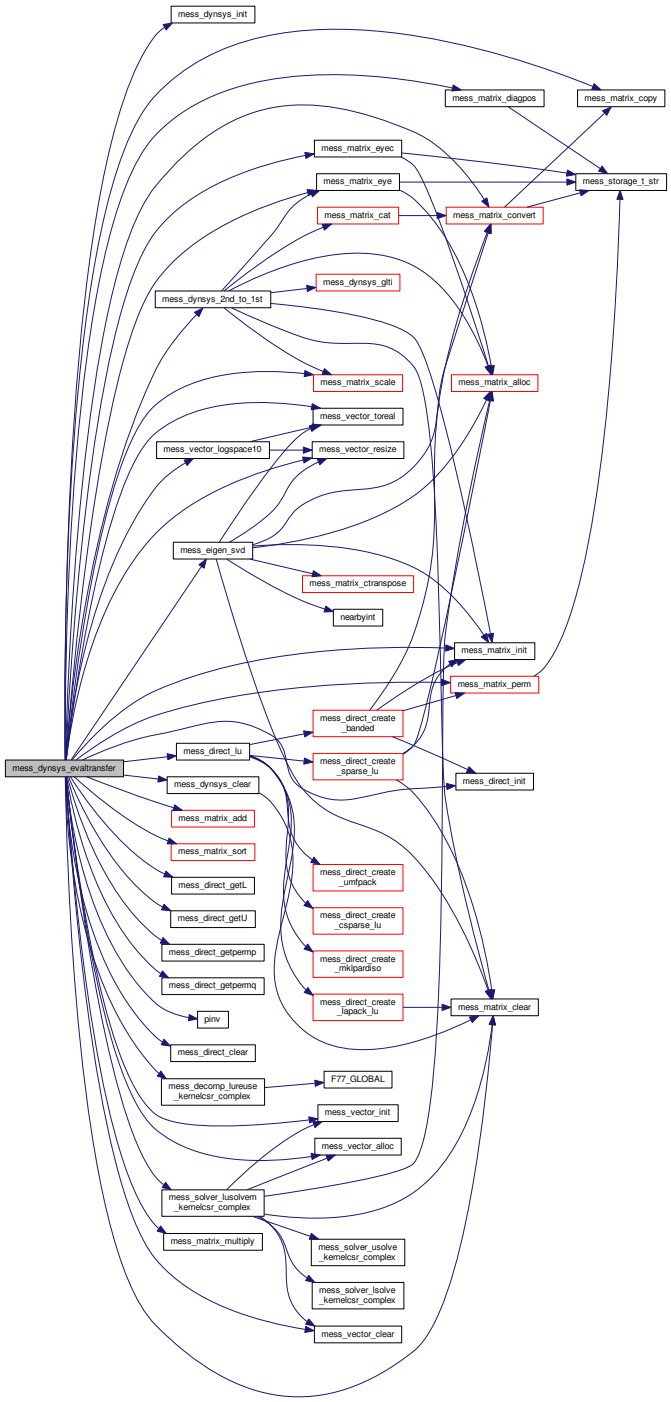
in	*sys*	input LTI System
in	*aa*	input lower bound for the interval
in	*bb*	input upper bound for the interval
in	*nsample*	input number of samples
in,out	*omega*	output interval
out	*G*	output $\max(svd(G(s)))$
out	*Gout*	pointer to a mess_matrix array of size nsample

Returns

zero on succes or a non-zero error value

The mess_dynsys_evaltransfer function evaluates the transfer function of a LTI or a GLTI system on a set of logarithmic equidistant points in $\left[10^{aa}, 10^{bb}\right]$.
The memory usage can be limited using the runtime configuration is OpenMP is enabled. If Gout is not NULL, the G(s) at the interpolation points is returned, too.

Here is the call graph for this function:



8.31.4.6 int mess_dynsys_glti (mess_dynsys *lti*, mess_matrix *A*, mess_matrix *B*, mess_matrix *C*, mess_matrix *E*)

Parameters

out	*lti*	mess_dynsys object
in	*A*	input system matrix

Parameters

in	*B*	input matrix
in	*C*	input output matrix
in	*E*	input mass matrix

Returns

zero on success or a non-zero error value

The mess_dynsys_glti function builds a mess_dynsys object for a generalized LTI system defined by A, B, C and E:

$$\begin{array}{lll} E\dot{x} & = Ax & +Bu \\ y & = Cx & \end{array}.$$

The matrices are not copied to the object, only a pointer is used.
When the object is cleared, the matrices will be cleared too.

Here is the call graph for this function:

8.31.4.7 int mess_dynsys_glti_copy (mess_dynsys *lti*, mess_matrix *A*, mess_matrix *B*, mess_matrix *C*, mess_matrix *E*)

Parameters

out	*lti*	LTI system
in	*A*	input system matrix
in	*B*	input matrix
in	*C*	input output matrix
in	*E*	input mass matrix

Returns

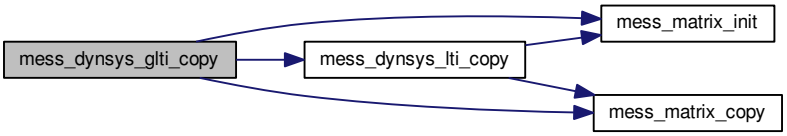
zero on success or a non-zero error value

The mess_dynsys_lti_copy function builds a mess_dynsys object as generalized LTI system defined by the matrices A, B, C and E:

$$\begin{array}{lll} E\dot{x} & = Ax & +Bu \\ y & = Cx & \end{array}.$$

The matrices will be copied to the object. That means $lti \rightarrow A$ is independent from A and so on.

Here is the call graph for this function:



8.31.4.8 int mess_dynsys_init (mess_dynsys * *sys*)

Parameters

in,out	*sys*	pointer to the object

Returns

always zero

The mess_dynsys_init function initializes a mess_dynsys object.

8.31.4.9 int mess_dynsys_isstable (mess_dynsys *sys*, int * *stable*)

Parameters

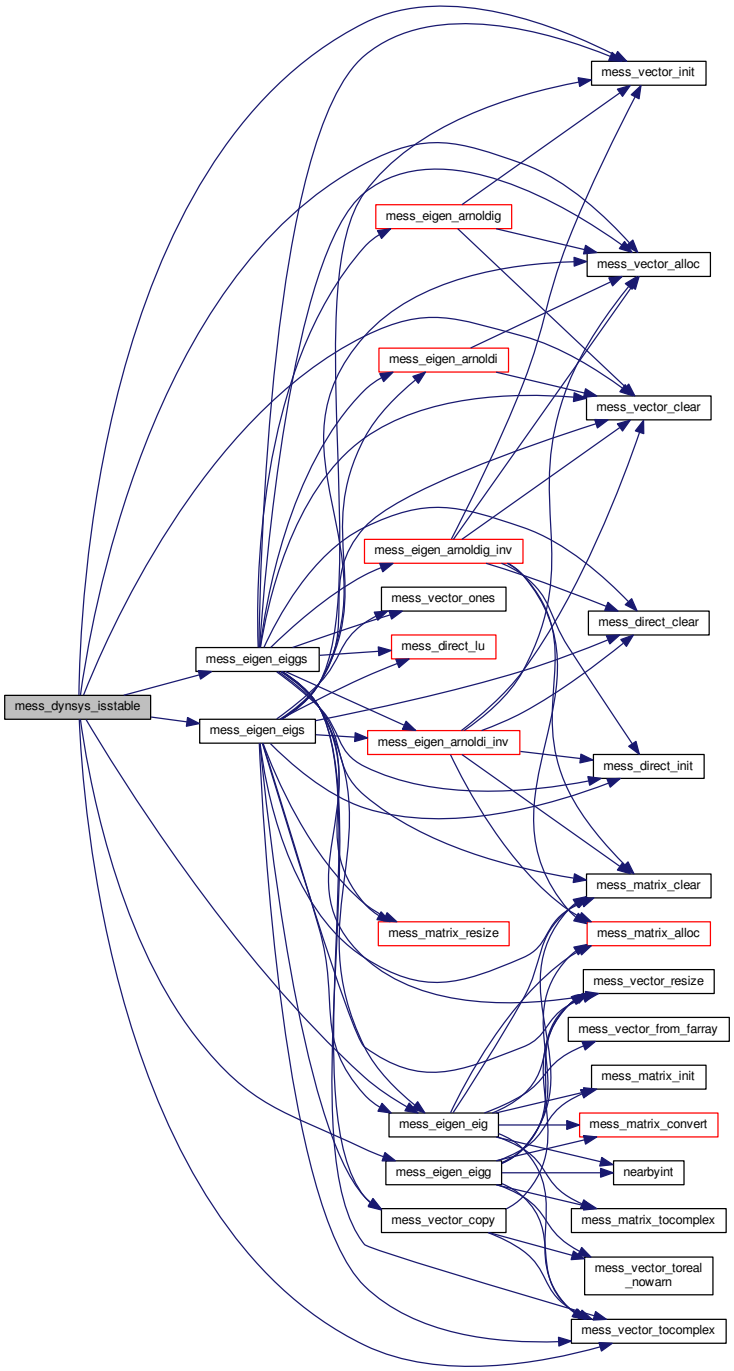
in	*sys*	input dynamical system
out	*stable*	output which is true if the system is stable, otherwise false

Returns

zero on succes or a non-zero error value

The mess_dynsys_isstable function checks if a given system is stable or not.
In case of sparse systems this is done approximately by computing some eigenvalues via an Arnoldi process.

Here is the call graph for this function:



8.31.4.10 int mess_dynsys_lti (mess_dynsys *lti*, mess_matrix *A*, mess_matrix *B*, mess_matrix *C*)

Parameters

out	*lti*	mess_dynsys object
in	*A*	input system matrix
in	*B*	input matrix

Parameters

in	*C*	input output matrix

Returns

zero on success or a non-zero error value

The mess_dynsys_lti function builds a mess_dynsys object for an LTI system defined by A, B and C:

$$\begin{array}{ll} \dot{x} & = Ax \quad +Bu \\ y & = Cx \end{array} .$$

The matrices are not copied to the object, only a pointer is used.
When the object is cleared, the matrices will be cleared too.

8.31.4.11 int mess_dynsys_lti_copy (mess_dynsys *lti*, mess_matrix *A*, mess_matrix *B*, mess_matrix *C*)

Parameters

out	*lti*	LTI system
in	*A*	input system matrix
in	*B*	input matrix
in	*C*	input output matrix

Returns

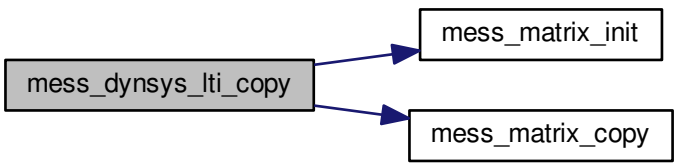
zero on success or a non-zero error value

The mess_dynsys_lti_copy function builds a mess_dynsys object as LTI system defined by the matrices A, B and C:

$$\begin{array}{ll} \dot{x} & = Ax \quad +Bu \\ y & = Cx \end{array} .$$

The matrices will be copied to the object. That means $lti \rightarrow A$ is independent from A.

Here is the call graph for this function:



8.31.4.12 int mess_dynsys_printinfo (mess_dynsys *sys*)

Parameters

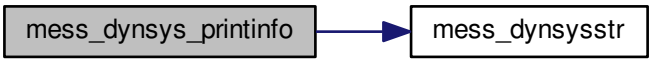
in	*sys*	input dynamical system

Returns

zero on success or a non-zero error value

The mess_dynsys_printinfo function prints basic information about a dynamical system.

Here is the call graph for this function:



8.31.4.13 int mess_dynsys_project_2nd_to_1st (mess_dynsys *sys2nd*, mess_matrix *V*, mess_matrix *W*, mess_dynsys *sys1st*)

Parameters

in	*sys2nd*	input second order system
in	*V*	input right projection matrix
in	*W*	input left projection matrix
out	*sys1st*	first order system

Returns

zero on succes or a non-zero error value

The mess_dynsys_project_2nd_to_1st function projects a second order system

$$\begin{array}{rcl} M\ddot{x}(t) + G\dot{x}(t) + \quad Kx(t) & = & Bu(t) \\ y & = & C_p x(t) \quad + C_v \dot{x}(t) \end{array}$$

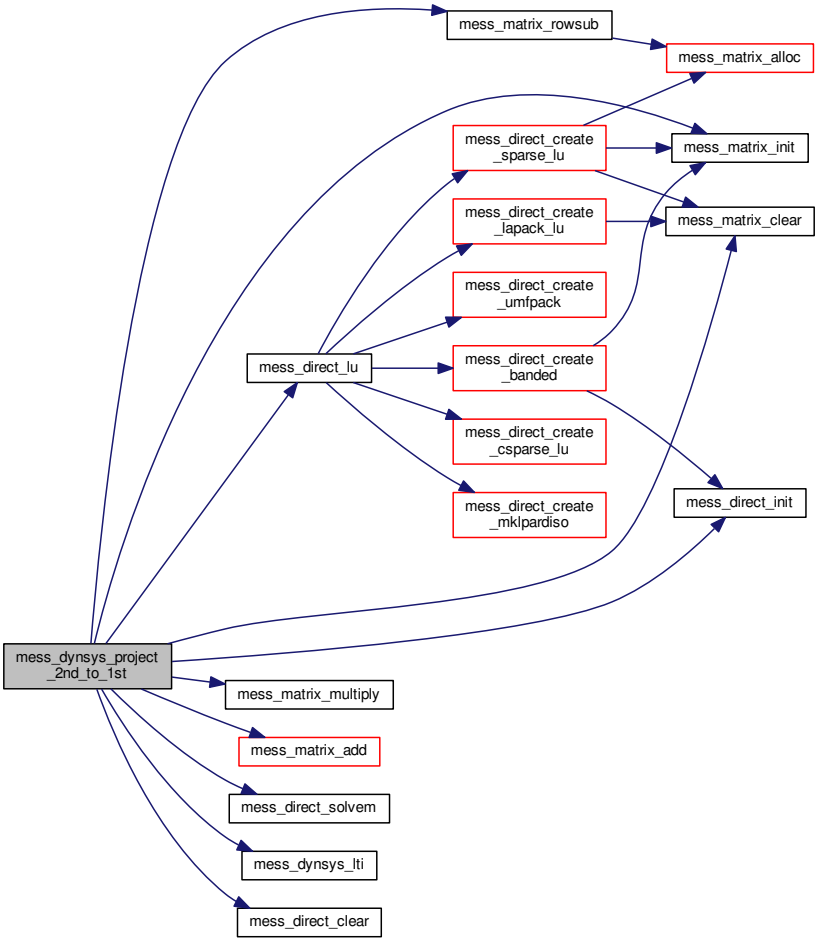
to a first order system

$$\begin{array}{rl} \dot{z} & = A_r z \quad + B_r u \\ y & = C_r x \end{array}$$

using matrices V and W:

$$\begin{array}{rcl} A_r & = & \begin{bmatrix} W_1^T & W_2^T \end{bmatrix} \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}G \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \\ B_r & = & \begin{bmatrix} W_1^T & W_2^T \end{bmatrix} \begin{bmatrix} 0 \\ B \end{bmatrix} \\ C_r & = & \begin{bmatrix} C_p & C_v \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \\ z & = & \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \end{array}.$$

Here is the call graph for this function:



8.31.4.14 int mess_dynsys_project_2nd_to_2nd (mess_dynsys *sys*, mess_matrix *V*, mess_matrix *W*, mess_dynsys *red*)

Parameters

in	*sys*	input system
in	*V*	input right projection matrix
in	*W*	input left projection matrix
out	*red*	reduced system

Returns

zero on succes or a non-zero error value

The mess_dynsys_project_2nd_to_2nd function projects a second order system

$$\begin{aligned} M\ddot{x}(t) + G\dot{x}(t) + \quad Kx(t) &= Bu(t) \\ y &= C_p x(t) \quad + C_v \dot{x}(t) \end{aligned}$$

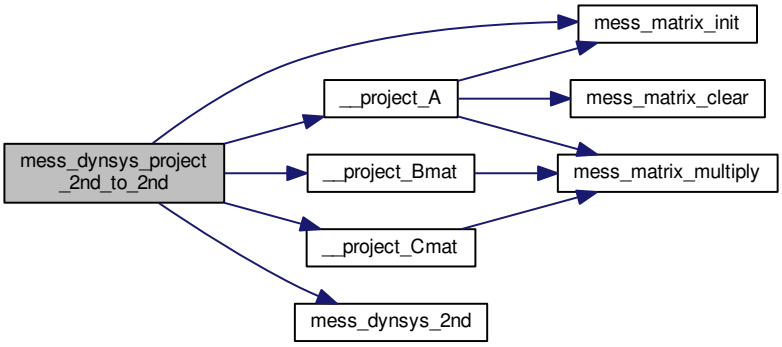
to another second order system

$$\begin{array}{rcl} M_r\ddot{x}(t) + G_r\dot{x}(t) + \quad K_r x(t) & = & B_r u(t) \\ y & = & Cp_r x(t) \quad + Cv_r \dot{x}(t) \end{array}$$

using matrices V and W:

$$\begin{array}{rcl} M_r & = & W^T M V, \\ G_r & = & W^T G V, \\ K_r & = & W^T K V, \\ B_r & = & W^T B, \\ Cp_r & = & CpV \\ Cv_r & = & CvV \end{array}$$

Here is the call graph for this function:



8.31.4.15 int mess_dynsys_project_to_glti (mess_dynsys *sys*, mess_matrix *V*, mess_matrix *W*, mess_dynsys *red*)

Parameters

in	*sys*	input system
in	*V*	input right projection matrix
in	*W*	input left projection matrix
out	*red*	reduced system

Returns

zero on succes or a non-zero error value

The mess_dynsys_project_to_lti function projects a GLTI system

$$\begin{array}{rl} E\dot{x} & = Ax \quad + Bu \\ y & = Cx \end{array}$$

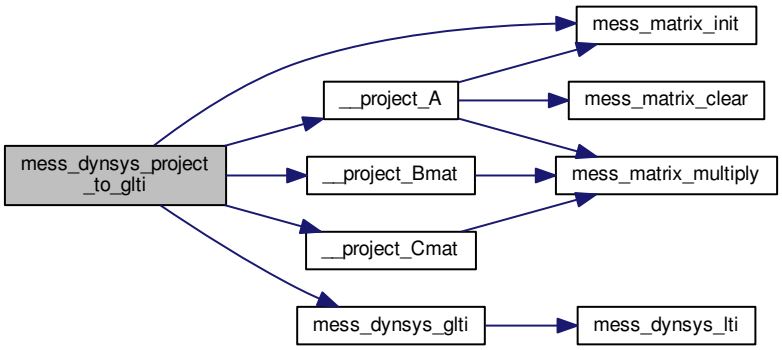
to another GLTI system

$$\begin{array}{rl} E_r\dot{x} & = A_r x \quad + B_r u \\ y & = C_r x \end{array}$$

with the aid of matrices V and W :

$$\begin{aligned} E_r &= W^T E V, \\ A_r &= W^T A V, \\ B_r &= W^T B, \\ C_r &= C V. \end{aligned}$$

Here is the call graph for this function:



8.31.4.16 int mess_dynsys_project_to_lti (mess_dynsys *sys*, mess_matrix *V*, mess_matrix *W*, mess_dynsys *red*)

Parameters

in	*sys*	input system
in	*V*	input right projection matrix
in	*W*	input left projection matrix
out	*red*	reduced system

Returns

zero on succes or a non-zero error value

The mess_dynsys_project_to_lti function projects a (G)LTI system

$$\begin{aligned} E\dot{x} &= Ax \quad + Bu \\ y &= Cx \end{aligned}$$

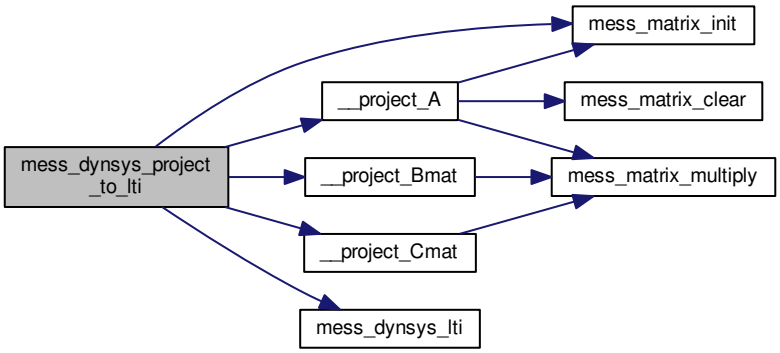
to a LTI system

$$\begin{aligned} \dot{x} &= A_r x \quad + B_r u \\ y &= C_r x \end{aligned}$$

with the aid matrices V and W:

$$\begin{aligned} A_r &= W^T A V, \\ B_r &= W^T B, \\ C_r &= C V. \end{aligned}$$

Here is the call graph for this function:



8.31.4.17 int mess_dynsys_transfer_diff (mess_int_t *nsample*, mess_matrix * *Go*, mess_matrix * *Gr*, double * *err2*, double * *rel2*, double * *errinf*, double * *relinf*, mess_vector *diffvec*)

Parameters

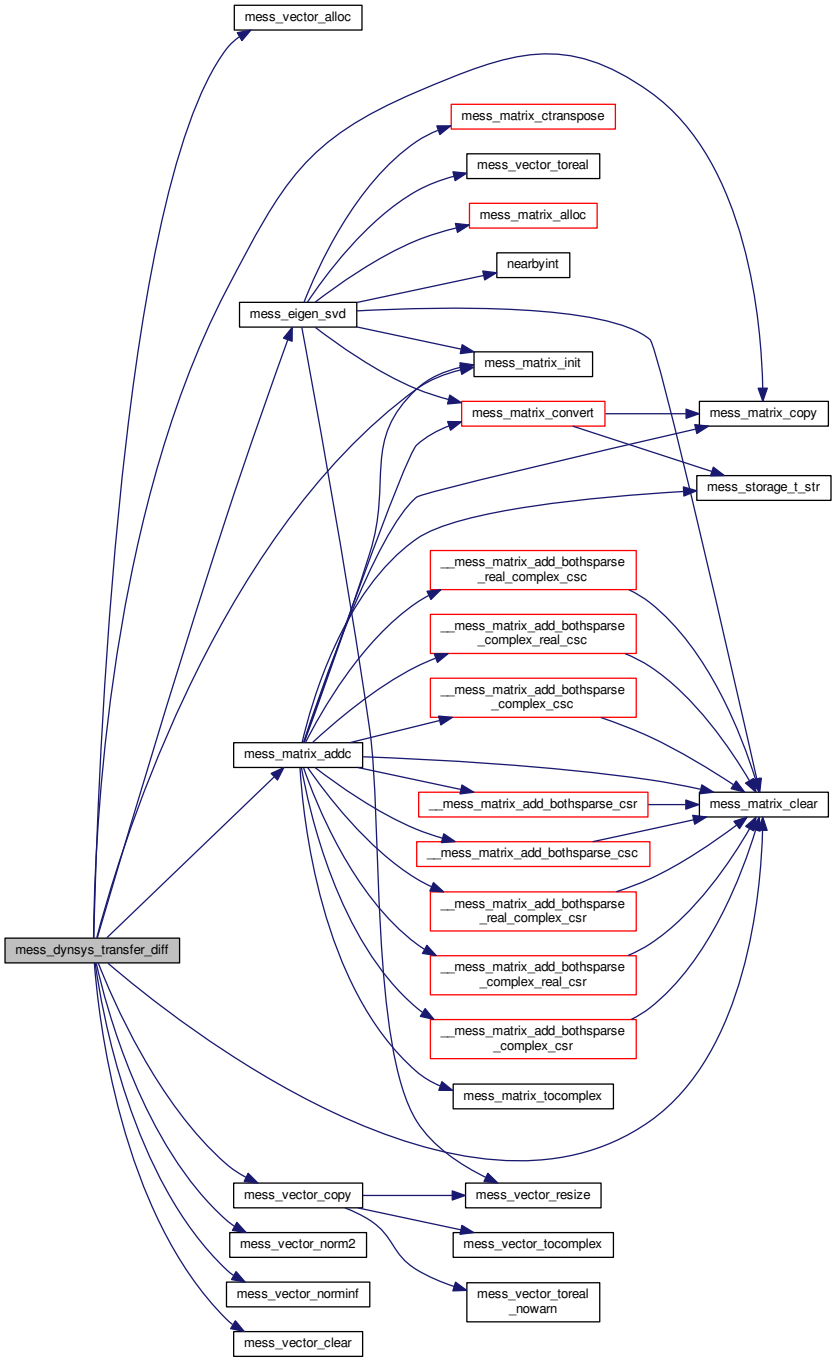
in	*nsample*	input number of samples contained in the Go and Gr array
in	*Go*	input array of G(i omega) of size nsample
in	*Gr*	input array of Gr(i omega) of size nsample
out	*err2*	two norm error
out	*rel2*	relative two norm error (w.r.t. Go)
out	*errinf*	infinity error
out	*relinf*	relative infinity error (w.r.t. Go)
out	*diffvec*	vector of the differences

Returns

zero on succes or a non-zero error value

The mess_dynsys_transfer_diff function computes various differences between two discretized transfer functions. The transfer functions are given as two array of size `nsample`.
Each array entry must be of the type mess_matrix. If $err2, rel2, errinf, relinf$ or $diffvec$ is set to `NULL`, the value will not be computed.

Here is the call graph for this function:



8.31.4.18 char* mess_dynsysstr (unsigned short *type*)

Parameters

in	*type*	input id of the type

Returns

name of the dynamical system type, otherwise "UNKNOWN"

The mess_dynsysstr function returns the name of a dynamical system type.

8.31.4.19 int mess_matrix_isstable (mess_matrix *A*, int * *stable*)

Parameters

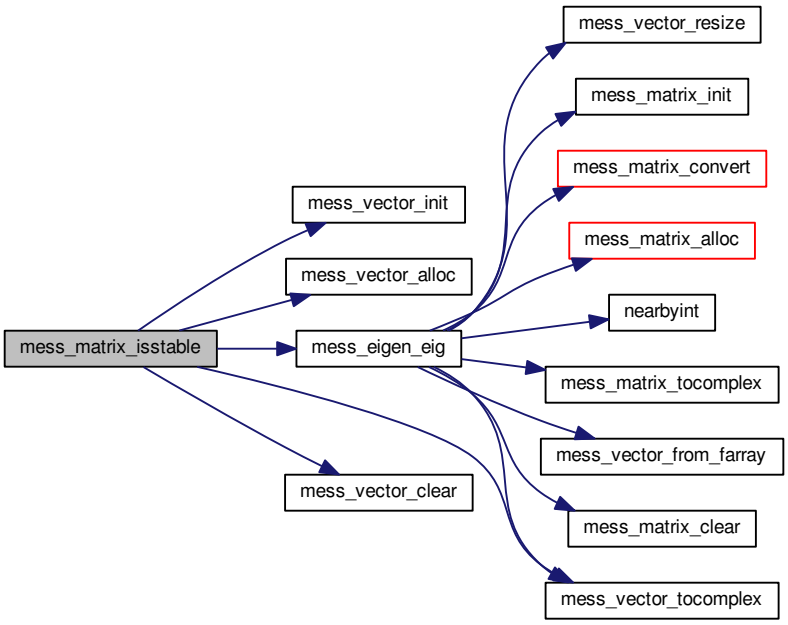
in	*A*	input matrix
out	*stable*	output which is true if A is stable, otherwise false

Returns

zero on succes or a non-zero error value

The mess_matrix_isstable function checks if a matrix is stable.
If the matrix is sparse this is done approximately by using Arnoldis method.

Here is the call graph for this function:



8.31.4.20 int mess_matrix_isstableg (mess_matrix *A*, mess_matrix *E*, int * *stable*)

Parameters

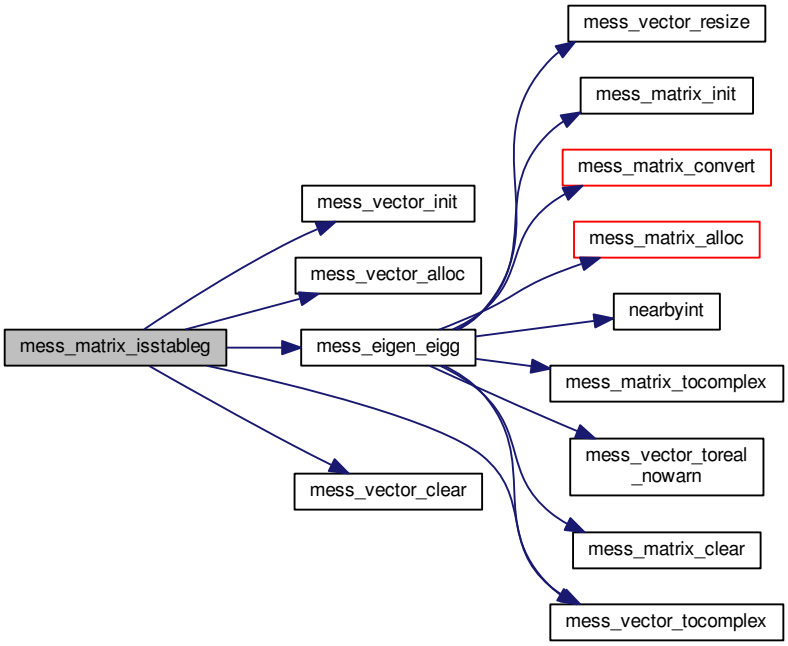
in	*A*	input first matrix
in	*E*	input second matrix
out	*stable*	output which is true if (A, E) is stable, otherwise false

Returns

zero on succes or a non-zero error value

The mess_matrix_isstableg function checks if the pair (A, E) is stable.
If it is sparse, the check is done approximately by using Arnoldis method.

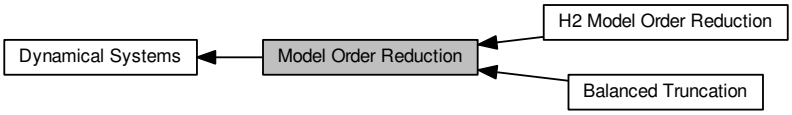
Here is the call graph for this function:



8.32 Model Order Reduction

This section contains categories reducing the the order of a GLTI system in different ways.

Collaboration diagram for Model Order Reduction:



Modules

- Balanced Truncation

  This category contains all necessary structures, definitions and functions to reduce the order of a GLTI system with the help of balanced truncation.
- H2 Model Order Reduction

  This category contains all necessary structures, definitions and functions to reduce the order of a LTI system with the help of the H_2 MOR.

8.32.1 Detailed Description

8.33 Balanced Truncation

This category contains all necessary structures, definitions and functions to reduce the order of a GLTI system with the help of balanced truncation.

Collaboration diagram for Balanced Truncation:

Data Structures

- struct mess_bt_options_st

  Options structure for balanced truncation .
- struct mess_bt_status_st

  Status structure for balanced truncation.

Macros

- #define MESS_BT_POSITION 0x01
- #define MESS_BT_VELOCITY 0x02
- #define MESS_BT_POSITION_VELOCITY 0x03
- #define MESS_BT_VELOCITY_POSITION 0x04

Typedefs

- typedef int(∗ mess_bt_chooseorder_func) (mess_vector, double ∗, mess_int_t ∗, mess_int_t, void ∗)

  Type definition for the choose order function.
- typedef struct mess_bt_options_st ∗ mess_bt_options

  Type definition for the balanced truncation options.
- typedef struct mess_bt_status_st ∗ mess_bt_status

  Type definition for the balanced truncation status.

Functions

- int mess_bt_options_init (mess_bt_options ∗opt)

  Initialize a mess_bt_options object.
- int mess_bt_options_clear (mess_bt_options ∗opt)

  Clear a mess_bt_options object.
- int mess_bt_options_print (mess_bt_options opt)

  Print a mess_bt_options object to stdout.
- const char ∗ mess_bt_getsotypestr (unsigned short so_type)

  Get the string of the so_type value.

- int mess_bt_status_init (mess_bt_status *status)

  Initialize a mess_bt_status object.
- int mess_bt_status_clear (mess_bt_status *status)

  Clear a mess_bt_status object.
- int mess_bt_status_print (mess_bt_status status)

  Print a mess_bt_status object to stdout.
- int mess_bt_chooseorder_default (mess_vector SIGMA, double *tol, mess_int_t *maxr, mess_int_t FOM, void *aux)

  Choose dimension of reduced order model adaptively for balanced truncation (BT).
- int mess_bt_chooseorder_minreal (mess_vector SIGMA, double *tol, mess_int_t *maxr, mess_int_t FOM, void *aux)

  Choose dimension of reduced order model adaptively for balanced truncation (BT) with cut of of nearly zero singular values.
- int mess_bt_lrsrm (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Low-rank square root balanced truncation for first order LTI systems.
- int mess_bt_gslrsrm (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Generalized low-rank square root balanced truncation for first order GLTI to LTI systems.
- int mess_bt_gglrsrm (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Generalized low-rank square root balanced truncation for first order GLTI to GLTI systems.
- int mess_bt_lrsrm_somor (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Low-rank square root balanced truncation for second order LTI systems to first order LTI systems.
- int mess_bt_lrsrm_somor_so (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Low-rank square root balanced truncation for second order LTI systems to second order LTI systems.
- int mess_bt_lrsrm_somor_so_gram (mess_dynsys sys, mess_matrix ZB, mess_matrix ZC, mess_bt_options btopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Low-rank square root balanced truncation for second order LTI systems to second order LTI systems (Gramian version).

8.33.1 Detailed Description

A generalized linear time invariant system

$$\begin{array}{rcll} E\dot{x}(t) & = & Ax(t) & +Bu(t) \quad , \\ y(t) & = & Cx(t) & \quad , \end{array}$$

where E is nonsingular and $A - \lambda E$ is stable, is called **balanced**, if the Controllability Gramian W_C and the Observability Gramian W_O, which are the solutions of the generalized Lyapunov Equations

$$\begin{array}{rrcl} AW_CE^T+ & EW_CA^T & = & -BB^T \quad , \\ A^T\hat{W}_OE+ & E^T\hat{W}_OA & = & -CC^T \quad , \\ & W_O & = & E^T\hat{W}_OE \quad , \end{array}$$

satisfy

$$W_C = W_O = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix}$$

with $\sigma_1 \geq \cdots \geq \sigma_n > 0$ and $\sqrt{\lambda(W_CW_O)} = \{\sigma_1, \cdots, \sigma_n\}$.

A **balanced truncation** can be computed by the Square-Root method. First, a Singular Value Decomposition (SVD) of the Gramians is computed

$$\begin{aligned} W_O &= U_O \Sigma_O U_O^T \quad , \\ W_C &= U_C \Sigma_C U_C^T \quad , \end{aligned}$$

to get the Cholesky factors of the Gramians

$$\begin{aligned} S &= \left(U_O \Sigma_O^{\frac{1}{2}} \right)^T \quad , \\ R &= \left(U_C \Sigma_C^{\frac{1}{2}} \right)^T \quad . \end{aligned}$$

Next, the SVD of SR^T is computed:

$$SR^T = U \Sigma \hat{V}^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} \begin{bmatrix} \hat{V}_1^T \\ \hat{V}_2^T \end{bmatrix} .$$

The truncation of discardable partitions $U_2, \hat{V}_2^T, \Sigma_2$ results in the reduced order model

$$\begin{aligned} W^T E V \dot{x}(t) &= W^T A V x(t) \quad + W^T B u(t) \quad , \\ y(t) &= C V x(t) \quad , \end{aligned}$$

where

$$\begin{aligned} W^T &= \Sigma_1^{-\frac{1}{2}} \hat{V}_1^T R E^{-1} \quad , \\ V &= S^T U_1 \Sigma_1^{-\frac{1}{2}} \quad . \end{aligned}$$

source: [1]

8.33.2 Macro Definition Documentation

8.33.2.1 #define MESS_BT_POSITION 0x01

Type of second order system: position.

8.33.2.2 #define MESS_BT_POSITION_VELOCITY 0x03

Type of second order system: position-velocity.

8.33.2.3 #define MESS_BT_VELOCITY 0x02

Type of second order system: velocity.

8.33.2.4 #define MESS_BT_VELOCITY_POSITION 0x04

Type of second order system: velocity-position.

8.33.3 Typedef Documentation

8.33.3.1 typedef int(∗ mess_bt_chooseorder_func) (mess_vector, double ∗, mess_int_t ∗, mess_int_t, void ∗)

To give the user the possibility to choose the order of the balanced truncation reduced model as they want, they can use their own chooser function that looks like this type definition.
In the case of success it needs to return 0. All other return values are interpreted as a failure.

8.33.3.2 typedef struct mess_bt_options_st* mess_bt_options

Type definition for better use of mess_bt_options .

8.33.3.3 typedef struct mess_bt_status_st* mess_bt_status

The mess_h2_status is a type definition for the balanced truncation status.

8.33.4 Function Documentation

8.33.4.1 int mess_bt_chooseorder_default (mess_vector *SIGMA*, double * *tol*, mess_int_t * *maxr*, mess_int_t *FOM*, void * *aux*)

Parameters

in	*SIGMA*	input vector with the HANKEL singular values
in	*tol*	input pointer to the cut off tolerance
in,out	*maxr*	pointer to the reduced order
in	*FOM*	input order of the original system (not used)
in	*aux*	input auxillary data (not used)

Returns

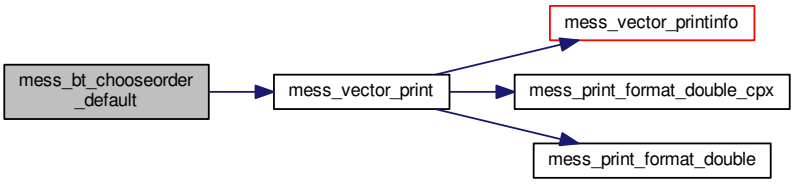
zero on success or a non-zero error value

The mess_bt_chooseorder_default function chooses the dimension of the reduced order adaptively be using the criteria

$$\|H - H_r\|_\infty < 2 \sum_{i=maxr+1}^{n} \sigma_i.$$

It is the default for the first order balanced truncation algorithms.

Here is the call graph for this function:



8.33.4.2 int mess_bt_chooseorder_minreal (mess_vector *SIGMA*, double * *tol*, mess_int_t * *maxr*, mess_int_t *FOM*, void * *aux*)

Parameters

in	*SIGMA*	input vector with the HANKEL singular values
in	*tol*	input pointer to the cut off tolerance
in,out	*maxr*	pointer to the reduced order
in	*FOM*	input order of the original system (not used)
in	*aux*	input auxillary data (not used)

Returns

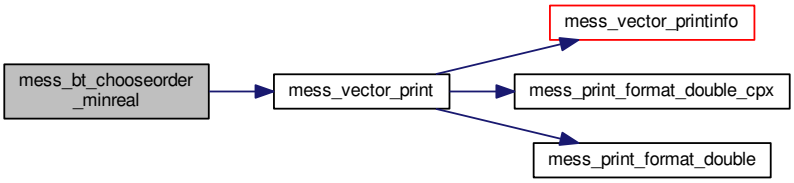
zero on success or a non-zero error value

The mess_bt_chooseorder_minreal function chooses the order of the reduced system such that

$$\frac{\sigma_{maxr}}{\sigma_1} < tol$$

is fulfilled.
It is the default for second order balanced truncation algorithms.

Here is the call graph for this function:



8.33.4.3 const char* mess_bt_getsotypestr (unsigned short *so_type*)

Parameters

in	*so_type*	input number of the type

Returns

zero on success and a string of the so_type

The mess_bt_getsotypestr function gets the string of a so_type value.

8.33.4.4 int mess_bt_gglrsrm (mess_dynsys *sys*, mess_bt_options *btopt*, mess_options *adiopt*, mess_matrix *V*, mess_matrix *W*, mess_bt_status *status*)

Parameters

in	*sys*	generalized input system
in	*btopt*	input balanced truncation options

Parameters

in	*adiopt*	input options for the underlying ADI Solver
out	*V*	right projection matrix
out	*W*	left projection matrix
out	*status*	status information

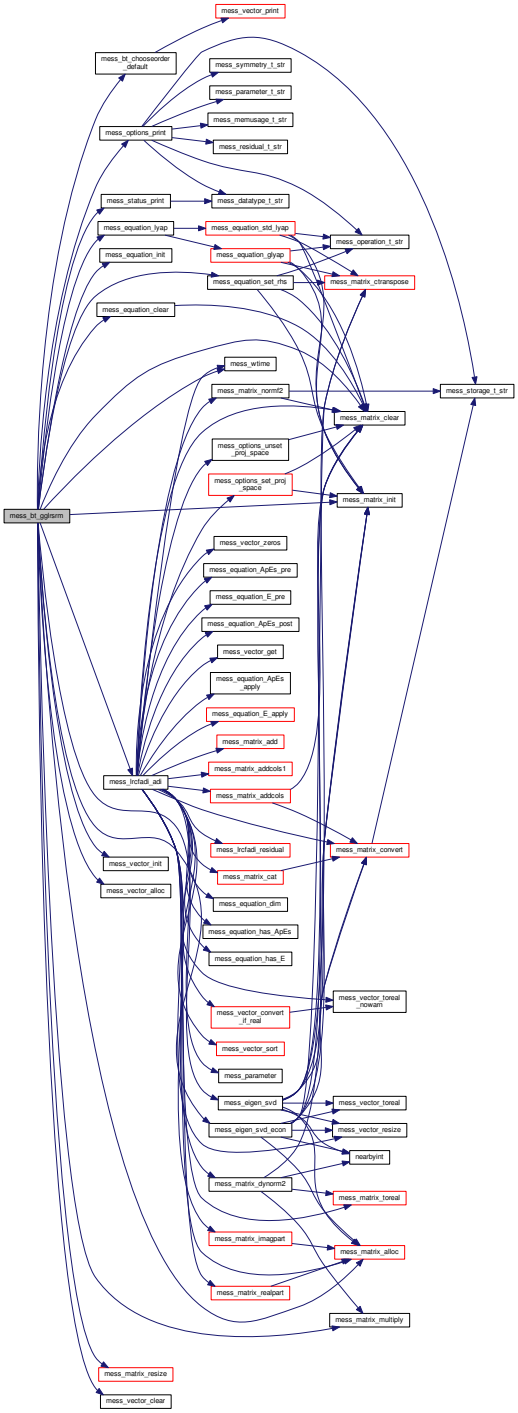
Returns

zero on success or a non-zero error value

The mess_bt_gglrsrm function computes the left and the right projection matrix for a balanced truncation reduced order model of a generalized state space system to a generalized state space system.
In order to get the reduced order GLTI model from the output, you have to call the mess_dynsys_project_to_glti function with the computed V and W.
In the options structure the **tol** value is given to set the cut off tolerance for the Hankel singular values. Also the **rdim** parameter is given to set the maximum wanted dimension of the reduced oder model.
If you do not want to choose the reduced order in an adaptive way you can specify a **chooseorder** function to decide it in your own way. The default choose order function is mess_bt_chooseorder_default.

The algorithm is based on Algorithm 7.3 in [28] .

Here is the call graph for this function:



8.33.4.5 int mess_bt_gslrsrm (mess_dynsys *sys*, mess_bt_options *btopt*, mess_options *adiopt*, mess_matrix *V*, mess_matrix *W*, mess_bt_status *status*)

Parameters

in	*sys*	generalized input system
in	*btopt*	input balanced truncation options

Parameters

in	*adiopt*	input options for the underlying ADI Solver
out	*V*	right projection matrix
out	*W*	left projection matrix
out	*status*	status information

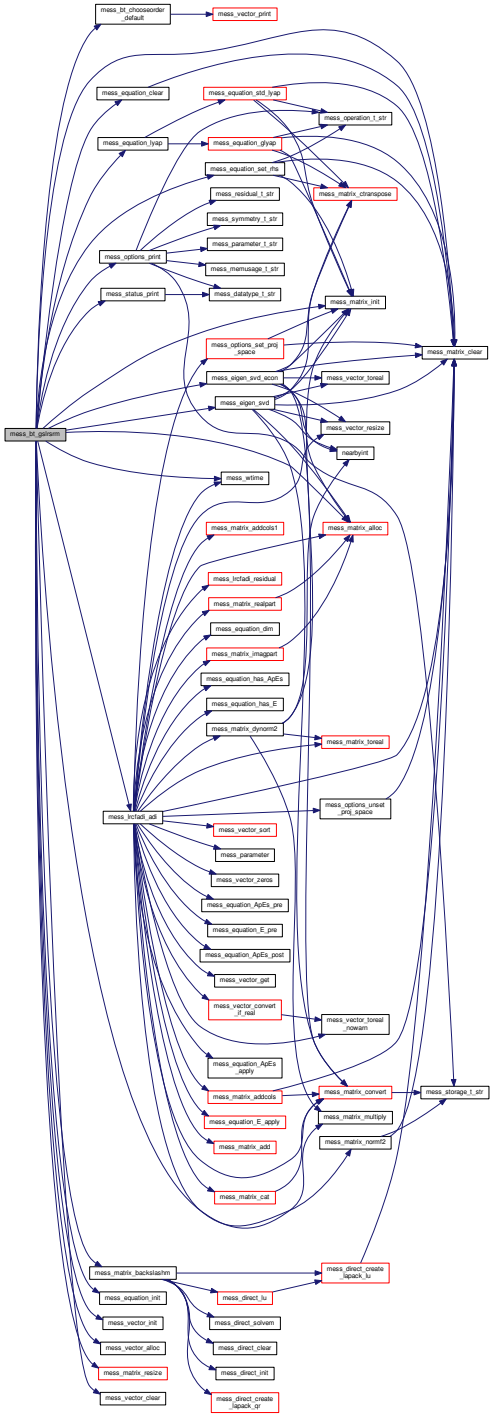
Returns

zero on success or a non-zero error value

The mess_bt_gslrsrm function computes the left and the right projection matrix for a balanced truncation reduced order model of a generalized state space system to a standard state space system.
In order to get the reduced order LTI model from the output, you have to call the mess_dynsys_project_to_lti function with the computed V and W.
In the mess_options::adi_res2_tol is given to set the cut off tolerance for the Hankel singular values. Also the **rdim** parameter is given to set the maximum wanted dimension of the reduced oder model.
If you do not want to choose the reduced order in a adaptive way so can specify a `chooseorder` function to decide it in your own way. The default choose order function is mess_bt_chooseorder_default.

The algorithm is based on Algorithm 7.2 in [28] .

Here is the call graph for this function:



8.33.4.6 int mess_bt_lrsrm (mess_dynsys *sys*, mess_bt_options *btopt*, mess_options *adiopt*, mess_matrix *V*, mess_matrix *W*, mess_bt_status *status*)

Parameters

in	*sys*	generalized input system
in	*btopt*	input balanced truncation options

Parameters

in	*adiopt*	input options for the underlying ADI Solver
out	*V*	right projection matrix
out	*W*	left projection matrix
out	*status*	status information

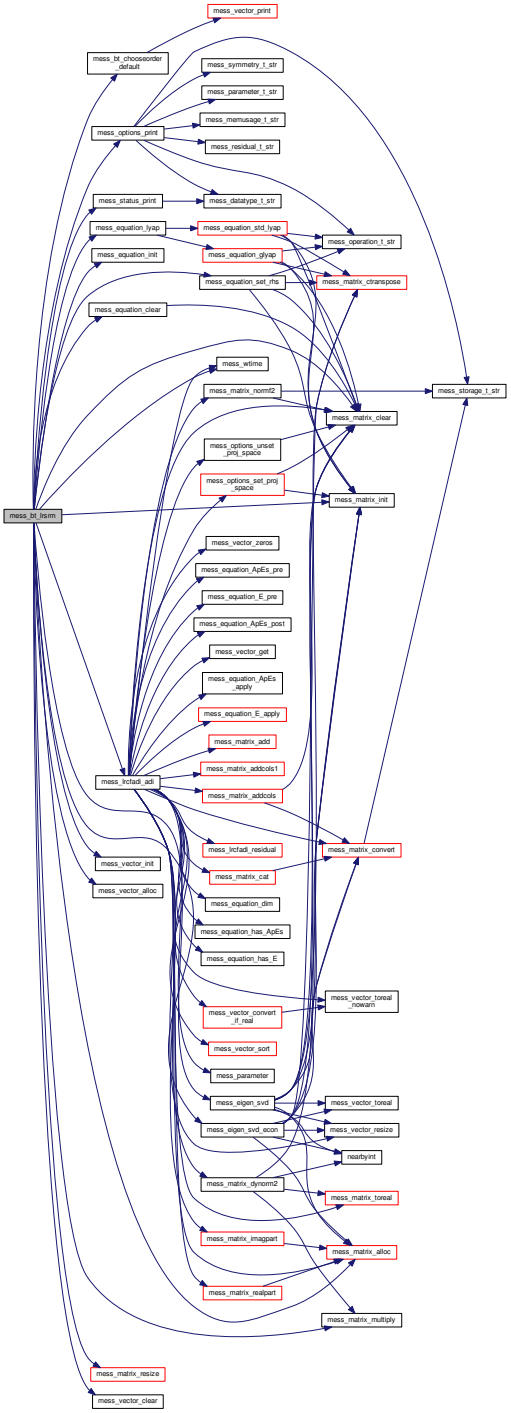
Returns

zero on success or a non-zero error value

The mess_bt_lrsrm function computes the left and the right projection matrix for a balanced truncation reduced order model.
In order to get the reduced order model from the output, you have to call the mess_dynsys_project_to_lti function with the computed V and W.
In the options structure the **tol** value is given to set the cut off tolerance for the Hankel singular values. Also the **rdim** parameter is given to set the maximum wanted dimension of the reduced oder model.
If you do not want to choose the reduced order in a adaptive way so can specify a **chooseorder** function to decide it in your own way. The default choose order function is mess_bt_chooseorder_default.

The algorithm is based on Algorithm 7.1 in [28] .

Here is the call graph for this function:



8.33.4.7 int mess_bt_lrsrm_somor (mess_dynsys *sys*, mess_bt_options *btopt*, mess_options *adiopt*, mess_matrix *V*, mess_matrix *W*, mess_bt_status *status*)

Parameters

in	*sys*	generalized input system
in	*btopt*	input balanced truncation options

Parameters

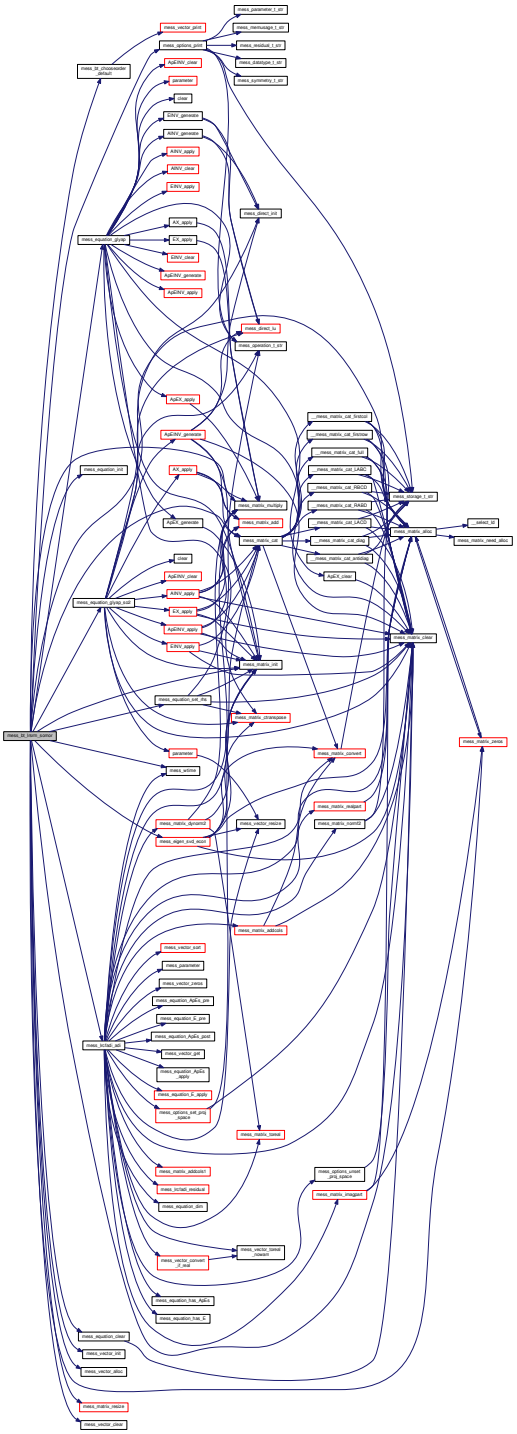
in	*adiopt*	input options for the underlying ADI Solver
out	*V*	right projection matrix
out	*W*	left projection matrix
out	*status*	status information

Returns

zero on success or a non-zero error value

The mess_bt_lrsrm_somor function computes the left and the right projection matrix for a balanced truncation reduced order model.
In order to get the reduced order model from the output, you have to call the mess_dynsys_project_2nd_to_1st function with the computed V and W.
In the options structure the **tol** value is given to set the cut off tolerance for the Hankel singular values. Also the **rdim** parameter is given to set the maximum wanted dimension of the reduced oder model.
If you do not want to choose the reduced order in a adaptive way so can specify a **chooseorder** function to decide it in your own way. The default choose order function is mess_bt_chooseorder_default.

Here is the call graph for this function:



8.33.4.8 int mess_bt_lrsrm_somor_so (mess_dynsys *sys*, mess_bt_options *btopt*, mess_options *adiopt*, mess_matrix *V*, mess_matrix *W*, mess_bt_status *status*)

Parameters

in	*sys*	generalized input system
in	*btopt*	input balanced truncation options

Parameters

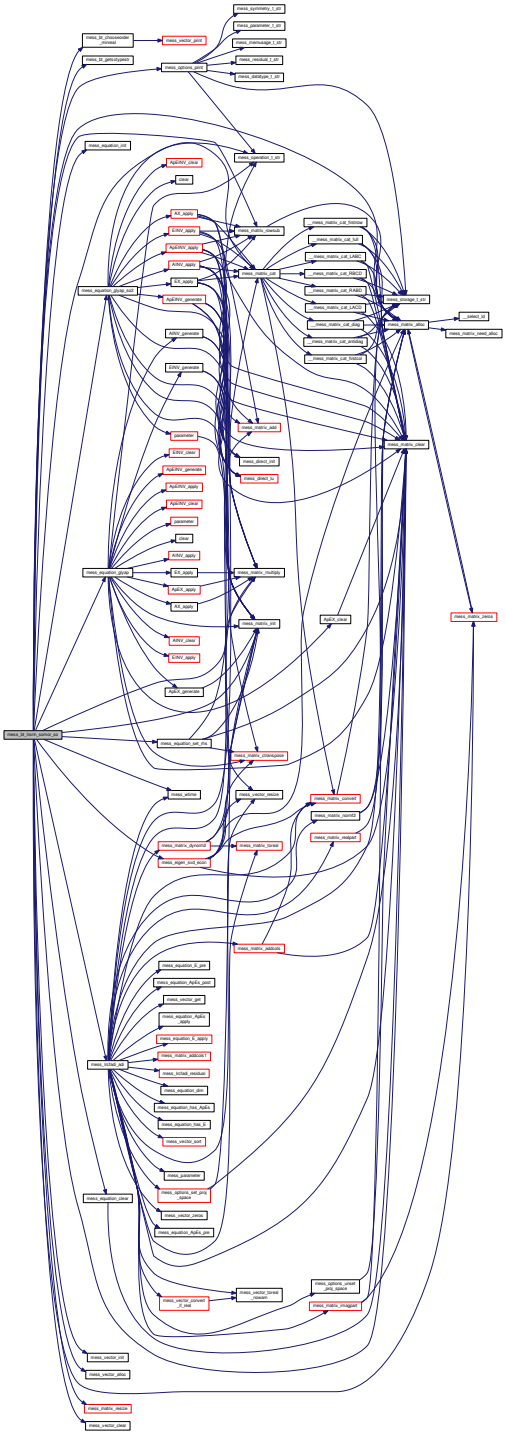
in	*adiopt*	input options for the underlying ADI Solver
out	*V*	right projection matrix
out	*W*	left projection matrix
out	*status*	status information

Returns

zero on success or a non-zero error value

The mess_bt_lrsrm_somor_so function computes the left and the right projection matrix for a balanced truncation reduced order model.
In order to get the reduced order model from the output, you have to call the mess_dynsys_project_2nd_to_2nd function with the computed V and W.
In the options structure the **tol** value is given to set the cut off tolerance for the Hankel singular values. Also the **rdim** parameter is given to set the maximum wanted dimension of the reduced oder model.
If you do not want to choose the reduced order in a adaptive way so can specify a **chooseorder** function to decide it in your own way. The default choose order function is mess_bt_chooseorder_minreal.

Here is the call graph for this function:



8.33.4.9 int mess_bt_lrsrm_somor_so_gram (mess_dynsys *sys*, mess_matrix *ZB*, mess_matrix *ZC*, mess_bt_options *btopt*, mess_matrix *V*, mess_matrix *W*, mess_bt_status *status*)

Parameters

in	*sys*	second order input system
in	*ZB*	input controllability gramian

Parameters

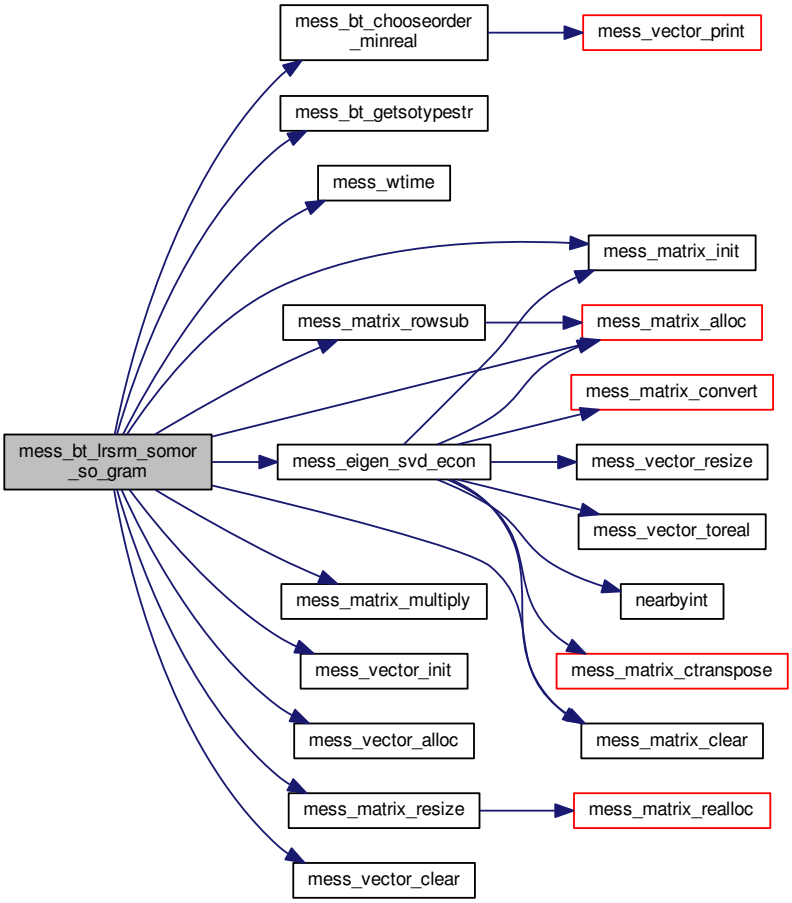
in	*ZC*	input observability gramian
in	*btopt*	input balanced truncation options
out	*V*	right projection matrix
out	*W*	left projection matrix
out	*status*	status information

Returns

zero on succes or a non-zero error value

The mess_bt_lrsrm_somor_so function computes the left and the right projection matrix for a balanced truncation reduced order model. In order to get the reduced order model from the output, you have to call the mess_dynsys↩_project_2nd_to_2nd function with the computed V and W.
In the options structure the **tol** value is given to set the cut off tolerance for the Hankel singular values. Also the **rdim** parameter is given to set the maximum wanted dimension of the reduced oder model. If you do not want to choose the reduced order in a adaptive way so can specify a **chooseorder** function to decide it in your own way. The default choose order function is mess_bt_chooseorder_minreal.

Here is the call graph for this function:



8.33.4.10 int mess_bt_options_clear (mess_bt_options * *opt*)

Parameters

in	*opt*	input pointer to the object

Returns

always zero

The mess_bt_options_clear function clears a mess_bt_options object.

8.33.4.11 int mess_bt_options_init (mess_bt_options * *opt*)

Parameters

in,out	*opt*	pointer to the object

Returns

always zero

The mess_bt_options_init function initializes a mess_bt_options object and set default values.

8.33.4.12 int mess_bt_options_print (mess_bt_options *opt*)

Parameters

in	*opt*	input mess_bt_options object

Returns

zero on success or a non-zero error value

The mess_bt_options_print function prints a mess_bt_options object to stdout.

Here is the call graph for this function:

8.33.4.13 int mess_bt_status_clear (mess_bt_status * *status*)

Parameters

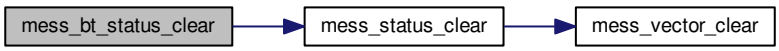
in	*status*	input pointer to the mess_bt_status object

Returns

always zero

The mess_bt_status_clear function clears a mess_bt_status object.

Here is the call graph for this function:



8.33.4.14 int mess_bt_status_init (mess_bt_status * *status*)

Parameters

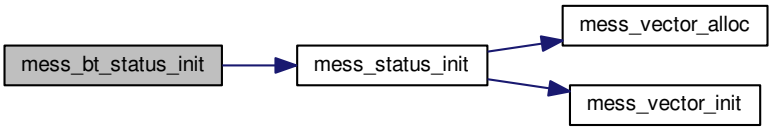
in,out	*status*	pointer to the mess_bt_status object

Returns

always zero

The mess_bt_status_init function initializes a mess_bt_status object.

Here is the call graph for this function:



8.33.4.15 int mess_bt_status_print (mess_bt_status *status*)

Parameters

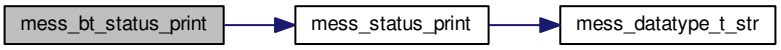
in	*status*	input mess_bt_status object to print

Returns

zero on success or a non-zero error value

The mess_bt_status_print function prints a mess_bt_status object to stdout.

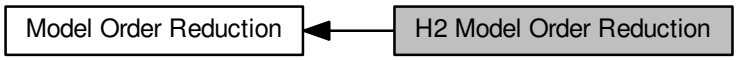
Here is the call graph for this function:



8.34 H2 Model Order Reduction

This category contains all necessary structures, definitions and functions to reduce the order of a LTI system with the help of the H_2 MOR.

Collaboration diagram for H2 Model Order Reduction:



Files

- file irka_common.c

  Common functions for IRKA.

Data Structures

- struct mess_h2_options_st

  Structure to store $\mathcal{H}_2$ options.
- struct mess_h2_status_st

  Structure to store $\mathcal{H}_2$ status.
- struct VWjob

Macros

- #define MESS_H2NORM_NO 0
- #define MESS_H2NORM_FULL 1
- #define MESS_H2NORM_UPDATE 2
- #define AR_W_VA 1

Typedefs

- typedef struct mess_h2_options_st $*$ mess_h2_options

  Type definition for the $\mathcal{H}_2$ options.
- typedef struct mess_h2_status_st $*$ mess_h2_status

  Type definition for the $\mathcal{H}_2$ status.

Functions

- int mess_h2_norm_internal (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix E, double *norm)

  Compute the $\mathcal{H}_2$-norm of a LTI System given as matrices.
- int mess_h2_norm (mess_dynsys LTI, double *norm)

  Compute the $\mathcal{H}_2$-norm of a LTI system.
- int mess_h2_error_internal (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix E, mess_matrix Ar, mess_matrix Br, mess_matrix Cr, mess_matrix Er, double *norm)

  Compute the $\mathcal{H}_2$ error of two LTI systems.
- int mess_h2_error (mess_dynsys ltiA, mess_dynsys ltiB, double *err)

  Compute the $\mathcal{H}_2$-error between two LTI systems.
- int mess_h2_irka_init (mess_matrix A, mess_direct Asolver, mess_matrix E, mess_int_t *r0, mess_int_t kp, mess_int_t km, mess_vector sigma)

  Initialize parameters for IRKA.
- int mess_h2_irka_biorth (mess_dynsys orig, mess_vector sigma, mess_h2_options opt, mess_dynsys reduced, mess_matrix V, mess_matrix W, mess_h2_status status)

  Compute a $\mathcal{H}_2$ reduced model via the IRKA algorithm (biorthonormal version).
- int mess_h2_tsia (mess_dynsys orig, mess_vector sigmaEX, mess_h2_options opt, mess_dynsys redu, mess_matrix V, mess_matrix W, mess_h2_status status)

  Compute a $\mathcal{H}_2$ reduced model via the TSIA algorithm.
- int mess_h2_tsiag (mess_dynsys orig, mess_vector sigmaEX, mess_h2_options opt, mess_dynsys redu, mess_matrix V, mess_matrix W, mess_h2_status status)

  Compute a $\mathcal{H}_2$ reduced model via the TSIA algorithm.
- int mess_h2_options_init (mess_h2_options *opt)

  Initialize a mess_h2_options object.
- int mess_h2_options_clear (mess_h2_options *opt)

  Clean up a mess_h2_options object.
- int mess_h2_options_print (mess_h2_options opt)

  Print a mess_h2_options object to screen.
- int mess_h2_status_init (mess_h2_status *status)

  Initialize a mess_h2_status object.
- int mess_h2_status_clear (mess_h2_status *status)

  Clean up a mess_h2_status object.
- int mess_h2_status_print (mess_h2_status status)

  Print a mess_h2_status object.
- int mess_h2_status_printfull (mess_h2_status status)

  Print a mess_h2_status object (full output).
- int mess_h2_status_write_mfile (mess_h2_status status, const char *filename)

  Write a mfile to plot in MATLAB .
- static int __project_A (mess_matrix W, mess_matrix A, mess_matrix V, mess_matrix Ar)

  Project A with W and V.
- static int __project_B (mess_matrix W, mess_vector b, mess_vector br)

  Project b.
- static int __project_Bmat (mess_matrix W, mess_matrix B, mess_matrix Br)

  Project B (matrix version).
- static int __project_C (mess_matrix V, mess_vector c, mess_vector cr)

  Project c.
- static int __project_Cmat (mess_matrix V, mess_matrix C, mess_matrix Cr)

  Project C (matrix version).
- static int __constructVW_seq (mess_matrix A, mess_direct Abase, mess_matrix E, mess_vector b, mess_vector c, mess_vector sigma, mess_matrix V, mess_matrix W)

  Construct V and W for IRKA (sequential).

- static int __constructVW_sec (mess_matrix A, mess_direct Abase, mess_matrix E, mess_vector b, mess_vector c, mess_vector sigma, mess_matrix V, mess_matrix W)

  Construct V and W for IRKA (OpenMP version).
- static void VW_worker (void *data)
- static int __constructVW_pre (mess_matrix A, mess_direct Abase, mess_matrix E, mess_vector b, mess_vector c, mess_vector sigma, mess_matrix V, mess_matrix W)

  Construct V and W for IRKA (OpenMP Preprocessing).
- static int __constructVW_pthread (mess_matrix A, mess_direct Abase, mess_matrix E, mess_vector b, mess_vector c, mess_vector sigma, mess_matrix V, mess_matrix W)

  Construct V and W for IRKA (OpenMP Preprocessing).
- static char * get_char (char *c1, char *c2)
- static int __constructVW (mess_matrix A, mess_direct Abase, mess_matrix E, mess_vector b, mess_vector c, mess_vector sigma, mess_matrix V, mess_matrix W)

  Construct V and W for IRKA (wrapper).
- static int __constructVWmat (mess_matrix A, mess_direct Abase, mess_matrix E, mess_matrix B, mess_matrix C, mess_vector sigma, mess_matrix V, mess_matrix W)

  Construct V and W for IRKA (wrapper, matrix inputs).

8.34.1 Detailed Description

A LTI system Σ of dimension n

$$\begin{array}{rcll} \dot{x}(t) & = & Ax(t) & +Bu(t) \quad , \\ y(t) & = & Cx(t) & \quad , \end{array}$$

with $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^n, C \in \mathbb{R}^{1\times n}$ is given with a transfer function

$$H(s) = C(sI - A)^{-1}B$$

and a reduced model Σ_r of dimension $n_r \ll n$

$$\begin{array}{rcll} \dot{x}_r(t) & = & A_r x(t) & +B_r u(t) \quad , \\ y_r(t) & = & C_r x(t) & \quad , \end{array}$$

with $A_r \in \mathbb{R}^{n_r\times n_r}, B \in \mathbb{R}^{n_r}, C \in \mathbb{R}^{1\times n_r}$ is searched with a transfer function

$$H_r(s) = C_r(sI - A_r)^-1B_r$$

fulfilling

$$\min \|H - H_r\|_{H_2}.$$

To solve this problem a projection to a reduced subspace is used:

$$\begin{array}{rcll} A_r & = & W^T AV & , \\ B_r & = & W^T B & , \\ C_r & = & CV & . \end{array}$$

The choice of $\mathcal{W} = span(W)$ and $\mathcal{V} = span(V)$, where $W^T V = I$ is crucial.

There are two types of H_2 MOR implemented:

- **IRKA** (iterative rational Krylov algorithm)
  This algorithm creates the demanded subspaces with the aid of the Krylov subspace method.
  The definition of the minimum will be weakened by the definition of a lokal minimum:

$$\exists \epsilon > 0 : \|H - H_r\|_{H_2} \leq \|H - \tilde{H}_r^\epsilon\|_{H_2} \quad \forall \|H_r - \tilde{H}_r^\epsilon\|_{H_2} \leq C\epsilon.$$

If H_r is such a local minimum of H, then it holds

$$H(-\mu_i) = H_r(-\mu_i), H'(-\mu_i) = H_r'(-\mu_i)$$

for all poles μ_i of H_r.

In pseudocode, the classical IRKA algorithm looks like

Input: $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^n, C \in \mathbb{R}^{1\times n}, \mu^{(0)} = \{\mu_1, \cdots, \mu_r\}$
Output: $A_r \in \mathbb{R}^{n_r\times n_r}, B \in \mathbb{R}^{n_r}, C \in \mathbb{R}^{1\times n_r}$ with a local minimum H_r

1. $k \leftarrow 0$
2. Determine $\mathcal{V} = span(V) = \{(\mu_1 I - A)^{-1}B, \cdots, (\mu_r I - A)^{-1}B\}$
3. Determine $\mathcal{W} = span(W) = \{(\mu_1 I - A)^{-T}C^T, \cdots, (\mu_r I - A)^{-T}C^T\}$
4. $W = W(W^TV)^{-T}$ such that $W^TV = I$
5. WHILE $\|\mu^{k+1} - \mu^k\| > tol$ DO
   (a) $A_r = W^TAV$
   (b) $\mu_l^k = \lambda_l(A_r)$
   (c) Determine $\mathcal{V} = span(V) = \{(\mu_1 I - A)^{-1}B, \cdots, (\mu_r I - A)^{-1}B\}$
   (d) Determine $\mathcal{W} = span(W) = \{(\mu_1 I - A)^{-T}C^T, \cdots, (\mu_r I - A)^{-T}C^T\}$
   (e) $W = W(W^TV)^{-T}$
   (f) $k \leftarrow k + 1$
6. $A_r = W^TAV, B_r = W^TB, C = CV$

source: [16]

- **TSIA** (two-sided iteration algorithm)
  This algorithm creates the demanded subspaces with the aid of the solution of two Sylvester Equations.
  The pseudocode of the two-sided iteration algorithm looks like
  Input: $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^n, C \in \mathbb{R}^{1\times n}$ and initial solution $A_r \in \mathbb{R}^{n_r\times n_r}, B \in \mathbb{R}^{n_r}, C \in \mathbb{R}^{1\times n_r}$
  Output: $A_r \in \mathbb{R}^{n_r\times n_r}, B \in \mathbb{R}^{n_r}, C \in \mathbb{R}^{1\times n_r}$

  1. FOR $i = 1, \cdots$ DO
     (a) Determine X_k and Y_k by solving

     $$AX_k + X_kA_r^T + BB_r^T = 0$$

     $$A^TY_k + Y_kA_r - C^TC_r = 0$$

     (b) $V = X_k, W = Y_k$
     (c) $[V, W] = Biorthogonalization(V, W)$
     (d) $A_r = W^TAV, B_r = W^TB, C_r = CV$
     END FOR

  source: [30]

8.34.2 Macro Definition Documentation

8.34.2.1 #define AR_W_VA 1

8.34.2.2 #define MESS_H2NORM_FULL 1

Calculate full $\mathcal{H}_2$ norm.

8.34.2.3 #define MESS_H2NORM_NO 0

Do not calculate $\mathcal{H}_2$ norm.

8.34.2.4 #define MESS_H2NORM_UPDATE 2

Calculate $\mathcal{H}_2$ norm with an update.

8.34.3 Typedef Documentation

8.34.3.1 typedef struct mess_h2_options_st* mess_h2_options

Type definition for better use of mess_h2_options_st .

8.34.3.2 typedef struct mess_h2_status_st* mess_h2_status

The mess_h2_status is a type definition for the $\mathcal{H}_2$ status.

8.34.4 Function Documentation

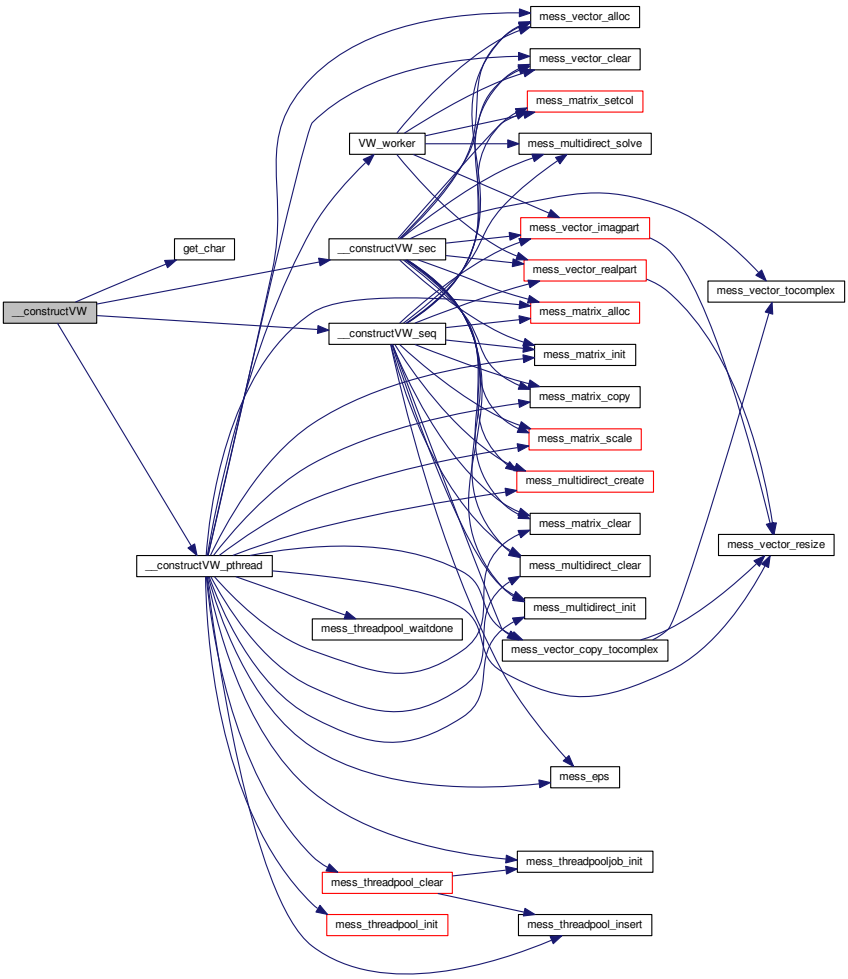
8.34.4.1 static int __constructVW (mess_matrix *A*, mess_direct *Abase*, mess_matrix *E*, mess_vector *b*, mess_vector *c*, mess_vector *sigma*, mess_matrix *V*, mess_matrix *W*) [static]

Parameters

in	*A*	input system matrix
in	*Abase*	input direct solver object
in	*E*	input mass matrix
in	*b*	input vector
out	*c*	output vector
in	*sigma*	input set of interpolation points
out	*V*	output span
out	*W*	output span

The __constructVW function contructs V and W for IRKA (wrapper).

Here is the call graph for this function:



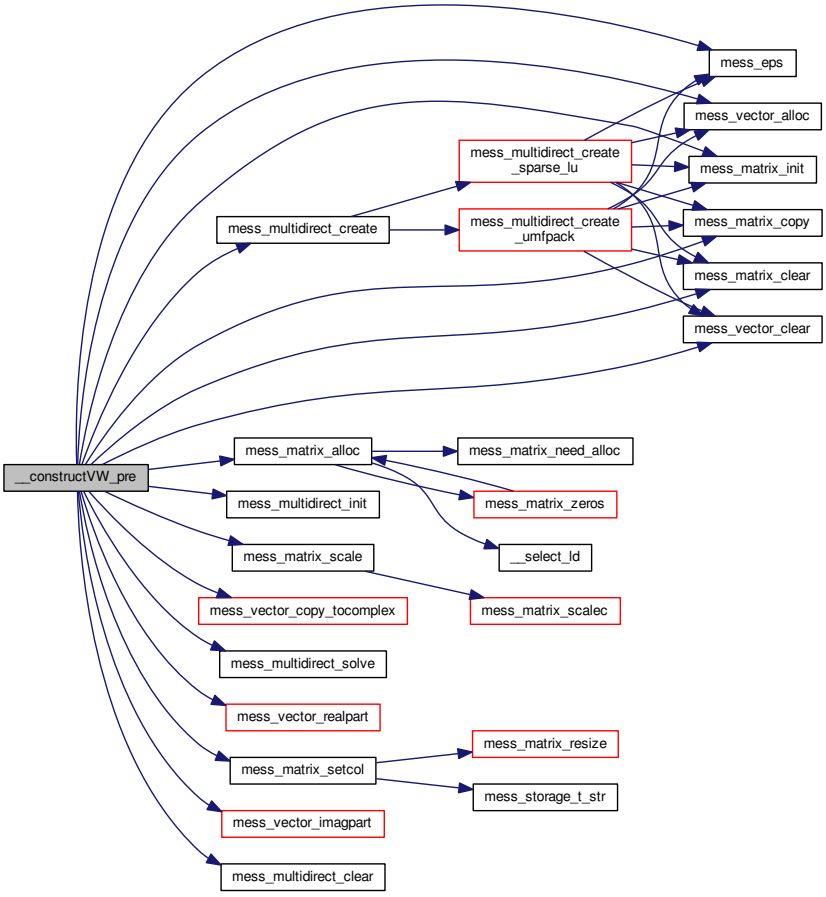
8.34.4.2 static int __constructVW_pre (mess_matrix *A*, mess_direct *Abase*, mess_matrix *E*, mess_vector *b*, mess_vector *c*, mess_vector *sigma*, mess_matrix *V*, mess_matrix *W*) [static]

Parameters

in	*A*	input system matrix
in	*Abase*	input direct solver object
in	*E*	input mass matrix
in	*b*	input vector
out	*c*	output vector
in	*sigma*	input set of interpolation points
out	*V*	output span
out	*W*	output span

The __constructVW function constructs the span V and W for IRKA (OpenMP Preprocessing).

Here is the call graph for this function:



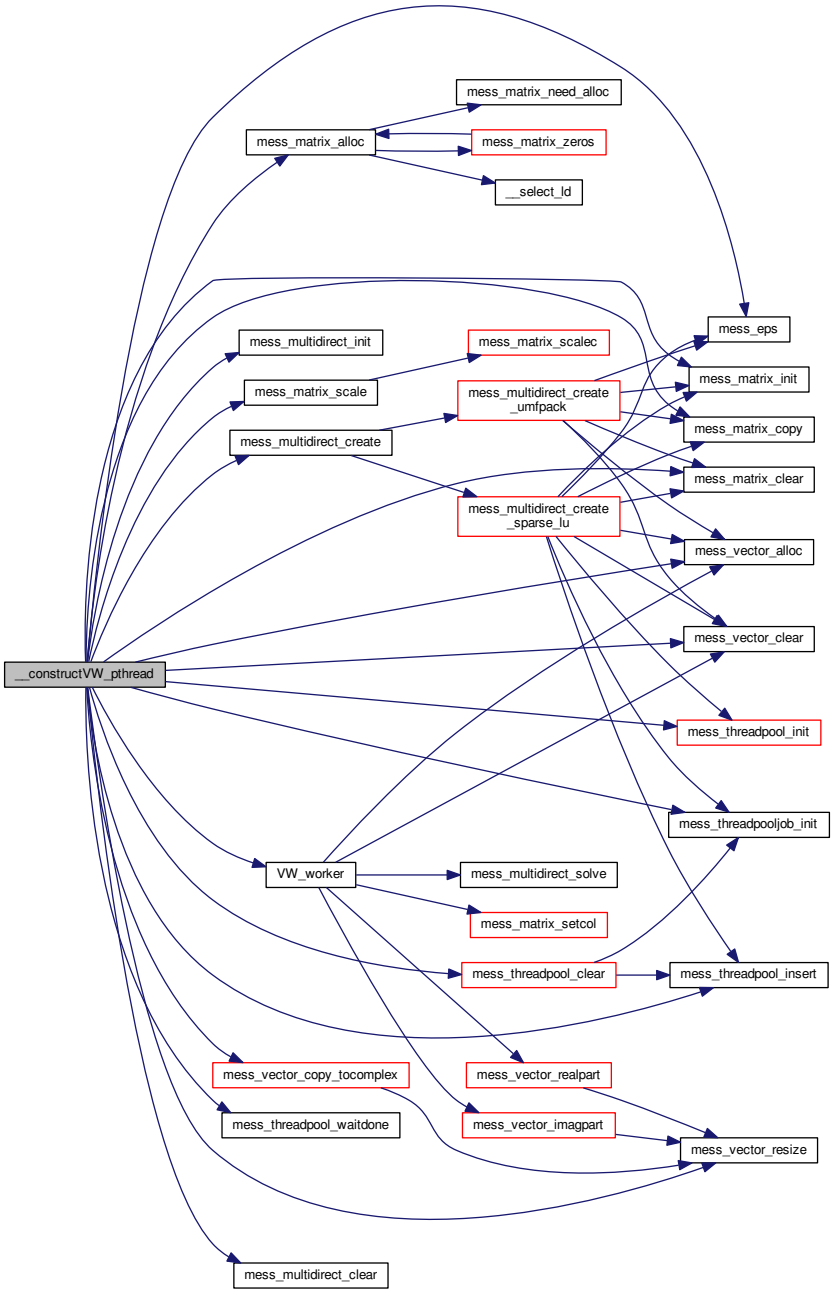
8.34.4.3 static int __constructVW_pthread (mess_matrix *A*, mess_direct *Abase*, mess_matrix *E*, mess_vector *b*, mess_vector *c*, mess_vector *sigma*, mess_matrix *V*, mess_matrix *W*) [static]

Parameters

in	*A*	input system matrix
in	*Abase*	input direct solver object
in	*E*	input mass matrix
in	*b*	input vector
out	*c*	output vector
in	*sigma*	input set of interpolation points
out	*V*	output span
out	*W*	output span

The __constructVW function constructs the span V and W for IRKA (PThread enabled version).

Here is the call graph for this function:



8.34.4.4 static int __constructVW_sec (mess_matrix *A*, mess_direct *Abase*, mess_matrix *E*, mess_vector *b*, mess_vector *c*, mess_vector *sigma*, mess_matrix *V*, mess_matrix *W*) [static]

Parameters

in	*A*	input system matrix
in	*Abase*	input direct solver object
in	*E*	input mass matrix
in	*b*	input vector

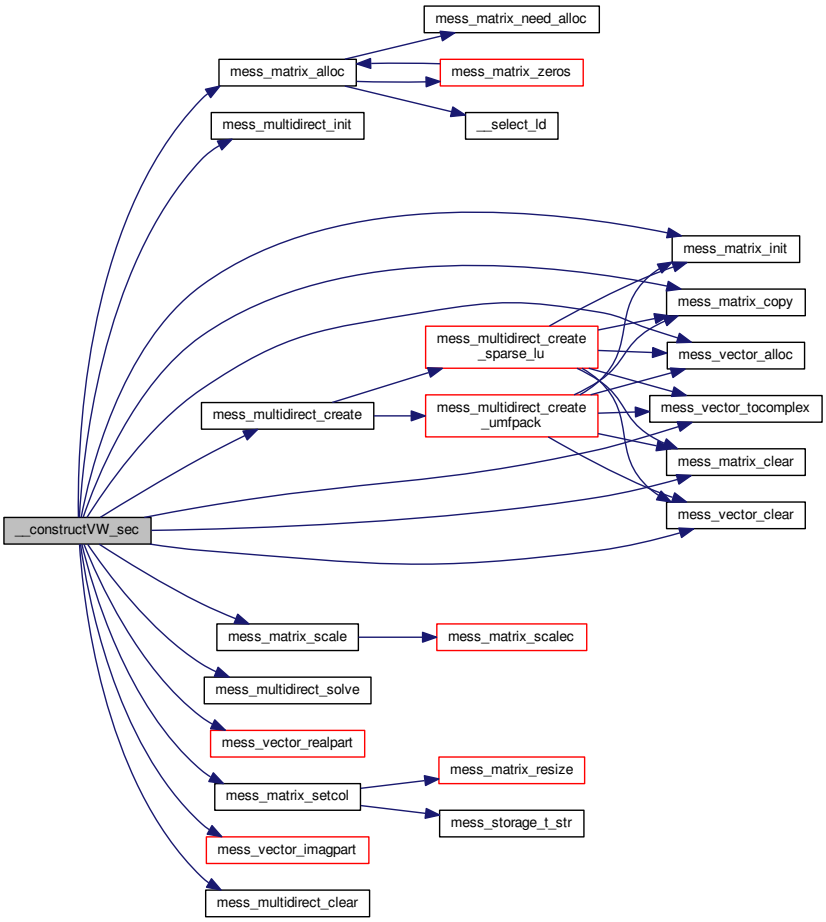
Parameters

out	*c*	output vector
in	*sigma*	input set of interpolation points
out	*V*	output span
out	*W*	output span

Returns

always zero The __constructVW_sec function constructs the span V and W for IRKA (OpenMP version).

Here is the call graph for this function:



8.34.4.5 static int __constructVW_seq (mess_matrix *A*, mess_direct *Abase*, mess_matrix *E*, mess_vector *b*, mess_vector *c*, mess_vector *sigma*, mess_matrix *V*, mess_matrix *W*) [static]

Parameters

in	*A*	input system matrix
in	*Abase*	input direct solver object

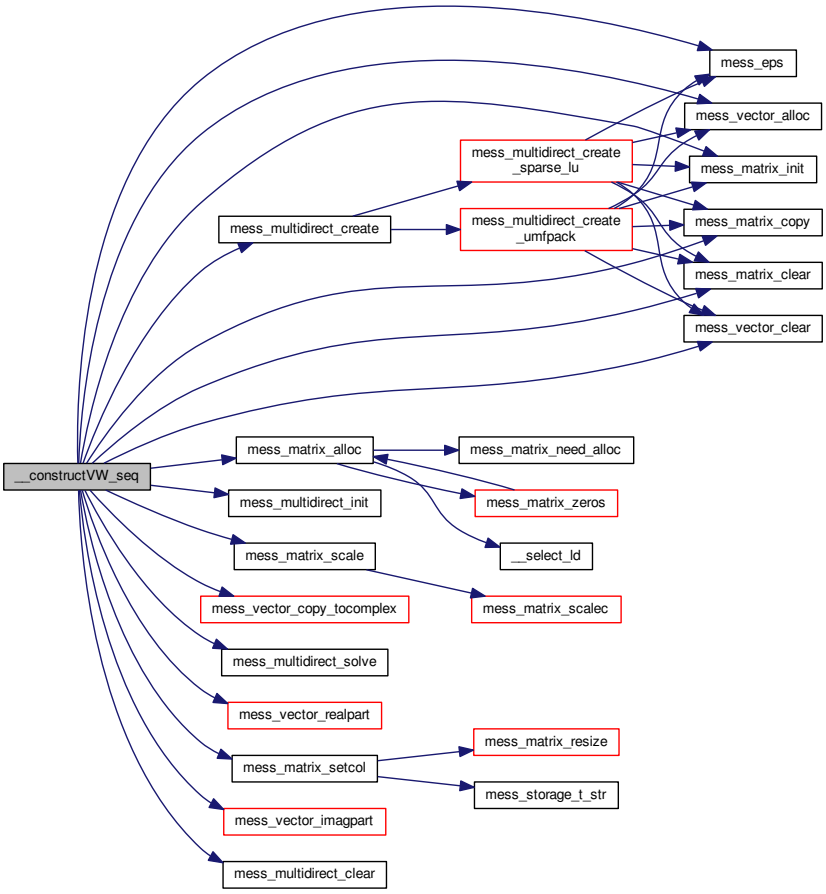
Parameters

in	*E*	input mass matrix
in	*b*	input vector
out	*c*	output vector
in	*sigma*	input set of interpolation points
out	*V*	output span
out	*W*	output span

Returns

always zero The __constructVW function constructs the span V and W for IRKA.

Here is the call graph for this function:



8.34.4.6 static int __constructVWmat (mess_matrix *A*, mess_direct *Abase*, mess_matrix *E*, mess_matrix *B*, mess_matrix *C*, mess_vector *sigma*, mess_matrix *V*, mess_matrix *W*) [static]

Parameters

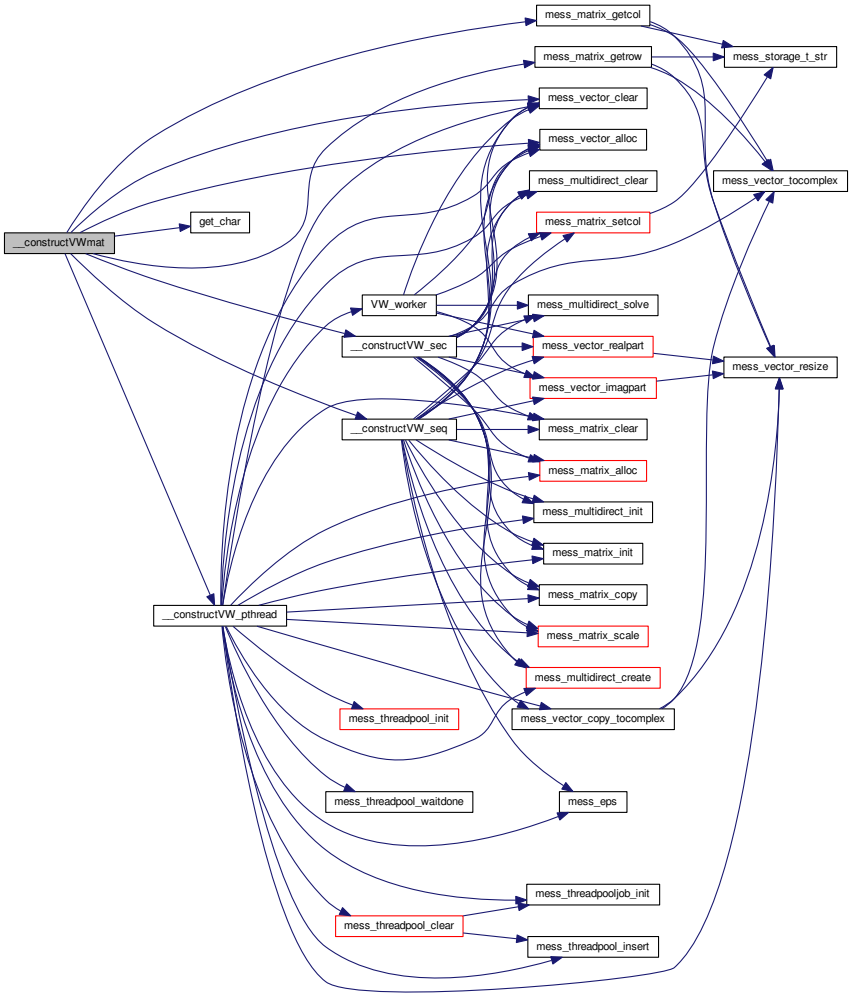
in	*A*	input system matrix

Parameters

in	*Abase*	input direct solver object
in	*E*	input mass matrix
in	*B*	input matrix
out	*C*	output matrix
in	*sigma*	input set of interpolation points
out	*V*	output span
out	*W*	output span

The __constructVWmat function contructs V and W for IRKA.

Here is the call graph for this function:



8.34.4.7 static int __project_A (mess_matrix *W*, mess_matrix *A*, mess_matrix *V*, mess_matrix *Ar*) [static]

Parameters

in	*W*	input left projector matrix

Parameters

in	*A*	input system matrix
in	*V*	input right projector matrix
out	*Ar*	output $Ar = W^T AV$

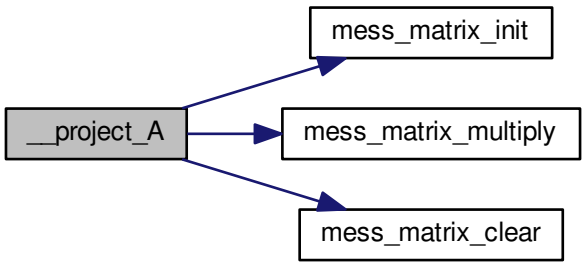
Returns

always zero

The __project_A function projects A, that means

$$Ar = W^T AV.$$

Here is the call graph for this function:



8.34.4.8 static int __project_B (mess_matrix *W*, mess_vector *b*, mess_vector *br*) [static]

Parameters

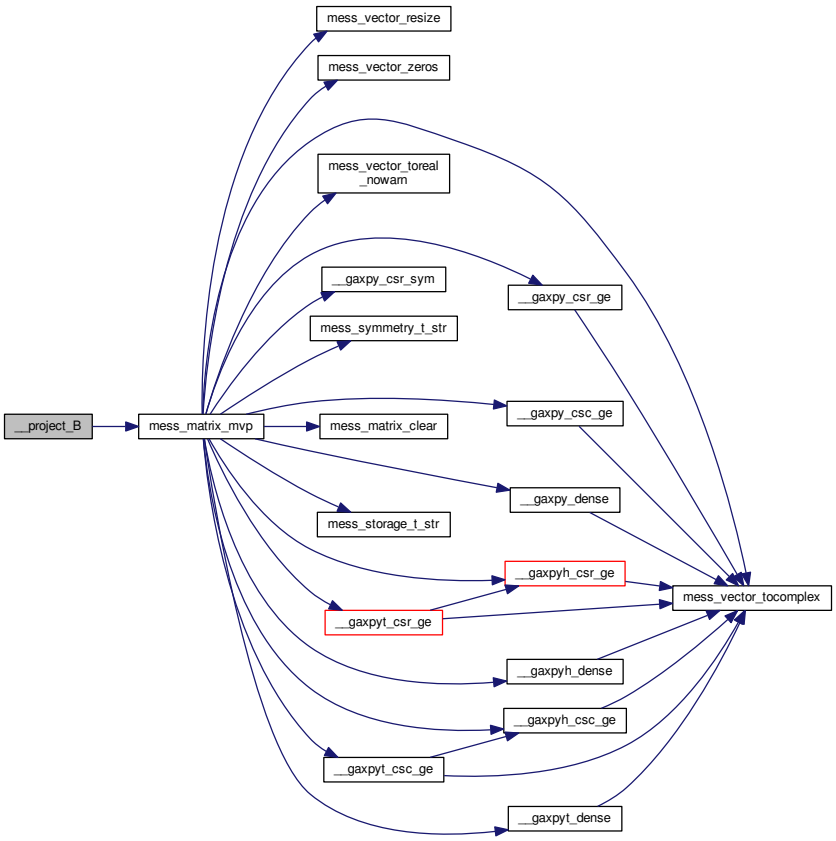
in	*W*	input left projector matrix
in	*b*	input vector
out	*br*	output $br = W^T B$

Returns

always zero The __project_B function projects b, that means

$$br = W^T b.$$

Here is the call graph for this function:



8.34.4.9 static int __project_Bmat (mess_matrix *W*, mess_matrix *B*, mess_matrix *Br*) [static]

Parameters

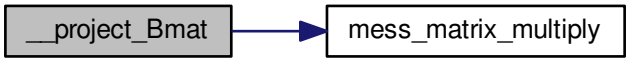
in	*W*	input left projector matrix
in	*B*	input matrix
out	*Br*	output $Br = W^T B$

Returns

always zero The __project_B function projects B, that means

$$Br = W^T B.$$

Here is the call graph for this function:



8.34.4.10 static int __project_C (mess_matrix *V*, mess_vector *c*, mess_vector *cr*) [static]

Parameters

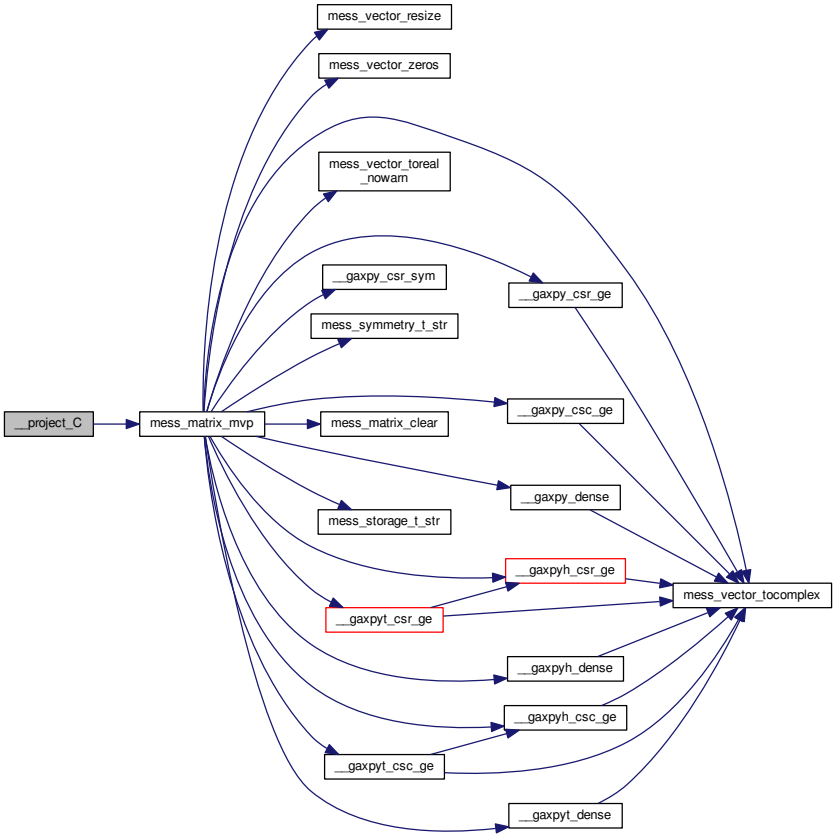
in	*V*	input left projector matrix
in	*c*	input vector
out	*cr*	output $cr = V^T * c$

Returns

always zero The __project_C function projects c, that means

$$cr = V^T C.$$

Here is the call graph for this function:



8.34.4.11 static int __project_Cmat (mess_matrix *V*, mess_matrix *C*, mess_matrix *Cr*) [static]

Parameters

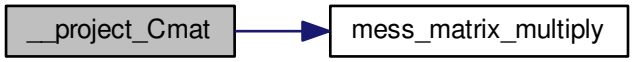
in	*V*	input right projector matrix
in	*C*	input matrix
in	*Cr*	input output $Cr = CV$

Returns

always zero The __project_Cmat function projects C, that means

$$Cr = CV.$$

Here is the call graph for this function:



8.34.4.12 static char* get_char (char * *c1*, char * *c2*) [static]

8.34.4.13 int mess_h2_error (mess_dynsys *ltiA*, mess_dynsys *ltiB*, double * *err*)

Parameters

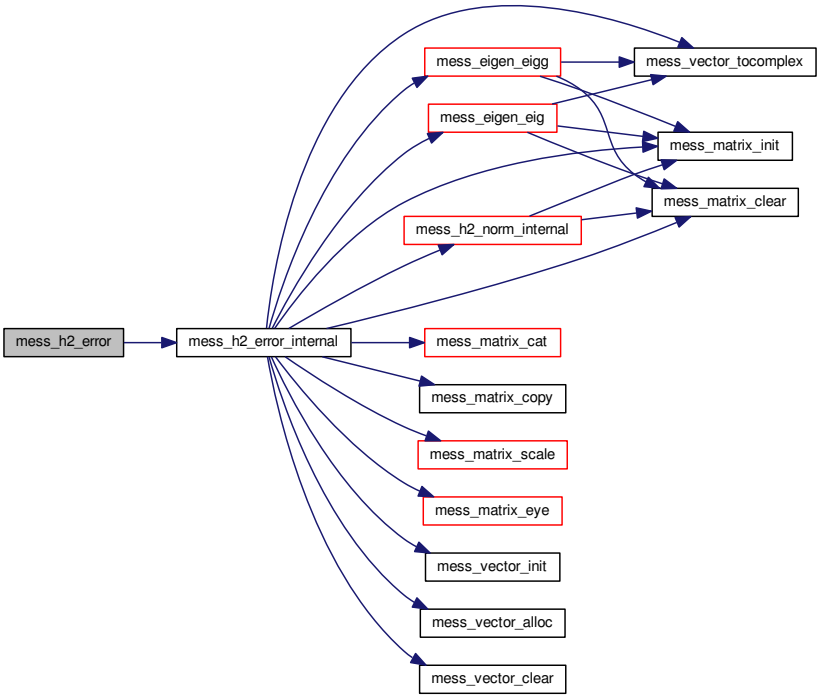
in	*ltiA*	input LTI system one
in	*ltiB*	input LTI system two
out	*err*	pointer to the error

Returns

zero on success or a non-zero error value

The mess_h2_error function computes the $\mathcal{H}_2$-error between `ltiA` and `ltiB`.
This function is only a wrapper around mess_h2_error_internal.

Here is the call graph for this function:



8.34.4.14 int mess_h2_error_internal (mess_matrix *A*, mess_matrix *B*, mess_matrix *C*, mess_matrix *E*, mess_matrix *Ar*, mess_matrix *Br*, mess_matrix *Cr*, mess_matrix *Er*, double * *norm*)

Parameters

in	*A*	input system matrix
in	*B*	input matrix
in	*C*	input output matrix
in	*E*	input mass Matrix (`NULL` if not given)
in	*Ar*	input system matrix of the second system
in	*Br*	input matrix of the second system
in	*Cr*	input output matrix of the second system
in	*Er*	input reduced mass matrix. (NULL if not given)
out	*norm*	pointer to the double value where the norm is put in

Returns

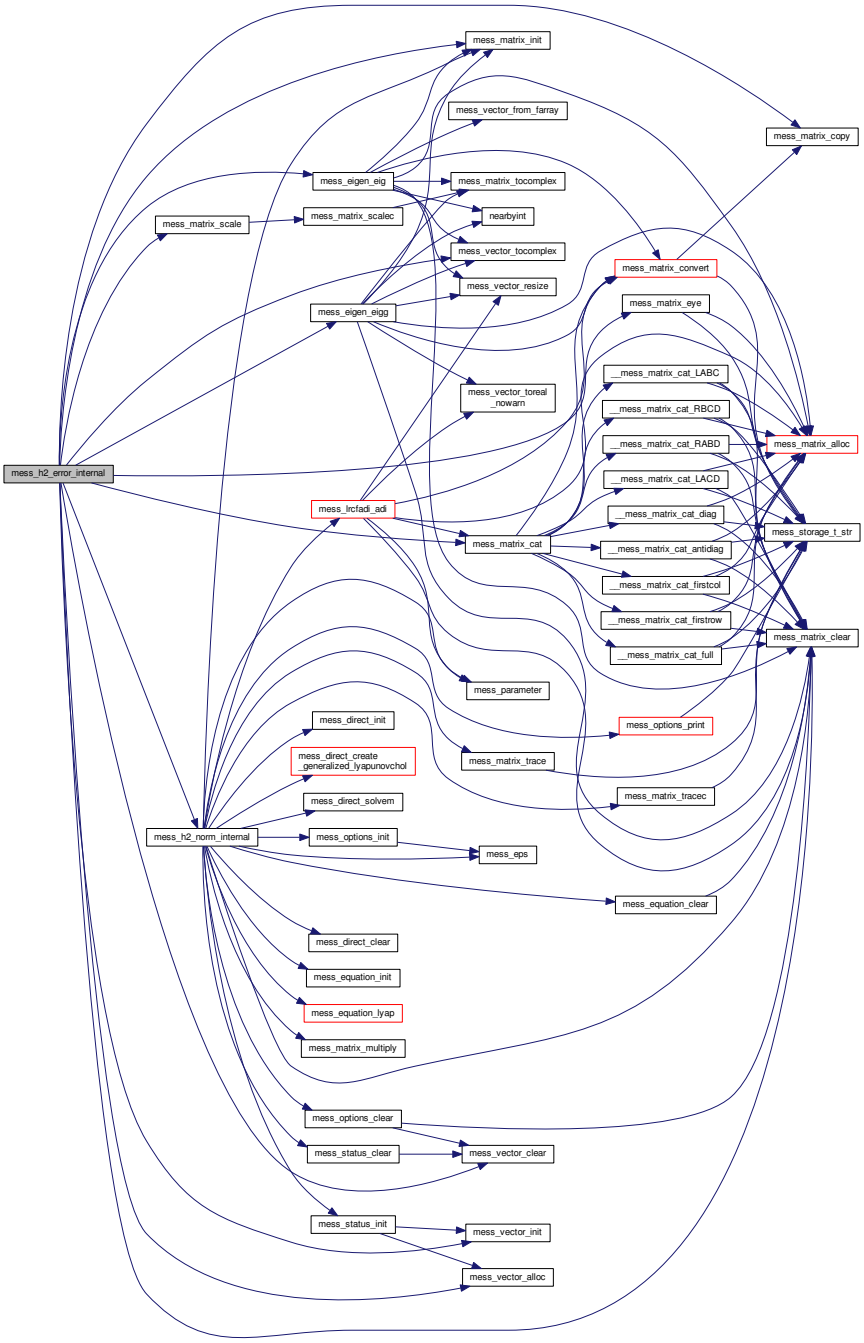
zero on success or a non-zero error value

The mess_h2_error function computes the $\mathcal{H}_2$-error of two SISO LTI systems using Lyapunov Equations.

The $\mathcal{H}_2$ error is computed as in mess_h2_norm_internal with input matrices

$$
\begin{aligned}
A_{big} &= \begin{bmatrix} A & 0 \\ 0 & Ar \end{bmatrix}, \\
B_{big} &= \begin{bmatrix} B \\ Br \end{bmatrix}, \\
C_{big} &= \begin{bmatrix} C & -Cr \end{bmatrix}, \\
E_{big} &= \begin{bmatrix} E & 0 \\ 0 & Er \end{bmatrix}.
\end{aligned}
$$

Here is the call graph for this function:



8.34.4.15 int mess_h2_irka_biorth (mess_dynsys *orig*, mess_vector *sigma*, mess_h2_options *opt*, mess_dynsys *reduced*, mess_matrix *V*, mess_matrix *W*, mess_h2_status *status*)

Parameters

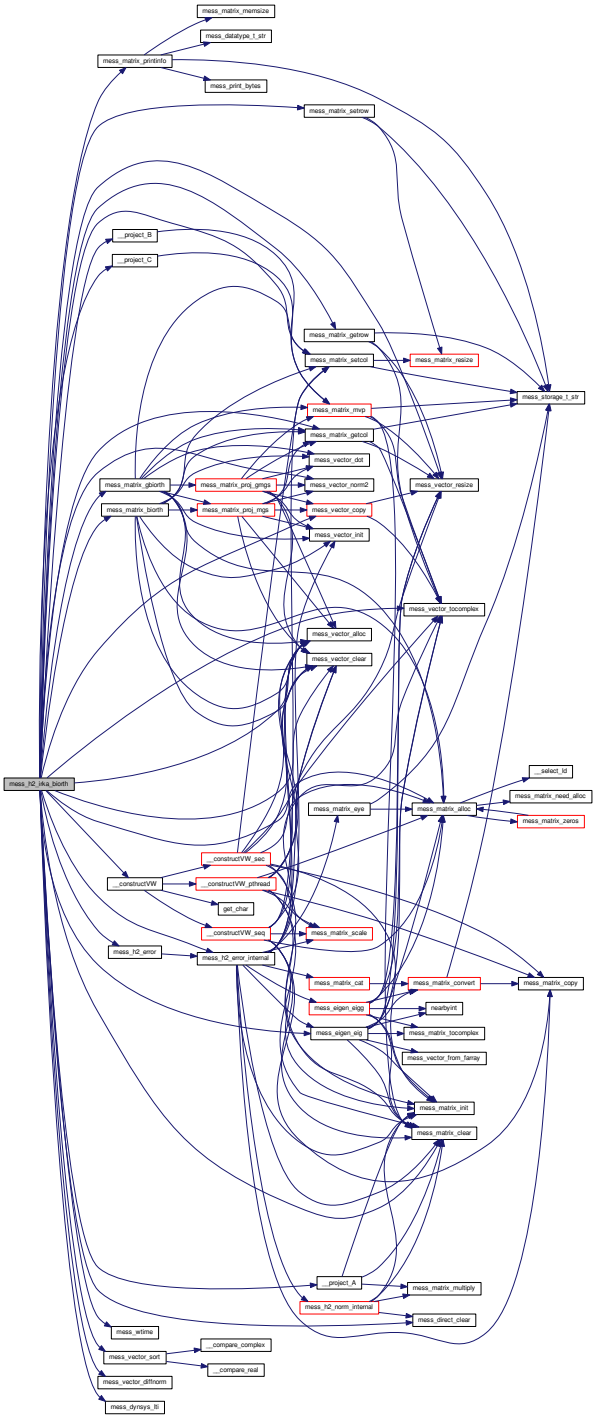
in	*orig*	input original system
in	*sigma*	input vector of initial interpolation points
in	*opt*	input options
out	*reduced*	reduced system
out	*V*	state space projection matrix
out	*W*	left projection matrix
out	*status*	output status

Returns

zero on success or a non-zero error value otherwise

The mess_h2_irka_biorth function computes a $\mathcal{H}_2$ reduced model with the IRKA algorithm.
The correction equation is solved via the biorthonormalization.

Here is the call graph for this function:



8.34.4.16 int mess_h2_irka_init (mess_matrix *A*, mess_direct *Asolver*, mess_matrix *E*, mess_int_t * *r0*, mess_int_t *kp*, mess_int_t *km*, mess_vector *sigma*)

Parameters

in	*A*	input system matrix
in	*Asolver*	input solver for A

Parameters

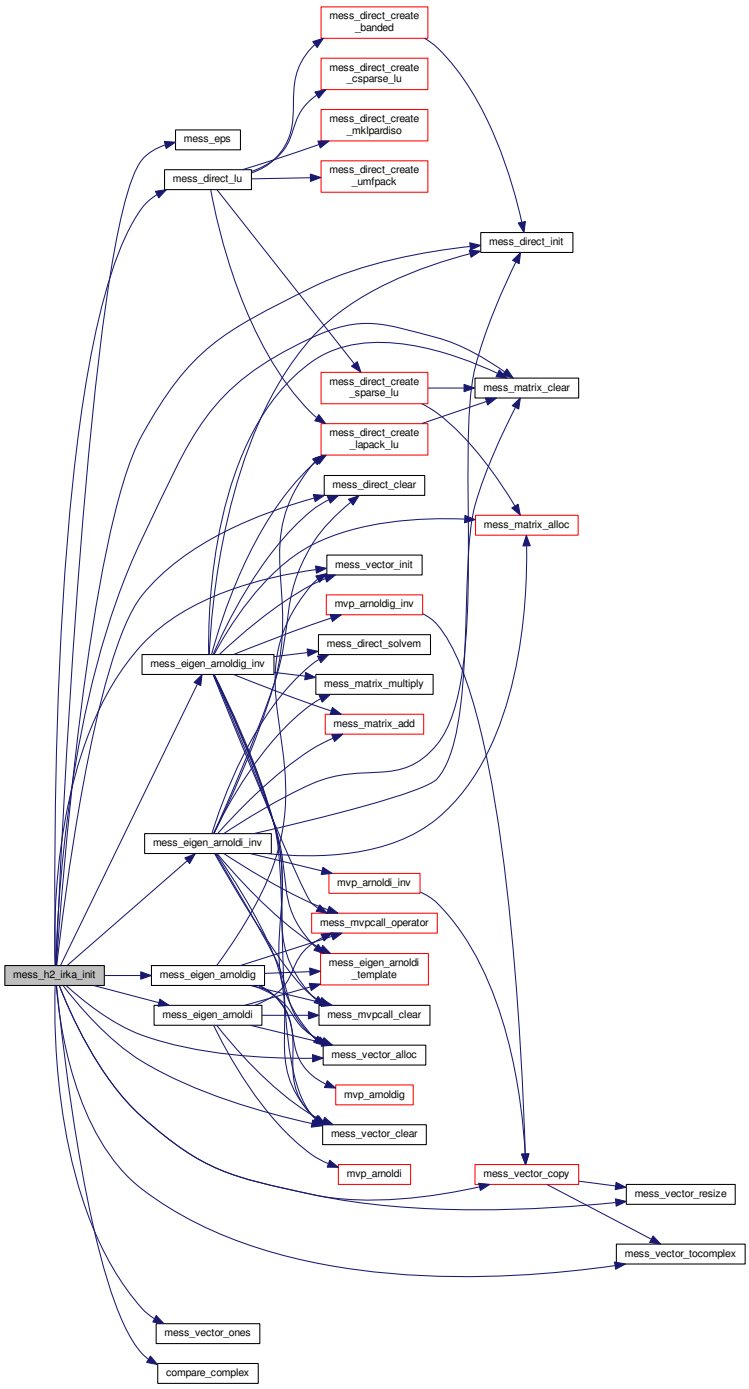
in	*E*	input mass matrix (optional)
in,out	*r0*	number of parameters
in	*kp*	input number of arnoldi steps w.r.t. A
in	*km*	input number of arnoldi steps w.r.t. $inv(A)$
out	*sigma*	vector with the parameters

Returns

zero on succes or a non-zero error value

The mess_h2_irka_init function initializes the parameters for the IRKA algorithm.

Here is the call graph for this function:



8.34.4.17 int mess_h2_norm (mess_dynsys *lti*, double * *norm*)

Parameters

in	*lti*	input given LTI system
out	*norm*	pointer to the norm

Returns

zero on success or a non-zero error value otherwise

The mess_h2_norm function computes the $\mathcal{H}_2$-norm of a LTI system

$$\begin{array}{rcll} E\ \dot{x} & = & Ax & +Bu \\ y & = & Cx & \end{array}$$

via

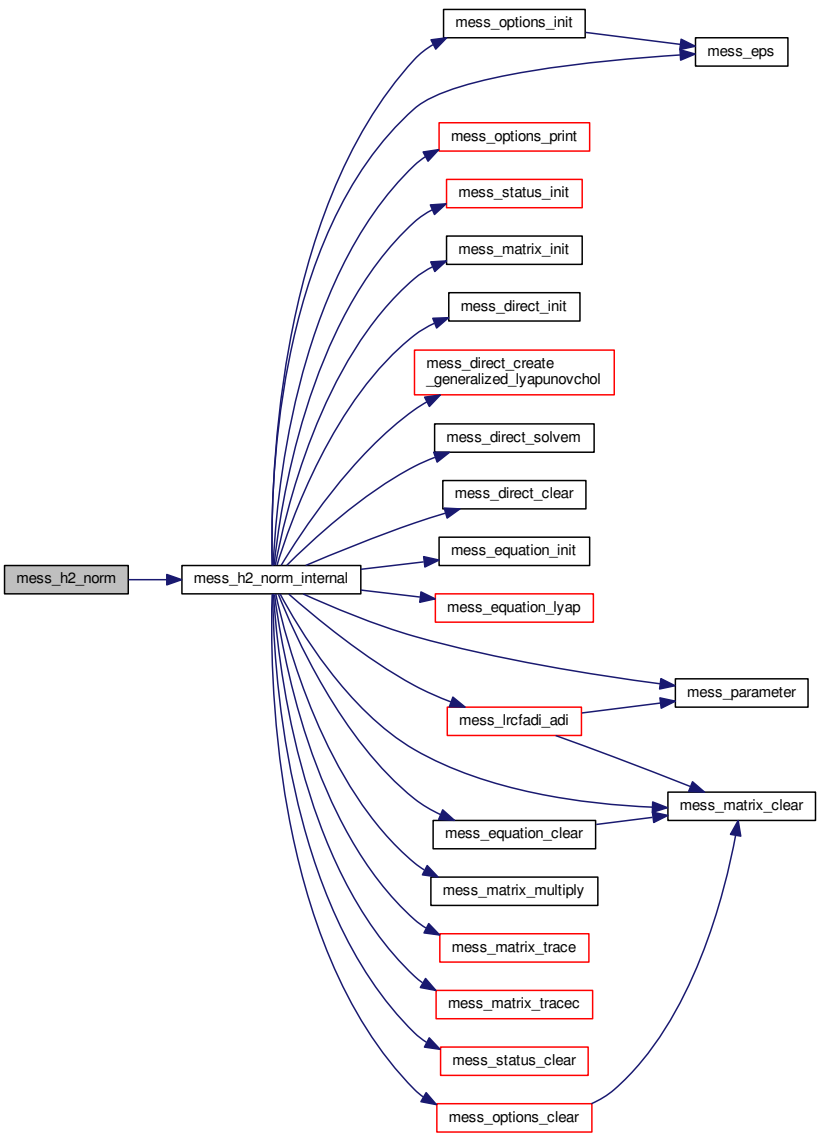
$$\|\Sigma\|_{\mathcal{H}_2} = \sqrt{trace\,(CPC^T)},$$

where Σ denotes the given LTI system and P is the solution of the Lyapunov Equation

$$APE^T + EPA^T + BB^T = 0.$$

This function is only a wrapper around mess_h2_norm_internal to work with the mess_dynsys object.

Here is the call graph for this function:



8.34.4.18 int mess_h2_norm_internal (mess_matrix *A*, mess_matrix *B*, mess_matrix *C*, mess_matrix *E*, double * *norm*)

Parameters

in	*A*	input system matrix
in	*B*	input matrix
in	*C*	input output matrix
in	*E*	input mass matrix (NULL if not need)
out	*norm*	pointer to the double value where the norm is put in

Returns

zero on success or a non-zero error value otherwise

The mess_h2_norm_internal function computes the $\mathcal{H}_2$-norm of a LTI system using Lyapunov Equations, i.e. the $\mathcal{H}_2$ norm is computed via

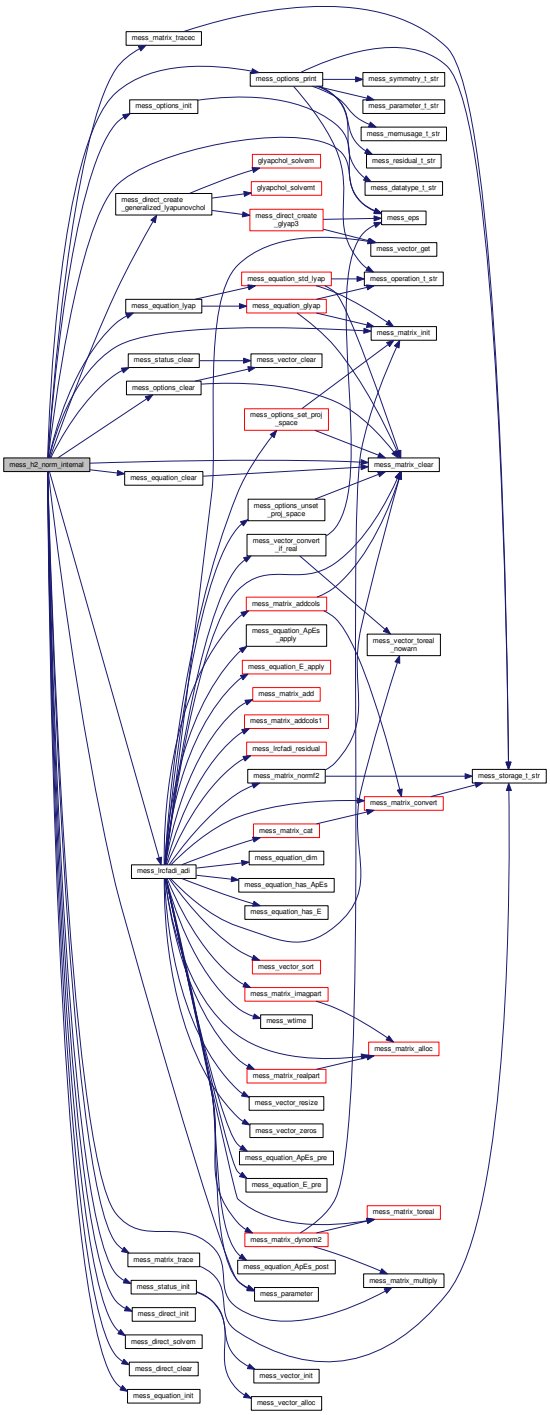
$$\|\Sigma\|_{\mathcal{H}_2} = \sqrt{trace\,(CPC^T)},$$

where Σ denotes the given LTI system and P is the solution of the Lyapunov Equation

$$APE^T + EPA^T + BB^T = 0.$$

This function is internally used by mess_h2_norm.
More details about the norm computation are available in [1] .

Here is the call graph for this function:



8.34.4.19 int mess_h2_options_clear (mess_h2_options * *opt*)

Parameters

in	*opt*	input pointer to the object

Returns

always zero

The mess_h2_options_clear function cleans up a mess_h2_options object

8.34.4.20 int mess_h2_options_init (mess_h2_options * *opt*)

Parameters

in,out	*opt*	pointer to the object

Returns

always zero

The mess_h2_options_init function initializes a mess_h2_options object.

8.34.4.21 int mess_h2_options_print (mess_h2_options *opt*)

Parameters

in	*opt*	input mess_h2_options object

Returns

zero on success or a non-zero error value otherwise

The mess_h2_options_print function prints a mess_h2_options object to the screen.

8.34.4.22 int mess_h2_status_clear (mess_h2_status * *status*)

Parameters

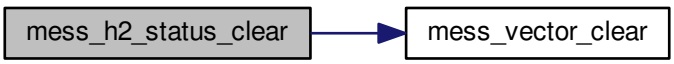
in	*status*	input pointer to the object

Returns

always zero

The mess_h2_status_clear function cleans up a mess_h2_status object.

Here is the call graph for this function:



8.34.4.23 int mess_h2_status_init (mess_h2_status * *status*)

Parameters

in,out	*status*	pointer to the status object

Returns

always zero

The mess_h2_status_init function initializes a mess_h2_status object.

Here is the call graph for this function:

8.34.4.24 int mess_h2_status_print (mess_h2_status *status*)

Parameters

in	*status*	input mess_h2_status object

Returns

zero on success or a non-zero error value

The mess_h2_status_print function prints a mess_h2_status object on the screen.

8.34.4.25 int mess_h2_status_printfull (mess_h2_status *status*)

Parameters

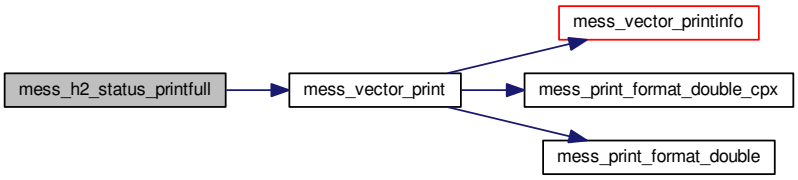
in	*status*	input mess_h2_status object

Returns

zero on success or a non-zero error value

The mess_h2_status_printfull function prints a mess_h2_status object on the screen.

Here is the call graph for this function:



8.34.4.26 int mess_h2_status_write_mfile (mess_h2_status *status*, const char * *filename*)

Parameters

in	*status*	input mess_h2_status object
in	*filename*	input name of the mfile

Returns

zero on success or a non-zero error value

The mess_h2_status_write_mfile function writes a mfile to get the status data easily to MATLAB.

8.34.4.27 int mess_h2_tsia (mess_dynsys *orig*, mess_vector *sigmaEX*, mess_h2_options *opt*, mess_dynsys *redu*, mess_matrix *V*, mess_matrix *W*, mess_h2_status *status*)

Parameters

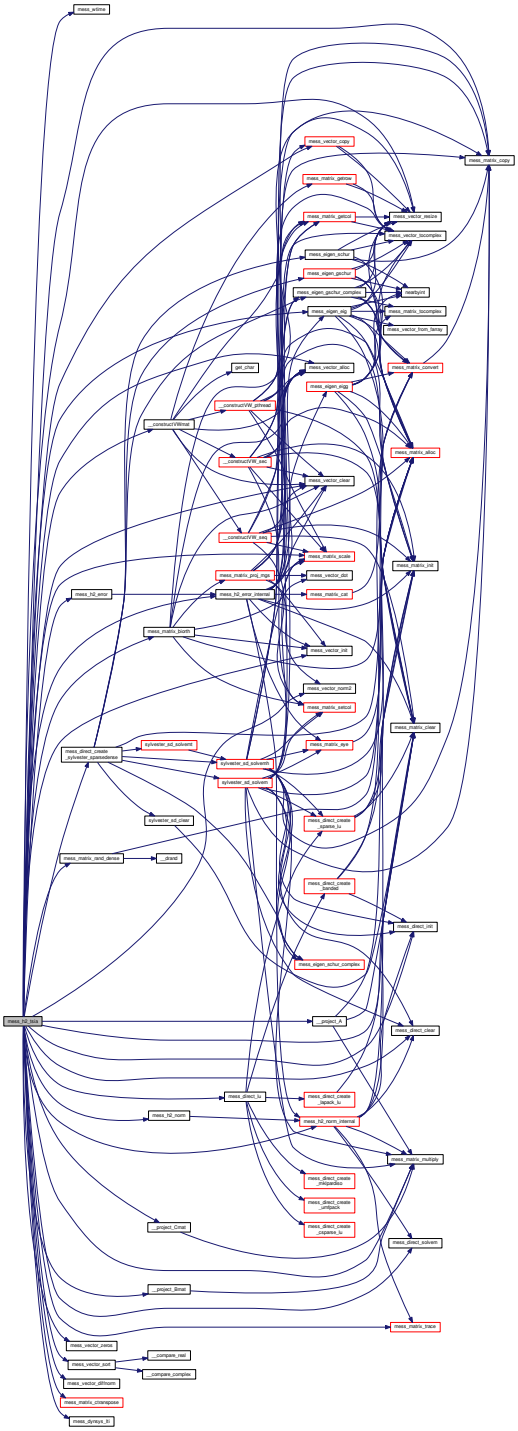
in	*orig*	input original system
in	*sigmaEX*	input vector of initial interpolation points (NULL if not needed)
in	*opt*	input options
out	*redu*	reduced system
in	*V*	input state space projection matrix
in	*W*	input left projection matrix
out	*status*	output status

Returns

zero on success or a non-zero error value otherwise

The mess_h2_tsia function computes a $\mathcal{H}_2$ reduced model via the TSIA algorithm.
If sigmaEX!=NULL and the input system is a SISO system, the first projection will be build from this interpolation points.
If no sigmaEX is given V and W are used as projection. If V and W are empty, random matrices are used.

Here is the call graph for this function:



8.34.4.28 int mess_h2_tsiag (mess_dynsys *orig*, mess_vector *sigmaEX*, mess_h2_options *opt*, mess_dynsys *redu*, mess_matrix *V*, mess_matrix *W*, mess_h2_status *status*)

Parameters

in	*orig*	input original system
in	*sigmaEX*	input vector of initial interpolation points (NULL if not needed)

Parameters

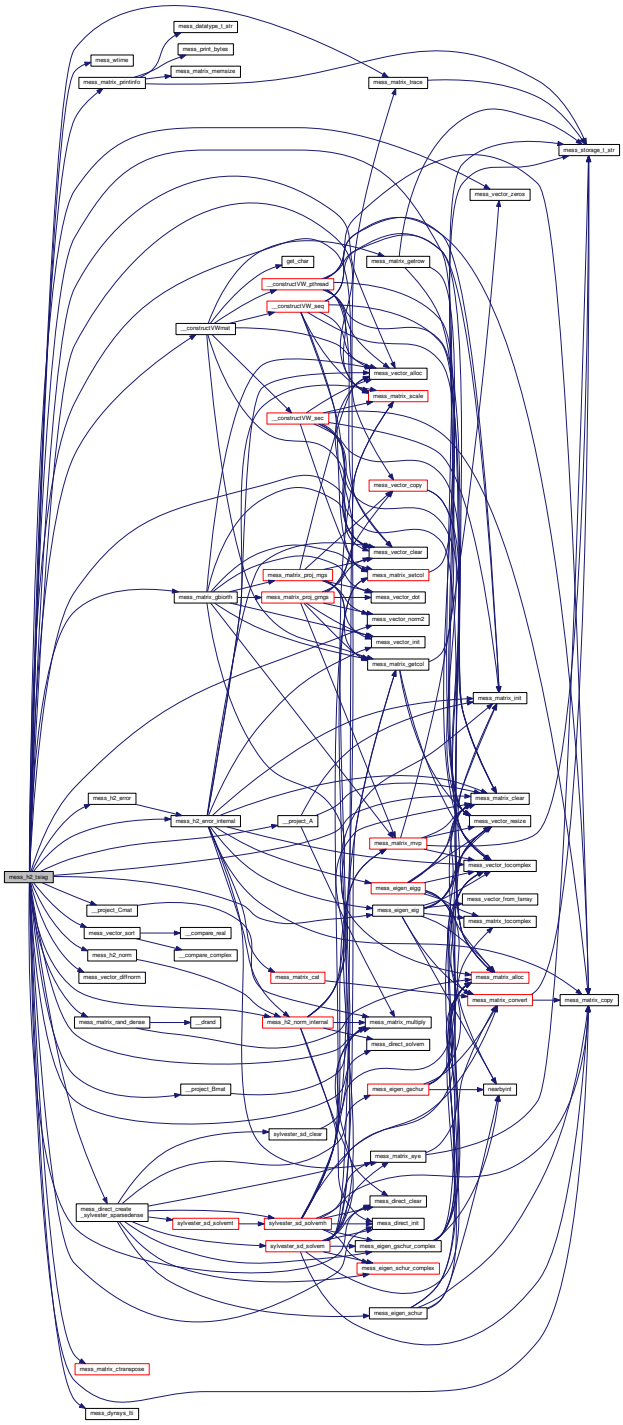
in	*opt*	input options
out	*redu*	reduced system
in	*V*	input state space projection matrix
in	*W*	input left projection matrix
out	*status*	output status

Returns

zero on success or a non-zero error value otherwise

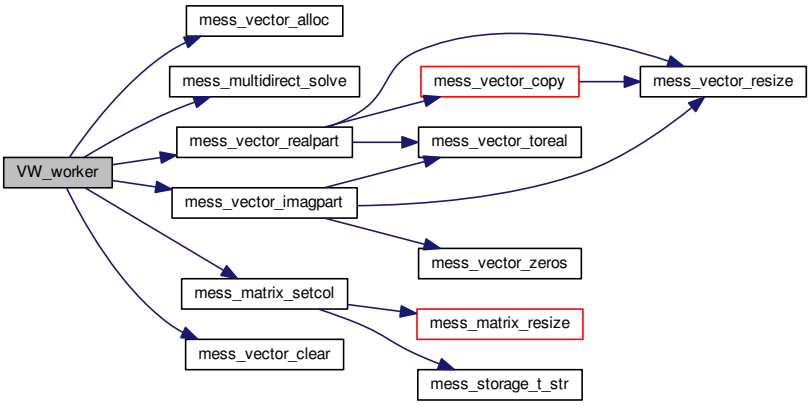
The mess_h2_tsia function computes a $\mathcal{H}_2$ reduced model via the TSIA algorithm.
If sigmaEX!=NULL and the input system is a SISO system, the first projection will be build from this interpolation points.
If no sigmaEX is given V and W are used as projection. If V and W are empty, random matrices are used.

Here is the call graph for this function:



8.34.4.29 static void VW_worker (void * *data*) [static]

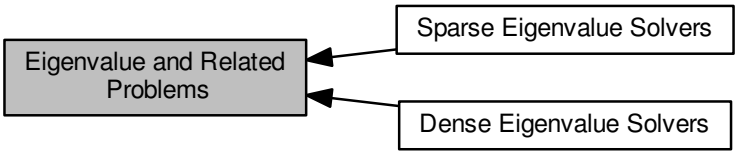
Here is the call graph for this function:



8.35 Eigenvalue and Related Problems

This section contains categories to compute eigenvalues, eigenvectors and solve related problems.

Collaboration diagram for Eigenvalue and Related Problems:



Modules

- Dense Eigenvalue Solvers

  This category contains all necessary functions to solve the eigenvalue problem and related problems.
- Sparse Eigenvalue Solvers

  This category contains all necessary structures, definitions, enumerations and functions to solve the eigenvalue problem and related problems for sparse matrices.

8.35.1 Detailed Description

A scalar $\lambda \in \mathbb{C}$ is called an **eigenvalue** of a matrix $A \in \mathbb{C}^{n \times n}$ if λ is a root of the **characteristic polynomial**

$$p(\lambda) := det(A - \lambda I).$$

A non-zero vector $x \in \mathbb{C}^n$ is called a (right) **eigenvector** of A associated with the eigenvalue λ if it fulfills

$$(A - \lambda I)x = 0.$$

The corresponding **left eigenvectors** satisfies

$$y^H A = \lambda y^H.$$

The problem to find scalars λ and non-zero vectors x satisfying

$$Ax = \lambda x$$

is called standard **eigenvalue problem**.

A **generalized eigenvalue problem** is the problem of finding scalars $\lambda \in \mathbb{C}$ and non-zero vectors $x \in \mathbb{C}^n$ that satisfy

$$Ax = \lambda Bx,$$

where $A, B \in \mathbb{C}^{n \times n}$.
The scalar λ is a root of $det(A - \lambda B)$ and is called **generalized eigenvalue** with corresponding (right) **eigenvector** x.
A non-zero vector $y \in \mathbb{C}^n$ fulfilling

$$y^H A = \lambda y^H B$$

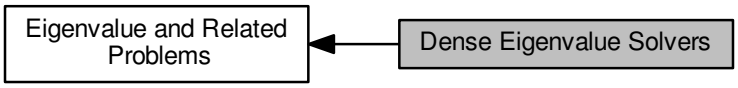
is called a **left eigenvector** of (A, B).

source: [http://www2.mpi-magdeburg.mpg.de/mpcsc/mitarbeiter/kuerschner/docs/masterthesis.↩pdf]

8.36 Dense Eigenvalue Solvers

This category contains all necessary functions to solve the eigenvalue problem and related problems.

Collaboration diagram for Dense Eigenvalue Solvers:



Functions

- int mess_eigen_eig (mess_matrix A, mess_vector ew, mess_matrix ev)

  Compute all eigenvalues and eigenvectors.
- int mess_eigen_eigg (mess_matrix A, mess_matrix B, mess_vector ew, mess_vector alpha, mess_vector beta, mess_matrix ev)

  Compute all eigenvalues and right eigenvectors of the generalized eigenvalue problem.
- int mess_eigen_schur_complex (mess_matrix A, mess_matrix T, mess_matrix U, mess_vector EV)

  Wrapper around both complex Schur decomposition routines.
- int mess_eigen_schur_complex_zgees (mess_matrix A, mess_matrix T, mess_matrix U, mess_vector EV)

  Compute the complex Schur decomposition of a matrix.
- int mess_eigen_schur_complex_post (mess_matrix A, mess_matrix T, mess_matrix U, mess_vector EV)

  Compute the complex Schur decomposition of a matrix using post processing.
- int mess_eigen_schur (mess_matrix A, mess_matrix T, mess_matrix U, mess_vector EV)

  Compute the real Schur decomposition of a matrix.
- int mess_eigen_gschur (mess_matrix A, mess_matrix B, mess_matrix S, mess_matrix T, mess_matrix U, mess_matrix V, mess_vector EVA, mess_vector EVB, mess_vector EV)

  Compute the real generalized Schur decomposition of a matrix pair (A, B).
- int mess_eigen_gschur_complex (mess_matrix A, mess_matrix B, mess_matrix S, mess_matrix T, mess_matrix U, mess_matrix V, mess_vector EVA, mess_vector EVB, mess_vector EV)

  Compute the generalized Schur decomposition of a matrix pair (A, B).
- int mess_eigen_svd (mess_matrix A, mess_vector S, mess_matrix U, mess_matrix V)

  Compute the singular value decompositon (SVD) of a matrix.
- int mess_eigen_svd_econ (mess_matrix A, mess_vector S, mess_matrix U, mess_matrix V)

  Compute the singular value decompositon (SVD) of a matrix.
- int mess_eigen_schur_to_evd (mess_matrix T, mess_matrix Q, mess_matrix V)

  Compute eigenvectors from a given Schur decomposition-.
- int mess_eigen_hessenberg_schur (mess_matrix hess, mess_vector ew, mess_matrix Q, mess_matrix T)

  Compute Schur decomposition of an upoer Hessenberg matrix.
- int mess_eigen_hessenberg_abs_largest (mess_matrix hess, mess_vector evector, mess_double_cpx_t *abs_largest_ev)

  Compute the largest eigenvalue in magnitude and corresponding eigenvector of a upper Hessenberg matrix,.
- int mess_eigen_sign (mess_matrix A, mess_matrix Z, mess_sign_scale_t scale)
- int mess_eigen_gsign (mess_matrix A, mess_matrix B, mess_matrix Z, mess_sign_scale_t scale)

  Compute the sign of a matrix pencil.

8.36.1 Detailed Description

It includes functions computing eigenvalues and left and right eigenvectors of one matrix or a pair of matrices and residuals of eigenvalues.

Additionally there are functions returning the real or complex **Schur decomposition** of a matrix $A \in \mathbb{C}^{n\times n}$, i.e.

$$T = Q^H AQ = diag(\lambda_1, \cdots, \lambda_n) + N,$$

where $Q \in \mathbb{C}^{n\times n}$ is an unitary matrix and $N \in \mathbb{C}^{n\times n}$ is a strictly upper triangular nilpotent matrix.
The real or complex **generalized Schur decomposition** of two matrices $A, B \in \mathbb{C}^{n\times n}$, i.e.

$$\begin{array}{rcll} S & = & Q^H AZ & , \\ T & = & Q^H BZ & , \end{array}$$

where S, T are (quasi) upper triangular and upper triangular matrices is also implemented.
Their diagonal entries reveal the eigenvalues of (A, B) by

- $\lambda_i = \frac{s_{ii}}{t_{ii}}$,
- $\lambda_i = \infty$ if $t_{ii} = 0$ and $s_{ii} \neq 0$,
- $\Lambda(A, B)$ = $\mathbb{C}$ if there is an i with $s_{ii} = t_{ii} = 0$,

for $i = 1, \cdots, n$.

Furthermore this category contains functions computing the **singular value decomposition (SVD)** of a matrix $A \in \mathbb{C}^{n\times n}$, i.e

$$A = U\Sigma V^H,$$

where $U \in \mathbb{C}^{m\times m}, V \in \mathbb{C}^{n\times n}$ and $\Sigma \in \mathbb{C}^{m\times n}$.The diagonal entries of the upper $(n \times n)$ matrix Σ are called singular values and are ordered such that

$$\sigma_{1,1} \geq \cdots \geq \sigma_{r,r} > \cdots > \sigma_{n,n} = 0,$$

where $r := rank(A)$.

Another implemented function returns the **sign** of a matrix $A \in \mathbb{C}^{n\times n}$, i.e.

$$sign(A) = A(A^2)^{-\frac{1}{2}}.$$

For more information about the computation of the sign of a matrix see Additional Matrix Equation Solvers.

source: [http://www2.mpi-magdeburg.mpg.de/mpcsc/mitarbeiter/kuerschner/docs/masterthesis.↩pdf]

8.36.2 Function Documentation

8.36.2.1 int mess_eigen_eig (mess_matrix *A*, mess_vector *ew*, mess_matrix *ev*)

Parameters

in	*A*	input matrix
out	*ew*	vector containing eigenvalues
out	*ev*	matrix containing eigenvectors (if not wanted set it to NULL)

Returns

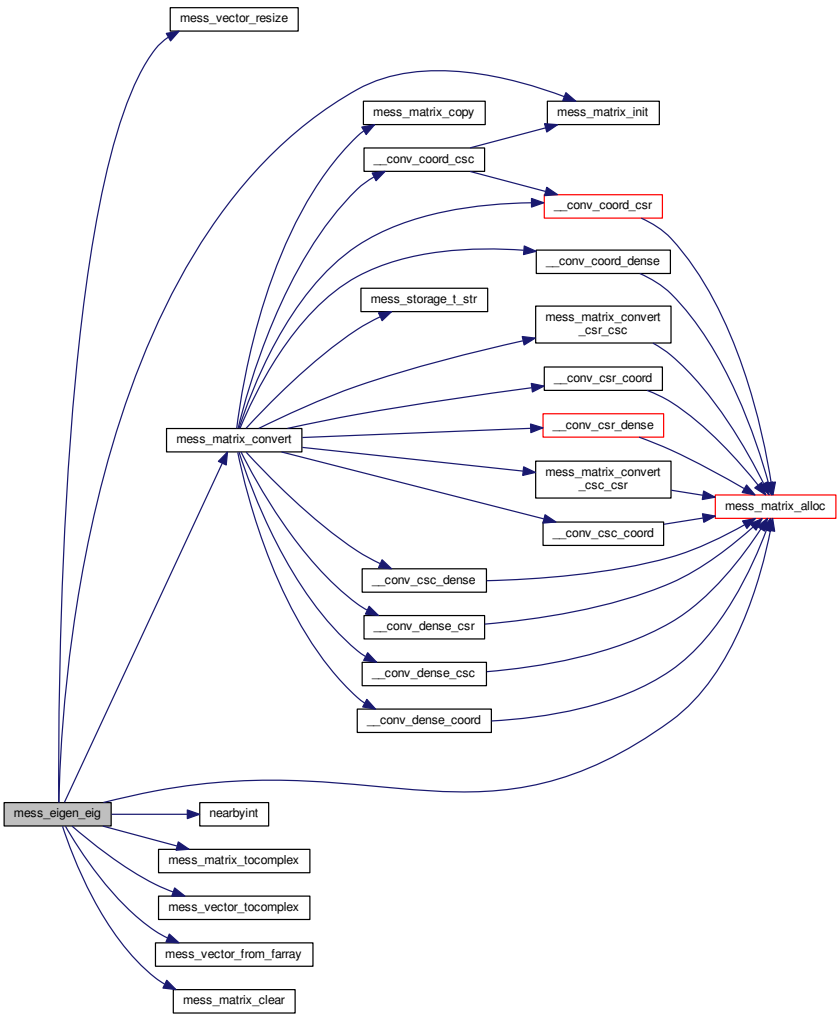
zero on success or a non-zero error value otherwise

The mess_eigen_eig function solves the eigenvalue problem

$$Ax = \lambda x.$$

Therefore it determines all eigenvalues λ stored in `ew` and if wanted all eigenvectors x stored in `ev` with xGEEV from `LAPACK`.

Here is the call graph for this function:



8.36.2.2 int mess_eigen_eigg (mess_matrix *A*, mess_matrix *B*, mess_vector *ew*, mess_vector *alpha*, mess_vector *beta*, mess_matrix *ev*)

Parameters

in	*A*	input matrix
in	*B*	input matrix

Parameters

out	*ew*	vector containing eigenvalues of the generalized eigenvalue problem (NULL if not wanted)
out	*alpha*	in combination with β a rational pair for the eigenvalues (NULL if not wanted)
out	*beta*	in combination with α a rational pair for the eigenvalues (NULL if not wanted)
out	*ev*	matrix containing right eigenvectors (if not wanted set it to NULL)

Returns

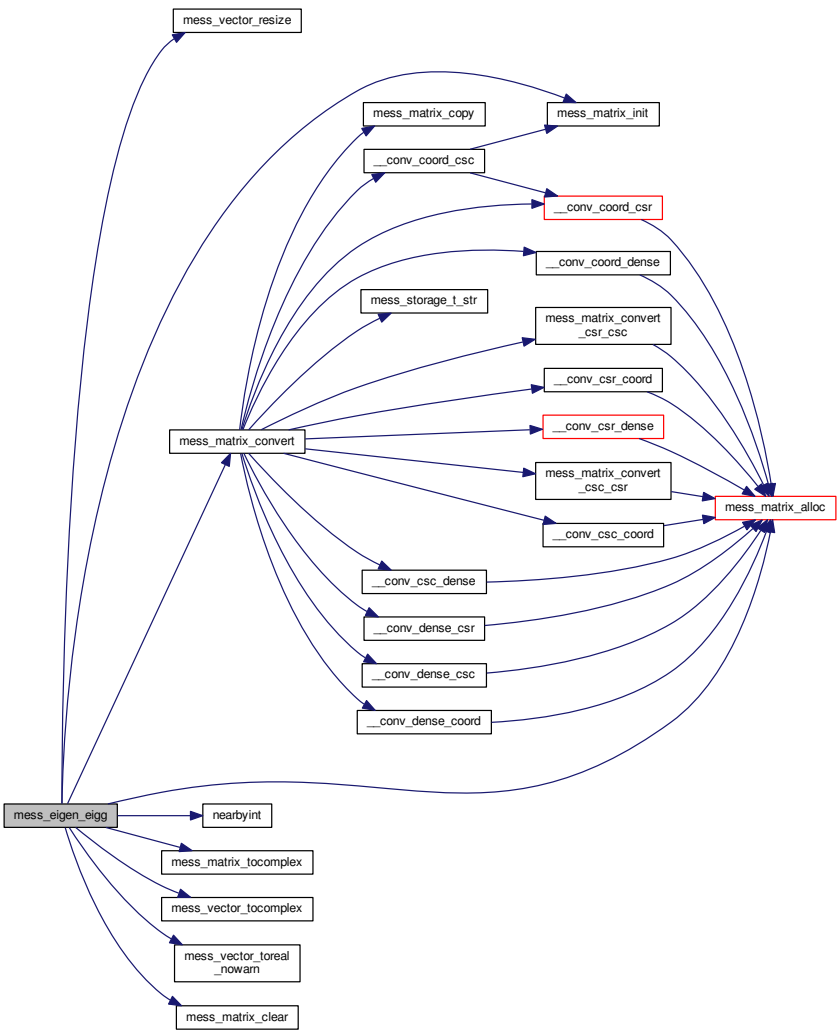
zero on success or a non-zero error value otherwise

The mess_eigen_eigg function computes all eigenvalues λ stored in ew and if wanted all right eigenvectors x stored in ev of a given real matrix pair (A, B), i.e. this function solves

$$Ax = \lambda Bx$$

with xGGEV from LAPACK.

Here is the call graph for this function:



8.36.2.3 int mess_eigen_gschur (mess_matrix *A*, mess_matrix *B*, mess_matrix *S*, mess_matrix *T*, mess_matrix *U*, mess_matrix *V*, mess_vector *EVA*, mess_vector *EVB*, mess_vector *EV*)

Parameters

in	*A*	input matrix to decompose
in	*B*	input matrix to decompose
out	*S*	Schur form of A
out	*T*	Schur form of B
in,out	*U*	left orthogonal transformation
in,out	*V*	right orthogonal transformation
in,out	*EVA*	generalized eigenvalues of A
in,out	*EVB*	generalized eigenvalues of B
in,out	*EV*	vector containing eigenvalues of the generalized eigenvalue problem

Returns

zero on success or a non-zero error value otherwise

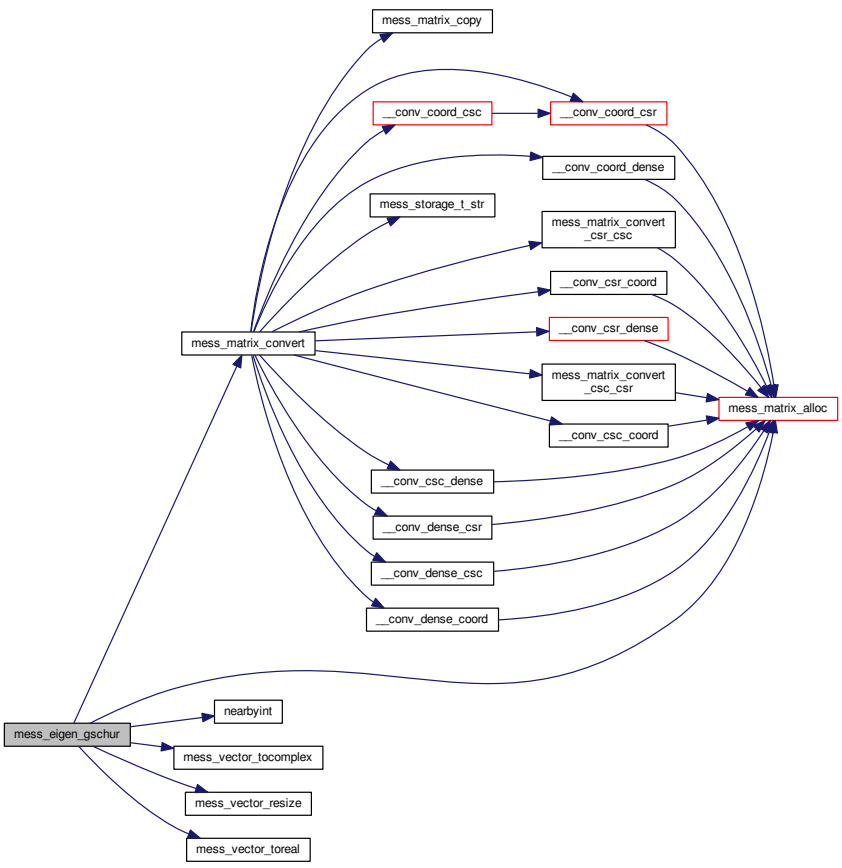
The mess_eigen_gschur function computes the real generalized Schur decomposition of A and B:

$$\begin{aligned} S &= U^T A V, \\ T &= U^T B V, \end{aligned}$$

where S is an upper quasi-triangular matrix and T is an upper triangular matrix and EVA and EVB using LAPACK's DGGES subroutine.
Additionally this function computes the eigenvalues EV of the generalized eigenvalue problem :

$$Ax = \lambda B x.$$

Here is the call graph for this function:



8.36.2.4 int mess_eigen_gschur_complex (mess_matrix *A*, mess_matrix *B*, mess_matrix *S*, mess_matrix *T*, mess_matrix *U*, mess_matrix *V*, mess_vector *EVA*, mess_vector *EVB*, mess_vector *EV*)

Parameters

in	*A*	input matrix to decompose
in	*B*	input matrix to decompose
out	*S*	Schur form of A
out	*T*	Schur form of B
out	*U*	left orthogonal transformation
out	*V*	right orthogonal transformation
out	*EVA*	generalized eigenvalues of A
out	*EVB*	generalized eigenvalues of B
out	*EV*	vector containing eigenvalues of the generalized eigenvalue problem

Returns

zero on success or a non-zero error value otherwise

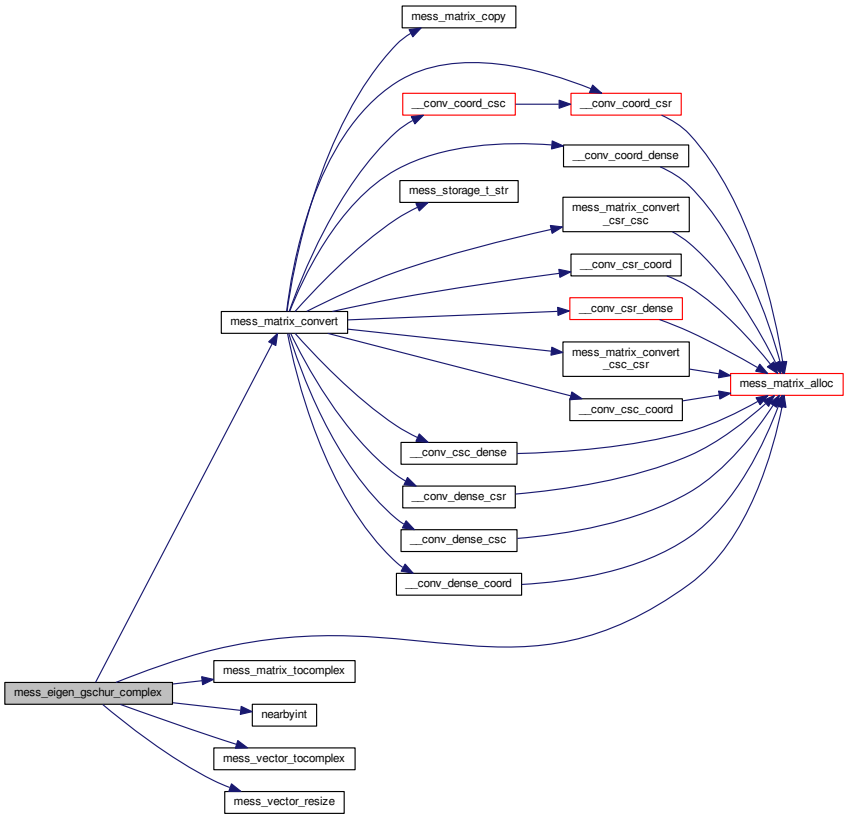
The mess_eigen_gschur_complex function computes the generalized Schur decomposition of A and B:

$$\begin{aligned} S &= U^T A V, \\ T &= U^T B V, \end{aligned}$$

where S and T are upper triangular matrices and EVA and EVB using LAPACK's ZGGES subroutine. Additionally this function computes the eigenvalues EV of the generalized eigenvalue problem :

$$Ax = \lambda B x.$$

Here is the call graph for this function:



8.36.2.5 int mess_eigen_gsign (mess_matrix *A*, mess_matrix *B*, mess_matrix *Z*, mess_sign_scale_t *scale*)

Parameters

in	*A*	input matrix
in	*B*	input matrix
out	*Z*	$Z = sign(A, B)$
in	*scale*	scaling method

Returns

zero on success or a non-zero error value otherwise

The mess_eigen_gsign function computes the sign of a matrix pencil (A, B) using the Newton sign function iteration.

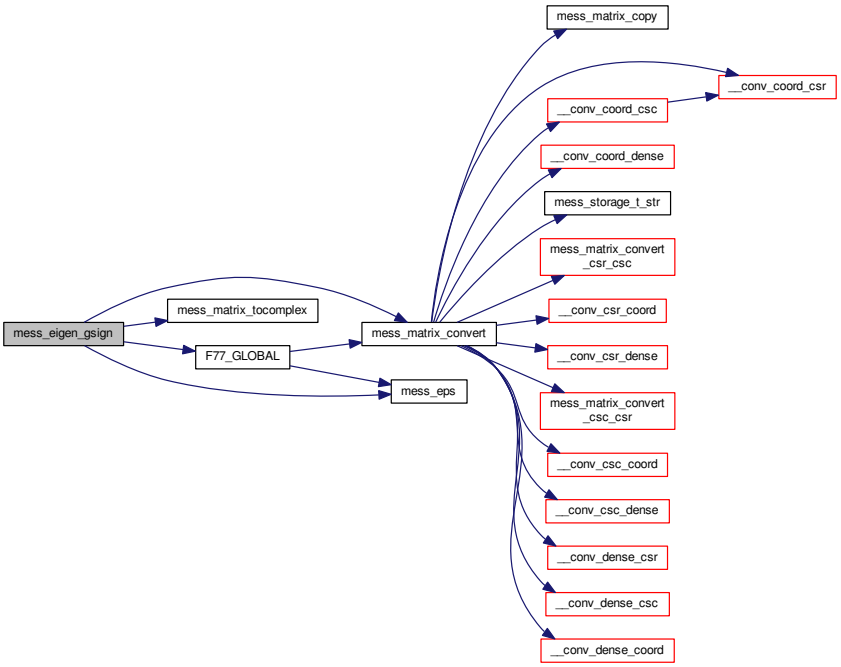
Attention

In the case of MESS_SIGN_SCALE_FRO we switch automatically to MESS_SIGN_SCALE_DET, because the Frobenius Norm scaling $c_k = \sqrt{(\frac{||B^{-1}A||_F}{||A^{-1}B||_F})}$ would introduce an additional solve.

See also

mess_sign_scale_t

Here is the call graph for this function:



8.36.2.6 int mess_eigen_hessenberg_abs_largest (mess_matrix *hess*, mess_vector *evector*, mess_double_cpx_t * *largest_ev*)

Parameters

in	*hess*	input upper Hessenberg Matrix
out	*evector*	output normalized eigenvector corresponding to largest eigenvalue
out	*largest_ev*	output largest eigenvalue in magnitude of `hess`

Returns

zero on success or a non-zero error value otherwise

The mess_eigen_hessenberg_abs_largest function computes the largest eigenvalue in magnitude and a corresp-nding eigenvector of a upper Hessenberg matrix `hess`.

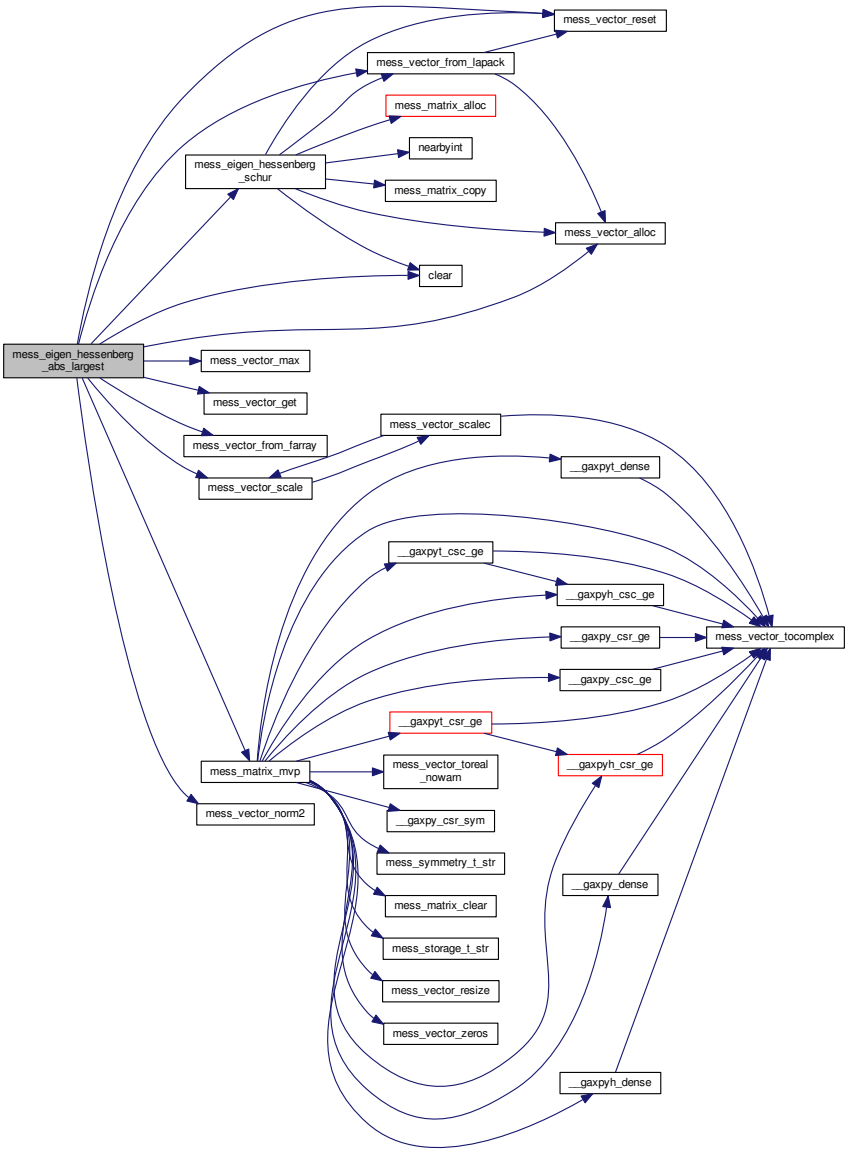
See also

mess_eigen_hessenberg_abs_largest
mess_eigen_schur_to_evd

Warning

It is not checked that `hess` is upper Hessenberg.

Here is the call graph for this function:



8.36.2.7 int mess_eigen_hessenberg_schur (mess_matrix *hess*, mess_vector *ev*, mess_matrix *Q*, mess_matrix *T*)

Parameters

in	*hess*	input upper Hessenberg Matrix
out	*ev*	output vector containing eigenvalues of `hessA`
out	*Q*	output unitary transformation matrix of Schur decomposition
out	*T*	output, (quasi)-upper triangular matrix of Schur decompsotion

Returns

zero on success or a non-zero error value otherwise

The mess_eigen_hessenberg_schur function computes the Schur decomposition of an upper Hessenberg matrix `hess`. If `hess` is MESS_REAL the real Schur decomposition is computed. This means that `T` is quasi upper triangular at return. If `hess` is MESS_COMPLEX the complex Schur decomposition is computed. It holds that $hess = QTQ^H$ and `Q` is orthogonal / unitary respectively. The function calls `DHSEQR` / `ZHSEQR` from `LAPACK`. If you want to get the eigenvector too, use mess_eigen_schur_to_evd with the returned Schur decomposition. If the Schur decomposition is not wanted, call mess_eigen_hessenberg_schur with `Q` and `T` equals `NULL`.

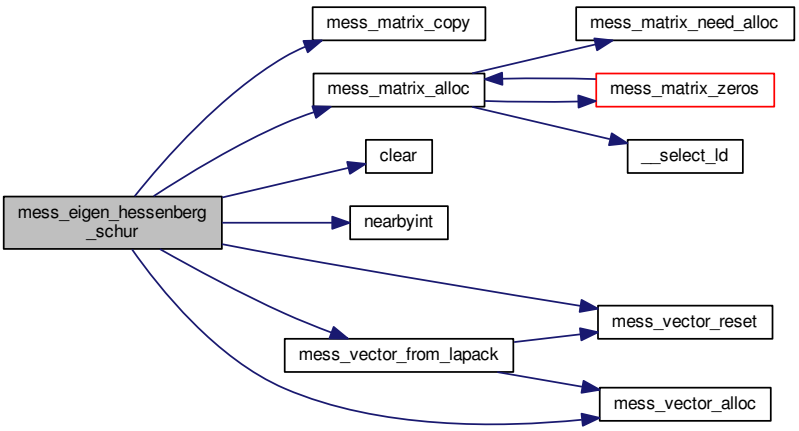
See also

mess_eigen_hessenberg_abs_largest
mess_eigen_schur_to_evd

Warning

It is not checked that `hess` is upper Hessenberg.

Here is the call graph for this function:



8.36.2.8 int mess_eigen_schur (mess_matrix *A*, mess_matrix *T*, mess_matrix *U*, mess_vector *EV*)

Parameters

in	*A*	input matrix to decompose
out	*T*	Schur form of A
out	*U*	orthogonal transformation (`NULL` if not needed)
out	*EV*	vector containing eigenvalues of A (`NULL` if not needed)

Returns

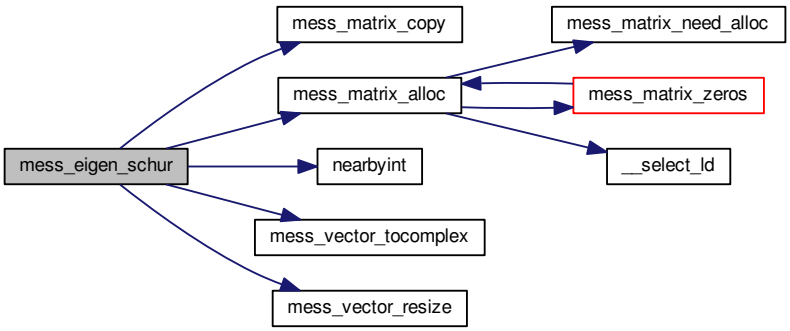
zero on success or a non-zero error value otherwise

The mess_eigen_schur function computes the real Schur decomposition of A

$$T = U^T A U$$
$$I_n = U^T U$$

using LAPACK's DGEES subroutine.

Here is the call graph for this function:



8.36.2.9 int mess_eigen_schur_complex (mess_matrix *A*, mess_matrix *T*, mess_matrix *U*, mess_vector *EV*)

Parameters

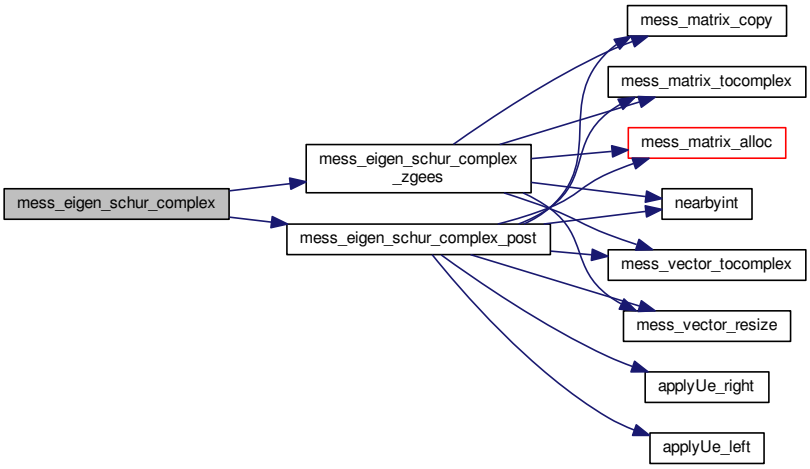
in	*A*	input matrix to decompose
out	*T*	Schur form of A
out	*U*	orthogonal transformation (NULL if not needed)
out	*EV*	vector containing eigenvalues of A

Returns

zero on success or a non-zero error value otherwise

The mess_eigen_schur_complex function is a wrapper around the two complex Schur decomposition subroutines mess_eigen_schur_complex_zgees and mess_eigen_schur_complex_post.

Here is the call graph for this function:



8.36.2.10 int mess_eigen_schur_complex_post (mess_matrix *A*, mess_matrix *T*, mess_matrix *U*, mess_vector *EV*)

Parameters

in	*A*	input matrix to decompose
out	*T*	Schur form of A
out	*U*	orthogonal transformation (NULL if not needed)
out	*EV*	vector containing eigenvalues of A

Returns

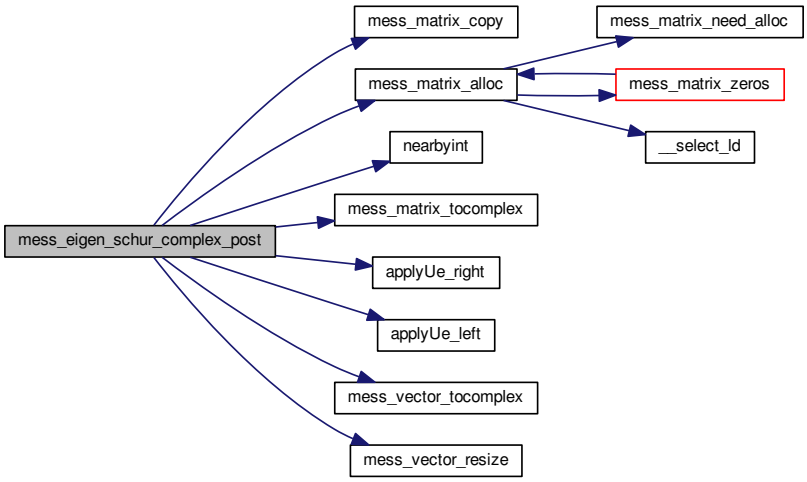
zero on success or a non-zero error value otherwise

The mess_eigen_schur_complex_post function computes the complex Schur decomposition of A:

$$T = U^H AU$$

using LAPACK's DGEES subroutine and a postprocessing step.

Here is the call graph for this function:



8.36.2.11 int mess_eigen_schur_complex_zgees (mess_matrix *A*, mess_matrix *T*, mess_matrix *U*, mess_vector *EV*)

Parameters

in	*A*	input matrix to decompose
out	*T*	Schur form of A
out	*U*	orthogonal transformation (NULL if not needed)
out	*EV*	vector containing eigenvalues of A

Returns

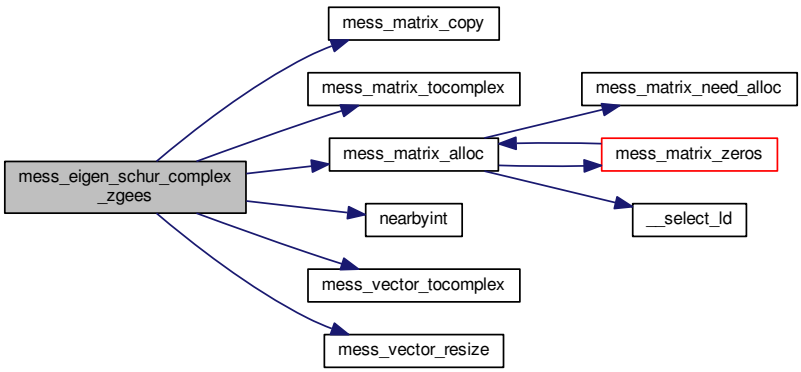
zero on success or a non-zero error value otherwise

The mess_eigen_schur_complex_zgees function computes the complex Schur decomposition of a A:

$$T = U^H AU$$

using LAPACK's ZGEES subroutine.

Here is the call graph for this function:



8.36.2.12 int mess_eigen_schur_to_evd (mess_matrix *T*, mess_matrix *Q*, mess_matrix *V*)

Parameters

in	*T*	input quasi upper triangular matrix of Schur decomposition
in	*Q*	input unitary transformation matrix of Schur decomposition
in	*V*	input matrix with eigenvectors as columns

Returns

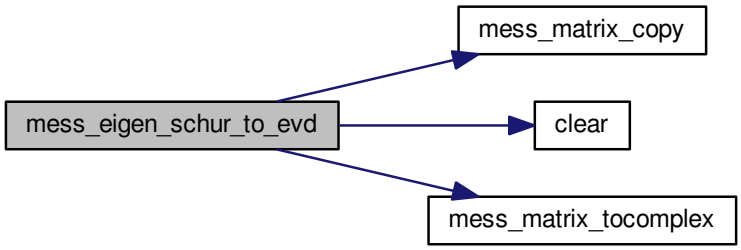
zero on success or a non-zero error value otherwise

The mess_eigen_schur_to_evd function computes the eigenvectors from a given Schur decomposition $A = Q * T * Q^H$. The eigenvectors are the columns of the matrix `V` at return. Pairs of complex conjugated eigenvectors are stored in consecutive columns. The function calls `DTREVC` / `ZTRVEC` from `LAPACK`.

See also

mess_eigen_hessenberg_schur

Here is the call graph for this function:



8.36.2.13 int mess_eigen_sign (mess_matrix *A*, mess_matrix *Z*, mess_sign_scale_t *scale*)

8.36.2.14 int mess_eigen_svd (mess_matrix *A*, mess_vector *S*, mess_matrix *U*, mess_matrix *V*)

Parameters

in	*A*	input matrix
out	*S*	vector containing singular values of A
out	*U*	left singular vectors
out	*V*	right singular vectors

Returns

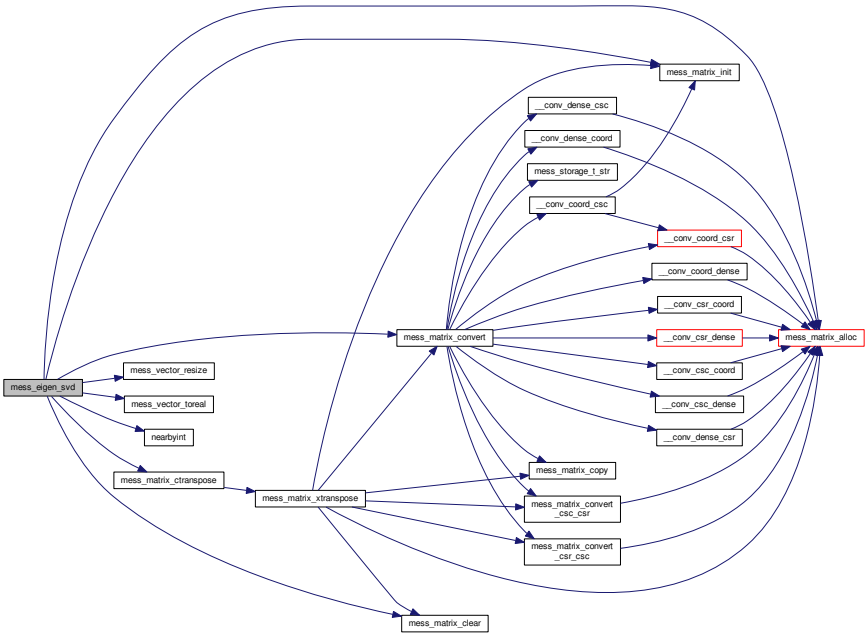
zero on success or a non-zero error value otherwise

The mess_eigen_svd function computes the singular value decompositon (SVD) of A:

$$A = USV^T.$$

If U or V is `NULL` the corresponding singular vector are not computed.
To compute the economy size SVD of a matrix please use mess_eigen_svd_econ.

Here is the call graph for this function:



8.36.2.15 int mess_eigen_svd_econ (mess_matrix *A*, mess_vector *S*, mess_matrix *U*, mess_matrix *V*)

Parameters

in	*A*	input matrix
out	*S*	vector containing singular values of A
out	*U*	left singular vectors
out	*V*	right singular vectors

Returns

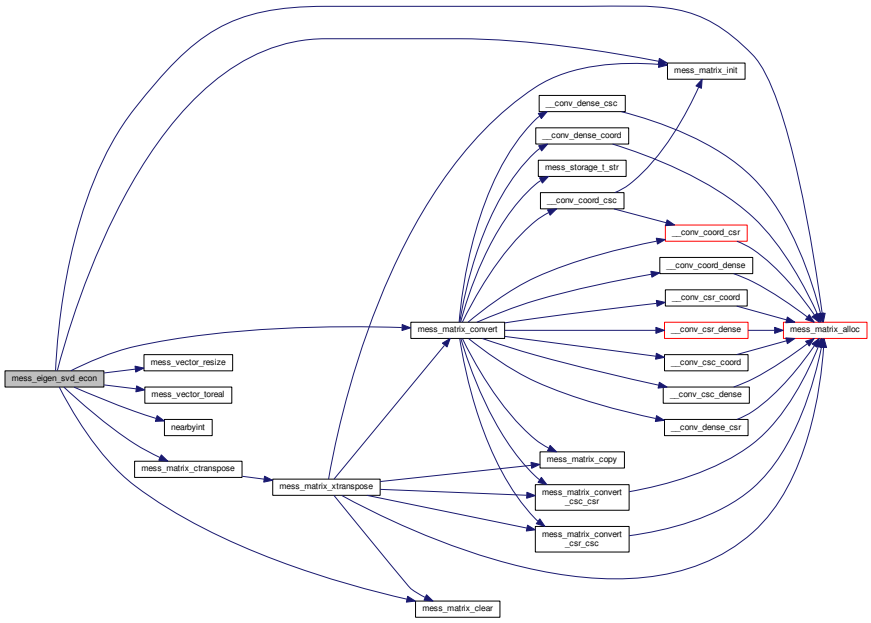
zero on success or a non-zero error value otherwise

The mess_eigen_svd_econ function computes the economy size singular value decompositon (SVD) of A:

$$A = USV^T.$$

If U or V is `NULL` the corresponding singular vector are not computed.
This function is similar to MATLAB `svd(X,0)` or `svd`(X,'econ').
To compute the normal SVD please use mess_eigen_svd.

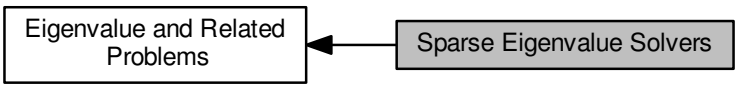
Here is the call graph for this function:



8.37 Sparse Eigenvalue Solvers

This category contains all necessary structures, definitions, enumerations and functions to solve the eigenvalue problem and related problems for sparse matrices.

Collaboration diagram for Sparse Eigenvalue Solvers:



Data Structures

- struct mess_eigen_arpack_options_t

  Options structure for `ARPACK` *.*

Macros

- #define MESS_EIGEN_ARPACK_DEFAULT { .tol = 0.0, .b0 = NULL, .ncv = 100, .maxit = 300, .which = MESS_EIGEN_ARPACK_LARGE_MAGNITUDE }

  Default initializer for the mess_eigen_arpack_options_t structure.

Enumerations

Functions

- int mess_eigen_arnoldi (mess_matrix A, mess_matrix B, mess_matrix K, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method with respect to A or $(A - UW)$.
- int mess_eigen_arnoldig (mess_matrix A, mess_matrix U, mess_matrix W, mess_direct E, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method with respect to $E^{-1}A$ or $E^{-1}(A - UW)$.
- int mess_eigen_arnoldig_inv (mess_direct A, mess_matrix U, mess_matrix W, mess_matrix E, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method on $A^{-1}E$ or $(A - UW)^{-1}E$.
- int mess_eigen_arnoldi_inv (mess_direct A, mess_matrix B, mess_matrix K, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method on A^{-1} or $(A - UW)^{-1}$.
- int mess_eigen_arnoldi_template (mess_mvpcall A, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method with respect to A.
- int mess_eigen_arnoldi_template_nrm (mess_mvpcall A, mess_int_t k, mess_vector sv, double *abseig)

  Compute absolut value of the largest eigenvalue using Arnoldi process.
- int mess_eigen_lanczos_template_nrm (mess_mvpcall A, mess_int_t k, mess_vector sv, mess_vector diag, mess_vector subdiag, mess_matrix V, double *abseig)

Lanczos method for 2-norm computations (real symmetric case).

- int mess_eigen_eigs (mess_matrix A, mess_int_t kp, mess_int_t km, mess_vector r, mess_vector ev)

  Compute some eigenvalues with the Arnoldi process.
- int mess_eigen_eiggs (mess_matrix A, mess_matrix E, mess_int_t kp, mess_int_t km, mess_vector r, mess_vector ev)

  Compute some generalized eigenvalues with the Arnoldi process.
- int mess_eigen_arpack (mess_matrix A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)

  An interface to the Arnoldi process from ARPACK.
- int mess_eigen_arpack_template (mess_mvpcall A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)

  An interface to the Arnoldi process from ARPACK *(template version).*
- int mess_eigen_arpack_lanczos (mess_matrix A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)

  An interface to the Lanczos process from ARPACK.
- int mess_eigen_arpack_lanczos_template (mess_mvpcall A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)

  An interface to the Lanczos process from ARPACK *(template version).*

8.37.1 Detailed Description

It includes different iterative methods to compute some approximate eigenvalues and their corresponding eigenvectors, like Arnoldi or Lanczos method, to compute only the largest absolute eigenvalue and its corresponding eigenvector with the aid of the power iteration or to compute the smallest absolute eigenvalue and its corresponding eigenvector using the inverse iteration.

The **power iteration** is a method to compute the largest absolute eigenvalue and its corresponding eigenvector. Therefor it computes in every iteration step $k = 0, 1, \cdots$

$$r_{k+1} = \frac{Ar_k}{\|Ar_k\|} = \frac{A^{k+1}r_0}{\|A^{k+1}r_0\|}$$

for a given initial vector $r_0 \in \mathbb{C}^n$ which fulfills $Ar_0 \neq 0$.
To compute the largest eigenvalue, the Rayleigh quotient is generated

$$\frac{r_k^H A r_k}{r_k^H r_k} = \frac{r_k^H r_{k+1}}{\|r_k\|_2^2}$$

converging to the largest eigenvalue.

This method is converging with a convergence ratio of $\left|\frac{\lambda_2}{\lambda_1}\right|$, where λ_2 denotes the second dominant eigenvalue.

The **inverse iteration** is a method to compute the smallest eigenvalue in absolute value and its corresponding eigenvector or to find an approximate eigenvector when an approximation to a corresponding eigenvalue (called shift) is already known.
Solving with a shift θ means to solve

$$\begin{array}{rcl} q & = & \frac{x}{\|x\|_2} \\ x & = & (A - \theta I)\backslash q \\ \lambda & = & q^T x \end{array}$$

with an initial guess x until a certain tolerance is reached. In this case λ is the eigenvalue closest to θ with corresponding eigenvector x.
The convergence rate of the inverse iteration is $|\frac{\lambda_1-\theta}{\lambda_2-\theta}|$, where $|\lambda_1 - \theta|^{-1}$ is the largest eigenvalue and $|\lambda_2 - \theta|^{-1}$ is the second largest eigenvalue of $(A - \theta I)^{-1}$.
if $\theta = 0$ $\frac{1}{\lambda}$ is the smallest absolute value and x its corresponding eigenvector.

The **Arnoldi method** is another algorithm to compute some eigenvalue and their corresponding eigenvalues. Therefor an orthonormal base of the Krylov space

$$\mathcal{K}_m(A,q) = span\{q, Aq, A^2q, \cdots, A^{m-1}q\}$$

is computed for an initial vector $q \in \mathbb{C}^n$.
Its output is an orthogonal matrix V and a Hessenberg matrix H and it holds

$$AV = VH.$$

Eigenvalues of H are also eigenvalues of A if $h_{m+1,m} = 0$. In case of small values $h_{m+1,m}$ eigenvalues of H or good approximations for eigenvalues of A.

The **Lanczos algorithm** can be considered as a simplified Arnoldi algorithm computing a tridiagonal and symmetric matrix

$$T_{m,m} = V_m^H A V_m,$$

which can be used to solve its eigenvalues $\lambda_i^{(m)}$ and their corresponding eigenvectors $x_i^{(m)}$. Eigenvalues of $T_{m,m}$ are approximate eigenvalues of A.

8.37.2 Macro Definition Documentation

8.37.2.1 #define MESS_EIGEN_ARPACK_DEFAULT { .tol = 0.0, .b0 = NULL, .ncv = 100, .maxit = 300, .which = MESS_EIGEN_ARPACK_LARGE_MAGNITUDE }

The `MESS_EIGEN_ARPACK_DEFAULT` macro initializes the mess_eigen_arpack_options_t structure to a useful set of default values.
The tolerance is set to the machine precision, the start vector to `NULL`, the maximum number of iterations to 300 and the number of Arnoldi vectors to 50.

8.37.3 Enumeration Type Documentation

8.37.3.1 enum mess_eigen_arpack_which_t

The mess_eigen_arpack_which_t enumeration selects the desired eigenvalues in the Arnoldi process performed by mess_eigen_arpack.

See also

mess_eigen_arpack_options_t
mess_eigen_arpack
mess_eigen_arpack_template

Remarks

The enumeration is only available if the preprocessor macro `MESS_HAVE_APRACK` is defined and **M.E.S.S.** is compiled with `ARPACK` support.

Enumerator

MESS_EIGEN_ARPACK_LARGE_MAGNITUDE `ARPACK` should compute eigenvalues of largest magnitude.

MESS_EIGEN_ARPACK_SMALL_MAGNITUDE `ARPACK` should compute eigenvalues of smallest magnitude.

MESS_EIGEN_ARPACK_LARGE_REALPART `ARPACK` should compute eigenvalues of largest real part.

Attention

This option is only valid for mess_eigen_arpack and mess_eigen_arpack_template.

MESS_EIGEN_ARPACK_SMALL_REALPART ARPACK should compute eigenvalues of smallest real part.

Attention

This option is only valid for mess_eigen_arpack and mess_eigen_arpack_template.

MESS_EIGEN_ARPACK_LARGE_IMAGPART ARPACK should compute eigenvalues of largest imaginary part.

Attention

This option is only valid for mess_eigen_arpack and mess_eigen_arpack_template.

MESS_EIGEN_ARPACK_SMALL_IMAGPART ARPACK should compute eigenvalues of smallest imaginary part.

Attention

This option is only valid for mess_eigen_arpack and mess_eigen_arpack_template.

MESS_EIGEN_ARPACK_LARGE_ALGEBRAIC ARPACK should compute the largest algebraic eigenvalues.

Attention

This option is only valid for mess_eigen_arpack_lanczos and mess_eigen_arpack_lanczos_template.

MESS_EIGEN_ARPACK_SMALL_ALGEBRAIC ARPACK should compute the smallest algebraic eigenvalues.

Attention

This option is only valid for mess_eigen_arpack_lanczos and mess_eigen_arpack_lanczos_template.

MESS_EIGEN_ARPACK_BOTH_ENDS ARPACK should compute the eigenvalues at both ends of the spectrum.

Attention

This option is only valid for mess_eigen_arpack_lanczos and mess_eigen_arpack_lanczos_template.

8.37.4 Function Documentation

8.37.4.1 int mess_eigen_arnoldi (mess_matrix *A*, mess_matrix *U*, mess_matrix *W*, mess_int_t *k*, mess_vector *sv*, mess_matrix *H*, mess_matrix *V*)

Parameters

in	*A*	input matrix for which the Arnoldi algorithm is supposed to run
in	*U*	input matrix to work with $(A - UW)$
in	*W*	input matrix to work with $(A - UW)$
in	*k*	input number of Arnoldi steps (usually $k \ll n$)
in	*sv*	input start vector
out	*H*	$(k + 1 \times k)$ upper Hessenberg matrix
out	*V*	$(n \times k + 1)$ orthogonal matrix

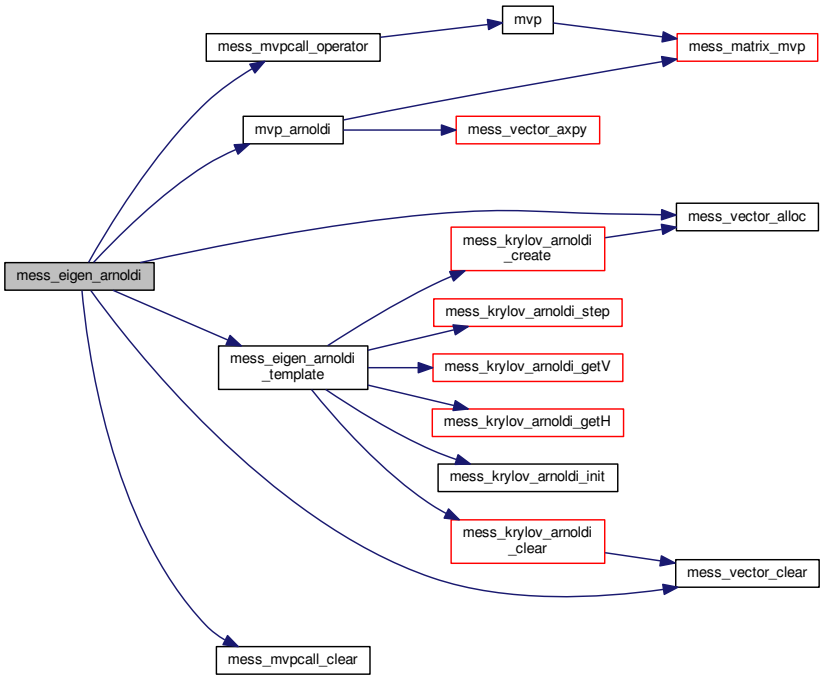
Returns

zero on success or a non-zero error value otherwise

The mess_eigen_arnoldi function computes matrices H and V (if $V \neq \texttt{NULL}$) fullfilling

$$\begin{aligned} V(:,1) &= \frac{sv}{\|sv\|}, \\ V^T V &= I, \\ AV(:,1:k) &= VH. \end{aligned}$$

Here is the call graph for this function:



8.37.4.2 int mess_eigen_arnoldi_inv (mess_direct *A*, mess_matrix *U*, mess_matrix *W*, mess_int_t *k*, mess_vector *sv*, mess_matrix *H*, mess_matrix *V*)

Parameters

in	*A*	input decomposed matrices for which the Arnoldi algorithm is supposed to run
in	*U*	input matrix for $(A - UW)^{-1}$
in	*W*	input matrix for $(A - UW)^{-1}$
in	*k*	input number of Arnoldi steps (usually $k \ll n$)
in	*sv*	input start vector
out	*H*	$(k + 1 \times k)$ upper Hessenberg matrix
out	*V*	$(n \times k + 1)$ orthogonal matrix

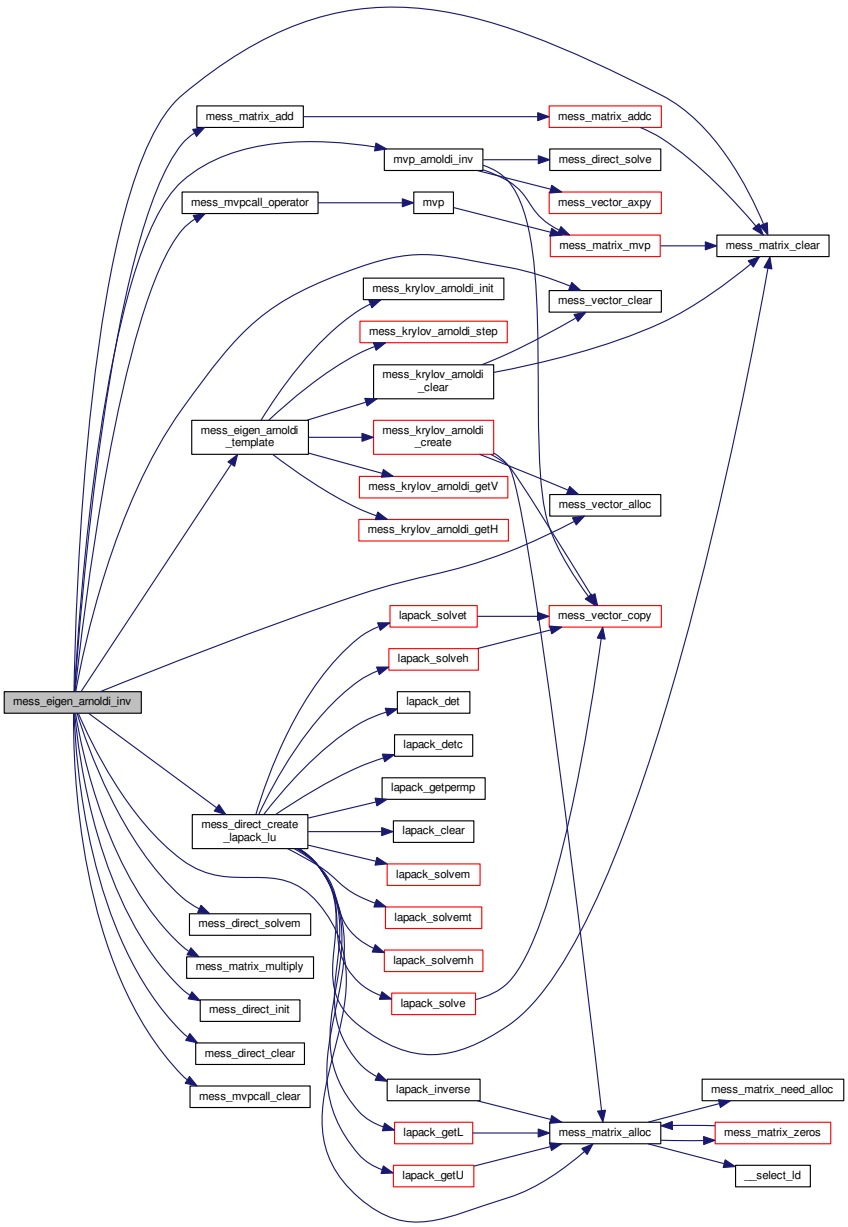
Returns

zero on success or a non-zero error value otherwise

The mess_eigen_arnoldi_inv function computes matrices H and V (if $V \neq \texttt{NULL}$) fullfilling

$$\begin{aligned} V(:,1) &= \frac{sv}{\|sv\|}, \\ V^T V &= I, \\ A^{-1} V(:,1:k) &= VH. \end{aligned}$$

Here is the call graph for this function:



8.37.4.3 int mess_eigen_arnoldi_template (mess_mvpcall *A*, mess_int_t *k*, mess_vector *sv*, mess_matrix *H*, mess_matrix *V*)

Parameters

in	*A*	input operator applying the A matrix for which the Arnoldi algorithm is supposed to run
in	*k*	input number of Arnoldi steps (usually $k \ll n$)
in	*sv*	input start vector
out	*H*	$(k+1 \times k)$ upper Hessenberg matrix
out	*V*	$(n \times k+1)$ orthogonal matrix

Returns

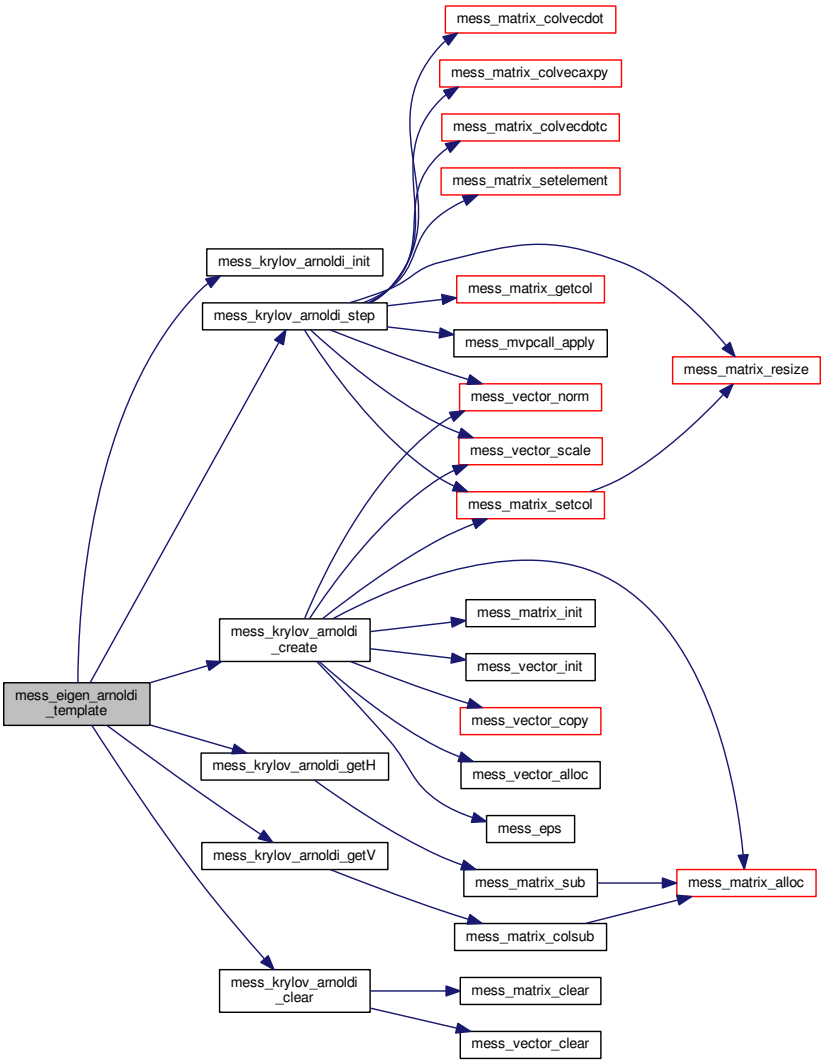
zero on success or a non-zero error value otherwise

The mess_eigen_arnoldi_template function computes matrices H and V (if $V \neq \texttt{NULL}$) fullfilling

$$\begin{aligned} V(:,1) &= \frac{sv}{\|sv\|}, \\ V^T V &= I, \\ AV(:,1:k) &= VH. \end{aligned}$$

If an early breakdown occurs then it holds that $AV = VH$. The matrix A is given as a function handler which computes $y = Ax$.

Here is the call graph for this function:



8.37.4.4 int mess_eigen_arnoldi_template_nrm (mess_mvpcall *A*, mess_int_t *k*, mess_vector *sv*, double * *abseig*)

Parameters

in	*A*	input operator applying the A matrix for which the Arnoldi algorithm is supposed to run
in	*k*	input number of Arnoldi steps (usually $k \ll n$)
in	*sv*	input start vector, non zero
out	*abseig*	output largest eigenvalue in absolute value

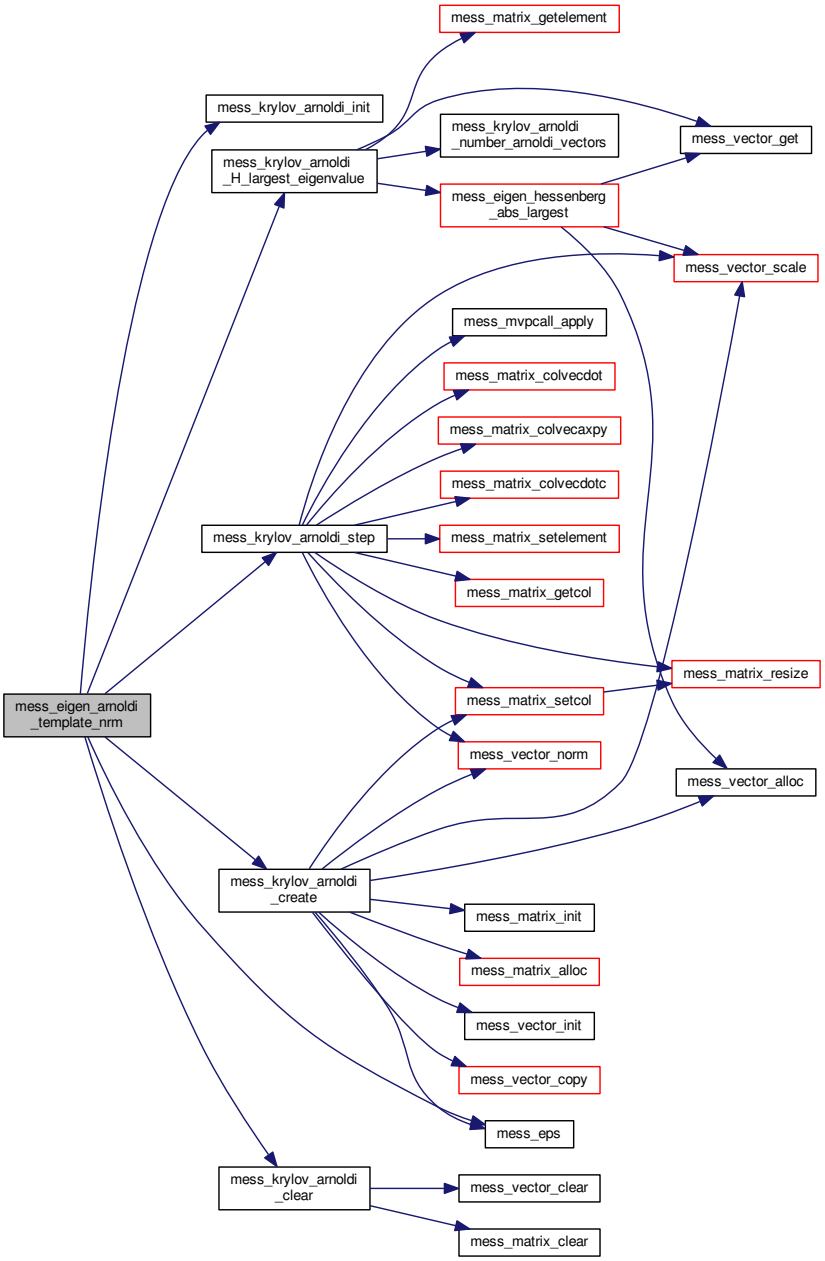
Returns

zero on success or a non-zero error value otherwise

The mess_eigen_arnoldi_template_nrm computes an approximation of the largest eigenvalue in magnitude and returns the magnitude. The function use an arnoldi process and computes the ritz values.

The algorithms stops if the maximum iteration number k is reached or if the error estimation is smaller than $neps()$ where n is the order of the operator. The matrix A is given as a function handler which computes $y = Ax$.

Here is the call graph for this function:



8.37.4.5 int mess_eigen_arnoldig (mess_matrix *A*, mess_matrix *U*, mess_matrix *W*, mess_direct *E*, mess_int_t *k*, mess_vector *sv*, mess_matrix *H*, mess_matrix *V*)

Parameters

in	*A*	input matrix for which the Arnoldi algorithm is supposed to run
in	*E*	input solver to solve $Ex = b$

Parameters

in	*U*	input matrix to work with $(A - UW)$
in	*W*	input matrix to work with $(A - UW)$
in	*k*	input number of Arnoldi steps (usually $k \ll n$)
in	*sv*	input start vector
out	*H*	$(k + 1 \times k)$ upper Hessenberg matrix
out	*V*	$(n \times k + 1)$ orthogonal matrix

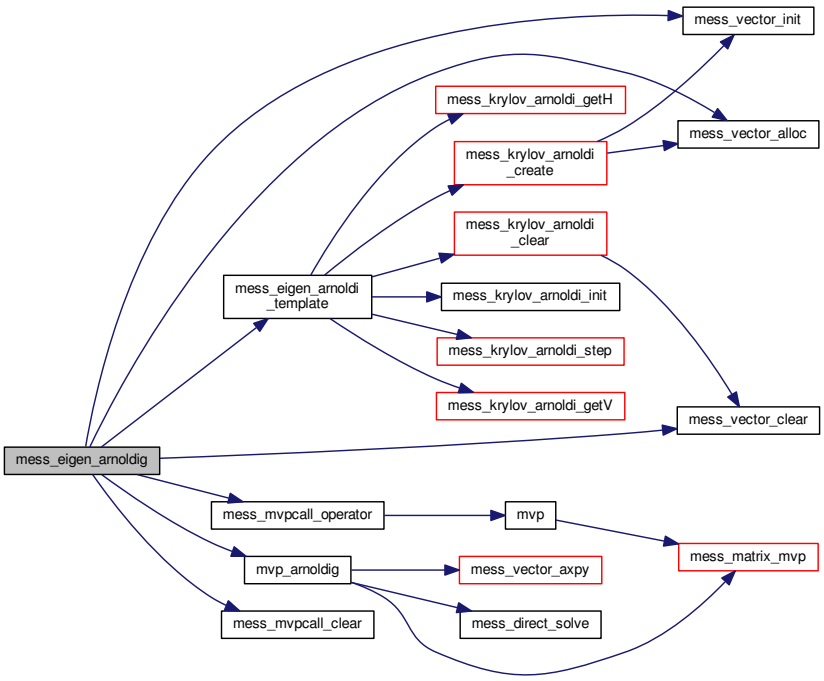
Returns

zero on success or a non-zero error value otherwise

The mess_eigen_arnoldig function computes matrices H and V (if $V \neq \mathrm{NULL}$) fullfilling

$$\begin{aligned} V(:,1) &= \frac{sv}{\|sv\|}, \\ V^T V &= I, \\ AV(:,1:k) &= VH. \end{aligned}$$

Here is the call graph for this function:



8.37.4.6 int mess_eigen_arnoldig_inv (mess_direct *A*, mess_matrix *U*, mess_matrix *W*, mess_matrix *E*, mess_int_t *k*, mess_vector *sv*, mess_matrix *H*, mess_matrix *V*)

Parameters

in	*A*	input decomposed matrices for which the Arnoldi algorithm is supposed to run

Parameters

in	*E*	input mass matrix
in	*U*	input matrix for $(A - UW)^{-1}$
in	*W*	input matrix for $(A - UW)^{-1}$
in	*k*	input number of Arnoldi steps (usually $k \ll n$);
in	*sv*	input start vector
out	*H*	$(k + 1 \times k)$ upper Hessenberg matrix
out	*V*	$(n \times k + 1)$ orthogonal matrix

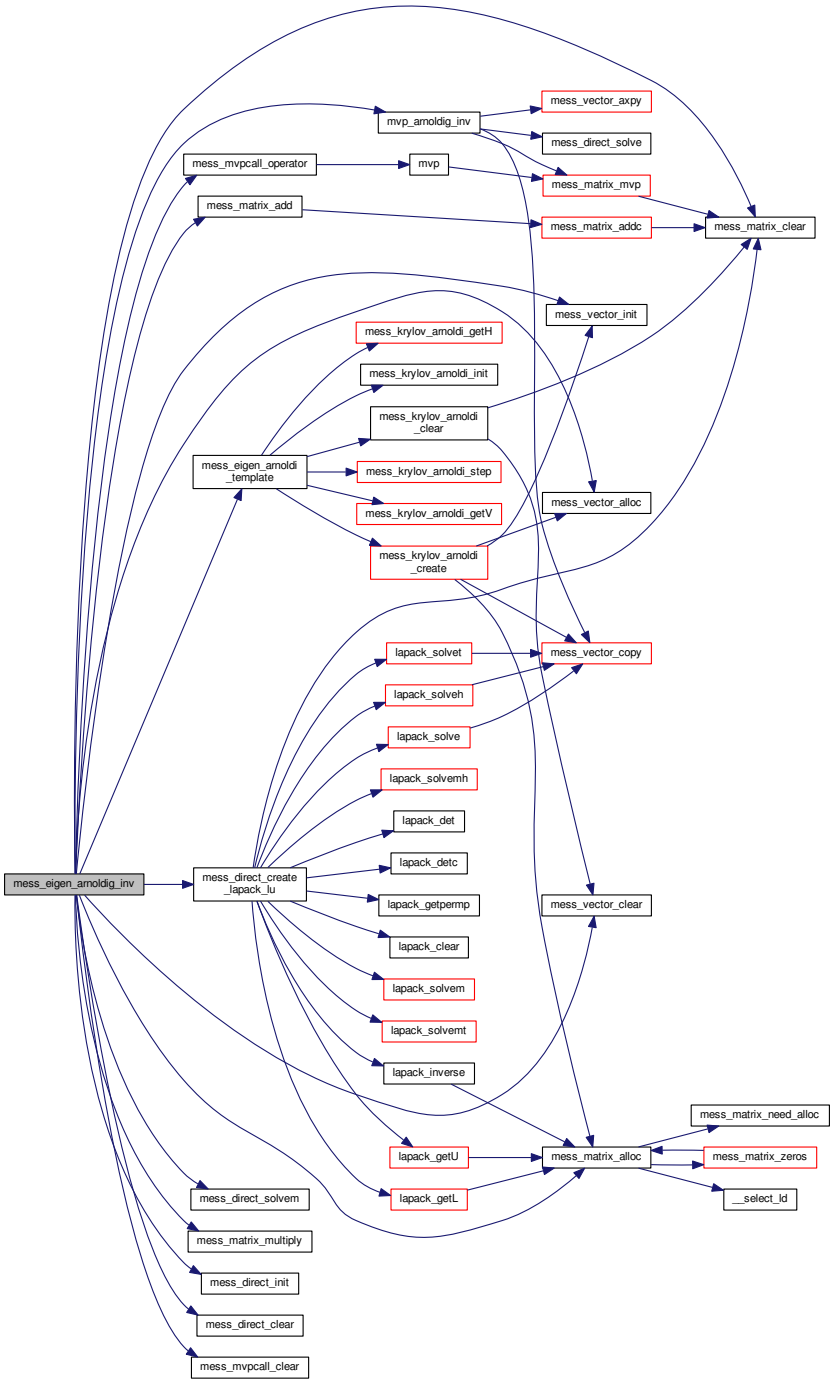
Returns

zero on success or a non-zero error value otherwise

The mess_eigen_arnoldig_inv function computes matrices H and V (if $V \neq \mathtt{NULL}$) fullfilling

$$\begin{aligned} V(:,1) &= \frac{sv}{\|sv\|}, \\ V^T V &= I, \\ A^{-1} V(:,1:k) &= VH. \end{aligned}$$

Here is the call graph for this function:



8.37.4.7 int mess_eigen_arpack (mess_matrix *A*, mess_int_t *nev*, mess_eigen_arpack_options_t *opt*, mess_vector *ev*, mess_matrix *V*)

Parameters

in	*A*	input matrix to compute its eigenvalues from
in	*nev*	input number of desired eigenvalues

Parameters

in	*opt*	input options structure to configure the Arnoldi process
out	*ev*	vector containing the computed eigenvalues
out	*V*	matrix containing the computed ritz vectors corresponding to the eigenvalues (if not desired set it to NULL)

Returns

zero on success or a non zero error code

The mess_eigen_arpack function calls ARPACK [21] to solve a large scale eigenvalue problem.
The internal Arnoldi process is configured using the opt structure. Please adjust the number of the Arnoldi vectors in the mess_eigen_arpack_options_t structure such that it fits to the number of desired eigenvalues.
The number ncv of the Arnoldi vectors and the number of desired eigenvalues must fulfill

$$ncv - nev \leq 2.$$

If only an operator is given instead of a matrix please use mess_eigen_arpack_template.

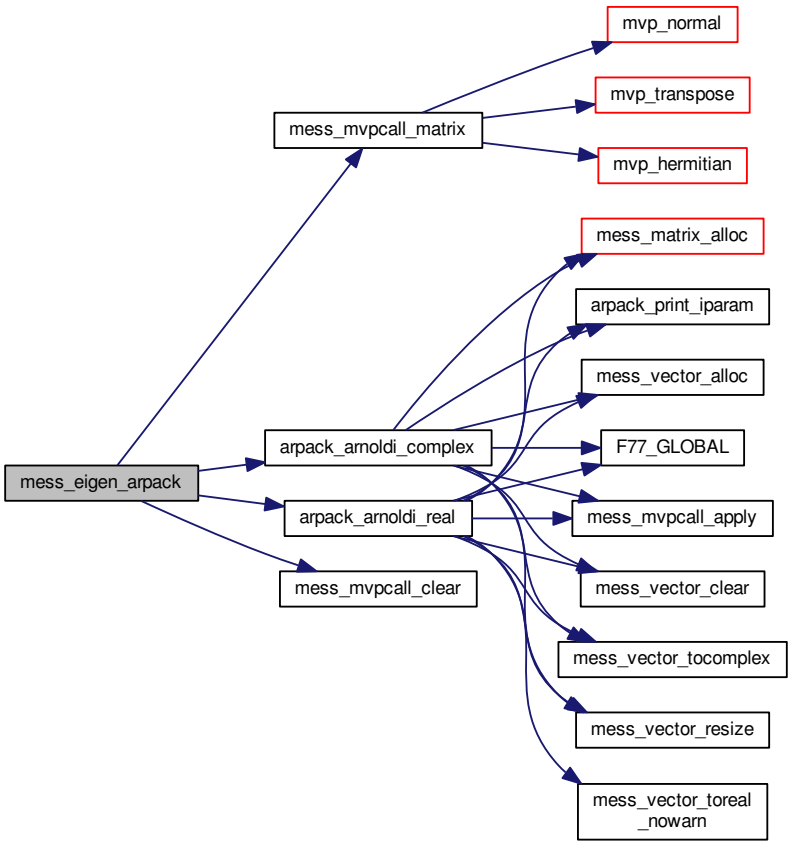
See also

mess_eigen_arpack_template
mess_eigen_arpack_options_t
mess_eigen_arpack_which_t

Remarks

The function does only exists if `MESS_HAVE_ARPACK` is set and **M.E.S.S.** is compiled with `ARPACK` support.

Here is the call graph for this function:



8.37.4.8 int mess_eigen_arpack_lanczos (mess_matrix *A*, mess_int_t *nev*, mess_eigen_arpack_options_t *opt*, mess_vector *ev*, mess_matrix *V*)

Parameters

in	*A*	input matrix to compute its eigenvalues from
in	*nev*	input number of desired eigenvalues
in	*opt*	input options structure to configure the Lanczos process
out	*ev*	vector containing the computed eigenvalues
out	*V*	matrix containing the computed ritz vectors corresponding to the eigenvalues (if not desired set it to `NULL`)

Returns

zero on success or a non zero error code

The mess_eigen_arpack_lanczos function calls ARPACK [21] to solve a large scale eigenvalue problem.
The internal Lanczos process is configured using the opt structure. Please adjust the number of the Lanczos vectors in the mess_eigen_arpack_options_t structure such that it fits to the number of desired eigenvalues.
The number `ncv` of the Lanczos vectors and the number of desired eigenvalues must fulfill

$$ncv - nev \leq 1.$$

If only an operator is given instead of a matrix please use mess_eigen_arpack_lanczos_template.

Attention

The input matrix must be real symmetric, i.e. $A = A^T$.
If the matrix is not symmetric please use mess_eigen_arpack instead.

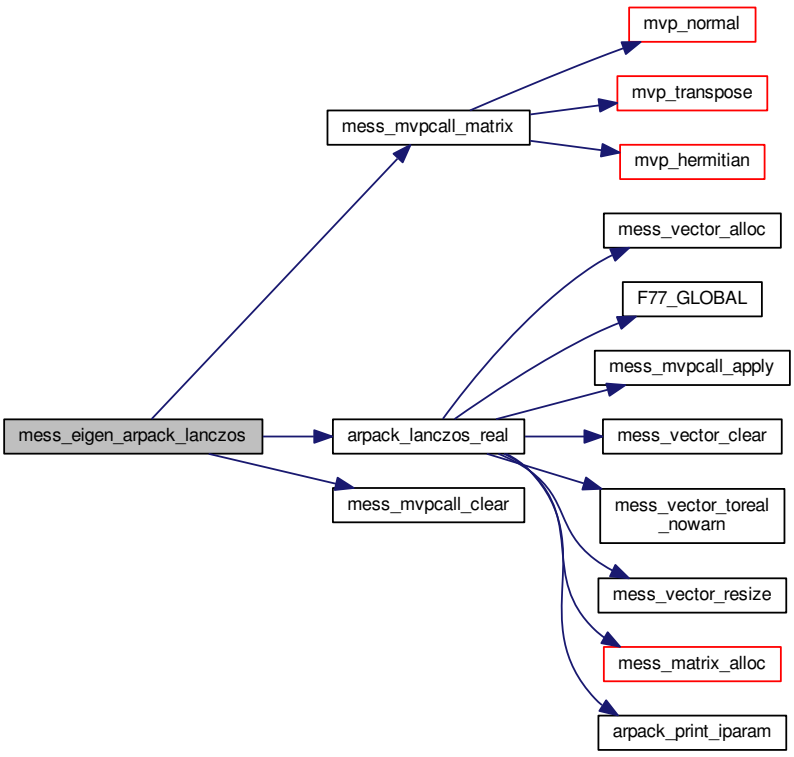
See also

mess_eigen_arpack_lanczos_template
mess_eigen_arpack_options_t
mess_eigen_arpack_which_t

Remarks

The function does only exists if `MESS_HAVE_ARPACK` is set and **M.E.S.S.** is compiled with ARPACK support.

Here is the call graph for this function:



8.37.4.9 int mess_eigen_arpack_lanczos_template (mess_mvpcall *A*, mess_int_t *nev*, mess_eigen_arpack_options_t *opt*, mess_vector *ev*, mess_matrix *V*)

Parameters

in	*A*	input matrix-vector product object defining the operator
in	*nev*	input number of desired eigenvalues
in	*opt*	input options structure to configure the Arnoldi process
out	*ev*	vector containing the computed eigenvalues
out	*V*	matrix containing the computed ritz vectors corresponding to the eigenvalues (if not desired set it to NULL)

Returns

zero on success or a non zero error code

The mess_eigen_arpack_lanczos_template function calls ARPACK [21] to solve a large scale eigenvalue problem. The internal Lanczos process is configured using the opt structure. Please adjust the number of the Lanczos vectors in the mess_eigen_arpack_options_t structure such that it fits to the number of desired eigenvalues.
The number ncv of the Lanczos vectors and the number of desired eigenvalues must fulfill

$$ncv - nev \geq 1.$$

If a matrix given instead of an operator the mess_eigen_arpack function can be used as well.

Attention

The input matrix must be real symmetric, i.e. $A = A^T$.
If the matrix is not symmetric please use mess_eigen_arpack instead.

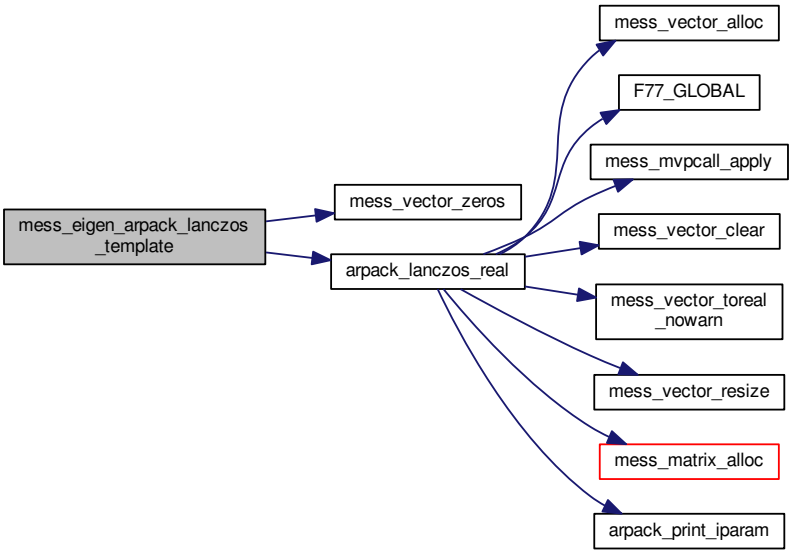
See also

mess_eigen_arpack_lanczos
mess_eigen_arpack_options_t
mess_eigen_arpack_which_t

Remarks

The function does only exists if MESS_HAVE_ARPACK is set and **M.E.S.S.** is compiled with ARPACK support.

Here is the call graph for this function:



8.37.4.10 int mess_eigen_arpack_template (mess_mvpcall *A*, mess_int_t *nev*, mess_eigen_arpack_options_t *opt*, mess_vector *ev*, mess_matrix *V*)

Parameters

in	*A*	input matrix-vector product object defining the operator
in	*nev*	input number of desired eigenvalues
in	*opt*	input options structure to configure the Arnoldi process
out	*ev*	vector containing the computed eigenvalues
out	*V*	matrix containing the computed ritz vectors corresponding to the eigenvalues (if not desired set it to NULL)

Returns

zero on success or a non zero error code

The mess_eigen_arpack_template function calls ARPACK [21] to solve a large scale eigenvalue problem.
The internal Arnoldi process is configured using the opt structure. Please adjust the number of the Arnoldi vectors in the mess_eigen_arpack_options_t structure such that it fits to the number of desired eigenvalues.
The number ncv of the Arnoldi vectors and the number of desired eigenvalues must fulfill

$$ncv - nev \leq 2.$$

If a matrix given instead of an operator the mess_eigen_arpack function can be used as well.

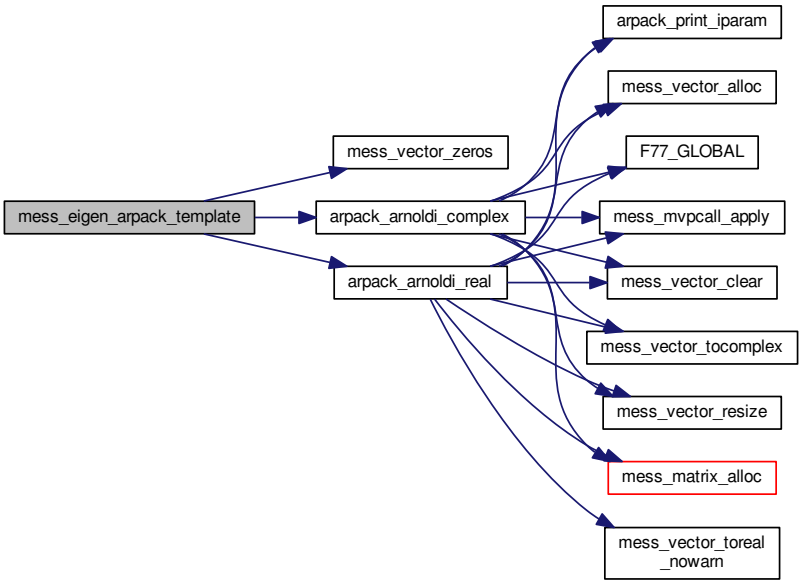
See also

mess_eigen_arpack
mess_eigen_arpack_options_t
mess_eigen_arpack_which_t

Remarks

The function does only exists if `MESS_HAVE_ARPACK` is set and **M.E.S.S.** is compiled with ARPACK support.

Here is the call graph for this function:



8.37.4.11 int mess_eigen_eiggs (mess_matrix *A*, mess_matrix *E*, mess_int_t *kp*, mess_int_t *km*, mess_vector *r*, mess_vector *ev*)

Parameters

in	*A*	input matrix
in	*E*	input matrix
in	*kp*	input number of Arnoldi steps with respect to A
in	*km*	input number of Arnoldi steps with respect to A^{-1}
in	*r*	input start vector (ones if `NULL`)
out	*ev*	vector containing computed eigenvalues

Returns

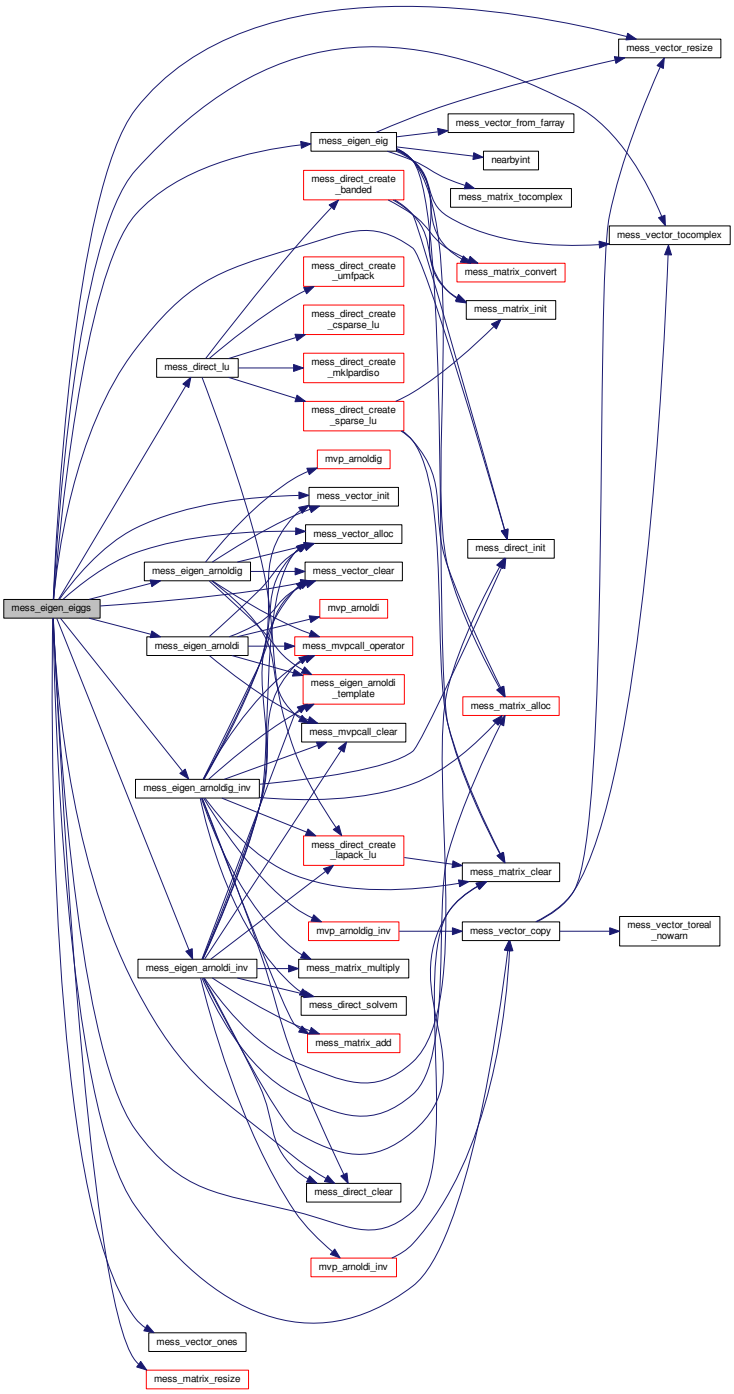
zero on success or a non-zero error value otherwise

The mess_eigen_eiggs function computes some eigenvalue approximations stored in `ev` for the generalized eigenvalue problem

$$Ax = \lambda Ex$$

with the Arnoldi process.

Here is the call graph for this function:



8.37.4.12 int mess_eigen_eigs (mess_matrix *A*, mess_int_t *kp*, mess_int_t *km*, mess_vector *r*, mess_vector *ev*)

Parameters

in	*A*	input matrix
in	*kp*	input number of Arnoldi steps with respect to A
in	*km*	input number of Arnoldi steps with respect to A^{-1}
in	*r*	input start vector (ones if NULL)
out	*ev*	vector containing computed eigenvalues

Returns

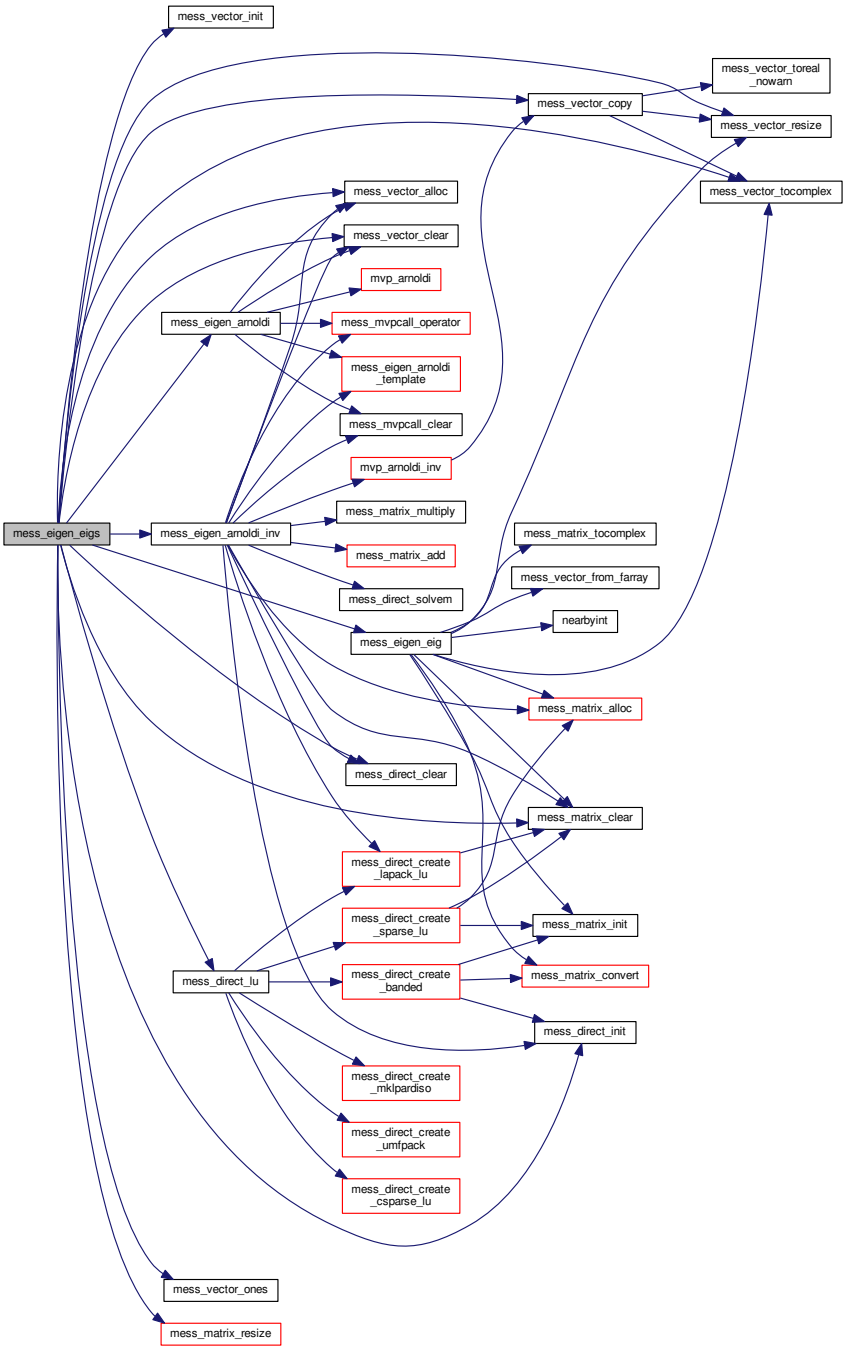
zero on success or a non-zero error value otherwise

The mess_eigen_eigs function computes some eigenvalue approximations stored in ev for the eigenvalue problem

$$Ax = \lambda x$$

with the Arnoldi process.

Here is the call graph for this function:



8.37.4.13 int mess_eigen_lanczos_template_nrm (mess_mvpcall *A*, mess_int_t *k*, mess_vector *sv*, mess_vector *diag*, mess_vector *subdiag*, mess_matrix *V*, double ∗ *abseig*)

Parameters

in	*A*	input operator applying the A matrix for which the Lanczos algorithm is supposed to run
in	*k*	input number of Lanczos steps (usually $k \ll n$)
in	*sv*	input start vector

Parameters

out	*diag*	k-vector containing diagonal elements of T
out	*subdiag*	$(k-1)$-vector containing subdiagonal elements of T
out	*V*	$(n \times k)$ orthogonal matrix
out	*abseig*	absolute value of the largest eigenvalue

Returns

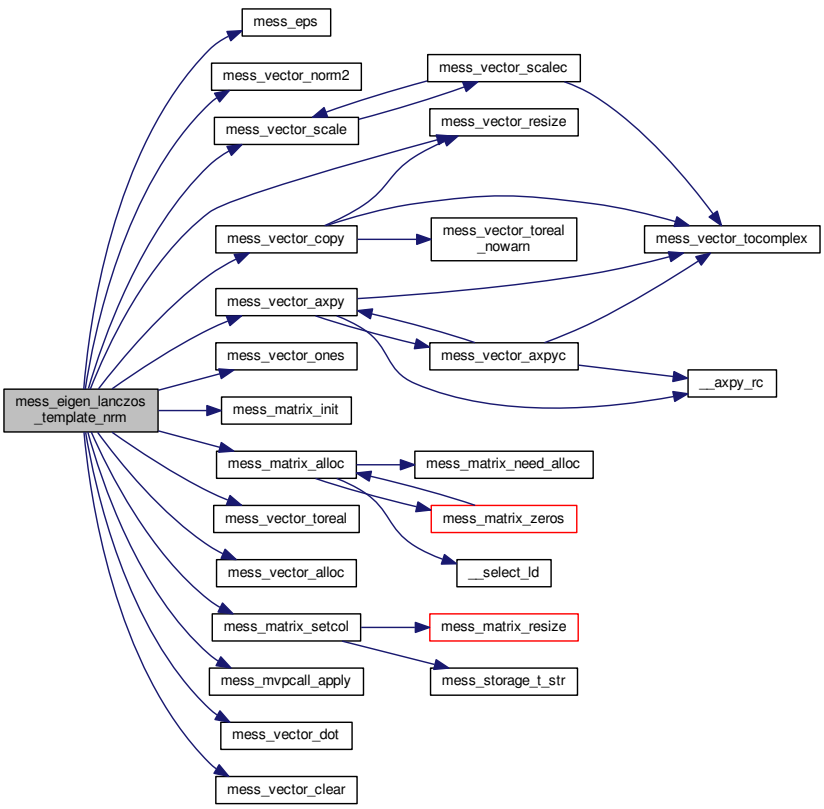
zero on success or a non-zero error value otherwise

The mess_eigen_lanczos_template_nrm function produces matrices T (symmetric, tridiagonal) and V (if $V \neq$ NULL) fullfilling

$$\begin{aligned} V(:,1) &= \frac{sv}{\|sv\|}, \\ V^T V &= I, \\ AV(:,1:k) &= VT(:,1:k). \end{aligned}$$

T is stored as k-vector diag and $(k-1)$-vector subdiag.
The algorithms stops if the maximum iteration number k is reached or the relative difference between the largest eigenvalues of two step is smaller than $\sqrt{eps}$.

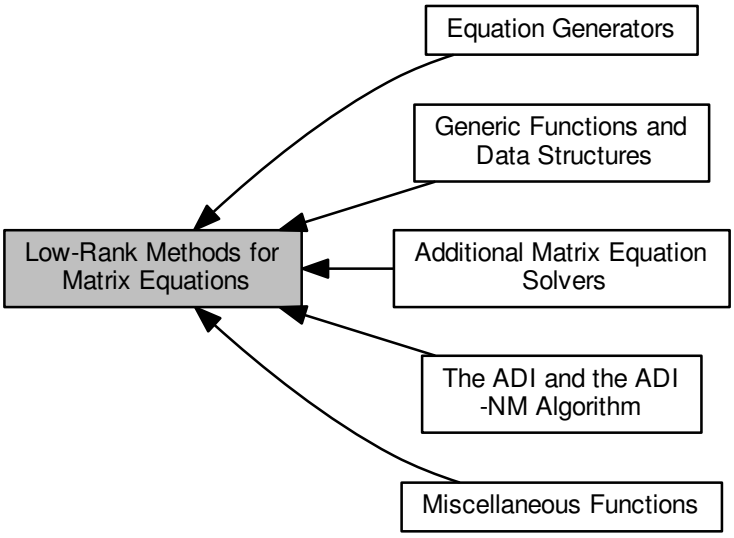
Here is the call graph for this function:



8.38 Low-Rank Methods for Matrix Equations

The Low-Rank Cholesky Factor (LRCF) are the state of the art large-scale matrix equations solvers.

Collaboration diagram for Low-Rank Methods for Matrix Equations:



Modules

- Generic Functions and Data Structures

  This category contains all necessary data structures, type definitions, enumerations and functions to handle LRCF methods.
- The ADI and the ADI-NM Algorithm

  This category contains solving algorihms and useful subroutines.
- Equation Generators

  This category contains functions to define and to build Lyapunov or Riccati Equations.
- Additional Matrix Equation Solvers

  This category contains additional matrix equation solvers, which are not based on the ADI process.
- Miscellaneous Functions

  This category contains several function that do not fit in any other category.

8.38.1 Detailed Description

We provide implementations of this concept to solve Lyapunov Equations

$$AXE^T + EXA^T + BB^T = 0$$

or

$$A^TXE + E^TXA + BB^T = 0$$

and Riccati Equations

$$AXE^T + EXA^T - BB^T + EXC^TCXE^T = 0$$

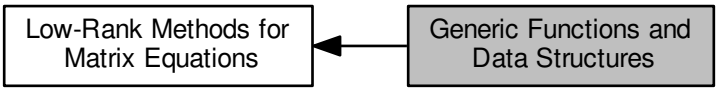
or

$$A^TXE + E^TXA - C^TC + E^TXBB^TCXE = 0.$$

8.39 Generic Functions and Data Structures

This category contains all necessary data structures, type definitions, enumerations and functions to handle LRCF methods.

Collaboration diagram for Generic Functions and Data Structures:



Data Structures

- struct __mess_lrcfadi_step

  Structure to pass information to the step function inside the LRCF iterations.
- struct mess_options_st

  Options for the LRCF-ADI algorithm family.
- struct mess_status_st

  Structure/object to save the final status of a LRCF algorithm.
- struct mess_equation_st

  Structure defining an equation as a set of function pointers.

Typedefs

- typedef struct __mess_lrcfadi_step mess_lrcfadi_step

  Structure to pass information to the step function inside the LRCF iterations.
- typedef struct mess_options_st mess_options_st

  Options for the LRCF-ADI algorithm family.
- typedef mess_options_st ∗ mess_options
- typedef struct mess_status_st mess_status_st

  Structure/object to save the final status of a LRCF algorithm.
- typedef mess_status_st ∗ mess_status
- typedef struct mess_equation_st ∗ mess_equation

  Type definition for the definition of an equation.

Enumerations

Functions

- int mess_equation_init (mess_equation ∗eqn)

  Initialize an empty mess_equation object.
- int mess_equation_clean (mess_equation eqn)

  Call cleanup functions for all internally generated data.

- int mess_equation_clear (mess_equation *eqn)

  Clean a mess_equation structure.
- int mess_equation_set_rhs (mess_equation eqn, mess_options opt, mess_matrix rhs)

  Set a new right hand side in a mess_equation object.
- int mess_equation_set_rhs2 (mess_equation eqn, mess_matrix rhs)

  Set a new right hand side in a mess_equation object without additional checks.
- int mess_status_init (mess_status *status)

  Initialize a mess_status object.
- int mess_status_clear (mess_status *status)

  Clear a mess_status object.
- int mess_status_print (mess_status stat)

  Print a mess_status object to the standard output.
- int mess_status_printfull (mess_status stat)

  Print a mess_status object to the standard output including embbed objects.
- int mess_status_copy (mess_status src, mess_status dest)

  Copy a mess_status object to another one.
- mess_int_t mess_status_memsize (mess_status status)

  Return the size of a mess_status in bytes.
- int mess_status_write (mess_status stat, const char *filename)

  Save the information of a mess_status object to a text file.
- int mess_options_init (mess_options *opt)

  Initialize LRCF-ADI options with default values.
- int mess_options_clear (mess_options *opt)

  Clear a mess_options object.
- int mess_options_print (mess_options opt)

  Display a mess_options object on standard output.
- int mess_options_set_proj_space (mess_options opt, mess_matrix proj_space)

  Copy the mess_matrix `proj_space`.
- int mess_options_unset_proj_space (mess_options opt)

  Delete the field `proj_space` from mess_options instance.
- int mess_options_set_K0 (mess_options opt, mess_matrix K0)

  Copy the mess_matrix `K0`.
- int mess_options_unset_K0 (mess_options opt)

  Delete the field `K0` from mess_options instance.
- int mess_options_set_W (mess_options opt, mess_matrix W)

  Copy the mess_matrix `W`.
- int mess_options_unset_W (mess_options opt)

  Delete the field `W` from mess_options instance.
- mess_int_t mess_options_memsize (mess_options opt)

  Return the size of a mess_options in bytes.
- int mess_options_copy (mess_options src, mess_options dest)

  Copy a mess_options object to another one.

8.39.1 Detailed Description

The data structures are implemented to set options for the LRCF-algorithm, to define the type of equation, to save the final status of the algorithm,to configure the LRCF-ADI based solver and to pass information to the step function inside the ADI iteration.
Additionally there are type definitions for all structures.
This category also contains initialization, clean up, destroy, save, copy, fill up, print and display functions.

8.39.2 Typedef Documentation

8.39.2.1 typedef struct mess_equation_st* mess_equation

The mess_status is a type definition for the definition of an equation.

8.39.2.2 typedef struct __mess_lrcfadi_step mess_lrcfadi_step

The mess_lrcfadi_step structure is used to pass information to a step function inside the ADI iteration.
The step function, if set, is executed at the end of every ADI or Newton iteration.
This structure is passed to the step function to provide information about the current state of the iteration.
The stop_user component can be used to provide a user supplied cancellation criterion.

Attention

Interal use only.

8.39.2.3 typedef mess_options_st* mess_options

8.39.2.4 typedef struct mess_options_st mess_options_st

The mess_options_st structure is used to configure the LRCF-ADI based solvers.
It contains information about the cancellation criteria, maximum iteration numbers, the shift parameter computation and similar options. It is used by all LRCF related solvers.
The initialization is done using mess_options_init and it is freed after usage by mess_options_clear. An overview about the current setting can be printed to the standard output using mess_options_print.

See also

mess_lradi
mess_lrnm

8.39.2.5 typedef mess_status_st* mess_status

8.39.2.6 typedef struct mess_status_st mess_status_st

8.39.3 Enumeration Type Documentation

8.39.3.1 enum mess_equation_t

The mess_equation_t enumeration type is used to define the type of the equation. Mostly used internally.

Enumerator

MESS_EQN_NONE No equation (only for initialization)

MESS_EQN_LYAP Lyapunov Equation

MESS_EQN_GLYAP Generalized Lyapunov Equation

MESS_EQN_RICCATI Riccati Equation

MESS_EQN_GRICCATI Generalized Riccati Equation

8.39.3.2 enum mess_memusage_t

The mess_memusage_t enumeration type is used to define the amount of memory the solvers of equation.

See also

mess_equation_std_lyap
mess_equation_std_riccati

Enumerator

MESS_MEMORY_LOW Lowest memory consuming variant.

MESS_MEMORY_MID Average memory consuming variant.

MESS_MEMORY_HIGH Highest memory using variant.

8.39.3.3 enum mess_parameter_t

The mess_parameter_t enumeration type is used to select the parameters inside the ADI process.
At the moment we have implemented three different parameter types. The parameters are computed using mess_lrcfadi_parameter.

Enumerator

MESS_LRCFADI_PARA_MINMAX Compute complex parameters using the heuristic for the min-max problem.

MESS_LRCFADI_PARA_MINMAX_REAL Compute real parameters using the heuristic for the min-max problem.

MESS_LRCFADI_PARA_WACHSPRESS Compute Wachspress parameters.

MESS_LRCFADI_PARA_ADAPTIVE_V Compute adaptive shifts with V as projection space.

MESS_LRCFADI_PARA_ADAPTIVE_Z Compute adaptive shifts with Z as projection space.

8.39.3.4 enum mess_residual_t

The mess_residual_t enumeration type is used to define the algorithm for computation of the residual.

See also

mess_equation_std_lyap
mess_equation_std_riccati

Enumerator

MESS_RESIDUAL_INDEFINITE Residual via Residual Factor Updates (fastest variant).

MESS_RESIDUAL_SPECTRAL Compute Largest Eigenvalue iteratively.

8.39.4 Function Documentation

8.39.4.1 int mess_equation_clean (mess_equation *eqn*)

Parameters

in,out	*eqn*	mess_equation object to clean

Returns

zero on success or a non zero error code

The mess_equation_clean function cleans the internally generated data of a mess_equation object, i.e. it calls the clear function for all internally defined operations without removing the whole equation from memory.
This function is useful if the equation is used later on again but the memory footprint of the internal data is too big.
If the equation should be removed completely from memory use mess_equation_clear instead.

See also

mess_equation_clear

8.39.4.2 int mess_equation_clear (mess_equation * *eqn*)

Parameters

in,out	*eqn*	pointer to the mess_equation structure

Returns

zero on success or a non-zero error value otherwise

The mess_equation_clear function clears a mess_equation structure.
It calls all internal clean up handlers to remove all created data properly from the memory.
If it is necessary to call the clean up handlers without removing the whole equation from memory use mess_equation_clean instead.

See also

mess_equation_clean

Here is the call graph for this function:

8.39.4.3 int mess_equation_init (mess_equation * *eqn*)

Parameters

in,out	*eqn*	pointer to equation structure

Returns

zero on success or a non-zero error value otherwise

The mess_equation_init function initializes an empty mess_equation object.
In order to setup a proper equation use one of the generator functions from Equation Generators.

8.39.4.4 int mess_equation_set_rhs (mess_equation *eqn*, mess_options *opt*, mess_matrix *rhs*)

Parameters

in,out	*eqn*	mess_equation object to modify
in	*opt*	input options structure of the current solver
in	*rhs*	input new right hand side

Returns

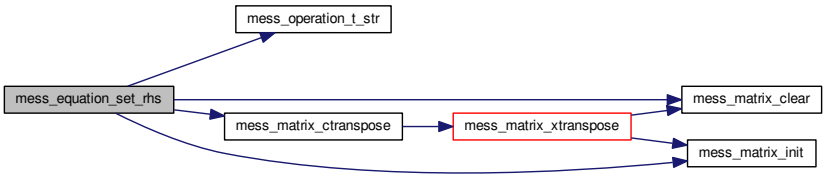
zero on success or a non-zero error value otherwise

The mess_equation_set_rhs function sets a new right hand side inside a mess_equation object.
The right hand side need to fit to the requirements of the equation type in mess_options.
If mess_options::type is MESS_OP_NONE the right hand side need to be a tall and skinny matrix with the correct number of rows. Otherwise it have to be the transposed.

See also

mess_equation_set_rhs2

Here is the call graph for this function:



8.39.4.5 int mess_equation_set_rhs2 (mess_equation *eqn*, mess_matrix *rhs*)

Parameters

in,out	*eqn*	mess_equation object to modify
in	*rhs*	input new right hand side

Returns

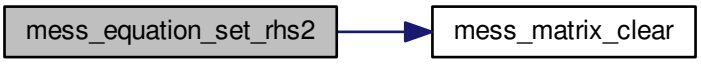
zero on success or a non-zero error value otherwise

The mess_equation_set_rhs2 function sets a new right hand side inside a mess_equation object without any sanity checks, except of checking for the equation type.

See also

mess_equation_set_rhs

Here is the call graph for this function:



8.39.4.6 int mess_options_clear (mess_options * *opt*)

Parameters

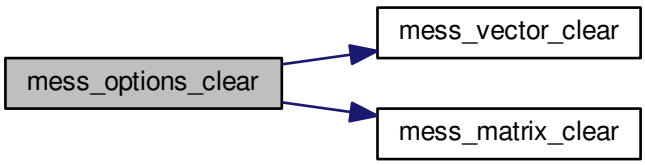
in,out	*opt*	pointer to a mess_options object

Returns

zero on success or a non-zero error value otherwise

The mess_options_clear function cleans up a mess_options object.

Here is the call graph for this function:



8.39.4.7 int mess_options_copy (mess_options *src*, mess_options *dest*)

Parameters

in	*src*	input source object to copy from
out	*dest*	destination object to copy to

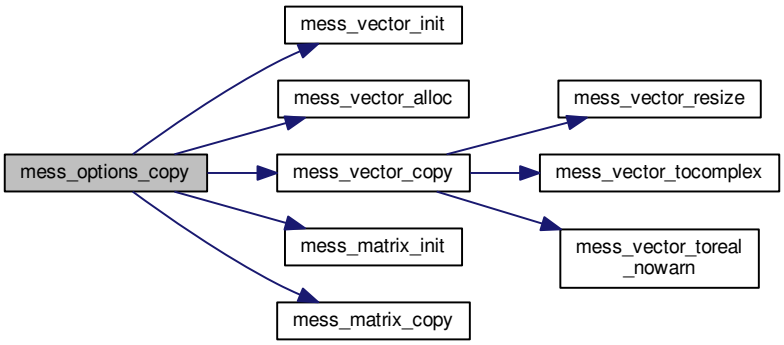
Returns

zero on success or a non-zero error value otherwise

The mess_options_copy function copies a mess_options object to another preinitialized one.

mess_options mess_options_init

Here is the call graph for this function:



8.39.4.8 int mess_options_init (mess_options * *opt*)

Parameters

in,out	*opt*	pointer to a mess_options object

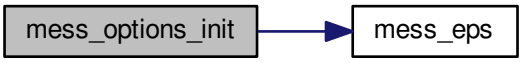
Returns

zero on success or a non-zero error value otherwise

The mess_options_init function initializes a mess_options object and sets the options to the default values.

mess_options

Here is the call graph for this function:



8.39.4.9 mess_int_t mess_options_memsize (mess_options *opt*)

Parameters

in	*opt*	input mess_options instance

Returns

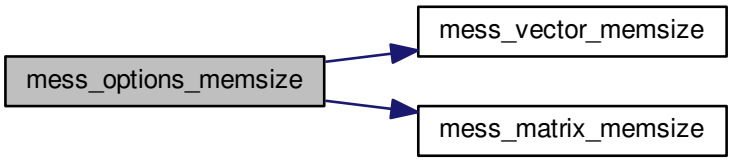
approximate size of the `stat` in bytes.

The mess_options_memsize function determines the size of `opt` in bytes.

See also

mess_matrix_memsize
mess_vector_memsize
mess_status_memsize

Here is the call graph for this function:



8.39.4.10 int mess_options_print (mess_options *opt*)

Parameters

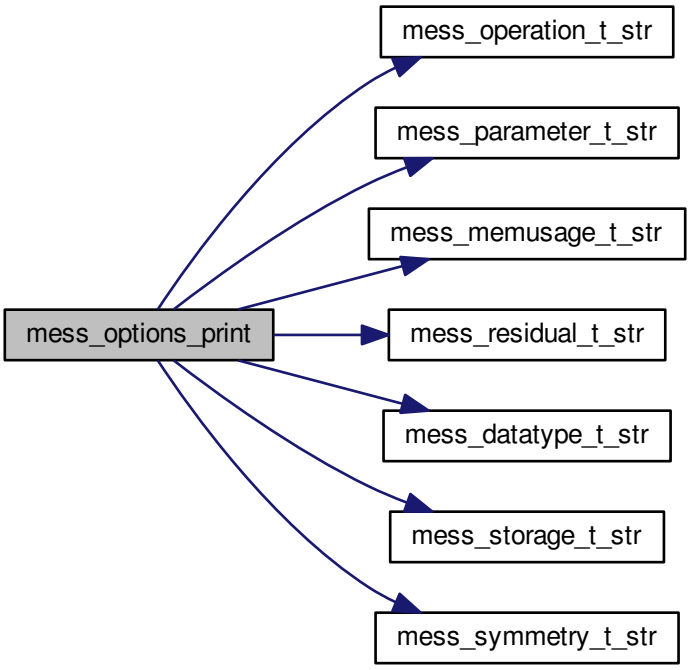
in	*opt*	input object to display

Returns

zero on success or a non-zero error value otherwise

The mess_options_print function displays the basis information of a mess_options object on the screen.

Here is the call graph for this function:



8.39.4.11 int mess_options_set_K0 (mess_options *opt*, mess_matrix *K0*)

Parameters

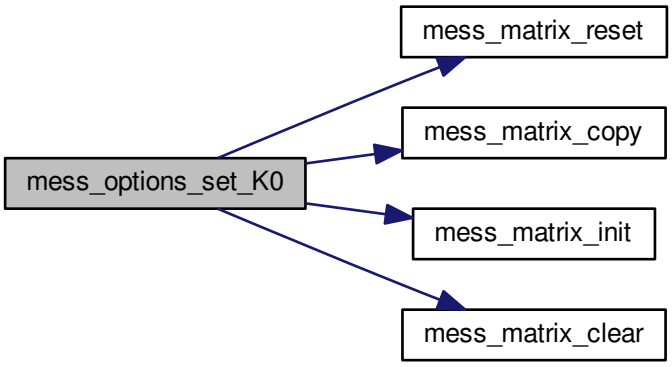
in	*opt*	input mess_options instance
in	*K0*	input mess_matrix instance

Returns

zero on success or a non-zero error value otherwise

The mess_options_set_K0 set the mess_matrix instance `K0` to the `K0` field of `opt`. Internally it is done by a copy.

Here is the call graph for this function:



8.39.4.12 int mess_options_set_proj_space (mess_options *opt*, mess_matrix *proj_space*)

Parameters

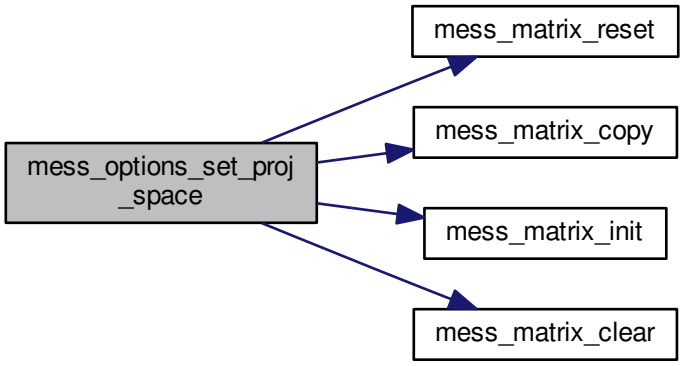
in	*opt*	input mess_options instance
in	*proj_space*	input mess_matrix instance

Returns

zero on success or a non-zero error value otherwise

The mess_options_set_proj_space set the mess_matrix instance `proj_space` to the `proj_space` field of `opt`. Internally it is done by a copy.

Here is the call graph for this function:



8.39.4.13 int mess_options_set_W (mess_options *opt*, mess_matrix *W*)

Parameters

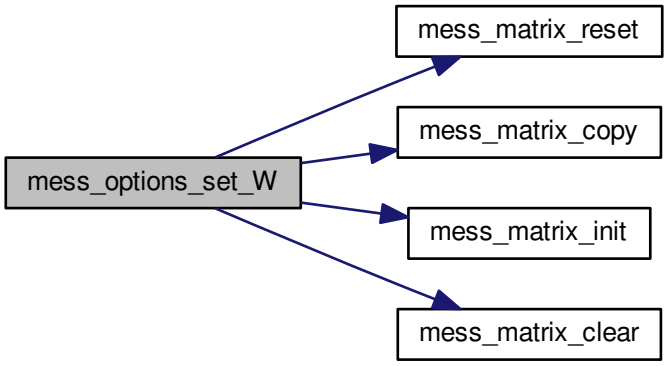
in	*opt*	input mess_options instance
in	*W*	input mess_matrix instance

Returns

zero on success or a non-zero error value otherwise

The mess_options_set_W set the mess_matrix instance `W` to the `W` field of `opt`. Internally it is done by a copy.

Here is the call graph for this function:



8.39.4.14 int mess_options_unset_K0 (mess_options *opt*)

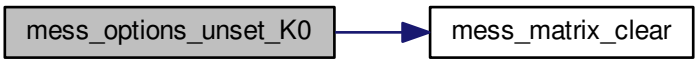
Parameters

in	*opt*	input mess_options instance

Returns

zero on success or a non-zero error value otherwise

Here is the call graph for this function:



8.39.4.15 int mess_options_unset_proj_space (mess_options *opt*)

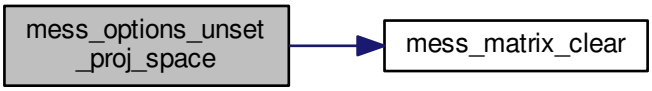
Parameters

in	*opt*	input mess_options instance

Returns

zero on success or a non-zero error value otherwise

Here is the call graph for this function:



8.39.4.16 int mess_options_unset_W (mess_options *opt*)

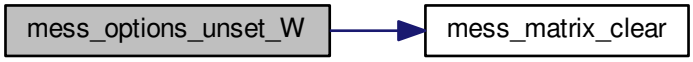
Parameters

in	*opt*	input mess_options instance

Returns

zero on success or a non-zero error value otherwise

Here is the call graph for this function:



8.39.4.17 int mess_status_clear (mess_status * *status*)

Parameters

in,out	*status*	pointer to the object

Returns

always zero

The mess_status_clear cleans up a mess_status object.

See also

mess_status_init

Here is the call graph for this function:

8.39.4.18 int mess_status_copy (mess_status *src*, mess_status *dest*)

Parameters

in	*src*	input source object to copy from
out	*dest*	output object to copy to

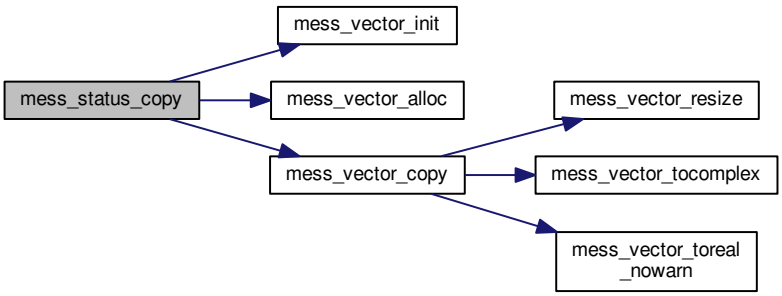
Returns

zero on success or a non-zero error value otherwise

The mess_status_copy function copies a mess_status object to another preinitialized one.

mess_status mess_status_init

Here is the call graph for this function:



8.39.4.19 int mess_status_init (mess_status * *status*)

Parameters

in,out	*status*	pointer to the object

Returns

zero on success or a non-zero error value otherwise

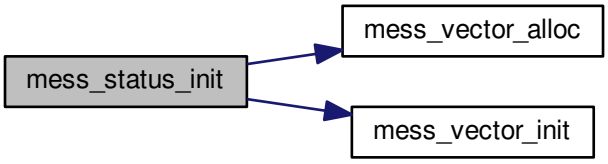
The mess_status_init function initializes a mess_status object with empty values.
In case of erros the following error codes can be returned:

- MESS_ERROR_MEMORY no memory left
- all errors from mess_vector_init

See also

mess_status
mess_options_init

Here is the call graph for this function:



8.39.4.20 mess_int_t mess_status_memsize (mess_status *stat*)

Parameters

in	*stat*	input mess_status instance

Returns

approximate size of the `stat` in bytes.

The mess_status_memsize function determines the size of `stat` in bytes.

See also

mess_matrix_memsize
mess_vector_memsize

Here is the call graph for this function:

8.39.4.21 int mess_status_print (mess_status *stat*)

Parameters

in	*stat*	input object to print

Returns

zero on success or a non-zero error value otherwise

The mess_status_print function prints basic infomation (except of the residual and rel_change vectors) on stdout.

See also

mess_status
mess_status_printfull
mess_status_write

Here is the call graph for this function:

8.39.4.22 int mess_status_printfull (mess_status *stat*)

Parameters

in	*stat*	input object to print

Returns

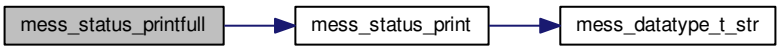
zero on success or a non-zero error value otherwise

The mess_status_printfull function displays all information contained in the given mess_status object to the standard output. This includes embedded status objects.

See also

mess_status
mess_status_print
mess_status_write

Here is the call graph for this function:



8.39.4.23 int mess_status_write (mess_status *stat*, const char * *filename*)

Parameters

in	*stat*	input object to save
in	*filename*	input filename of the text file

Returns

zero on success or a non-zero error value otherwise

The mess_status_write function saves all information contained in a mess_status object to a text file $filename$.

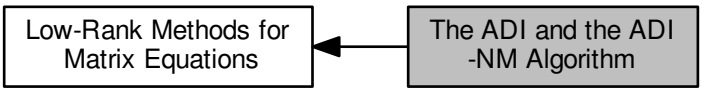
See also

mess_status
mess_status_print
mess_status_printfull

8.40 The ADI and the ADI-NM Algorithm

This category contains solving algorihms and useful subroutines.

Collaboration diagram for The ADI and the ADI-NM Algorithm:



Functions

- int mess_parameter (mess_equation eqn, mess_options opt, mess_status stat)

  Wrapper around mess_lrcfadi_parameter except that mess_equation.parameter is preferred (if available).
- int mess_lrcfadi_parameter (mess_equation eqn, mess_options opt, mess_status stat)

  Compute parameters for the low-rank Cholesky factor ADI iteration.
- int mess_lrcfadi_getritzvalues (mess_equation eqn, mess_options opt, mess_vector ritz)

  Compute a set of Ritz values from a given equation.
- int mess_lrcfadi_residual (mess_equation eqn, mess_options opt, mess_matrix Z, double ∗nrm)

  Compute 2-norm residual of an equation.
- int mess_lradi (mess_equation eqn, mess_options opt, mess_status stat, mess_matrix Z)

  Wrapper/Alternative name for mess_lrcfadi_adi.
- int mess_lrcfadi_adi (mess_equation eqn, mess_options opt, mess_status stat, mess_matrix Z)

  Solve a (generalized) Lyapunov Equation using the low-rank-Cholesky-factor ADI.
- int mess_lrnm (mess_equation eqn, mess_options opt, mess_status stat, mess_matrix Z)

  Wrapper/Alternative name for mess_lrcfadi_nm.
- int mess_lrcfadi_nm (mess_equation eqn, mess_options opt, mess_status stat, mess_matrix Z)

  Solve an algebraic Riccati Equation using the low-rank-Cholesky factor Newton method.
- int mess_lrcfadi_galerkin (mess_equation eqn, mess_options opt, mess_equation_t eqn_type, mess_matrix Z)

  Refine a solution of a matrix equation using Galerkin projection.
- int mess_lrcfadi_nm_linesearch_poly (mess_matrix W, mess_matrix Wnew, mess_matrix deltaK, mess_matrix deltaKnew, double ∗alpha, double ∗beta, double ∗delta, double ∗gamma, double ∗epsilon, double ∗zeta)

  Compute the coefficients of the polynomial for linesearch.
- int mess_lrcfadi_nm_linesearch_lambda (double ∗alpha, double ∗beta, double ∗delta, double ∗gamma, double ∗epsilon, double ∗zeta, double ∗lambda)

  Compute the optimal step size for linesearch.
- int mess_lrcfadi_nm_linesearch (mess_matrix W, mess_matrix Wnew, mess_matrix deltaK, mess_matrix deltaKnew, double ∗lambda)

  Compute the optimal step size for linesearch.

8.40.1 Detailed Description

It includes the parameter computation for the low-rank Cholesky factor alternate direction implicit (LRCF-ADI) and the computation of Ritz values or 2-norm residual of an equation. Additionally this category contains different solving functions for Lyapunov and Riccati Equations.
The **LRCF-ADI** iteration computes an approximated solution $X \approx ZZ^H$ of a Lyapunov Equation

$$AX + XA^T + BB^T = 0$$

by rewriting the ADI iteration

$$\begin{array}{lrcl} & X_0 & = & 0 \\ \text{for } j = 1, \cdots, J & & & \\ & (A + p_I)X_{j-\frac{1}{2}} & = & -BB^T - X_{j-1}(A^T - p_j I) \\ & (A + p_I)X_j^T & = & -BB^T - X_{j-\frac{1}{2}}^T(A^T - p_j I) \end{array}$$

as one step iteration and factorize $X_i = Z_i Z_i^T, i = 0, \cdots, J$.
This iteration can be also applied to generalized Lyapunov Equations

$$AXE^T + EXA^T + BB^T = 0$$

if E is regular. In this case the ADI iteration holds with $A := E^{-1}A$ and $B := E^{-1}B$.

Another implemented algorithm that is based on the LRCF-ADI is the **LRCF-ADI with Galerkin-projection acceleration (LRCF-ADI-GP)**.
In this method, the LRCF-ADI iterate Z_i is computed. Afterwards an orthogonal basis is computed via a QR factorization to project and to solve this smaller (generalized) Lyapunov Equation. Then Z_i will be updated.

One method to compute the solution of the Riccati Equation

$$\begin{array}{rcl} A^T X + XA - XBB^T X + C^T C & = & 0 \\ \text{or} & & \\ AX + XA^T - XCCX + BB^T & = & 0 \end{array}$$

or the generalized Riccati Equation

$$\begin{array}{rcl} A^T XE + E^T XA - E^T XBB^T XE + C^T C & = & 0 \\ \text{or} & & \\ AXE^T + EXA^T - EXC^C XE^T + BB^T & = & 0 \end{array}$$

is the **LRCF Newton method (LRCF-NM)** .
In this algorithm the (generalized) Riccati Equation is formulated as a (generalized) Lyapunov Equation with the help of Newtons iteration

$$\mathcal{R}'|_X(N_l) = -\mathcal{R}(X_l), \quad X_{l+1} = X_l + N_l, \quad l = 0, 1, \cdots$$

where $\mathcal{R}(X)$ corresponds to the (generalized) Riccati Equation and the Frechét derivate $\mathcal{R}'|_X$ is a Lyapunov operator. In every Newton step a (generalized) Lyapunov Equation need to be solved.
source: [http://www2.mpi-magdeburg.mpg.de/mpcsc/mitarbeiter/saak/lehre/Matrixgleichungen/↩ Saak_09WS.pdf, http://www2.mpi-magdeburg.mpg.de/mpcsc/mitarbeiter/saak/talks/saak↩ _MatEq_BS09.pdf]

8.40.2 Function Documentation

8.40.2.1 int mess_lradi (mess_equation *eqn,* mess_options *opt,* mess_status *stat,* mess_matrix *Z*)

Parameters

in	*eqn*	input Equation object defining Lyapunov Equation
in	*opt*	input options for iteration
out	*stat*	status information about iteration
out	*Z*	output factor Z of the solution $X \approx ZZ^T$

Returns

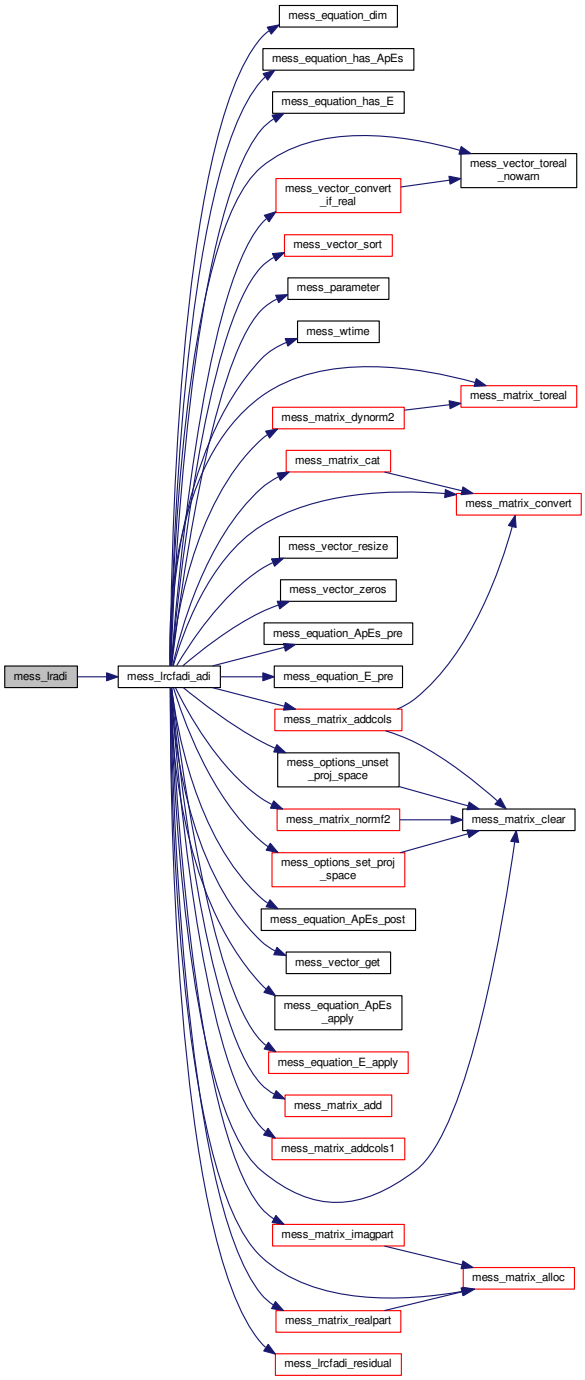
zero on success or a non zero error code otherwise

The mess_lradi function is a wrapper around mess_lrcfadi_adi.

See also

mess_lrcfadi_adi

Here is the call graph for this function:



8.40.2.2 int mess_lrcfadi_adi (mess_equation *eqn*, mess_options *opt*, mess_status *stat*, mess_matrix *Z*)

Parameters

in	*eqn*	input Equation object defining Lyapunov Equation
in	*opt*	input options for iteration
out	*stat*	status information about iteration

Parameters

out	Z	output factor Z of the solution $X \approx ZZ^T$

Returns

zero on success or a non-zero error value otherwise

The mess_lrcfadi_adi function solves a standard Lyapunov Equation

$$AX + XA^T + BB^T = 0$$

or a generalized Lyapunov Equation

$$AXE^T + EXA^T + BB^T = 0$$

using the low-rank Cholesky factor alternate direction implicit (LRCF-ADI) method.
The implementation produces a real solution factor even if complex shift parameters are used.
Details about the algorithm and its enhancements can be found in [10], [28] and [19] .
The matrices are not passed in directly. They need to be part of a mess_equation object before which can be setup using mess_equation_lyap or mess_equation_glyap before. The selection of the equation, normal or transposed, is done when this object is created.

Before the function is called a set of iteration parameters need to be computed. Normally this is done by calling mess_lrcfadi_parameter. If other shifts have been computed before they need to be put into the mess_options↩ ::adi_shifts_p vector.

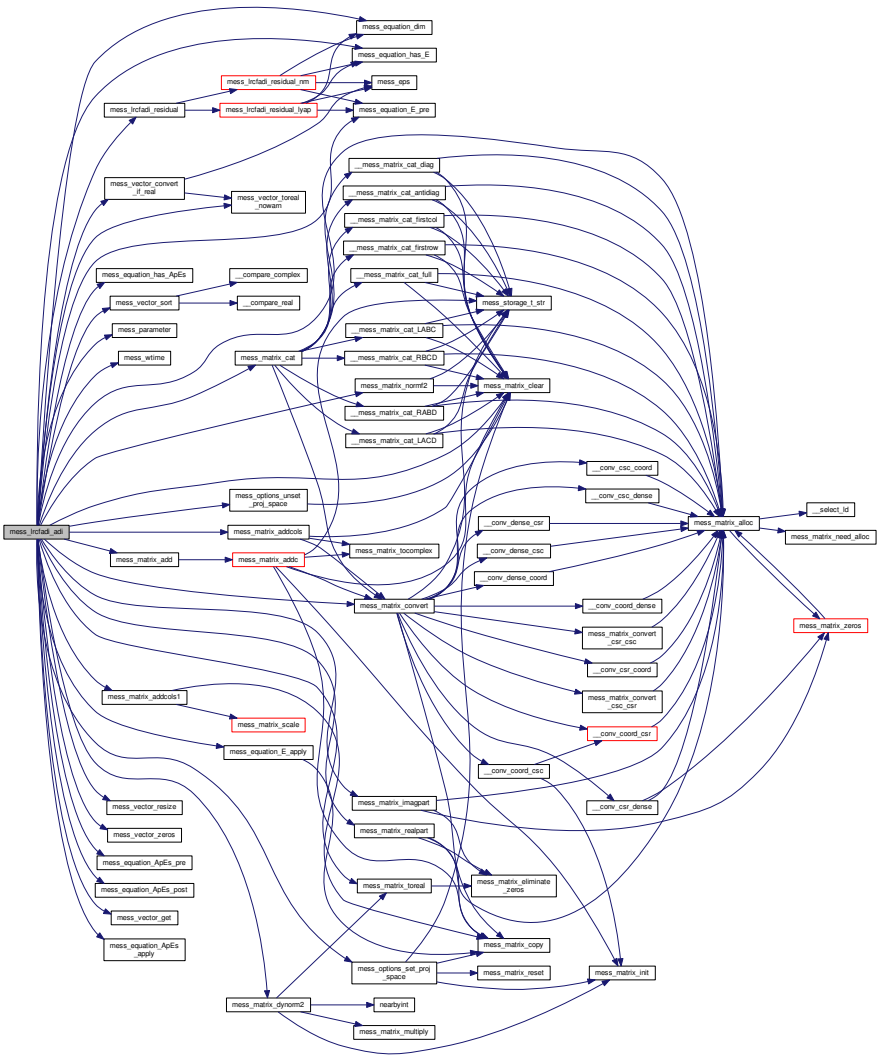
The options for the ADI iteration like the maximum iteration number and the cancellation criteria are configured by the `opt` argument. This argument is of type mess_options.

If relative and absoulte residual is large than `1e+12` the algorithm return immediately an error.

See also

mess_lrcfadi_lrcfadi
mess_options
mess_status

Here is the call graph for this function:



8.40.2.3 int mess_lrcfadi_galerkin (mess_equation *eqn*, mess_options *opt*, mess_equation_t *eqn_type*, mess_matrix *Z*)

Parameters

in	*eqn*	input equation object of the underlying matrix equation
in	*opt*	input operations object containing additional options
in,out	*Z*	factor Z of the solution $X \approx ZZ^T$, updated with the refinement
in	*eqn_type*	input type of equation (Lyapunov, Riccati)

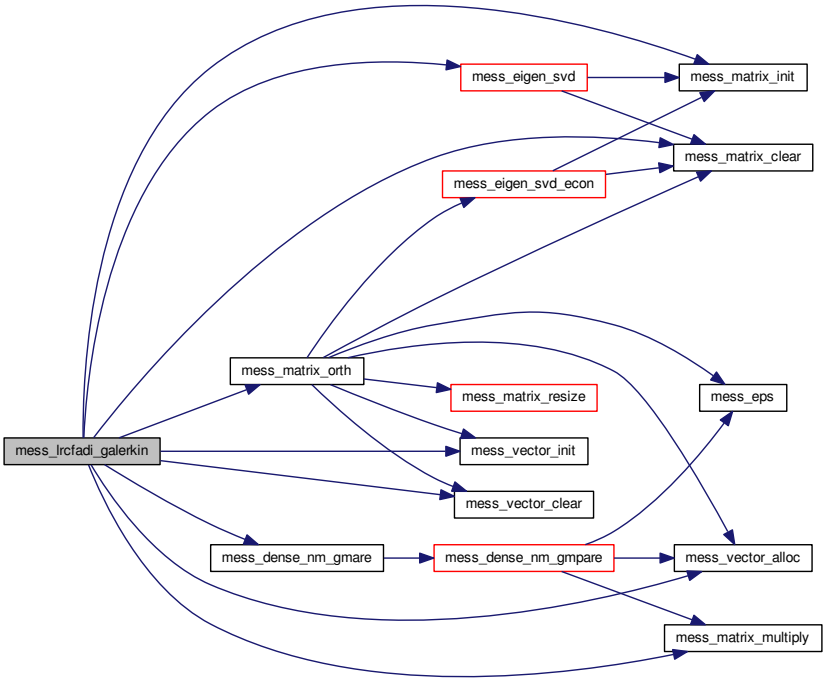
Returns

zero on success or a non-zero error value otherwise

The mess_lrcfadi_galerkin function projects an equation to the subspace spanned by the matrix Z. After the projection the small equation is solved with a classical dense matrix equation solver and the correct solution of the small equation is lifted back to the original space.
It supports standard and generalized Riccati Equations.

The projected Riccati Equation will always be solved using a dense Newton-method.
The function mess_dense_nm_gmare is called.

Here is the call graph for this function:



8.40.2.4 int mess_lrcfadi_getritzvalues (mess_equation *eqn*, mess_options *opt*, mess_vector *ritz*)

Parameters

in	*eqn*	input equation object from which the Ritz values are computed
in	*opt*	input options for solver and shift parameter computation
out	*ritz*	vector containing Ritz values of the operator

Returns

zero on success or a non zero error code

The mess_lrcfadi_getritzvalues function computes a set of Ritz values of a given equation object.
Depending on the parameters mess_options::adi_shifts_arp_p and mess_options::adi_shifts_arp_m two Krylov

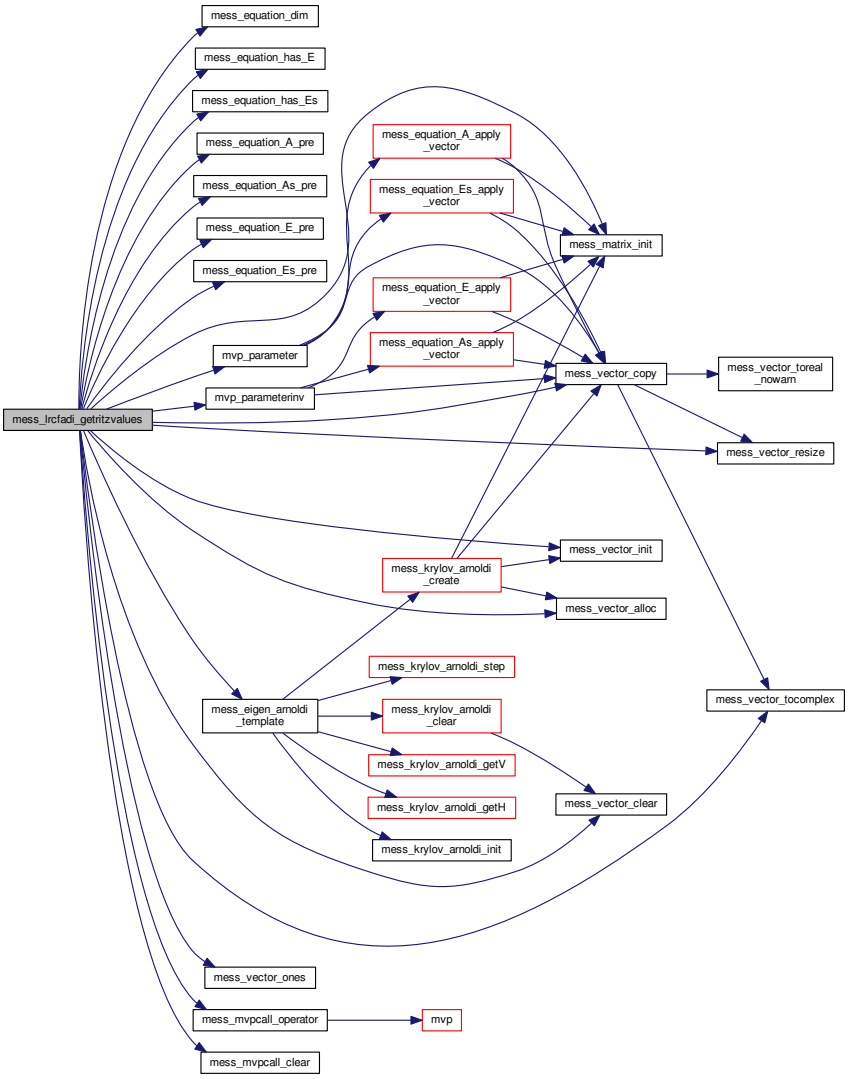
subspaces are computed using an Arnoldi method. These parameters set the dimension of the desired subspaces with respect to the operator (arp_p) and its inverse (arp_m).
The Ritz values are combined in the output vector $ritz$.

In general this function is similar to mess_eigen_eigs but works with a more abstract equation object and its defined operators instead of matrices.

See also

mess_lrcfadi_parameter
mess_eigen_arnoldi_template

Here is the call graph for this function:



8.40.2.5 int mess_lrcfadi_nm (mess_equation *eqn*, mess_options *opt*, mess_status *status*, mess_matrix *Zout*)

Parameters

in	*eqn*	input equation object defining Riccati Equation
in	*opt*	input options object to configure the Newton iteration and the internal ADI
out	*status*	status object containing all details about the iteration
out	*Zout*	factor Z of the solution $X \approx ZZ^T$

Returns

zero on success or a non zero error code otherwise

The mess_lrcfadi_nm function solves algebraic Riccati Equations (ARE)

$$A^T X + XA - XBB^T X + C^T C = 0$$

or

$$AX + XA^T - XC^T CX + BB^T = 0$$

using the low-rank-Cholesky factor Newton method (LRCF-NM).
The solution of generalized algebraic Riccati Equations (GARE)

$$A^T XE + E^T XA - E^T XBB^T XE + C^T C = 0$$

or

$$AXE^T + EXA^T - EXC^T CXE^T + BB^T = 0$$

is also supported.
This function works using a previously created mess_equation object which defines all necessary operations for matrices. Internally a special mess_equation object is created to handle the internal ADI iteration.
Details about the algorithm can be found in [28] and [12] .

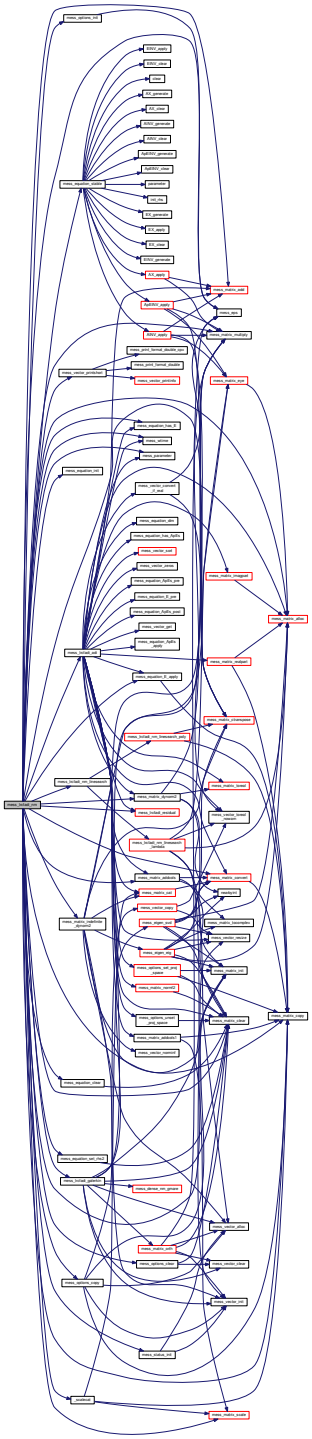
The necessary equation objects are generated using mess_equation_riccati or mess_equation_griccati for example.

The iteration is controlled by the corresponding entries in the given options object.

See also

mess_equation_st
mess_lrcfadi_adi
mess_lrcfadi_lrcfnm

Here is the call graph for this function:



8.40.2.6 int mess_lrcfadi_nm_linesearch (mess_matrix *W*, mess_matrix *Wnew*, mess_matrix *deltaK*, mess_matrix *deltaKnew*, double * *lambda*)

Parameters

in	*W*	input last residual factor from mess_lradi
in	*Wnew*	input actual residual factor from mess_lradi
in	*deltaK*	input last change in feedback.
in	*deltaKnew*	input actual change in feedback.
out	*lambda*	output optimal step length λ of the polynomial.

Returns

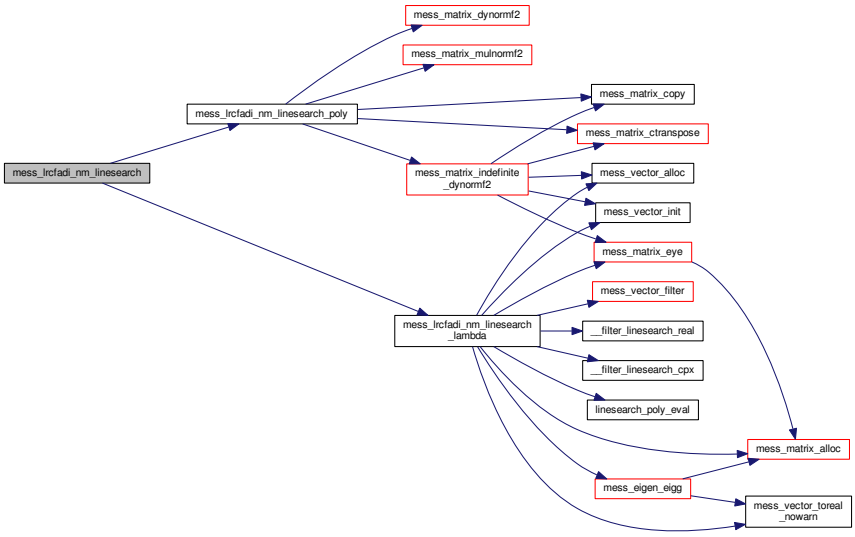
zero on success or a non-zero error value otherwise

The mess_lrcfadi_nm_linesearch function computes the optimal feasible step lenght λ. More Information can be found in [7].

See also

mess_lrcfadi_nm_linesearch_lambda
mess_lrcfadi_nm_linesearch_poly

Here is the call graph for this function:



8.40.2.7 int mess_lrcfadi_nm_linesearch_lambda (double * *alpha*, double * *beta*, double * *delta*, double * *gamma*, double * *epsilon*, double * *zeta*, double * *lambda*)

Parameters

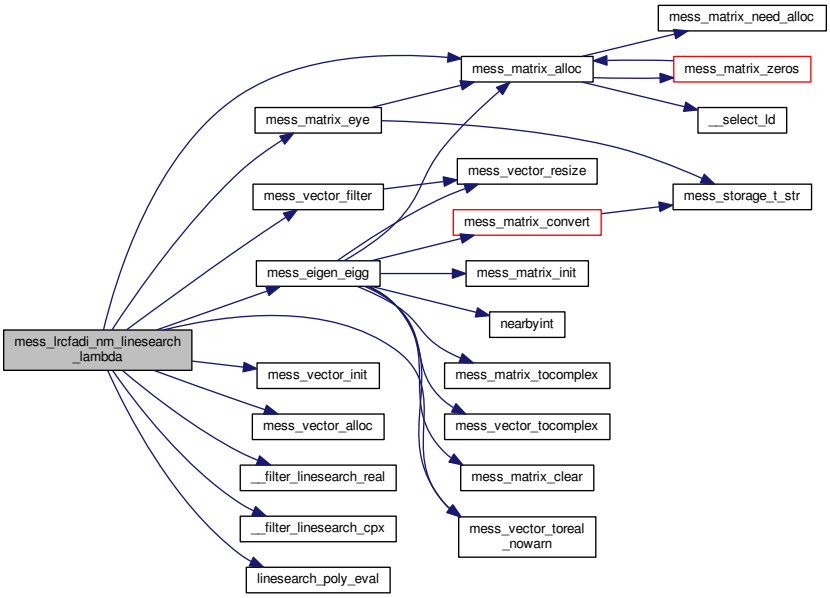
in	*alpha*	input coefficient α of the polynomial.
in	*beta*	input coefficient β of the polynomial.
in	*delta*	input coefficient δ of the polynomial.
in	*gamma*	input coefficient γ of the polynomial.
in	*epsilon*	input coefficient ϵ of the polynomial.
in	*zeta*	input coefficient ζ of the polynomial.
out	*lambda*	output optimal step length λ of the polynomial.

Returns

zero on success or a non-zero error value otherwise

The mess_lrcfadi_nm_linesearch_lambda function computes the optimal feasible step lenght λ based on the coefficients of the polynomial. More Information can be found in [7].

Here is the call graph for this function:



8.40.2.8 int mess_lrcfadi_nm_linesearch_poly (mess_matrix *W*, mess_matrix *Wnew*, mess_matrix *deltaK*, mess_matrix *deltaKnew*, double * *alpha*, double * *beta*, double * *delta*, double * *gamma*, double * *epsilon*, double * *zeta*)

Parameters

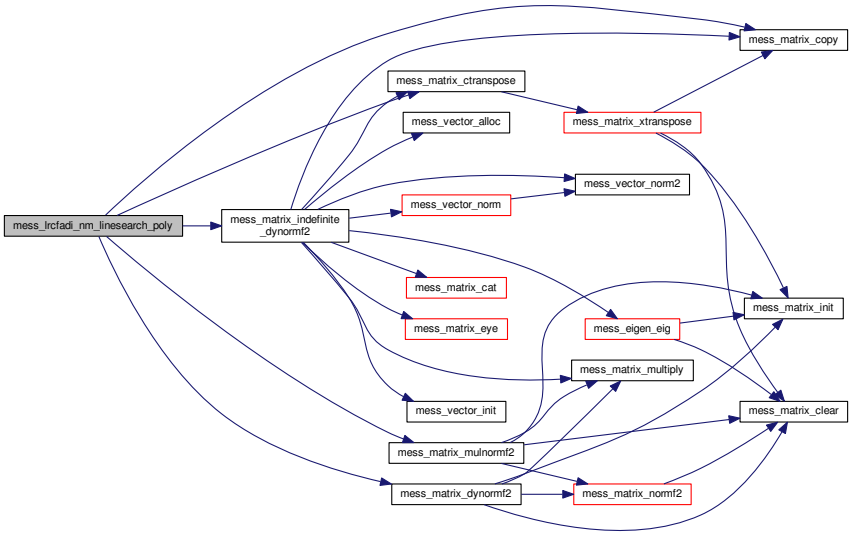
in	*W*	input last residual factor from mess_lradi
in	*Wnew*	input actual residual factor from mess_lradi
in	*deltaK*	input last change in feedback.
in	*deltaKnew*	input actual change in feedback.
out	*alpha*	output coefficient α of the polynomial.
out	*beta*	output coefficient β of the polynomial.
out	*delta*	output coefficient δ of the polynomial.
out	*gamma*	output coefficient γ of the polynomial.
out	*epsilon*	output coefficient ϵ of the polynomial.
out	*zeta*	output coefficient ζ of the polynomial.

Returns

zero on success or a non-zero error value otherwise

The mess_lrcfadi_nm_linesearch_poly function computes the coefficients of the polynomial needed by linesearch. More Information can be found in [7].

Here is the call graph for this function:



8.40.2.9 int mess_lrcfadi_parameter (mess_equation *eqn*, mess_options *opt*, mess_status *stat*)

Parameters

in	*eqn*	input equation object defining the desired equation
in,out	*opt*	options controlling the process
out	*stat*	status object for returning some states

Returns

zero on success or a non-zero error value otherwise

The mess_lrcfadi_parameter function computes parameters for the LRCF-ADI process.
Depending on the given mess_options.adi_shifts_paratype field in `opt` minmax, real minmax or Wachpress parameters are computed. The type is defined using the mess_parameter_t enumeration.
More details about shift parameters and their computation can be found in [28] and [25] .

The parameter computation is controlled by 4 values from the Options object:

- mess_options.adi_shifts_l0 field in `opt` the desired number of shift parameters,
- mess_options.adi_shifts_arp_p field in `opt` sets the number of steps in the Arnoldi process with respect to the operator,

- mess_options.adi_shifts_arp_m field in `opt` sets the number of steps in the Arnoldi process with respect to the inverse of the operator,

- mess_options.adi_shifts_paratype field in `opt` sets the parameter type. The necessary Ritz-values are computed via mess_lrcfadi_getritzvalues.

The mess_options::adi_shifts_paratype is one of following enumeration values:

- MESS_LRCFADI_PARA_MINMAX - compute (complex) parameters out of the rational min-max problem.

- MESS_LRCFADI_PARA_MINMAX_REAL - compute (real) parameters out of the rational min-max problem. (They are basically the same as MESS_LRCFADI_PARA_MINMAX but the imaginary part is neglected.)

- MESS_LRCFADI_PARA_WACHSPRESS - compute Wachspress parameters ([29]).

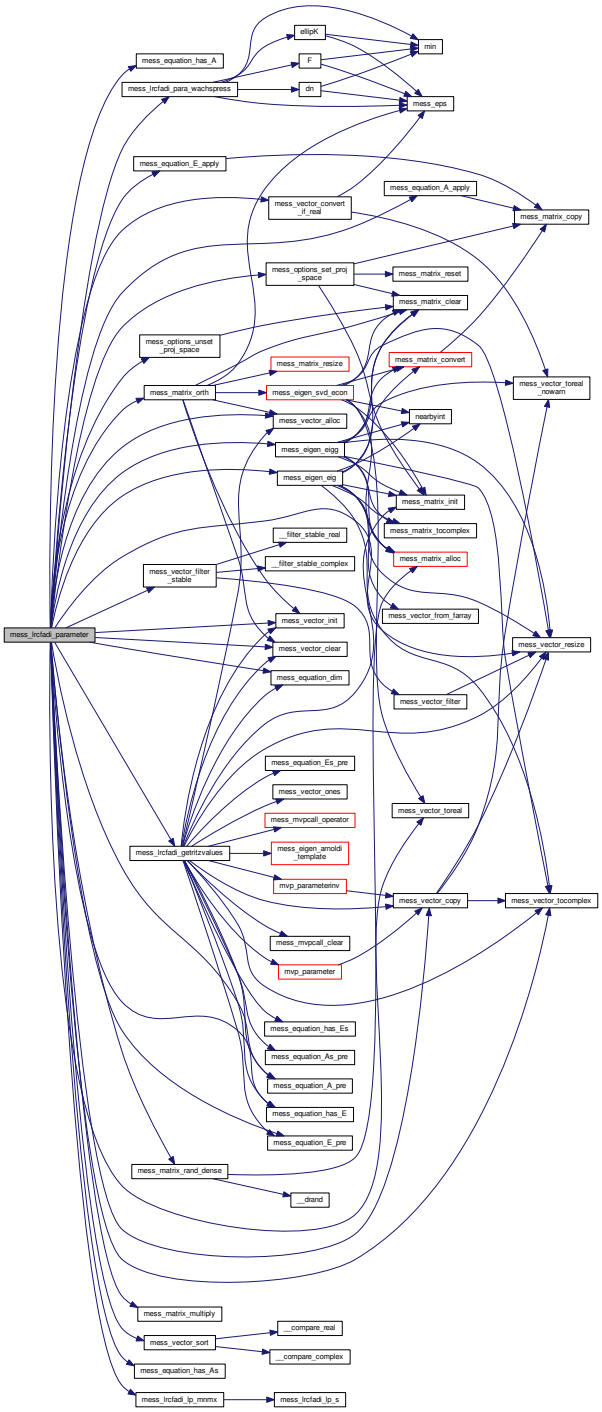
- MESS_LRCFADI_PARA_ADAPTIVE_V - compute adaptive shift parameters ([9]).

- MESS_LRCFADI_PARA_ADAPTIVE_Z - compute adaptive shift parameters ([9]).

See also

mess_lrcfadi_para
mess_lrcfadi_getritzvalues

Here is the call graph for this function:



8.40.2.10 int mess_lrcfadi_residual (mess_equation *eqn*, mess_options *opt*, mess_matrix *Z*, double * *nrm*)

Parameters

in	*eqn*	input object defining an equation
in	*opt*	input options for the equation and the solver
in	*Z*	input factor Z of the solution $X \approx ZZ^T$

Parameters

out	*nrm*	output 2-norm of the residual

Returns

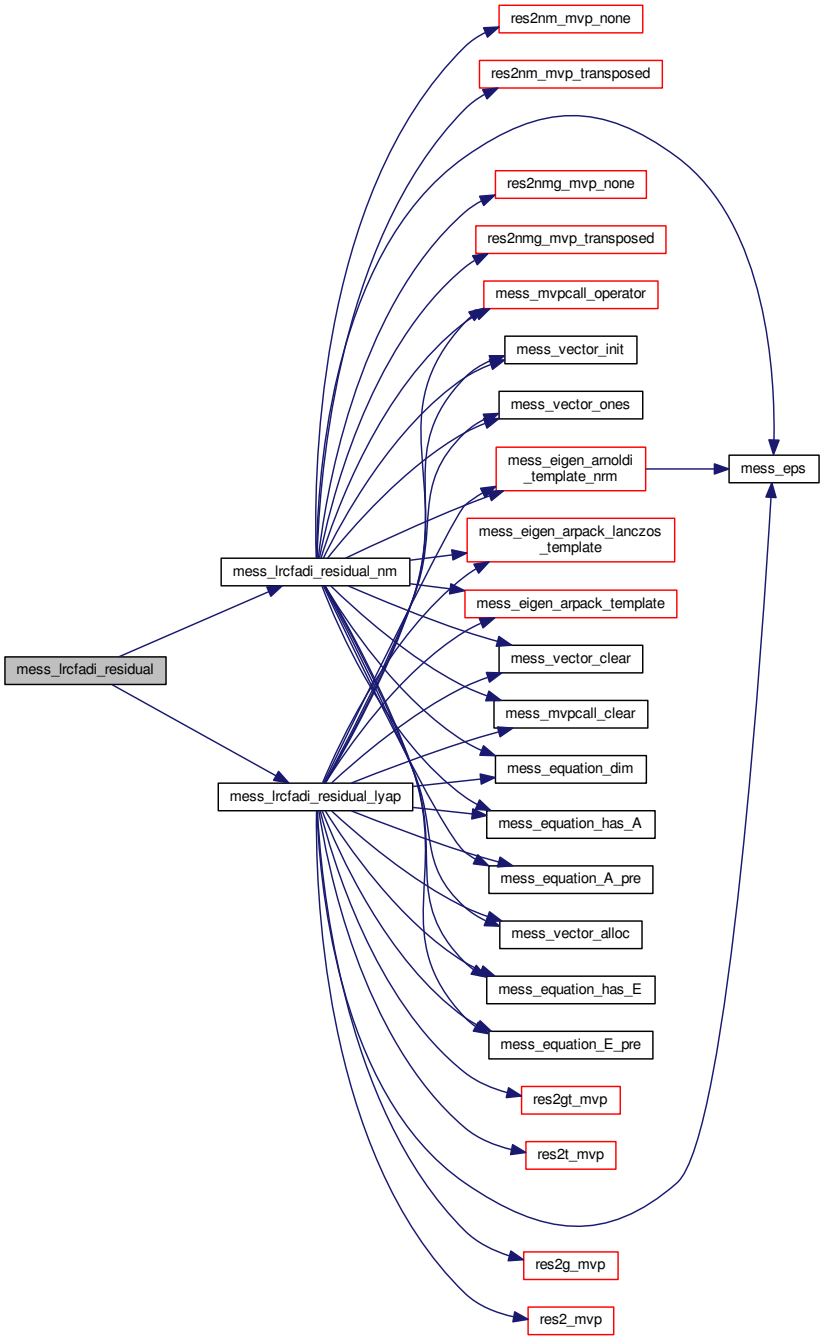
zero on success or a non-zero error value otherwise

The mess_lrcfadi_residual function computes the 2-norm of an equation object. It supports Lyapunov Equations as well as Riccati Equations.

Remarks

If **M.E.S.S.** is compiled with ARPACK support the Arnoldi Process from ARPACK is used to compute the largest eigenvalue. Otherwise the mess_eigen_arnoldi_template_nrm function is used.

Here is the call graph for this function:



8.40.2.11 int mess_lrnm (mess_equation *eqn*, mess_options *opt*, mess_status *status*, mess_matrix *Zout*)

Parameters

in	*eqn*	input equation object defining Riccati Equation
in	*opt*	input options object to configure the Newton iteration and the internal ADI
out	*status*	status object containing all details about the iteration
out	*Zout*	factor Z of the solution $X \approx ZZ^T$

Returns

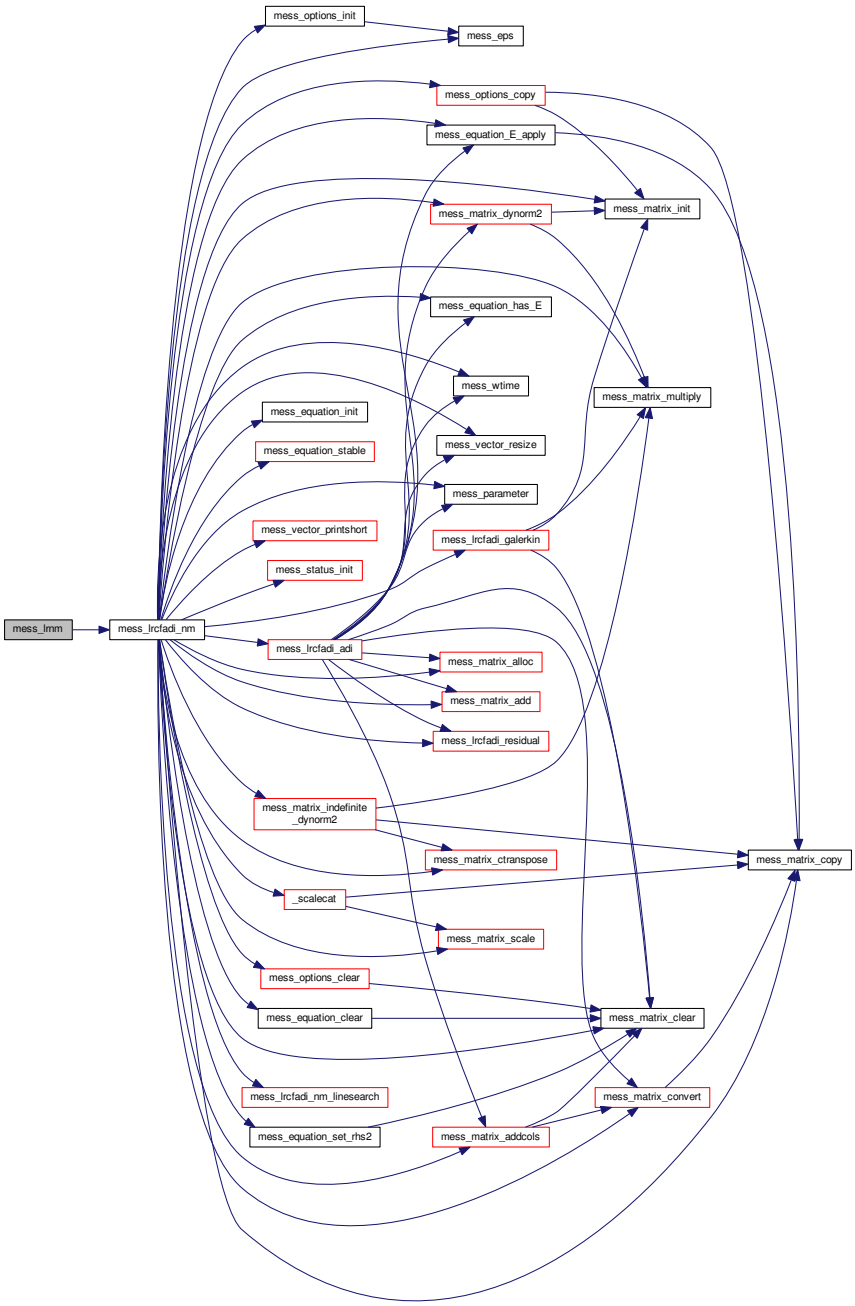
zero on success or a non zero error code otherwise

The mess_lrnm function is a wrapper for mess_lrcfadi_nm.

See also

mess_lrcfadi_nm

Here is the call graph for this function:



8.40.2.12 int mess_parameter (mess_equation *eqn*, mess_options *opt*, mess_status *stat*)

Parameters

in	*eqn*	input equation object defining the desired equation
in	*opt*	options controlling the process
out	*stat*	status object for returning some states

Returns

zero on success or a non-zero error value otherwise

The mess_parameter function calls mess_equation.parameter, if it not points to NULL. Otherwise the mess_lrcfadi_parameter is called.

Attention

Do not call mess_parameter inside mess_equation.parameter.

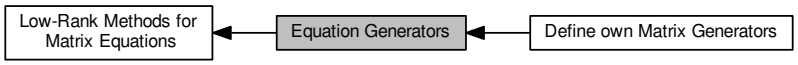
See also

mess_lrcfadi_para
mess_lrcfadi_getritzvalues
mess_equation.parameter

8.41 Equation Generators

This category contains functions to define and to build Lyapunov or Riccati Equations.

Collaboration diagram for Equation Generators:



Modules

- Define own Matrix Generators

  Defining own Matrix Generators by implementing basic operations.

Functions

- int mess_equation_lyap (mess_equation eqn, mess_options opt, mess_matrix A, mess_matrix E, mess_matrix B)

  Setup a mess_Equation object to solve a matrix-valued Lyapunov Equation.
- int mess_equation_riccati (mess_equation eqn, mess_options opt, mess_matrix A, mess_matrix E, mess_matrix B, mess_matrix C)

  Setup a mess_equation object to solve a matrix-valued Riccati Equation.
- int mess_equation_std_lyap (mess_equation eqn, mess_options opt, mess_matrix A, mess_matrix B)

  Setup a mess_Equation object to solve a standard Lyapunov Equation.
- int mess_equation_std_riccati (mess_equation eqn, mess_options opt, mess_matrix A, mess_matrix B, mess_matrix C)

  Setup a mess_equation object to solve a standard Riccati Equation.
- int mess_equation_glyap (mess_equation eqn, mess_options opt, mess_matrix A, mess_matrix E, mess_matrix B)

  Setup a mess_Equation object to solve a generalized Lyapunov Equation.
- int mess_equation_griccati (mess_equation eqn, mess_options opt, mess_matrix A, mess_matrix E, mess_matrix B, mess_matrix C)

  Setup a mess_equation object to solve a generalized Riccati Equation.
- int mess_equation_glyap_so1 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix D, mess_matrix K, mess_matrix B, double lowerbound, double upperbound)

  Generate an Equation object to a Lyapunov Equation arising from a second order system.
- int mess_equation_griccati_so1 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix D, mess_matrix K, mess_matrix B, mess_matrix C, double lowerbound, double upperbound)

  Generate an equation object to a Riccati Equation arising from a second order system.
- int mess_equation_glyap_so2 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix D, mess_matrix K, mess_matrix B, double lowerbound, double upperbound)

  Generate an Equation object to a generalized Lyapunov Equation arising from a second order system.
- int mess_equation_griccati_so2 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix D, mess_matrix K, mess_matrix B, mess_matrix C, double lowerbound, double upperbound)

  Generate an equation object to a generalized Riccati Equation arising from a second order system.
- int mess_equation_glyap_dae2 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix A, mess_matrix G, mess_matrix B, double delta)

Generate an Equation object to solve a generalized Lyapunov Equation from Hessenberg Index 2 DAE.

- int mess_equation_griccati_dae2 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix A, mess_matrix G, mess_matrix B, mess_matrix C, double delta)

  Generate an equation object to a Riccati Equation arising from a Hessenberg Index 2 DAE.

- int mess_equation_glyap_dae1 (mess_equation e, mess_options opt, mess_matrix E11, mess_matrix A11, mess_matrix A12, mess_matrix A21, mess_matrix A22, mess_matrix rhs)

  Generate an Equation object to solve a generalized Lyapunov Equation from Index 1 DAE.

- int mess_equation_griccati_dae1 (mess_equation e, mess_options opt, mess_matrix E11, mess_matrix A11, mess_matrix A12, mess_matrix A21, mess_matrix A22, mess_matrix B, mess_matrix C)

  Generate an Equation object to solve a generalized Lyapunov Equation from Index 1 DAE.

- int mess_equation_stable (mess_equation e, mess_options opt, mess_equation base, mess_matrix B, mess_matrix K)

  Generate a stabilized Lyapunov Equation object.

8.41.1 Detailed Description

It includes functions to define the type of a (generalized) Lyapunov Equation

$$op(A)Xop(E)^T + op(E)Xop(A)^T + op(B)op(B)^T = 0$$

or a (generalized) Riccati Equation

$$\begin{array}{rcl} AXE^T + EXA^T - EXC^TCXE^T + BB^T & = & 0 \\ A^TXE + E^TX^A - E^TXBB^TXE + C^TC & = & 0 \end{array}$$

where $E = I$ in case of a continous Lyapunov or Riccati Equation.
There are also functions to build up generalized Lyapunov and Riccati Equations from a second order LTI system

$$\begin{array}{rcll} M\ddot{x} + D\dot{x} + K \quad x & = & Bu & , \\ y & = & C_p x \quad +C_v\dot{x} & , \end{array}$$

which will be transformed to a first order system

$$\begin{array}{rcl} \underbrace{\begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix}}_{\mathcal{E}} \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} & = & \underbrace{\begin{bmatrix} 0 & -K \\ -K & -D \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ B \end{bmatrix}}_{\mathcal{B}} u, \\ y & = & \underbrace{\begin{bmatrix} C_p & C_v \end{bmatrix}}_{\mathcal{C}} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}. \end{array}$$

In case of so1 or to a first order system

$$\begin{array}{rcl} \underbrace{\begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix}}_{\mathcal{E}} \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} & = & \underbrace{\begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ B \end{bmatrix}}_{\mathcal{B}} u, \\ y & = & \underbrace{\begin{bmatrix} C_p & C_v \end{bmatrix}}_{\mathcal{C}} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}. \end{array}$$

in case of so2.
The generalized Lyapunov Equation

$$\mathcal{A}X\mathcal{E}^T + \mathcal{E}X\mathcal{A}^T + \mathcal{B}\mathcal{B}^T = 0$$

or the generalized Riccati Equation

$$\mathcal{A}^TX\mathcal{E} + \mathcal{E}^TX\mathcal{A} - \mathcal{E}^TX\mathcal{B}\mathcal{B}^TX\mathcal{E} + \mathcal{C}^T\mathcal{C} = 0$$

are build out of this system.
This category also includes function to build up generalized Lyapunov or Riccati Equations from a Hessenberg Index 2 DAE

$$\begin{array}{rcll} M\dot{z} & = & Az & +Gp+Bu \quad , \\ 0 & = & G^T z & \quad , \\ y & = & Cz & \quad , \end{array}$$

where G need to have full column rank and M has to be symmetric and positive definite.
It is also possible to generate a low-rank updated generalized Lyapunov Equation

$$op(A-BK)Xop(E)^T + op(E)Xop(A-BK)^T + op(Q)op(Q)^T = 0$$

and to update the matrix K in this equation.

8.41.2 Function Documentation

8.41.2.1 int mess_equation_glyap (mess_equation *e*, mess_options *opt*, mess_matrix *A*, mess_matrix *E*, mess_matrix *B*)

Parameters

in,out	*e*	mess_equation object
in	*opt*	input mess_options object
in	*A*	input system matrix of the equation
in	*E*	input mass matrix of the equation
in	*B*	input right hand side of the equation

Returns

zero on success or a non-zero error value otherwise

The mess_equation_glyap function sets up a mess_equation object such that the Lyapunov equation

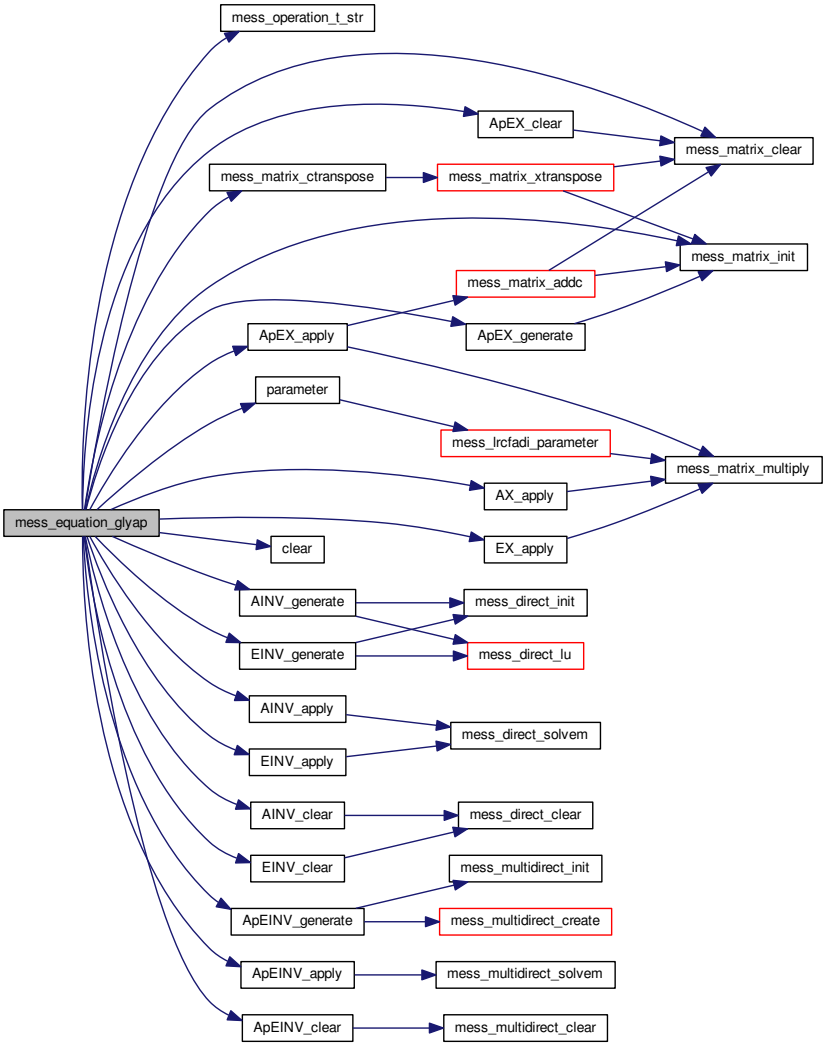
$$op(A)Xop(E)^T + op(E)Xop(A)^T + op(B)op(B)^T = 0$$

is represented.
Matrices need to be real. The operation is selected by the mess_options::type component.
If mess_options::type attribute of `opt` is set to MESS_OP_NONE the ADI process will solve the standard Lyapunov Equation, i.e. $op(A) = A$.
Otherwise if mess_options::type is set to MESS_OP_TRANSPOSE the transposed Lyapunov Equation with $op(A) = A^T$ will be solved. The resulting object can be used with the mess_lrcfadi_adi solver.

See also

mess_lrcfadi_adi
mess_equation_lyap
mess_equation_st

Here is the call graph for this function:



8.41.2.2 int mess_equation_glyap_dae1 (mess_equation *e*, mess_options *opt*, mess_matrix *E11*, mess_matrix *A11*, mess_matrix *A12*, mess_matrix *A21*, mess_matrix *A22*, mess_matrix *rhs*)

Parameters

in,out	*e*	equation object
in	*opt*	input options for ADI iteration
in	*E11*	input $E_{11} \in \mathbb{R}^{n_1 \times n_1}$ matrix of the DAE system
in	*A11*	input $A_{11} \in \mathbb{R}^{n_1 \times n_1}$ matrix of the DAE system
in	*A12*	input $A_{12} \in \mathbb{R}^{n_1 \times n_2}$ matrix of the DAE system

Parameters

in	*A21*	input $A_{21} \in \mathbb{R}^{n_2 \times n_1}$ matrix of the DAE system
in	*A22*	input $A_{22} \in \mathbb{R}^{n_2 \times n_2}$ matrix of the DAE system
in	*rhs*	input $rhs \in \mathbb{R}^{n_1+n_2 \times p}$ or $rhs \in \mathbb{R}^{p \times n_1+n_2}$

Returns

zero on success or a non zero error code otherwise

The mess_equation_glyap_dae1 function creates a mess_equation object from Index 1 DAE System

$$\begin{bmatrix} E_{11} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u$$

Preliminaries:

- $E_{11}, A_{11} \in \mathbb{R}^{n_1 \times n_1}, A_{12} \in \mathbb{R}^{n_1 \times n_2}, A_{21} \in \mathbb{R}^{n_2 \times n_1}, A_{22} \in \mathbb{R}^{n_2 \times n_2}$.
- A_{22} is invertible.

Using that A_{22} is invertible leads to the ODE system

$$E_{11} \frac{dx_1}{dt} = (A_{11} - A_{12} A_{22}^{-1} A_{21}) x_1 + (B(1:n_1,:) - A_{12} A_{22}^{-1} B(n_1+1:end,:)) u$$

Depending on the mess_operation_t defined in mess_options::type we build an equation structure to solve the following Lyapunov Equations: If the operation type is MESS_OP_NONE, we solve:

$$\begin{aligned} E_{11} X (A_{11} - A_{12} A_{22}^{-1} A_{21})^T + (A_{11} - A_{12} A_{22}^{-1} A_{21}) X E_{11}^T &= -BB^T \\ B &:= (rhs(1:n_1,:) - A_{12} A_{22}^{-1} rhs(n_1+1:end,:)) \end{aligned}$$

otherwise

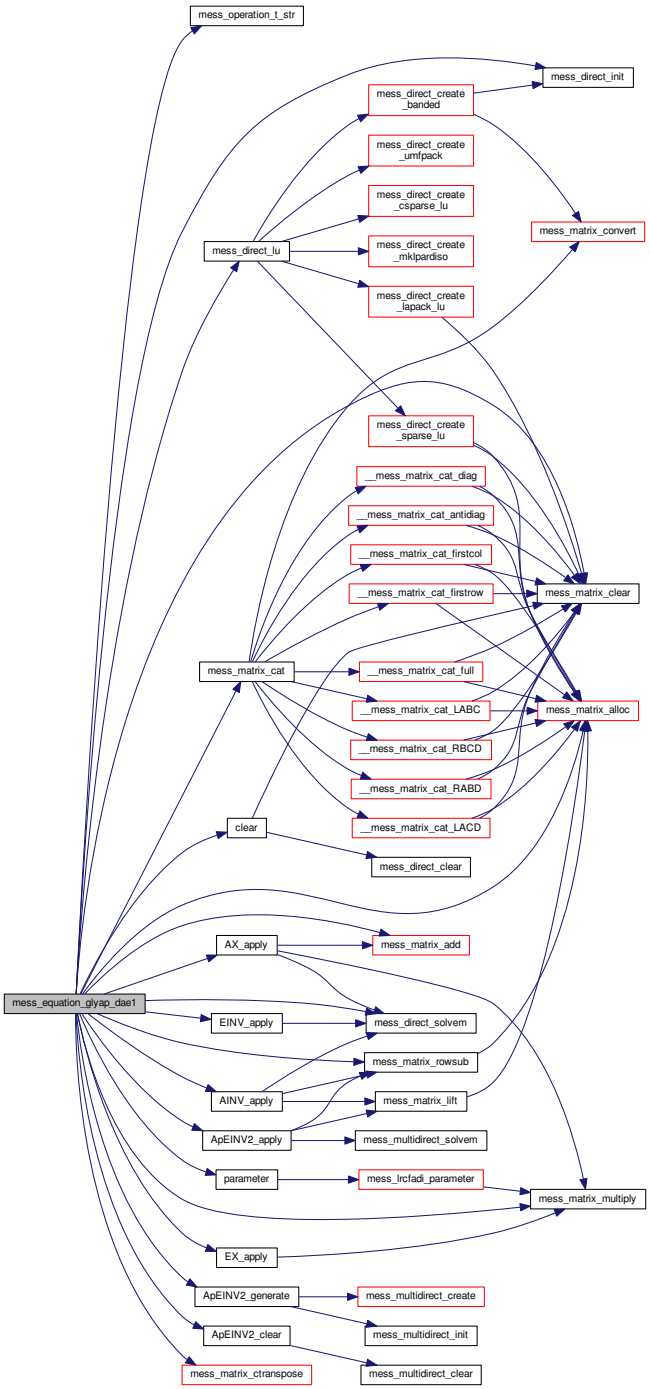
$$\begin{aligned} E_{11}^T X (A_{11} - A_{12} A_{22}^{-1} A_{21}) + (A_{11} - A_{12} A_{22}^{-1} A_{21})^T X E_{11} &= -B^T B \\ B &:= (rhs(:,1:n_1) - rhs(:,n_1+1:end) A_{22}^{-1} A_{21}). \end{aligned}$$

See also

mess_lrcfadi_adi
mess_equation_lyap
mess_equation_st
mess_equation_griccati_dae1

References: [17]

Here is the call graph for this function:



8.41.2.3 int mess_equation_glyap_dae2 (mess_equation *e*, mess_options *opt*, mess_matrix *M*, mess_matrix *A*, mess_matrix *G*, mess_matrix *rhs*, double *delta*)

Parameters

in,out	*e*	equation object
in	*opt*	input options for ADI iteration

Parameters

in	*M*	input M matrix of the DAE system
in	*A*	input A matrix of the DAE system
in	*G*	input G matrix for the linear algebraic constraint
in	*rhs*	input B or C (unprojected) right hand side of the equation
in	*delta*	input δ for the infinite eigenvalues of the matrix pencil

Returns

zero on success or a non zero error code otherwise

The mess_equation_glyap_dae2 function creates a mess_equation object from a Hessenberg Index 2 DAE System

$$\begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{dz}{dt} \\ 0 \end{bmatrix} = \begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} z \\ p \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u$$

Preliminaries: $A \in \mathbb{R}^{n_v \times n_v}, B \in \mathbb{R}^{n_v \times p}, C \in \mathbb{R}^{n_a \times n_v}$.
$n_p < n_v$.
$\delta \in \mathbb{R}$ and $\delta < 0$.
$M \in \mathbb{R}^{n_v \times n_v}$ must be symmetric and positive definite.
$G \in \mathbb{R}^{n_v \times n_p}$ and G has full rank n_p.
The matrix pencil

$$\left(\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}, \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \right)$$

belongs to that system. To avoid infinite eigenvalues, we use

$$\left(\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}, \begin{bmatrix} M & \delta G \\ \delta G^T & 0 \end{bmatrix} \right)$$

as matrix pencil for shift parameter computation in mess_lrcfadi_parameter. This transforms all infinite eigenvalues to $\frac{1}{\delta}$.
Depending on the mess_options::type component a normal or transposed projected Lyapunov Equation is created.
The discrete Leray Projector is applied implicitly.
For more details look at [2] [3].

Depending on the mess_options::memory_usage component of the mess_options structure the memory usage is controlled.

- mess_options::memory_usage = 0: In every iteration a new solver is computed in ApEINV_apply.
- mess_options::memory_usage = 1: For every shift a solver is precomputed and reused in ApEINV_apply.
- mess_options::type = MESS_OP_NONE: $(\Pi A) X M^T + M X (\Pi A)^T = -(\Pi B)(\Pi B)^T$
- mess_options::type = MESS_OP_TRANSPOSE: $(\Pi A)^T X M + M^T X (\Pi A) = -(C \Pi^T)^T (C \Pi^T)$

If your system is unstable use mess_equation_stable to build the stabilized operator $(A - BK^T)$ or $(A - KB^T)$, to ensure the proper application of Π also after stabilization This function sets the B component of the given mess↩_equation_st structure to

mess_options::type = MESS_OP_NONE:

$$\Pi B$$

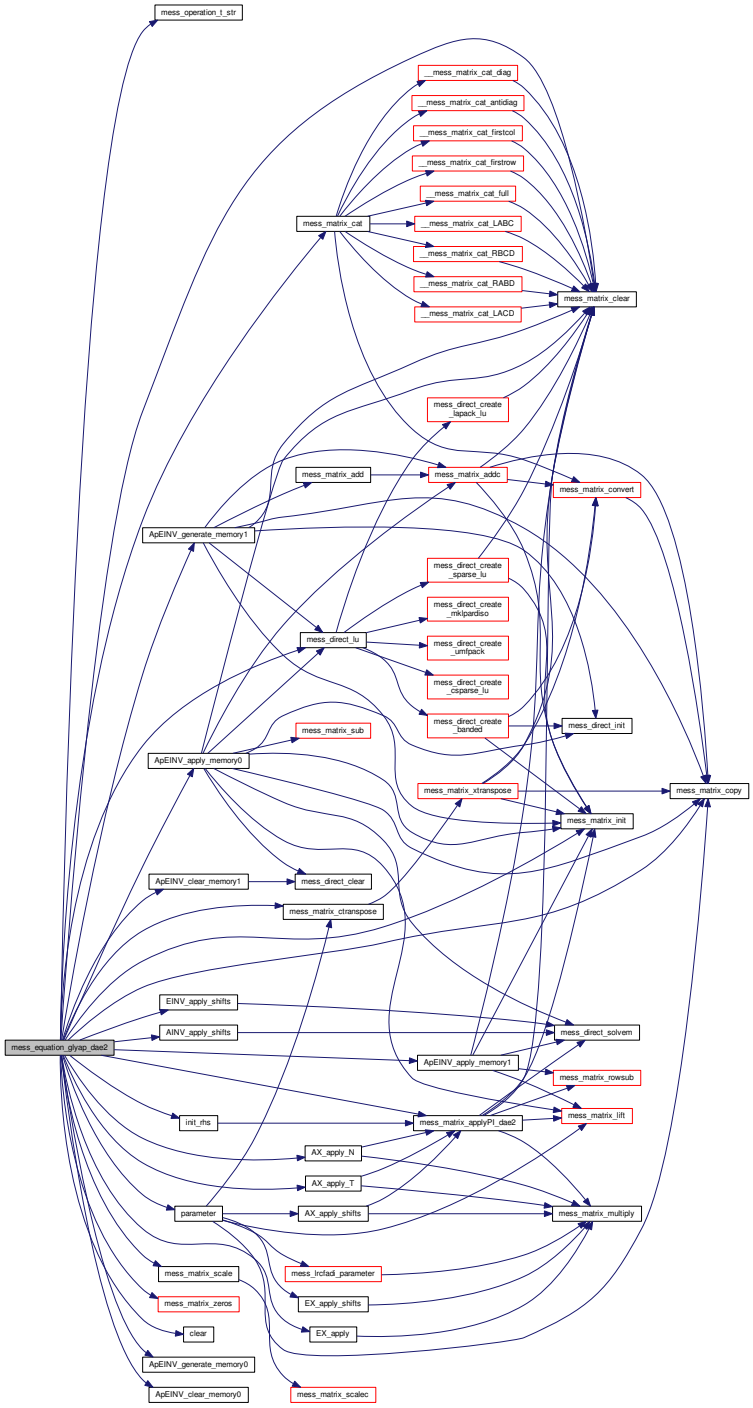
mess_options::type = MESS_OP_TRANSPOSE:

$$B \Pi^T.$$

See also

mess_lrcfadi_adi
mess_equation_lyap
mess_equation_st
mess_equation_stable

Here is the call graph for this function:



8.41.2.4 int mess_equation_glyap_so1 (mess_equation *e*, mess_options *opt*, mess_matrix *M*, mess_matrix *D*, mess_matrix *K*, mess_matrix *B*, double *lowerbound*, double *upperbound*)

Parameters

out	*e*	equation object to setup
in	*opt*	input options for the ADI iteration
in	*M*	input M matrix of the LTI system
in	*D*	input D matrix of the LTI system
in	*K*	input K matrix if the LTI system
in	*B*	input right hand side B of the LTI system
in	*lowerbound*	input lower bound for absolute value of shift parameter
in	*upperbound*	input upper bound for absolute value of shift parameter

Returns

zero on success or a non zero error code otherwise

The mess_equation_glyap_so1 function creates an equation object from a second order LTI system

$$M\ddot{x} + D\dot{x} + Kx \quad = \quad Bu \ ,$$

where M, D and K are matrices of dimension n.
Furthermore K need to be invertible and M positive definite.
The system is implicitly transformed to its first order representation:

$$\underbrace{\begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix}}_{\mathcal{E}} \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & -K \\ -K & -D \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ B \end{bmatrix}}_{\mathcal{B}} u$$

and the mess_lrcfadi_adi function will solve

$$\mathcal{A}X\mathcal{E}^T + \mathcal{E}X\mathcal{A}^T + \mathcal{B}\mathcal{B}^T = 0$$

or

$$\mathcal{A}^T X\mathcal{E} + \mathcal{E}^T X\mathcal{A} + \mathcal{B}^T\mathcal{B} = 0$$

If mess_options::type equals MESS_OP_NONE the right hand side B must be passed in as

$$\mathcal{B} = \begin{bmatrix} 0 \\ B \end{bmatrix} \in \mathbb{R}^{2n\times t}$$

otherwise

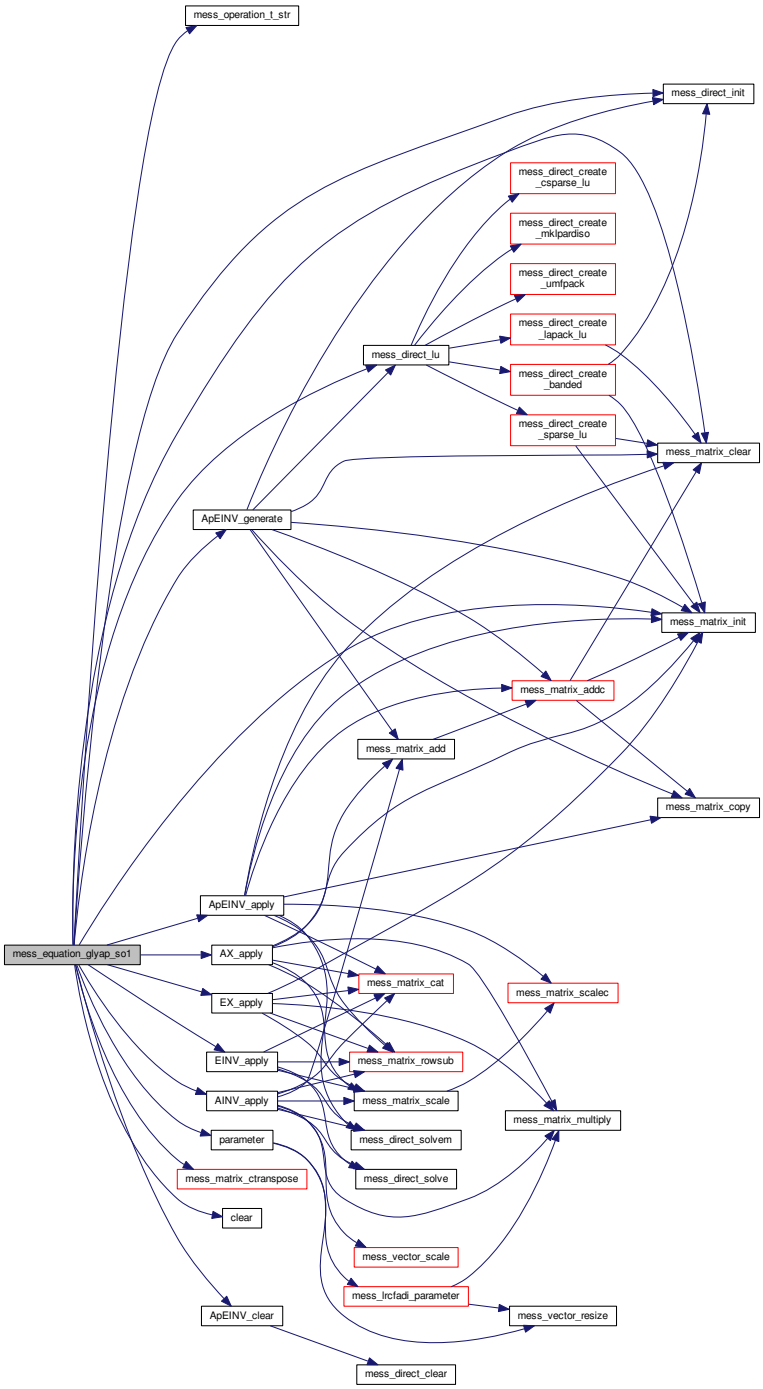
$$\mathcal{B} = \begin{bmatrix} 0 & B^T \end{bmatrix} \in \mathbb{R}^{t\times 2n}$$

The linearization of the second order system leads to quadratic shifts. To avoid numerical problems one can defines lower and upper bounds for the absolute value of the shift. Shifts that not fullfill the inequalitiy $lowerbound < |p| < upperbound$ are automatically sorted out. Common choices are $lowerbound = 10^{-5}$ and $upperbound = 10^5$.

See also

mess_lrcfadi_adi
mess_equation_lyap

Here is the call graph for this function:



8.41.2.5 int mess_equation_glyap_so2 (mess_equation *e*, mess_options *opt*, mess_matrix *M*, mess_matrix *D*, mess_matrix *K*, mess_matrix *B*, double *lowerbound*, double *upperbound*)

Parameters

out	*e*	equation object to setup
in	*opt*	input options for the ADI iteration
in	*M*	input M matrix of the LTI system
in	*D*	input D matrix of the LTI system
in	*K*	input K matrix if the LTI system
in	*B*	input right hand side B of the LTI system
in	*lowerbound*	input lower bound for absolute value of shift parameter
in	*upperbound*	input upper bound for absolute value of shift parameter

Returns

zero on success or a non zero error code otherwise

The mess_equation_glyap_so2 function creates an equation object from a second order LTI system

$$M\ddot{x} + D\dot{x} + Kx \quad = \quad Bu$$

where $M, D, K \in \mathbb{R}^{n \times n}$.
Furthermore K need to be invertible and M positive definite.
The system is implicitly transformed to its first order representation:

$$\underbrace{\begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix}}_{\mathcal{E}} \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ B \end{bmatrix}}_{\mathcal{B}} u$$

and the mess_lrcfadi_adi function will solve

$$\mathcal{A}X\mathcal{E}^T + \mathcal{E}X\mathcal{A}^T + \mathcal{B}\mathcal{B}^T = 0$$

or

$$\mathcal{A}^T X\mathcal{E} + \mathcal{E}^T X\mathcal{A} + \mathcal{B}^T\mathcal{B} = 0$$

If mess_options::type equals MESS_OP_NONE the right hand side B must be passed in as

$$\mathcal{B} = \begin{bmatrix} 0 \\ B \end{bmatrix} \in \mathbb{R}^{2n \times t},$$

otherwise

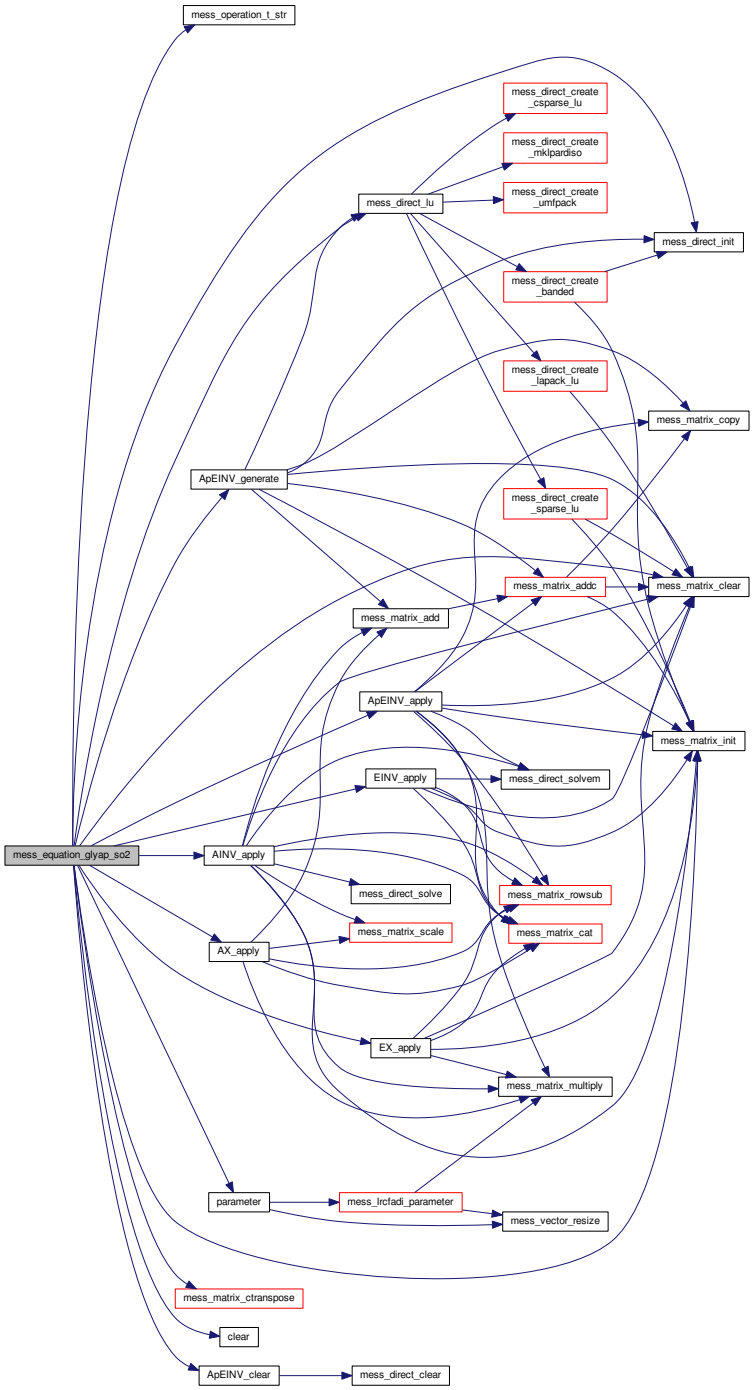
$$\mathcal{B} = \begin{bmatrix} 0 & B^T \end{bmatrix} \in \mathbb{R}^{t \times 2n}$$

The linearization of the second order system leads to quadratic shifts. To avoid numerical problems one can defines lower and upper bounds for the absolute value of the shift. Shifts that not fullfill the inequalitiy $lowerbound < |p| < upperbound$ are automatically sorted out. Common choices are $lowerbound = 10^{-5}$ and $upperbound = 10^5$.

See also

mess_lrcfadi_adi
mess_equation_lyap

Here is the call graph for this function:



8.41.2.6 int mess_equation_griccati (mess_equation *e*, mess_options *opt*, mess_matrix *A*, mess_matrix *E*, mess_matrix *B*, mess_matrix *C*)

Parameters

in,out	*e*	mess_equation object
in	*opt*	input mess_options object

Parameters

in	*A*	input system matrix of the equation
in	*E*	input mass matrix of the equation
in	*B*	input factor of the matrix in the quadratic term
in	*C*	input right hand side

Returns

zero on success or a non-zero error value otherwise

The mess_equation_griccati function sets up a mess_equation object such that the Riccati equation

$$AXE^T + EXA^T - EXC^TCXE^T + BB^T = 0$$

or the transposed one

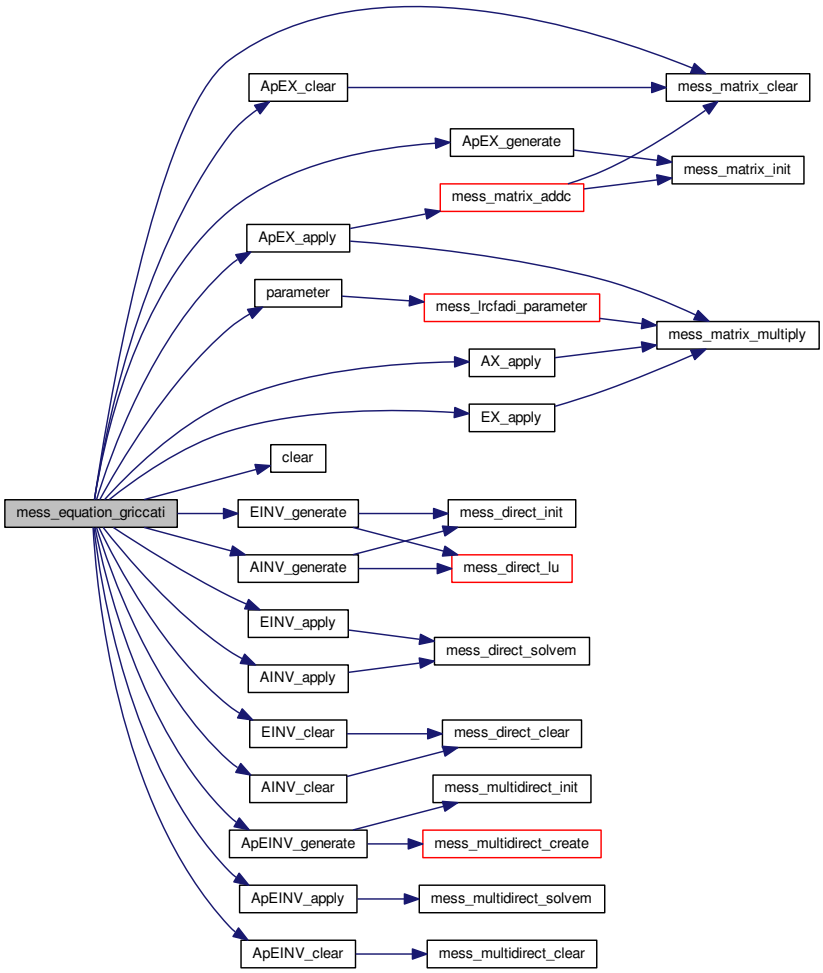
$$A^TXE + E^TXA - E^TXBB^TXE + C^TC = 0$$

is represented.
Depending on the mess_options::type component in the mess_options object the first or the second one is solved. The first one corresponds to th that is equal to MESS_OP_NONE and the second one corresponds to the mess_options::type that is equal to MESS_OP_TRANSPOSE.

See also

mess_lrcfadi_adi
mess_equation_riccati
mess_equation_st
mess_lrcfadi_nm

Here is the call graph for this function:



8.41.2.7 int mess_equation_griccati_dae1 (mess_equation *e*, mess_options *opt*, mess_matrix *E11*, mess_matrix *A11*, mess_matrix *A12*, mess_matrix *A21*, mess_matrix *A22*, mess_matrix *B*, mess_matrix *C*)

Parameters

in,out	*e*	equation object
in	*opt*	input options for ADI iteration
in	*E11*	input $E_{11} \in \mathbb{R}^{n_1 \times n_1}$ matrix of the DAE system
in	*A11*	input $A_{11} \in \mathbb{R}^{n_1 \times n_1}$ matrix of the DAE system
in	*A12*	input $A_{12} \in \mathbb{R}^{n_1 \times n_2}$ matrix of the DAE system
in	*A21*	input $A_{21} \in \mathbb{R}^{n_2 \times n_1}$ matrix of the DAE system
in	*A22*	input $A_{22} \in \mathbb{R}^{n_2 \times n_2}$ matrix of the DAE system
in	*B*	input $B \in \mathbb{R}^{n_1+n_2 \times p}$ or $B \in \mathbb{R}^{n_1 \times p}$
in	*C*	input $C \in \mathbb{R}^{q \times n_1}$ or $C \in \mathbb{R}^{q \times n_1+n_2}$

Returns

zero on success or a non zero error code otherwise

The mess_equation_glyap_dae1 function creates a mess_equation object from Index 1 DAE System.

$$\begin{bmatrix} E_{11} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + Bu$$
$$y = C \begin{bmatrix} x_1 & x_2 \end{bmatrix}$$

Preliminaries:

- $E_{11}, A_{11} \in \mathbb{R}^{n_1 \times n_1}, A_{12} \in \mathbb{R}^{n_1 \times n_2}, A_{21} \in \mathbb{R}^{n_2 \times n_1}, A_{22} \in \mathbb{R}^{n_2 \times n_2}$.
- A_{22} is invertible.

Using that A_{22} is invertible leads to the ODE system

$$E_{11} \frac{dx_1}{dt} = (A_{11} - A_{12} A_{22}^{-1} A_{21}) x_1 + (B(1:n_1,:) - A_{12} A_{22}^{-1} B(n_1+1:end,:))u$$
$$y = C \begin{bmatrix} x_1 \\ -A_{22}^{-1} A_{21} x_1 \end{bmatrix} - C \begin{bmatrix} 0 \\ A_{22}^{-1} B(n_1+1:end,:)u \end{bmatrix}$$

Depending on the mess_operation_t defined in mess_options::type we build an equation structure to solve the following Riccati Equations: If the operation type is MESS_OP_NONE, we solve:

$$E_{11} X (A_{11} - A_{12} A_{22}^{-1} A_{21})^T + (A_{11} - A_{12} A_{22}^{-1} A_{21}) X E_{11}^T - E_{11} X C^T C X E_{11}^T + \tilde{B} \tilde{B}^T = 0$$
$$\tilde{B} := (B(1:n_1,:) - A_{12} A_{22}^{-1} B(n_1+1:end,:))$$

otherwise

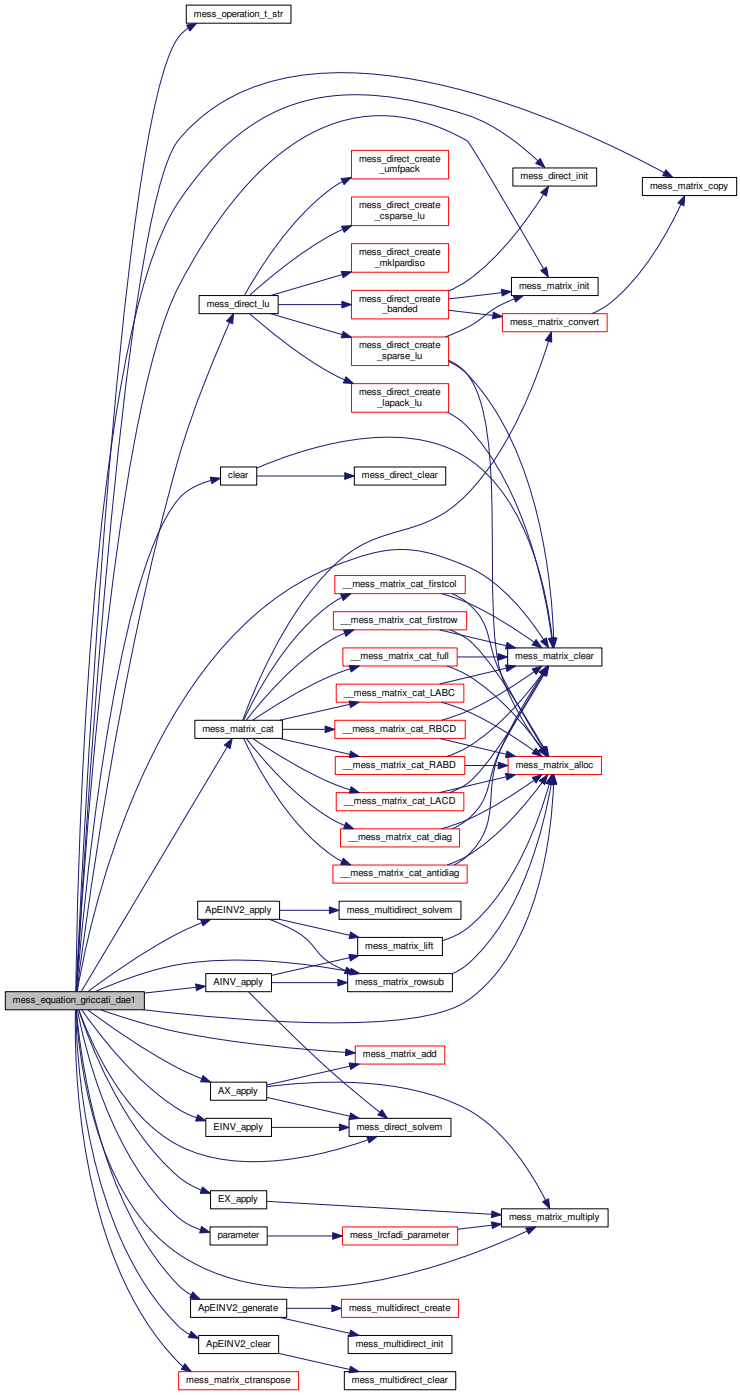
$$E_{11}^T X (A_{11} - A_{12} A_{22}^{-1} A_{21}) + (A_{11} - A_{12} A_{22}^{-1} A_{21})^T X E_{11} - E_{11}^T X B B^T X E_{11} + \tilde{C}^T \tilde{C} = 0$$
$$\tilde{C} := (C(:,1:n_1) - C(:,n_1+1:end) A_{22}^{-1} A_{21}).$$

See also

mess_lrcfadi_nm
mess_equation_lyap
mess_equation_st
mess_equation_glyap_dae1

References: [17]

Here is the call graph for this function:



8.41.2.8 int mess_equation_griccati_dae2 (mess_equation *e*, mess_options *opt*, mess_matrix *M*, mess_matrix *A*, mess_matrix *G*, mess_matrix *B*, mess_matrix *C*, double *delta*)

Parameters

out	*e*	equation object to setup
in	*opt*	input options for the ADI iteration

Parameters

in	*M*	input M matrix of the DAE system
in	*A*	input A matrix of the DAE system
in	*G*	input G matrix for the linear algebraic constraint
in	*B*	input right hand side B for the control
in	*C*	input observer matrix for the output y
in	*delta*	input shifting parameter for the infinite eigenvalues of the matrix pencil

Returns

zero on success or a non zero error code otherwise

The mess_equation_griccati_dae2 function creates a mess_equation object from a Hessenberg Index 2 DAE System

$$\begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{dz}{dt} \\ 0 \end{bmatrix} = \begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} z \\ p \end{bmatrix}$$
$$y = Cz$$

Preliminaries:

- $A \in \mathbb{R}^{n_v \times n_v}, B \in \mathbb{R}^{n_v \times n_r}, C \in \mathbb{R}^{n_a \times n_v}$
- $n_v > n_p$
- matrix $M \in \mathbb{R}^{n_v \times n_v}$ need to be positive definite.
- $G \in \mathbb{R}^{n_v \times n_p}$ and G has column rank n_p

The following matrix pencil belongs to that system

$$\left(\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}, \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \right).$$

To avoid infinite eigenvalues, we use

$$\left(\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}, \begin{bmatrix} M & \delta G \\ \delta G^T & 0 \end{bmatrix} \right)$$

as matrix pencil for shift parameter computation in mess_lrcfadi_parameter. This transforms all infinite eigenvalues to $\frac{1}{\delta}$.

Depending on the component mess_options::type a normal or transposed Π^T -projected Riccati Equation is created.
The discrete Leray projector is applied implicitly. For more details look at [2] [3] .

Depending on memory_usage component of the mess_options structure the memory usage is controlled. mess_options::memory_usage = 0: In every iteration a new solver is computed in ApEINV_apply. mess_options::memory_usage = 1: For every shift a solver is precomputed and reused in ApEINV_apply.

mess_options::type = MESS_OP_NONE:

$$\Pi A \Pi^T X \Pi M^T \Pi^T + \Pi M \Pi^T X \Pi A^T \Pi^T - \Pi M \Pi^T X \Pi C^T C \Pi^T X \Pi M^T \Pi^T + \Pi B B^T \Pi^T = 0$$

mess_options::type = MESS_OP_TRANSPOSE:

$$\Pi A^T \Pi^T X \Pi M \Pi^T + \Pi M^T \Pi^T X \Pi A \Pi^T - \Pi M^T \Pi^T X \Pi B B^T \Pi^T X \Pi M \Pi^T + \Pi C^T C \Pi^T = 0$$

If your system is unstable use the `K0` component of mess_options structure to provide an initial stabilizing feedback for mess_lrcfadi_nm.
This function sets the B and C component of the given mess_equation_st structure.

mess_options::type = MESS_OP_NONE:

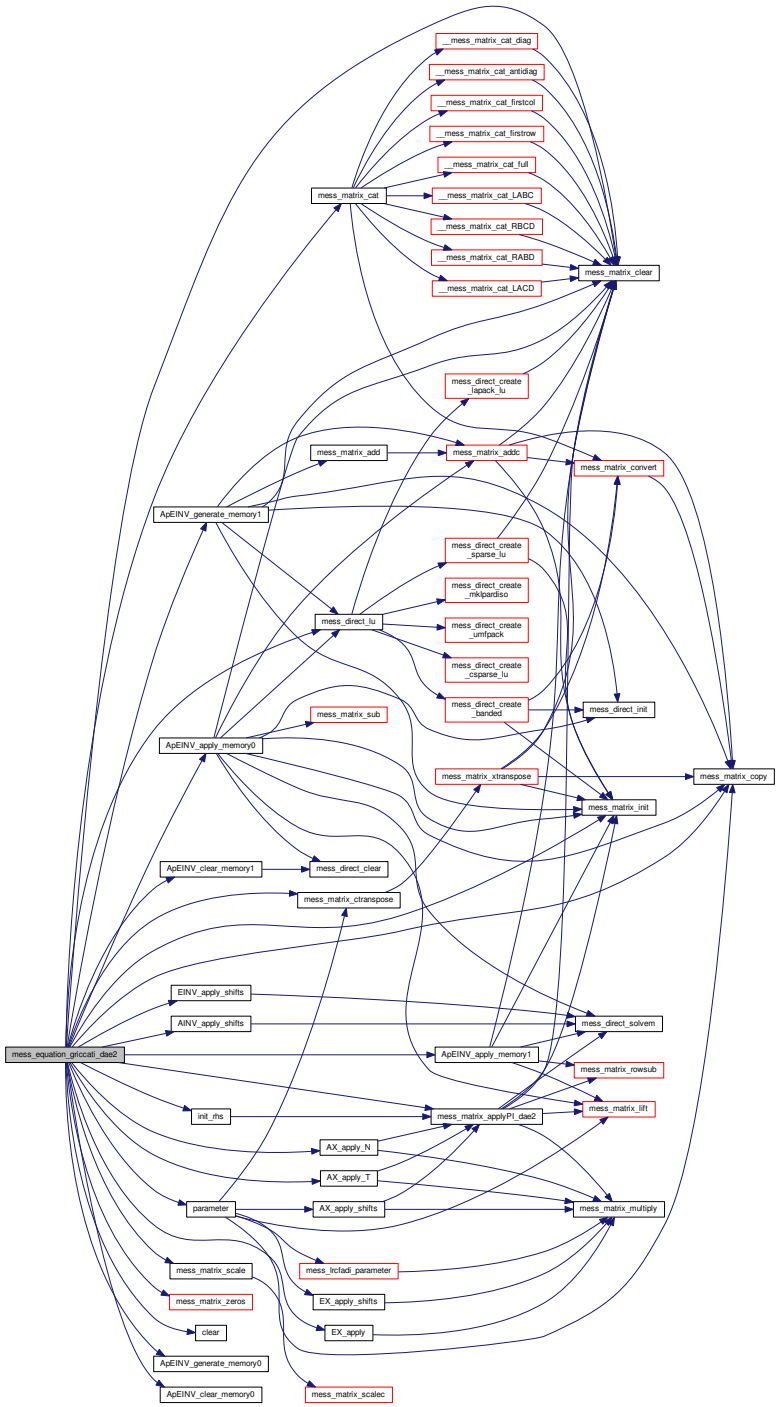
$$\begin{aligned} B &= \Pi B \\ C &= C \end{aligned}$$

mess_options::type = MESS_OP_TRANSPOSE:

$$\begin{aligned} B &= B \\ C &= C \Pi^T. \end{aligned}$$

```
\sa mess_equation_lyap
\sa mess_equation_st
\sa mess_lrcfadi_nm
```

Here is the call graph for this function:



8.41.2.9 int mess_equation_griccati_so1 (mess_equation *e*, mess_options *opt*, mess_matrix *M*, mess_matrix *D*, mess_matrix *K*, mess_matrix *B*, mess_matrix *C*, double *lowerbound*, double *upperbound*)

Parameters

out	*e*	equation object to setup
in	*opt*	input options for the ADI iteration

Parameters

in	*M*	input M matrix of the LTI system
in	*D*	input D matrix of the LTI system
in	*K*	input K matrix of the LTI system
in	*B*	input B matrix of the LTI system
in	*C*	input observer matrix of the LTI system
in	*lowerbound*	input lower bound for absolute value of shift parameter
in	*upperbound*	input upper bound for absolute value of shift parameter

Returns

zero on success or a non zero error code otherwise

The mess_equation_griccati_so1 function creates an equation object from a second order LTI system

$$\begin{array}{rcl} M\ddot{x} + D\dot{x} + K \quad x & = & Bu \\ y & = & Cx \end{array}$$

where $M, D, K \in \mathbb{R}^{n \times n}$.
Furthermore K need to be invertible and M positive definite.
The system is implicitly transformed to its first order representation:

$$\begin{array}{rcl} \underbrace{\begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix}}_{\mathcal{E}} \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} & = & \underbrace{\begin{bmatrix} 0 & -K \\ -K & -D \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ B \end{bmatrix}}_{\mathcal{B}} u \\ y & = & \underbrace{C}_{\mathcal{C}} x \quad . \end{array}$$

The Riccati solver mess_lrcfadi_nm will then solve on

$$\mathcal{A}^T X \mathcal{E} + \mathcal{E}^T X \mathcal{A} - \mathcal{E}^T X \mathcal{B} \mathcal{B}^T X \mathcal{E} + \mathcal{C}^T \mathcal{C} = 0$$

or

$$\mathcal{A} X \mathcal{E}^T + \mathcal{E} X \mathcal{A}^T - \mathcal{E} X \mathcal{C}^T \mathcal{C} X \mathcal{E}^T + \mathcal{B} \mathcal{B}^T = 0$$

If mess_options::type MESS_OP_NONE

$$\begin{array}{rcll} \mathcal{B} & = & B & \in \mathbb{R}^{n \times t}, \\ \mathcal{C} & = & \begin{bmatrix} 0 & C \end{bmatrix} & \in \mathbb{R}^{s \times 2n}, \end{array}$$

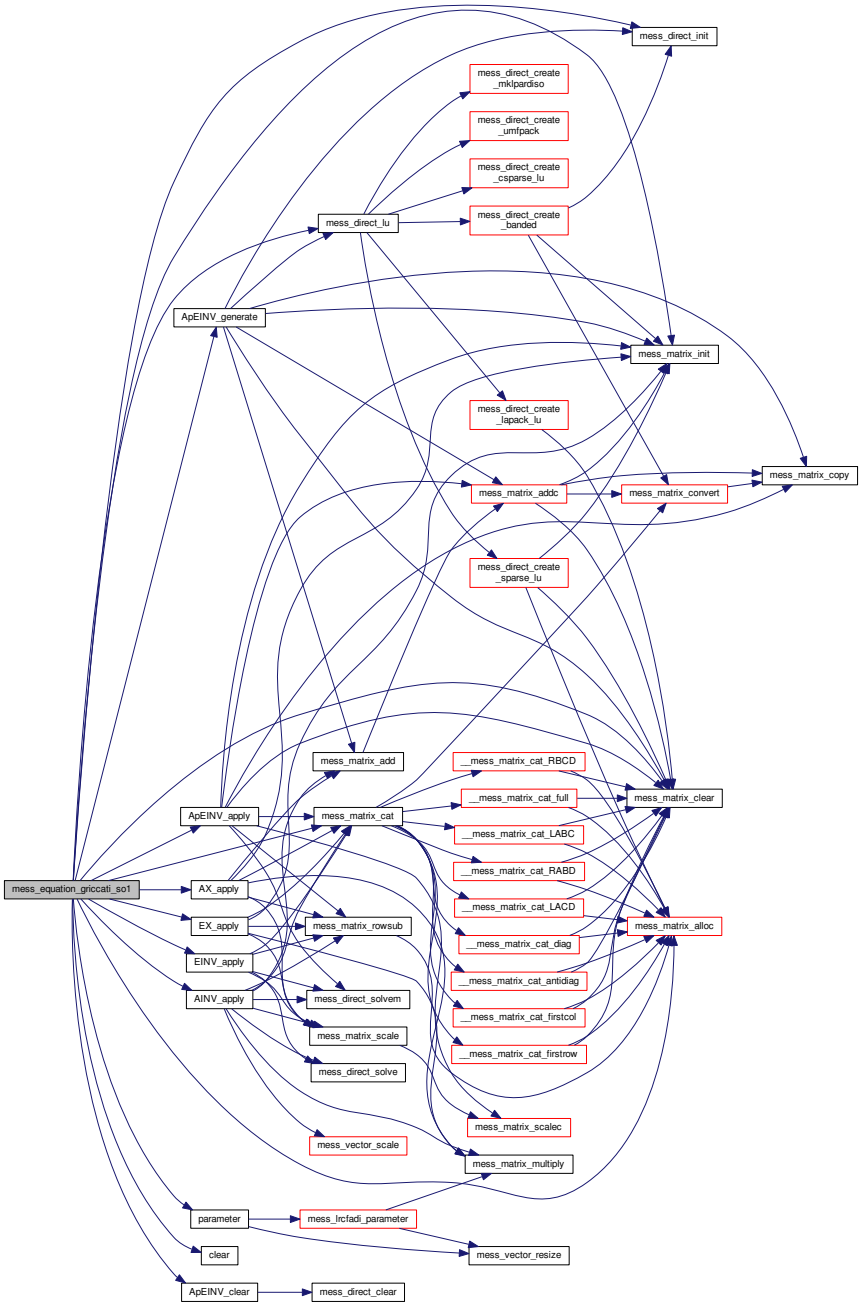
otherwise

$$\begin{array}{rcll} \mathcal{B} & = & \begin{bmatrix} 0 \\ B \end{bmatrix} & \in \mathbb{R}^{2n \times t}, \\ \mathcal{C} & = & C & \in \mathbb{R}^{s \times n} \end{array}$$

```
The linearization of the second order system leads to quadratic shifts.
To avoid numerical problems one can defines lower and upper bounds for the absolute value of the shift.
Shifts that not fullfill the inequalitiy \form#821 are automatically sorted out.
Common choices are \form#832 and \form#833.

\sa mess_lrcfadi_nm
```

Here is the call graph for this function:



8.41.2.10 int mess_equation_griccati_so2 (mess_equation *e*, mess_options *opt*, mess_matrix *M*, mess_matrix *D*, mess_matrix *K*, mess_matrix *B*, mess_matrix *C*, double *lowerbound*, double *upperbound*)

Parameters

out	*e*	equation object to setup
in	*opt*	input options for the ADI iteration
in	*M*	input M matrix of the LTI system
in	*D*	input D matrix of the LTI system

Parameters

in	*K*	input K matrix of the LTI system
in	*B*	input B matrix of the LTI system
in	*C*	input observer matrix of the LTI system
in	*lowerbound*	input lower bound for absolute value of shift parameter
in	*upperbound*	input upper bound for absolute value of shift parameter

Returns

zero on success or a non zero error code otherwise

The mess_equation_griccati_so2 function creates an equation object from a second order LTI system

$$\begin{array}{rcl} M\ddot{x} + D\dot{x} + K \quad x & = & Bu \\ y & = & Cx \end{array}$$

where $M, D, K \in \mathbb{R}^{n \times n}$.
Furthermore K need to be invertible and M positive definite.
The system is implicitly transformed to its first order representation:

$$\underbrace{\begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix}}_{\mathcal{E}} \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ B \end{bmatrix}}_{\mathcal{B}} u$$

and the Riccati solver mess_lrcfadi_nm will then solve on

$$\mathcal{A}^T X \mathcal{E} + \mathcal{E}^T X \mathcal{A} - \mathcal{E}^T X \mathcal{B} \mathcal{B}^T X \mathcal{E} + \mathcal{C}^T \mathcal{C} = 0$$

or

$$\mathcal{A} X \mathcal{E}^T + \mathcal{E} X \mathcal{A}^T - \mathcal{E} X \mathcal{C}^T \mathcal{C} X \mathcal{E}^T + \mathcal{B} \mathcal{B}^T = 0$$

If mess_options::type equals MESS_OP_NONE

$$\begin{array}{rcll} \mathcal{B} & = & B & \in \mathbb{R}^{n \times t}, \\ \mathcal{C} & = & \begin{bmatrix} 0 & C \end{bmatrix} & \in \mathbb{R}^{s \times 2n}, \end{array}$$

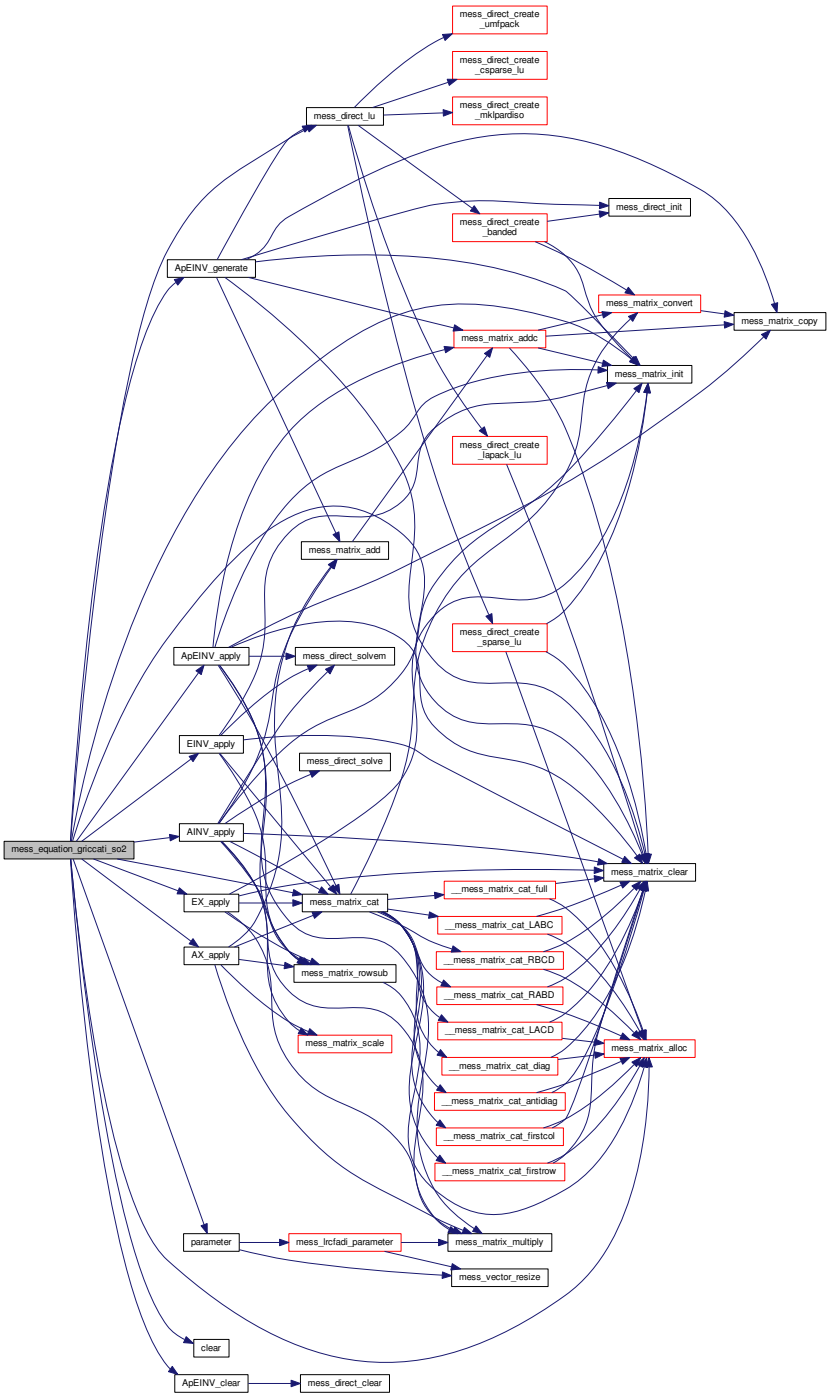
otherwise

$$\begin{array}{rcll} \mathcal{B} & = & \begin{bmatrix} 0 \\ B \end{bmatrix} & \in \mathbb{R}^{2n \times t}, \\ \mathcal{C} & = & C & \in \mathbb{R}^{s \times n} \end{array}$$

```
The linearization of the second order system leads to quadratic shifts.
To avoid numerical problems one can defines lower and upper bounds for the absolute value of the shift.
Shifts that not fullfill the inequalitiy \form#821 are automatically sorted out.
Common choices are \form#832 and \form#833.


\sa mess_lrcfadi_nm
\sa mess_equation_st
```

Here is the call graph for this function:



8.41.2.11 int mess_equation_lyap (mess_equation *e*, mess_options *opt*, mess_matrix *A*, mess_matrix *E*, mess_matrix *B*)

Parameters

in,out	*e*	mess_equation object
in	*opt*	input mess_options object

Parameters

in	*A*	input system matrix of equation
in	*E*	input mass matrix of the equation, NULL if it is the identity
in	*B*	input right hand side ofequation

Returns

zero on success or a non-zero error value otherwise

The mess_equation_lyap function sets up a mess_equation object such that the Lyapunov equation

$$op(A)Xop(E)^T + op(E)Xop(A)^T + op(B)op(B)^T = 0$$

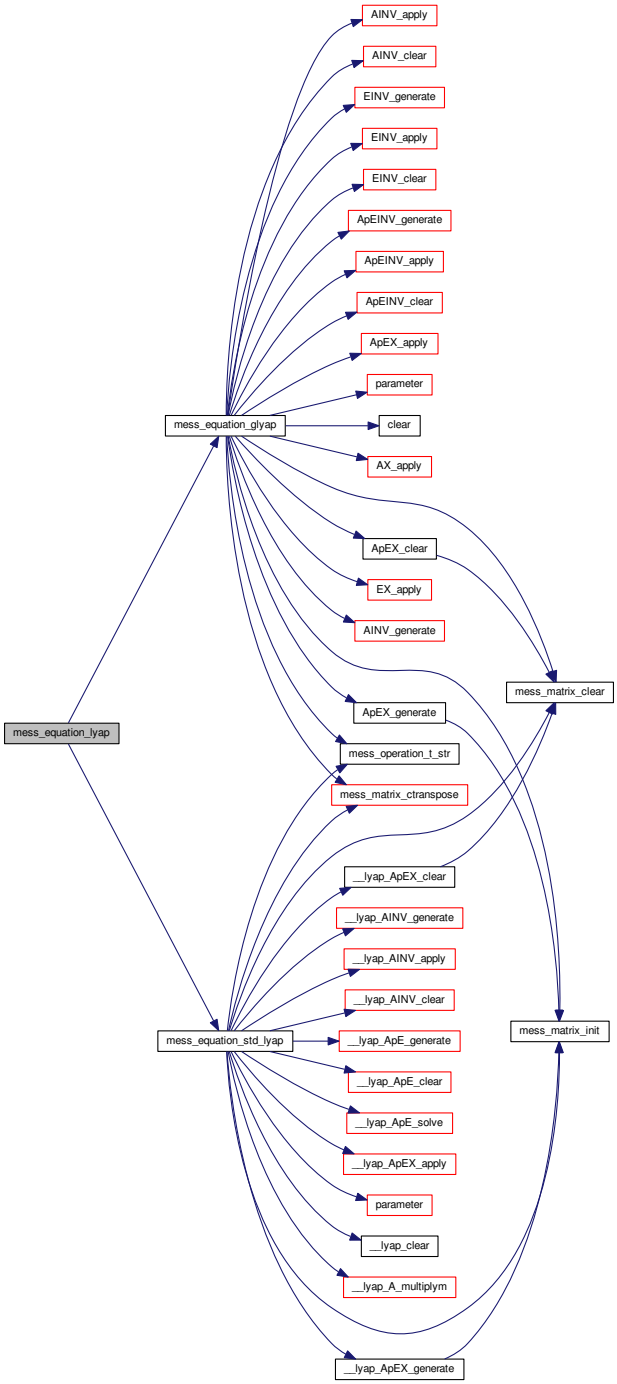
is represented. Matrices need to be real.
The operation is selected by the mess_options::type component.
i.e. $op(A) = A$.
Otherwise if the mess_options::type is set to MESS_OP_TRANSPOSE the transposed Lyapunov Equation will be solved, i.e. $op(A) = A^T$.
The resulting object can be used with the mess_lrcfadi_adi solver.

Depending on if the E parameter is NULL or not the mess_equation_lyap or mess_equation_glyap function is called to setup the equation object.

See also

mess_lrcfadi_adi
mess_equation_lyap
mess_equation_glyap
mess_equation_st
mess_lrcfadi_nm

Here is the call graph for this function:



8.41.2.12 int mess_equation_riccati (mess_equation *e*, mess_options *opt*, mess_matrix *A*, mess_matrix *E*, mess_matrix *B*, mess_matrix *C*)

Parameters

in,out	*e*	mess_equation object
in	*opt*	input mess_options object

Parameters

in	*A*	input system matrix of equation
in	*E*	input mass matrix of the equation, NULL if it is the identity
in	*B*	input factor of the matrix in quadratic term
in	*C*	input right hand side

Returns

zero on success or a non-zero error value otherwise

The mess_equation_riccati function sets up a mess_equation object such that the Riccati equation

$$AXE^T + EXA^T - EXC^TCXE^T + BB^T = 0$$

or the transposed one

$$A^TXE + E^TXA - E^TXBB^TXE + C^TC = 0$$

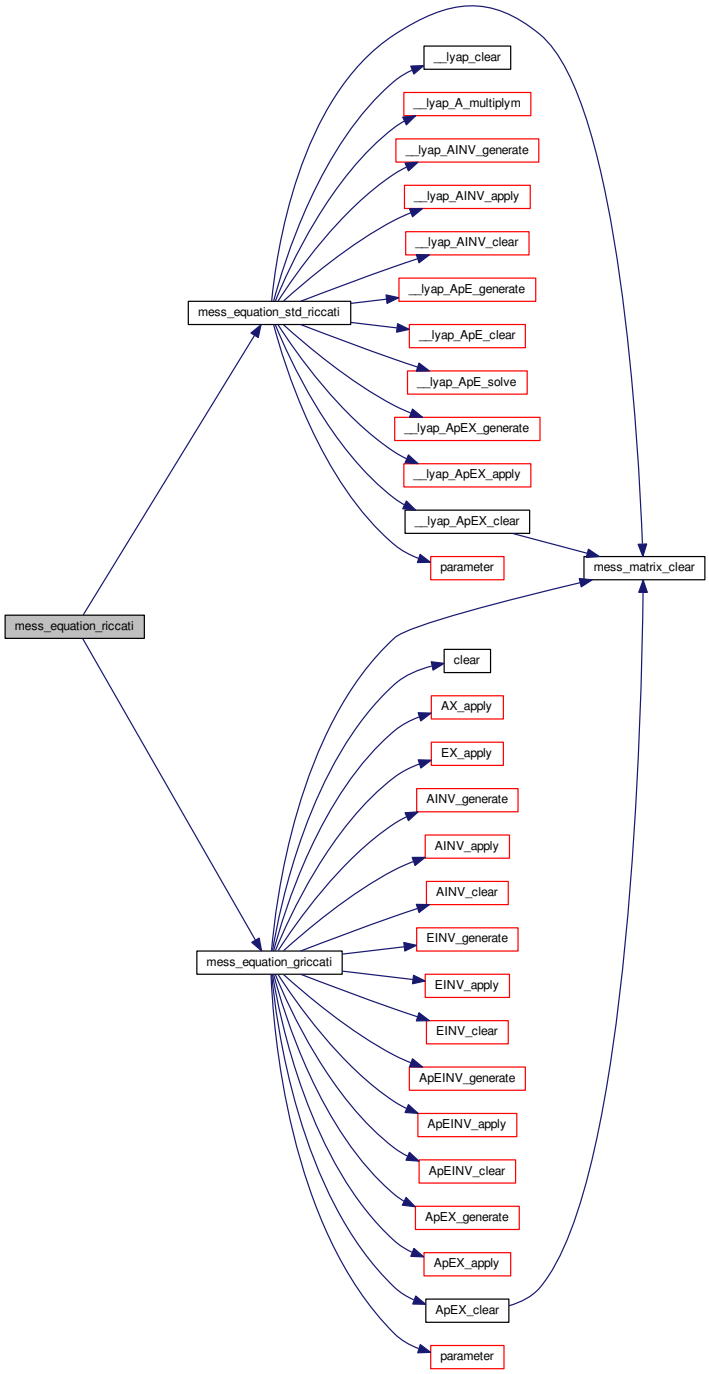
is represented.
Depending on the mess_options::type component the first or the second equation is solved.
The first one corresponds to the mess_options::type that is equal to MESS_OP_NONE and the second one corresponds to the mess_options::type that is equal to MESS_OP_TRANSPOSE.

Depending on if the E parameter is NULL or not the mess_equation_riccati or mess_equation_griccati function is called to setup the equation object.

See also

mess_lrcfadi_adi
mess_equation_riccati
mess_equation_griccati
mess_equation_st
mess_lrcfadi_nm

Here is the call graph for this function:



8.41.2.13 int mess_equation_stable (mess_equation *e*, mess_options *opt*, mess_equation *child*, mess_matrix *U*, mess_matrix *V*)

Parameters

out	*e*	generated object
in	*opt*	input options for Lyapunov Equation
in	*child*	input child equation to use
in	*U*	input matrix
in	*V*	input matrix

Returns

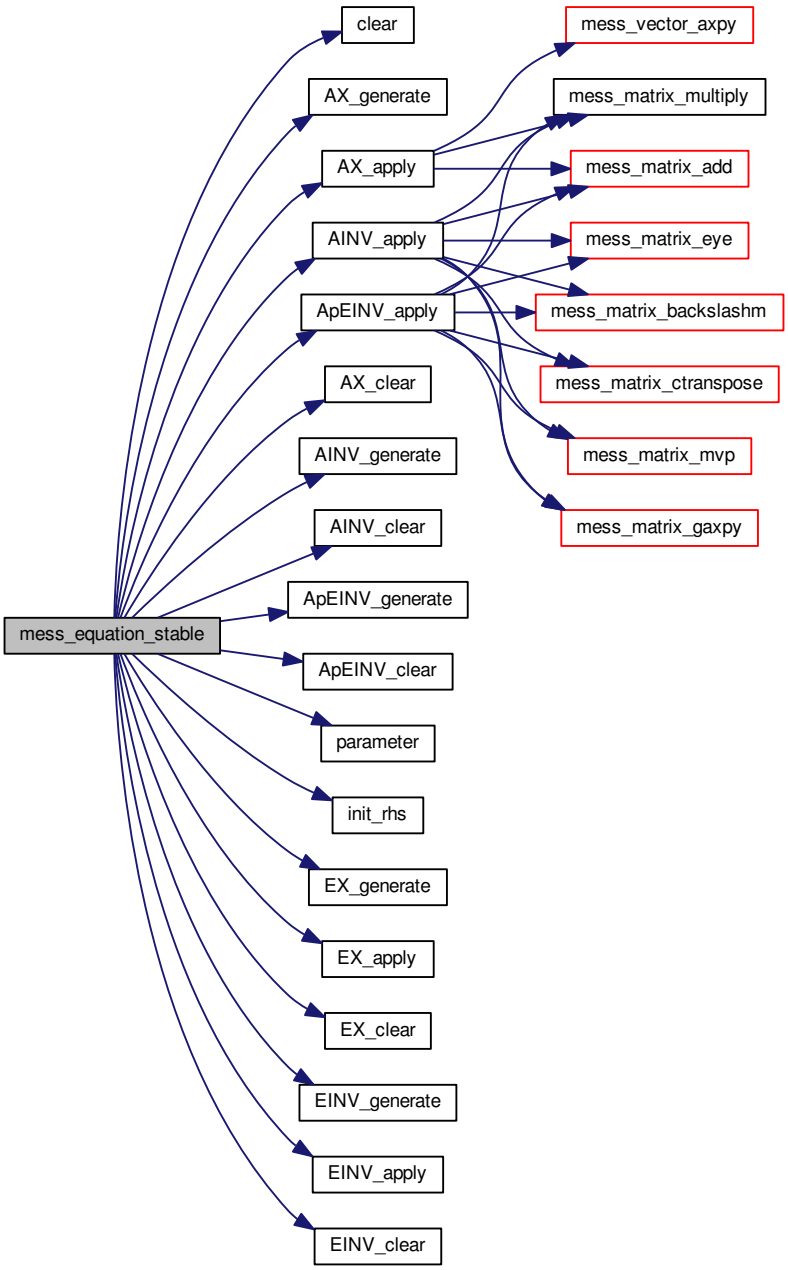
zero on success or a non zero error code otherwise

The mess_equation_stable function generates a stabilized operator $(A-UV)$ to solve. The resulting operator uses the function handles defined in `child` and the Sherman-Morrison-Woodbury formula to perform solving with the stabilized operator. Please not that `U` and `V` must have the same size and `U` and `V` are referenced by the stabilized equation `e`.

See also

mess_equation_lyap
mess_equation_glyap

Here is the call graph for this function:



8.41.2.14 int mess_equation_std_lyap (mess_equation *e*, mess_options *opt*, mess_matrix *A*, mess_matrix *B*)

Parameters

in,out	*e*	mess_equation object
in	*opt*	input mess_options object
in	*A*	input system matrix of equation
in	*B*	input right hand side ofequation

Returns

zero on success or a non-zero error value otherwise

The mess_equation_std_lyap function sets up a mess_equation object such that the Lyapunov equation

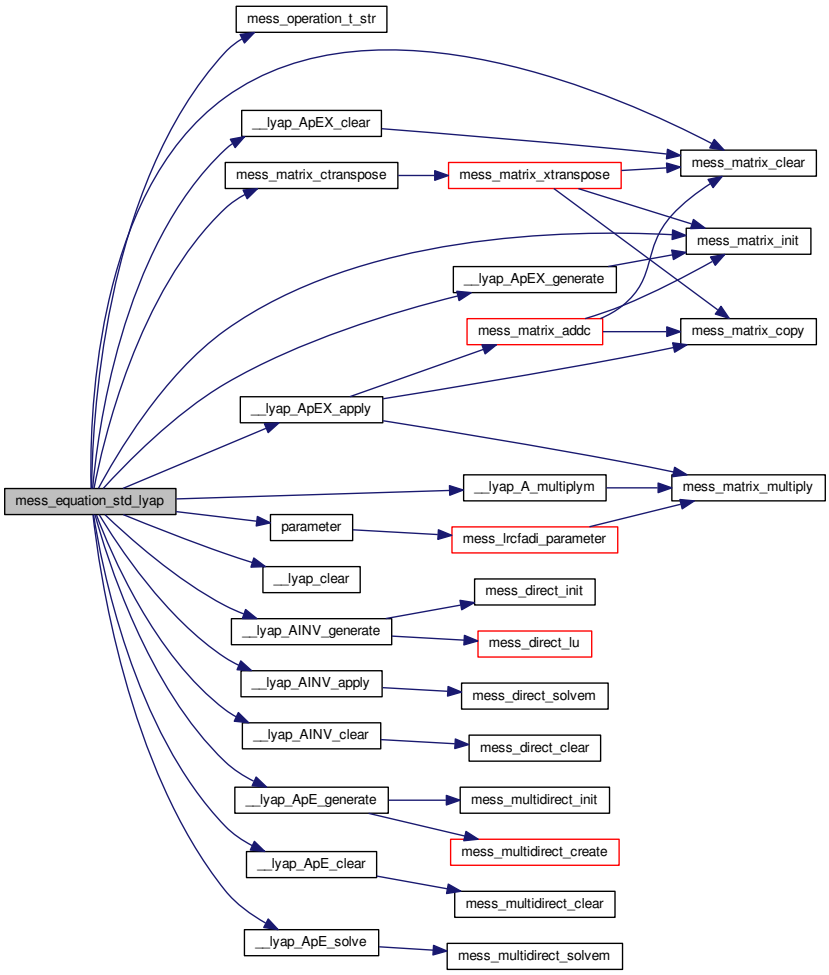
$$op(A)X + Xop(A)^T + op(B)op(B)^T = 0$$

is represented. Matrices need to be real.
The operation is selected by the mess_options::type component of the structure.
If the mess_options::type is set to MESS_OP_NONE the ADI process will solve the standard Lyapunov Equation, i.e. $op(A) = A$.
Otherwise if mess_options::type is set to MESS_OP_TRANSPOSE the transposed Lyapunov Equation will be solved, i.e. $op(A) = A^T$.
The resulting object can be used with the mess_lrcfadi_adi solver.

See also

mess_lrcfadi_adi
mess_equation_glyap
mess_equation_st
mess_lrcfadi_nm

Here is the call graph for this function:



8.41.2.15 int mess_equation_std_riccati (mess_equation *e*, mess_options *opt*, mess_matrix *A*, mess_matrix *B*, mess_matrix *C*)

Parameters

in,out	*e*	mess_equation object
in	*opt*	input mess_options object
in	*A*	input system matrix of equation
in	*B*	input factor of the matrix in quadratic term
in	*C*	input right hand side

Returns

zero on success or a non-zero error value otherwise

The mess_equation_std_riccati function sets up a mess_equation object such that the Riccati equation

$$AX + XA^T - XC^TCX + BB^T = 0$$

or the transposed one

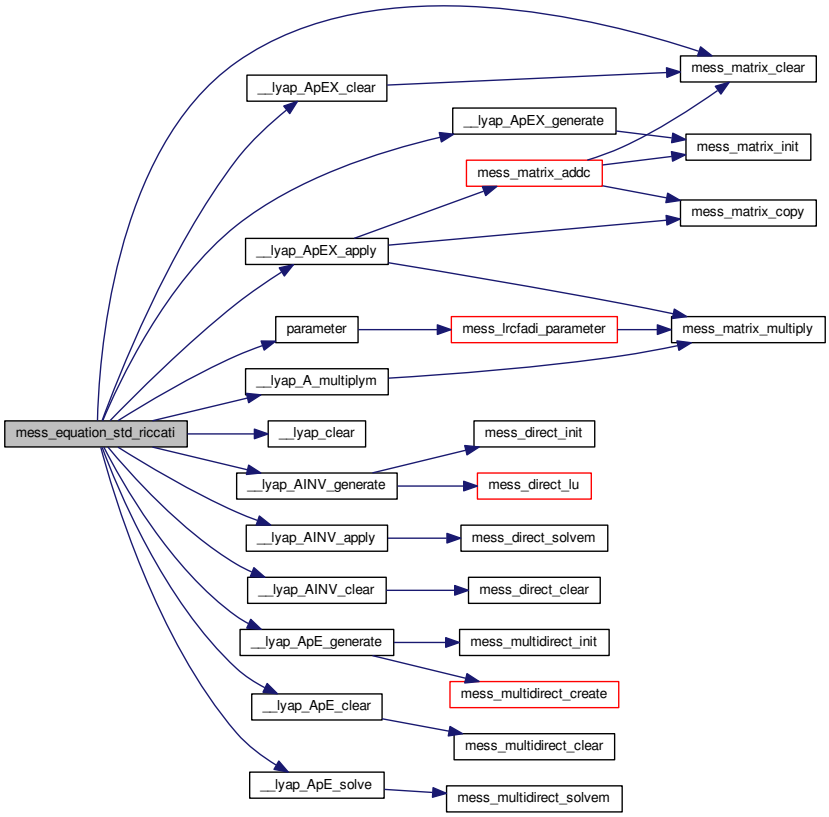
$$A^TX + XA - XBB^TX + C^TC = 0$$

is represented.
Depending on the mess_options::type component the first or the second equation is solved.
The first one corresponds to a mess_options::type that is equal to MESS_OP_NONE and the second one corresponds to a mess_options::type that is equal to MESS_OP_TRANSPOSE.

See also

mess_lrcfadi_adi
mess_equation_griccati
mess_equation_st
mess_lrcfadi_nm

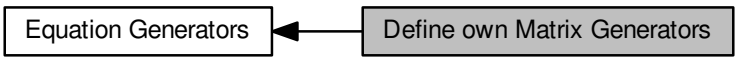
Here is the call graph for this function:



8.42 Define own Matrix Generators

Defining own Matrix Generators by implementing basic operations.

Collaboration diagram for Define own Matrix Generators:



Functions

- mess_int_t mess_equation_dim (mess_equation eqn)

  Return the dimension of the equation of object.
- mess_int_t mess_equation_has_A (mess_equation eqn)

  Checks if $x = \mathcal{A}y$ is implemented.
- mess_int_t mess_equation_has_As (mess_equation eqn)

  Checks if $\mathcal{A}x = y$ is implemented.
- mess_int_t mess_equation_has_E (mess_equation eqn)

  Checks if $x = \mathcal{E}y$ is implemented.
- mess_int_t mess_equation_has_Es (mess_equation eqn)

  Checks if $\mathcal{E}x = y$ is implemented.
- mess_int_t mess_equation_has_ApE (mess_equation eqn)

  Checks if $x = (\mathcal{A} + \mathcal{E})y$ is implemented.
- mess_int_t mess_equation_has_ApEs (mess_equation eqn)

  Checks if $(\mathcal{A} + p\mathcal{E})x = y$ is implemented.
- int mess_equation_A_pre (mess_equation eqn)

  Initialization function of the linear mapping $x = \mathcal{A}y$.
- int mess_equation_A_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $x = \mathcal{A}y$.
- int mess_equation_A_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute $x = \mathcal{A}y$.
- int mess_equation_A_apply_vector (mess_equation eqn, mess_operation_t op, mess_vector in, mess_vector out)

  Compute $x = \mathcal{A}y$ vector valued.
- int mess_equation_E_pre (mess_equation eqn)

  Initialization function of the linear mapping $x = \mathcal{E}y$.
- int mess_equation_E_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $x = \mathcal{E}y$.
- int mess_equation_E_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute $x = \mathcal{E}y$.
- int mess_equation_E_apply_vector (mess_equation eqn, mess_operation_t op, mess_vector in, mess_vector out)

  Compute $x = \mathcal{E}y$ vector valued.
- int mess_equation_As_pre (mess_equation eqn)

  Initialization function of the linear mapping $\mathcal{A}x = y$.

- int mess_equation_As_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $\mathcal{A}x = y$.
- int mess_equation_As_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute $\mathcal{A}x = y$.
- int mess_equation_As_apply_vector (mess_equation eqn, mess_operation_t op, mess_vector in, mess_vector out)

  Compute $\mathcal{A}x = y$ vector valued.
- int mess_equation_Es_pre (mess_equation eqn)

  Initialization function of the linear mapping $\mathcal{E}x = y$.
- int mess_equation_Es_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $\mathcal{E}x = y$.
- int mess_equation_Es_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute $\mathcal{E}x = y$.
- int mess_equation_Es_apply_vector (mess_equation eqn, mess_operation_t op, mess_vector in, mess_vector out)

  Compute $\mathcal{E}\S = y$ vector valued.
- int mess_equation_ApE_pre (mess_equation eqn, mess_vector parameters)

  Initialization function of the linear mapping $x = (\mathcal{A} + p\mathcal{E})y$.
- int mess_equation_ApE_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $x = (\mathcal{A} + \mathcal{E})y$.
- int mess_equation_ApE_apply (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t p_idx, mess_matrix in, mess_matrix out)

  Compute $x = (\mathcal{A} + p\mathcal{E})y$.
- int mess_equation_ApE_apply_vector (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t p_idx, mess_vector in, mess_vector out)

  Compute $x = (\mathcal{A} + p\mathcal{E})y$ with vectors..
- int mess_equation_ApEs_pre (mess_equation eqn, mess_vector parameters)

  Initialization function of the linear mapping $(\mathcal{A} + p\mathcal{E})x = y$.
- int mess_equation_ApEs_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $(\mathcal{A} + \mathcal{E})x = y$.
- int mess_equation_ApEs_apply (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t p_idx, mess_matrix in, mess_matrix out)

  Compute $(\mathcal{A} + p\mathcal{E})x = y$.
- int mess_equation_ApEs_apply_vector (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t p_idx, mess_vector in, mess_vector out)

  Compute $(\mathcal{A} + p\mathcal{E})x = y$ with vectors..

8.42.1 Detailed Description

8.42.2 Function Documentation

8.42.2.1 int mess_equation_A_apply (mess_equation *eqn,* mess_operation_t *op,* mess_matrix *in,* mess_matrix *out*)

Parameters

in	*eqn*	input Equation object $x = \mathcal{A}y$ blongs to
in	*op*	input Opertion to apply
in	*in*	input Input matrix y
in	*out*	input Ouput matrix x

Returns

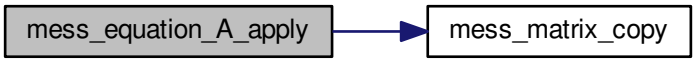
zero on success or a non zero error code

The mess_equation_A_apply function computes $x = op(\mathcal{A})x$, where x and y are matrices. For vectors use mess_equation_A_apply_vector instead. If the object does not provide a AX.apply function, $\mathcal{A} = I$ is assumed.

See also

mess_equation_A_apply_vector
mess_equation_E_apply

Here is the call graph for this function:



8.42.2.2 int mess_equation_A_apply_vector (mess_equation *eqn*, mess_operation_t *op*, mess_vector *in*, mess_vector *out*)

Parameters

in	*eqn*	input Equation object $x = \mathcal{A}y$ blongs to
in	*op*	input Opertion to apply
in	*in*	input Input vector y
in	*out*	input Ouput vector x

Returns

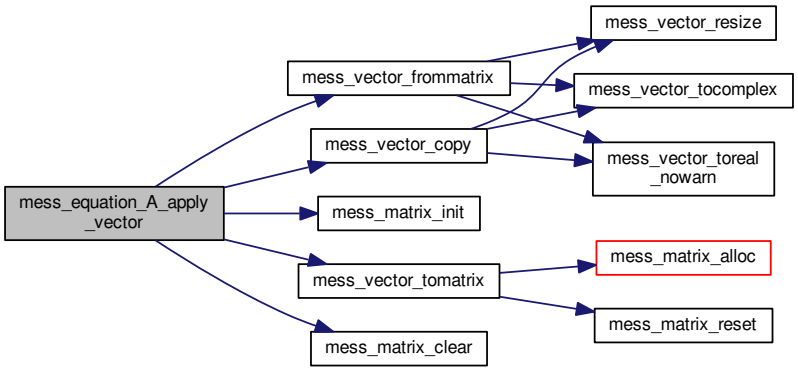
zero on success or a non zero error code

The mess_equation_A_apply_vector function computes $x = op(\mathcal{A})x$, where x and y are vectors. For matrices use mess_equation_A_apply. If the object does not provide a AX.apply function, $\mathcal{A} = I$ is assumed.

See also

mess_equation_A_apply
mess_equation_E_apply

Here is the call graph for this function:



8.42.2.3 int mess_equation_A_post (mess_equation *eqn*)

Parameters

in	*eqn*	input Equation object $x = \mathcal{A}y$ belongs to

Returns

zero on success or a non zero error code

The mess_equation_A_post function finalizes the $x = \mathcal{A}y$ operator. That means it calls the clear function of the equation object if it exits otherwise it will return immediately.

See also

mess_equation_A_pre
mess_equation_A_apply
mess_equation_E_apply

8.42.2.4 int mess_equation_A_pre (mess_equation *eqn*)

Parameters

in	*eqn*	input Equation object $\mathcal{A}$ belongs to

Returns

zero on success or a non zero error code

The mess_equation_A_pre function initializes the $x = \mathcal{A}y$ operation of a given equation object. It is called normally once before the operator is uses the first time. If the equation object does not provide an initialization functions the mess_equation_A_pre function returns immediately.

See also

mess_equation_A_apply
mess_equation_A_post
mess_equation_E_apply

8.42.2.5 int mess_equation_ApE_apply (mess_equation *eqn*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *p_idx*, mess_matrix *in*, mess_matrix *out*)

Parameters

in	*eqn*	input equation object $x = (\mathcal{A} + p\mathcal{E})y$
in	*op*	input opertion to apply
in	*p*	input current shift parameter
in	*p_idx*	input postion of the current parameter in the previously passed parameter set
in	*in*	input matrix y
in	*out*	input matrix x

Returns

zero on success or a non zero error code

The mess_equation_ApE_apply function computes $x = op(\mathcal{A} + p\mathcal{E})x$, where x and y are matrices. For vectors use mess_equation_ApE_apply_vector instead. If the object does not provide a ApE.apply function, an error is returned. The parameter is given by its value p and the postition in the parameter vector passed to the mess_equation_ApE_pre function before. If the index p_idx is lower than 0, the operator should be implemented such that the value of p is used.

See also

mess_equation_A_apply
mess_equation_E_apply

8.42.2.6 int mess_equation_ApE_apply_vector (mess_equation *eqn*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *p_idx*, mess_vector *in*, mess_vector *out*)

Parameters

in	*eqn*	input Equation object $x = (\mathcal{A} + p\mathcal{E})y$
in	*op*	input Opertion to apply
in	*p*	input Current shift parameter
in	*p_idx*	input Postiion of the current parameter in the previously passed parameter set
in	*in*	input Input vector y
in	*out*	input Ouput vector x

Returns

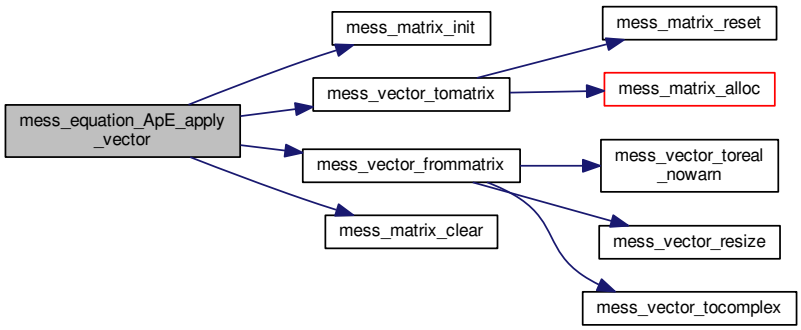
zero on success or a non zero error code

The mess_equation_ApE_apply_vector function computes $x = op(\mathcal{A} + p\mathcal{E})x$, where x and y are vectors. For matrices use mess_equation_ApE_apply instead. If the object does not provide a ApE.apply function, an error is returned. The parameter is given by its value p and the postition in the parameter vector passed to the mess_equation_ApE_pre function before. If the index p_idx is lower than 0, the operator should be implemented such that the value of p is used.

See also

mess_equation_A_apply_vector
mess_equation_E_apply_vector

Here is the call graph for this function:



8.42.2.7 int mess_equation_ApE_post (mess_equation *eqn*)

Parameters

in	*eqn*	input Equation object $x = (\mathcal{A} + \mathcal{E})y$ belongs to

Returns

zero on success or a non zero error code

The mess_equation_ApE_post function finalizes the $x = (\mathcal{A} + \mathcal{E})y$ operator. That means it calls the clear function of the equation object if it exits otherwise it will return immediately.

See also

mess_equation_ApE_pre
mess_equation_ApE_apply

8.42.2.8 int mess_equation_ApE_pre (mess_equation *eqn*, mess_vector *parameters*)

Parameters

in	*eqn*	input Equation object $\mathcal{A} + p\mathcal{E}$ belongs to
in	*parameters*	input Vector containing the shift parameters

Returns

zero on success or a non zero error code

The mess_equation_ApE_pre function initializes the $x = (\mathcal{A} + \mathcal{E})y$ operation of a given equation object. It is called normally once before the operator is uses the first time. If the equation object does not provide an initialization functions the mess_equation_ApE_pre function returns immediately.

See also

mess_equation_ApE_apply
mess_equation_ApE_post

8.42.2.9 int mess_equation_ApEs_apply (mess_equation *eqn*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *p_idx*, mess_matrix *in*, mess_matrix *out*)

Parameters

in	*eqn*	input equation object $(\mathcal{A} + p\mathcal{E})x = y$
in	*op*	input opertion to apply
in	*p*	input current shift parameter
in	*p_idx*	input postion of the current parameter in the previously passed parameter set
in	*in*	input matrix y
in	*out*	input matrix x

Returns

zero on success or a non zero error code

The mess_equation_ApEs_apply function computes $op(\mathcal{A} + p\mathcal{E})x = y$, where x and y are matrices. For vectors use mess_equation_ApEs_apply_vector instead. If the object does not provide a ApE.apply function, an error is returned. The parameter is given by its value p and the postition in the parameter vector passed to the mess_equation_ApEs_pre function before. If the index p_idx is lower than 0, the operator should be implemented such that the value of p is used.

See also

mess_equation_As_apply
mess_equation_Es_apply

8.42.2.10 int mess_equation_ApEs_apply_vector (mess_equation *eqn*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *p_idx*, mess_vector *in*, mess_vector *out*)

Parameters

in	*eqn*	input Equation object $(\mathcal{A} + p\mathcal{E})x = y$

Parameters

in	*op*	input Opertion to apply
in	*p*	input Current shift parameter
in	*p_idx*	input Postion of the current parameter in the previously passed parameter set
in	*in*	input vector y
out	*out*	output vector x

Returns

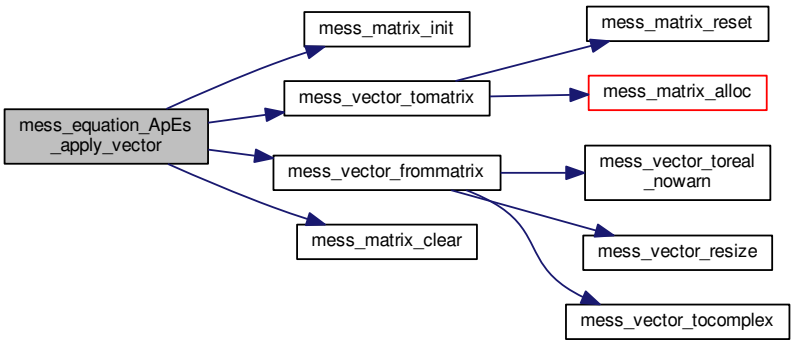
zero on success or a non zero error code

The mess_equation_ApEs_apply_vector function computes $op(\mathcal{A} + p\mathcal{E})x = y$, where x and y are vectors. For matrices use mess_equation_ApEs_apply instead. If the object does not provide a ApE.apply function, an error is returned. The parameter is given by its value p and the postition in the parameter vector passed to the mess_equation_ApEs_pre function before. If the index p_idx is lower than 0, the operator should be implemented such that the value of p is used.

See also

mess_equation_A_apply_vector
mess_equation_E_apply_vector

Here is the call graph for this function:



8.42.2.11 int mess_equation_ApEs_post (mess_equation *eqn*)

Parameters

in	*eqn*	input Equation object $(\mathcal{A} + \mathcal{E})x = y$ belongs to

Returns

zero on success or a non zero error code

The mess_equation_ApEs_post function finalizes the $(\mathcal{A}+\mathcal{E})x = y$ operator. That means it calls the clear function of the equation object if it exits otherwise it will return immediately.

See also

mess_equation_ApEs_pre
mess_equation_ApEs_apply

8.42.2.12 int mess_equation_ApEs_pre (mess_equation *eqn,* mess_vector *parameters*)

Parameters

in	*eqn*	input Equation object $\mathcal{A} + p\mathcal{E}$ belongs to
in	*parameters*	input Vector containing the shift parameters

Returns

zero on success or a non zero error code

The mess_equation_ApEs_pre function initializes the $(\mathcal{A}+\mathcal{E})x = y$ operation of a given equation object. It is called normally once before the operator is uses the first time. If the equation object does not provide an initialization functions the mess_equation_ApEs_pre function returns immediately.

See also

mess_equation_ApEs_apply
mess_equation_ApEs_post

8.42.2.13 int mess_equation_As_apply (mess_equation *eqn,* mess_operation_t *op,* mess_matrix *in,* mess_matrix *out*)

Parameters

in	*eqn*	input Equation object $\mathcal{A}x = y$ blongs to
in	*op*	input Opertion to apply
in	*in*	input Input matrix y
in	*out*	input Ouput matrix x

Returns

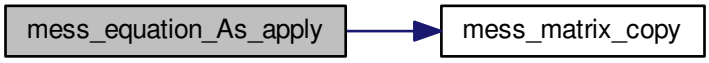
zero on success or a non zero error code

The mess_equation_As_apply function computes $op(\mathcal{A})x = y$, where x and y are matrices. For vectors use mess_equation_As_apply_vector instead. If the object does not provide a AINV.apply function, $\mathcal{A} = I$ is assumed.

See also

mess_equation_As_apply_vector
mess_equation_Es_apply

Here is the call graph for this function:



8.42.2.14 int mess_equation_As_apply_vector (mess_equation *eqn*, mess_operation_t *op*, mess_vector *in*, mess_vector *out*)

Parameters

in	*eqn*	input Equation object $\mathcal{A}x = y$ blongs to
in	*op*	input Opertion to apply
in	*in*	input Input vector y
in	*out*	input Ouput vector x

Returns

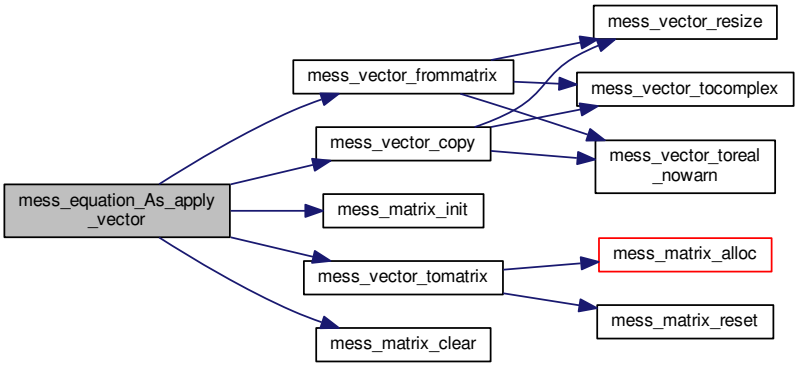
zero on success or a non zero error code

The mess_equation_As_apply_vector function computes $op(\mathcal{A})x = y$, where x and y are vectors. For matrices use mess_equation_As_apply. If the object does not provide a AX.apply function, $\mathcal{A} = I$ is assumed.

See also

mess_equation_As_apply
mess_equation_Es_apply

Here is the call graph for this function:



8.42.2.15 int mess_equation_As_post (mess_equation *eqn*)

Parameters

in	*eqn*	input Equation object $\mathcal{A}x = y$ belongs to

Returns

zero on success or a non zero error code

The mess_equation_As_post function finalizes the $\mathcal{A}x = y$ operator. That means it calls the clear function of the equation object if it exits otherwise it will return immediately.

See also

mess_equation_As_pre
mess_equation_As_apply
mess_equation_Es_apply

8.42.2.16 int mess_equation_As_pre (mess_equation *eqn*)

Parameters

in	*eqn*	input Equation object $\mathcal{A}$ belongs to

Returns

zero on success or a non zero error code

The mess_equation_As_pre function initializes the $\mathcal{A}x = y$ operation of a given equation object. It is called normally once before the operator is uses the first time. If the equation object does not provide an initialization functions the mess_equation_As_pre function returns immediately.

See also

mess_equation_As_apply
mess_equation_As_post
mess_equation_Es_apply

8.42.2.17 mess_int_t mess_equation_dim (mess_equation *eqn*)

Parameters

in	*eqn*	input Equation object

Returns

The dimension or a negative error code

The mess_equation_dim function returns the dimension of the equation object.

8.42.2.18 int mess_equation_E_apply (mess_equation *eqn*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*)

Parameters

in	*eqn*	input Equation object $x = \mathcal{E}y$ blongs to
in	*op*	input Opertion to apply
in	*in*	input Input matrix y
in	*out*	input Ouput matrix x

Returns

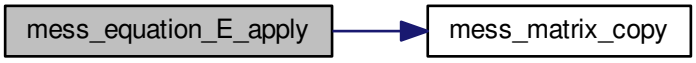
zero on success or a non zero error code

The mess_equation_E_apply function computes $x = op(\mathcal{E})x$, where x and y are matrices. For vectors use mess_equation_E_apply_vector instead. If the object does not provide a EX.apply function, $\mathcal{E} = I$ is assumed.

See also

mess_equation_E_apply_vector
mess_equation_E_apply

Here is the call graph for this function:



8.42.2.19 int mess_equation_E_apply_vector (mess_equation *eqn*, mess_operation_t *op*, mess_vector *in*, mess_vector *out*)

Parameters

in	*eqn*	input Equation object $x = \mathcal{E}y$ blongs to
in	*op*	input Opertion to apply
in	*in*	input Input vector y
in	*out*	input Ouput vector x

Returns

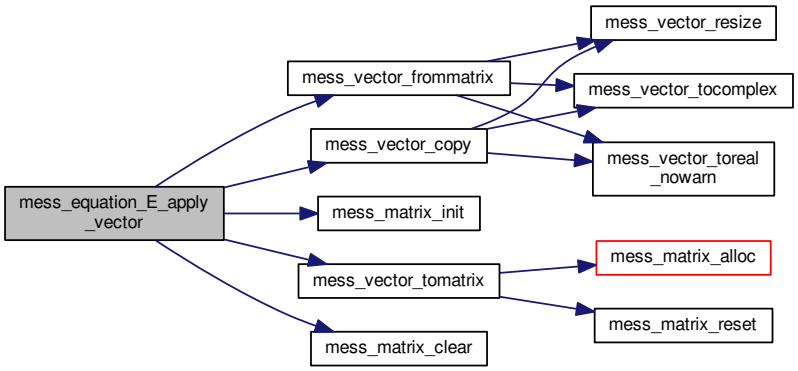
zero on success or a non zero error code

The mess_equation_E_apply_vector function computes $x = op(\mathcal{E})x$, where x and y are vectors. For matrices use mess_equation_E_apply. If the object does not provide a EX.apply function, $\mathcal{E} = I$ is assumed.

See also

mess_equation_E_apply
mess_equation_E_apply

Here is the call graph for this function:



8.42.2.20 int mess_equation_E_post (mess_equation *eqn*)

Parameters

in	*eqn*	input Equation object $x = \mathcal{E}y$ belongs to

Returns

zero on success or a non zero error code

The mess_equation_E_post function finalizes the $x = \mathcal{E}y$ operator. That means it calls the clear function of the equation object if it exits otherwise it will return immediately.

See also

mess_equation_E_pre
mess_equation_E_apply
mess_equation_E_apply

8.42.2.21 int mess_equation_E_pre (mess_equation *eqn*)

Parameters

in	*eqn*	input Equation object $\mathcal{E}$ belongs to

Returns

zero on success or a non zero error code

The mess_equation_E_pre function initializes the $x = \mathcal{E}y$ operation of a given equation object. It is called normally once before the operator is uses the first time. If the equation object does not provide an initialization functions the mess_equation_E_pre function returns immediately.

See also

mess_equation_E_apply
mess_equation_E_post
mess_equation_E_apply

8.42.2.22 int mess_equation_Es_apply (mess_equation *eqn*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*)

Parameters

in	*eqn*	input Equation object $\mathcal{E}x = y$ blongs to
in	*op*	input Opertion to apply
in	*in*	input Input matrix y
in	*out*	input Ouput matrix x

Returns

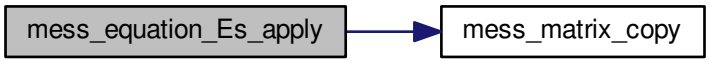
zero on success or a non zero error code

The mess_equation_Es_apply function computes $op(\mathcal{E})x = y$, where x and y are matrices. For vectors use mess↩_equation_Es_apply_vector instead. If the object does not provide a EINV.apply function, $\mathcal{E} = I$ is assumed.

See also

mess_equation_Es_apply_vector
mess_equation_As_apply

Here is the call graph for this function:



8.42.2.23 int mess_equation_Es_apply_vector (mess_equation *eqn*, mess_operation_t *op*, mess_vector *in*, mess_vector *out*)

Parameters

in	*eqn*	input Equation object $\mathcal{E}x = y$ blongs to
in	*op*	input Opertion to apply
in	*in*	input Input vector y
in	*out*	input Ouput vector x

Returns

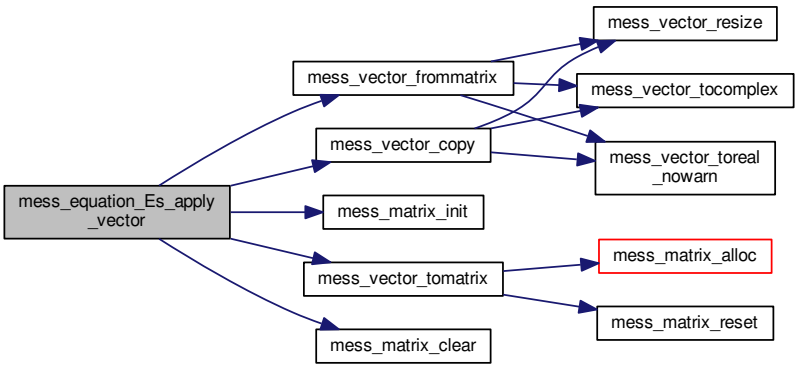
zero on success or a non zero error code

The mess_equation_Es_apply_vector function computes $op(\mathcal{E})x = y$, where x and y are vectors. For matrices use mess_equation_Es_apply. If the object does not provide a AX.apply function, $\mathcal{E} = I$ is assumed.

See also

mess_equation_As_apply
mess_equation_Es_apply

Here is the call graph for this function:



8.42.2.24 int mess_equation_Es_post (mess_equation *eqn*)

Parameters

in	*eqn*	input Equation object $\mathcal{E}x = y$ belongs to

Returns

zero on success or a non zero error code

The mess_equation_Es_post function finalizes the $\mathcal{E}x = y$ operator. That means it calls the clear function of the equation object if it exits otherwise it will return immediately.

See also

mess_equation_Es_pre
mess_equation_Es_apply
mess_equation_Es_apply

8.42.2.25 int mess_equation_Es_pre (mess_equation *eqn*)

Parameters

in	*eqn*	input Equation object $\mathcal{E}$ belongs to

Returns

zero on success or a non zero error code

The mess_equation_Es_pre function initializes the $\mathcal{E}x = y$ operation of a given equation object. It is called normally once before the operator is uses the first time. If the equation object does not provide an initialization functions the mess_equation_Es_pre function returns immediately.

See also

mess_equation_Es_apply
mess_equation_Es_post
mess_equation_Es_apply

8.42.2.26 mess_int_t mess_equation_has_A (mess_equation *eqn*)

Parameters

in	*eqn*	input

Returns

true if the operation is available, false otherwise inlcuding errors.

The mess_equation_has_A function returns true if the operator provides a $x = \mathcal{A}y$ function.

See also

mess_equation_has_As
mess_equation_has_E
mess_equation_has_Es
mess_equation_has_ApE
mess_equation_has_ApEs

8.42.2.27 mess_int_t mess_equation_has_ApE (mess_equation *eqn*)

Parameters

in	*eqn*	input

Returns

true if the operation is available, false otherwise inlcuding errors.

The mess_equation_has_ApE function returns true if the operator provides a $x = (\mathcal{A} + \mathcal{E})y$ function.

See also

mess_equation_has_A
mess_equation_has_As
mess_equation_has_E
mess_equation_has_Es
mess_equation_has_ApEs

8.42.2.28 mess_int_t mess_equation_has_ApEs (mess_equation *eqn*)

Parameters

in	*eqn*	input

Returns

true if the operation is available, false otherwise inlcuding errors.

The mess_equation_has_ApEs function returns true if the operator provides a $(\mathcal{A} + p\mathcal{E})x = y$ function.

See also

mess_equation_has_A
mess_equation_has_As
mess_equation_has_E
mess_equation_has_Es
mess_equation_has_ApE

8.42.2.29 mess_int_t mess_equation_has_As (mess_equation *eqn*)

Parameters

in	*eqn*	input

Returns

true if the operation is available, false otherwise inlcuding errors.

The mess_equation_has_As function returns true if the operator provides a $\mathcal{A}x = y$ function.

See also

mess_equation_has_A
mess_equation_has_E
mess_equation_has_Es
mess_equation_has_ApE
mess_equation_has_ApEs

8.42.2.30 mess_int_t mess_equation_has_E (mess_equation *eqn*)

Parameters

in	*eqn*	input

Returns

true if the operation is available, false otherwise inlcuding errors.

The mess_equation_has_E function returns true if the operator provides a $x = \mathcal{E}y$ function.

See also

mess_equation_has_Es
mess_equation_has_A
mess_equation_has_As
mess_equation_has_ApE
mess_equation_has_ApEs

8.42.2.31 mess_int_t mess_equation_has_Es (mess_equation *eqn*)

Parameters

in	*eqn*	input

Returns

true if the operation is available, false otherwise inlcuding errors.

The mess_equation_has_Es function returns true if the operator provides a $\mathcal{E}x = y$ function.

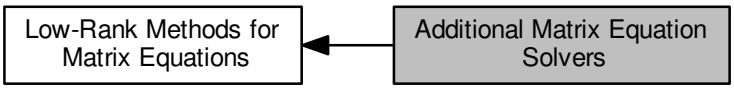
See also

mess_equation_has_E
mess_equation_has_A
mess_equation_has_As
mess_equation_has_ApE
mess_equation_has_ApEs

8.43 Additional Matrix Equation Solvers

This category contains additional matrix equation solvers, which are not based on the ADI process.

Collaboration diagram for Additional Matrix Equation Solvers:



Enumerations

Functions

- int mess_dense_nm_gmare (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X)

  Implementation of the dense Newton-method to solve the generalized Riccati Equation $A^TXE + E^TXA - E^TXGXE + Q = 0$.
- int mess_dense_nm_gpare (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X)

  Implementation of the dense Newton-method to solve the generalized Riccati Equation $A^TXE + E^TXA + E^TXGXE + Q = 0$.
- int mess_dense_nm_gmpare (mess_matrix X0, mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_int_t plus, mess_int_t linesearch, mess_operation_t trans, mess_int_t maxit, mess_norm_t nrm, double absres_tol, double relres_tol, double ∗absres, double ∗relres, mess_matrix RES, mess_matrix X)

  Implementation of the dense Newton method to solve the Riccati Equation $A^TXE + EXA + / - XGX + Q = 0$.
- int mess_dense_res_gmpare (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X, mess_int_t plus, mess_operation_t trans, mess_norm_t nrm_t, double ∗res, double ∗rel)

  Compute the residual of the algebraic (generalized) Riccati (or Lyapunov) equation.
- int mess_lrcf_gsignfac (mess_matrix A, mess_matrix E, mess_matrix B, mess_int_t ∗maxit, double ∗tol, mess_matrix Z, mess_sign_scale_t scale)

  Solve a generalized Lyapunov Equation $A^TXE + E^TXA + B^TB = 0$ *via sign function iteration.*
- int mess_lrcf_signfac (mess_matrix A, mess_matrix B, mess_int_t ∗maxit, double ∗tol, mess_matrix Z, mess_sign_scale_t scale)

  Solve a standard Lyapunov Equation $A^TX + XA + B^TB = 0$ *via sign function iteration.*
- int mess_lrcf_gsignber (mess_matrix A, mess_matrix E, mess_matrix B, mess_int_t ∗maxit, double ∗tol, mess_matrix Z, mess_sign_scale_t scale)

  Solve a generalized Bernoulli equation $A^TXE + E^TXA - E^TXB^TBXE = 0$ *via sign function iteration.*
- int mess_lrcf_signber (mess_matrix A, mess_matrix B, mess_int_t ∗maxit, double ∗tol, mess_matrix Z, mess_sign_scale_t scale)

  Solve a standard Bernoulli equation $A^TX + XA - XB^TBX = 0$ *via sign function iteration.*

8.43.1 Detailed Description

It includes the implementation of the classical Newton iteration to solve a Riccati Equation and the sign function iteration to solve a Lyapunov Equation.

The classical dense **Newton method** solves a (generalized) Riccati Equation

$$A^T XE + E^T XA - E^T XGXE + Q = 0$$

by computing the solution of the resulting internal (generalized) Lyapunov Equation

$$(A - GX_iE)^T X_{i+1}E + E^T X_{i+1}(A - GX_iE) + \left(Q + E^T X_i GX_i E\right) = 0$$

with the aid of a dense Lyapunov solver.
For $E = I$ the generalized Riccati Equation becomes a standard Riccati Equation and hence the generalized Lyapunov Equation becomes a standard Lyapunov Equation.

The **sign function iteration** solves the standard Lyapunov Equation

$$A^T X + XA + B^T B = 0$$

with the aid of the matrix sign function and the Newton iteration.
If $Z \in \mathbb{R}^{n\times n}$ is a matrix with no eigenvalues on the imaginary axis then there exists a nonsingular matrix $S \in \mathbb{R}^{n\times n}$ such that $Z = S^{-1}\begin{bmatrix} J_+ & 0 \\ 0 & J_- \end{bmatrix} S$, where $\lambda(J_+) \subset \mathbb{C}_+, \lambda(J_-) \subset \mathbb{C}_-$.
The the **matrix sign function** is defined by

$$sign(Z) := S^{-1}\begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} S.$$

To compute the matrix sign function, the Newton iteration is applied to $(sign(Z))^2 = I$

$$\begin{array}{rcll} Z_0 & = & Z & , \\ Z_{i+1} & = & \frac{1}{2}\left(Z_i + Z_i^{-1}\right) & . \end{array}$$

This iteration converges globally quadratically to the sign of Z.
To solve the Lyapunov Equation as defined above, this iteration is applied to $Z := \begin{bmatrix} A^T & BB^T \\ 0 & -A \end{bmatrix}$.
The solution X is obtained from

$$sign(Z) = \lim_{i\to\infty} Z_i = \begin{bmatrix} -I_n & 2X \\ 0 & I_n \end{bmatrix}.$$

The same holds for the generalized Lyapunov Equation

$$\hat{A}^T XE + E^T X\hat{A} + \hat{B}^T \hat{B} = 0$$

with $A := \hat{A}E^{-1}, B := \hat{B}E^{-1}$.
source: [http://www2.mpi-magdeburg.mpg.de/mpcsc/benner/pub/BaurB_gamm2004.pdf]

8.43.2 Enumeration Type Documentation

8.43.2.1 enum mess_sign_scale_t

The mess_sign_scale_t enumeration type defines scaling type for speedup sign function iteration.

Enumerator

MESS_SIGN_SCALE_NONE no scaling

MESS_SIGN_SCALE_FRO Frobenius scaling $\sqrt{\frac{||X_k||_F}{||X_k^{-1}||_F}}$ or $\sqrt{\frac{||X_k||_F}{||EX_k^{-1}E||_F}}$

MESS_SIGN_SCALE_DET Determinante scaling $|\det(X_k)|^{\frac{1}{n}}$ or $\left(\frac{|\det(X_k)|}{|\det(E)|}\right)^{\frac{1}{n}}$

8.43.3 Function Documentation

8.43.3.1 int mess_dense_nm_gmare (mess_matrix *A*, mess_matrix *E*, mess_matrix *Q*, mess_matrix *G*, mess_matrix *X*)

Parameters

in	*A*	input matrix A of the Riccati Equation
in	*E*	input matrix E of the Riccati Equation, `NULL` if `E` is identity
in	*Q*	input matrix Q of the Riccati Equation
in	*G*	input matrix G of the Riccati Equation
out	*X*	solution of the Riccati Equation

Returns

zero on success or a non zero error code otherwise

The mess_dense_nm_gmare function solves the generalized algebraic Riccati Equation (GARE)

$$A^T XE + E^T XA - E^T XGXE + Q = 0$$

using the Newton method.
The resulting internal generalized Lyapunov Equation

$$(A - GX_iE)^T X_{i+1}E + E^T X_{i+1}(A - GX_iE) - \left(Q + E^T X_i GX_i E\right) = 0$$

is solved using a dense generalized Lyapunov solver.

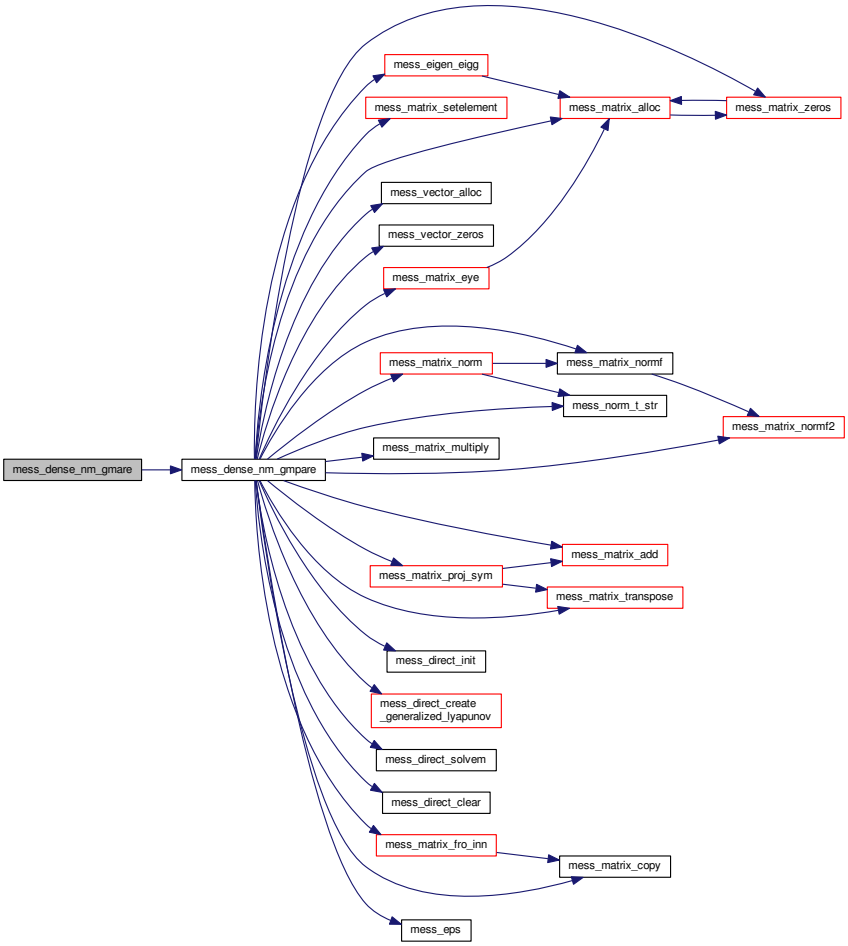
Attention

This function completely relies on dense computations. It returns an error if it is called with sparse matrices.

See also

mess_direct_create_generalized_lyapunov
mess_lrcfadi_nm
mess_dense_nm_gmpare

Here is the call graph for this function:



8.43.3.2 int mess_dense_nm_gmpare (mess_matrix *X0*, mess_matrix *A*, mess_matrix *E*, mess_matrix *Q*, mess_matrix *G*, mess_int_t *plus*, mess_int_t *linesearch*, mess_operation_t *trans*, mess_int_t *maxit*, mess_norm_t *nrm_t*, double *absres_tol*, double *relres_tol*, double * *absres*, double * *relres*, mess_matrix *RES*, mess_matrix *X*)

Parameters

in	*X0*	input starting guess symmetric matrix $X0$, NULL if zero initual guess
in	*A*	input matrix A of the Riccati Equation
in	*E*	input matrix E of the Riccati Equation, NULL if E is identity
in	*Q*	input symmetric matrix Q of the Riccati Equation
in	*G*	input symmetric matrix G of the Riccati Equation
in	*plus*	input if zero then Riccati Equation with $-$ is solved, otherwise $+$

Parameters

in	*linesearch*	input it nonzero linesearch is used
in	*trans*	input parameter to use the dual equation
in	*maxit*	input maximum number of iterations
in	*nrm_t*	input mess_norm_t the desired type of norm
in	*absres_tol*	input tolerance for absolute residual
in	*relres_tol*	input tolerance for relative residual
out	*absres*	output achieved absolute residual, NULL if not wanted
out	*relres*	output achieved relative residual, NULL if not wanted
out	*RES*	output residual matrix, NULL if not wanted
out	*X*	solution of the Riccati Equation

Returns

zero on success or a non zero error code otherwise

The mess_dense_nm_gmpare function solves the algebraic Riccati Equation (ARE) with dense real symmetric positive definite square matrices G and Q and dense real stable square matrix A. If plus is nonzero the positive ARE with $+$ in front of the quadratic term is solved, otherwise the ARE with $-$ is solved. If E points to NULL, then E is assumed to be the identity.

Depending on trans the following AREs are solved:

If trans is MESS_OP_NONE

$$AXE^T + EXA^T + / - EXGXE^T + Q = 0$$

otherwise

$$A^TXE + E^TXA + / - E^TXGXE + Q = 0$$

The function uses a Newton method and the resulting Lyapunov Equations are solved using mess_direct_create↩_generalized_lyapunov.

If X0 does not points to NULL, this matrix is used as an initual guess, otherwise a zero inital guess is used.

The newton iteration solves for the next Newton step, instead of the directly for the next Newton iteration. This is due to the numerical benfits, see [5] and [6].

The polynomial root finding problem is solved via an generalized eigenvalue problem see [18].

Attention

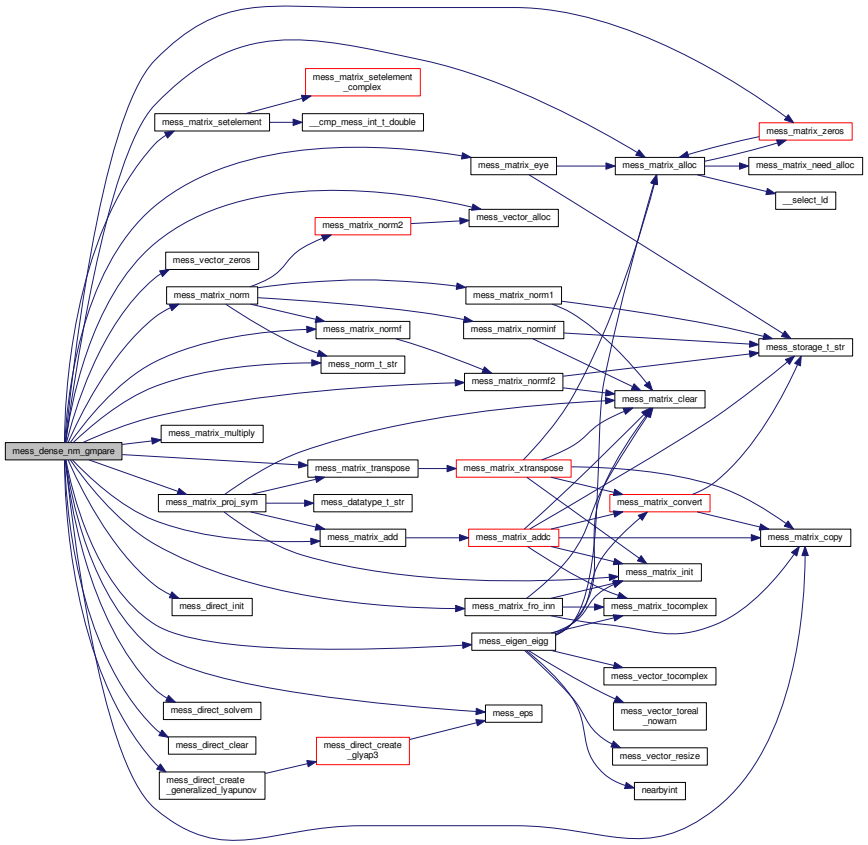
This function completely relies on real and dense computations. It returns an error if it is called with complex or sparse matrices.
The definitenes, symmetry and stability conditions of the matrices are not checked.

See also

mess_direct_create_generalized_lyapunov
mess_lrcfadi_nm

See [5], [6] and [18] for references.

Here is the call graph for this function:



8.43.3.3 int mess_dense_nm_gpare (mess_matrix *A*, mess_matrix *E*, mess_matrix *Q*, mess_matrix *G*, mess_matrix *X*)

Parameters

in	*A*	input matrix A of the Riccati Equation
in	*E*	input matrix E of the Riccati Equation, NULL if E is identity
in	*Q*	input matrix Q of the Riccati Equation
in	*G*	input matrix G of the Riccati Equation
out	*X*	solution of the Riccati Equation

Returns

zero on success or a non zero error code otherwise

The mess_dense_nm_gpare function solves the generalized algebraic Riccati Equation (GARE)

$$A^T XE + E^T XA + E^T XGXE + Q = 0$$

using the Newton method.
The resulting internal generalized Lyapunov Equation

$$(A - GX_iE)^T X_{i+1}E + E^T X_{i+1}(A - GX_iE) + \left(Q + E^T X_i GX_i E\right) = 0$$

is solved using a dense generalized Lyapunov solver.

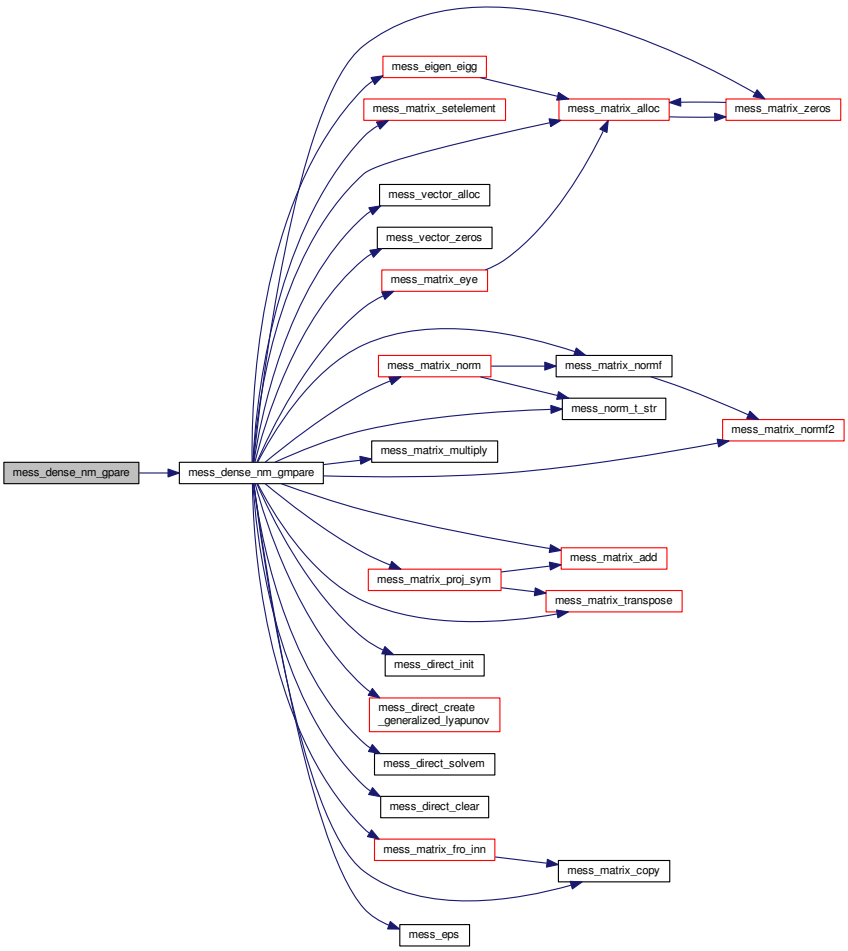
Attention

This function completely relies on dense computations. It returns an error if it is called with sparse matrices.

See also

mess_direct_create_generalized_lyapunov
mess_lrcfadi_nm
mess_dense_nm_gmpare

Here is the call graph for this function:



8.43.3.4 int mess_dense_res_gmpare (mess_matrix *A*, mess_matrix *E*, mess_matrix *Q*, mess_matrix *G*, mess_matrix *X*, mess_int_t *plus*, mess_operation_t *trans*, mess_norm_t *nrm_t*, double * *res*, double * *rel*)

Parameters

in	*A*	input matrix
in	*E*	input matrix E of the Riccati Equation, NULL if E is identity
in	*Q*	input symmetric matrix
in	*G*	input symmetric matrix (NULL if G is zero -> Lyapunov Equation)
in	*X*	input symmetric solution of the Riccati Equation
in	*plus*	input if zero then Riccati Equation with $-$ is considered, otherwise $+$
in	*trans*	input parameter to use the dual equation
in	*nrm↩_t*	input parameter to control desired norm type
out	*res*	output asbolute residual
out	*rel*	output relative residual (absolute residual divided by norm of Q)

Returns

zero on success or a non zero error code otherwise

The mess_dense_res_gmpare function computes residual of dense algebraic Riccati Equation (ARE).

If plus is nonzero the positive ARE with $+$ in front of the quadratic term is considered, otherwise the ARE with $-$ is considered.

If E points to NULL, then E is assumed to be the identity.

Depending on trans the following reisduals are computed

If trans is MESS_OP_NONE

$$\|AXE^T + EXA^T + / - EXGXE^T + Q\|$$

otherwise

$$\|A^TXE + E^TXA + / - E^TXGXE + Q\|$$

If G is NULL the residual of a Lyapunov Equation is computed.

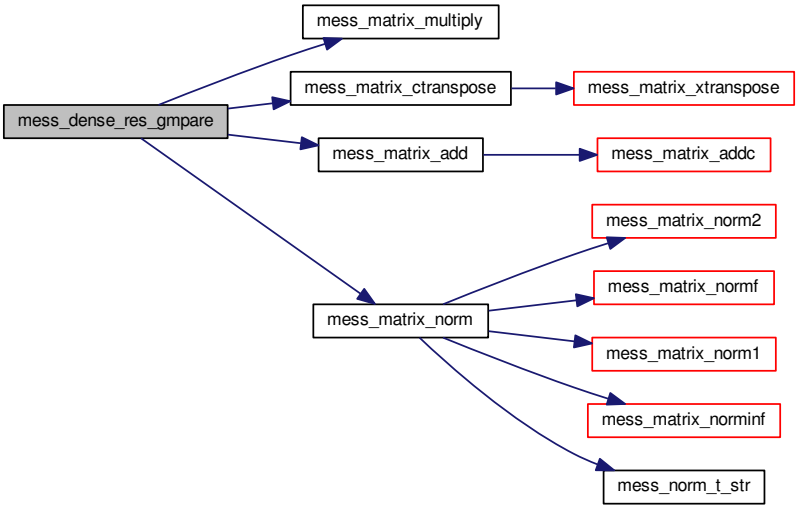
nrm_t is used to define the desired norm.

rel gives the relative residual

$$rel = res/\|Q\|$$

, if not wanted call the function with rel equals NULL.

Here is the call graph for this function:



8.43.3.5 int mess_lrcf_gsignber (mess_matrix *A*, mess_matrix *E*, mess_matrix *B*, mess_int_t * *maxit*, double * *tol*, mess_matrix *Z*, mess_sign_scale_t *scale*)

Parameters

in	*A*	$(n \times n)$-matrix
in	*E*	$(n \times n)$-matrix
in	*B*	$(p \times n)$-matrix
in,out	*maxit*	on input: maximum number of iterations on output: number of performed iterations
in,out	*tol*	on input: convergence tolerance on output: achieved tolerance
in	*Z*	numerical full rank factor Z of the solution $X \approx ZZ^T$
in	*scale*	type of scaling for speedup

Returns

zero on success or a non zero error value otherwise

The mess_lrcf_gsignber function solves a generalized algebraic bernoulli equation

$$A^T XE + E^T XA - E^T XB^T BXE = 0$$

via the sign function iteration ([4]) . The solution is computed as a low rank factor Z of the solution $X \approx ZZ^T$.

The algorithm stops if the maximum iteration number $maxit$ is reached or relative Error in Frobenius-norm of the is smaller than tol

$$\|A_{k+1} - A_k\|_F < tol\|A_{k+1}\|_F.$$

At exit $maxit$ contains the number of performed iterations and tol the achieved tolerance.

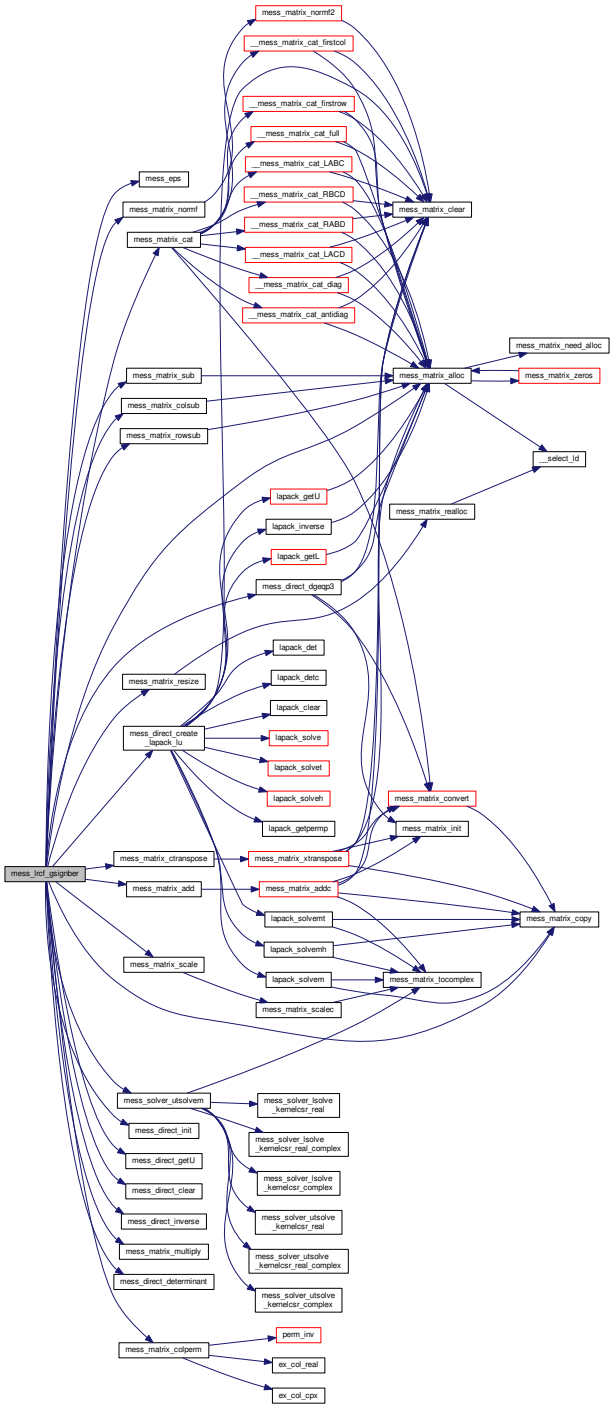
Attention

The function works in dense arithmetics and does not support sparse matrices at all.

See also

mess_lrcf_signlyap
mess_lrcf_gsignlyap
mess_lrcf_signber
mess_sign_scale_t

Here is the call graph for this function:



8.43.3.6 int mess_lrcf_gsignfac (mess_matrix *A*, mess_matrix *E*, mess_matrix *B*, mess_int_t * *maxit*, double * *tol*, mess_matrix *Z*, mess_sign_scale_t *scale*)

Parameters

in	*A*	input $(n \times n)$-matrix
in	*E*	input $(n \times n)$-matrix
in	*B*	input $(p \times n)$-matrix
in,out	*maxit*	on input: maximum number of iterations on output: number of performed iterations
in,out	*tol*	on input: convergence tolerance on output: achieved tolerance
out	*Z*	numerical full rank factor Z of the solution $X \approx ZZ^T$
out	*scale*	type of scaling for speedup

Returns

zero on success or a non-zero error value otherwise

The mess_lrcf_gsignfac function solves a generalized Lyapunov Equation

$$A^T XE + E^T XA + B^T B = 0$$

via the sign function iteration ([11]) . The solution is computed as a low rank factor Z of the solution $X \approx ZZ^T$.

The algorithm stops if the maximum iteration number $maxit$ is reached or if the Frobenius-norm of the current iterate is smaller than tol

$$\|A_i + E\|_F < tol.$$

At exit $maxit$ contains the number of performed iterations and tol the achieved tolerance.

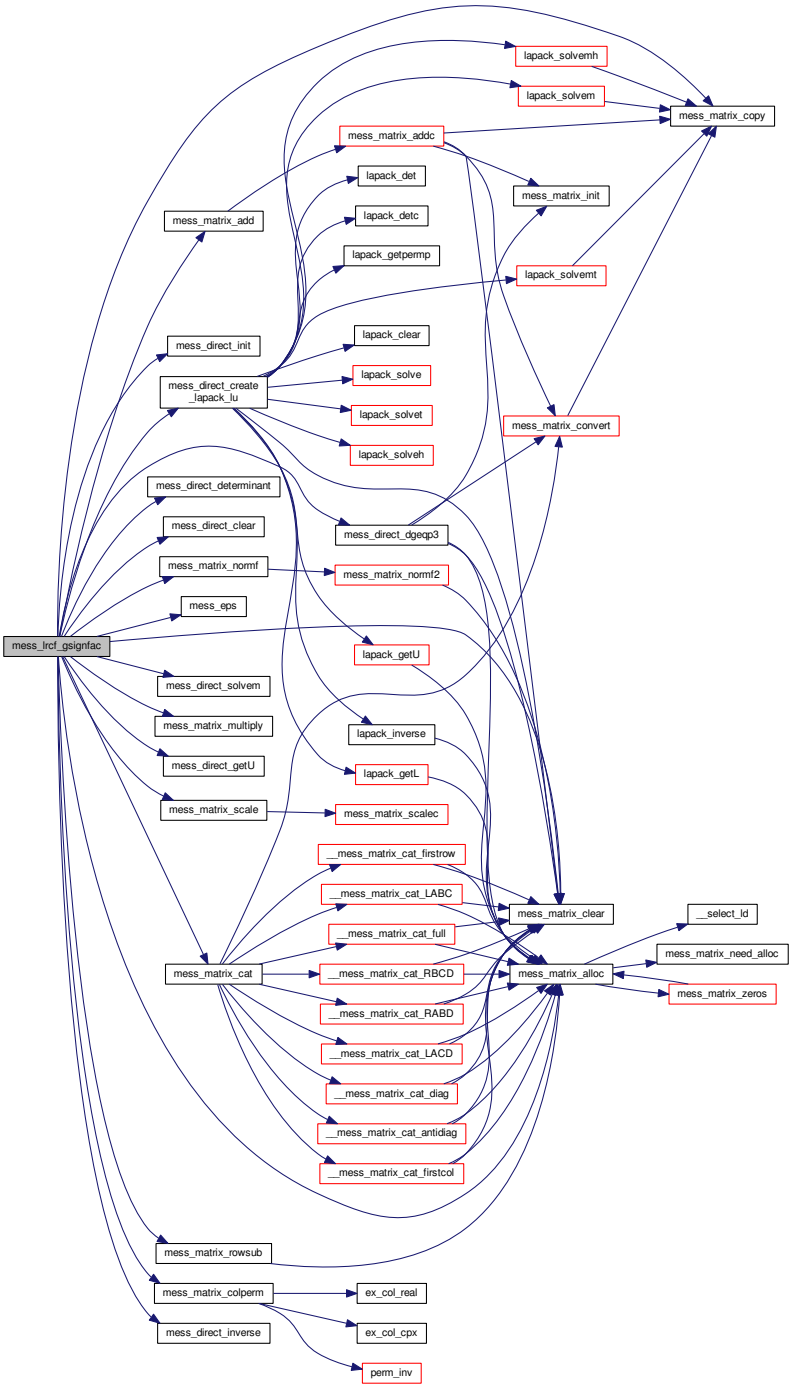
Attention

The function works in dense arithmetics and does not support sparse matrices at all.

See also

mess_lrcf_signfac
mess_lrcf_adi
mess_sign_scale_t

Here is the call graph for this function:



8.43.3.7 int mess_lrcf_signber (mess_matrix *A*, mess_matrix *B*, mess_int_t * *maxit*, double * *tol*, mess_matrix *Z*, mess_sign_scale_t *scale*)

Parameters

in	*A*	$(n \times n)$-matrix
in	*B*	$(p \times n)$-matrix
in,out	*maxit*	on input: maximum number of iterations on output: number of performed iterations
in,out	*tol*	on input: convergence tolerance on output: achieved tolerance
in	*Z*	numerical full rank factor Z of the solution $X \approx ZZ^T$
in	*scale*	type of scaling for speedup

Returns

zero on success or a non zero error value otherwise

The mess_lrcf_signber function solves a standard algebraic bernoulli equation

$$A^T X + XA + XB^T BX = 0$$

via the sign function iteration ([4]) . The solution is computed as a low rank factor Z of the solution $X \approx ZZ^T$.

The algorithm stops if the maximum iteration number $maxit$ is reached or relative Error in Frobenius-norm of the is smaller than tol

$$\|A_{k+1} - A_k\|_F < tol\|A_{k+1}\|_F.$$

At exit $maxit$ contains the number of performed iterations and tol the achieved tolerance.

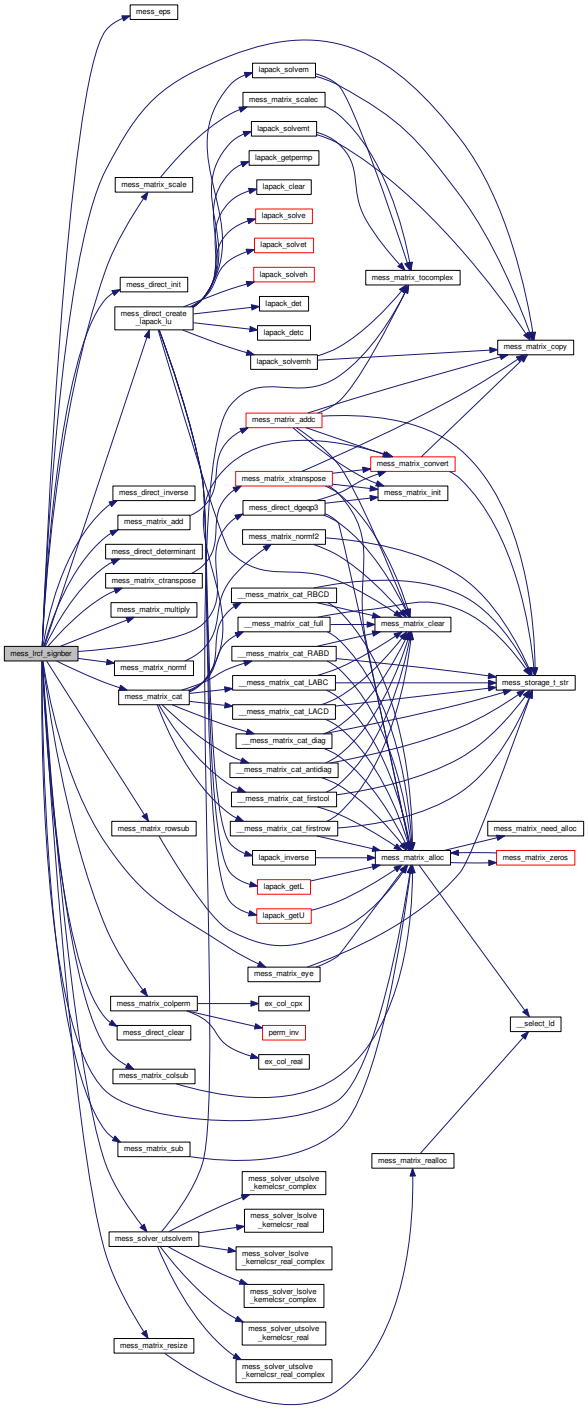
Attention

The function works in dense arithmetics and does not support sparse matrices at all.

See also

mess_lrcf_gsignlyap
mess_lrcf_signlyap
mess_lrcf_gsignber
mess_sign_scale_t

Here is the call graph for this function:



8.43.3.8 int mess_lrcf_signfac (mess_matrix *A*, mess_matrix *B*, mess_int_t * *maxit*, double * *tol*, mess_matrix *Z*, mess_sign_scale_t *scale*)

Parameters

in	*A*	input $(n \times n)$-matrix
in	*B*	input $(p \times n)$-matrix
in,out	*maxit*	on input: maximum number of iterations on output: number of performed iterations
in,out	*tol*	on input: convergence tolerance on output: achieved tolerance
in	*Z*	input numerical full rank factor Z of the solution $X \approx ZZ^T$
in	*scale*	input type of scaling for speedup

Returns

zero on success or a non-zero error value otherwise

The mess_lrcf_signfac function solves a standard Lyapunov Equation

$$A^T X + XA + B^T B = 0$$

via the sign function iteration ([11]) . The solution is computed as a low rank factor Z of the solution $X \approx ZZ^T$.

The algorithm stops if the maximum iteration number $maxit$ is reached or Frobenius-norm of the current iterate is smaller than tol

$$\|A_i + I\|_F < tol.$$

At exit $maxit$ contains the number of performed iterations and tol the achieved tolerance.

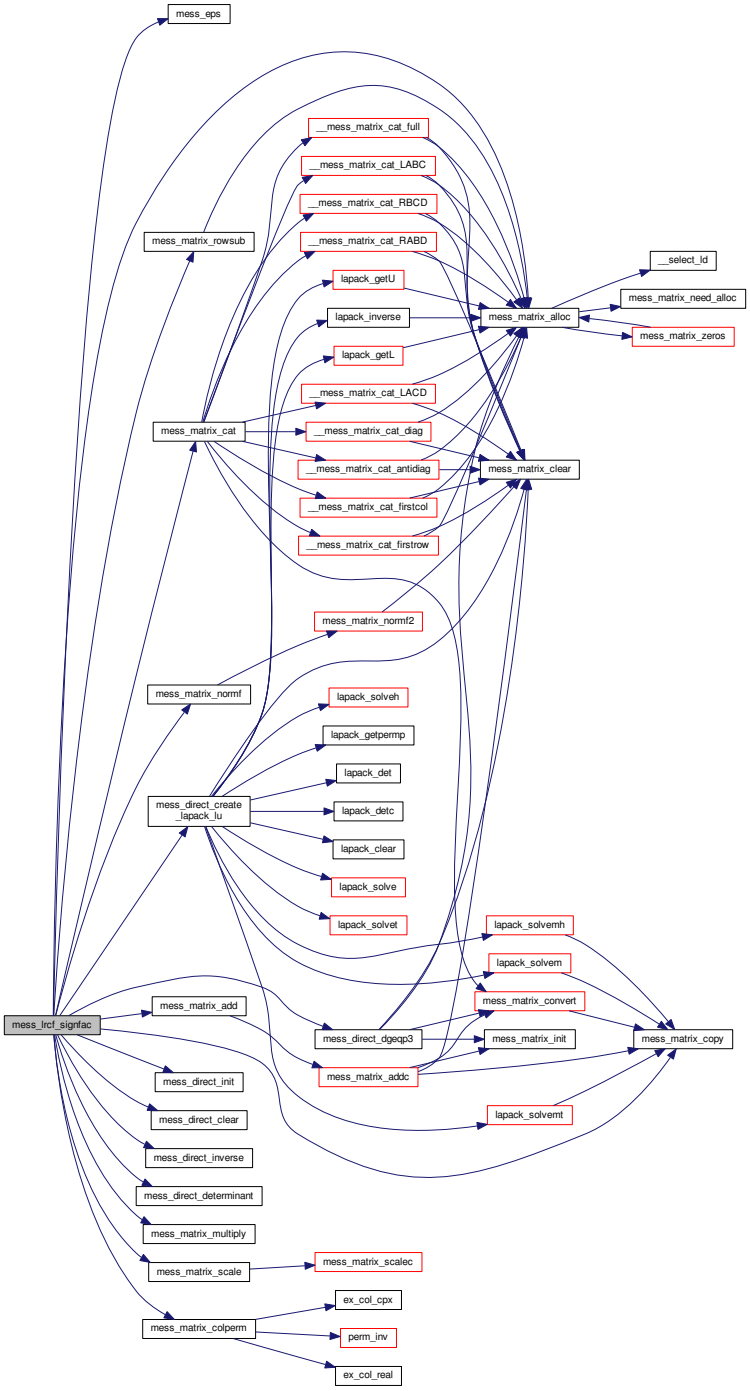
Attention

The function works in dense arithmetics and does not support sparse matrices at all.

See also

mess_lrcf_gsignfac
mess_lrcf_adi
mess_sign_scale_t

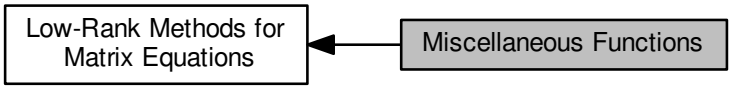
Here is the call graph for this function:



8.44 Miscellaneous Functions

This category contains several function that do not fit in any other category.

Collaboration diagram for Miscellaneous Functions:



Functions

- int mess_lrcfadi_lp_s (mess_double_cpx_t *p, mess_int_t lp, mess_double_cpx_t *set, mess_int_t lset, double *max_r, mess_int_t *ind)

  Compute the maximum magnitude of the rational ADI function.
- int mess_lrcfadi_lp_mnmx (mess_double_cpx_t *rw, mess_int_t lrw, mess_int_t l0, mess_double_cpx_t *p, mess_int_t *lp)

  Compute ADI parameters solving the ADI minimax problem suboptimal.
- int mess_lrcfadi_fac2diff (mess_matrix Zold, mess_matrix Z, double *chg)

  Compute 2-norm difference between two factored solutions of a matrix equation.
- int mess_lrcfadi_facFdiff (mess_matrix Zold, mess_matrix Z, double *chg)

  Compute the relative Frobenius-norm difference between two factored solutions of a matrix equation.
- int mess_lrcfadi_para_wachspress (double a, double b, double alpha, double tol, double *p, mess_int_t *lp)

  Compute Wachspress parameters.
- int mess_lrcfadi_ccsvd (mess_matrix Z, double ccTol)

  Perform a SVD based column compression on low rank factor.
- int mess_lrcfadi_colcompress (mess_matrix Z, double ccTol)

  Perform a column compression on a low rank factor.
- int mess_lrcfadi_res2 (mess_matrix F, mess_matrix G, mess_matrix Z, double *nrm)

  Compute 2-norm of the Lyapunov residual via an Arnoldi process.
- int mess_lrcfadi_res2BK (mess_matrix F, mess_matrix G, mess_matrix Z, mess_matrix B, mess_matrix K, double *nrm)

  Compute the 2-norm of the Lyapunov residual (BK version).
- int mess_lrcfadi_res2g (mess_matrix F, mess_matrix M, mess_matrix G, mess_matrix Z, double *nrm)

  Compute the 2-norm of the generalized Lyapunov residual.
- int mess_lrcfadi_res2gBK (mess_matrix F, mess_matrix M, mess_matrix G, mess_matrix Z, mess_matrix B, mess_matrix K, double *nrm)

  Compute the 2-norm of the generalized Lyapunov residual (BK-Version).
- int mess_lrcfadi_res2t (mess_matrix F, mess_matrix G, mess_matrix Z, double *nrm)

  Compute 2 norm of the Lyapunov residual (transposed version).
- int mess_lrcfadi_res2gt (mess_matrix F, mess_matrix M, mess_matrix G, mess_matrix Z, double *nrm)

  Compute the 2-norm of the generalized Lyapunov residual (transposed version).
- int mess_lrcfadi_res2g_so (mess_matrix M, mess_matrix G, mess_matrix K, mess_matrix RHS, mess_matrix Z, double *nrm)

- int mess_lrcfadi_res2gt_so (mess_matrix M, mess_matrix G, mess_matrix K, mess_matrix RHS, mess_matrix Z, double *nrm)
- int mess_lrcfadi_res2nm (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix Z, double *nrm)

  Compute the 2-norm of the Riccati residual.
- int mess_lrcfadi_res2nmg (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix M, mess_matrix Z, double *nrm)

  Compute the 2-norm of the generalized Riccati residual.

8.44.1 Detailed Description

It includes functions to

- compute shift parameter solving the ADI min max problem suboptimal
- compute the 2-norm difference of an old and a new solution factor
- compute the maximal magnitude of the rational ADI function over a discrete subset of the left complex half-plane
- perform a column compression on the low rank factor.

8.44.2 Function Documentation

8.44.2.1 int mess_lrcfadi_ccsvd (mess_matrix *Z*, double *ccTol*)

Parameters

in,out	*Z*	input/output current ADI factor
in	*ccTol*	input cut-off tolerance for the singular values

Returns

zero on success or a non-zero error value otherwise

The mess_lrcfadi_ccsvd function performs a column compression on a low rank factor. It is the current backend behind mess_lrcfadi_colcompress.

It works using a SVD on the input matrix Z and decide uppon the rank of the matrix using the cut-off tolerance `ccTol`. All singular values that are smaller than

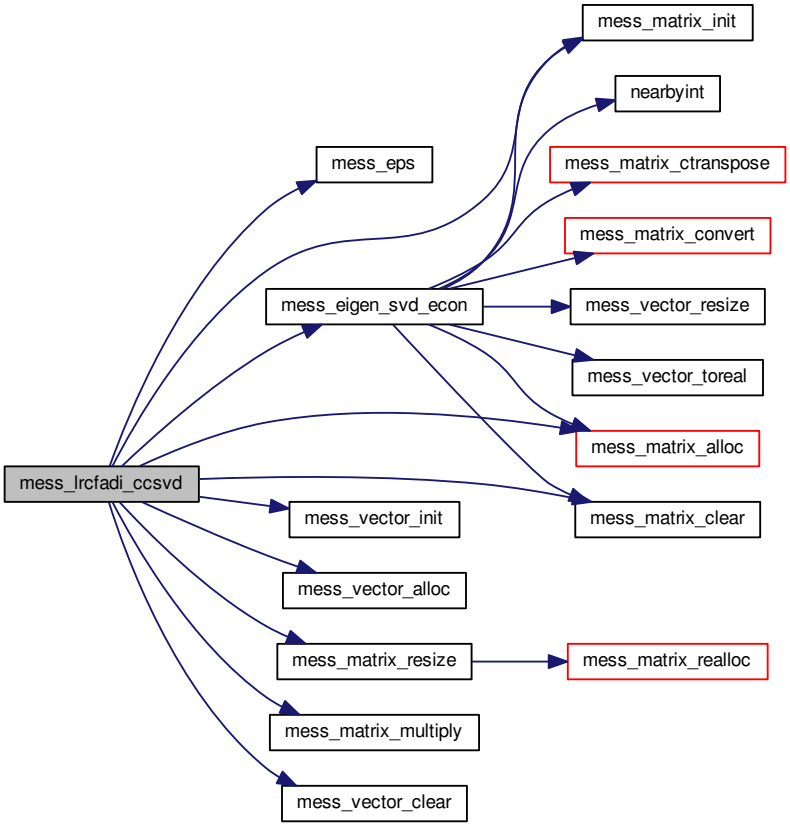
$$MAX(rows(Z), cols(Z)) \cdot ccTol \cdot \sigma_1$$

are neglected, where σ_1 is the largest singular value.

See also

mess_lrcfadi_colcompress

Here is the call graph for this function:



8.44.2.2 int mess_lrcfadi_colcompress (mess_matrix *Z*, double *ccTol*)

Parameters

in,out	*Z*	current low rank factor
in	*ccTol*	input cut-off tolerance in the rank decission

Returns

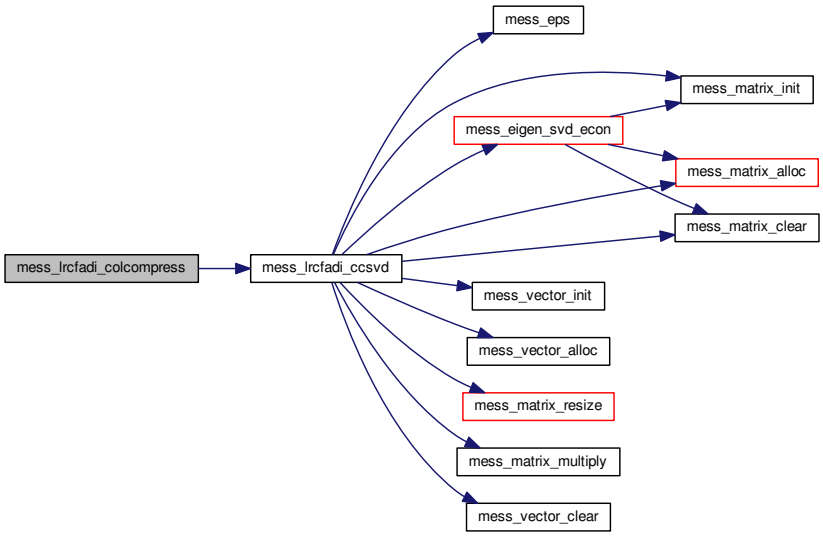
zero on success or a non-zero error value otherwise

The mess_lrcfadi_colcompress function performs a column compression on a low rank factor iterate. The detailed description can be found in [28] . This function is only a wrapper around mess_lrcfadi_ccsvd in order to change its implementation in future times.

See also

mess_lrcfadi_ccsvd

Here is the call graph for this function:



8.44.2.3 int mess_lrcfadi_fac2diff (mess_matrix *Zold*, mess_matrix *Z*, double * *chg*)

Parameters

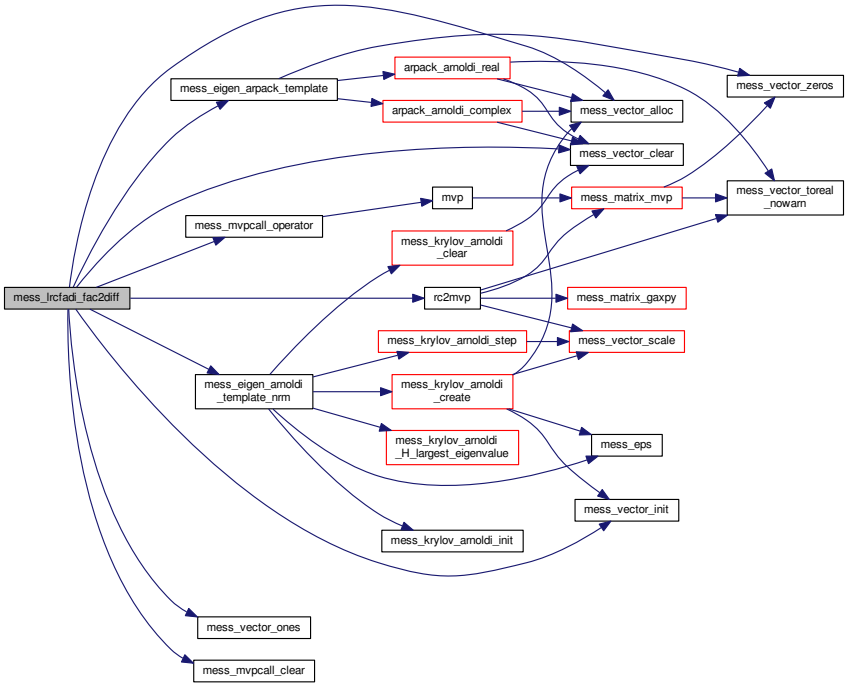
in	*Zold*	input first factor
in	*Z*	input input second factor
out	*chg*	2 norm difference

Returns

zero on success or a non-zero error value otherwise

The mess_lrcfadi_fac2diff function computes the 2-norm difference of $|\underbrace{Z_{old}Z_{old}^T}_{X_{old}} - \underbrace{ZZ^T}_{X}|_2$ without forming X_{old} and X.

Here is the call graph for this function:



8.44.2.4 int mess_lrcfadi_facFdiff (mess_matrix *Zold*, mess_matrix *Z*, double * *chg*)

Parameters

in	*Zold*	input old Z factor
in	*Z*	input new Z factor
out	*chg*	relative Frobenius-norm difference

Returns

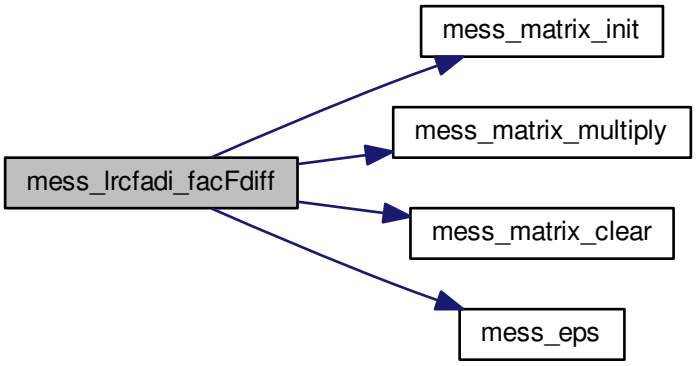
zero on success or a non-zero error value otherwise

The mess_lrcfadi_facFdiff function computes the relative Frobenius-norm difference

$$chg = \| \underbrace{Z_{old} Z_{old}^T}_{X_{old}} - \underbrace{Z Z^T}_{X} \|_F \cdot \|X\|_{F^{-1}}$$

without forming X_{old} and X.

Here is the call graph for this function:



8.44.2.5 int mess_lrcfadi_lp_mnmx (mess_double_cpx_t * *rw*, mess_int_t *lrw*, mess_int_t *adi_shifts_l0*, mess_double_cpx_t * *p*, mess_int_t * *lp*)

Parameters

in	*rw*	input vector containing numbers in the open left half plane, which approximate the spectrum of the corresponding matrix, e.g. a set of Ritz values (The set must be closed with respect to complex conjugation.)
in	*lrw*	input length of the rw vector
in,out	*adi_shifts↩ _l0*	desired number of shift parameters (length $(rw) >= adi_shifts_l0$) (The algorithm delivers either adi_shifts_l0 or $adi_shifts_l0 + 1$ parameters!)
out	*p*	a adi_shifts_l0- or $adi_shifts_l0 + 1$-vector of suboptimal ADI parameters
out	*lp*	length of p

Returns

zero on success or a non-zero error value otherwise

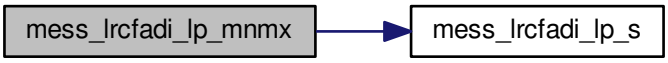
The mess_lrcfadi_lp_mnmx function computes ADI parameters p by solving the rational min-max problem

$$\min_{p_j \in \mathbb{R}, j=1,\cdots,lrw} \max_{\lambda \in rw} \left| \prod_{j=1}^{lrw} \frac{p_j - \lambda}{p_j + \lambda} \right|$$

suboptimal.
The delivered parameter set is closed under complex conjugation.

Details can be found in [25] or [28] .

Here is the call graph for this function:



8.44.2.6 int mess_lrcfadi_lp_s (mess_double_cpx_t * *p*, mess_int_t *lp*, mess_double_cpx_t * *set*, mess_int_t *lset*, double * *max_r*, mess_int_t * *ind*)

Parameters

in	*p*	input vector of ADI parameters
in	*lp*	input length of the p vector
in	*set*	input vector representing the discrete set
in	*lset*	input length of the set vector
out	*max↩ _r*	maximal magnitude of the rational ADI function over the set
out	*ind*	index - maximum is attained for $set(ind)$

Returns

zero on success or a non-zero error value otherwise

The mess_lrcfadi_lp_s function computes the maximum magnitude of the rational ADI function over a discrete subset of the left complex half plane.
The function is only an auxilary routine for mess_lrcfadi_lp_mnmx.

Details can be found in [25] or [28] .

If a `NULL-pointer` is given MESS_ERROR_NULLPOINTER is returned.

See also

mess_lrcfadi_lp_mnmx

8.44.2.7 int mess_lrcfadi_para_wachspress (double *a*, double *b*, double *alpha*, double *tol*, double * *p*, mess_int_t * *lp*)

Parameters

in	*a*	input lower spectral bound
in	*b*	input upper spectral bound
in	*alpha*	input sector angle
in	*tol*	input tolerance
out	*p*	vector containing Wachspress parameters
out	*lp*	length of vector p

Returns

zero on success or a non-zero error value otherwise

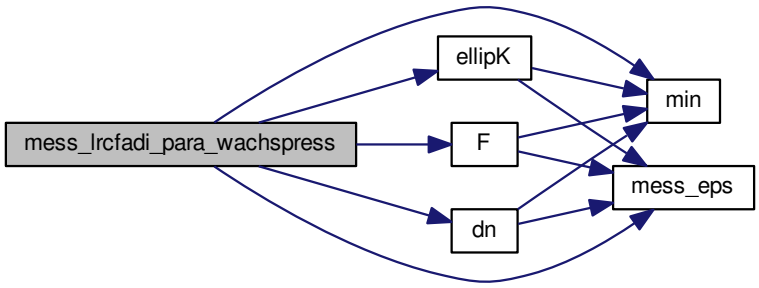
The mess_lrcfadi_para_wachspress function computes optimal ADI-shift parameters according to the formulas given in [29] for matrices with real spectrum.
The spectral bounds a, b and the sector angle $alpha$ are given as

$$\begin{aligned} a &= \min_{i=1,\cdots,n} \Big(\mathrm{Re}(\lambda_i)\Big), \\ b &= \max_{i=1,\cdots,n} \Big(\mathrm{Re}(\lambda_i)\Big), \\ \alpha &= \tan^{-1} \max_{i=1,\cdots,n} \left|\frac{\mathrm{Im}(\lambda_i)}{\mathrm{Re}(\lambda_i)}\right|, \end{aligned}$$

where $\lambda_1, \cdots, \lambda_n$ are eigenvalues of the negative matrix.
Elliptic functions and integrals are computed to a tolerance tol.

More details can be found in [29].

Here is the call graph for this function:



8.44.2.8 int mess_lrcfadi_res2 (mess_matrix *F*, mess_matrix *G*, mess_matrix *Z*, double * *nrm*)

Parameters

in	*F*	input system matrix
in	*G*	input right hand side
in	*Z*	input computed Z
out	*nrm*	output computed 2-norm of the residual

Returns

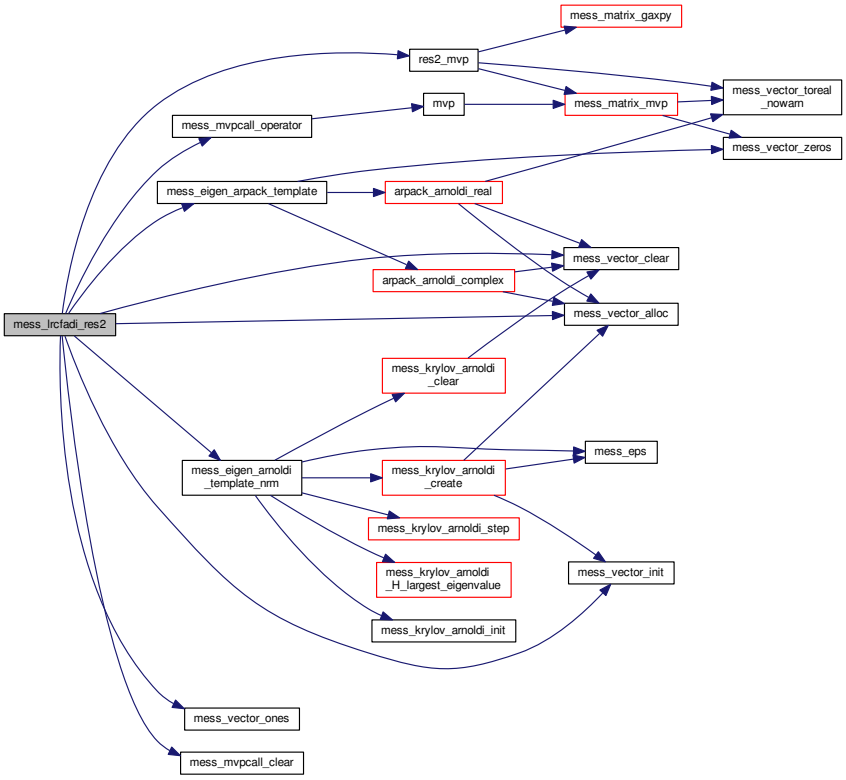
zero on succes or a non zero error value

The mess_lrcfadi_res2 function computes the 2-norm of the Lyapunov residual of Z exploiting the symmetry of the residual operator

$$R := FZZ^T + ZZ^TF^T + GG^T.$$

That means it computes the spectral radius of this operator by a power iteration, exploiting the sparsity of F and the rectangular structure of Z and G.
Thus it can be computed in O(n) effort and is therefore much cheaper than the computation of the e.g. the Frobenius-norm.
The default maximum number of iterations is 20.
The default tolerance is set to $\sqrt{u}$.

Here is the call graph for this function:



8.44.2.9 int mess_lrcfadi_res2BK (mess_matrix *F*, mess_matrix *G*, mess_matrix *Z*, mess_matrix *B*, mess_matrix *K*, double $*$ *nrm*)

Parameters

in	*F*	input system matrix
in	*G*	input right hand side
in	*B*	input martix for $(F - BK)$
in	*K*	input matrix for $(F - BK)$
in	*Z*	input computed Z
out	*nrm*	output computed 2-norm of the residual

Returns

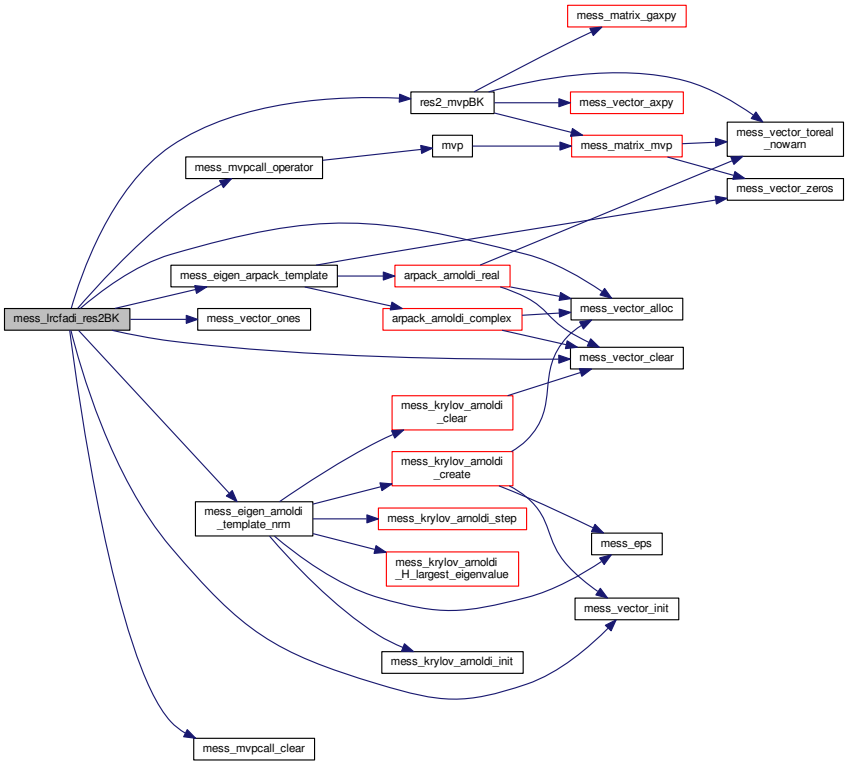
zero on success or a non-zero error value otherwise

The mess_lrcfadi_res2BK function computes the 2-norm of the Lyapunov residual of Z exploiting the symmetry of the residual operator

$$R := (F - BK)ZZ^T + ZZ^T(F - BK)^T + GG^T.$$

That means it computes the spectral radius of this operator by a power iteration, exploiting the sparsity of F and rectangular structure of Z and G.
Thus it can be computed in O(n) effort and is therefore much cheaper than the computation of the e.g. the Frobenius-norm.
The default maximum number of iterations is 20.
The default tolerance is set to $\sqrt{u}$.

Here is the call graph for this function:



8.44.2.10 int mess_lrcfadi_res2g (mess_matrix *F*, mess_matrix *M*, mess_matrix *G*, mess_matrix *Z*, double * *nrm*)

Parameters

in	*F*	input system matrix
in	*G*	input right hand side
in	*M*	input mass matrix
in	*Z*	input computed Z
out	*nrm*	output computed 2-norm of the residual

Returns

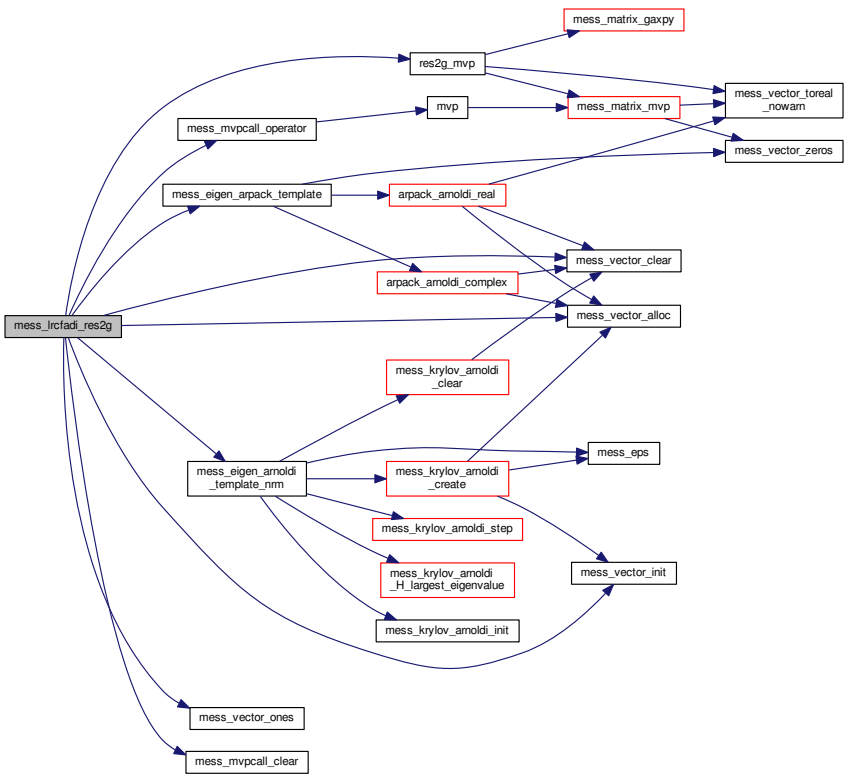
zero on success or a non-zero error value otherwise

The mess_lrcfadi_res2g function computes the 2-norm of the generalized Lyapunov residual of Z exploiting the symmetry of the residual operator

$$R := FZZ^TM^T + MZZ^TF^T + GG^T.$$

That means it computes the spectral radius of this operator by a power iteration, exploiting the sparsity of F and rectangular structure of Z and G.
Thus it can be computed in O(n) effort and is therefore much cheaper than the computation of the e.g. the Frobenius-norm.
The default maximum number of iterations is 20.
The default tolerance is set to $\sqrt{u}$.

Here is the call graph for this function:



8.44.2.11 int mess_lrcfadi_res2g_so (mess_matrix *M*, mess_matrix *G*, mess_matrix *K*, mess_matrix *RHS*, mess_matrix *Z*, double ∗ *nrm*)

8.44.2.12 int mess_lrcfadi_res2gBK (mess_matrix *F*, mess_matrix *M*, mess_matrix *G*, mess_matrix *Z*, mess_matrix *B*, mess_matrix *K*, double ∗ *nrm*)

Parameters

in	*F*	input system matrix
in	*G*	input right hand side

Parameters

in	*M*	input mass matrix
in	*B*	input matrix for $(F - BK)$
in	*K*	input matrix for $(F - BK)$
in	*Z*	input computed Z
out	*nrm*	output computed 2-norm of the residual

Returns

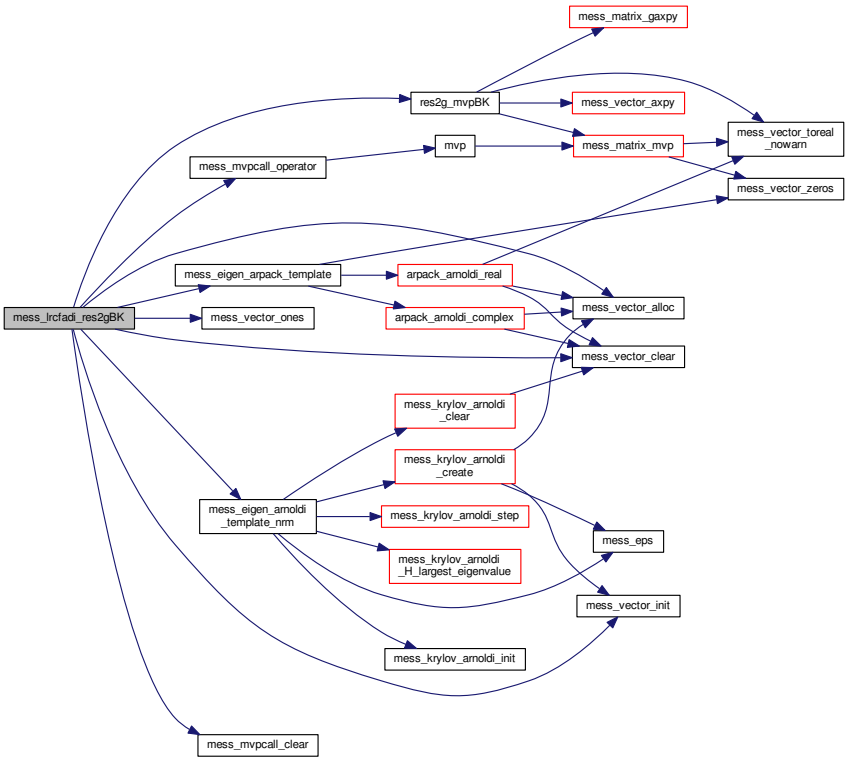
zero on success or a non-zero error value otherwise

The mess_lrcfadi_res2gBK function computes the 2-norm of the generalized Lyapunov residual of Z exploiting the symmetry of the residual operator

$$R := FZZ^TM + ZZ^TF^TM + GG^T.$$

That means it computes the spectral radius of this operator by a power iteration, exploiting the sparsity of F and rectangular structure of Z and G.
Thus it can be computed in O(n) effort and is therefore much cheaper than the computation of the e.g. the Frobenius-norm.
The default maximum number of iterations is 20.
The default tolerance is set to $\sqrt{u}$.

Here is the call graph for this function:



8.44.2.13 int mess_lrcfadi_res2gt (mess_matrix *F*, mess_matrix *M*, mess_matrix *G*, mess_matrix *Z*, double * *nrm*)

Parameters

in	*F*	input system matrix
in	*G*	input right hand side
in	*M*	input mass matrix
in	*Z*	input computed Z
out	*nrm*	output computed 2-norm of the residual

Returns

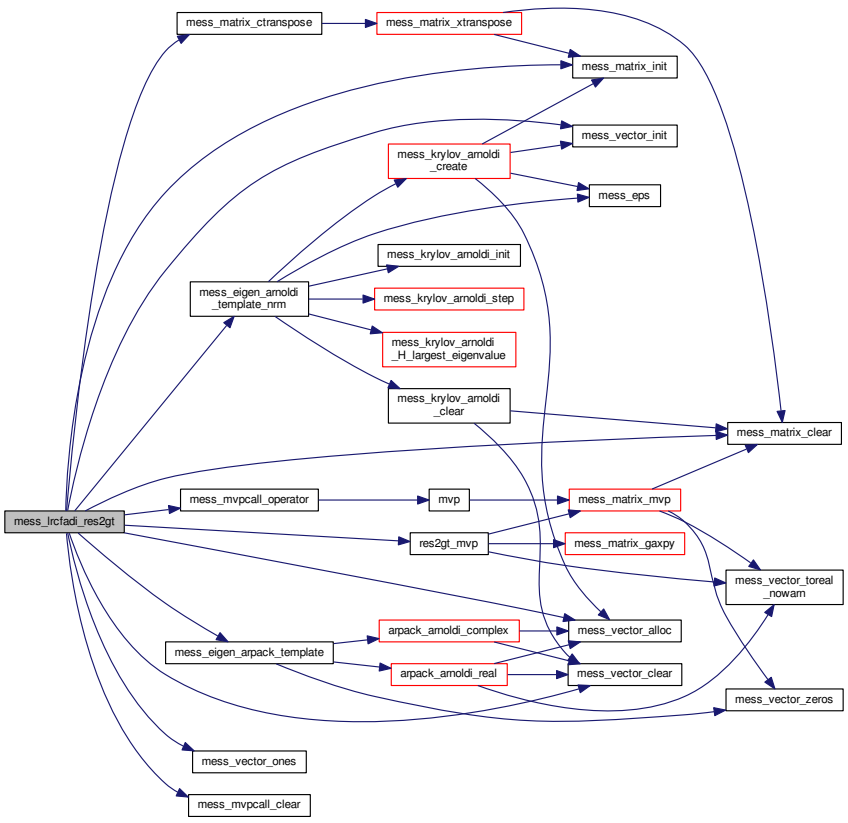
zero on success or a non-zero error value otherwise

The mess_lrcfadi_res2gt function computes the 2 Norm of the generalized Lyapunov residual of Z exploiting the symmetry of the residual operator

$$R := F^T Z Z^T M + M' Z Z^T F + G G^T.$$

That means it computes the spectral radius of this operator by a power iteration, exploiting the sparsity of F and rectangular structure of Z and G.
Thus it can be computed in O(n) effort and is therefore much cheaper than the computation of the e.g. the Frobenius-norm.
The default maximum number of iterations is 20.
The default tolerance is set to $\sqrt{u}$.

Here is the call graph for this function:



8.44.2.14 int mess_lrcfadi_res2gt_so (mess_matrix *M*, mess_matrix *G*, mess_matrix *K*, mess_matrix *RHS*, mess_matrix *Z*, double ∗ *nrm*)

8.44.2.15 int mess_lrcfadi_res2nm (mess_matrix *A*, mess_matrix *B*, mess_matrix *C*, mess_matrix *Z*, double ∗ *nrm*)

Parameters

in	*A*	input matrix
in	*B*	input matrix
in	*C*	input matrix
in	*Z*	input computed Z
out	*nrm*	output computed 2-norm of the residual

Returns

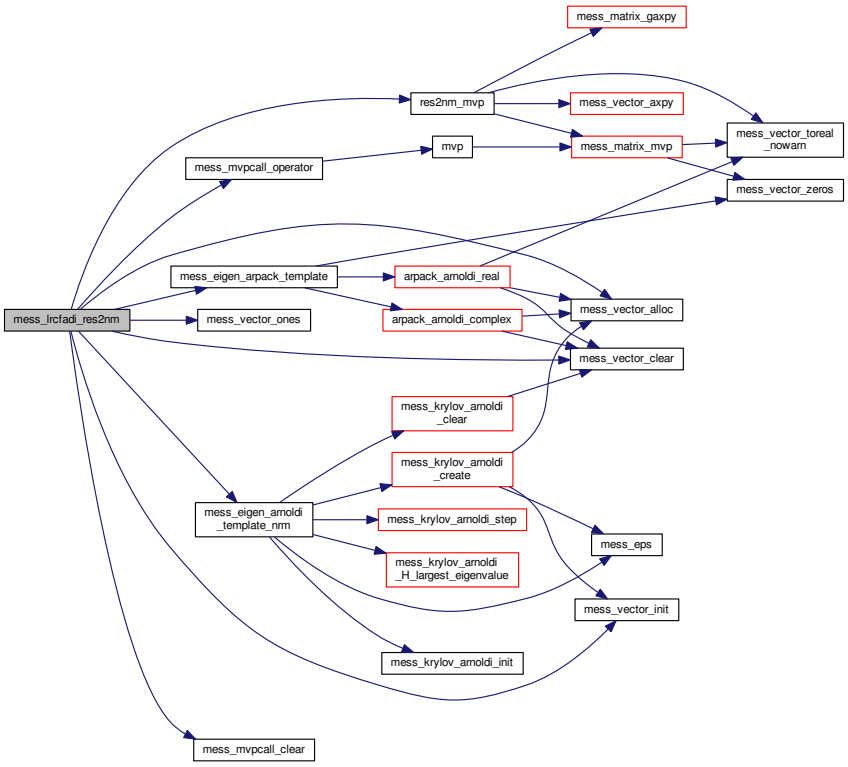
zero on success or a non-zero error value otherwise

The mess_lrcfadi_res2nm function computes the 2-norm of the Riccati residual of Z exploiting the symmetry of the residual operator

$$R := A^T Z Z^T + Z Z^T A + C^T C - Z Z^T B B^T Z Z^T.$$

That means it computes the spectral radius of this operator by a power iteration, exploitig the sparsity of A and rectangular structure of B, C and Z.
Thus it can be computed in O(n) effort and is therefore much cheaper than the computation of the e.g. the Frobenius-norm.

Here is the call graph for this function:



8.44.2.16 int mess_lrcfadi_res2nmg (mess_matrix *A*, mess_matrix *B*, mess_matrix *C*, mess_matrix *M*, mess_matrix *Z*, double * *nrm*)

Parameters

in	*A*	input matrix
in	*B*	input matrix
in	*C*	input matrix
in	*M*	input matrix
in	*Z*	input computed Z
out	*nrm*	computed 2-norm of the residual

Returns

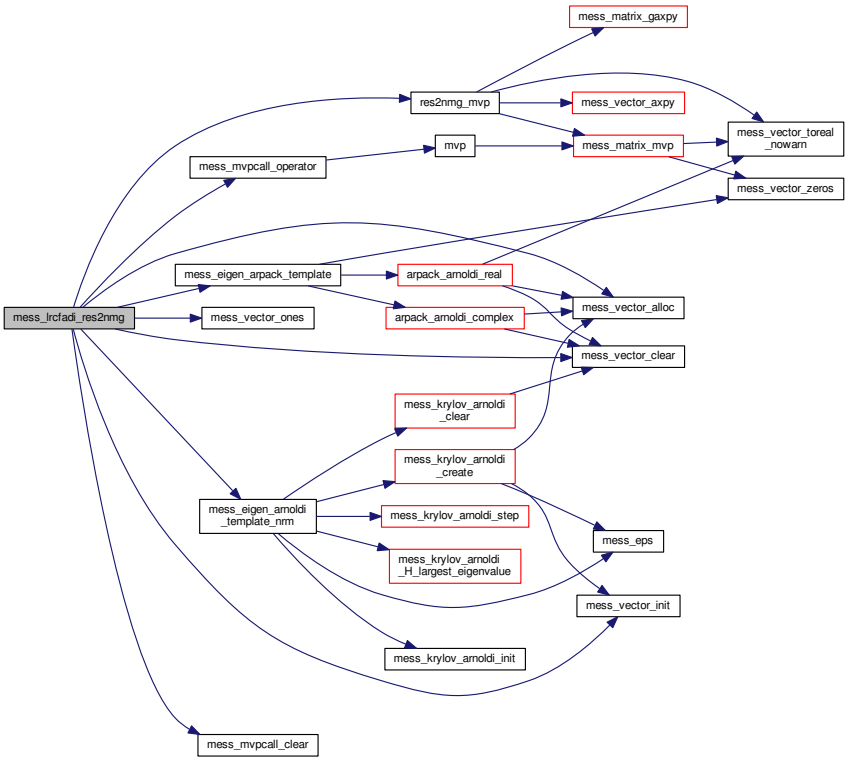
zero on success or a non-zero error value otherwise

The mess_lrcfadi_res2nmg function computes the 2-norm of the generalized Riccati residual of Z exploiting the symmetry of the residual operator

$$R := A^T Z * Z^T M + M^T Z Z^T A + C^T C - M^T Z Z^T B B^T Z Z^T M.$$

That means it computes the spectral radius of this operator by a power iteration, exploitig the sparsity of A and rectangular structure of B, C and Z.
Thus it can be computed in O(n) effort and is therefore much cheaper than the computation of the e.g. the Frobenius-norm.

Here is the call graph for this function:



8.44.2.17 int mess_lrcfadi_res2t (mess_matrix *F*, mess_matrix *G*, mess_matrix *Z*, double * *nrm*)

Parameters

in	*F*	input system matrix
in	*G*	input right hand side
in	*Z*	input computed ZS
out	*nrm*	output computed 2-norm of the residual

Returns

zero on success or a non-zero error value otherwise

The mess_lrcfadi_res2t function computes the 2-norm of the Lyapunov residual of Z exploiting the symmetry of the residual operator

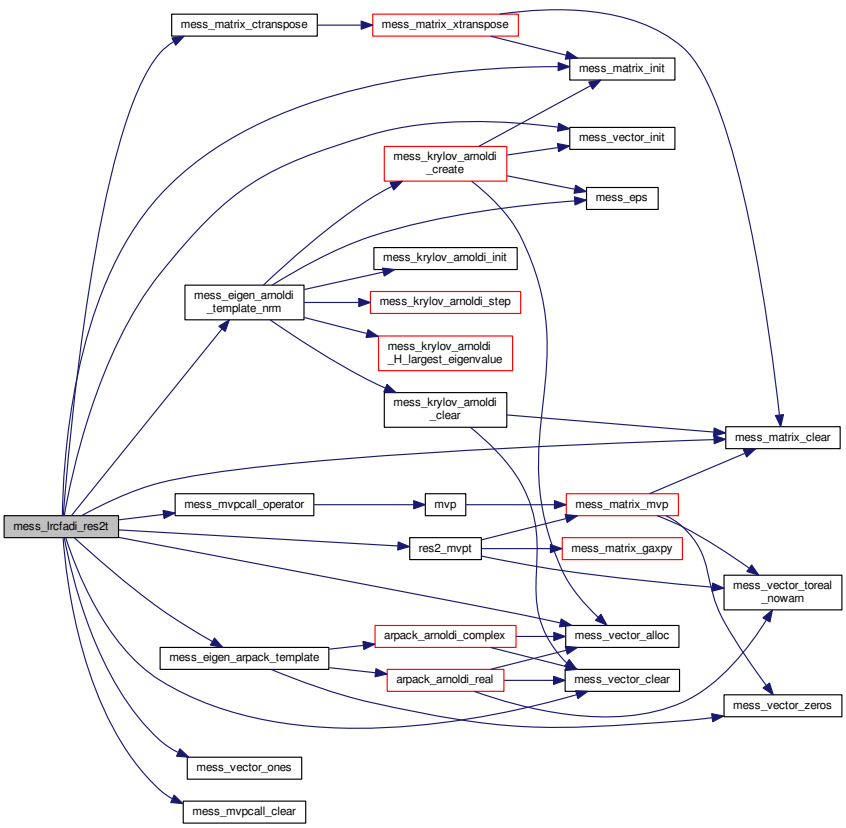
$$R := F'ZZ^T + ZZ^TF + GG^T$$

or

$$R := F'ZZ^T + ZZ^TF + G^TG$$

depending on the dimensions of G.
It is automatically transposed internal if G has the wrong dimension.
That means it computes the spectral radius of this operator by a power iteration, exploiting the sparsity of F and rectangular structure of Z and G.
Thus it can be computed in O(n) effort andis therefore much cheaper than the computation of the e.g. the Frobenius-norm.
The default maximum number of iterations is 20.
The default tolerance is set to $\sqrt{u}$.

Here is the call graph for this function:



8.45 Easy Frontends to Main Functions

This section contains a fronend to compute a solution for some major functions.

Functions

- int mess_lyap (mess_matrix A, mess_matrix E, mess_matrix B, mess_matrix Z)

  Frontend to compute a factorized solution of a (generalized) Lyapunov Equation.
- int mess_care (mess_matrix A, mess_matrix E, mess_matrix B, mess_matrix C, mess_matrix Z)

  Frontend to compute a factorized solution of a (generalized) Riccati Equation.
- int mess_glyap (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix Y, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix X)

  Solve a standard / generalized Lyapunov equation.
- int mess_gstein (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix Y, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix X)

  Solve a standard / generalized Stein equation.
- int mess_tglyap (mess_operation_t op, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix Y, mess_matrix X)

  Solve a standard / generalized Lyapunov equation from given Schur decomposition.
- int mess_tgstein (mess_operation_t op, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix Y, mess_matrix X)

  Solve a standard / generalized Stein equation from given Schur decomposition.

8.45.1 Detailed Description

It includes a function to solve a Lyapunov Equation

$$AX + XA^T + BB^T = 0$$

or a generalized Lyapunov Equation

$$AXE^T + EXA^T + BB^T = 0.$$

Therefor it returns the Cholesky factor Z of the solution X such that $X \approx ZZ^T$.

8.45.2 Function Documentation

8.45.2.1 int mess_care (mess_matrix *A*, mess_matrix *E*, mess_matrix *B*, mess_matrix *C*, mess_matrix *Z*)

Parameters

in	*A*	input matrix of the Riccati Equation A
in	*E*	input matrix of the Riccati Equation E (NULL if E does not exist)
in	*B*	input matrix of the Riccati Equation B
in	*C*	input matrix of the Riccati Equation C
out	*Z*	Cholesky factor Z of the solution $X \approx ZZ^T$

Returns

zero on success or a non-zero error value otherwise

The mess_care function solves either

$$A^T X + XA - XBB^T X + C^T C = 0$$

or

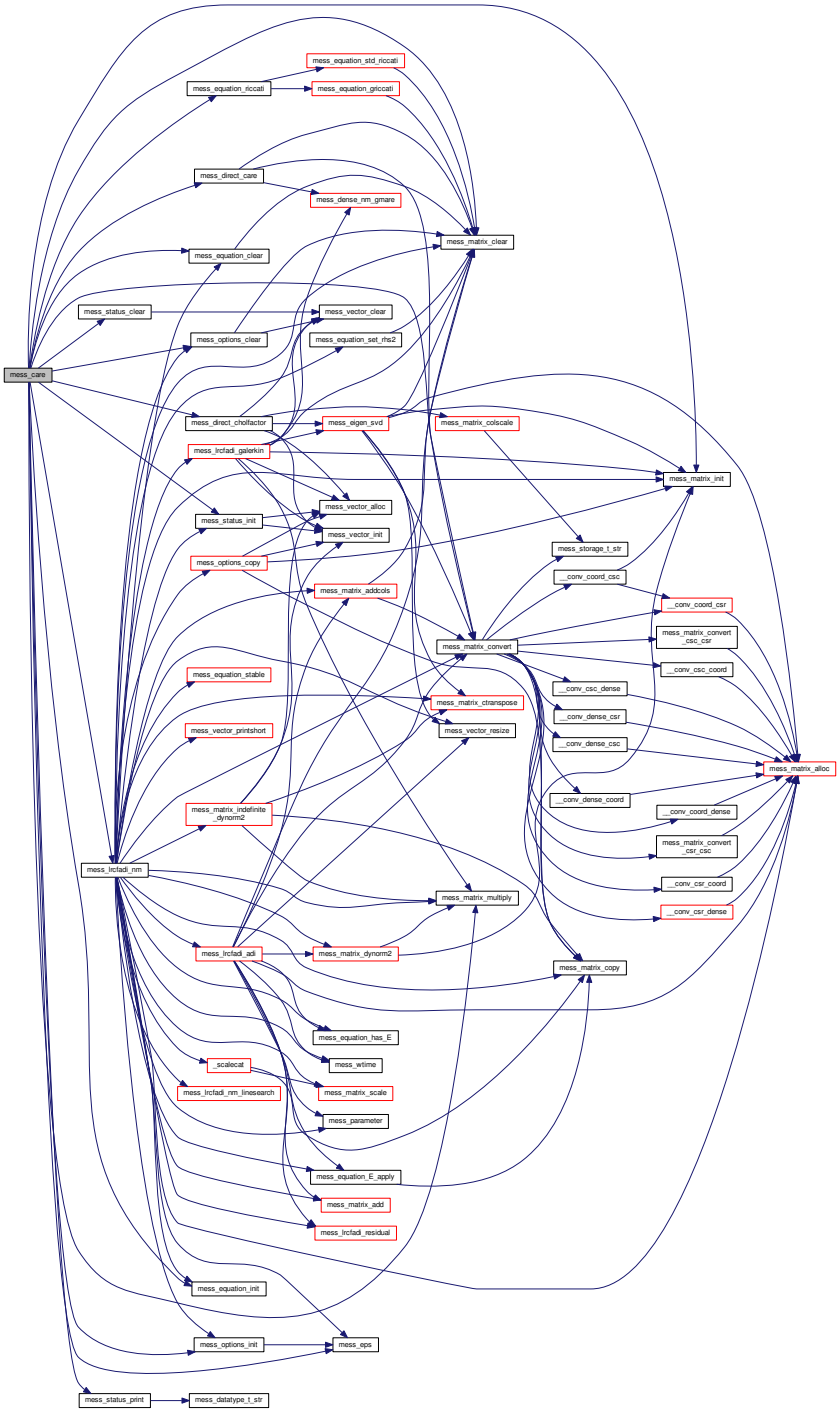
$$A^T XE + E^T XA - E^T XBB^T XE + C^T C = 0$$

depending on the existence of the matrix E.
If the matrix object for E is set to `NULL` the standard Riccati Equation is solved, otherwise the generalized one is solved.
Depending on the storage type of the matrix A the solver decides whether a dense solver mess_direct_care or a sparse solver mess_lrcfadi_nm is used. In both cases the solution is returned as a Cholesky factor $X \approx ZZ^T$.

Attention

The mess_lrcfadi_nm uses its default parameters set by mess_options_init.
If these settings are not applicable to the given problem, the equation have to be solved by configuring the ADI properly and call mess_lrcfadi_adi separately.

Here is the call graph for this function:



8.45.2.2 int mess_glyap (mess_operation_t *op*, mess_matrix *A*, mess_matrix *E*, mess_matrix *Y*, mess_matrix *Ahat*, mess_matrix *QA*, mess_matrix *Ehat*, mess_matrix *QE*, mess_matrix *X*)

Parameters

in	*op*	input determine which equation has to be solved
in	*A*	input real dense matrix A

Parameters

in	*E*	input real dense matrix E, (optional NULL if no generalized equation)
in	*Y*	input real dense matrix symmetric Y
out	*Ahat*	output real dense matrix Schur Form of A, (optional NULL if not wanted)
out	*QA*	output real dense matrix Schur transformation matrix of A, (optional NULL if not wanted)
out	*Ehat*	output real dense matrix Schur Form of E, (optional NULL if not wanted)
out	*QE*	output real dense matrix Schur transformation matrix of E, (optional NULL if not wanted)
out	*X*	output real dense symmetrix matrix, solution of equation X

Returns

zero on success or a non-zero error value otherwise

The mess_glyap function solves the standard / generalized Lyapunov equation.

Depending on E and op the following equation is solved:

- E is NULL and MESS_OP_NONE $AX + XA^T + Y = 0$.
- E is not NULL and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^TX + XA + Y = 0$.
- E is NULL and MESS_OP_NONE $AXE^T + EXA^T + Y = 0$.
- E is not NULL and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^TXE + E^TXA + Y = 0$.

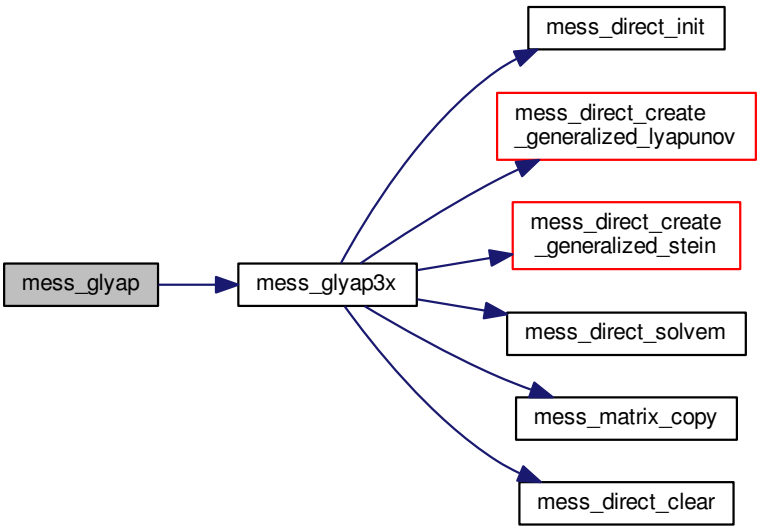
The right hand side matrix Y must be real and symmetric.

In order to solve the equations a (generalized) Schur decomposition of A / (A, E) is computed internally.

If a standard equation is solved then the real Schur decomposition fullfills: $A = Q_A \hat{A} {Q_A}^H$. If a generalized equation is solved then the real Schur decomposition fullfills: $A = Q_A \hat{A} {Q_E}^H$ and $E = Q_A \hat{E} {Q_E}^H$.

If Ahat, QA, Ehat or QE is not NULL then the corresponding matrix of the Schur decomposition is written to that argument.

Here is the call graph for this function:



8.45.2.3 int mess_gstein (mess_operation_t *op*, mess_matrix *A*, mess_matrix *E*, mess_matrix *Y*, mess_matrix *Ahat*, mess_matrix *QA*, mess_matrix *Ehat*, mess_matrix *QE*, mess_matrix *X*)

Parameters

in	*op*	input determine which equation has to be solved
in	*A*	input real dense matrix A
in	*E*	input real dense matrix E, (optional NULL if no generalized equation)
in	*Y*	input real dense matrix symmetric Y
out	*Ahat*	output real dense matrix Schur Form of A, (optional NULL if not wanted)
out	*QA*	output real dense matrix Schur transformation matrix of A, (optional NULL if not wanted)
out	*Ehat*	output real dense matrix Schur Form of E, (optional NULL if not wanted)
out	*QE*	output real dense matrix Schur transformation matrix of E, (optional NULL if not wanted)
out	*X*	output real dense symmetrix matrix, solution of equation X

Returns

zero on success or a non-zero error value otherwise

The mess_gstein function solves the standard / generalized Stein equation.

Depending on E and op the following equation is solved:

- E is NULL and MESS_OP_NONE $AXA^T - X + Y = 0$.
- E is NULL and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $AXA^T - X + Y = 0$.
- E is not NULL and MESS_OP_NONE $AXA^T - EXE^T + Y = 0$.
- E is not NULL and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^TXA - E^TXE + Y = 0$.

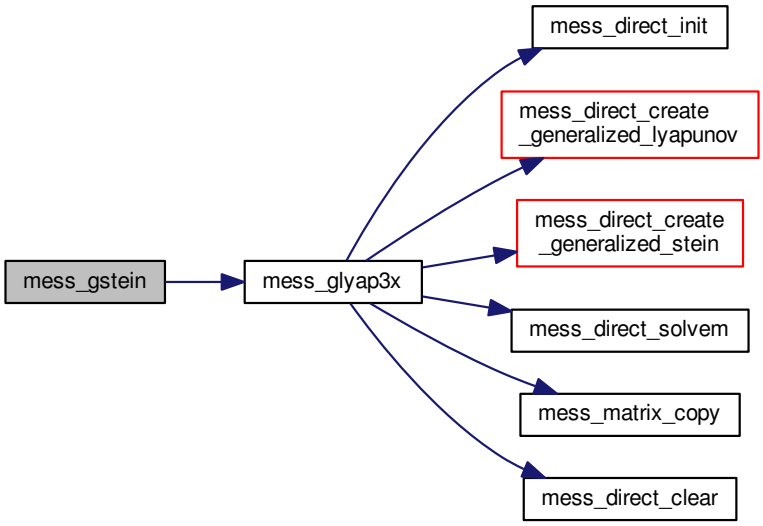
The right hand side matrix Y must be real and symmetric.

In order to solve the equations a (generalized) Schur decomposition of A / (A, E) is computed internally.

If a standard equation is solved then the real Schur decomposition fullfills: $A = Q_A \hat{A} {Q_A}^H$. If a generalized equation is solved then the real Schur decomposition fullfills: $A = Q_A \hat{A} {Q_E}^H$ and $E = Q_A \hat{E} {Q_E}^H$.

If Ahat, QA, Ehat or QE is not NULL then the corresponding matrix of the Schur decomposition is written to that argument.

Here is the call graph for this function:



8.45.2.4 int mess_lyap (mess_matrix *A*, mess_matrix *E*, mess_matrix *B*, mess_matrix *Z*)

Parameters

in	*A*	input A matrix of the Lyapunov Equation
in	*E*	input E matrix of the Lyapunov Equation (NULL if E does not exist)
in	*B*	input right hand side of the Lyapunov Equation
out	*Z*	Cholesky factor Z of the solution $X \approx ZZ^T$

Returns

zero on success or a non-zero error value otherwise

The mess_lyap function solves either

$$AX + XA^T + BB^T = 0$$

or

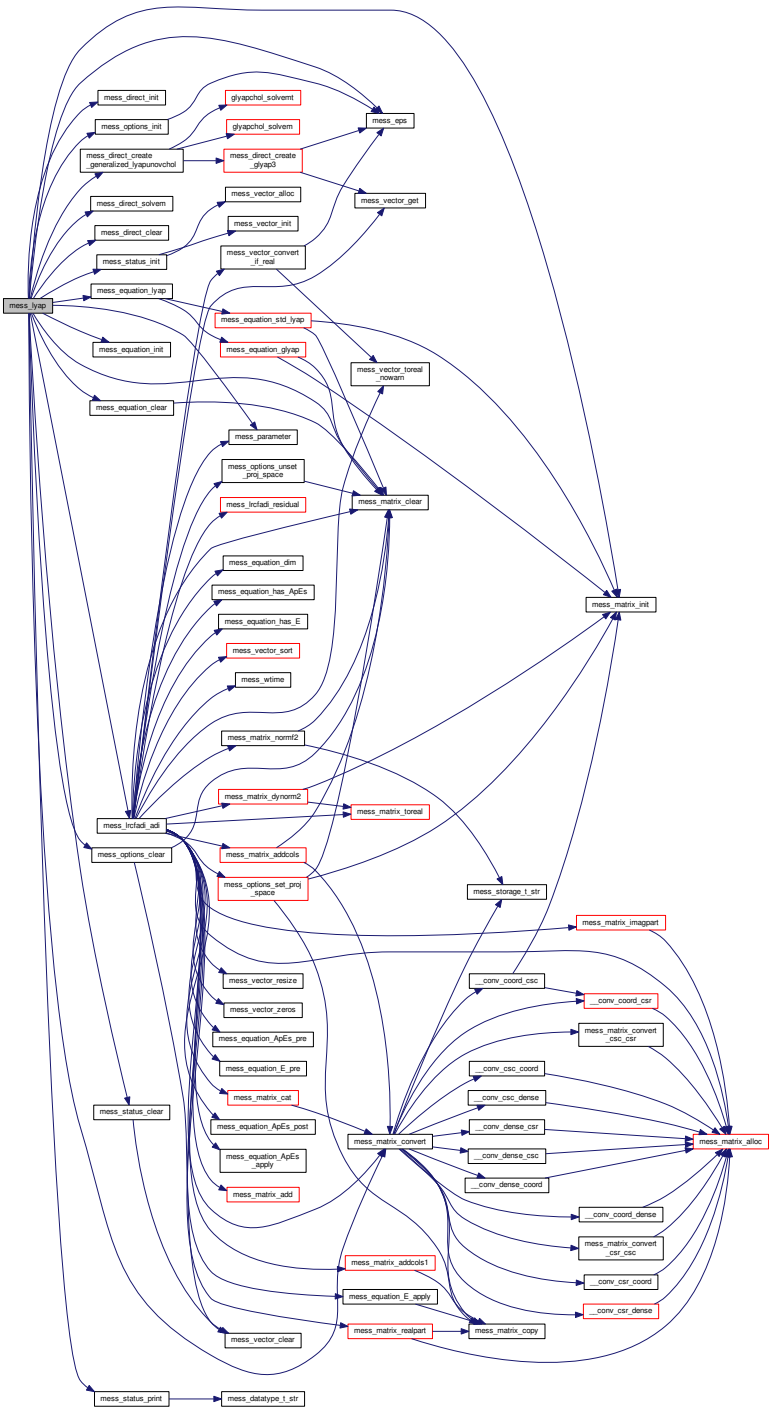
$$AXE^T + EXA^T + BB^T = 0$$

depending on the existence of the matrix E.
If the matrix object for E is set to `NULL` the standard Lyapunov Equation is solved, otherwise the generalized one is solved.
Depending on the storage type of the matrix A the solver decides whether a dense solver mess_direct_create_generalized_lyapunovchol or a sparse solver mess_lrcfadi_adi is used. In both cases the solution is returned as a Cholesky factor $X \approx ZZ^T$.

Attention

The mess_lrcfadi_adi uses its default parameters set by mess_options_init.
If these settings are not applicable to the given problem, the equation have to be solved by configuring the ADI properly and call mess_lrcfadi_adi separately.

Here is the call graph for this function:



8.45.2.5 int mess_tglyap (mess_operation_t *op*, mess_matrix *Ahat*, mess_matrix *QA*, mess_matrix *Ehat*, mess_matrix *QE*, mess_matrix *Y*, mess_matrix *X*)

Parameters

in	*op*	input determine which equation has to be solved
in	*Ahat*	input real dense matrix Schur Form of A
in	*QA*	input real dense matrix Schur transformation matrix of A
in	*Ehat*	input real dense matrix Schur Form of E, (only necessary for generalized equation)
in	*QE*	input real dense matrix Schur transformation matrix of E, (only necessary for generalized equation)
in	*Y*	input real dense matrix symmetric Y
out	*X*	output real dense symmetrix matrix, solution of equation X

Returns

zero on success or a non-zero error value otherwise

The mess_tglyap function solves the standard / generalized Lyapunov equation.

Depending on `Ehat` and `op` the following equation is solved:

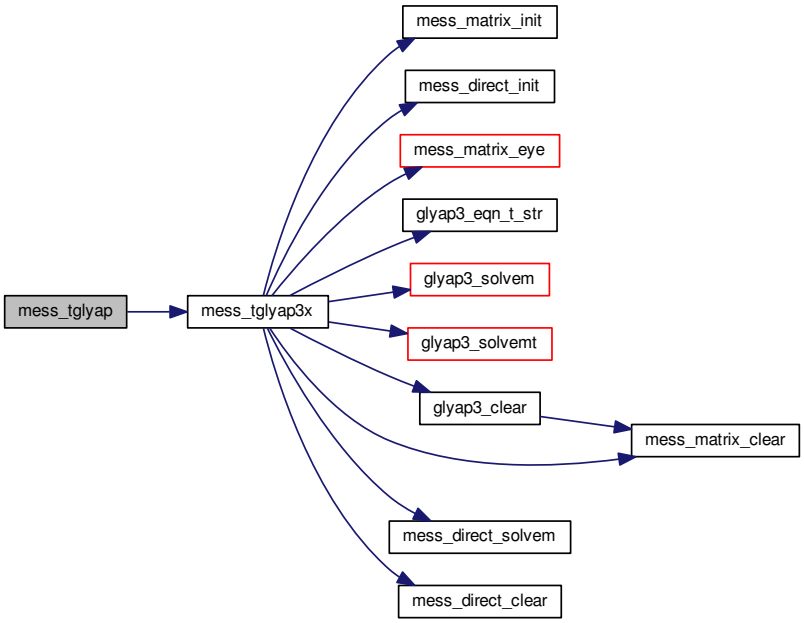
- `Ehat` is `NULL` and MESS_OP_NONE $AX + XA^T + Y = 0$.
- `Ehat` is `NULL` and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^T X + XA + Y = 0$.
- `Ehat` is not `NULL` and MESS_OP_NONE $AXE^T + EXA^T + Y = 0$.
- `Ehat` is not `NULL` and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^T XE + E^T XA + Y = 0$.

The right hand side matrix `Y` must be real and symmetric.

In order to solve the equations a (generalized) Schur decomposition of A / (A, E) is requiered.

If a standard equation is solved then the real Schur decomposition must fullfill: $A = Q_A \hat{A} {Q_A}^H$. If a generalized equation is solved then the real Schur decomposition must fullfill: $A = Q_A \hat{A} {Q_E}^H$ and $E = Q_A \hat{E} {Q_E}^H$.

Here is the call graph for this function:



8.45.2.6 int mess_tgstein (mess_operation_t *op*, mess_matrix *Ahat*, mess_matrix *QA*, mess_matrix *Ehat*, mess_matrix *QE*, mess_matrix *Y*, mess_matrix *X*)

Parameters

in	*op*	input determine which equation has to be solved
in	*Ahat*	input real dense matrix Schur Form of A
in	*QA*	input real dense matrix Schur transformation matrix of A
in	*Ehat*	input real dense matrix Schur Form of E, (only necessary for generalized equation)
in	*QE*	input real dense matrix Schur transformation matrix of E, (only necessary for generalized equation)
in	*Y*	input real dense matrix symmetric Y
out	*X*	output real dense symmetrix matrix, solution of equation X

Returns

zero on success or a non-zero error value otherwise

The mess_tglyap function solves the standard / generalized Stein equation.

Depending on `Ehat` and `op` the following equation is solved:

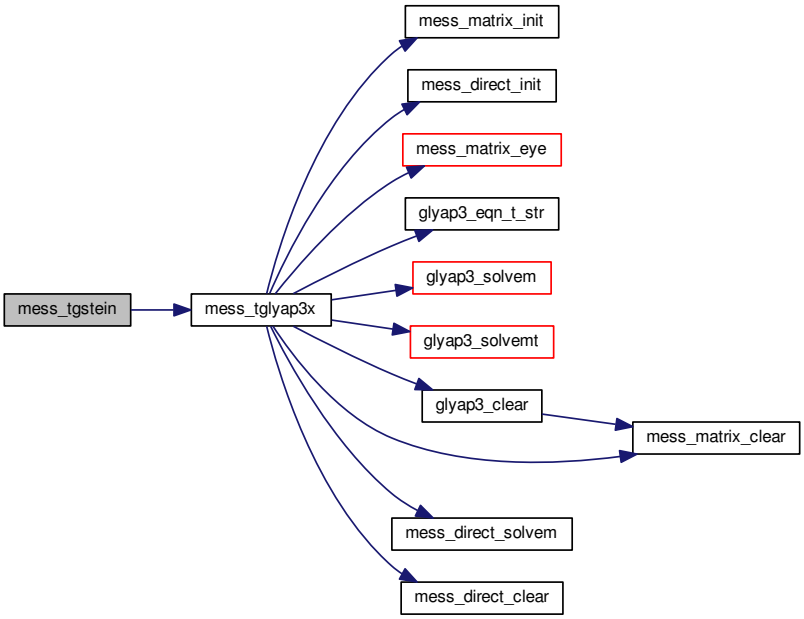
- `Ehat` is `NULL` and MESS_OP_NONE $AX + XA^T + Y = 0$.
- `Ehat` is `NULL` and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^T X + XA + Y = 0$.
- `Ehat` is not `NULL` and MESS_OP_NONE $AXE^T + EXA^T + Y = 0$.
- `Ehat` is not `NULL` and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^T XE + E^T XA + Y = 0$.

The right hand side matrix `Y` must be real and symmetric.

In order to solve the equations a (generalized) Schur decomposition of A / (A, E) is requiered.

If a standard equation is solved then the real Schur decomposition must fullfill: $A = Q_A \hat{A} {Q_A}^H$. If a generalized equation is solved then the real Schur decomposition must fullfill: $A = Q_A \hat{A} {Q_E}^H$ and $E = Q_A \hat{E} {Q_E}^H$.

Here is the call graph for this function:



8.46 Graph Algorithms

This section contains different graph algorithms.

Functions

- int mess_graph_reach (mess_matrix G, mess_matrix B, mess_int_t k, mess_int_t *top, mess_int_t *xi, const mess_int_t *pinv)

  Compute the reachability set from a node in a graph.
- int mess_graph_dfs (mess_matrix G, mess_int_t j, mess_int_t *top, mess_int_t *xi, mess_int_t *pstack, const mess_int_t *pinv)

  Perform a depth-first-search on a directed graph defined by a matrix.
- int mess_graph_lblockpart (mess_matrix G, mess_int_t *perm, mess_int_t *nblocks, mess_int_t *blocks)

  Compute a lower triangular block partition of a matrix.

8.46.1 Detailed Description

It includes functions to

- calculate the reachability set of the nodes over a given graph
- perform a depth-first-search (DFS) on a graph
- permute a matrix of a graph.

The DFS algorithm and the function to compute the reachability set only works for MESS_CSC matrices.
The permutation algorithm only works for MESS_CSR matrices.

8.46.2 Function Documentation

8.46.2.1 int mess_graph_dfs (mess_matrix *G*, mess_int_t *j*, mess_int_t * *top*, mess_int_t * *xi*, mess_int_t * *pstack*, const mess_int_t * *pinv*)

Parameters

in	*G*	input matrix defining graph
in	*pinv*	input inverse row permutation of G (NULL if it does not exist)
in	*j*	input start node for the DFS
out	*top*	index of the element in the chi stack
out	*xi*	stack $xi[top, \cdots, n-1]$ containing set of reached nodes
in	*pstack*	input work array of dimension n

Returns

0 on success or a non zero error code otherwise

The mess_graph_dfs function performs a depth-first-search (DFS) on a graph G starting at node j.
The reached nodes are stored in the xi array $xi[top, \cdots, n-1]$.

The functions are based on [13].

Attention

It only works for Compressed Sparse Column matrices.

8.46.2.2 int mess_graph_lblockpart (mess_matrix *G*, mess_int_t ∗ *perm*, mess_int_t ∗ *nblocks*, mess_int_t ∗ *blocks*)

Parameters

in	*G*	input matrix containing graph
out	*perm*	output permutation
out	*nblocks*	number of blocks
out	*blocks*	array containing start indices of the blocks

Returns

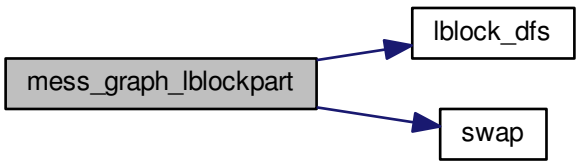
zero on success or a non zero error code otherwise

The mess_graph_lblockpart function computes a permutation to transform matrix G into a lower block triangular matrix.
It returns a permutation array $perm$ which can be used to transform the matrix G to a lower block triangular one.
The $nblocks$ parameter contains the number of blocks and the $blocks$ array contains starting nodes, that means the index of the upper left diagonal entry where a block starts.

Attention

It only work for Compressed Sparse Row matrices.

Here is the call graph for this function:



8.46.2.3 int mess_graph_reach (mess_matrix *G*, mess_matrix *B*, mess_int_t *k*, mess_int_t ∗ *top*, mess_int_t ∗ *xi*, const mess_int_t ∗ *pinv*)

Parameters

in	*G*	input matrix defining graph
in	*B*	input matrix or column with the starting points

Parameters

in	*k*	input number of the column in B which should be used
out	*top*	index of the top element of the stack xi
out	*xi*	stack with the result (length $2 * n$!!!!!)
in	*pinv*	input inverse row permutation for G

Returns

0 on success or a non zero error code otherwise

The mess_graph_reach function calculates the reachability set of the nodes $B(:, k)$ over the graph G. The reached nodes are stored in the xi array $xi[top, \cdots, n-1]$.

The functions are based on [13].

Attention

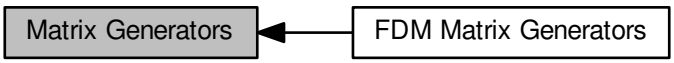
It only works for Compressed Sparse Column matrices.

Here is the call graph for this function:

8.47 Matrix Generators

This section includes definitions and functions to generate matrices.

Collaboration diagram for Matrix Generators:



Modules

- FDM Matrix Generators

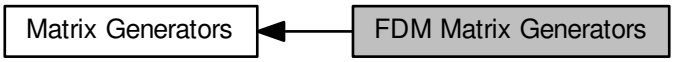
  This category contains a type definition and functions concerning the finite difference discretization.

8.47.1 Detailed Description

8.48 FDM Matrix Generators

This category contains a type definition and functions concerning the finite difference discretization.

Collaboration diagram for FDM Matrix Generators:



Typedefs

- typedef double(∗ mess_matgen_fdm_function) (double x, double y)

  Type definition for the f_x, f_y and g function in the FDM operator.

Functions

- int mess_matgen_fdmmatrix (mess_matrix A, mess_int_t n0, mess_matgen_fdm_function fnfx, mess_matgen_fdm_function fnfy, mess_matgen_fdm_function fng)

  Generate a stiffness matrix A for the finite difference discretization (equidistant grid) of the PDE.
- int mess_matgen_fdmvector (mess_vector v, mess_int_t n0, mess_matgen_fdm_function func)

  Generate a vector v containing values of a function $f(x,y)$ on an equidistant grid in the interior of the unit square.
- int mess_matgen_fdmcolumn (mess_matrix B, mess_int_t n0, mess_matgen_fdm_function func)

  Generate a one-column matrix containing values of a function $f(x,y)$ on an equidistant grid in the interior of the unit square.
- int mess_matgen_fdmrow (mess_matrix C, mess_int_t n0, mess_matgen_fdm_function func)

  Generates a one-row matrix containing values of a function $f(x,y)$ on an equidistant grid in the interior of the unit square.

8.48.1 Detailed Description

The type definition creates functions f_x, f_y and g of the partial differential equation (PDE)

$$\Delta u - f_x \frac{\mathrm{d}u}{\mathrm{d}x} - f_y \frac{\mathrm{d}u}{\mathrm{d}y} - gu = RHS$$

on $\Omega = (0,1) \times (0,1)$ with boundary condition

$$u = 0$$

on $d\Omega$.
There are also functions generating the stiffness matrix A for the finite difference discretization of this PDE, a $(n \times 1)$ matrix B, a $(1 \times n)$ matrix C and a vector v containing the values of a function $f(x,y)$ on an equidistant grid in the interior of the unit square.

8.48.2 Typedef Documentation

8.48.2.1 typedef double(∗ mess_matgen_fdm_function) (double x, double y)

The type definition creates functions f_x, f_y and g of the partial differential equation (PDE)

$$\Delta u - f_x \frac{\mathrm{d}u}{\mathrm{d}x} - f_y \frac{\mathrm{d}u}{\mathrm{d}y} - gu = RHS$$

on $\Omega = (0,1) \times (0,1)$ with boundary condition

$$u = 0$$

on $d\Omega$.

8.48.3 Function Documentation

8.48.3.1 int mess_matgen_fdmcolumn (mess_matrix *B*, mess_int_t *n0*, mess_matgen_fdm_function *func*)

Parameters

out	*B*	$(n \times 1)$ matrix of order $n = n_0^2$ containing the values of $f(x,y)$
in	*n0*	input number of inner grid points in each dimension
in	*func*	input the function f in the space variables x and y (if NULL, $func = 0$)

Returns

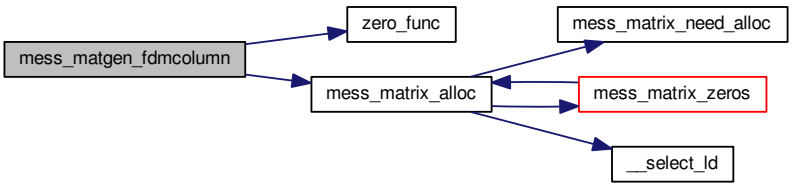
zero on success or a non-zero error value otherwise

The mess_matgen_fdmcolumn function generates a $(n \times 1)$ matrix B which contains the values of a function $f(x,y)$ on an equidistant grid in the interior of the unit square.
The grid points are numbered consistently with those used in the function mess_matgen_fdmmatrix .

This function is just used as an easy way to generate test problems rather than to solve PDEs.

It is the same as mess_matgen_fdmvector but with an other output format.

Here is the call graph for this function:



8.48.3.2 int mess_matgen_fdmmatrix (mess_matrix *A*, mess_int_t *n0*, mess_matgen_fdm_function *fnfx*, mess_matgen_fdm_function *fnfy*, mess_matgen_fdm_function *fng*)

Parameters

out	*A*	$(n \times n)$ sparse stiffness matrix, where $n = n0^2$
in	*n0*	input number of inner grid points in each dimension
in	*fnfx*	input pointer to the function describing f_x (if NULL, $f_x = 0$)
in	*fnfy*	input pointer to the function describing f_y (if NULL, $f_y = 0$)
in	*fng*	input pointer to the function describing g (if NULL, $g = 0$)

Returns

zero on success or a non-zero error value otherwise

The mess_matgen_fdmmatrix function generates the stiffness matrix A for the finite difference discretization (equidistant grid) of the PDE

$$\Delta u - f_x \frac{du}{dx} - f_y \frac{du}{dy} - gu = RHS$$

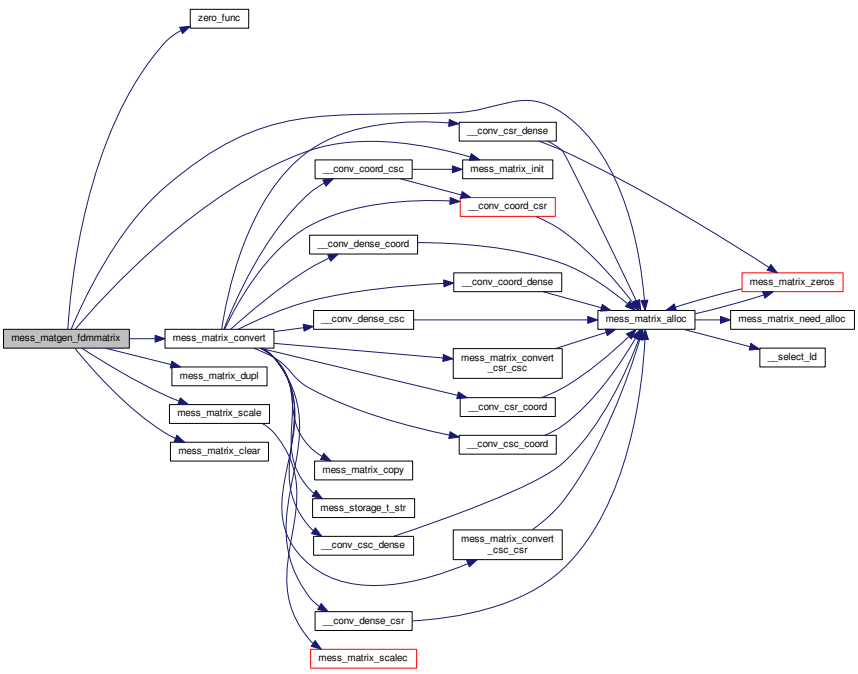
on Ω with boundary condition

$$u = 0$$

on $d\Omega$, where $\Omega = (0,1) \times (0,1)$.

This function is just used as an easy way to generate test problems rather than to solve PDEs.

Here is the call graph for this function:



8.48.3.3 int mess_matgen_fdmrow (mess_matrix *C*, mess_int_t *n0*, mess_matgen_fdm_function *func*)

Parameters

out	*C*	$(1 \times n)$ matrix of order $n = n_0^2$ containing the values of $f(x, y)$
in	*n0*	input number of inner grid points in each dimension
in	*func*	input the function f in the space variables x and y (if NULL, $func = 0$)

Returns

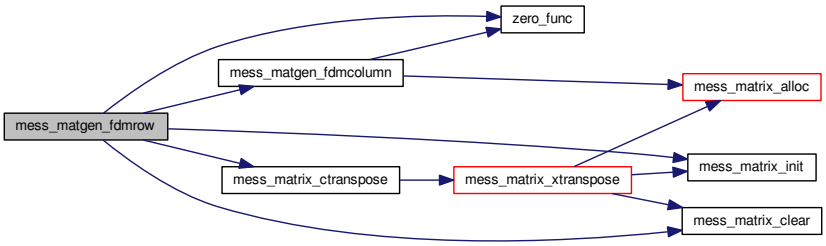
zero on success or a non-zero error value otherwise

The mess_matgen_fdmrow function generates a $(1 \times n)$ matrix C which contains the values of a function $f(x, y)$ on an equidistant grid in the interior of the unit square.
The grid points are numbered consistently with those used in the function mess_matgen_fdmmatrix .

This function is just used as an easy way to generate test problems rather than to solve PDEs.

It is the same as mess_matgen_fdmvector but with an other output format.

Here is the call graph for this function:



8.48.3.4 int mess_matgen_fdmvector (mess_vector *v*, mess_int_t *n0*, mess_matgen_fdm_function *func*)

Parameters

out	*v*	vector of order $n = n_0^2$ containing the values of $f(x, y)$
in	*n0*	input number of inner grid points in each dimension
in	*func*	input the function f in the space variables x and y (if NULL, $func = 0$)

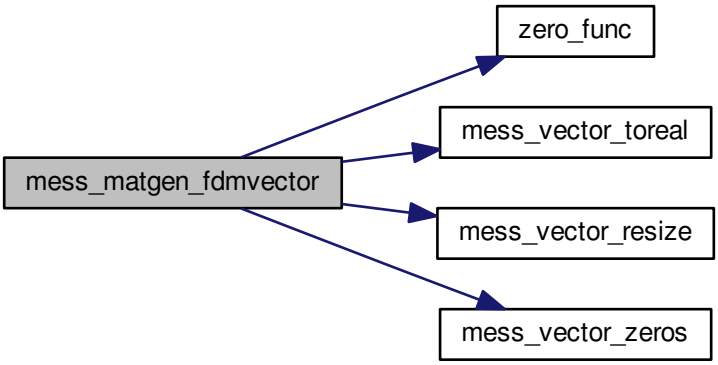
Returns

zero on success or a non-zero error value otherwise

The mess_matgen_fdmvector function generates a vector v which contains the values of a function $f(x, y)$ on an equidistant grid in the interior of the unit square.
The grid points are numbered consistently with those used in the function mess_matgen_fdmmatrix .

This function is just used as an easy way to generate test problems rather than to solve PDEs.

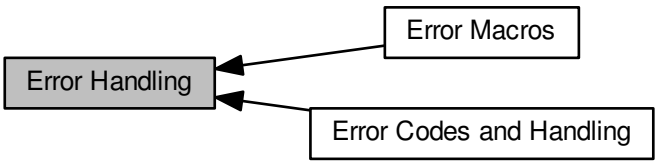
Here is the call graph for this function:



8.49 Error Handling

This section includes different error and warning definitions and functions.

Collaboration diagram for Error Handling:



Modules

- Error Codes and Handling

  This category contains different error definitions and functions.
- Error Macros

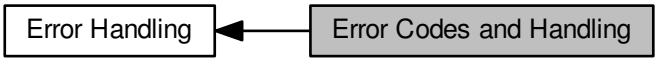
  This category contains different error and warning definitions.

8.49.1 Detailed Description

8.50 Error Codes and Handling

This category contains different error definitions and functions.

Collaboration diagram for Error Codes and Handling:



Macros

- #define MESS_SUCCESS 0
- #define MESS_ERROR_PYTHON 225
- #define MESS_ERROR_DYNSYS 229
- #define MESS_ERROR_CONVERGE 230
- #define MESS_ERROR_GENERAL 231
- #define MESS_ERROR_MEMLOC 233
- #define MESS_ERROR_LAPACK 234
- #define MESS_ERROR_EIGENVALUES 235
- #define MESS_ERROR_NOSUPPORT 236
- #define MESS_ERROR_MISSING 237
- #define MESS_ERROR_UMFPACK 238
- #define MESS_ERROR_MEMORY 239
- #define MESS_ERROR_NOTLOAD 240
- #define MESS_ERROR_NOPATTERN 242
- #define MESS_ERROR_NOCOMPLEX 243
- #define MESS_ERROR_SINGULAR 244
- #define MESS_ERROR_DIMENSION 245
- #define MESS_ERROR_CONVERT 246
- #define MESS_ERROR_SYMMETRIC 247
- #define MESS_ERROR_ARGUMENTS 248
- #define MESS_ERROR_STORAGETYPE 249
- #define MESS_ERROR_DATATYPE 250
- #define MESS_ERROR_DATA 251
- #define MESS_ERROR_WRONG_HEADER 252
- #define MESS_ERROR_FILEIO 253
- #define MESS_ERROR_NULLPOINTER 254
- #define MESS_ERROR_NOTNULLPOINTER 255
- #define MESS_ERROR_MISC 256
- #define MESS_ERROR_CHOLMOD 257
- #define MESS_ERROR_MKLPARDISO 258
- #define MESS_ERROR_PYTHON_SIG 259

Functions

- char * mess_get_error (int code)

  Return an error message.
- int mess_set_errorlevel (int level)

  Set error level.

Variables

- int mess_error_level

8.50.1 Detailed Description

Definitions include input, output, convergence, runtime, memory, implementation errors.
The functions create strings to describe the error or set a certain error level.

8.50.2 Macro Definition Documentation

8.50.2.1 #define MESS_ERROR_ARGUMENTS 248

Passed arguments are wrong.

8.50.2.2 #define MESS_ERROR_CHOLMOD 257

`CHOLMOD` returned an error

8.50.2.3 #define MESS_ERROR_CONVERGE 230

Convergence error

8.50.2.4 #define MESS_ERROR_CONVERT 246

Conversion between the matrix formats is impossible.

8.50.2.5 #define MESS_ERROR_DATA 251

While reading a matrix there is wrong data in the input stream.

8.50.2.6 #define MESS_ERROR_DATATYPE 250

Matrix does not have the right data type.

8.50.2.7 #define MESS_ERROR_DIMENSION 245

The dimension of the given objects does not fit.

8.50.2.8 #define MESS_ERROR_DYNSYS 229

Dynamical system error

8.50.2.9 #define MESS_ERROR_EIGENVALUES 235

The called function has problems with the eigenvalues of the matrix.

8.50.2.10 #define MESS_ERROR_FILEIO 253

Error while file I/O

8.50.2.11 #define MESS_ERROR_GENERAL 231

General error

8.50.2.12 #define MESS_ERROR_LAPACK 234

`LAPACK` returned an error

8.50.2.13 #define MESS_ERROR_MEMLOC 233

Memory location error

8.50.2.14 #define MESS_ERROR_MEMORY 239

Memory allocation was not possible.

8.50.2.15 #define MESS_ERROR_MISC 256

Miscellaneous error

8.50.2.16 #define MESS_ERROR_MISSING 237

The called function is not implemented.

8.50.2.17 #define MESS_ERROR_MKLPARDISO 258

`MKL-PARDISO` return an error

8.50.2.18 #define MESS_ERROR_NOCOMPLEX 243

The called function does not work on complex matrices.

8.50.2.19 #define MESS_ERROR_NOPATTERN 242

The called function does not work on pattern matrices.

8.50.2.20 #define MESS_ERROR_NOSUPPORT 236

The called function is not supported by the compile time configuration.

8.50.2.21 #define MESS_ERROR_NOTLOAD 240

The requested index in a multisolver is not loaded.

8.50.2.22 #define MESS_ERROR_NOTNULLPOINTER 255

A not nullpointer is passed to a function, but it is assumed to point to null.

8.50.2.23 #define MESS_ERROR_NULLPOINTER 254

A nullpointer is passed to a function.

8.50.2.24 #define MESS_ERROR_PYTHON 225

Error in the Python interface or Executing `Python`.

8.50.2.25 #define MESS_ERROR_PYTHON_SIG 259

Got Ctrl-C from Python

8.50.2.26 #define MESS_ERROR_SINGULAR 244

The given matrix is singular.

8.50.2.27 #define MESS_ERROR_STORAGETYPE 249

Matrix does not have the right storage format.

8.50.2.28 #define MESS_ERROR_SYMMETRIC 247

Problems with symmetric matrices

8.50.2.29 #define MESS_ERROR_UMFPACK 238

`UMFPACK` returned an error.

8.50.2.30 #define MESS_ERROR_WRONG_HEADER 252

Matrix file has the wrong file header.

8.50.2.31 #define MESS_SUCCESS 0

Successful return

8.50.3 Function Documentation

8.50.3.1 char* mess_get_error (int *code*)

Parameters

in	*code*	input number of error

Returns

string with error message described in $code$

The mess_get_error function returns a string which describes the error named in $code$.

8.50.3.2 int mess_set_errorlevel (int *level*)

Parameters

in	*level*	input number of error level to set

Returns

zero on success or a non-zero error value otherwise

The mess_set_errorlevel function sets the error level of **M.E.S.S.**
Possible error values are:

- 0 - display no errors,
- 1 - display only errors (default for release),
- 2 - display errors and warnings,
- 3 - display errors, warnings and infos (default for debug).

8.50.4 Variable Documentation

8.50.4.1 int mess_error_level

Global variable to set the error level/verbosity.

8.51 Error Macros

This category contains different error and warning definitions.

Collaboration diagram for Error Macros:

Macros

- #define MSG_ERROR(text, args...)
- #define MSG_WARN(text, args...)
- #define MSG_INFO(text, args...)
- #define MSG_WARN_NOBT(text, args...)
- #define MSG_PRINT(text, args...)
- #define MSG_FNAME(nn) static const char∗ __funcname = nn
- #define FUNCTION_FAILURE_HANDLE(err, cond, fun)
- #define FUNCTION_FAILURE_HANDLE_OMP(err, cond, fun)

8.51.1 Detailed Description

It includes definitions to print warnings concerning the number of the LOGLEVEL. Additionally it contains definitions to print the name of the current function or error messages concerning the LOGLEVEL.

8.51.2 Macro Definition Documentation

8.51.2.1 #define FUNCTION_FAILURE_HANDLE(*err, cond, fun*)

Value:

```
if ( (cond)) {
      \
        MSG_ERROR(" %s returned with %d - %s\n", #fun, (int)err,
      mess_get_error((int)err));          \
        return ((int)err);                                                    \
    }
```

Function failure handler

8.51.2.2 #define FUNCTION_FAILURE_HANDLE_OMP(*err, cond, fun*)

Value:

```
if ( (cond)) {
      \
        MSG_ERROR(" %s returned with %d - %s\n", #fun, (int) err,
      mess_get_error((int)err));          \
    }
```

8.51.2.3 #define MSG_ERROR(*text, args...*)

Value:

```
if ( mess_error_level > 0){ \
    csc_error_message("%s: %s(%5d) - error: \t" text ,__FILE__,__funcname,__LINE__, ## args); \
    csc_show_backtrace(); \
    fflush(stderr);\
}
```

Print error messages on stderr if LOGLEVEL is bigger than zero.

8.51.2.4 #define MSG_FNAME(*nn*) static const char* __funcname = nn

Name of the current function

8.51.2.5 #define MSG_INFO(*text, args...*)

Value:

```
if ( mess_error_level > 2) { \
    csc_info_message("%s: %s(%5d) - info: \t" text ,__FILE__,__funcname,__LINE__, ## args); \
    fflush(stderr); \
}
```

Print error messages on stderr if LOGLEVEL is bigger than three.

8.51.2.6 #define MSG_PRINT(*text, args...*)

Value:

```
{               \
    csc_print_message(text, ## args); \
    fflush(stdout); \
}
```

Print usual messages on stdout.

8.51.2.7 #define MSG_WARN(*text, args...*)

Value:

```
if ( mess_error_level > 1) { \
    csc_warn_message("%s: %s(%5d) - warning: \t" text ,__FILE__,__funcname,__LINE__, ## args); \
    csc_show_backtrace(); \
    fflush(stderr); \
}
```

Print warning messages on stderr if LOGLEVEL is bigger than one.

8.51.2.8 #define MSG_WARN_NOBT(*text, args...*)

Value:

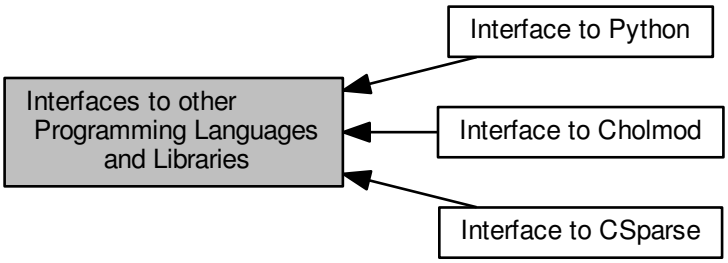
```
if ( mess_error_level > 0) { \
    csc_warn_message("%s: %s(%5d) - warning: \t" text ,__FILE__,__funcname,__LINE__, ## args); \
    fflush(stderr); \
}
```

Print a warning without the backtrace

8.52 Interfaces to other Programming Languages and Libraries

This section contains several categories converting objects of another programming language to C objects and vice versa.

Collaboration diagram for Interfaces to other Programming Languages and Libraries:



Modules

- Interface to CSparse

  There are no definitions or functions in this category.
- Interface to Cholmod

  This category contains different conversion functions.
- Interface to Python

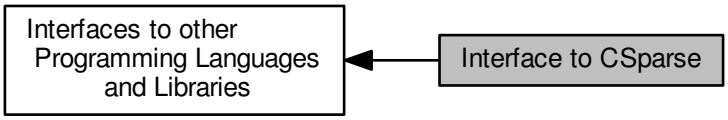
  This category contains different files, definitions and functions for `Python`.

8.52.1 Detailed Description

8.53 Interface to CSparse

There are no definitions or functions in this category.

Collaboration diagram for Interface to CSparse:



Functions

- static int mess_matrix_from_csparse_di (const cs_di *input, mess_matrix output)

  Copy a real matrix from `CSparse` *to* ***M.E.S.S.*** *(double-integer variant).*
- static int mess_matrix_from_csparse (const cs *input, mess_matrix output)

  Default wrapper for mess_matrix_from_csparse_di.
- static int mess_matrix_from_csparse_dl (const cs_dl *input, mess_matrix output)

  Copy a real matrix from `CSparse` *to* ***M.E.S.S.*** *(double-long variant).*
- static int mess_matrix_from_csparse_ci (const cs_ci *input, mess_matrix output)

  Copy a complex matrix from `CSparse` *to* ***M.E.S.S.*** *(complex-int variant).*
- static int mess_matrix_from_csparse_cl (const cs_cl *input, mess_matrix output)

  Copy a complex matrix from `CSparse` *to* ***M.E.S.S.*** *(complex-long variant).*
- static int mess_matrix_to_csparse_di (mess_matrix input, cs_di **o)

  Copy a real matrix from ***M.E.S.S.*** *to* `CSparse` *(double-int variant).*
- static int mess_matrix_to_csparse (mess_matrix input, cs **o)

  Wrapper around mess_matrix_to_csparse_di.
- static int mess_matrix_to_csparse_dl (mess_matrix input, cs_dl **o)

  Copy a real matrix from ***M.E.S.S.*** *to* `CSparse` *(double-long variant).*
- static int mess_matrix_to_csparse_ci (mess_matrix input, cs_ci **o)

  Copy a complex matrix from ***M.E.S.S.*** *to* `CSparse` *(complex-int variant).*
- static int mess_matrix_to_csparse_cl (mess_matrix input, cs_cl **o)

  Copy a complex matrix from ***M.E.S.S.*** *to* `CSparse` *(complex-long variant).*

8.53.1 Detailed Description

There are only static functions in the `CSparse` structure interface_csparse.h.

8.53.2 Function Documentation

8.53.2.1 static int mess_matrix_from_csparse (const cs * *input*, mess_matrix *output*) [static]

Parameters

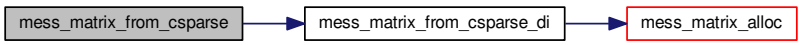
in	*input*	CSparse input matrix
out	*output*	mess output matrix

Returns

zero on success or a non-zero error value

The mess_matrix_from_csparse is a wrapper to mess_matrix_from_csparse_di.

Here is the call graph for this function:



8.53.2.2 static int mess_matrix_from_csparse_ci (const cs_ci * *input*, mess_matrix *output*) [static]

Parameters

in	*input*	input CSparse complex matrix with int index values
out	*output*	output mess_matrix

Returns

zero on success or a non-zero error value

The mess_matrix_from_csparse_ci function copies a complex matrix from CSparse to **M.E.S.S.**.

Here is the call graph for this function:

8.53.2.3 static int mess_matrix_from_csparse_cl (const cs_cl * *input*, mess_matrix *output*) [static]

Parameters

in	*input*	input CSparse complex matrix with long index values
out	*output*	output mess_matrix

Returns

zero on success or a non-zero error value

The mess_matrix_from_csparse_cl function copies a complex matrix from CSparse to **M.E.S.S.**.

Here is the call graph for this function:

8.53.2.4 static int mess_matrix_from_csparse_di (const cs_di * *input*, mess_matrix *output*) [static]

Parameters

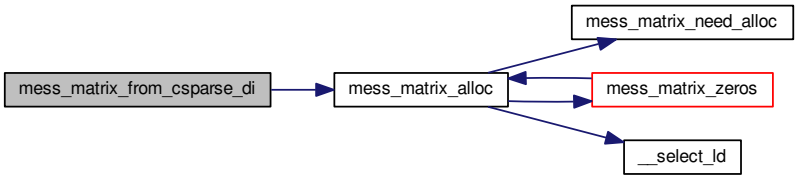
in	*input*	input CSparse real matrix with int index values
out	*output*	output mess_matrix

Returns

zero on success or a non-zero error value

The mess_matrix_from_csparse_di function copies a real matrix from CSparse to **M.E.S.S.**.

Here is the call graph for this function:



8.53.2.5 static int mess_matrix_from_csparse_dl (const cs_dl * *input*, mess_matrix *output*) [static]

Parameters

in	*input*	input CSparse real matrix with long index values
out	*output*	output mess_matrix

Returns

zero on success or a non-zero error value

The mess_matrix_from_csparse_dl function copies a real matrix from CSparse to mess.

Here is the call graph for this function:

8.53.2.6 static int mess_matrix_to_csparse (mess_matrix *input*, cs ** *o*) [static]

Parameters

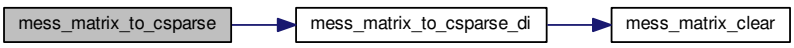
in	*input*	input mess_matrix
out	*o*	output CSparse matrix

The mess_matrix_to_csparse function is a wrapper to mess_matrix_to_csparse_di.

Attention

Interal use only.

Here is the call graph for this function:



8.53.2.7 static int mess_matrix_to_csparse_ci (mess_matrix *input*, cs_ci ** *o*) [static]

Parameters

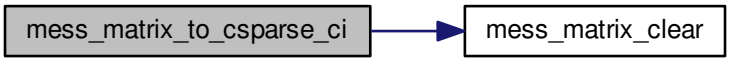
in	*input*	input mess_matrix
out	*o*	output CXSparse matrix with int index values

The mess_matrix_to_csparse_ci function copies a complex matrix from **M.E.S.S.** to CSparse.

Attention

Interal use only.

Here is the call graph for this function:



8.53.2.8 static int mess_matrix_to_csparse_cl (mess_matrix *input*, cs_cl ** *o*) [static]

Parameters

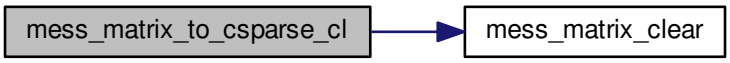
in	*input*	input mess_matrix
out	*o*	output CSparse matrix with long index values

The mess_matrix_to_csparse_cl function copies a complex matrix from **M.E.S.S.** to CSparse.

Attention

Interal use only.

Here is the call graph for this function:



8.53.2.9 static int mess_matrix_to_csparse_di (mess_matrix *input*, cs_di ** *o*) [static]

Parameters

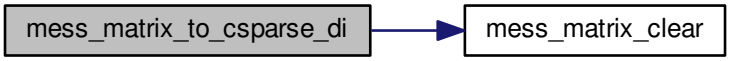
in	*input*	input mess_matrix
out	*o*	output CSparse matrix with int index values

The mess_matrix_to_csparse_di function copies a real matrix from **M.E.S.S.** to CSparse.

Attention

Interal use only.

Here is the call graph for this function:



8.53.2.10 static int mess_matrix_to_csparse_dl (mess_matrix *input*, cs_dl ** *o*) [static]

Parameters

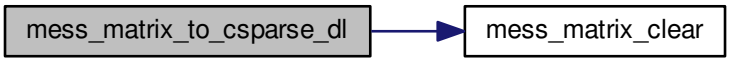
in	*input*	input mess_matrix
out	*o*	output CSparse matrix with long index values

The mess_matrix_to_csparse_dl function copies a real matrix from **M.E.S.S.** to CSparse.

Attention

Interal use only.

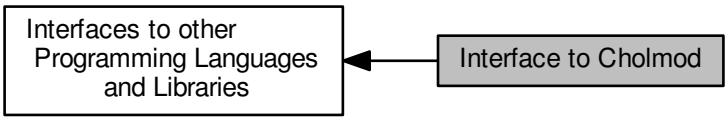
Here is the call graph for this function:



8.54 Interface to Cholmod

This category contains different conversion functions.

Collaboration diagram for Interface to Cholmod:



Functions

- int mess_matrix_convert_dense_to_cholmod (mess_matrix A, cholmod_dense **A_cholmod, cholmod_common *c)
- int mess_matrix_convert_cholmod_to_dense (cholmod_dense *A_cholmod, mess_matrix A, cholmod_common *c)
- int mess_matrix_convert_csc_to_cholmod (mess_matrix A, cholmod_sparse **A_cholmod, cholmod_common *c)
- int mess_matrix_convert_cholmod_to_csc (cholmod_sparse *A_cholmod, mess_matrix A, cholmod_common *c)
- int mess_vector_convert_dense_to_cholmod (mess_vector v, cholmod_dense **v_cholmod, cholmod_common *c)
- int mess_vector_convert_cholmod_to_dense (cholmod_dense *v_cholmod, mess_vector v, cholmod_common *c)

8.54.1 Detailed Description

It includes functions to convert a dense mess_matrix to a `cholmod_dense` datastructure, a compressed sparse column (MESS_CSC) mess_matrix to a `cholmod_sparse` data structure and a mess_vector to a `cholmod_dense` datastructure (or vice versa).

8.54.2 Function Documentation

8.54.2.1 int mess_matrix_convert_cholmod_to_csc (cholmod_sparse * *A_cholmod*, mess_matrix *A*, cholmod_common * *c*)

8.54.2.2 int mess_matrix_convert_cholmod_to_dense (cholmod_dense * *A_cholmod*, mess_matrix *A*, cholmod_common * *c*)

8.54.2.3 int mess_matrix_convert_csc_to_cholmod (mess_matrix *A*, cholmod_sparse ** *A_cholmod*, cholmod_common * *c*)

8.54.2.4 int mess_matrix_convert_dense_to_cholmod (mess_matrix *A*, cholmod_dense ** *A_cholmod*, cholmod_common * *c*)

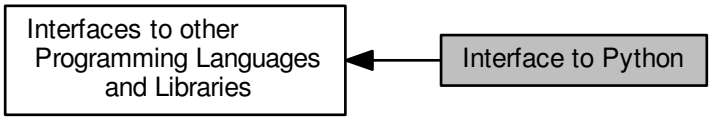
8.54.2.5 int mess_vector_convert_cholmod_to_dense (cholmod_dense * *v_cholmod*, mess_vector *v*, cholmod_common * *c*)

8.54.2.6 int mess_vector_convert_dense_to_cholmod (mess_vector *v*, cholmod_dense ** *v_cholmod*, cholmod_common * *c*)

8.55 Interface to Python

This category contains different files, definitions and functions for `Python`.

Collaboration diagram for Interface to Python:



Files

- file pymess.c

  Functions which should be called from `Python`.

Data Structures

- struct module_state

Macros

- #define NPY_NO_DEPRECATED_API NPY_1_7_API_VERSION
- #define PY_ARRAY_UNIQUE_SYMBOL MESS_VECTOR_MATRIX_PYTHON_C_API
- #define ConvStringtoC(X) PyBytes_AsString(X)
- #define GETSTATE(m) (&_state)
- #define INITERROR return
- #define PYCMESS_ERROR(ret, message) { if ((ret) != 0) { PyErr_SetString(PyExc_RuntimeError,message); return NULL; } }
- #define MESS_Py_XDECREF(...)
- #define PY_SET_LONG(obj, name, value)
- #define PY_SET_DOUBLE(obj, name, value)
- #define PY_SET_VECTOR(obj, name, value)
- #define PY_SET_BOOL(obj, name, value)
- #define PY_GET_LONG(dest, from, name)
- #define PY_GET_DOUBLE(dest, from, name)
- #define PY_GET_VECTOR(dest, from, name)
- #define PY_GET_MATRIX(dest, from, name)
- #define PYMESS_READ_MATRIX(mem, cmatrix, pymatrix, name, errormark)

Functions

- static void init_numpy (void)
- PyObject * pymess_mess_is_debug (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_zlib (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_bzip2 (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_umfpack (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_amd (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_colamd (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_cholmod (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_csparse (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_superlu (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_mklpardiso (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_arpack (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_matio (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_openmp (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_mess64 (PyObject *self, PyObject *args)
- PyObject * pymess_mess_git_id (PyObject *self, PyObject *args)

  Call the mess_git_id function and returns git id, which was used to configure the C-M.E.S.S. build.
- PyObject * pymess_mess_git_branch (PyObject *self, PyObject *args)

  Call the mess_git_branch function and returns git branch, which was used to configure the C-M.E.S.S. build.
- void init_c_interface (void)
- int pymess_is_reimplemented (PyObject *base, PyObject *instance, const char *attr)

  Check if a function is reimplemented by a derived object or equal to the one contained in the base class.
- mess_equation eqn_conv_lyap (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_lyap_so1 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati_so1 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_lyap_so2 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati_so2 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_lyap_dae1 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati_dae1 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_lyap_dae2 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati_dae2 (PyObject *obj, mess_freelist mem)
- PyObject * pymess_test_matrix (PyObject *self, PyObject *args)

  Test functions that can be called from `Python` to ensure that all the conversions are working.
- PyObject * pymess_test_vector (PyObject *self, PyObject *args)

  Test functions that can be called from `Python` to ensure that all the conversions are working.
- PyObject * pymess_test_status (PyObject *self, PyObject *args)
- PyObject * pymess_test_equation (PyObject *self, PyObject *args)
- PyObject * pymess_lrnm (PyObject *self, PyObject *args)
- PyObject * pymess_lradi (PyObject *self, PyObject *args)
- PyObject * pymess_dense_nm_gmpare (PyObject *self, PyObject *args)

  Wrapper around the interface mess_dense_nm_gmpare for use with `Python`.
- PyObject * pymess_lyap (PyObject *self, PyObject *args)

  Wrapper around the easy to use interface mess_lyap for use with `Python`.
- PyObject * pymess_sylvester_sparsedense (PyObject *self, PyObject *args)

  Wrapper around the interface mess_direct_create_sylvester_sparsedense for use with `Python`.
- PyObject * pymess_care (PyObject *self, PyObject *args)

  Wrapper around the easy to use interface mess_care for use with `Python`.
- PyObject * pymess_glyap (PyObject *self, PyObject *args)

  Interface to mess_glyap for use with `Python`.
- PyObject * pymess_gstein (PyObject *self, PyObject *args)

Interface to mess_gstein for use with Python.

- PyObject * pymess_eps (PyObject *self, PyObject *args)

  Return machine epsilon. The pymess_eps is callable as eps and calls internally the mess_eps function and returns the machine epsilon.
- PyObject * pymess_mess_version (PyObject *self, PyObject *args)

  Call the mess_version function. The pymess_mess_version is callable as mess_version and calls internally the mess_version function.
- PyObject * pymess_mess_version_verbose (PyObject *self, PyObject *args)

  Call the mess_version_verbose function. The pymess_mess_version_verbose is callable as mess_version_verbose and calls internally the mess_version_verbose function.
- PyObject * pymess_mess_version_major (PyObject *self, PyObject *args)

  Call the mess_version_major function and returns the major version number.
- PyObject * pymess_mess_version_minor (PyObject *self, PyObject *args)

  Call the mess_version_minor function and returns the minor version number.
- PyObject * pymess_mess_version_patch (PyObject *self, PyObject *args)

  Call the mess_version_patch function and returns the patch version number.
- PyObject * pymess_direct_select (PyObject *self, PyObject *args)

  Wrapper around mess_direct_lu_select for use with Python.
- PyObject * pymess_direct_chol_select (PyObject *self, PyObject *args)

  Wrapper around mess_direct_chol_select for use with Python.
- PyObject * pymess_multidirect_select (PyObject *self, PyObject *args)

  Wrapper around mess_multidirect_select for use with Python.
- mess_matrix matrix_to_c (PyObject *data)

  Converts Python matrix to mess_matrix.
- PyObject * matrix_to_python (mess_matrix c_matrix)

  Converts mess_matrix to Python matrix.
- mess_vector vector_to_c (PyObject *data)

  Converts Python vector to mess_vector.
- PyObject * vector_to_python (mess_vector c_vector, int copy)

  Converts mess_vector to Python vector.
- mess_options mess_options_from_python (PyObject *options)

  Parse the attributes for the lrcfadi algorithm from PyObject options argument.
- PyObject * mess_status_to_python (mess_status stat)

  Convert a mess_status object from C to Python.
- PyObject * pymess_set_errorlevel (PyObject *self, PyObject *args)

  Wrapper around mess_set_errorlevel for use with Python.
- mess_equation mess_equation_from_python (PyObject *obj, mess_freelist mem, mess_equation_t eqntype, int *need_callback)

Variables

- static struct module_state _state
- static PyMethodDef interface_python_methods [ ]

  Function which are callable from Python. Edit below function if you need to add more functions that you wish to be called from Python Avoid editing any other piece of code in the Initialisation section.

8.55.1 Detailed Description

It includes files converting a Python matrix or array to a C matrix or array (and vice versa) and calling functions from Python.
Additionally there are function to convert a mess_matrix or a mess_vector to a Python matrix or vector (and vice versa) and to parse attributes for the LRCF-ADI algorithm from a PyObject options argument.

8.55.2 Macro Definition Documentation

8.55.2.1 #define ConvStringtoC(X) PyBytes_AsString(X)

8.55.2.2 #define GETSTATE(m) (&_state)

8.55.2.3 #define INITERROR return

8.55.2.4 #define MESS_Py_XDECREF(...)

Value:

```
{ \
    void * _stopper = (void * ) -1; \
    PyObject*  _M[] = { __VA_ARGS__, _stopper}; \
    int _i; \
    for ( _i = 0; _M[_i] != _stopper ; _i++) {\
        Py_XDECREF(_M[_i]);\
    }\
}
```

8.55.2.5 #define NPY_NO_DEPRECATED_API NPY_1_7_API_VERSION

8.55.2.6 #define PY_ARRAY_UNIQUE_SYMBOL MESS_VECTOR_MATRIX_PYTHON_C_API

8.55.2.7 #define PY_GET_DOUBLE(*dest, from, name*)

Value:

```
{                                                   \
    PyObject *pv = PyObject_GetAttrString(from, name);                              \
    if (pv){                                                                        \
        dest = PyFloat_AsDouble(pv);                                                \
        Py_DECREF(pv);                                                              \
    }                                                                               \
}
```

8.55.2.8 #define PY_GET_LONG(*dest, from, name*)

Value:

```
{                                                   \
    PyObject *pv = PyObject_GetAttrString(from, name);                          \
    if (pv){                                                                    \
        dest = PyLong_AsLong(pv);                                               \
        Py_DECREF(pv);                                                          \
    }                                                                           \
}
```

8.55.2.9 #define PY_GET_MATRIX(*dest, from, name*)

Value:

```
{                                                   \
    PyObject *pv = PyObject_GetAttrString(from, name);                              \
    if (pv){                                                                        \
        dest = matrix_to_c(pv);                                                     \
        if(dest) mess_matrix_sort(dest);                                            \
        Py_DECREF(pv);                                                              \
    }else{                                                                          \
        dest = NULL;                                                                \
    }                                                                               \
}
```

8.55.2.10 #define PY_GET_VECTOR(*dest*, *from*, *name*)

Value:

```
{                                                   \
    PyObject *pv = PyObject_GetAttrString(from,name);                                \
    if (pv !=NULL && pv != Py_None) {                                                \
        dest = vector_to_c(pv);                                                      \
        Py_DECREF(pv);                                                               \
    }else{                                                                           \
        Py_XDECREF(pv);                                                              \
        dest = NULL;                                                                 \
    }                                                                                \
}
```

8.55.2.11 #define PY_SET_BOOL(*obj*, *name*, *value*)

Value:

```
do { \
    if ( (value) ){ \
        PyObject_SetAttrString((obj), (name), Py_True);\
    } else { \
        PyObject_SetAttrString((obj), (name), Py_False);\
    }\
} while (0);
```

8.55.2.12 #define PY_SET_DOUBLE(*obj*, *name*, *value*)

Value:

```
do { \
    PyObject * _value = PyFloat_FromDouble((value)); \
    PyObject_SetAttrString((obj), (name), _value); \
    Py_XDECREF(_value); \
} while (0);
```

8.55.2.13 #define PY_SET_LONG(*obj*, *name*, *value*)

Value:

```
do { \
    PyObject * _value = PyLong_FromLong((value)); \
    PyObject_SetAttrString((obj), (name), _value); \
    Py_XDECREF(_value); \
} while (0);
```

8.55.2.14 #define PY_SET_VECTOR(*obj*, *name*, *value*)

Value:

```
do { \
    PyObject * _value = vector_to_python(value,1); \
    PyObject_SetAttrString((obj), (name), _value); \
    Py_XDECREF(_value); \
} while (0);
```

8.55.2.15 #define PYCMESS_ERROR(*ret, message*) { if ((ret) != 0) { PyErr_SetString(PyExc_RuntimeError,message); return NULL; } }

8.55.2.16 #define PYMESS_READ_MATRIX(*mem, cmatrix, pymatrix, name, errormark*)

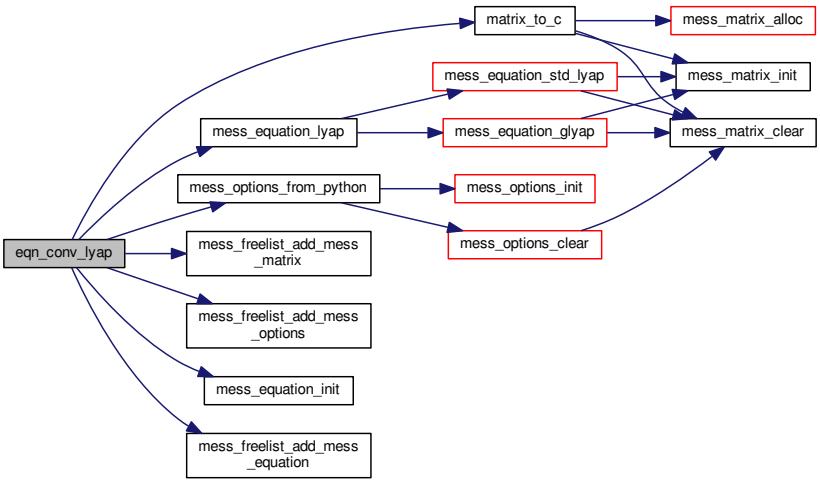
Value:

```
(cmatrix) = matrix_to_c((pymatrix));
                \
    if(!(cmatrix)){
        \
        PyErr_SetString(PyExc_RuntimeError,"Cannot transfer matrix "#name" from Python to C-M.E.S.S.");
        \
        goto errormark;
        \
    }
        \
    mess_freelist_add_mess_matrix(mem,(cmatrix));
```

8.55.3 Function Documentation

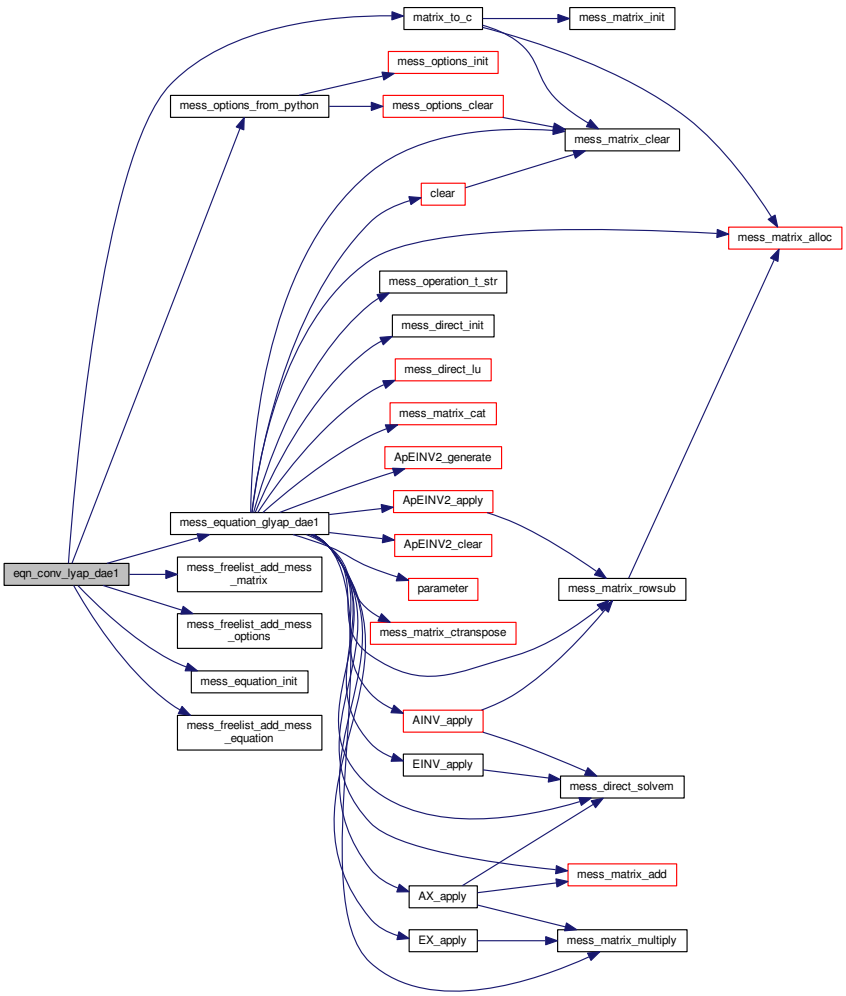
8.55.3.1 mess_equation eqn_conv_lyap (PyObject * *obj*, mess_freelist *mem*)

Here is the call graph for this function:



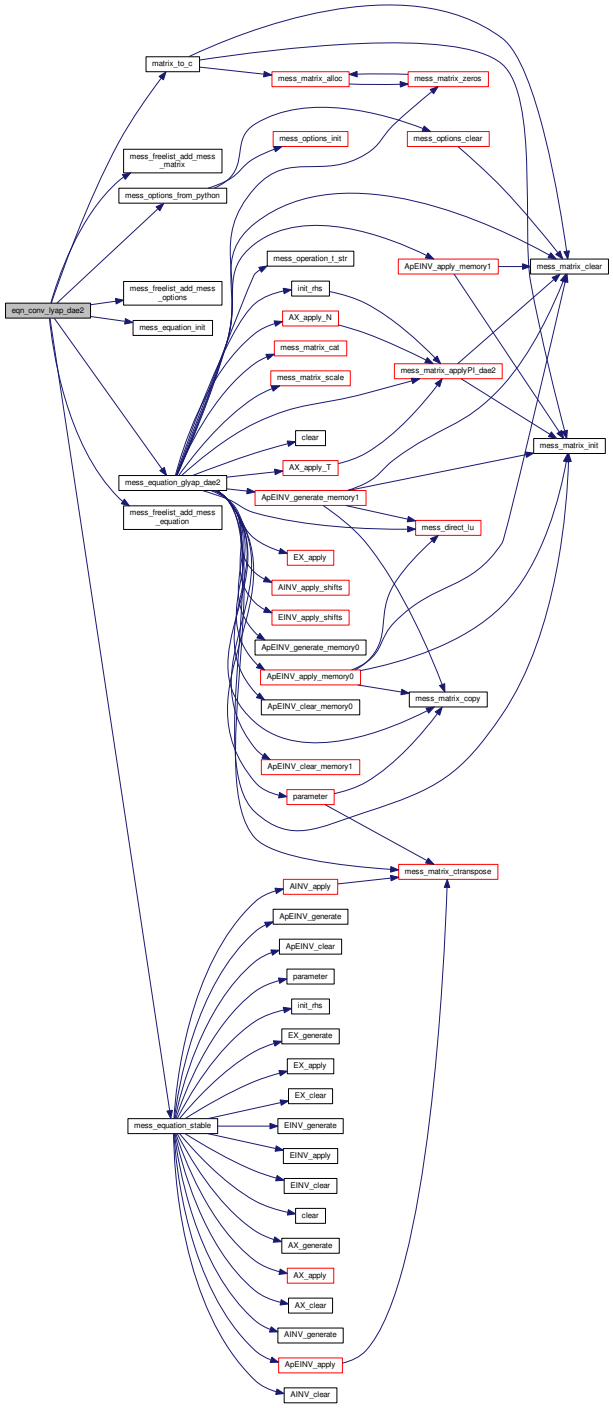
8.55.3.2 mess_equation eqn_conv_lyap_dae1 (PyObject * *obj,* mess_freelist *mem*)

Here is the call graph for this function:



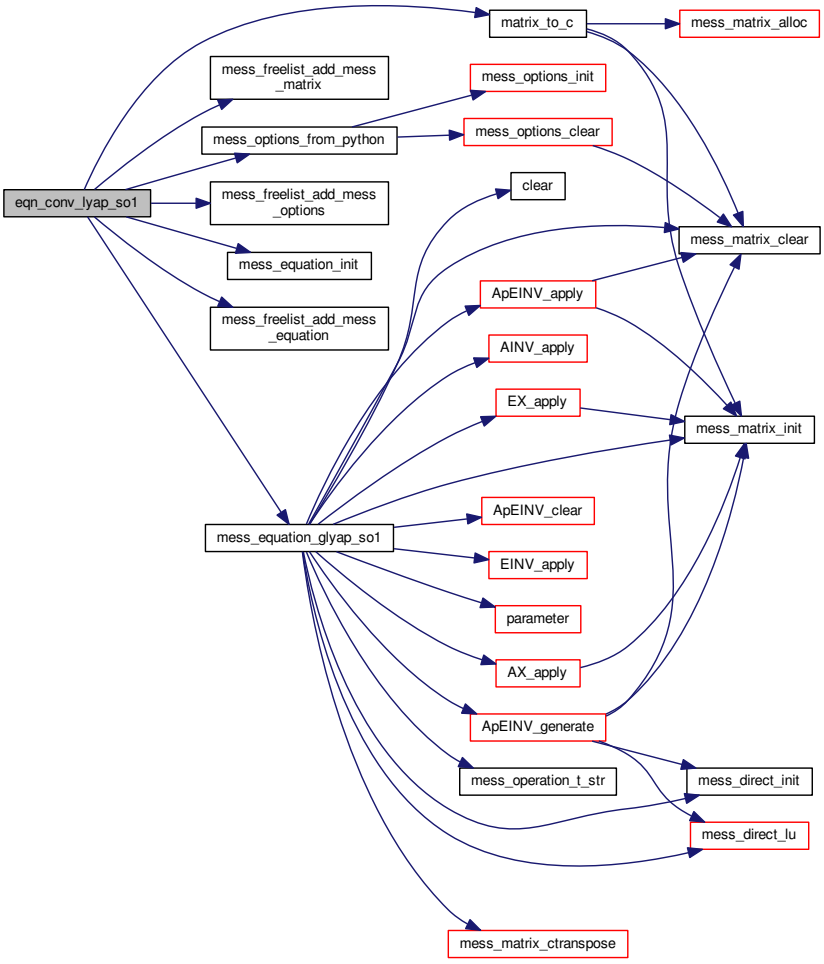
8.55.3.3 mess_equation eqn_conv_lyap_dae2 (PyObject * *obj,* mess_freelist *mem*)

Here is the call graph for this function:



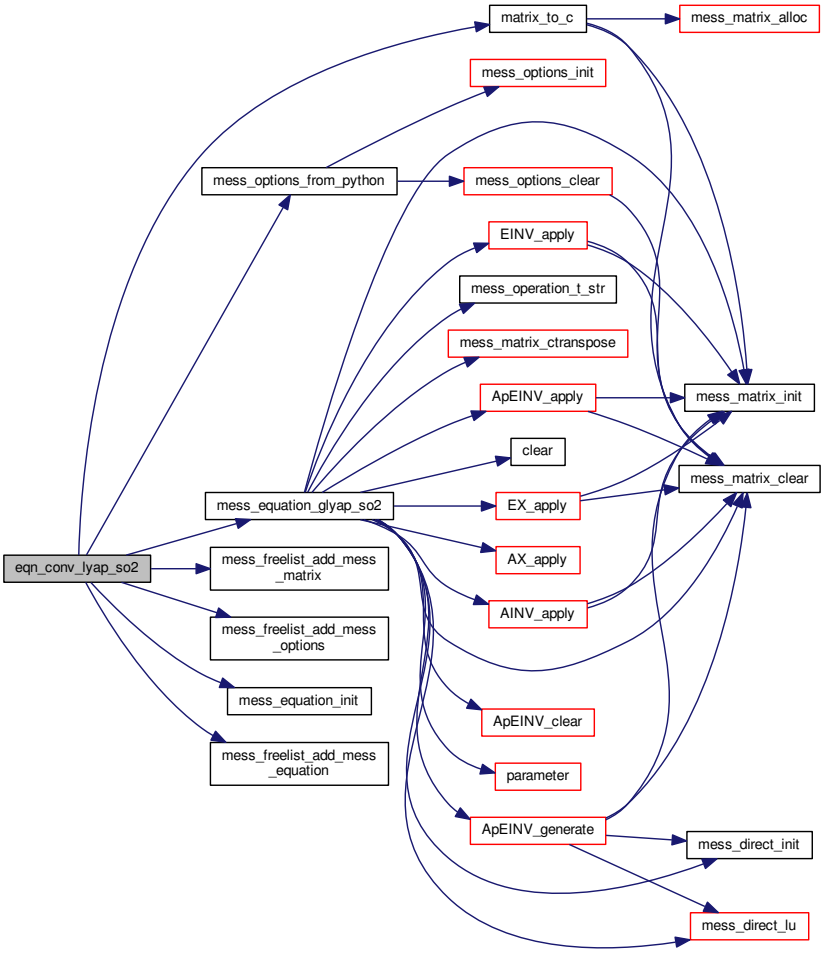
8.55.3.4 mess_equation eqn_conv_lyap_so1 (PyObject * *obj*, mess_freelist *mem*)

Here is the call graph for this function:



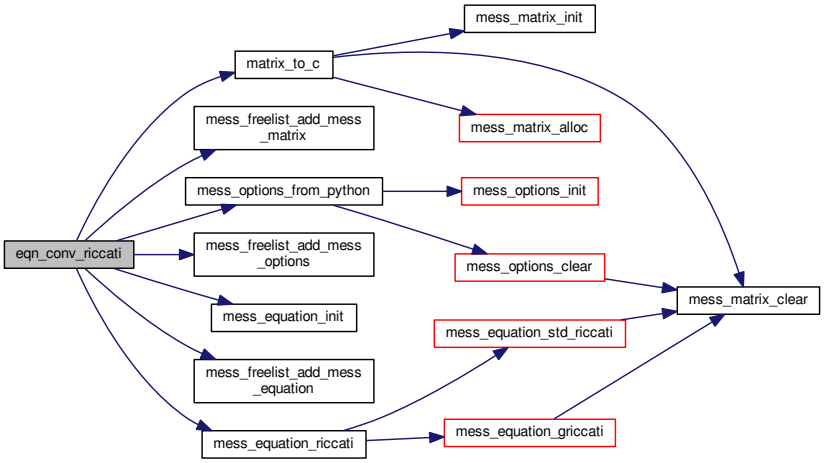
8.55.3.5 mess_equation eqn_conv_lyap_so2 (PyObject * *obj*, mess_freelist *mem*)

Here is the call graph for this function:



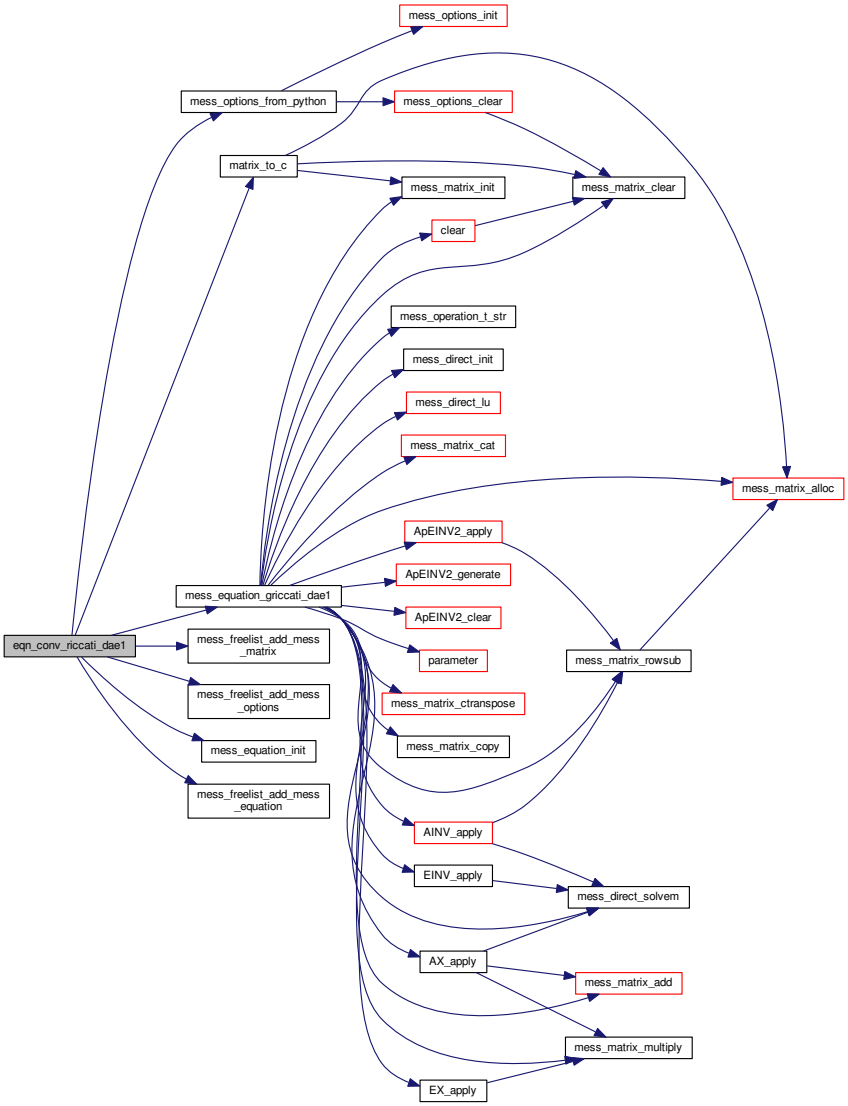
8.55.3.6 mess_equation eqn_conv_riccati (PyObject * *obj*, mess_freelist *mem*)

Here is the call graph for this function:



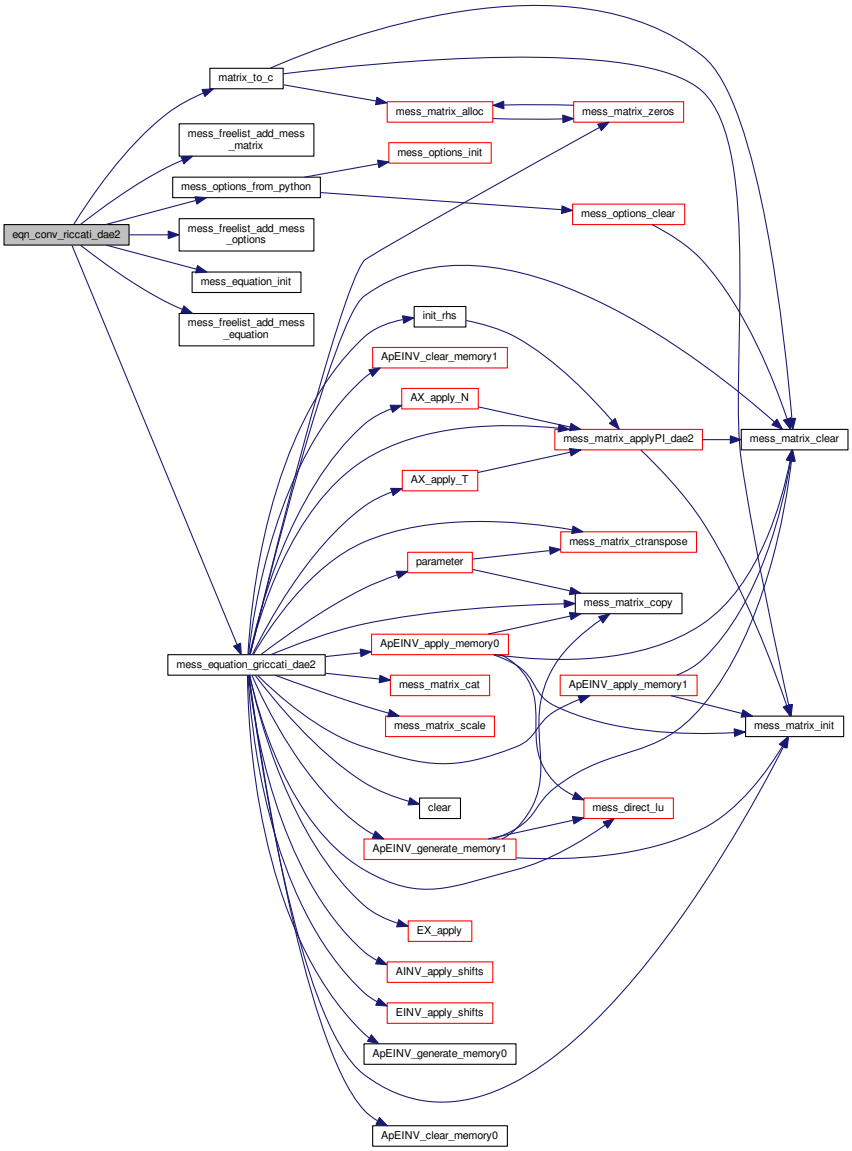
8.55.3.7 mess_equation eqn_conv_riccati_dae1 (PyObject * *obj*, mess_freelist *mem*)

Here is the call graph for this function:



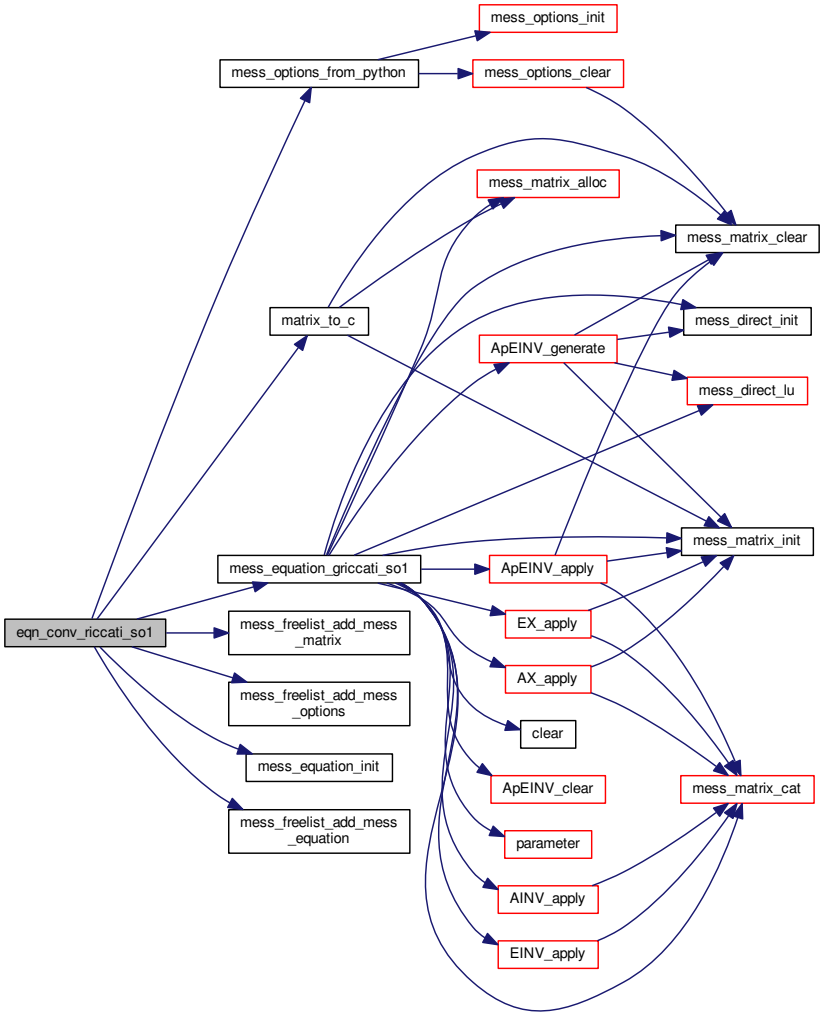
8.55.3.8 mess_equation eqn_conv_riccati_dae2 (PyObject * *obj*, mess_freelist *mem*)

Here is the call graph for this function:



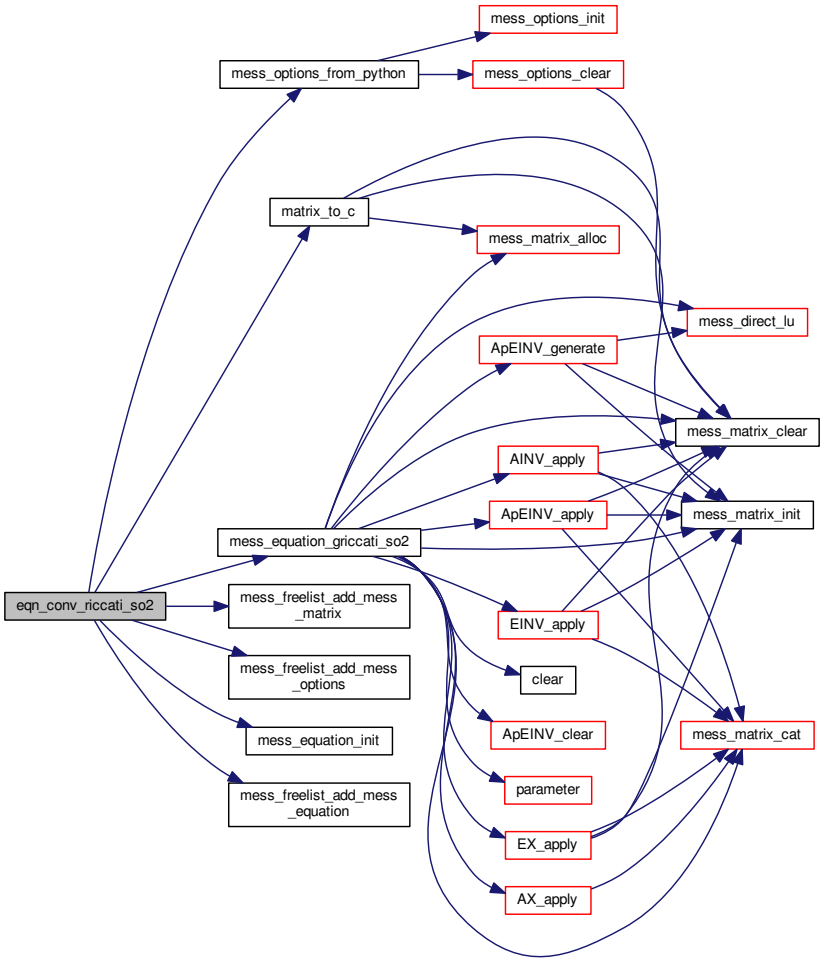
8.55.3.9 mess_equation eqn_conv_riccati_so1 (PyObject * *obj*, mess_freelist *mem*)

Here is the call graph for this function:



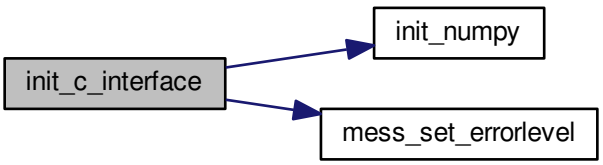
8.55.3.10 mess_equation eqn_conv_riccati_so2 (PyObject * *obj*, mess_freelist *mem*)

Here is the call graph for this function:



8.55.3.11 void init_c_interface (void)

Here is the call graph for this function:



8.55.3.12 static void init_numpy (void) [static]

8.55.3.13 mess_matrix matrix_to_c (PyObject * *data*)

Parameters

in	*data*	input Python matrix

Returns

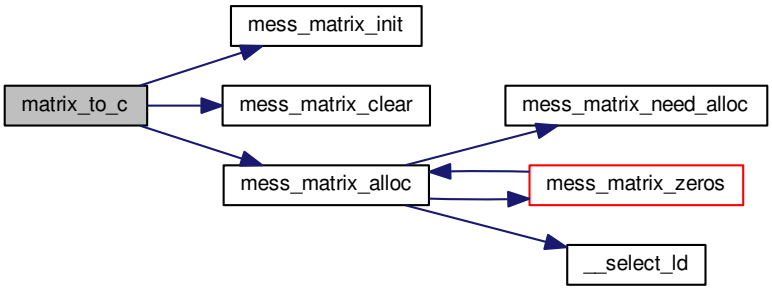
converted matrix as mess_matrix

The matrix_to_c function supports the numpy ndarray and scipy csc coo and csr matrix and the following datatypes

- int8
- int16
- int32
- int64
- float32
- float64
- complex64
- complex128

If the matrix cannot be converted into a MESS internal matrix NULL is returned. This either identifies that an error occured or it is an empty matrix or of unknown type. Please check the return value before your use it. If a conversion error occured the Python-Execption is set.

Here is the call graph for this function:



8.55.3.14 PyObject* matrix_to_python (mess_matrix *c_matrix*)

Parameters

in	*c_matrix*	input mess_matrix

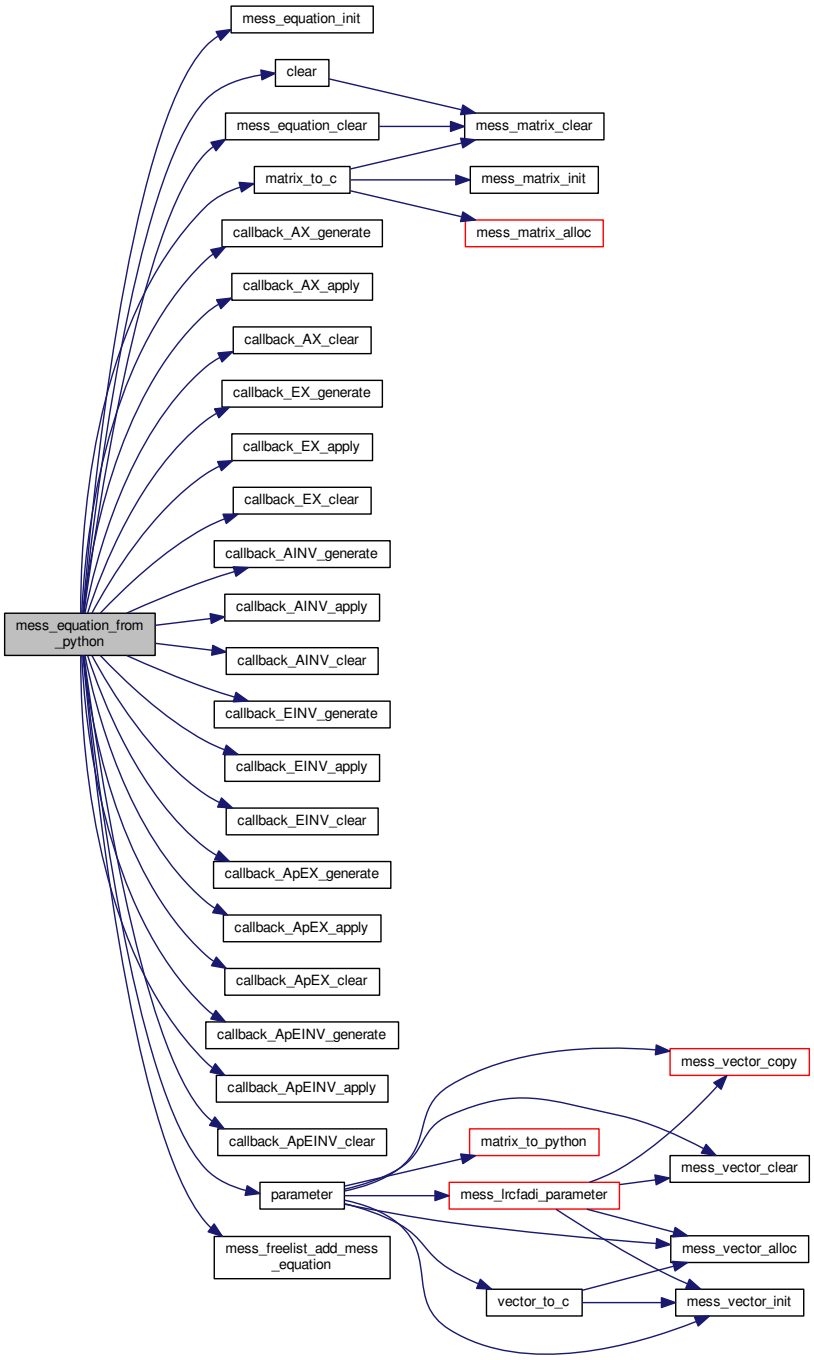
Returns

Python matrix Attention the matrix is moved from C to Python. The matrix does not longer exist correctly in C.

Here is the call graph for this function:

8.55.3.15 mess_equation mess_equation_from_python (PyObject * *obj,* mess_freelist *mem,* mess_equation_t *eqntype,* int * *need_callback*)

Here is the call graph for this function:



8.55.3.16 mess_options mess_options_from_python (PyObject * *options*)

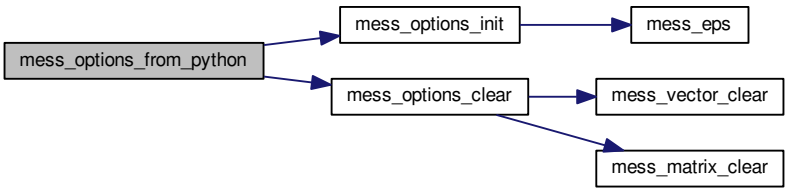
Parameters

in	*options*	input PyObject with opt for lrcfadi algorithm

Returns

parsed mess_options from PyObject options

Here is the call graph for this function:



8.55.3.17 PyObject* mess_status_to_python (mess_status *status*)

Parameters

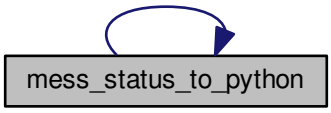
in	*status*	input mess_status object

Returns

The mess_status object convertet to an instance of pymess.status or Py_None.

The mess_status_to_python function converts a mess_status object to an instance of pymess.status.

Here is the call graph for this function:



8.55.3.18 PyObject* pymess_care (PyObject * *self*, PyObject * *args*)

Parameters

in	*self*	input
in	*args*	input

Returns

Python object containing Z or None in case of an error and an exception.

The pymess_care function provides and wrapper around the mess_care function in order to use from Python. The function is called as

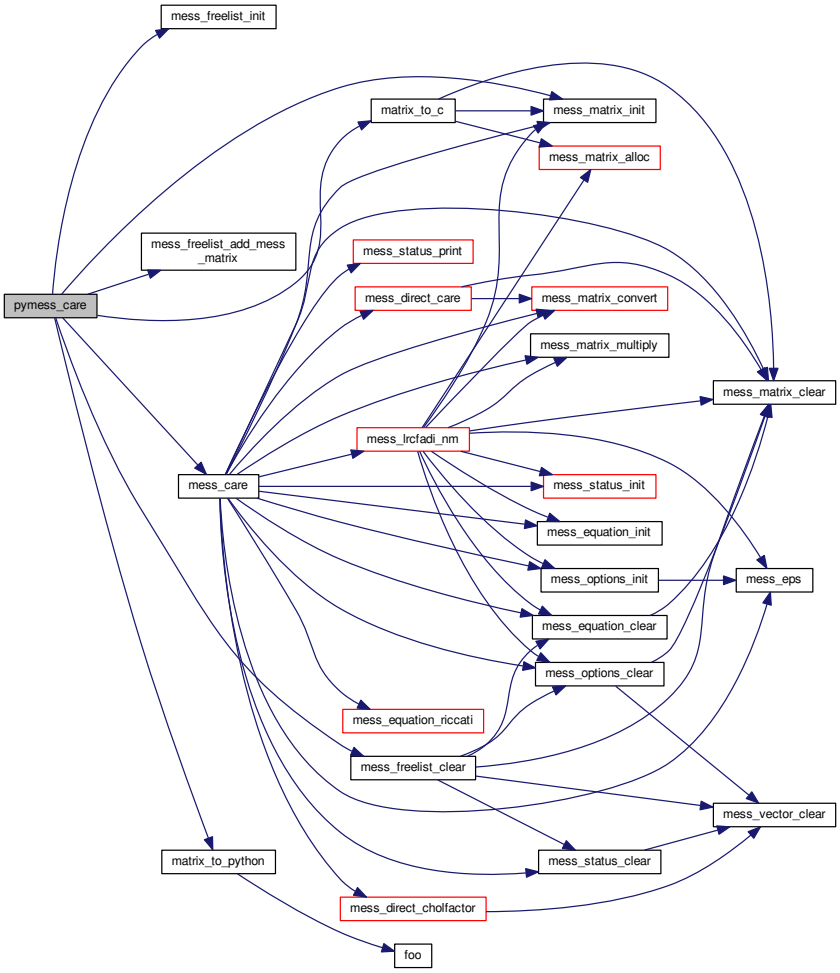
```
1  [py]
2 Z = pymess.care(a, None, b, c )
```

to solve the standard Riccati Equation $A^TX + TXA - XBB^TX + C^TC = 0$ and as

```
1  [py]
2 Z = pymess.care(a, e, b, c)
```

in case of the generalized Riccati Equation $A^TXE + E^TXA - E^TXBB^TXE + C^TC = 0$.

Here is the call graph for this function:



8.55.3.19 PyObject* pymess_dense_nm_gmpare (PyObject * *self*, PyObject * *args*)

Parameters

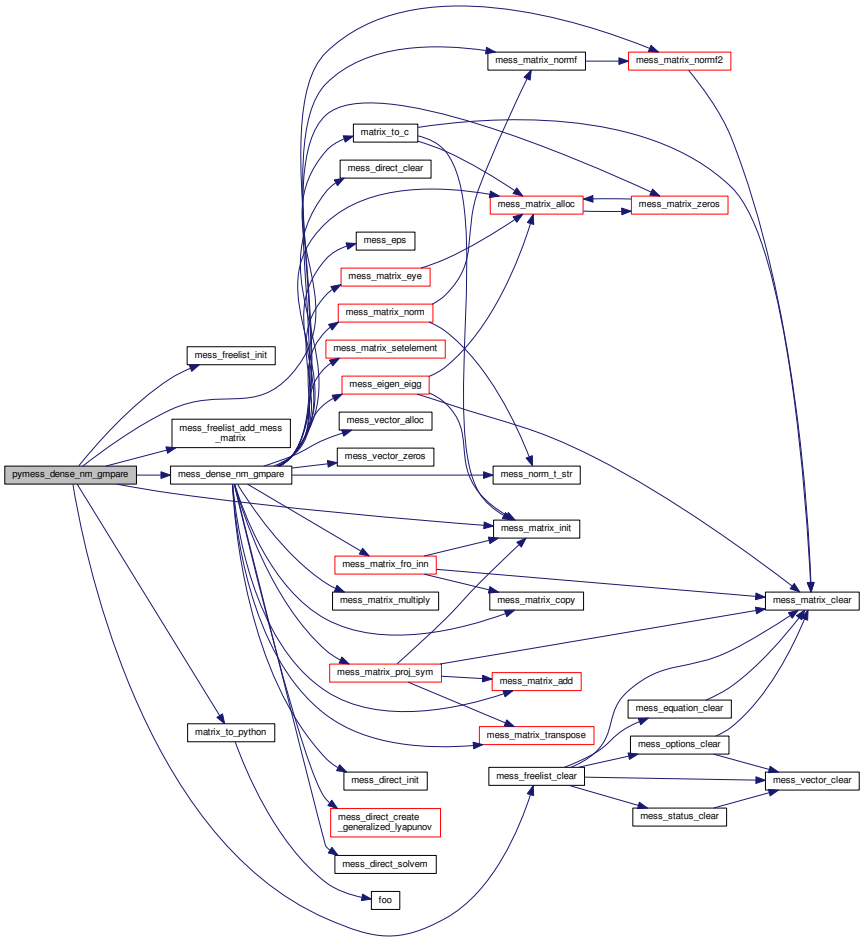
in	*self*	input
in	*args*	input

Returns

`Python` object containing X or None in case of an error / exception.

The pymess_dense_nm_gmpare function provides a wrapper around the mess_dense_nm_gmpare function in order to use from `Python`.

Here is the call graph for this function:



8.55.3.20 PyObject∗ pymess_direct_chol_select (PyObject ∗ *self,* PyObject ∗ *args*)

Parameters

in	*self*	input
in	*args*	input

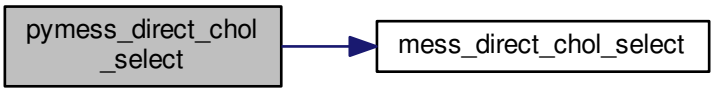
Returns

Python None.

The pymess_direct_chol_select function provides and wrapper arount the mess_direct_chol_select function in order to use from Python. The function is called as

```
[py]
Z = pymess.direct_chol_select( MESS_DIRECT_DEFAULTCHOL )
```

Here is the call graph for this function:



8.55.3.21 PyObject* pymess_direct_select (PyObject * *self*, PyObject * *args*)

Parameters

in	*self*	input
in	*args*	input

Returns

Python None.

The pymess_direct_select function provides and wrapper arount the mess_direct_lu_select function in order to use from Python. The function is called as

```
[py]
Z = pymess.direct_select( MESS_DIRECT_UMFPACK_LU )
```

to use UMFPACK as a direct solver.

Here is the call graph for this function:

8.55.3.22 PyObject* pymess_eps (PyObject * *self*, PyObject * *args*)

Here is the call graph for this function:

8.55.3.23 PyObject* pymess_glyap (PyObject * *self*, PyObject * *args*)

Parameters

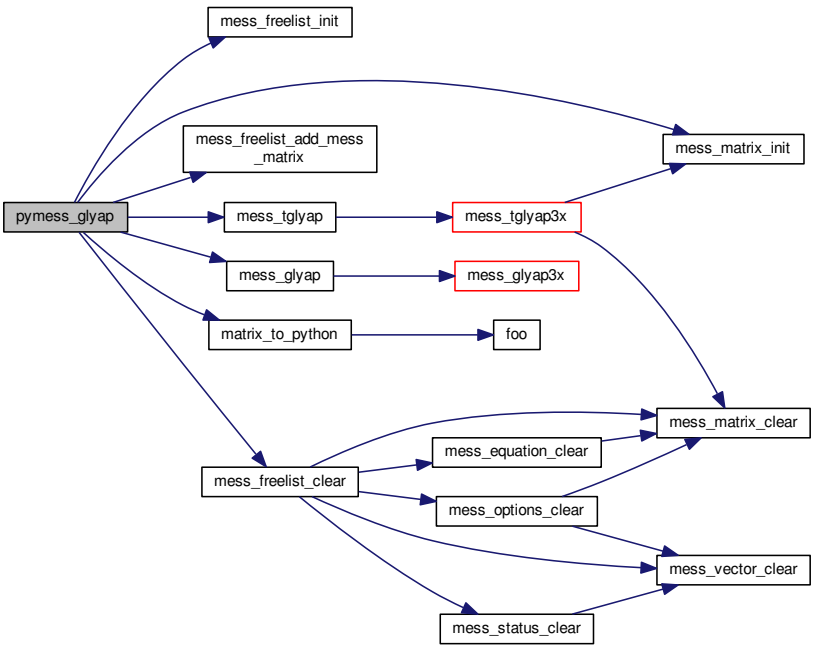
in	*self*	input
in	*args*	input

Returns

`Python` object containing X and optional the schur decomposition or None in case of an error / exception.

The pymess_glyap function provides an interface to mess_glyap function in order to use from `Python`. The function is called as

Here is the call graph for this function:



8.55.3.24 PyObject* pymess_gstein (PyObject * *self*, PyObject * *args*)

Parameters

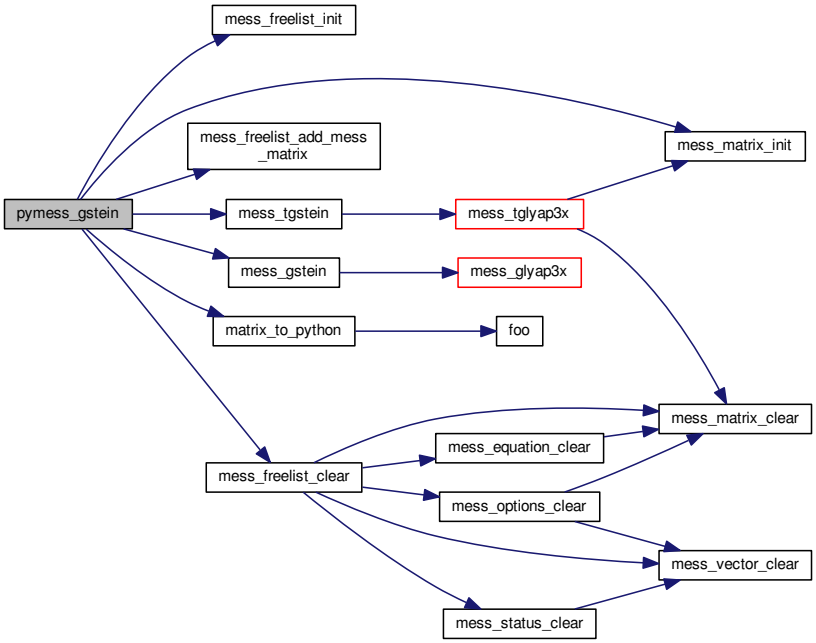
in	*self*	input
in	*args*	input

Returns

`Python` object containing X and optional the schur decomposition or None in case of an error / exception.

The pymess_gstein function provides an interface to mess_gstein function in order to use from `Python`. The function is called as

Here is the call graph for this function:



8.55.3.25 int pymess_is_reimplemented (PyObject * *base*, PyObject * *instance*, const char * *attr*)

Parameters

in	*base*	input PyObject containing the base class
in	*instance*	input Instance of a object
in	*attr*	input Name of the attribute to check

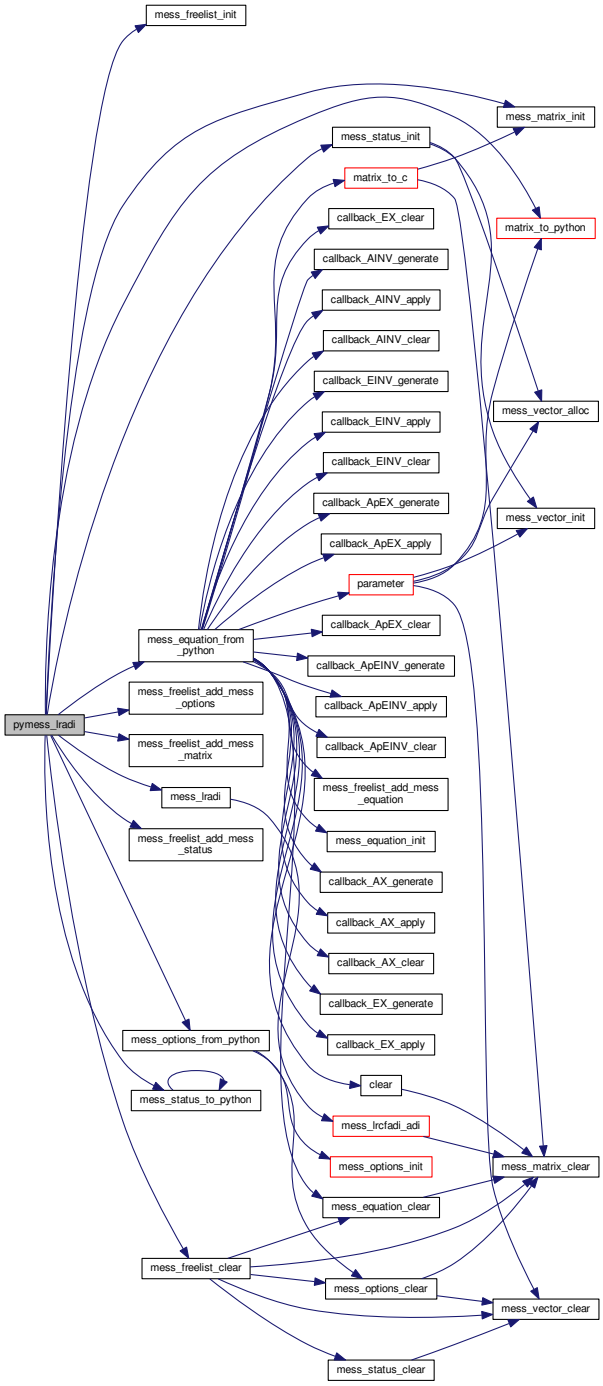
Returns

-1 in case of an error, 0 if the function only exists in the base class, or one if it is reimplemented.

The pymess_is_reimplemented function checks if a given instance reimplements the function named by "attr" or if the one implemented in the base class is used.

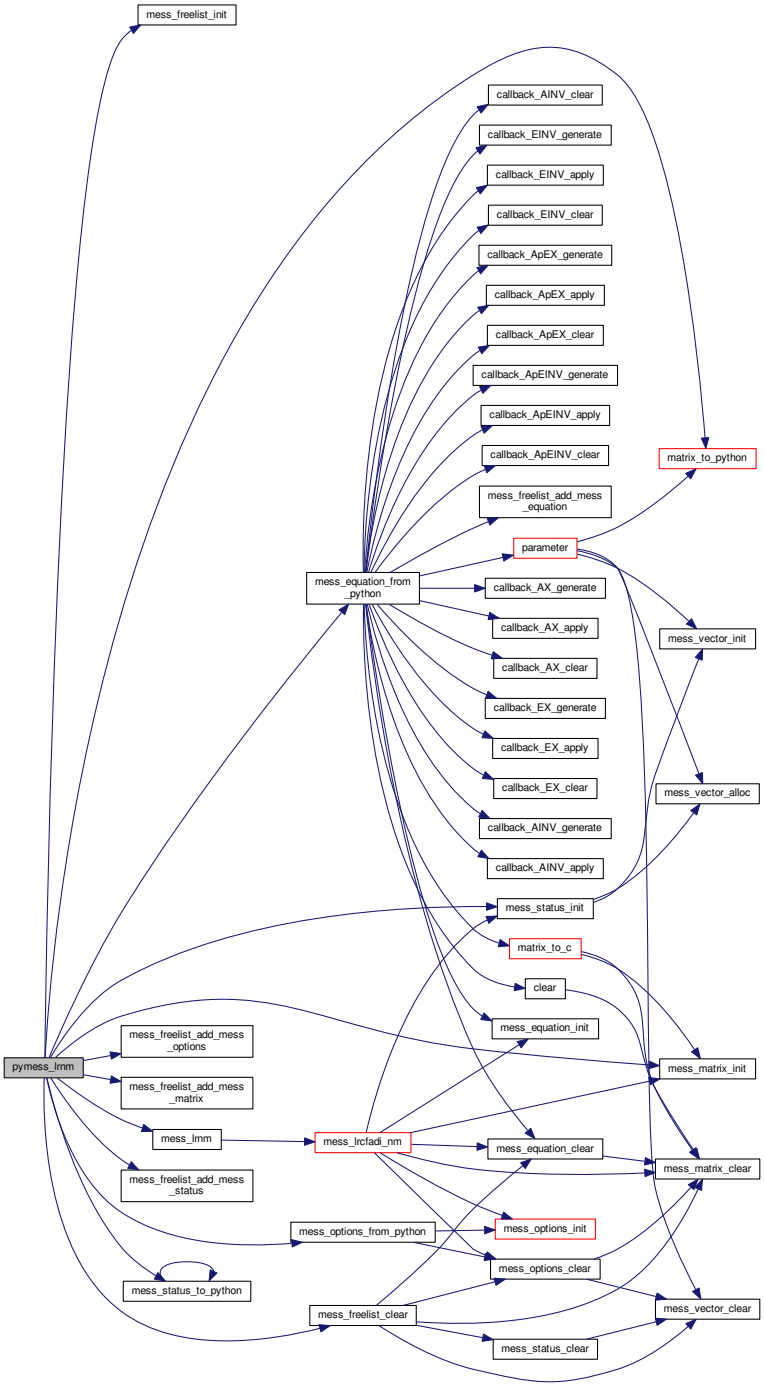
8.55.3.26 PyObject∗ pymess_lradi (PyObject ∗ *self,* PyObject ∗ *args*)

Here is the call graph for this function:



8.55.3.27 PyObject* pymess_lrnm (PyObject * *self*, PyObject * *args*)

Here is the call graph for this function:



8.55.3.28 PyObject* pymess_lyap (PyObject * *self*, PyObject * *args*)

Parameters

in	*self*	input
in	*args*	input

Returns

Python object containing Z or None in case of an error and an exception.

The pymess_lyap function provides and wrapper arount the mess_lyap function in order to use from Python. The function is called as

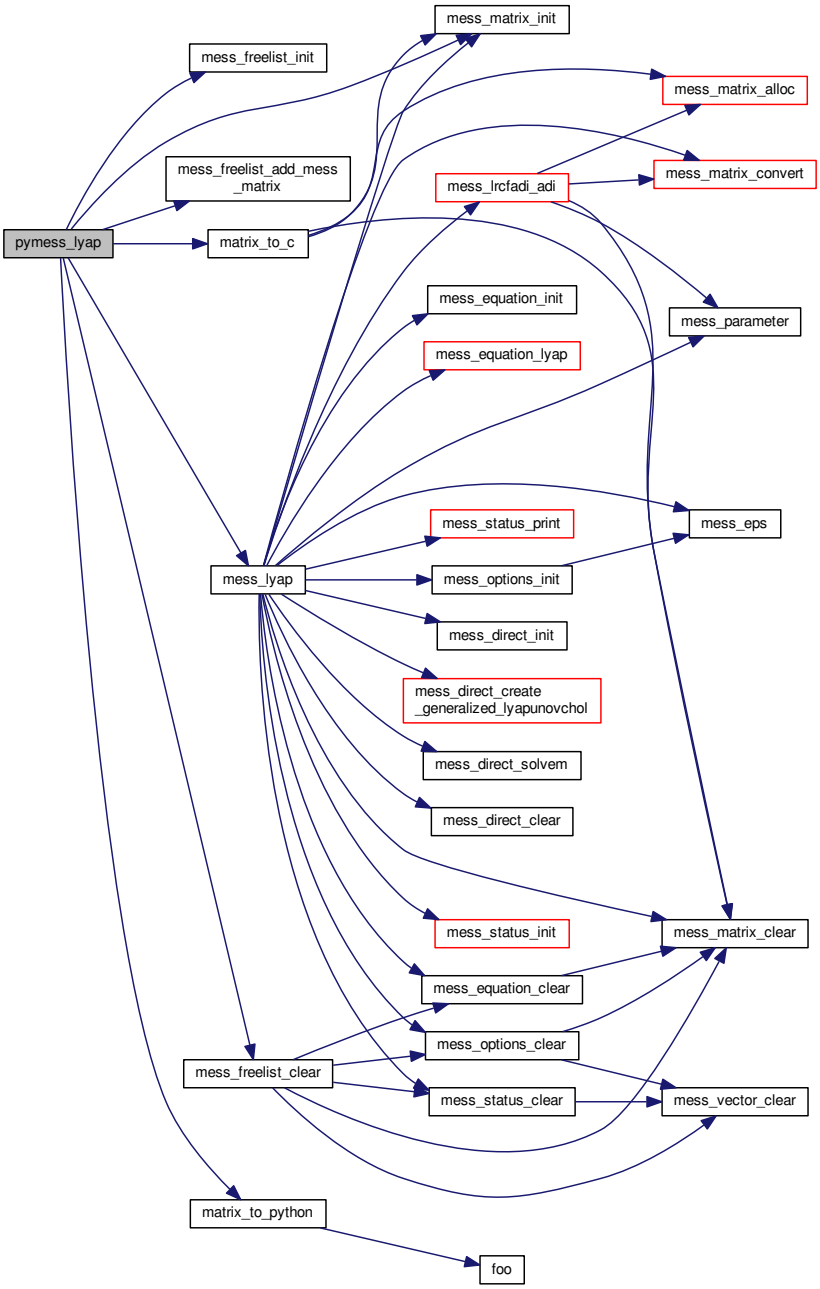
```
[py]
Z = pymess.lyap(a, None, b)
```

to solve the standard Lyapunov Equation $AX + XA^T + BB^T = 0$ and as

```
[py]
Z = pymess.lyap(a, e ,b )
```

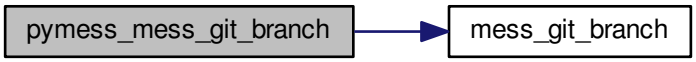
in case of the generalized Lyapunov Equation $AXE^T + EXA^T + BB^T = 0$.

Here is the call graph for this function:



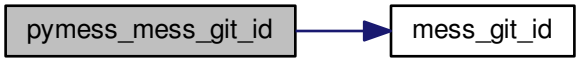
8.55.3.29 PyObject* pymess_mess_git_branch (PyObject * *self*, PyObject * *args*)

Here is the call graph for this function:



8.55.3.30 PyObject* pymess_mess_git_id (PyObject * *self*, PyObject * *args*)

Here is the call graph for this function:



8.55.3.31 PyObject* pymess_mess_have_amd (PyObject * *self*, PyObject * *args*)

8.55.3.32 PyObject* pymess_mess_have_arpack (PyObject * *self*, PyObject * *args*)

8.55.3.33 PyObject* pymess_mess_have_bzip2 (PyObject * *self*, PyObject * *args*)

8.55.3.34 PyObject* pymess_mess_have_cholmod (PyObject * *self*, PyObject * *args*)

8.55.3.35 PyObject* pymess_mess_have_colamd (PyObject * *self*, PyObject * *args*)

8.55.3.36 PyObject* pymess_mess_have_csparse (PyObject * *self*, PyObject * *args*)

8.55.3.37 PyObject* pymess_mess_have_matio (PyObject * *self*, PyObject * *args*)

8.55.3.38 PyObject* pymess_mess_have_mess64 (PyObject * *self*, PyObject * *args*)

8.55.3.39 PyObject* pymess_mess_have_mklpardiso (PyObject * *self*, PyObject * *args*)

8.55.3.40 PyObject* pymess_mess_have_openmp (PyObject * *self*, PyObject * *args*)

8.55.3.41 PyObject* pymess_mess_have_superlu (PyObject * *self*, PyObject * *args*)

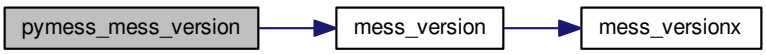
8.55.3.42 PyObject* pymess_mess_have_umfpack (PyObject * *self*, PyObject * *args*)

8.55.3.43 PyObject* pymess_mess_have_zlib (PyObject * *self*, PyObject * *args*)

8.55.3.44 PyObject* pymess_mess_is_debug (PyObject * *self*, PyObject * *args*)

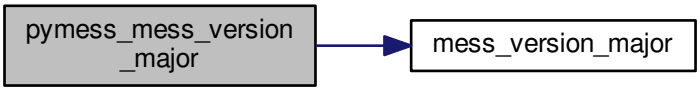
8.55.3.45 PyObject* pymess_mess_version (PyObject * *self*, PyObject * *args*)

Here is the call graph for this function:



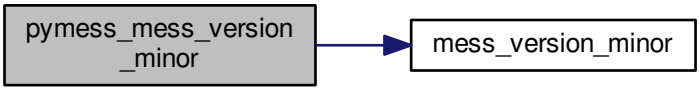
8.55.3.46 PyObject* pymess_mess_version_major (PyObject * *self*, PyObject * *args*)

Here is the call graph for this function:



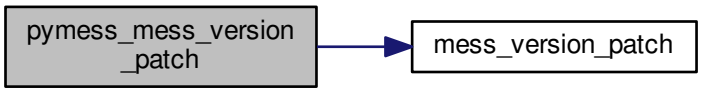
8.55.3.47 PyObject* pymess_mess_version_minor (PyObject * *self*, PyObject * *args*)

Here is the call graph for this function:



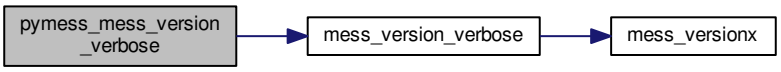
8.55.3.48 PyObject* pymess_mess_version_patch (PyObject * *self*, PyObject * *args*)

Here is the call graph for this function:



8.55.3.49 PyObject* pymess_mess_version_verbose (PyObject * *self*, PyObject * *args*)

Here is the call graph for this function:



8.55.3.50 PyObject* pymess_multidirect_select (PyObject * *self*, PyObject * *args*)

Parameters

in	*self*	input
in	*args*	input

Returns

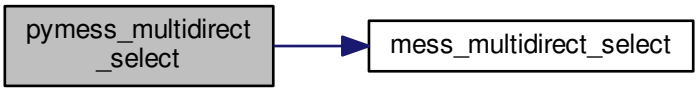
`Python` None.

The pymess_multidirect_select function provides and wrapper arount the mess_multidirect_select function in order to use from `Python`. The function is called as

```
1  [py]
2 Z = pymess.multidirect_select( MESS_MULTIDIRECT_UMFPACK_LU )
```

to use `UMFPACK` as a multi direct solver.

Here is the call graph for this function:



8.55.3.51 PyObject* pymess_set_errorlevel (PyObject * *self*, PyObject * *args*)

Parameters

in	*self*	input
in	*args*	input

Returns

Python None.

The pymess_set_errorlevel function provides and wrapper arount the mess_set_errorlevel function in order to use from Python. The function is called as

```
[py]
pymess.set_errorlevel(0)
pymess.set_errorlevel(1)
pymess.set_errorlevel(2)
pymess.set_errorlevel(3)
```

Here is the call graph for this function:

8.55.3.52 PyObject* pymess_sylvester_sparsedense (PyObject * *self*, PyObject * *args*)

Parameters

in	*self*	input
in	*args*	input

Returns

Python object containing X or None in case of an error / exception.

The pymess_sylvester_sparsedense function provides a wrapper around the mess_direct_create_sylvester_↩ sparsedense function in order to use from Python. The function is called as

```
[py]
Z = pymess.sylvester_sparsedense(a, None, None, b, m)
```

to solve the standard Sylvester Equation $AX + XB + M = 0$, as

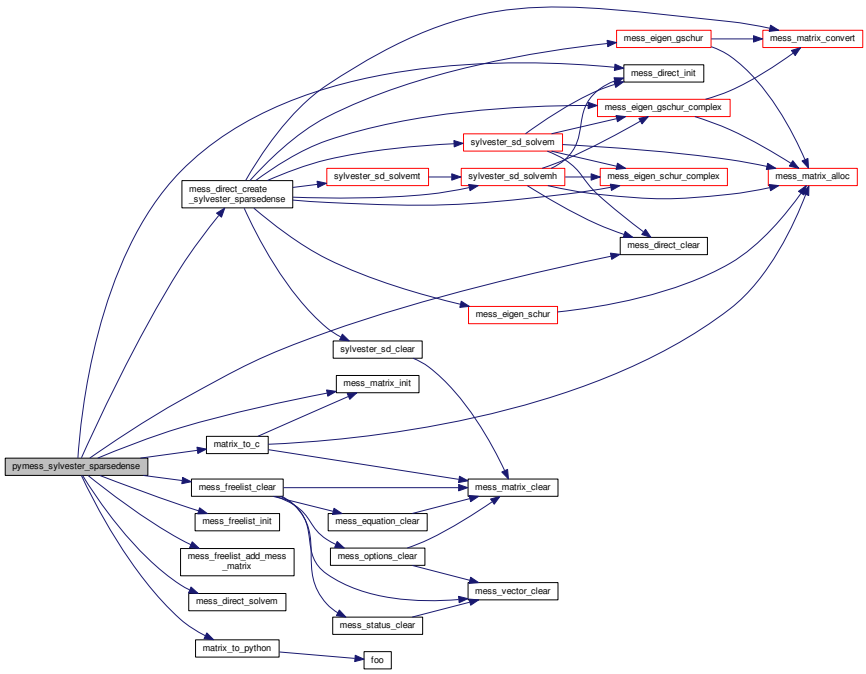
```
[py]
Z = pymess.sylvester_sparsedense(a, None, e, b, m)
```

to solve the semi-generalized Sylvester Equation $AX + EXB + M = 0$ and as

```
[py]
Z = pymess.sylvester_sparsedense(a, f, e, b, m)
```

to solve the generalized Sylvester Equation $AXF + EXB + M = 0$.

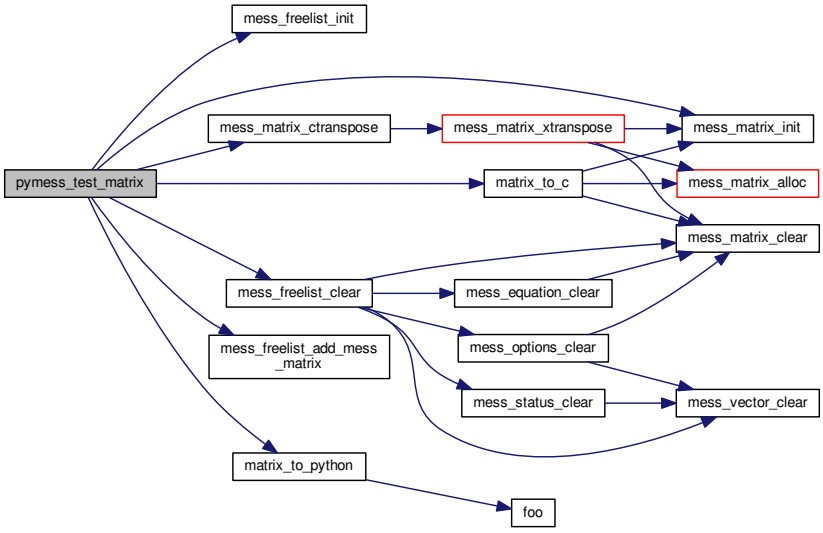
Here is the call graph for this function:



8.55.3.53 PyObject* pymess_test_equation (PyObject * *self*, PyObject * *args*)

8.55.3.54 PyObject* pymess_test_matrix (PyObject * *self*, PyObject * *args*)

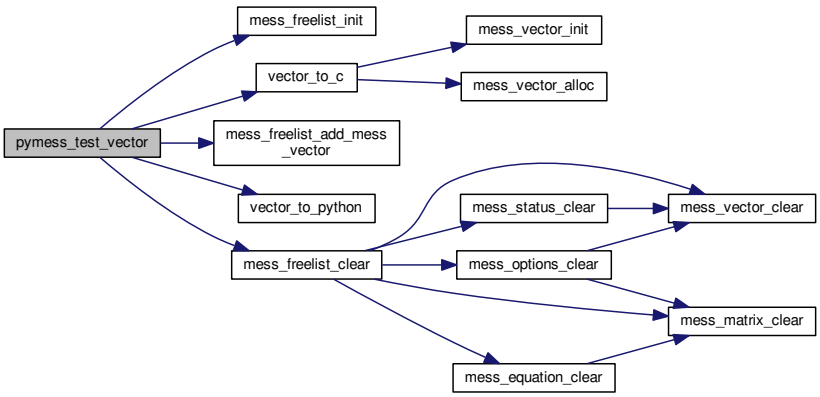
Here is the call graph for this function:



8.55.3.55 PyObject* pymess_test_status (PyObject * *self*, PyObject * *args*)

8.55.3.56 PyObject* pymess_test_vector (PyObject * *self*, PyObject * *args*)

Here is the call graph for this function:



8.55.3.57 mess_vector vector_to_c (PyObject * *data*)

Parameters

in	*data*	input Python vector

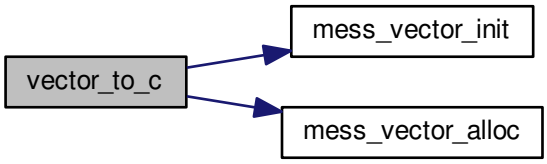
Returns

converted vector as mess_vector

The vector_to_c function supports the numpy ndarray and the following datatypes

- int8
- int16
- int32
- int64
- float32
- float64
- complex64
- complex128

Here is the call graph for this function:



8.55.3.58 PyObject* vector_to_python (mess_vector *c_vector*, int *copy*)

Parameters

in	*c_vector*	input mess_vector
in	*copy*	input set to true of the vector should be copied to Python

Returns

vector as Python Object

The vector_to_python function converts a mess_vector a numpy one. If the copy argument is true the data is copied otherwise move to Python.

8.55.4 Variable Documentation

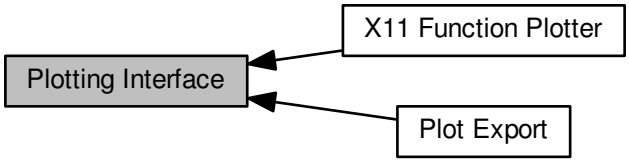
8.55.4.1 struct module_state _state [static]

8.55.4.2 PyMethodDef interface_python_methods[ ] [static]

8.56 Plotting Interface

This section includes categories defining a function plot.

Collaboration diagram for Plotting Interface:



Modules

- X11 Function Plotter

  This category contains data structures, type definitions and functions for a function plot based on X11.
- Plot Export

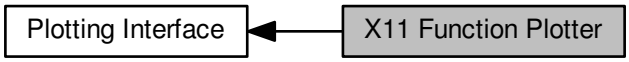
  This category contains data structures, type definitions and functions for a function plot.

8.56.1 Detailed Description

8.57 X11 Function Plotter

This category contains data structures, type definitions and functions for a function plot based on X11.

Collaboration diagram for X11 Function Plotter:



Data Structures

- struct __mess_plotdata

  Structure for one data element in a plot.
- struct __mess_plotter

  Structure to hold a complete plotter.

Typedefs

- typedef struct __mess_plotdata mess_plotdata

  Structure for one data element in a plot.
- typedef struct __mess_plotter * mess_plot

  Type definition for the plotter structure.

Functions

- int mess_plot_create (mess_plot *p, mess_int_t width, mess_int_t height, char *title, int xscale, int yscale, int data)

  Create a plotting window.
- int mess_plot_addData (mess_plot p, int data, double x, double y)

  Add new data values to a plot.
- int mess_plot_clearData (mess_plot p, int data)

  Clear all data of a data element in a plot.
- int mess_plot_setLabel (mess_plot p, int data, char *label)

  Set label of a data element.
- int mess_plot_setColor (mess_plot p, int data, char *color)

  Set color of a data element.
- int mess_plot_save (mess_plot p, char *filename)

  Save a plot to a xpm image.
- int mess_plot_update (mess_plot p)

  Update / redraw a plot.
- int mess_plot_close (mess_plot *p)

  Close a plotting window and clean up all internal data.

8.57.1 Detailed Description

It includes data structures representing a complete plot or one data element in a plot.
There are definitions for a linear or logarithmic scaled plot and type definitions for the structure for one data element in a plot and for the use of the plotter structure.
Additionally this category contains several functions concerning a plot, e.g.

- adding new data values to a plot
- clearing all data of a data element in a plot
- creating a plotting window
- saving or updating a plot
- setting the label or colour of a data element.

This module provides a small and easy to use values plotter based on the X11 library. It can be used to create linear and logarithmic plots.
An example to plot sine and cosine will be:

```
@ref mess_plot plot;
@ref mess_int_t i, N=1000;
double x,y1,y2;
double h = 2.0*3.14159/N;
@ref mess_plot_create(&plot, 400, 300, "Test-Plot",
      MESS_PLOT_LIN, MESS_PLOT_LIN, 2);
@ref mess_plot_setLabel(plot, 0, "sin(x)");
@ref mess_plot_setColor(plot, 0, "Red");
@ref mess_plot_setLabel(plot, 1, "cosine(x)");
@ref mess_plot_setColor(plot, 1, "Blue");
for ( i = 0; i <= N; i++){
 x = h*i;
 y1 = sin(x);
 y2 = cos(x);
 @ref mess_plot_addData(plot,0, x,y1 );
 @ref mess_plot_addData(plot,1, x,y2 );
}
@ref mess_plot_update(plot);
sleep(10);
@ref mess_plot_close(&plot);
```

8.57.2 Typedef Documentation

8.57.2.1 typedef struct __mess_plotter* mess_plot

The mess_plot type definition is for the plotter structure to use in programms.

8.57.2.2 typedef struct __mess_plotdata mess_plotdata

The mess_plotdata structure holds all information of a data element in a plot.

8.57.3 Function Documentation

8.57.3.1 int mess_plot_addData (mess_plot *p*, int *data*, double *x*, double *y*)

Parameters

in,out	*p*	plot to add the data
in	*data*	input number of data
in	*x*	input x coordinate of data
in	*y*	input y coordinate of data

Returns

zero on success or a non-zero error value otherwise

The mess_plot_addData function adds a new value (x, y) to the data element with number $data$ and updates the graphics.

8.57.3.2 int mess_plot_clearData (mess_plot *p*, int *data*)

Parameters

in,out	*p*	plot with data
in	*data*	input number of the data element

Returns

zero on success or a non-zero error value otherwise

The mess_plot_clearData function clears all data in the data element with the number $data$.

8.57.3.3 int mess_plot_close (mess_plot ∗ *p*)

Parameters

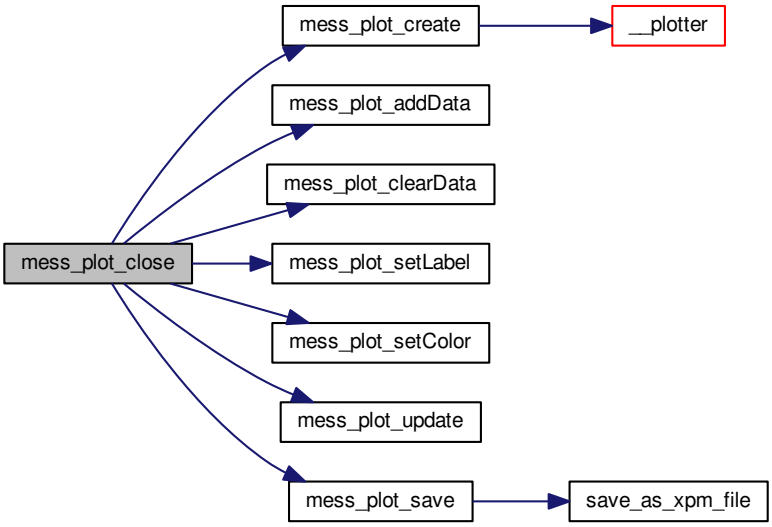
in,out	*p*	pointer to plot

Returns

zero on success or a non-zero error value otherwise

The mess_plot_close function closes a plot and destroys all internal data.

Here is the call graph for this function:



8.57.3.4 int mess_plot_create (mess_plot * *pl*, mess_int_t *width*, mess_int_t *height*, char * *title*, int *xscale*, int *yscale*, int *data*)

Parameters

in,out	*pl*	pointer to plot structure
in	*width*	input width of plot
in	*height*	input height of plot
in	*title*	input title of plotting window
in	*xscale*	input type of x scaling
in	*yscale*	input type of y scaling
in	*data*	input number of different data

Returns

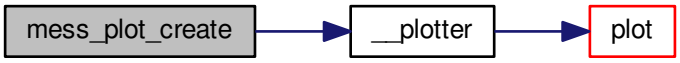
zero on success or a non-zero error value otherwise

The mess_plot_create function creates a new plotting window of the size $height \times width$ with $title$ as title.
The $xscale$ and $yscale$ arguments can be

- MESS_PLOT_LIN,
- MESS_PLOT_LOG

to specify if the scaling is linear or logarithmic.
The $data$ arguments specify the number of different data in the plot.

Here is the call graph for this function:



8.57.3.5 int mess_plot_save (mess_plot *p*, char * *filename*)

Parameters

in	*p*	input plot to save
in	*filename*	input filename of image

Returns

zero on success or a non-zero error value otherwise

The mess_plot_save function saves a given plot p to a file called $filename$ in a XPM image format.

Here is the call graph for this function:

8.57.3.6 int mess_plot_setColor (mess_plot *p*, int *data*, char * *color*)

Parameters

in,out	*p*	plot
in	*data*	input number of the data element
in	*color*	input colorname

Returns

zero on success or a non-zero error value otherwise

The mess_plot_setColor function sets the color of the data element with the number $data$.
Colorname can be one of all names your X11 implementation allow.

8.57.3.7 int mess_plot_setLabel (mess_plot *p*, int *data*, char * *label*)

Parameters

in,out	*p*	plot
in	*data*	input number of the data element
in	*label*	input label

Returns

zero on success or a non-zero error value otherwise

The mess_plot_setLabel function sets the label of the data element with the number $data$.
The label will be cut off after 40 characters.

8.57.3.8 int mess_plot_update (mess_plot *p*)

Parameters

in,out	*p*	plot

Returns

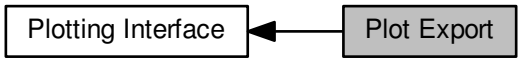
zero on success or a non-zero error value otherwise

The mess_plot_update function updates/redraws a given plot.

8.58 Plot Export

This category contains data structures, type definitions and functions for a function plot.

Collaboration diagram for Plot Export:



Data Structures

- struct __mess_plotExportData

  Structure for one data element in a plot.
- struct __mess_plotterScript

  Structure to hold a complete plotter.

Typedefs

- typedef struct __mess_plotExportData mess_plotExportData

  Structure for one data element in a plot.
- typedef struct __mess_plotterScript * mess_plotExport

  Type definition for the plotter structure.

Enumerations

Functions

- int mess_plotExport_init (mess_plotExport *p, char *title, mess_plot_axis_scale_t xscale, mess_plot_axis↩_scale_t yscale, char *xLabel, char *yLabel, int data)

  Initialize all data structures and variables.
- int mess_plotExport_addData (mess_plotExport p, int data, double x, double y)

  Add new data values to a plot.
- int mess_plotExport_clearData (mess_plotExport p, int data)

  Clear all data of a data element in a plot.
- int mess_plotExport_setLegendPos (mess_plotExport p, char *orientation)

  Set position of the legend.
- int mess_plotExport_setLabel (mess_plotExport p, int data, char *label)

  Set label of a data element.
- int mess_plotExport_setColor (mess_plotExport p, int data, char *color)

  Set the color of a plot.
- int mess_plotExport_setType (mess_plotExport p, int data, char *type)

  Set the style of a plot.
- int mess_plotExport_save (mess_plotExport p, const char *path)

Save all data to a .dat file.

- int mess_plotExport_createGnuScript (mess_plotExport p)

  Create a GNU plot script file.
- int mess_plotExport_createGnuScript_filename (mess_plotExport p, const char *path)

  Create a GNU plot script file.
- int mess_plotExport_createTikzScript (mess_plotExport p)

  Generate a TikZ-diagram and store it in a .tex-file.
- int mess_plotExport_createTikzScript_filename (mess_plotExport p, const char *path)

  Generate a TikZ-diagram and store it in a .tex-file.
- int mess_plotExport_clear (mess_plotExport *plot)

  Clear all data in a plot.

8.58.1 Detailed Description

It includes data structures representing a complete plot or one data element in a plot.
There are definitions for a linear or logarithmic scaled plot and type definitions for the structure for one data element in a plot and for the use of the plotter structure.
Additionally this category contains several functions concerning a plot, e.g.

- adding new data values to a plot
- basic functions for hashtables
- clearing all data of a data element or all information in a plot
- creating or closing a plotting window
- saving all data to a .dat file
- setting the label of a data element
- setting the position of a legend
- setting the style or colour of a plot.

This module provides a small and easy to use values plotter based on the X11 library. It can be used to create linear and logarithmic plots.
An example to plot sine and cosine will be:

```
@ref mess_plotScript plot;
@ref mess_int_t i, N=1000;
double x,y1,y2;
double h = 2.0*3.14159/N;
@ref mess_gnuplot_create(&plot, 400, 300, "Test-Plot", MESS_PLOT_LIN,
      MESS_PLOT_LIN, 2);
@ref mess_gnuplot_setlabel(plot, 0, "sin(x)");
@ref mess_gnuplot_setcolor(plot, 0, "Red");
@ref mess_gnuplot_setlabel(plot, 1, "cosine(x)");
@ref mess_gnuplot_setcolor(plot, 1, "Blue");
for ( i = 0; i <= N; i++){
 x = h*i;
 y1 = sin(x);
 y2 = cos(x);
 @ref mess_gnuplot_adddata(plot,0, x,y1 );
 @ref mess_gnuplot_adddata(plot,1, x,y2 );
}
@ref mess_gnuplot_createScript();
```

8.58.2 Typedef Documentation

8.58.2.1 typedef struct __mess_plotterScript* mess_plotExport

The mess_plotExport type definition is for the plotter structure to use in programms.

Attention

Internal use only.

8.58.2.2 typedef struct __mess_plotExportData mess_plotExportData

The mess_plotExportData structure holds all information of a data element in a plot.

Attention

Internal use only.

8.58.3 Enumeration Type Documentation

8.58.3.1 enum mess_plot_axis_scale_t

Enumerator

MESS_PLOT_LIN Linear axis scaling for plot.

MESS_PLOT_LOG Logarithmic axis scaling for plot.

8.58.4 Function Documentation

8.58.4.1 int mess_plotExport_addData (mess_plotExport *p*, int *data*, double *x*, double *y*)

Parameters

in	*p*	input plot to add data
in	*data*	input number of data
in	*x*	input x-coordinate of data
in	*y*	input y-coordinate of data

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_addData function adds a new value (x, y) to the data element with the number $data$ and updates the graphics.

8.58.4.2 int mess_plotExport_clear (mess_plotExport * *pl*)

Parameters

in,out	*pl*	plot with data

Returns

always zero

The mess_plotExport_clear function clears all information in the plot pl.

8.58.4.3 int mess_plotExport_clearData (mess_plotExport *p*, int *data*)

Parameters

in,out	*p*	plot with data
in	*data*	input number of data element

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_clearData function clears all data in a data element with number $data$ in a plot p.

8.58.4.4 int mess_plotExport_createGnuScript (mess_plotExport *p*)

Parameters

in	*p*	input plot

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_createGnuScript function will use all known data from the plot to create a script file. This script file will be saved as a _script.gnu file with the plots title as a prefix in a folder with the same title.

Here is the call graph for this function:

8.58.4.5 int mess_plotExport_createGnuScript_filename (mess_plotExport *p*, const char * *path*)

Parameters

in	*p*	input plot
in	*path*	input destination folder to store the GNU plot script

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_createGnuScript_filename function will use all known data from the plot to create a script file.
This script file will be saved as a _script.gnu file with the plots title as a prefix.

Here is the call graph for this function:

8.58.4.6 int mess_plotExport_createTikzScript (mess_plotExport *p*)

Parameters

in	*p*	input plot with data

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_createTikzScript function evaluates the information stored in a plot and creates a Tik↩Z-picture.
This will be saved as a .tex-file and can be included to a TeX-document.

Here is the call graph for this function:

8.58.4.7 int mess_plotExport_createTikzScript_filename (mess_plotExport *p*, const char * *path*)

Parameters

in	*p*	input plot with data
in	*path*	input destination folder to store the TikZ Script

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_createTikzScript_filename function evaluates the information stored in a plot and creates a TikZ-picture.
This will be saved as a .tex-file in the destination folder $path$ and can be included to a TeX-document.

Here is the call graph for this function:

8.58.4.8 int mess_plotExport_init (mess_plotExport ∗ *p*, char ∗ *title*, mess_plot_axis_scale_t *xscale*, mess_plot_axis_scale_t *yscale*, char ∗ *xLabel*, char ∗ *yLabel*, int *data*)

Parameters

in,out	*p*	plot
in	*title*	input plot title
in	*xscale*	input scale of the x axis (not used)
in	*yscale*	input scale of the y axis (not used)
in	*xLabel*	input label of the x axis
in	*yLabel*	input label of the y axis
in	*data*	input number of data elements

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_init function initializes all variables of a plot.

8.58.4.9 int mess_plotExport_save (mess_plotExport *p*, const char ∗ *path*)

Parameters

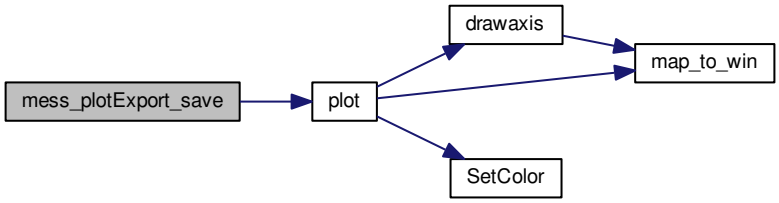
in	*p*	input plot
in	*path*	input destination folder

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_save function saves all data added to a .dat file

Here is the call graph for this function:



8.58.4.10 int mess_plotExport_setColor (mess_plotExport *p*, int *data*, char * *color*)

Parameters

in,out	*p*	plot
in	*data*	input number of the data element
in	*color*	input colorstyle to be used for this dataset

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_setColor function sets the color of the plot.
Color need to be given as a character colourname. Possible characters are:

- "blue"
- "green"
- "orange"
- "red"
- etc.

For more colornames look up 'show palette colornames'.

8.58.4.11 int mess_plotExport_setLabel (mess_plotExport *p*, int *data*, char * *label*)

Parameters

in,out	*p*	plot
in	*data*	input number of the data element
in	*label*	input the label

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_setLabel function sets the label of a data element with the number $data$.
The label will be cut off after 40 characters.

8.58.4.12 int mess_plotExport_setLegendPos (mess_plotExport *p*, char * *orientation*)

Parameters

in,out	*p*	plot to add legend
in	*orientation*	input orientation of the key-box used in the diagramm

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_setLegendPos function sets the position or orientation of the legend used in the diagramm.
Possible orientation values are

- "belowCenter",
- "bottomRight",
- "bottomLeft",
- "bottom",
- "left",
- "right",
- "topRight",
- "topLeft",
- "top".

8.58.4.13 int mess_plotExport_setType (mess_plotExport *p*, int *data*, char * *type*)

Parameters

in,out	*p*	plot
in	*data*	input number of the data element
in	*type*	input type of style to use for this dataset

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_setType function sets the type of the dataset.
Type need to be given as a character typename.Possible types are

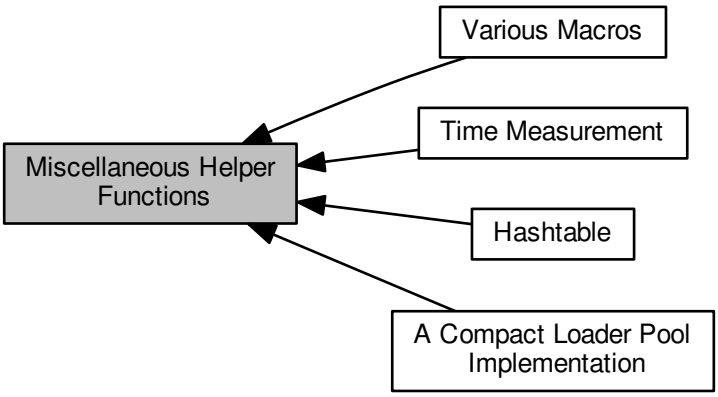
- "boxes",

- "dots",
- "impules",
- "lines" ,
- "points",
- "steps",
- etc.

8.59 Miscellaneous Helper Functions

This section includes several categories and some important functions.

Collaboration diagram for Miscellaneous Helper Functions:



Modules

- Hashtable

  This category contains datastructures, type definitions and functions to handle a hashtable.
- A Compact Loader Pool Implementation

  This category contains several structures, type definitions and functions concerning a threadpool.
- Time Measurement

  This category contains several time measurement functions.
- Various Macros

  This category contains several macros and definitions.

Data Structures

- struct mess_freelist_st

  Structure containing a minimal memory manage in order to avoid memory leaks in the `Python` *and* ***MEX-M.E.S.S.*** *interface.*

Macros

- #define isfinite(x) (sizeof(x) == sizeof(float))?__finitef(x):(sizeof(x)==sizeof(double))?__finite(x):__finitel(x)

Typedefs

- typedef struct mess_freelist_st * mess_freelist

  Structure containing a minimal memory manage in order to avoid memory leaks in the `Python` *and* ***MEX-M.E.S.S.*** *interface.*

Functions

- int mess_freelist_init (mess_freelist *list)

  Initialized a mess_freelist instance.
- int mess_freelist_print (mess_freelist list)

  Print a mess_freelist instance.
- int mess_freelist_add_ptr (mess_freelist list, void *ptr)

  Insert an arbitrary pointer in the memory manager.
- int mess_freelist_add_mess_vector (mess_freelist list, mess_vector vec)

  Insert a mess_vector in the memory manager.
- int mess_freelist_add_mess_matrix (mess_freelist list, mess_matrix mat)

  Insert a mess_matrix in the memory manager.
- int mess_freelist_add_mess_equation (mess_freelist list, mess_equation eqn)

  Insert a mess_equation in the memory manager.
- int mess_freelist_add_mess_options (mess_freelist list, mess_options opt)

  Insert a mess_options structure in the memory manager.
- int mess_freelist_add_mess_status (mess_freelist list, mess_status stat)

  Insert a mess_status structure in the memory manager.
- int mess_freelist_clear (mess_freelist *list)

  Free all objects stored in the memory manager.
- int mess_init (void)

  *Initialize **M.E.S.S.**.*
- int mess_exit (void)

  *Clean up some initial **M.E.S.S.** structures.*
- double mess_eps (void)
- int mess_int_t_size (void)

  Return size of mess_int_t.
- void mess_version (void)

  *Print **M.E.S.S.** version.*
- void mess_version_verbose (void)

  *Print **M.E.S.S.** version information.*
- int mess_version_major (void)
- int mess_version_minor (void)
- int mess_version_patch (void)
- const char * mess_git_id (void)

  *Return the git id which was used to configure the **M.E.S.S.** build.*
- const char * mess_git_branch (void)

  *Return the name of the git branch which was used to configure the **M.E.S.S.** build.*
- void mess_openmp_info (void)

  Display information about OpenMP.
- int mess_is_debug (void)
- int mess_have_zlib (void)
- int mess_have_bzip2 (void)
- int mess_have_umfpack (void)
- int mess_have_amd (void)
- int mess_have_colamd (void)
- int mess_have_cholmod (void)
- int mess_have_csparse (void)
- int mess_have_superlu (void)
- int mess_have_mklpardiso (void)
- int mess_have_arpack (void)
- int mess_have_matio (void)

- int mess_have_openmp (void)
- int mess_have_mess64 (void)
- const char * mess_datatype_t_str (const mess_datatype_t data_type)

  Return a human readable name of a data type value.
- const char * mess_storage_t_str (const mess_storage_t store_type)

  Return a human readable name of a storage type value.
- const char * mess_symmetry_t_str (const mess_symmetry_t symmetry)

  Return a human readable name of a symmetry type value.
- const char * mess_operation_t_str (const mess_operation_t op)

  Return the human readable name of an operation type value.
- const char * mess_equation_t_str (const mess_equation_t eqn)

  Convert the mess_equation_t to a human readable string.
- const char * mess_parameter_t_str (const mess_parameter_t para)

  Convert the mess_parameter_t to a human readable string.
- const char * mess_memusage_t_str (const mess_memusage_t mem)

  Convert the mess_memusage_t to a human readable string.
- const char * mess_norm_t_str (const mess_norm_t nrm)

  Convert the mess_norm_t to a human readable string.
- const char * mess_sign_scale_t_str (const mess_sign_scale_t sc_type)

  Convert the mess_sign_scale_t to a human readable string.
- const char * mess_direct_lupackage_t_str (const mess_direct_lupackage_t lu_type)

  Convert the mess_direct_lupackage_t to a human readable string.
- const char * mess_direct_cholpackage_t_str (const mess_direct_cholpackage_t chol_type)

  Convert the mess_direct_cholpackage_t to a human readable string.
- const char * mess_multidirect_t_str (const mess_multidirect_t ml_type)

  Convert the mess_multidirect_t to a human readable string.
- const char * mess_residual_t_str (const mess_residual_t res_type)

  Convert the mess_residual_t to a human readable string.
- int mess_print_format_select (mess_print_format_t print_type)

  Change the default behaviour of printing.
- int mess_print_format_double (double d)

  Print a double value using MSG_PRINT.
- int mess_print_format_double_cpx (mess_double_cpx_t cpx)

  Print a mess_double_cpx_t value using MSG_PRINT.
- void mess_print_bytes (mess_int_t size)

  Print a memory size.
- void * __mess_malloc (size_t size)

  Wrapper around malloc.
- void * __mess_realloc (void *ptr, size_t newsize)

  Wrapper around realloc.
- void * __mess_calloc (size_t nmemb, size_t size)

  Wrapper around calloc.
- void __mess_free (void *ptr)

  Wrapper around free.
- void mess_set_malloc (void *malloc_ptr, void *reallloc_ptr, void *calloc_ptr, void *free_ptr)

  Set the malloc implementation to be used.
- double nearbyint (double x)

  Round a value.
- int mess_matrix_st_offset (int *size, int *offsets, char **fieldnames)

  Information about memory laout of mess_matrix_st.
- int mess_vector_st_offset (int *size, int *offsets, char **fieldnames)

Information about memory laout of mess_vector_st.

- int mess_status_st_offset (int ∗size, int ∗offsets, char ∗∗fieldnames)

  Information about memory laout of mess_status_st.

- int mess_options_st_offset (int ∗size, int ∗offsets, char ∗∗fieldnames)

  Information about memory laout of mess_options_st.

- int mess_equation_st_offset (int ∗size, int ∗offsets, char ∗∗fieldnames)

  Information about memory laout of mess_equation_st.

- int mess_direct_st_offset (int ∗size, int ∗offsets, char ∗∗fieldnames)

  Information about memory laout of mess_direct_st.

8.59.1 Detailed Description

There are some crucial function like

- cleaning up the initial mess structures
- initializing mess

and some helpful functions, e.g.

- calculating machine precision
- converting a string to lower or upper case
- displaying intformation about the OpenMP environment
- getting a line from a file
- getting information about the CPU, memory and the swap space of the system
- printing memory size
- removing whitespaces from the beginning and the end of a string
- returning the readable name of a data, storage, symmetry or operation type value
- setting a new current temporary path and generating a random filename inside this path.

8.59.2 Macro Definition Documentation

8.59.2.1 #define isfinite(x) (sizeof(x) == sizeof(float))?__finitef(x):(sizeof(x)==sizeof(double))?__finite(x):__finitel(x)

8.59.3 Typedef Documentation

8.59.3.1 typedef struct mess_freelist_st ∗ mess_freelist

Due to the fact that C does not have a gargabe collector and **M.E.S.S.** does not implement reference counting we have to track the allocated data in some cases.

Attention

Interal use only.

8.59.4 Function Documentation

8.59.4.1 void∗ __mess_calloc (size_t *nmemb*, size_t *size*)

Parameters

in	*nmemb*	input Number of elements to allocate
in	*size*	input Size of one element

Returns

a pointer to the allocated memory location.

The __mess_calloc function is a configurable wrapper around calloc

8.59.4.2 void __mess_free (void * *ptr*)

Parameters

in	*ptr*	input Pointer to the memory which should be freed

The __mess_free function is a configurable wrapper around free.

8.59.4.3 void* __mess_malloc (size_t *size*)

Parameters

in	*size*	input size of the required memory location

Returns

a pointer to the allocated memory

The __mess_malloc function is a configurable wrapper around malloc.

8.59.4.4 void* __mess_realloc (void * *ptr*, size_t *newsize*)

Parameters

in	*ptr*	input Old pointer to the memory location
in	*newsize*	input New size of the memory location

Returns

a pointer to the new resized memory location

The __mess_realloc function is a configurable wrapper around realloc.

8.59.4.5 const char* mess_datatype_t_str (const mess_datatype_t *data_type*)

Parameters

in	*data_type*	input data type value

Returns

constant pointer to a human readable string containing the name of the data type

The mess_datatype_t_str function translates a data type constant to the corresponding string and returns a constant pointer to this string.

See also

mess_datatype_t

8.59.4.6 const char* mess_direct_cholpackage_t_str (const mess_direct_cholpackage_t *chol_type*)

Parameters

in	*chol_type*	type of direct solver

Returns

human readable string of `chol_type`

The mess_direct_cholpackage_t_str function converts a mess_direct_cholpackage_t to a human readable string.

8.59.4.7 const char* mess_direct_lupackage_t_str (const mess_direct_lupackage_t *lu_type*)

Parameters

in	*lu_type*	type of direct solver

Returns

human readable string of `lu_type`

The mess_direct_lupackage_t_str function converts a mess_direct_lupackage_t to a human readable string.

8.59.4.8 int mess_direct_st_offset (int * *size*, int * *offsets*, char ** *fieldnames*)

Parameters

in,out	*size*	input/output contains `sizeof` value
in,out	*offsets*	input/output contains relative address of fields of mess_direct_st
in,out	*fieldnames*	input/outout contains correspondig field names of
	offsets	

Returns

number of fields of mess_direct_st

The mess_direct_st_offset function returns informations about the memory layout of mess_direct_st. We assume that `offsets` and `fieldnames` points to preallocated memory. You can call the function also with `NULL` pointers, in this case the pointer is not used.

Here is the call graph for this function:

8.59.4.9 double mess_eps (void)

8.59.4.10 int mess_equation_st_offset (int * *size*, int * *offsets*, char ** *fieldnames*)

Parameters

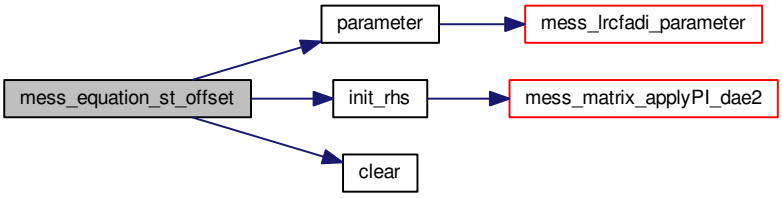
in,out	*size*	input/output contains `sizeof` value
in,out	*offsets*	input/output contains relative address of fields of mess_equation_st
in,out	*fieldnames*	input/outout contains correspondig field names of
	offsets	

Returns

number of fields of mess_equation_st

The mess_equation_st_offset function returns informations about the memory layout of mess_equation_st. We assume that `offsets` and `fieldnames` points to preallocated memory. You can call the function also with `NULL` pointers, in this case the pointer is not used.

Here is the call graph for this function:



8.59.4.11 const char* mess_equation_t_str (const mess_equation_t *eqn_type*)

Parameters

in	*eqn_type*	equation type

Returns

human readable string of `eqn_type`

The mess_equation_t_str function converts a mess_equation_t to a human readable string.

8.59.4.12 int mess_exit (void)

The mess_exit function clears the configuration.
Normaly you do not have to run it.

8.59.4.13 int mess_freelist_add_mess_equation (mess_freelist *list*, mess_equation *eqn*)

Parameters

in,out	*list*	list containing the allocated objects
in	*eqn*	equation to add to `list`

The mess_freelist_add_mess_equation function inserts `eqn` in `list`.

8.59.4.14 int mess_freelist_add_mess_matrix (mess_freelist *list*, mess_matrix *mat*)

Parameters

in,out	*list*	list containing the allocated objects
in	*mat*	matrix to add to `list`

The mess_freelist_add_mess_matrix function inserts `mat` in `list`.

8.59.4.15 int mess_freelist_add_mess_options (mess_freelist *list*, mess_options *opt*)

Parameters

in,out	*list*	List containing the allocated objects
in	*opt*	options to add to the list

The mess_freelist_add_mess_options function inserts a mess_status instance in the list of allocated objects.

8.59.4.16 int mess_freelist_add_mess_status (mess_freelist *list*, mess_status *stat*)

Parameters

in,out	*list*	List containing the allocated objects
in	*stat*	status to add to the `list`

The mess_freelist_add_mess_status function inserts `stat` in `list`.

8.59.4.17 int mess_freelist_add_mess_vector (mess_freelist *list*, mess_vector *vec*)

Parameters

in,out	*list*	list containing the allocated objects
in	*vec*	vector to add to the `list`

The mess_freelist_add_mess_vector function inserts `vec` in `list`.

8.59.4.18 int mess_freelist_add_ptr (mess_freelist *list*, void * *ptr*)

Parameters

in,out	*list*	List containing the allocated objects
in	*ptr*	pointer to add to the `list`

The mess_freelist_add_ptr function inserts an arbitrary pointer in the list of allocated objects.

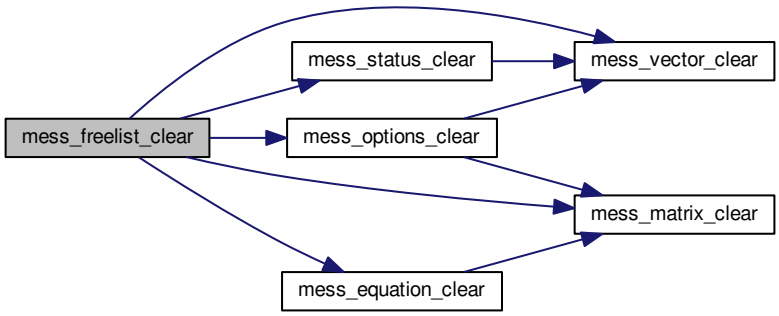
8.59.4.19 int mess_freelist_clear (mess_freelist * *list*)

Parameters

in	*list*	input list of objects to free

The mess_freelist_clear function removes all objetcs in the list from memory.

Here is the call graph for this function:



8.59.4.20 int mess_freelist_init (mess_freelist * *list*)

Parameters

in,out	*list*	allocated mess_freelist instance.

Initialize a mess_freelist instance.

8.59.4.21 int mess_freelist_print (mess_freelist *list*)

Parameters

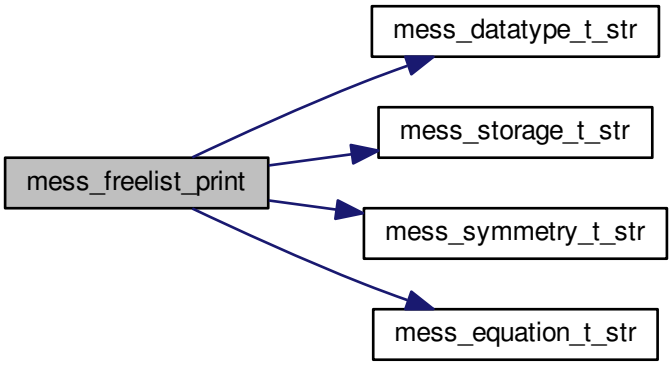
in	*list*	the mess_freelist instance.s

Returns

0.

Print content of mess_freelist instance `list`.

Here is the call graph for this function:



8.59.4.22 const char* mess_git_branch (void)

Returns

name of git branch.

The mess_git_branch returns the name of the git branch which was used to configure the **M.E.S.S.** build.

8.59.4.23 const char* mess_git_id (void)

Returns

git id SHA-1 hash value of length 40.

The mess_git_id returns the git id (SHA-1 hash value of length 40) which was used to configure the **M.E.S.S.** build.

See also

mess_git_branch

8.59.4.24 int mess_have_amd (void)

8.59.4.25 int mess_have_arpack (void)

8.59.4.26 int mess_have_bzip2 (void)

8.59.4.27 int mess_have_cholmod (void)

8.59.4.28 int mess_have_colamd (void)

8.59.4.29 int mess_have_csparse (void)

8.59.4.30 int mess_have_matio (void)

8.59.4.31 int mess_have_mess64 (void)

8.59.4.32 int mess_have_mklpardiso (void)

8.59.4.33 int mess_have_openmp (void)

8.59.4.34 int mess_have_superlu (void)

8.59.4.35 int mess_have_umfpack (void)

8.59.4.36 int mess_have_zlib (void)

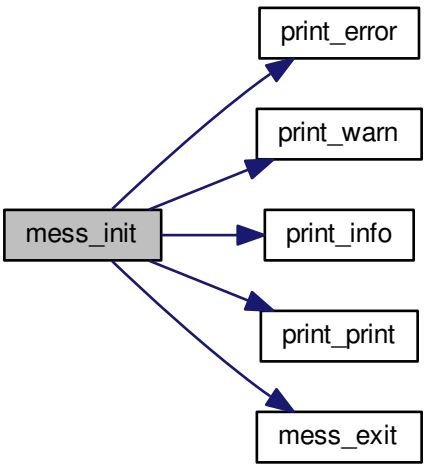
8.59.4.37 int mess_init (void)

Returns

zero on success or a non-zero error value otherwise

The mess_init function intializes **M.E.S.S.**.
You have only to call it, if you want to use the dynamic solver registration in a static library.

Here is the call graph for this function:



8.59.4.38 int mess_int_t_size (void)

The mess_int_t_size returns the size of mess_int_t datatype.

8.59.4.39 int mess_is_debug (void)

8.59.4.40 int mess_matrix_st_offset (int * *size*, int * *offsets*, char ** *fieldnames*)

Parameters

in,out	*size*	input/output contains `sizeof` value
in,out	*offsets*	input/output contains relative address of fields of mess_matrix_st
in,out	*fieldnames*	input/outout contains correspondig field names of
	offsets	

Returns

number of fields of mess_matrix_st

The mess_matrix_st_offset function returns informations about the memory layout of mess_matrix_st. We assume that `offsets` and `fieldnames` points to preallocated memory. You can call the function also with `NULL` pointers, in this case the pointer is not used.

8.59.4.41 const char* mess_memusage_t_str (const mess_memusage_t *mem_type*)

Parameters

in	*mem_type*	memory type

Returns

human readable string of `mem_type`

The mess_memusage_t_str function converts a mess_memusage_t to a human readable string.

8.59.4.42 const char* mess_multidirect_t_str (const mess_multidirect_t *ml_type*)

Parameters

in	*ml_type*	type of multidirect solver

Returns

human readable string of `ml_type`

The mess_multidirect_t_str function converts a mess_multidirect_t to a human readable string.

8.59.4.43 const char* mess_norm_t_str (const mess_norm_t *nrm_type*)

Parameters

in	*nrm_type*	memory type

Returns

human readable string of `nrm_type`

The mess_norm_t_str function converts a mess_norm_t to a human readable string.

8.59.4.44 void mess_openmp_info (void)

The mess_openmp_info function displays some information about the OpenMP environment including

- current threads,
- number of CPUs,
- maximal threads,
- thread ID.

8.59.4.45 const char* mess_operation_t_str (const mess_operation_t *op*)

Parameters

in	*op*	input operation type value

Returns

constant pointer to a human readable string containing the name of the operation type

The mess_operation_t_str function translates an operation type constant to the corresponding string and returns a constant pointer to this string.

See also

mess_operation_t

8.59.4.46 int mess_options_st_offset (int * *size*, int * *offsets*, char ** *fieldnames*)

Parameters

in,out	*size*	input/output contains `sizeof` value
in,out	*offsets*	input/output contains relative address of fields of mess_options_st
in,out	*fieldnames*	input/outout contains correspondig field names of
	offsets	

Returns

number of fields of mess_options_st

The mess_options_st_offset function returns informations about the memory layout of mess_options_st. We assume that `offsets` and `fieldnames` points to preallocated memory. You can call the function also with `NULL` pointers, in this case the pointer is not used.

8.59.4.47 const char* mess_parameter_t_str (const mess_parameter_t *para_type*)

Parameters

in	*para_type*	parameter type

Returns

human readable string of `para_type`

The mess_parameter_t_str function converts a mess_parameter_t to a human readable string.

8.59.4.48 void mess_print_bytes (mess_int_t *size*)

Parameters

in	*size*	input size to print

The mess_print_bytes prints out a memory size with the correct unit.

8.59.4.49 int mess_print_format_double (double *d*)

Parameters

in	*d*	input value to print

Returns

lenght of printed characters.

8.59.4.50 int mess_print_format_double_cpx (mess_double_cpx_t *cpx*)

Parameters

in	*cpx*	input value to print

Returns

lenght of printed characters.

8.59.4.51 int mess_print_format_select (mess_print_format_t *p*)

Parameters

in	*p*	input mess_print_format_t type to change printing behaviour

Returns

zero on success or a non-zero error value

The mess_print_format_select function selects a printing format used for printing `double` and mess_double_↩cpx_t values of mess_matrix and mess_vector structures.

8.59.4.52 const char∗ mess_residual_t_str (const mess_residual_t *res_type*)

Parameters

in	*res_type*	type of residual method

Returns

human readable string of `res_type`

The mess_residual_t_str function converts a mess_residual_t to a human readable string.

8.59.4.53 void mess_set_malloc (void ∗ *malloc_ptr,* void ∗ *realloc_ptr,* void ∗ *calloc_ptr,* void ∗ *free_ptr*)

Parameters

in	*malloc_ptr*	input Pointer to the malloc function
in	*realloc_ptr*	input Pointer to the realloc function
in	*calloc_ptr*	input Pointer to the calloc function
in	*free_ptr*	input Pointer to the free function

The mess_set_malloc function sets a new memory management function. That means that the typical C memory management functions are replaced by the given ones. That given function must have the same calling sequence as the ones provide by libc. If the mess_set_malloc function is never called, the standard libc routines are used. If the default needs to be reseted use

```
@ref mess_set_malloc(NULL, NULL, NULL, NULL);
```

The primary intention behind this function is to be able to use the `MATLAB` memory management when compiling a MEX file.

Remarks

It only affects memory allocation done by **`M.E.S.S.`** Memory allocations done by subpackages like `UMFP←ACK`, `CSparse`, ... are not covered by this mechanism.

Attention

The function access internal static variables and is not thread-safe. It must be called outside any multithreaded code.

8.59.4.54 const char∗ mess_sign_scale_t_str (const mess_sign_scale_t *sc_type*)

Parameters

in	*sc_type*	type of sign scaling method

Returns

human readable string of `sc_type`

The mess_sign_scale_t_str function converts a mess_sign_scale_t to a human readable string.

8.59.4.55 int mess_status_st_offset (int * *size*, int * *offsets*, char ** *fieldnames*)

Parameters

in,out	*size*	input/output contains `sizeof` value
in,out	*offsets*	input/output contains relative address of fields of mess_status_st
in,out	*fieldnames*	input/outout contains correspondig field names of
	offsets	

Returns

number of fields of mess_status_st

The mess_status_st_offset function returns informations about the memory layout of mess_status_st. We assume that `offsets` and `fieldnames` points to preallocated memory. You can call the function also with `NULL` pointers, in this case the pointer is not used.

8.59.4.56 const char* mess_storage_t_str (const mess_storage_t *store_type*)

Parameters

in	*store_type*	input storage type value

Returns

constant pointer to a human readable string containing the name of the storage type

The mess_storage_t_str function translates a storage type constant to the corresponding string and returns a constant pointer to this string.

See also

mess_storage_t

8.59.4.57 const char* mess_symmetry_t_str (const mess_symmetry_t *sym*)

Parameters

in	*sym*	input symmetry type value

Returns

constant pointer to a human readable string containing the name of the symmetry type

The mess_symmetry_t_str function translates a symmetry type constant to the corresponding string and returns a constant pointer to this string.

See also

mess_symmetry_t

8.59.4.58 int mess_vector_st_offset (int * *size*, int * *offsets*, char ** *fieldnames*)

Parameters

in,out	*size*	input/output contains `sizeof` value
in,out	*offsets*	input/output contains relative address of fields of mess_vector_st
in,out	*fieldnames*	input/outout contains correspondig field names of
	offsets	

Returns

number of fields of mess_vector_st

The mess_vector_st_offset function returns informations about the memory layout of mess_vector_st. We assume that `offsets` and `fieldnames` points to preallocated memory. You can call the function also with `NULL` pointers, in this case the pointer is not used.

8.59.4.59 void mess_version (void)

The mess_version prints out the **`M.E.S.S.`** version and the mess_int_t data type.

Here is the call graph for this function:

8.59.4.60 int mess_version_major (void)

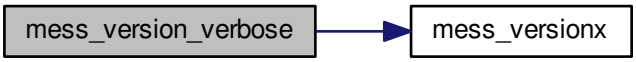
8.59.4.61 int mess_version_minor (void)

8.59.4.62 int mess_version_patch (void)

8.59.4.63 void mess_version_verbose (void)

The mess_version_verbose prints defailed out the **M.E.S.S.** version and the mess_int_t data type.

Here is the call graph for this function:



8.59.4.64 double nearbyint (double *x*)

Parameters

in	*x*	input value

Returns

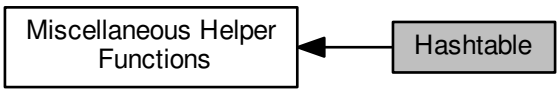
integer value of x

The nearbyint function rounds a value x to an integral value and returns it.

8.60 Hashtable

This category contains datastructures, type definitions and functions to handle a hashtable.

Collaboration diagram for Hashtable:



It includes functions to

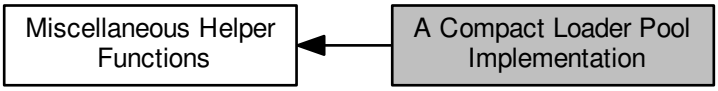
- create a hashtable,
- delete a hashtable,
- find, insert or remove an object in a hashtable,
- show content of a hashtable.

Additionally there are datastructures to represent a hashtable and corresponding type definitions.

8.61 A Compact Loader Pool Implementation

This category contains several structures, type definitions and functions concerning a threadpool.

Collaboration diagram for A Compact Loader Pool Implementation:



Data Structures

- struct mess_threadpooljob_st

  Representation of a thread pool job.
- struct mess_threadpool_st

  Representation of a thread pool.

Typedefs

- typedef struct mess_threadpooljob_st ∗ mess_threadpooljob

  Type definition of a thread pool job.
- typedef struct mess_threadpool_st ∗ mess_threadpool

  Type definition of a thread pool.

Functions

- int mess_threadpool_init (mess_threadpool ∗pool, int threads, int max_size)

  Create a loader pool.
- void ∗ mess_threadpool_worker (void ∗arg)

  Implementation of the loader thread.
- int mess_threadpool_clear (mess_threadpool ∗pool)

  Stop and clean up a loader pool.
- int mess_threadpool_insert (mess_threadpool pool, mess_threadpooljob job, mess_int_t ∗jobid)

  Insert a task in a loader pool.
- int mess_threadpool_isdone (mess_threadpool pool, mess_int_t id)

  Check if a job is done.
- int mess_threadpool_waitdone (mess_threadpool pool, mess_int_t id)

  Wait until a job is done.
- int mess_threadpooljob_init (mess_threadpooljob ∗job)

  Create a job for a loader pool.
- int mess_threadpooljob_clear (mess_threadpooljob ∗job)

  Stop and clean up a loader pool job.

8.61.1 Detailed Description

It includes structures and type definitions of a thread pool and jobs of a thread pool, implementations of the loader thread and functions to create, stop and exit a thread pool or to create, exit a job and to insert a task to the loader pool.
Additionally there are functions to wait until a job is done and to check if a job is done.

8.61.2 Typedef Documentation

8.61.2.1 typedef struct mess_threadpool_st* mess_threadpool

The mess_threadpool type definition contains all information to represent a threadpool.

8.61.2.2 typedef struct mess_threadpooljob_st* mess_threadpooljob

The mess_threadpooljob type definition contains all information to represent a job for the threadpool.

8.61.3 Function Documentation

8.61.3.1 int mess_threadpool_clear (mess_threadpool * *pool*)

Parameters

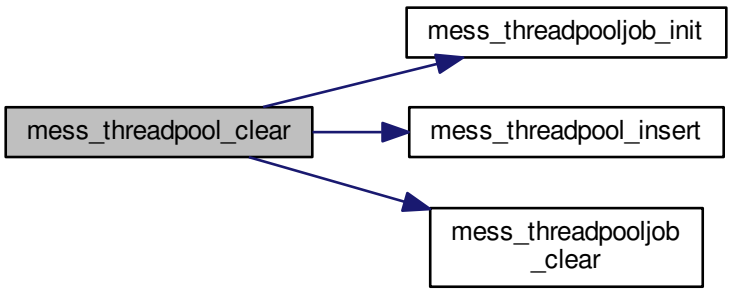
in,out	*pool*	pool to stop

Returns

zero on success or a non-zero error value otherwise

The mess_threadpool_clear function sends a stop job to all threads and cleans up all data structures.

Here is the call graph for this function:



8.61.3.2 int mess_threadpool_init (mess_threadpool * *pool*, int *threads*, int *max_size*)

Parameters

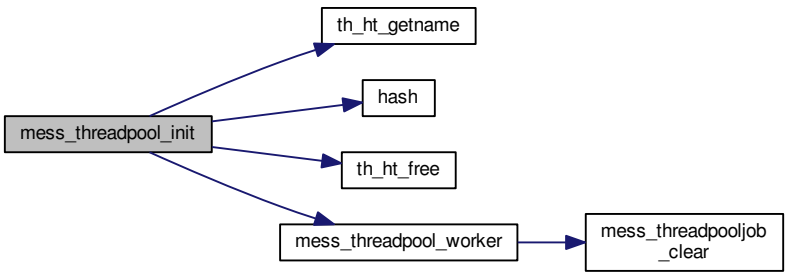
out	*pool*	pointer to a threadpool object
in	*threads*	input number of threads to create
in	*max_size*	input maximal size of the queue

Returns

zero on success or a non-zero error value otherwise

The mess_threadpool_init function initializes load threads of the cache to disk property of a spmv.
Parameter $threads$ describes the amount of loader threads which should be created, usually it should be one.
The size parameter max_{size} sets the maximum length of the queue which holds the indices to load, a good size is a number of shifts but it depends on the usage of the shift parameters in the algorithm.

Here is the call graph for this function:



8.61.3.3 int mess_threadpool_insert (mess_threadpool *pool*, mess_threadpooljob *job*, mess_int_t * *jobid*)

Parameters

in,out	*pool*	threadpool object where the job should be insert
in	*job*	input job object to put in the pool
out	*jobid*	output id of the job

Returns

zero on success or a non-zero error value otherwise

The mess_threadpool_insert function inserts a job into a loader pool $pool$ getting a identification $jobid$.

8.61.3.4 int mess_threadpool_isdone (mess_threadpool *pool*, mess_int_t *id*)

Parameters

in	*pool*	input loader pool
in	*id*	input id to check

Returns

1 if a job is not done, otherwise the job is done

The mess_threadpool_isdone function checks if a job with identification id is done.

8.61.3.5 int mess_threadpool_waitdone (mess_threadpool *pool*, mess_int_t *id*)

Parameters

in	*pool*	input loader pool
in	*id*	input id to wait

Returns

zero on success or a non-zero error value otherwise

The mess_threadpool_waitdone function waits until a job with identification id is done.

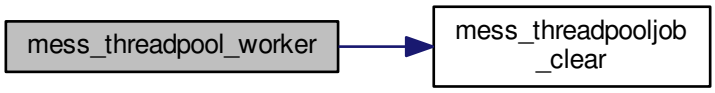
8.61.3.6 void* mess_threadpool_worker (void * *arg*)

Parameters

in	*arg*	input argument from the task pool

The mess_threadpool_worker function realizes the load algorithm for a index specified in arg.
If the values are loaded, a signal is send to a waiting thread.

Here is the call graph for this function:



8.61.3.7 int mess_threadpooljob_clear (mess_threadpooljob * *job*)

Parameters

in,out	*job*	job object to stop

Returns

zero on success or a non-zero error value otherwise

The mess_threadpooljob_clear function stops and cleans up a job for a loader pool.

8.61.3.8 int mess_threadpooljob_init (mess_threadpooljob * *job*)

Parameters

out	*job*	job object to create

Returns

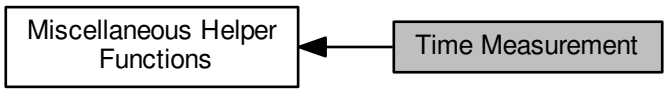
zero on success or a non-zero error value otherwise

The mess_threadpooljob_init function creates a job for a loader pool.

8.62 Time Measurement

This category contains several time measurement functions.

Collaboration diagram for Time Measurement:



Data Structures

- struct mess_timer_st

  Representation of a timer.

Typedefs

- typedef struct mess_timer_st ∗ mess_timer

  Type definition of a timer.

Functions

- double mess_wtime (void)

  Return a timer value.
- double mess_ctime (void)

  Get the current CPU time.
- int mess_timer_init (mess_timer ∗timer)

  Initialize a timer object.
- int mess_timer_clear (mess_timer ∗timer)

  Clean up a mess_timer object.
- double mess_timer_get (mess_timer timer)

  Get current elasped time since starting the timer.
- double mess_timer_getandreset (mess_timer timer)

  Get elapsed time since starting the timer and reset.
- double mess_timer_diff (mess_timer timerStart, mess_timer timerEnd)

  Get difference between two timers.

8.62.1 Detailed Description

It includes functions to initialize or clean up a timer object.
Additionally there are some functions to get a timer value, the current elapsed time since starting the timer (and the reset of the timer) or the difference between two timers. It also includes a wall-time and a cpu-time function.

8.62.2 Typedef Documentation

8.62.2.1 typedef struct mess_timer_st* mess_timer

The mess_timer type definition defines a timer.

8.62.3 Function Documentation

8.62.3.1 double mess_ctime (void)

Returns

a double value containing the enlasped CPU time

The mess_ctime function can be used for easy CPU time measuring. It returns a time as a `double`. The time measuring will work like the following:

```
double tStart, tEnd, tRun;
tStart = mess_ctime();
doSomething();
tEnd = mess_ctime();
tRun = tEnd-tStart;
```

8.62.3.2 int mess_timer_clear (mess_timer * *timer*)

Parameters

in,out	*timer*	pointer to a mess_timer object

Returns

zero on success or a non-zero error value otherwise

The mess_timer_clear function clears a mess_timer object from memory.

8.62.3.3 double mess_timer_diff (mess_timer *timerStart*, mess_timer *timerEnd*)

Parameters

in	*timerStart*	input first mess_timer object
in	*timerEnd*	input second mess_timer object

Returns

difference between timers

The mess_timer_diff function returns the difference between two timers.

8.62.3.4 double mess_timer_get (mess_timer *timer*)

Parameters

in	*timer*	input mess_timer object

Returns

elapsed time since starting the timer

The mess_timer_get function returns the elapsed time since the given timer was started or reseted.
Time will be returned in seconds.

Here is the call graph for this function:

8.62.3.5 double mess_timer_getandreset (mess_timer *timer*)

Parameters

in,out	*timer*	a mess_timer object

Returns

elapsed time since start ing the timer

The mess_timer_getandreset function returns the elapsed time since the timer was started and restarts the timer.

Here is the call graph for this function:

8.62.3.6 int mess_timer_init (mess_timer * *timer*)

Parameters

in,out	*timer*	pointer to timer object

Returns

zero on success or a non-zero error value otherwise

The mess_timer_init function initializes a mess_timer object and starts the timer.
It uses the POSIX Realtime Extensisons POSIX.1-2001.

See also

mess_wtime

Here is the call graph for this function:

8.62.3.7 double mess_wtime (void)

Returns

time

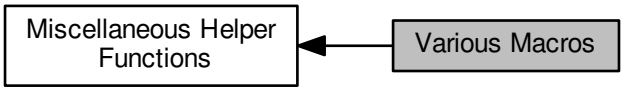
The mess_wtime function returns a time as a `double` and can be used for easy runtime measuring.
The time measuring will works like the following:

```
double tStart, tEnd, tRun;
tStart = mess_wtime();
doSomething();
tEnd = mess_wtime();
tRun = tEnd-tStart;
```

8.63 Various Macros

This category contains several macros and definitions.

Collaboration diagram for Various Macros:



Macros

- #define MESS_MAX(a, b) (((a)<(b)) ? (b):(a))

  Macro for Maximium of two numbers. The MESS_MAX macro return the maximum of two numbers.
- #define MESS_MIN(a, b) (((a)<(b)) ? (a):(b))

  Macro for Minimum of two numbers. The MESS_MIN macro return the minimum of two numbers.
- #define MESS_INIT_MATRICES(...)

  Macro to init a list of matrices.
- #define MESS_RESET_MATRICES(...)

  Macro to reset a list of matrices.
- #define MESS_CLEAR_MATRICES(...)

  Macro to clear a list of matrices.
- #define MESS_INIT_VECTORS(...)

  Macro to init a list of mess_vector instances.
- #define MESS_CLEAR_VECTORS(...)

  Macro to clear a list of vectors.
- #define MESS_INIT_DIRECTS(...)

  Macro to init a list of direct solvers.
- #define MESS_CLEAR_DIRECTS(...)

  Macro to clear a list of direct solvers.
- #define MESS_INIT_EQUATIONS(...)

  Macro to init a list of mess_equation instances.
- #define MESS_CLEAR_EQUATIONS(...)

  Macro to clear a list of mess_equation instances.
- #define MESS_CLEAR_POINTERS(...)

  Macro to clear a list of pointers.
- #define mess_free(X)

  Macro to free memory.
- #define mess_try_alloc(mem, cast, X)

  Macro to allocate memory (malloc).
- #define mess_try_alloc2(mem, cast, X)

  Macro to allocate memory (malloc).
- #define mess_try_calloc(mem, cast, n, size)

  Macro to allocate memory (calloc).

- #define mess_try_calloc2(mem, cast, n, size)

  Macro to allocate memory (calloc).
- #define mess_try_realloc(mem, cast, X)

  Macro to allocate memory (realloc).
- #define mess_try_realloc2(mem, cast, X)

  Macro to allocate memory (realloc).
- #define mess_try_alloc_omp(mem, cast, X)

  Macro to allocate memory (malloc).
- #define MESS_CALL_IF_EXIST(func, args...)

  Macro to call a static function (if available) with arguments.
- #define mess_check_nullpointer(X) if ((X)==NULL) { MSG_ERROR("%s points to NULL\n", #X); return MESS_ERROR_NULLPOINTER; }

  Macro to check if an object points to `NULL`.
- #define mess_check_notnullpointer(X) if ((X)!=NULL) { MSG_ERROR("%s points not to NULL\n", #X); return MESS_ERROR_NOTNULLPOINTER; }

  Macro to check if an object does not point to `NULL`.
- #define mess_check_real(X) if (!MESS_IS_REAL((X))) { MSG_ERROR("%s have to be real, current = %s \n", #X, mess_datatype_t_str((X)->data_type)); return MESS_ERROR_DATATYPE; }

  Macro to check if a matrix or vector is real.
- #define mess_check_complex(X) if (!MESS_IS_COMPLEX((X))) { MSG_ERROR("%s have to be complex\n", #X); return MESS_ERROR_DATATYPE; }

  Macro to check if a matrix or vector is complex.
- #define mess_check_square(X) if ((X)->rows!=(X)->cols) { MSG_ERROR("%s have to be square, rows = %ld, cols = %ld\n", #X, (long) (X)->rows, (long) (X)->cols); return MESS_ERROR_DIMENSION;}

  Macro to check if a matrix is square.
- #define mess_check_csc(X) if(!MESS_IS_CSC((X))) { MSG_ERROR("%s have to be a CSC stored matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

  Macro to check if a matrix is stored in the CSC storage.
- #define mess_check_csr(X) if(!MESS_IS_CSR((X))) { MSG_ERROR("%s have to be a CSR stored matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

  Macro to check if a matrix is stored in the CSR storage.
- #define mess_check_dense(X) if(!MESS_IS_DENSE((X))) { MSG_ERROR("%s have to be a dense matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

  Macro to check if a matrix is stored in the DENSE storage.
- #define mess_check_sparse(X) if(!(MESS_IS_CSC((X)) || MESS_IS_CSR((X)) || MESS_IS_COORD((X)))) { MSG_ERROR("%s have to be a sparse matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

  Macro to check if a matrix is stored in the sparse storage format.
- #define mess_check_coord(X) if(!MESS_IS_COORD((X))) { MSG_ERROR("%s have to be a coordinate matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

  Macro to check if a matrix is stored in the COORD storage.
- #define mess_check_nocoord(X) if(MESS_IS_COORD((X))) { MSG_ERROR("%s is a coordinate matrix. Coordinate matrices are not supported.\n", #X); return MESS_ERROR_STORAGETYPE;}

  Macro to check if a matrix is not stored in the CSC storage.
- #define mess_check_positive(X) if ((X) <= 0) { MSG_ERROR("%s have to be positive\n", #X); return MESS_ERROR_ARGUMENTS;}

  Macro to check if an object is positive.
- #define mess_check_nonnegative(X) if ((X) < 0) { MSG_ERROR("%s have to be nonnegative\n", #X); return MESS_ERROR_ARGUMENTS;}

  Macro to check if an object is nonnegative.
- #define mess_check_same_size(X, Y) if (((X)->rows != (Y)->rows) || ((X)->cols != (Y)->cols)) { MSG_ERROR("%s and %s must have the same size. (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT ", " MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->rows, Y->cols); return (MESS_ERROR_DIMENSION); }

Macro to check if two matrices have the same size.

- #define mess_check_same_dim(X, Y) if (((X)->dim != (Y)->dim)) { MSG_ERROR("%s and %s must have the same dim. " MESS_PRINTF_INT " <-> " MESS_PRINTF_INT ")\n",#X,#Y, X->dim, Y->dim); return (MESS_ERROR_DIMENSION); }

  Macro to check if two structs have the same dim.

- #define mess_check_same_rows(X, Y) if (((X)->rows != (Y)->rows)) { MSG_ERROR("%s and %s must have the number of rows. (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT ", " MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->rows, Y->cols); return (MESS_ERROR_DIMEN↩SION); }

  Macro to check if two matrices or vectors have the same number of rows.

- #define mess_check_same_cols(X, Y) if (((X)->cols != (Y)->cols)) { MSG_ERROR("%s and %s must have the number of columns. (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT ", " MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->rows, Y->cols); return (MESS_ERROR_DIMENS↩ION); }

  Macro to check if two matrices or vectors have the same number of columns.

- #define mess_check_same_colsrows(X, Y) if (((X)->cols != (Y)->rows)) { MSG_ERROR("Number of cols of %s must be the same as the number of rows of %s . (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT ", " MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->rows, Y->cols); return (MESS_ERROR_DIMENSION); }

  Macro to check if the number of rows of a matrix is identical to the number of columns of another matrix.

- #define mess_check_same_colsdim(X, Y) if (((X)->cols != (Y)->dim)) { MSG_ERROR("Number of cols of %s must be the same as the dimension of %s . (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->dim); return (MESS_ERROR_DIMENSION); }

  Macro to check if the number of columns of a matrix is identical to the dimension of a vector.

- #define mess_check_same_rowsdim(X, Y) if (((X)->rows != (Y)->dim)) { MSG_ERROR("Number of rows of %s must be the same as the dimension of %s . (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->dim); return (MESS_ERROR_DIMENSION); }

  Macro to check if the number of rows of a matrix is identical to the dimension of a vector.

- #define mess_check_same_datatype(X, Y) if (((X)->data_type != (Y)->data_type)) { MSG_ERROR("%s and %s must have the same data type.\n",#X,#Y); return(MESS_ERROR_DATATYPE); }

  Macro to check if two objects have the same datatype.

- #define mess_check_same_storetype(X, Y) if (((X)->store_type != (Y)->store_type)) { MSG_ERROR("%s and %s must have the same storage type.\n",#X,#Y); return (MESS_ERROR_STORAGETYPE); }

  Macro to check if two objects have the same storage type.

- #define mess_check_real_or_complex(X) if ((!MESS_IS_REAL(X) && ! MESS_IS_COMPLEX(X))) { MS↩G_ERROR("%s must be real or complex.\n",#X); return (MESS_ERROR_DATATYPE); }

  Macro to check if a matrix or a vector is real or complex.

- #define mess_check_operation_type(X)

  Macro to check if an operation on a matrix is correct.

- #define mess_check_storage_type(X)

  Macro to check if a given storage type is valid.

- #define mess_check_datatype(DT)

  Macro to check if an datatype is correct.

- #define mess_check_true(X) if ((X) == 0) { MSG_ERROR("Assertion %s == true failed.\n", #X); return M↩ESS_ERROR_ARGUMENTS; }

  Check if a value is true.

8.63.1 Detailed Description

These macros concern the memory manager, vectors and matrices.

8.63.2 Macro Definition Documentation

8.63.2.1 #define MESS_CALL_IF_EXIST(*func, args...*)

Value:

```
if ( func != NULL){      \
            ret = func(args);    \
            FUNCTION_FAILURE_HANDLE(ret, (ret!=0), func); \
        }
```

The MESS_CALL_IF_EXIST allow to call a static function `func` (if not points to `NULL`) with given list of arguments. If the function `func` points to `NULL`, the function is not called. Please be sure that a variable `ret` of type mess↩_int_t is predefined for the FUNCTION_FAILURE_HANDLE. Internal use only.

8.63.2.2 #define mess_check_complex(X) if (!MESS_IS_COMPLEX((X))) { MSG_ERROR("%s have to be complex\n", #X); return MESS_ERROR_DATATYPE; }

Check if a matrix or vector is complex.

8.63.2.3 #define mess_check_coord(X) if(!MESS_IS_COORD((X))) { MSG_ERROR("%s have to be a coordinate matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

Check if a matrix is stored in the COORD storage.

8.63.2.4 #define mess_check_csc(X) if(!MESS_IS_CSC((X))) { MSG_ERROR("%s have to be a CSC stored matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

Check if a matrix is stored in the CSC storage.

8.63.2.5 #define mess_check_csr(X) if(!MESS_IS_CSR((X))) { MSG_ERROR("%s have to be a CSR stored matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

Check if a matrix is stored in the CSR storage.

8.63.2.6 #define mess_check_datatype(*DT*)

Value:

```
if ( ((DT) != MESS_REAL) && ( (DT)!=MESS_COMPLEX) ) { \
    MSG_ERROR("The given mess_datatype_t is not supported. Invalid Argument\n");\
    return ( MESS_ERROR_DATATYPE);  }
```

Check if an datatype is correct.

8.63.2.7 #define mess_check_dense(X) if(!MESS_IS_DENSE((X))) { MSG_ERROR("%s have to be a dense matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

Check if a matrix is stored in the DENSE storage.

8.63.2.8 #define mess_check_nocoord(X) if(MESS_IS_COORD((X))) { MSG_ERROR("%s is a coordinate matrix. Coordinate matrices are not supported.\n", #X); return MESS_ERROR_STORAGETYPE;}

Check if a matrix is not stored in the CSC storage.

8.63.2.9 #define mess_check_nonnegative(X) if ((X) < 0) { MSG_ERROR("%s have to be nonnegative\n", #X); return MESS_ERROR_ARGUMENTS;}

Check if an object is nonnegative.

8.63.2.10 #define mess_check_notnullpointer(X) if ((X)!=NULL) { MSG_ERROR("%s points not to NULL\n", #X); return MESS_ERROR_NOTNULLPOINTER; }

Check if an object does not point to `NULL`.

8.63.2.11 #define mess_check_nullpointer(X) if ((X)==NULL) { MSG_ERROR("%s points to NULL\n", #X); return MESS_ERROR_NULLPOINTER; }

Check if an object points to `NULL`.

8.63.2.12 #define mess_check_operation_type(X)

Value:

```
if ( ((X) != MESS_OP_NONE) && ( (X)!=MESS_OP_TRANSPOSE) && ( (
      X) != MESS_OP_HERMITIAN )) { \
    MSG_ERROR("The given mess_operation_t is not supported. Invalid Argument\n");\
    return ( MESS_ERROR_ARGUMENTS);  }
```

Check if an operation on a matrix is correct.

8.63.2.13 #define mess_check_positive(X) if ((X) <= 0) { MSG_ERROR("%s have to be positive\n", #X); return MESS_ERROR_ARGUMENTS;}

Check if an object is positive.

8.63.2.14 #define mess_check_real(X) if (!MESS_IS_REAL((X))) { MSG_ERROR("%s have to be real, current = %s \n", #X, mess_datatype_t_str((X)->data_type)); return MESS_ERROR_DATATYPE; }

Check if a matrix or vector is real.

8.63.2.15 #define mess_check_real_or_complex(X) if ((!MESS_IS_REAL(X) && ! MESS_IS_COMPLEX(X))) { MSG_ERROR("%s must be real or complex.\n",#X); return (MESS_ERROR_DATATYPE); }

Check if a matrix or a vector is real or complex.

8.63.2.16 #define mess_check_same_cols(X, Y) if (((X)->cols != (Y)->cols)) { MSG_ERROR("%s and %s must have the number of columns. (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT ", " MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->rows, Y->cols); return (MESS_ERROR_DIMENSION); }

Check if two matrices or vectors have the same number of columns.

8.63.2.17 #define mess_check_same_colsdim(X, Y) if (((X)->cols != (Y)->dim)) { MSG_ERROR("Number of cols of %s must be the same as the dimension of %s . (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->dim); return (MESS_ERROR_DIMENSION); }

Check if the number of columns of a matrix is identical to the dimension of a vector.

8.63.2.18 #define mess_check_same_colsrows(X, Y) if (((X)->cols != (Y)->rows)) { MSG_ERROR("Number of cols of %s must be the same as the number of rows of %s . (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT ", " MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->rows, Y->cols); return (MESS_ERROR_DIMENSION); }

Check if the number of rows of a matrix is identical to the number of columns of another matrix.

8.63.2.19 #define mess_check_same_datatype(X, Y) if (((X)->data_type != (Y)->data_type)) { MSG_ERROR("%s and %s must have the same data type.\n",#X,#Y); return(MESS_ERROR_DATATYPE); }

Check if two objects have the same datatype.

8.63.2.20 #define mess_check_same_dim(X, Y) if (((X)->dim != (Y)->dim)) { MSG_ERROR("%s and %s must have the same dim. " MESS_PRINTF_INT " <-> " MESS_PRINTF_INT ")\n",#X,#Y, X->dim, Y->dim); return (MESS_ERROR_DIMENSION); }

Check if two structs have the same dim.

8.63.2.21 #define mess_check_same_rows(X, Y) if (((X)->rows != (Y)->rows)) { MSG_ERROR("%s and %s must have the number of rows. (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT ", " MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->rows, Y->cols); return (MESS_ERROR_DIMENSION); }

Check if two matrices or vectors have the same number of rows.

8.63.2.22 #define mess_check_same_rowsdim(X, Y) if (((X)->rows != (Y)->dim)) { MSG_ERROR("Number of rows of %s must be the same as the dimension of %s . (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->dim); return (MESS_ERROR_DIMENSION); }

Check if the number of rows of a matrix is identical to the dimension of a vector.

8.63.2.23 #define mess_check_same_size(X, Y) if (((X)->rows != (Y)->rows) || ((X)->cols != (Y)->cols)) { MSG_ERROR("%s and %s must have the same size. (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT ", " MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->rows, Y->cols); return (MESS_ERROR_DIMENSION); }

Check if two matrices have the same size.

8.63.2.24 #define mess_check_same_storetype(X, Y) if (((X)->store_type != (Y)->store_type)) { MSG_ERROR("%s and %s must have the same storage type.\n",#X,#Y); return (MESS_ERROR_STORAGETYPE); }

Check if two objects have the same storage type.

8.63.2.25 #define mess_check_sparse(X) if(!(MESS_IS_CSC((X)) || MESS_IS_CSR((X)) || MESS_IS_COORD((X)))) { MSG_ERROR("%s have to be a sparse matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

Check if a matrix is stored in the sparse storage format.

8.63.2.26 #define mess_check_square(X) if ((X)->rows!=(X)->cols) { MSG_ERROR("%s have to be square, rows = %ld, cols = %ld\n", #X, (long) (X)->rows, (long) (X)->cols); return MESS_ERROR_DIMENSION;}

Check if a matrix is square.

8.63.2.27 #define mess_check_storage_type(X)

Value:

```
if ( ((X) != MESS_DENSE) && ( (X)!=MESS_CSC) && ( (X) !=
      MESS_CSR )  && ( (X) != MESS_COORD )) { \
    MSG_ERROR("The given mess_storage_t is not supported. Invalid Argument\n");\
    return ( MESS_ERROR_ARGUMENTS);  }
```

Check if a given storage type is valid.

8.63.2.28 #define mess_check_true(X) if ((X) == 0) { MSG_ERROR("Assertion %s == true failed.\n", #X); return MESS_ERROR_ARGUMENTS; }

The macro checks if a given value is true. In this way it is similar to assert.

8.63.2.29 #define MESS_CLEAR_DIRECTS(...)

Value:

```
{ \
    void * _stopper = (void * ) -1; \
    mess_direct*  _M[] = { __VA_ARGS__, _stopper}; \
    int _i; \
    for ( _i = 0; _M[_i] != _stopper ; _i++) {\
        if ( *_M[_i] != NULL) mess_direct_clear(_M[_i]);\
    }\
}
```

The MESS_CLEAR_DIRECTS macro allow to clear a list of direct solvers within one call. It uses a variable argument list macro. The calling sequence is

```
MESS_CLEAR_DIRECTS(&direct1, &direct2, &direct3, ...);
```

8.63.2.30 #define MESS_CLEAR_EQUATIONS(...)

Value:

```
{ \
    void * _stopper = (void * ) -1; \
    mess_equation*  _M[] = { __VA_ARGS__, _stopper}; \
    int _i; \
    for ( _i = 0; _M[_i] != _stopper ; _i++) {\
        if ( *_M[_i] != NULL) mess_equation_clear(_M[_i]);\
    }\
}
```

The MESS_CLEAR_EQUATIONS macro allow to clear a list of direct solvers within one call. It uses a variable argument list macro. The calling sequence is

```
MESS_CLEAR_EQUATIONS(&eqn1, &eqn2, &eqn3, ...);
```

8.63.2.31 #define MESS_CLEAR_MATRICES(...)

Value:

```
{ \
    void * _stopper = (void * ) -1; \
    mess_matrix*  _M[] = { __VA_ARGS__, _stopper}; \
    int _i; \
    for ( _i = 0; _M[_i] != _stopper ; _i++) {\
        if ( *_M[_i] != NULL) mess_matrix_clear(_M[_i]);\
    }\
}
```

The MESS_CLEAR_MATRICES macro allow to clear a list of matrices within one call. It uses a variable argument list macro. The calling sequence is

```
MESS_CLEAR_MATRICES(&A, &B, &C, ...);
```

Before the matrices are cleared it is check if they are already `NULL`.

8.63.2.32 #define MESS_CLEAR_POINTERS(...)

Value:

```
{ \
    void * _stopper = (void * ) -1; \
    void*  _M[] = { __VA_ARGS__, _stopper}; \
    int _i; \
    for ( _i = 0; _M[_i] != _stopper ; _i++) {\
        if ( _M[_i] != NULL){\
            free(_M[_i]);\
            _M[_i]=NULL;\
        }\
    }\
}
```

The MESS_CLEAR_POINTERS macro allow to free a list of pointers within one call. It uses a variable argument list macro. The calling sequence is

```
MESS_CLEAR_POINTERS(A, B, C, ...);
```

Before the pointers are cleared it is check if they are already `NULL`.

8.63.2.33 #define MESS_CLEAR_VECTORS(...)

Value:

```
{ \
    void * _stopper = (void * ) -1; \
    mess_vector*  _M[] = { __VA_ARGS__, _stopper}; \
    int _i; \
    for ( _i = 0; _M[_i] != _stopper ; _i++) {\
        if ( *_M[_i] != NULL) mess_vector_clear(_M[_i]);\
    }\
}
```

The MESS_CLEAR_VECTORS macro allow to clear a list of vectors within one call. It uses a variable argument list macro. The calling sequence is

```
MESS_CLEAR_VECTORS(&v, &v2, &v3, ...);
```

Before the vectors are cleared it is check if they are already `NULL`.

8.63.2.34 #define mess_free(X)

Value:

```
do {\
    if ( (X) != NULL ) { \
        __mess_free((X));\
        (X) = NULL; \
    }\
} while (0)
```

The mess_free macro is a wrapper around free which checks for a NULL pointer first and sets the freed pointer to NULL afterwards.

8.63.2.35 #define MESS_INIT_DIRECTS(...)

Value:

```
{ \
    void * _stopper = (void * ) -1; \
    mess_direct*  _M[] = { __VA_ARGS__, _stopper}; \
    int _i; \
    for ( _i = 0; _M[_i] != _stopper ; _i++) {\
        mess_direct_init(_M[_i]);\
    }\
}
```

The MESS_INIT_DIRECTS macro allow to init a list of direct solvers within one call. It uses a variable argument list macro. The calling sequence is

```
MESS_INIT_DIRECTS(&direct1, &direct2, &direct3, ...);
```

8.63.2.36 #define MESS_INIT_EQUATIONS(...)

Value:

```
{ \
    void * _stopper = (void * ) -1; \
    mess_equation*  _M[] = { __VA_ARGS__, _stopper}; \
    int _i; \
    for ( _i = 0; _M[_i] != _stopper ; _i++) {\
        mess_equation_init(_M[_i]);\
    }\
}
```

The MESS_INIT_EQUATIONS macro allow to init a list of mess_equation instances within one call. It uses a variable argument list macro. The calling sequence is

```
MESS_INIT_EQUATIONS(&eqn1, &eqn2, &eqn3, ...);
```

8.63.2.37 #define MESS_INIT_MATRICES(...)

Value:

```
{ \
    void * _stopper = (void * ) -1; \
    mess_matrix*  _M[] = { __VA_ARGS__, _stopper}; \
    int _i; \
    for ( _i = 0; _M[_i] != _stopper ; _i++) {\
        mess_matrix_init(_M[_i]);\
    }\
}
```

The MESS_INIT_MATRICES macro allow to init a list of matrices within one call. It uses a variable argument list macro. The calling sequence is

```
MESS_INIT_MATRICES(&A, &B, &C, ...);
```

8.63.2.38 #define MESS_INIT_VECTORS(...)

Value:

```
{ \
    void * _stopper = (void * ) -1; \
    mess_vector*  _M[] = { __VA_ARGS__, _stopper}; \
    int _i; \
    for ( _i = 0; _M[_i] != _stopper ; _i++) {\
        mess_vector_init(_M[_i]);\
    }\
}
```

The MESS_INIT_VECTORS macro allow to init a list of vectors within one call. It uses a variable argument list macro. The calling sequence is

```
MESS_INIT_VECTORS(&v1, &v2, &v3, ...);
```

8.63.2.39 #define MESS_MAX(a, b) (((a)<(b)) ? (b):(a))

8.63.2.40 #define MESS_MIN(a, b) (((a)<(b)) ? (a):(b))

8.63.2.41 #define MESS_RESET_MATRICES(...)

Value:

```
{ \
    void * _stopper = (void * ) -1; \
    mess_matrix  _M[] = { __VA_ARGS__, _stopper}; \
    int _i; \
    for ( _i = 0; _M[_i] != _stopper ; _i++) {\
        MESS_MATRIX_RESET(_M[_i]);\
    }\
}
```

The MESS_RESET_MATRICES macro allow to reset a list of matrices within one call. It uses a variable argument list macro. The calling sequence is

```
MESS_RESET_MATRICES(A, B, C, ...);
```

8.63.2.42 #define mess_try_alloc(*mem, cast,* X)

Value:

```
{ \
    (mem) = (cast) __mess_malloc((X)); \
    if ( (X)>0 &&  (mem) == NULL ) { \
        MSG_ERROR("no memory left to allocate: %s size: %lu\n", #mem, (unsigned long)(
      X));\
        return (MESS_ERROR_MEMORY) ;\
    } \
}
```

The mess_try_alloc macro allocates new memory of possible with C's malloc. Otherwise it returns an error.

8.63.2.43 #define mess_try_alloc2(*mem, cast,* X)

Value:

```
{ \
    (mem) = (cast) __mess_malloc((X)); \
    if ( (X) > 0 && (mem) == NULL ) { \
        MSG_ERROR("no memory left to allocate: %s size: %lu\n", #mem, (unsigned long)(
      X));\
    } \
}
```

The mess_try_alloc macro allocates new memory of possible with C's malloc. In contrast to mess_try_alloc it does not return an error.

8.63.2.44 #define mess_try_alloc_omp(*mem, cast,* X)

Value:

```
do { \
    (mem) = (cast) __mess_malloc((X));\
    if ( (X) > 0 &&  (mem) == NULL ) { \
        MSG_ERROR("no memory left to allocate: %s size: %lu\n", #mem, (unsigned long)(
      X));\
        err = MESS_ERROR_MEMORY; \
    }\
} while (0)
```

The mess_try_alloc_omp macro allocates new memory of possible with C's malloc. Otherwise it returns an error.

8.63.2.45 #define mess_try_calloc(*mem, cast,* n, *size*)

Value:

```
{ \
    (mem) =(cast) __mess_calloc((n),(size));\
    if ( ((n)*(size))>0 && (mem) == NULL) { \
        MSG_ERROR("no memory left to allocate: %s size: %lu\n", #mem, (unsigned long)(size
      )*(n));\
        return (MESS_ERROR_MEMORY);\
    }\
}
```

The mess_try_calloc macro allocates new memory of possible with C's calloc. Otherwise it returns an error.

8.63.2.46 #define mess_try_calloc2(*mem, cast,* n, *size*)

Value:

```
{ \
    (mem) =(cast) __mess_calloc((n),(size));\
    if ( ((n)*(size))> 0 && (mem) == NULL) { \
        MSG_ERROR("no memory left to allocate: %s size: %lu\n", #mem, (unsigned long)(size
      )*(n));\
    }\
}
```

The mess_try_calloc macro allocates new memory of possible with C's calloc. In contrast to mess_try_calloc this function does not return an error.

8.63.2.47 #define mess_try_realloc(*mem, cast,* X)

Value:

```
{ \
    (mem) = (cast) __mess_realloc((mem),(X)); \
    if ((X) > 0  && (mem) == NULL){ \
        MSG_ERROR("no memory left to allocate: %s size: %lu\n", #mem,(unsigned long)(
      X));\
        return (MESS_ERROR_MEMORY);\
    }\
}
```

The mess_try_realloc macro allocates new memory of possible with C's realloc. Otherwise it returns an error.

8.63.2.48 #define mess_try_realloc2(*mem, cast,* X)

Value:

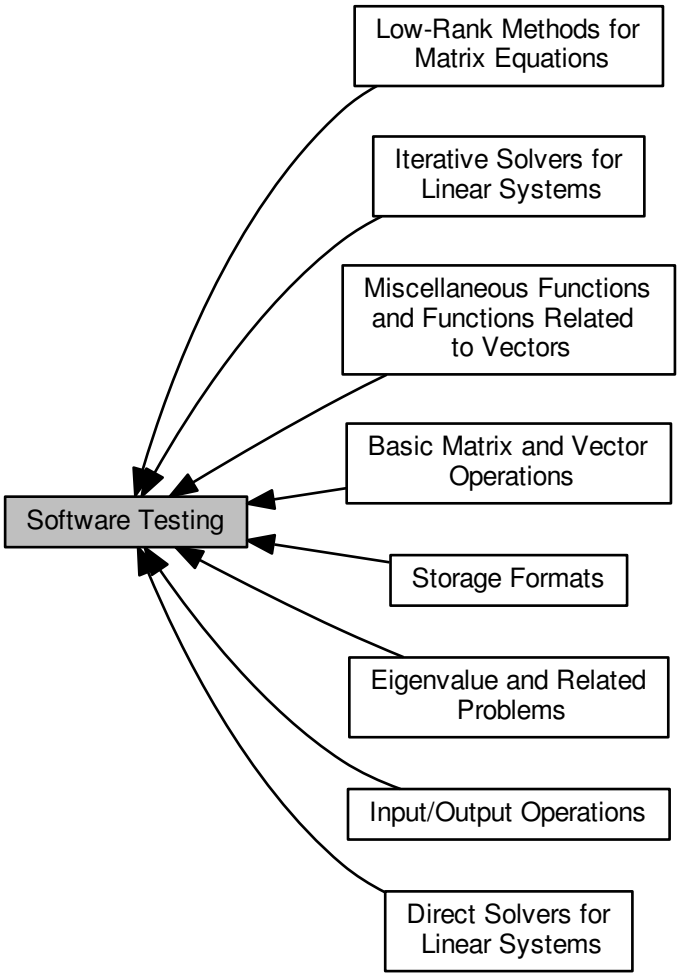
```
{ \
    (mem) = (cast) __mess_realloc((mem),(X)); \
    if ( (X) > 0 && (mem) == NULL){ \
        MSG_ERROR("no memory left to allocate: %s size: %lu\n", #mem,(unsigned long)(
      X));\
    }\
}
```

The mess_try_realloc macro allocates new memory of possible with C's realloc. In contrast to mess_try_realloc this function does not return an error.

8.64 Software Testing

This section contains different categories with examples to proof the correctness of certain functions.

Collaboration diagram for Software Testing:



Modules

- Basic Matrix and Vector Operations

  In this category implemented basic matrix and vector operations are proofed.
- Direct Solvers for Linear Systems

  In this category implemented direct solver for linear systems are proofed.
- Eigenvalue and Related Problems

  In this category implemented files to compute eigenvalues, eigenvectors and solve related problems are proofed.
- Input/Output Operations

  This category contains checks of different input and output operations.

- Iterative Solvers for Linear Systems

  In this category implemented iterative solver for linear systems are proofed.
- Low-Rank Methods for Matrix Equations

  This category contains checks of low-rank methods for different matrix equations.
- Storage Formats

  This category contains checks of conversions to different storage formats.
- Miscellaneous Functions and Functions Related to Vectors

  This category contains checks for misscellaneous function and function related to vectors.

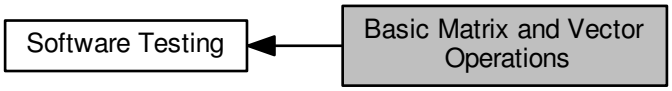
8.64.1 Detailed Description

This section contains different categories with examples to proof the correctness of certain functions. The test function could also be used as examples for serveral functions in **`M.E.S.S.`**. The tests are categorized similar to the origin of the tested functions.

8.65 Basic Matrix and Vector Operations

In this category implemented basic matrix and vector operations are proofed.

Collaboration diagram for Basic Matrix and Vector Operations:



Files

- file check_add.c

  Check matrix addition.
- file check_addcol2p.c

  Check the mess_matrix_addcols2p function.
- file check_any.c

  Check function mess_matrix_any.
- file check_bandwidth.c

  Check mess_matrix_bandwidth function.
- file check_biorth.c

  Check the construction of a biorthonormal basis for two matrices.
- file check_cat.c

  Check the concatenation of two matrices.
- file check_colaxpy2.c

  Check the mess_matrix_colaxpy2 function.
- file check_coldot.c

  Check the column dot product.
- file check_coldotc.c

  Check the column dot product of a complex matrix.
- file check_coldotcE.c

  Check the mess_matrix_coldotcE function.
- file check_coldotE.c

  Check the mess_matrix_coldotE function.
- file check_colnorm.c

  Check the column norm.
- file check_colscale.c

  Check the column scaling function of a matrix.
- file check_colsub.c

  Check the column submatrix computation.
- file check_colvecaxpy.c

  Check the axpy scalar vector product.
- file check_colvecdot.c

  Check the column vector product.

- file check_colvecdotc.c

  Check the complex column vector product.
- file check_condest.c

  Check the condition number estimation.
- file check_decomp.c

  Check the mess_matrix_decomp function.
- file check_diag.c

  Check the output of diagonal elements of a matrix.
- file check_dynorm2.c

  Check the 2-norm computation of a matrix.
- file check_dynorm2t.c

  Check the 2-norm computation of a transposed matrix.
- file check_dynorm_indefinite2.c

  Check the mess_matrix_indefinite_dynorm2 function.
- file check_dynorm_indefinitef.c

  Check the mess_matrix_indefinite_dynormf and mess_matrix_indefinite_dynormf2 functions.
- file check_dynormf.c

  Check the Frobenius-norm computation of a matrix.
- file check_dynormft.c

  Check the Frobenius-norm computation of a transposed matrix.
- file check_eliminate_zeros.c

  Check the mess_matrix_eliminate_zeros function.
- file check_eps.c

  Check the machine epsilon calculation.
- file check_fro_inn.c

  Check the frobenius inner product.
- file check_gaxpy.c

  Check the matrix-vector product with an update of a vector.
- file check_getelement.c

  Check if a matrix element is returned correctly.
- file check_matrix_biorth.c

  Check the biorthonormal basis construction.
- file check_matrix_colaxpy.c

  Check the column update of a matrix using an axpy-operation.
- file check_matrix_gbiorth.c

  Check the biorthonormal basis construction with respect to a weighted scalar product.
- file check_matrix_null.c

  Check the orthogonal basis construction of the nullspace.
- file check_matrix_sum.c

  Check the column and row sum computation of a matrix.
- file check_mgs.c

  Check the ortohogonal basis construction using the modified Gram-Schmidt process.
- file check_mgs_inplace.c

  Check the ortohogonal basis construction using the modified Gram-Schmidt process (inplace version).
- file check_mgsadd.c

  Check the orthogonal basis construction using the modified Gram-Schmidt process (column adding version).
- file check_mul.c

  Check the multiplication of two matrices.
- file check_mul_small.c

  Check the multiplication of two matrices (version with small matrices).
- file check_mul_tall.c

Check the multiplication of two matrices (version with tall matrices).

- file check_mulnormf2.c

  Check the Frobenius-norm computation of a matrix.
- file check_mvp.c

  Check matrix-vector product computation.
- file check_norm.c

  Check norm functions.
- file check_one_value.c

  Check the mess_matrix_one_value and mess_matrix_one_valuec functions.
- file check_orth.c

  Check the orthogonal basis construction using SVD.
- file check_proj_sym.c

  Check the proj_sym function.
- file check_rank.c

  Check rank estimation.
- file check_rowsub.c

  Check the row submatrix computation.
- file check_scalem.c

  Check the mess_matrix_rowscalem and mess_matrix_colscalem.
- file check_set_get_col_row.c

  Check the overwriting of a matrix column.
- file check_setelement.c

  Check the overwriting of a matrix element.
- file check_sub.c

  Check the sub matrix computation.
- file check_trace.c

  Check the trace computation.
- file check_transpose.c

  Check the transpose function.
- file check_tridiag.c

  Check mess_matrix_tridiag function.
- file check_kron.c

  Check the Kronecker product computation.
- file check_lift.c

  Check the mess_matrix_lift and mess_vector_lift function.

8.65.1 Detailed Description

There are files proving the correctness of following matrix operations

- add an additional zero block to a matrix
- add (computed) columns to a matrix
- add matrices
- compute a submatrix
- compute different norms of a matrix
- compute the column and row sum
- compute the Kronecker product

- compute the matrix-vector product
- compute the norm of a matrix column
- compute the rank of a matrix
- compute the trace of a matrix
- compute the vector product between a column of a matrix and a vector
- compute the (weighted) dot product between two columns of a matrix
- construct a biorthonormal basis (with respect to weighted scalar product)
- concatenate matrices
- estimate the condition number of a matrix
- get an element of a matrix
- multiplicate two matrices
- overwrite a matrix element
- overwrite a matrix row
- scale a matrix column
- transpose a matrix
- update columns of a matrix

and of the following vector operations

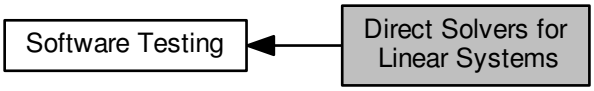
- add additional zeros to a vector
- compute the axpy scalar product.

Additionally the calculation of the machine epsilon is proofed.

8.66 Direct Solvers for Linear Systems

In this category implemented direct solver for linear systems are proofed.

Collaboration diagram for Direct Solvers for Linear Systems:



Files

- file check_backslash.c

  Check the solution of a system of linear equations.
- file check_det.c

  Check the det and detc functions of the solver.
- file check_dgeqp3.c

  Check the mess_direct_dgeqp3 function.
- file check_direct_care.c

  Check the mess_direct_care and mess_direct_care_res2 function.
- file check_getLU.c

  Check the getL and getU functions of the solver.
- file check_hessenberg.c

  Check the mess_direct_create_hessenberg_lu solver.
- file check_lapackqr.c

  Check functions of mess_direct_create_lapack_qr for over and underdetermined systems.
- file check_pinv.c

  Check the computation of the pseudoinverse of a matrix.
- file check_trisolve.c

  Check the functions in trisolve.c.
- file glyap_test.c

  Check the solution of generalized Lyapunov Equations.
- file gstein_test.c

  Check the solution of generalized Stein Equations.
- file inverse.c

  Check the computation of the inverse of a matrix.
- file linsolve.c

  Check the solution of a linear system of equations.
- file linsolve_multi.c

  Check the solution of a shifted linear system of equations.
- file linsolvem.c

  Check the solution of a linear system of equations (matrix version).
- file linsolvem_multi.c

  Check the solution of a shifted linear system of equations (matrix version).
- file tests/direct/sylvester_dense.c

  Check the solution of dense Sylvester equations.
- file tests/direct/sylvester_sparsedense.c

  Check the solution of a sparse-dense Sylvester equations with a small second matrix.

8.66.1 Detailed Description

There are files proving the correctness of the solution for

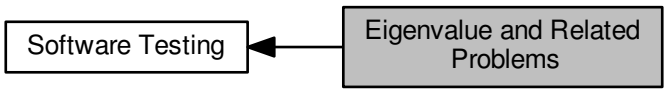
- a (generalized) Lyapunov equation
- a (generalized) Sylvester equation
- a matrix valued system of linear equations
- a (shifted) system of linear equations
- the computation of the inverse of a matrix.

Additionally function handles defined in trisolve.c are proofed.

8.67 Eigenvalue and Related Problems

In this category implemented files to compute eigenvalues, eigenvectors and solve related problems are proofed.

Collaboration diagram for Eigenvalue and Related Problems:



Files

- file check_arnoldi_template.c

  Check the mess_eigen_arnoldi_template function.
- file check_arnoldi_template_nrm.c

  Check the mess_eigen_arnoldi_template_nrm function.
- file check_arpack.c

  Check the solution of the eigenvalue problem using the Arnoldi process from `ARPACK`.
- file check_arpack_lanczos.c

  Check the solution of the eigenvalue problem using the Lanczos process from `ARPACK`.
- file check_eig.c

  Check the solution of the eigenvalue problem using xGEEV from `LAPACK`.
- file check_eig2.c

  Check the solution of the eigenvalue problem using xGEEV from `LAPACK` *(complex version).*
- file check_eig_complex.c

  Check the solution of the eigenvalue problem using xGEEV from `LAPACK` *(complex version).*
- file check_eig_hessenberg_schur.c

  Check the mess_eigen_hessenberg_schur, mess_eigen_schur_to_evd.
- file check_eig_hessenberg_schur_largest_ev.c

  Check the mess_eigen_hessenberg_abs_largest.
- file check_gsign.c

  Check the mess_eigen_gsign function.
- file check_schur.c

  Check the mess_eigen_schur and mess_eigen_schur_complex functions.
- file check_sign.c

  Check the mess_eigen_sign function.
- file check_svd.c

  Check the singular value decomposition.
- file check_svd_complex.c

  Check the singular value decomposition (complex version).
- file check_svd_econ.c

  Check the economy size singular value decomposition.
- file check_svd_econ_complex.c

  Check the economy size singular value decomposition (complex version).

8.67.1 Detailed Description

There are files proving the correctness of the solution of the eigenvalue problem using

- `DGEEV` from `LAPACK`
- the Arnoldi process
- the Lanczos process
- `xGEEV` from `LAPACK`.

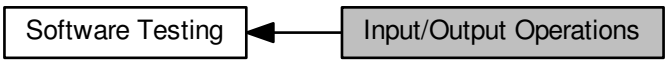
Additionally there are files that are proving the correctness of

- the computation of the largest eigenvalue
- the (economy size) singular value decomposition.

8.68 Input/Output Operations

This category contains checks of different input and output operations.

Collaboration diagram for Input/Output Operations:



Files

- file check_fputs_block.c

  Check writing of large blocks.
- file check_matrixio.c

  Check IO operations on mess_matrix instances.
- file check_vectorio.c

  Check IO operations on mess_vector instances.

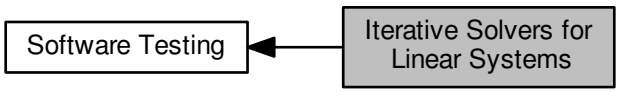
8.68.1 Detailed Description

In this category are files checking the correctness of input and output operations on vectors.

8.69 Iterative Solvers for Linear Systems

In this category implemented iterative solver for linear systems are proofed.

Collaboration diagram for Iterative Solvers for Linear Systems:



Files

- file check_itsolver.c

  Check different iterative solvers.
- file check_precond.c

  Check different preconditioner.

8.69.1 Detailed Description

There are files checking different iterative solver, e.g.

- BICGSTAB
- CG
- GMRES
- Gauss-Seidel method
- Jacobi method
- restarted GMRES
- (symmetric) successive overrelaxation ((S)SOR)

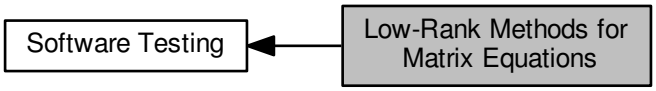
and different preconditioners, e.g.

- diagonal preconditioner
- $ILU(0)$ preconditioner
- $ILU(k)$ preconditioner
- (S)SOR preconditioner

8.70 Low-Rank Methods for Matrix Equations

This category contains checks of low-rank methods for different matrix equations.

Collaboration diagram for Low-Rank Methods for Matrix Equations:



Files

- file check_ber_sign_fac.c

  Check the computation of the solution of Bernoulli equations using matrix sign function and its residual.
- file check_dae2_handles_adi.c

  Check function handles for Hessenberg index 2 DAE systems, which are necessary for mess_lrcfadi_adi.
- file check_dae2_handles_para.c

  Check function handles for Hessenberg index 2 DAE systems, which are necessary for mess_lrcfadi_parameter.
- file check_dense_nm.c

  Check the dense Newton Method for dense algebraic Riccati equations.
- file check_dense_nm_tridiag.c

  Check the solution of a generalized Riccati Equation in transposed and nontransposed version using mess_dense↩_nm_gmpare.
- file check_linesearch.c

  Check the linesearch function defined in lrnm_linesearch.c.
- file check_lradi_dae1.c

  Check the solution of a Lyapunov Equation using the LRCF-ADI method and DAE 1 function handles.
- file check_lradi_dae2.c

  Check the solution of a Lyapunov Equation using the LRCF-ADI method and DAE 2 function handles.
- file check_lradi_glyap.c

  Check the solution of a Lyapunov Equation using the LRCF-ADI method.
- file check_lradi_so.c

  Check the solution of a Lyapunov Equation arising from a second order system.
- file check_lradi_unsym.c

  Check the solution of standard Lyapunov Equations using the LRCF-ADI method.
- file check_lrnm_dae1.c

  Check the solution of a Lyapunov Equation using the LRCF-ADI method and DAE 1 function handles.
- file check_lrnm_dae2.c

  Check the solution of generalized Riccati Equation arising from a Hessenberg index 2 DAE system using LRCF-NM.
- file check_lrnm_g.c

  Check the solution of a generalized Riccati Equation in transposed and nontransposed version using LRCF-NM.
- file check_lrnm_so.c

  Check the solution of a Riccati Equation arising from a second order system.
- file check_lrnm_unsym.c

  Check the solution of Riccati Equations using LRCF-NM.

- file check_lyap_sign_fac.c

  Check the computation of the solution of Lyapunov Equations using matrix sign function and its residual.
- file check_so1_handles.c

  Check function handles for second order systems (so1).
- file check_so2_handles.c

  Check function handles for second order systems (so2).

8.70.1 Detailed Description

There are files proving the correctness of the solution for

- generalized Riccati Equations arising from a Hessenberg index DAE system using LRCF-NM
- (generalized) Riccati Equations using LRCF-NM
- Lyapunov Equations arising from a second order system
- Lyapunov Equations using the LRCF-ADI method
- Riccati Equations arising from a second order system

and checking function handles for Hessenberg index 2 DAE and second order systems.

8.71 Storage Formats

This category contains checks of conversions to different storage formats.

Collaboration diagram for Storage Formats:

Files

- file check_cholmod.c

  Check cholmod.c functions.
- file check_cimport.c

  Check the creation of different storage type matrices from given arrays.
- file check_cimport_c2d.c

  Check the creation of a dense mess_matrix from a C 2 dimensional array.
- file check_convert.c

  Check the conversion subroutine.
- file check_mvpcall_matrix.c

  Check the mess_mvpcall_matrix function.
- file check_real_imag_part.c

  Check the mess_matrix_realpart mess_matrix_imagpart and mess_matrix_complex_from_parts functions.

8.71.1 Detailed Description

In this category are files that proof the correct conversion

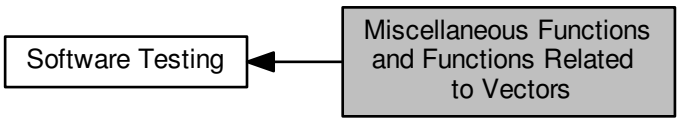
- from a C 2 dimensional array to a dense mess_matrix
- from a `cholmod_dense` or `cholmod_sparse` datastructure to a dense or MESS_CSC mess_matrix storage format (and vice versa)
- from an arbitrary mess_matrix storage format to another mess_matrix storage format
- to a MESS_CSC, MESS_CSR or MESS_COORD mess_matrix.

There is also a file checking the correctness of the computed view on a matrix.

8.72 Miscellaneous Functions and Functions Related to Vectors

This category contains checks for misscellaneous function and function related to vectors.

Collaboration diagram for Miscellaneous Functions and Functions Related to Vectors:



Files

- file check_freelist.c

  Check the mess_freelist_t structure.
- file check_vector_perm.c

  Check the permutation function for vectors.

8.72.1 Detailed Description

In this category are files that checks the correctness of

- mess_vector_perm_inplace

8.73 Interfaces_matlab

Macros

- #define FUNCTION_FAILURE_HANDLE_MEX(err, cond, fun) if ((cond)) { char text[1024]; snprintf(text,1023," %s returned with %d - %s\n", #fun, err, mess_get_error(err)); mexErrMsgTxt(text); }

  Function failure handler for MATLAB.
- #define mess_check_struct(X, str) if (!mxIsStruct(X)) { csc_error_message(str); return MESS_ERROR_A↩RGUMENTS; }

  Check if the given mxArray is a structure.

Functions

- char * mess_classname_from_mexmess (const mxArray *instance)

  Get a class name from MATLAB *object.*
- double mess_double_from_mexmess (const mxArray *Amatlab)

  Get a real valued scalar from MATLAB.
- int mxIsSubClass (const mxArray *instance, char *class)
- mxArray * mxCreateComplexScalar (mess_double_cpx_t s)
- mess_double_cpx_t mess_complex_from_mexmess (const mxArray *Amatlab)

  Get a complex valued scalar from MATLAB.
- void init_mexmess ()

  Initialize the ***M.E.S.S.*** *to be used in a MeX function.*
- mess_direct_cholpackage_t mess_direct_cholpackage_t_from_mexmess (const mxArray *instance)
- mess_direct_lupackage_t mess_direct_lupackage_t_from_mexmess (const mxArray *instance)
- mess_equation_t mess_equation_t_from_mexmess (const mxArray *instance)
- mess_memusage_t mess_memusage_t_from_mexmess (const mxArray *instance)
- mess_multidirect_t mess_multidirect_t_from_mexmess (const mxArray *instance)
- mess_operation_t mess_operation_t_from_mexmess (const mxArray *instance)
- mess_parameter_t mess_parameter_t_from_mexmess (const mxArray *instance)
- mess_residual_t mess_residual_t_from_mexmess (const mxArray *instance)
- mess_norm_t mess_norm_t_from_mexmess (const mxArray *instance)
- mxArray * mess_direct_cholpackage_t_to_mexmess (mess_direct_cholpackage_t chol_t)
- mxArray * mess_direct_lupackage_t_to_mexmess (mess_direct_lupackage_t chol_t)
- mxArray * mess_equation_t_to_mexmess (mess_equation_t eqn_t)
- mxArray * mess_memusage_t_to_mexmess (mess_memusage_t mem)
- mxArray * mess_multidirect_t_to_mexmess (mess_multidirect_t mdirect_t)
- mxArray * mess_operation_t_to_mexmess (mess_operation_t op)
- mxArray * mess_parameter_t_to_mexmess (mess_parameter_t para)
- mxArray * mess_residual_t_to_mexmess (mess_residual_t res_t)
- mxArray * mess_norm_t_to_mexmess (mess_norm_t nrm_t)
- mess_options mess_options_from_mexmess (const mxArray *instance)
- mxArray * mess_options_to_mexmess (mess_options opt)
- mxArray * mess_status_to_mexmess (mess_status stat)
- int mess_callback_equation_mexmess (mess_equation eqn, mess_options opt, const mxArray *meqn)
- mess_equation eqn_conv_lyap (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_lyap_dae1 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_lyap_dae2 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_lyap_so1 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_lyap_so2 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric_dae1 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric_dae2 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)

- mess_equation eqn_conv_ric_so1 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric_so2 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation mess_equation_from_mexmess (mxArray *m_eqn, mxArray *m_opt, mess_freelist mem, mess_equation_t eqn_type)
- mess_matrix mess_matrix_from_mexmess (const mxArray *Amatlab)

  Copy a matrix from MATLAB *to a mess_matrix.*
- mxArray * mess_matrix_to_mexmess (mess_matrix A)

  Copy a matrix from ***M.E.S.S.*** *to* MATLAB.
- mess_vector mess_vector_from_mexmess (const mxArray *Amatlab)

  Copy a vector/matrix from MATLAB *to a column vector in* ***M.E.S.S.***.
- mxArray * mess_vector_to_mexmess (mess_vector A)

  Copy a vector from ***M.E.S.S.*** *to* MATLAB.
- void mex_mess_direct_lu_select (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_direct_chol_select (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_multidirect_select (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_amd (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_arpack (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_bzip2 (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_cholmod (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_colamd (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_csparse (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_matio (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_mess64 (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_mklpardiso (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_openmp (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_superlu (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_umfpack (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_zlib (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_is_debug (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_version (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version_major (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version_minor (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version_patch (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version_verbose (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_git_id (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_git_branch (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_lyap (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_care (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_dense_nm_gmpare (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_lrnm (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_lradi (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_sylvester_sparsedense (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_sylvester_dense (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_glyap (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_gstein (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_direct_cholpackage_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_direct_lupackage_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_equation_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_memusage_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_multidirect_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_operation_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_parameter_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_residual_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_norm_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_vector (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_matrix (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_status (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_options (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])

8.73.1 Detailed Description

8.73.2 Macro Definition Documentation

8.73.2.1 #define FUNCTION_FAILURE_HANDLE_MEX(*err, cond, fun*) if ((cond)) { char text[1024]; snprintf(text,1023," %s returned with %d - %s\n", #fun, err, mess_get_error(err)); mexErrMsgTxt(text); }

Function failure handler for MATLAB

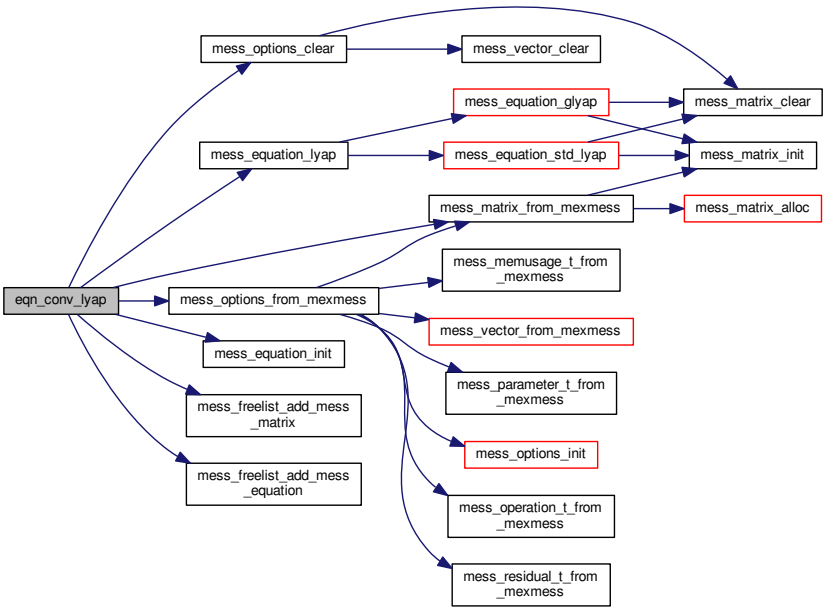
8.73.2.2 #define mess_check_struct(X, *str*) if (!mxIsStruct(X)) { csc_error_message(str); return MESS_ERROR_ARGUMENTS; }

The macro forces to be X an instance of a MATLAB structure otherwise an error is printed.

8.73.3 Function Documentation

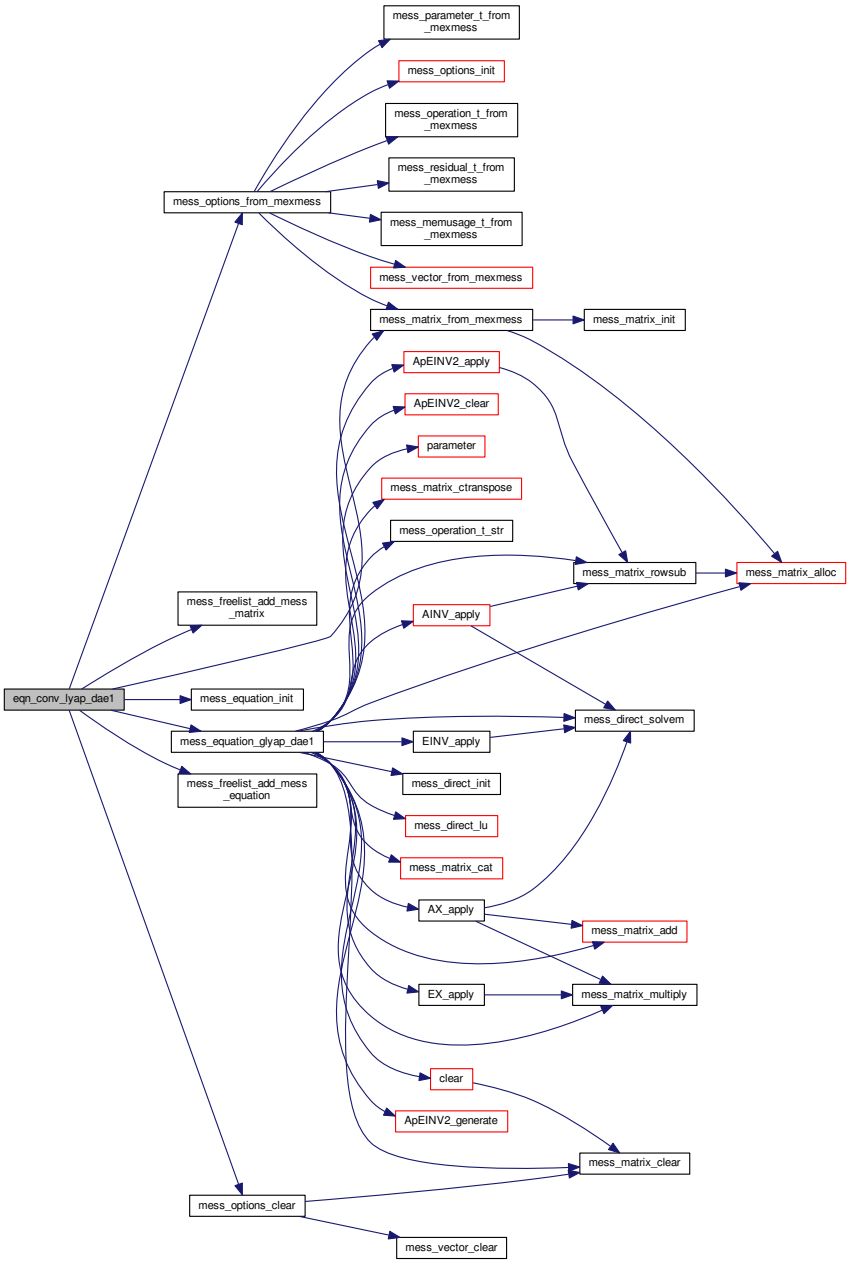
8.73.3.1 mess_equation eqn_conv_lyap (const mxArray * *m_eqn*, const mxArray * *m_opt*, mess_freelist *mem*)

Here is the call graph for this function:



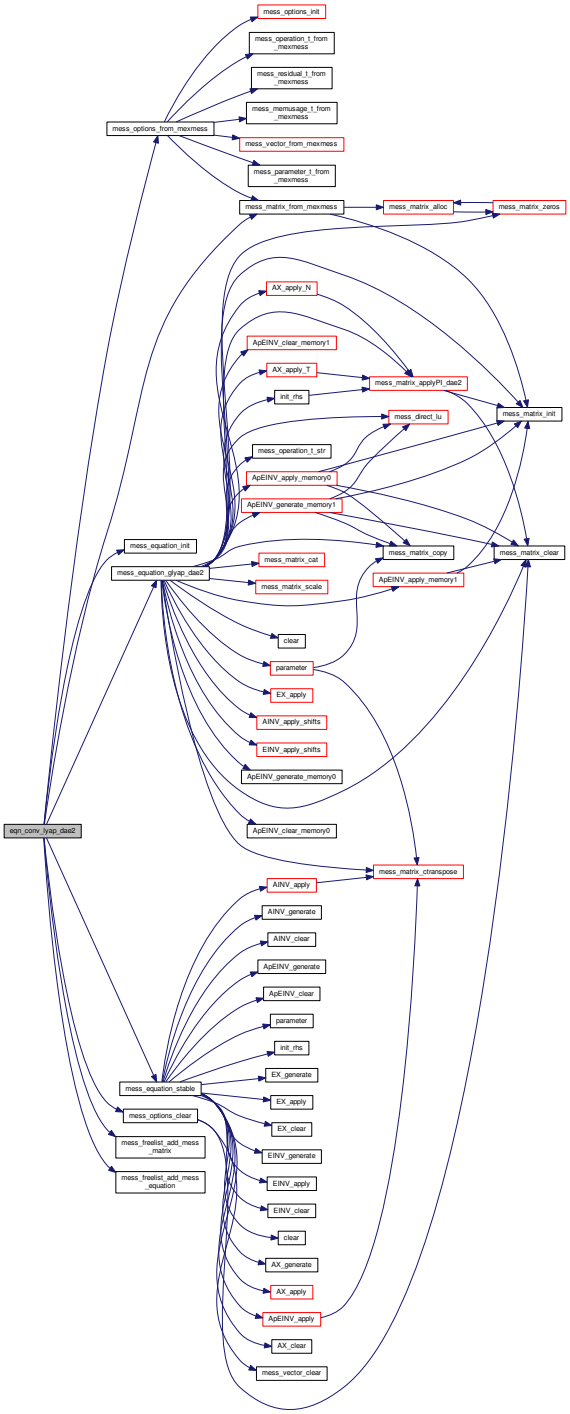
8.73.3.2 mess_equation eqn_conv_lyap_dae1 (const mxArray * *m_eqn,* const mxArray * *m_opt,* mess_freelist *mem*)

Here is the call graph for this function:



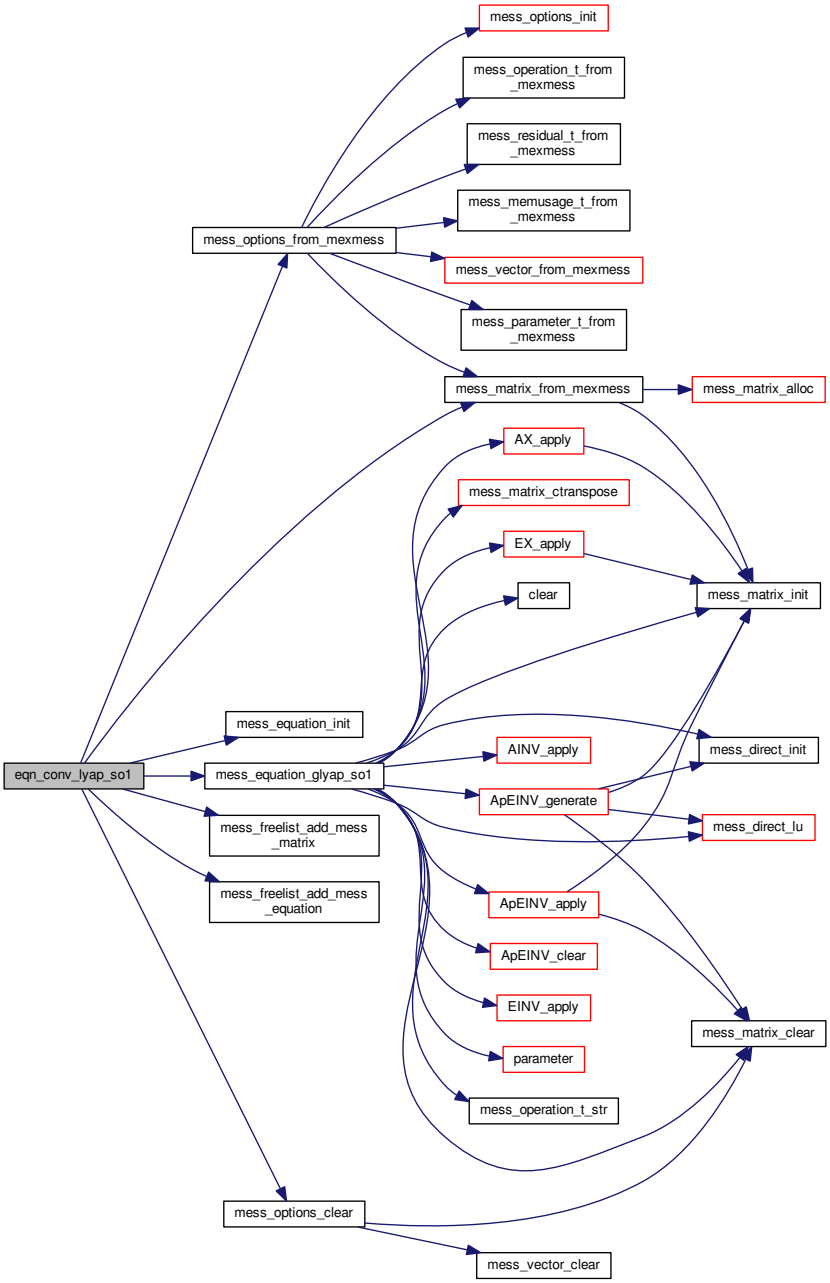
8.73.3.3 mess_equation eqn_conv_lyap_dae2 (const mxArray * *m_eqn,* const mxArray * *m_opt,* mess_freelist *mem*)

Here is the call graph for this function:



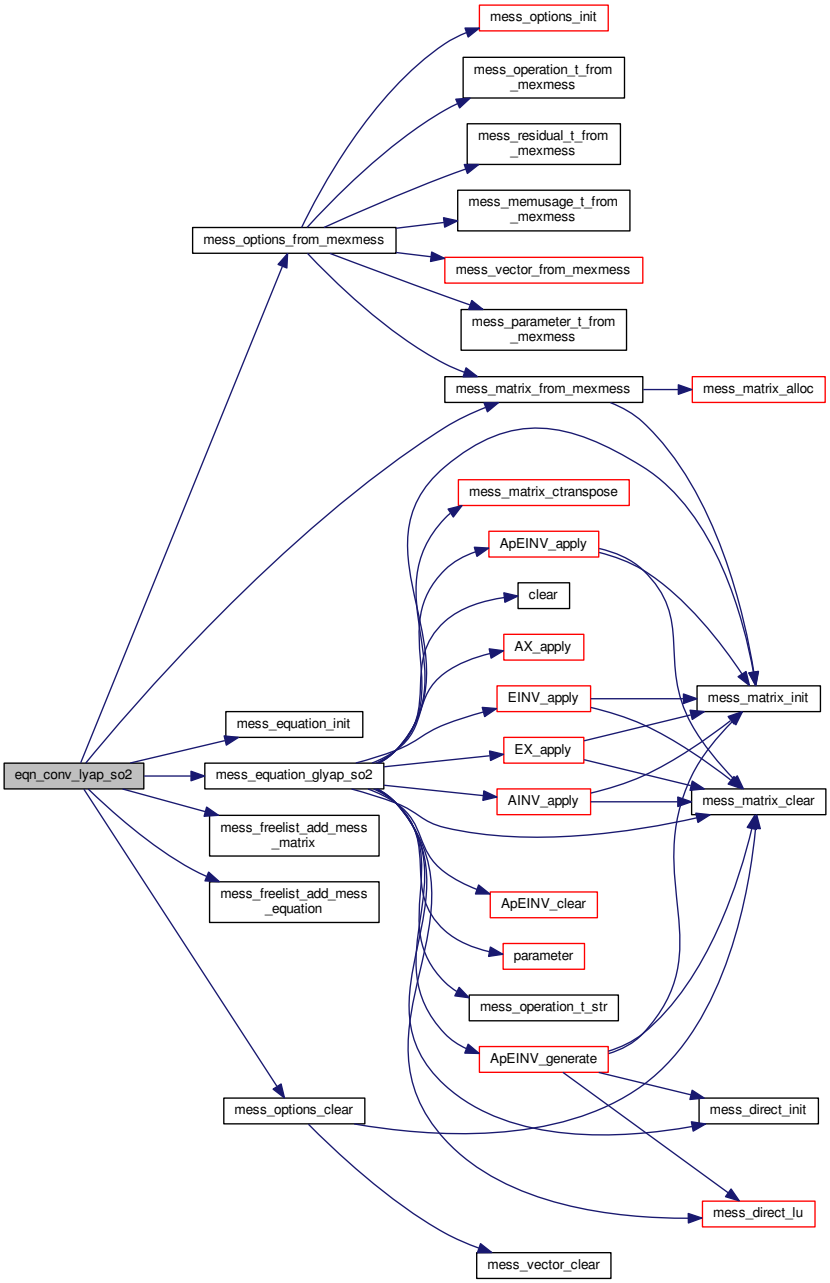
8.73.3.4 mess_equation eqn_conv_lyap_so1 (const mxArray * *m_eqn,* const mxArray * *m_opt,* mess_freelist *mem*)

Here is the call graph for this function:



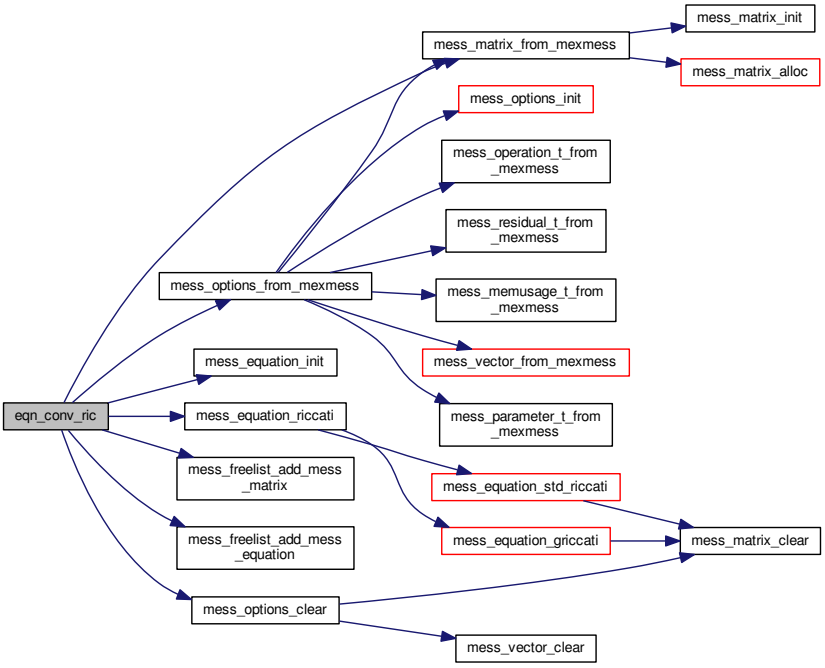
8.73.3.5 mess_equation eqn_conv_lyap_so2 (const mxArray * *m_eqn,* const mxArray * *m_opt,* mess_freelist *mem*)

Here is the call graph for this function:



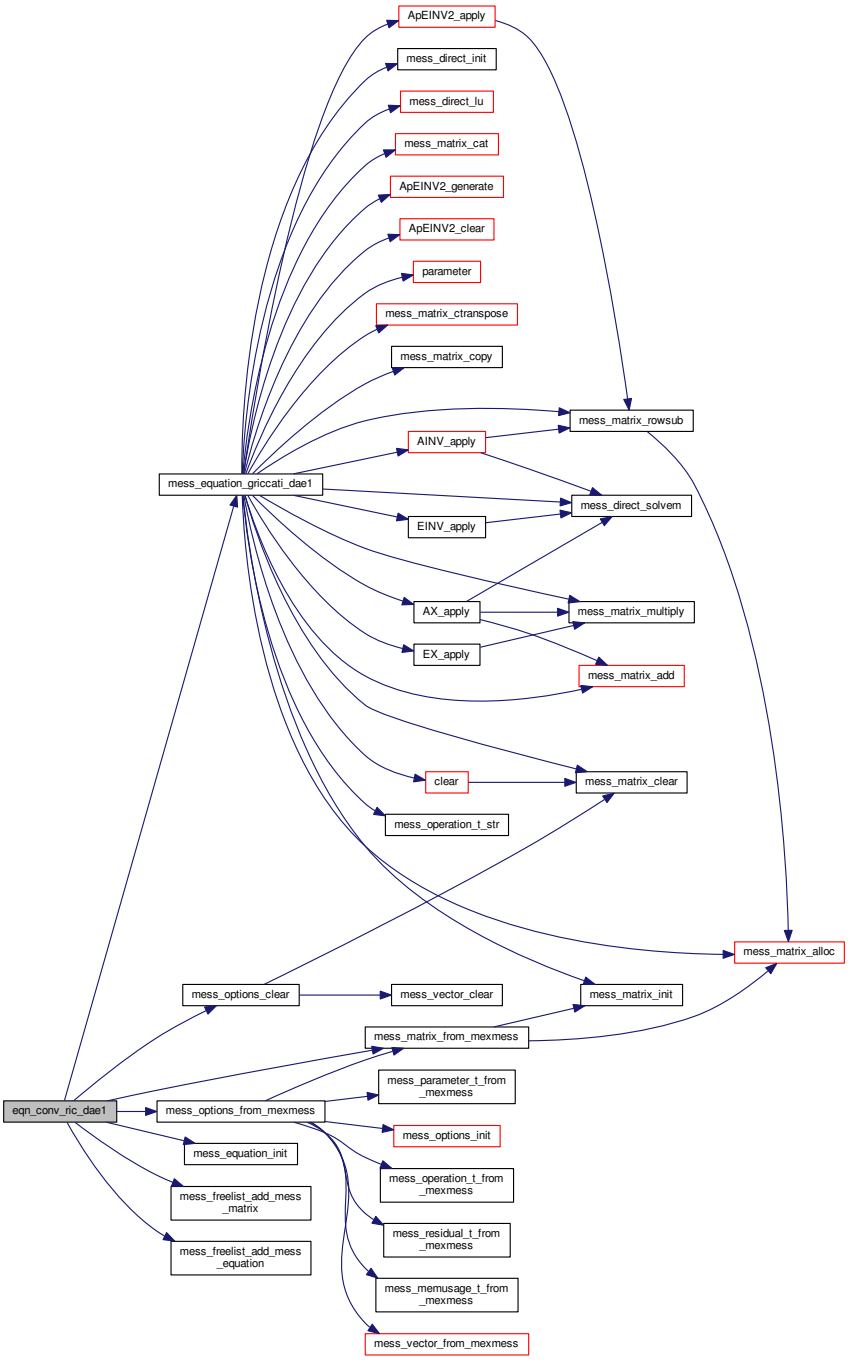
8.73.3.6 mess_equation eqn_conv_ric (const mxArray * *m_eqn,* const mxArray * *m_opt,* mess_freelist *mem*)

Here is the call graph for this function:



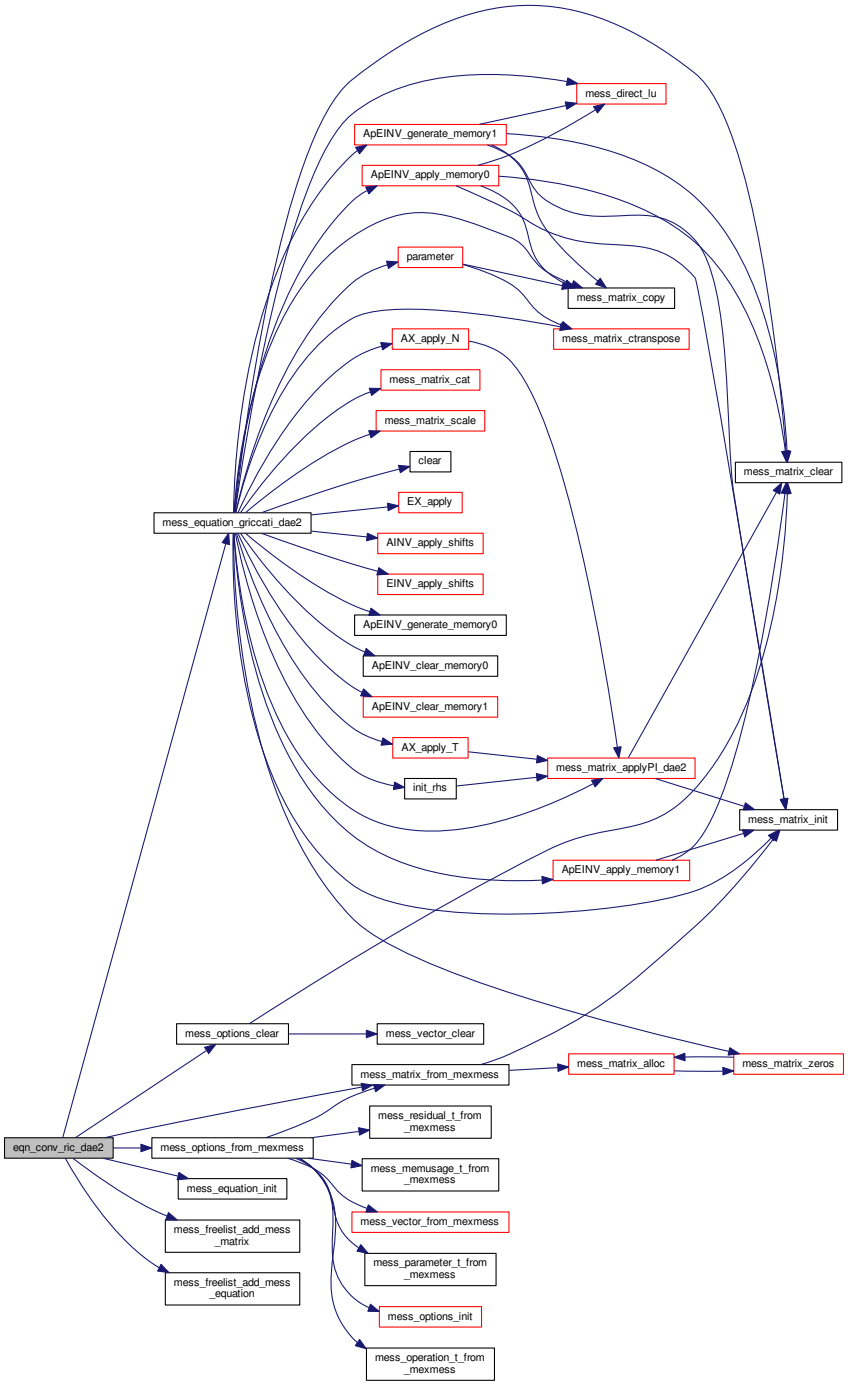
8.73.3.7 mess_equation eqn_conv_ric_dae1 (const mxArray * *m_eqn,* const mxArray * *m_opt,* mess_freelist *mem*)

Here is the call graph for this function:



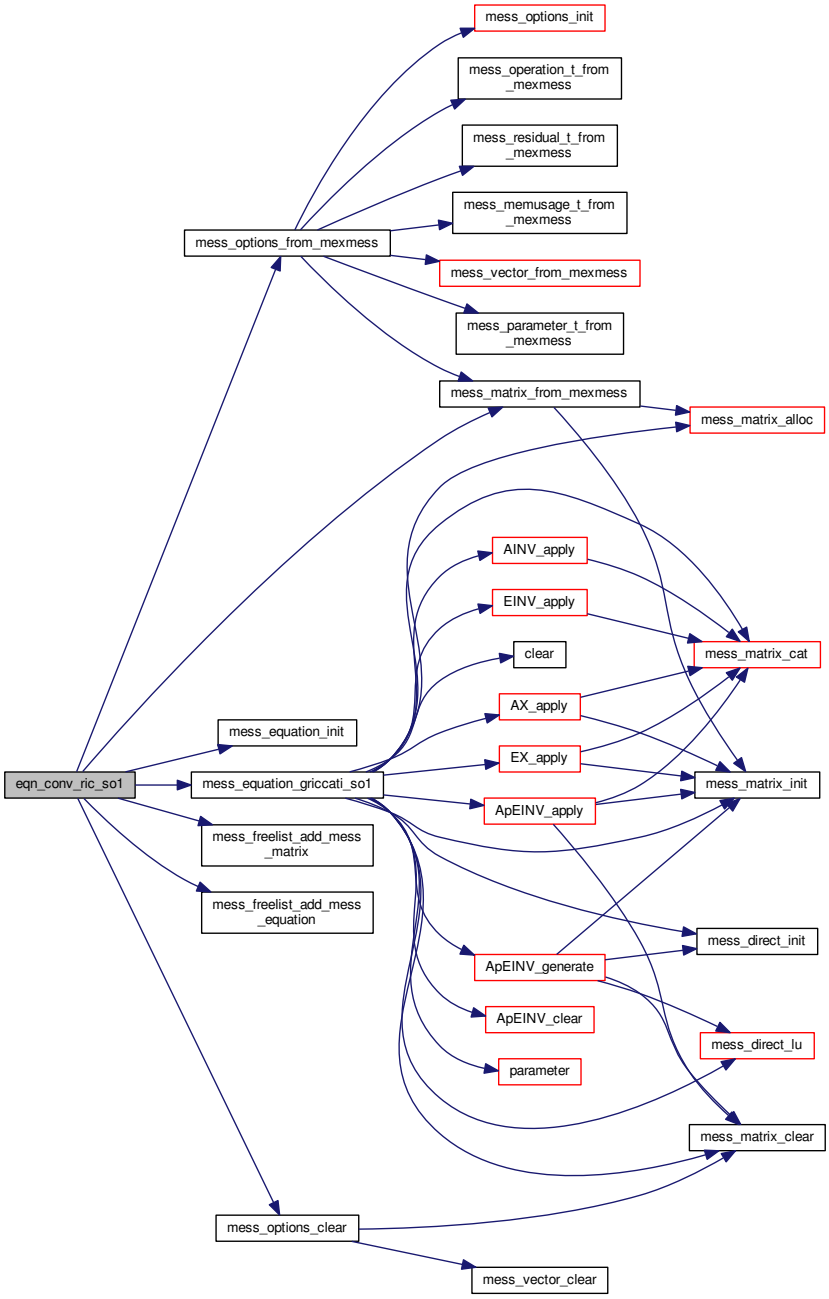
8.73.3.8 mess_equation eqn_conv_ric_dae2 (const mxArray * *m_eqn,* const mxArray * *m_opt,* mess_freelist *mem*)

Here is the call graph for this function:



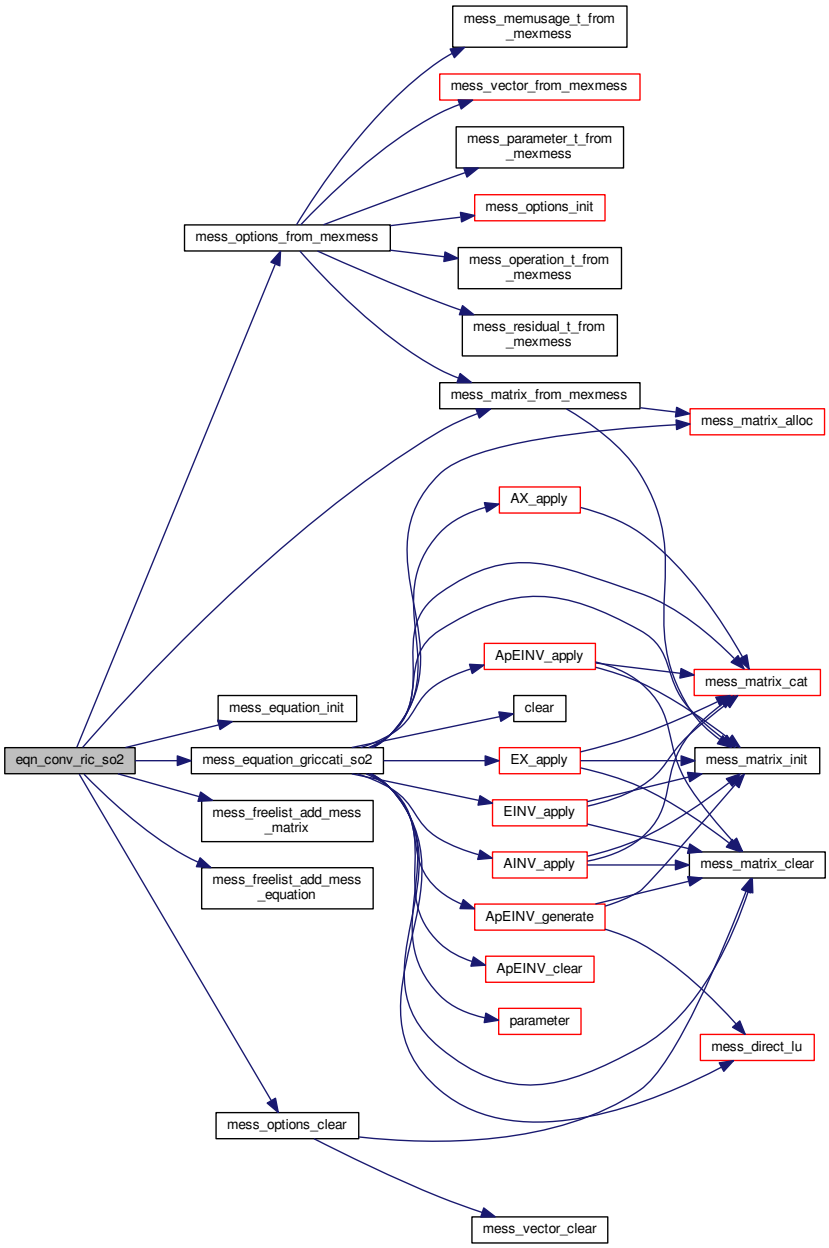
8.73.3.9 mess_equation eqn_conv_ric_so1 (const mxArray * *m_eqn*, const mxArray * *m_opt*, mess_freelist *mem*)

Here is the call graph for this function:



8.73.3.10 mess_equation eqn_conv_ric_so2 (const mxArray * *m_eqn,* const mxArray * *m_opt,* mess_freelist *mem*)

Here is the call graph for this function:



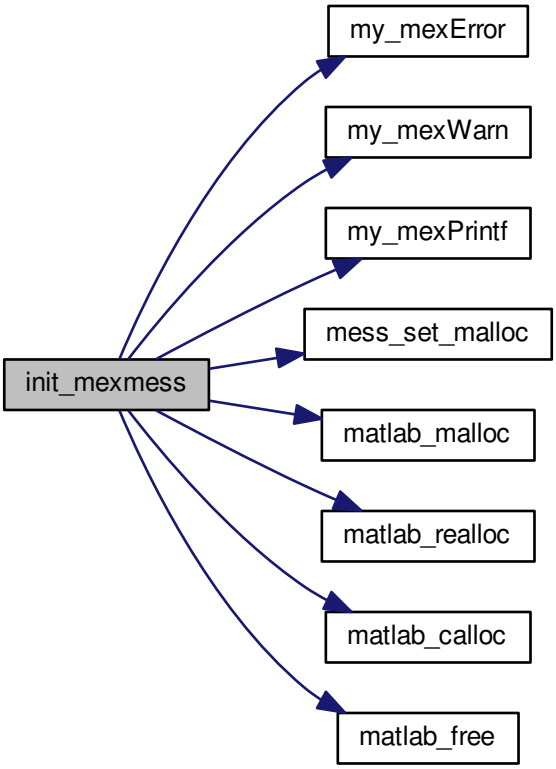
8.73.3.11 void init_mexmess ( )

Returns

zero on success or a non zero error code otherwise

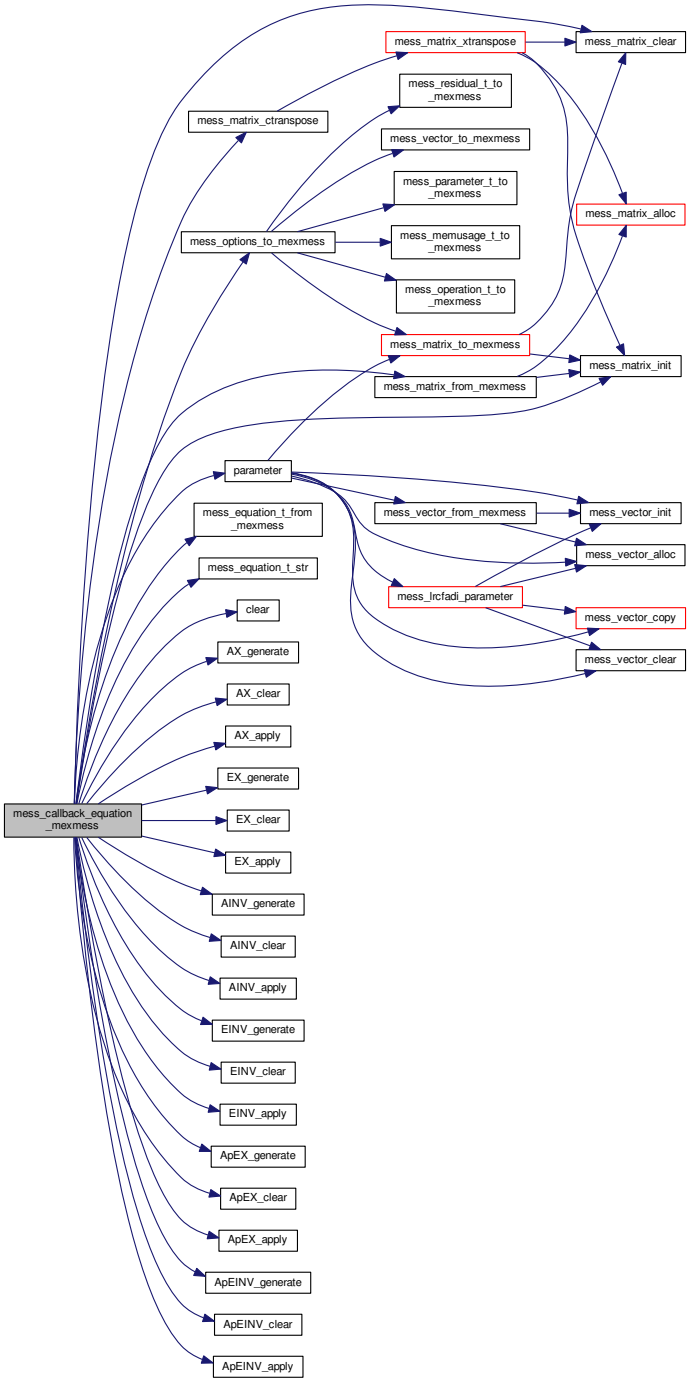
The init_mexmess function initializes the mex helper functions. It needs to called once at the begin of a mex file.

Here is the call graph for this function:



8.73.3.12 int mess_callback_equation_mexmess (mess_equation *eqn*, mess_options *opt*, const mxArray * *meqn*)

Here is the call graph for this function:



8.73.3.13 char* mess_classname_from_mexmess (const mxArray * *instance*)

Parameters

in	*instance*	input instance of class

Returns

pointer of charaters with class name or NULL if failure occurs.

The mess_classname_from_mexmess get the classname from a MATLAB instance.

8.73.3.14 mess_double_cpx_t mess_complex_from_mexmess (const mxArray * *Amatlab*)

Parameters

in	*Amatlab*	input pointer to a mxArray

Returns

first complex double value from $Amatlab$

The mess_complex_from_mexmess function get a complex double value from MATLAB .

8.73.3.15 mess_direct_cholpackage_t mess_direct_cholpackage_t_from_mexmess (const mxArray * *instance*)

8.73.3.16 mxArray* mess_direct_cholpackage_t_to_mexmess (mess_direct_cholpackage_t *chol_t*)

8.73.3.17 mess_direct_lupackage_t mess_direct_lupackage_t_from_mexmess (const mxArray * *instance*)

8.73.3.18 mxArray* mess_direct_lupackage_t_to_mexmess (mess_direct_lupackage_t *chol_t*)

8.73.3.19 double mess_double_from_mexmess (const mxArray * *Amatlab*)

Parameters

in	*Amatlab*	input pointer to a mxArray

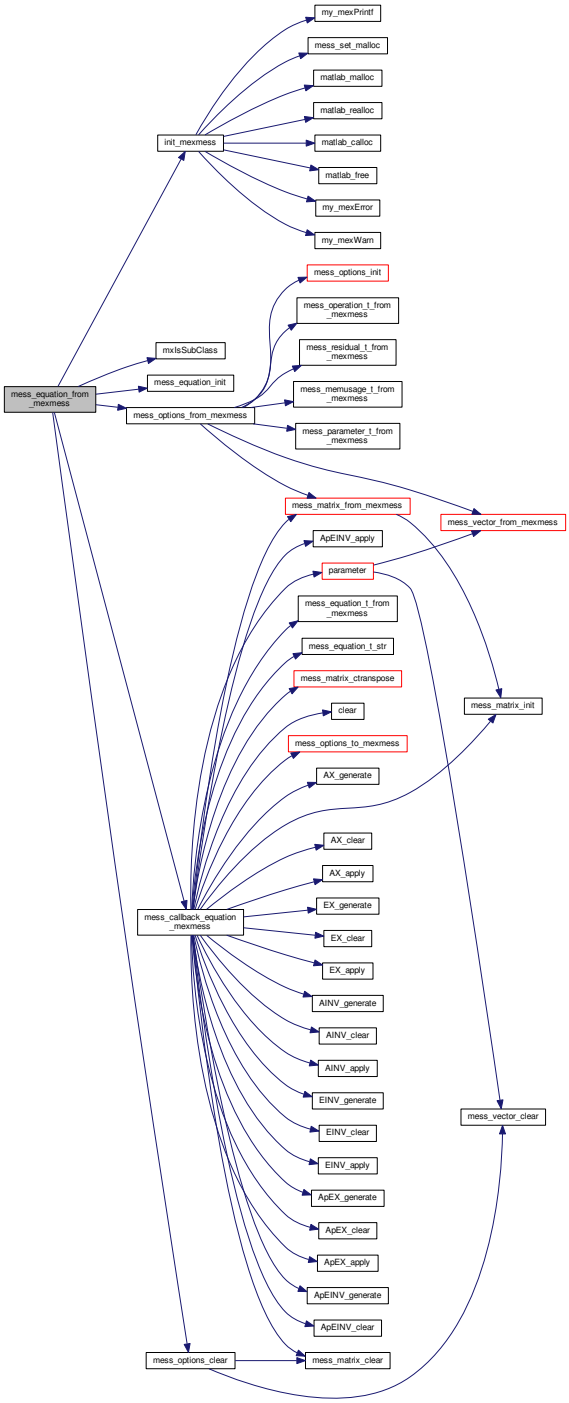
Returns

first real double value from $Amatlab$

The mess_double_from_mexmess function get a real double value from MATLAB .

8.73.3.20 mess_equation mess_equation_from_mexmess (mxArray * *m_eqn,* mxArray * *m_opt,* mess_freelist *mem,* mess_equation_t *eqn_type*)

Here is the call graph for this function:



8.73.3.21 mess_equation_t mess_equation_t_from_mexmess (const mxArray * *instance*)

8.73.3.22 mxArray* mess_equation_t_to_mexmess (mess_equation_t *eqn_t*)

8.73.3.23 mess_matrix mess_matrix_from_mexmess (const mxArray * *Amatlab*)

Parameters

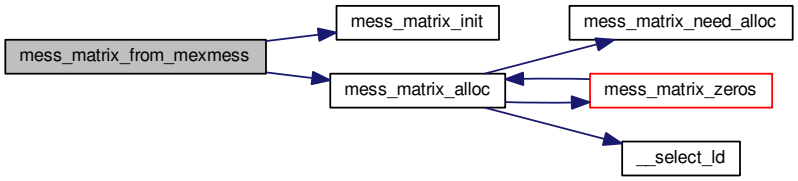
in	*Amatlab*	input MATLAB matrix

Returns

NULL if an error occured or the matrix is empty or a mess_matrix with the content copied from the MATLAB object

The mess_matrix_from_mexmess function copies a matrix from MATLAB to **M.E.S.S.**.
If the matrix is sparse data is stored in the compressed column format.

Here is the call graph for this function:



8.73.3.24 mxArray* mess_matrix_to_mexmess (mess_matrix *A*)

Parameters

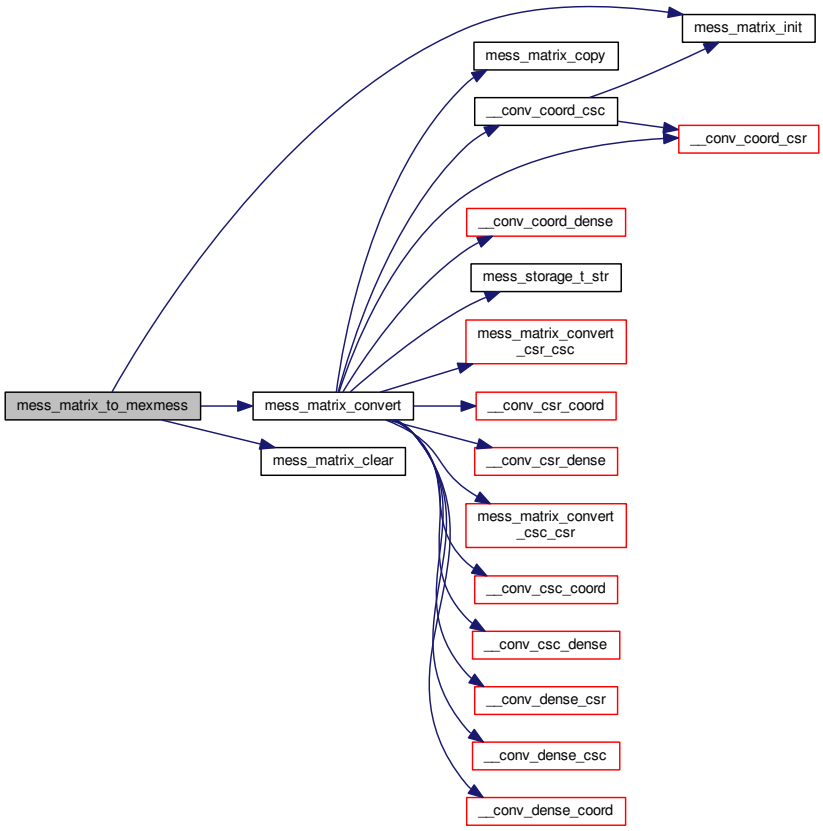
in	*A*	input mess_matrix

Returns

NULL in case of an error or a MATLAB mxArray containing the input matrix copied into the MATLAB format.

The mess_matrix_to_mexmess function copies a matrix from **M.E.S.S.** to MATLAB .

Here is the call graph for this function:



8.73.3.25 mess_memusage_t mess_memusage_t_from_mexmess (const mxArray * *instance*)

8.73.3.26 mxArray* mess_memusage_t_to_mexmess (mess_memusage_t *mem*)

8.73.3.27 mess_multidirect_t mess_multidirect_t_from_mexmess (const mxArray * *instance*)

8.73.3.28 mxArray* mess_multidirect_t_to_mexmess (mess_multidirect_t *mdirect_t*)

8.73.3.29 mess_norm_t mess_norm_t_from_mexmess (const mxArray * *instance*)

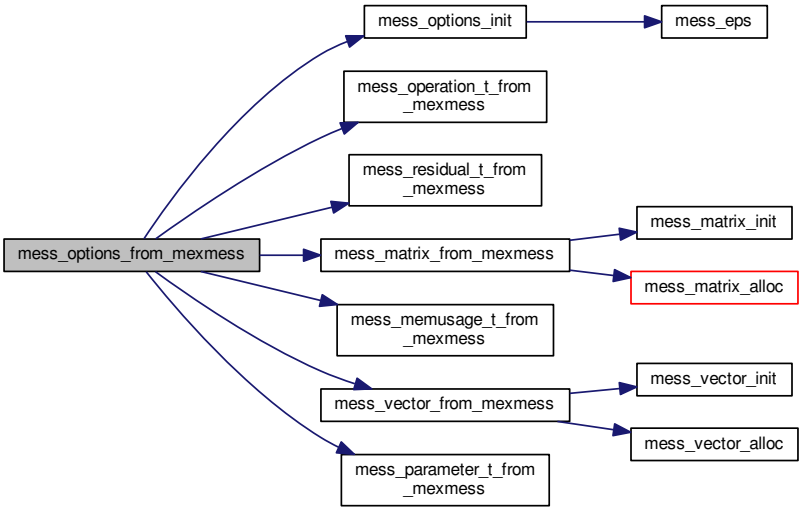
8.73.3.30 mxArray* mess_norm_t_to_mexmess (mess_norm_t *nrm_t*)

8.73.3.31 mess_operation_t mess_operation_t_from_mexmess (const mxArray * *instance*)

8.73.3.32 mxArray* mess_operation_t_to_mexmess (mess_operation_t *op*)

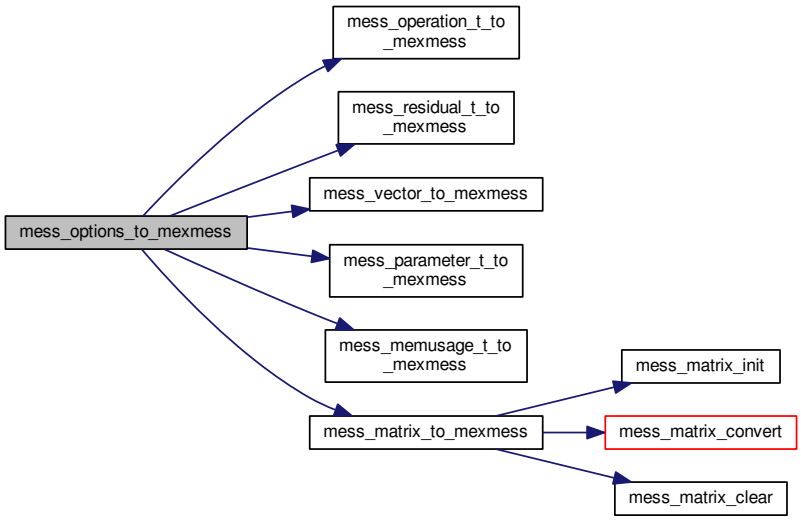
8.73.3.33 mess_options mess_options_from_mexmess (const mxArray * *instance*)

Here is the call graph for this function:



8.73.3.34 mxArray* mess_options_to_mexmess (mess_options *opt*)

Here is the call graph for this function:



8.73.3.35 mess_parameter_t mess_parameter_t_from_mexmess (const mxArray * *instance*)

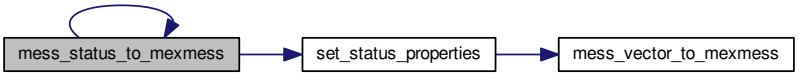
8.73.3.36 mxArray* mess_parameter_t_to_mexmess (mess_parameter_t *para*)

8.73.3.37 mess_residual_t mess_residual_t_from_mexmess (const mxArray * *instance*)

8.73.3.38 mxArray* mess_residual_t_to_mexmess (mess_residual_t *res_t*)

8.73.3.39 mxArray* mess_status_to_mexmess (mess_status *stat*)

Here is the call graph for this function:



8.73.3.40 mess_vector mess_vector_from_mexmess (const mxArray * *Amatlab*)

Parameters

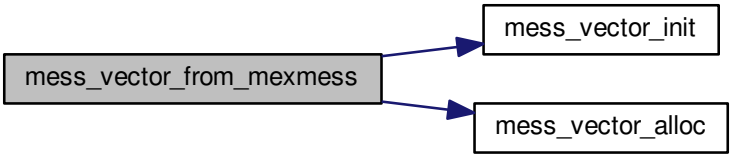
in	*Amatlab*	input MATLAB object

Returns

NULL if an error occured or the vector is empty or a mess_vector with the content copied from the MATLAB object

The mess_vector_from_mexmess function copies a vector or a matrix from MATLAB to **M.E.S.S.**.
Data is stored in a column-wise way if the input is a matrix.

Here is the call graph for this function:



8.73.3.41 mxArray* mess_vector_to_mexmess (mess_vector *A*)

Parameters

in	*A*	input MESS vector

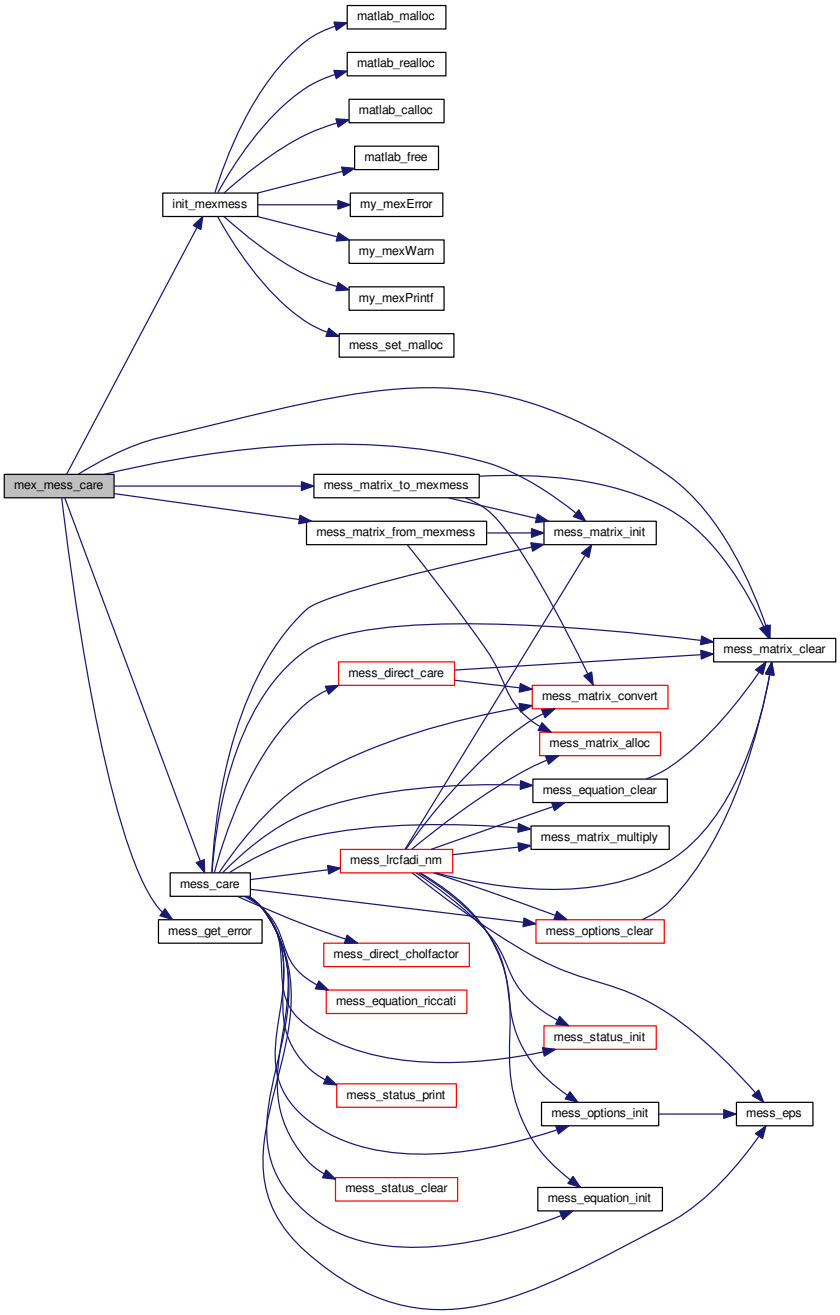
Returns

the MATLAB vector representaion of the input vector.

The mess_vector_to_mexmess function copies a vector from **M.E.S.S.** to a column vector in MATLAB.

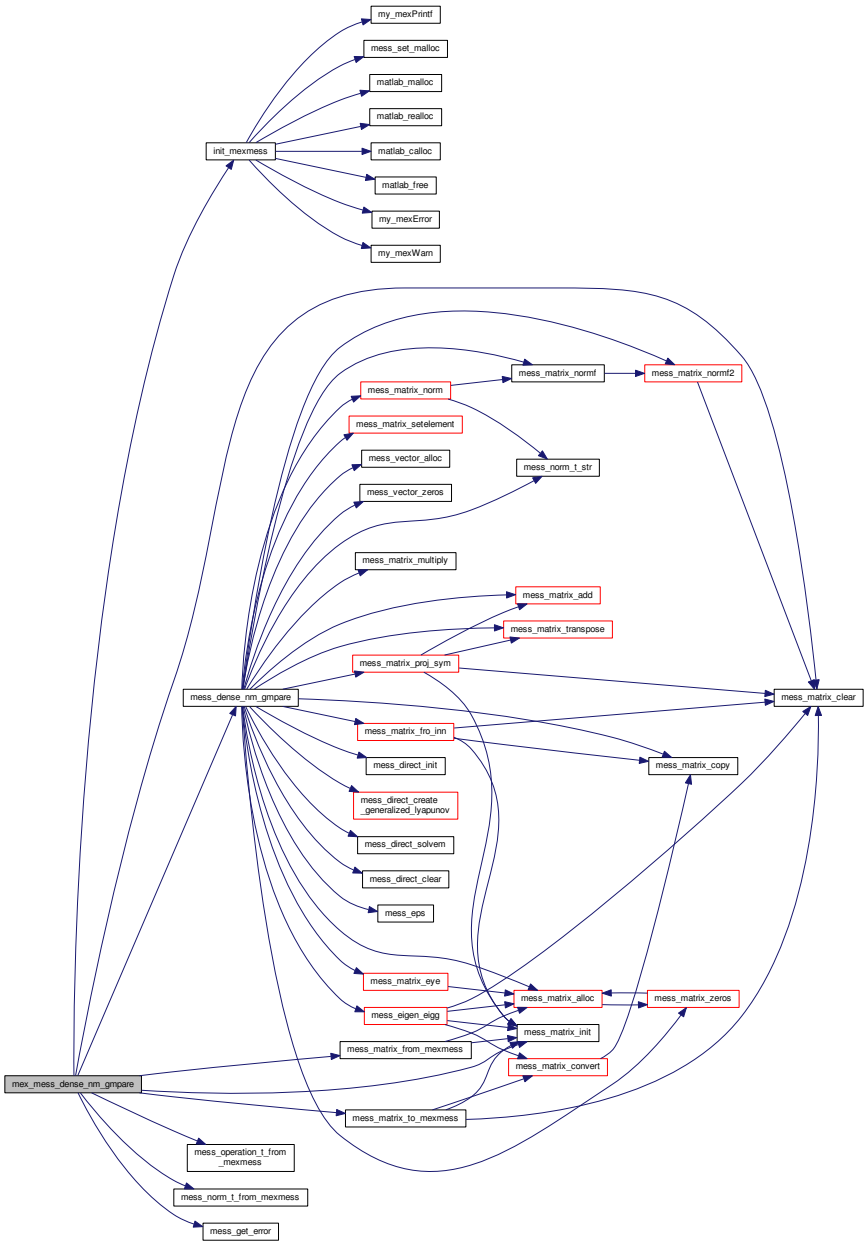
8.73.3.42 void mex_mess_care (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



8.73.3.43 void mex_mess_dense_nm_gmpare (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



8.73.3.44 void mex_mess_direct_chol_select (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

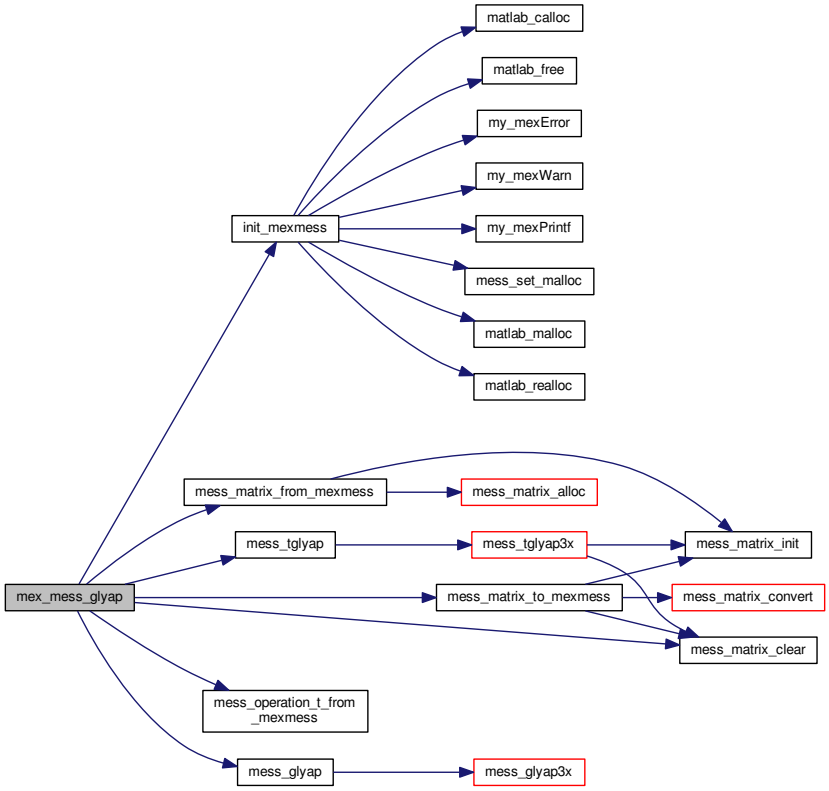
8.73.3.45 void mex_mess_direct_lu_select (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.46 void mex_mess_git_branch (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

8.73.3.47 void mex_mess_git_id (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

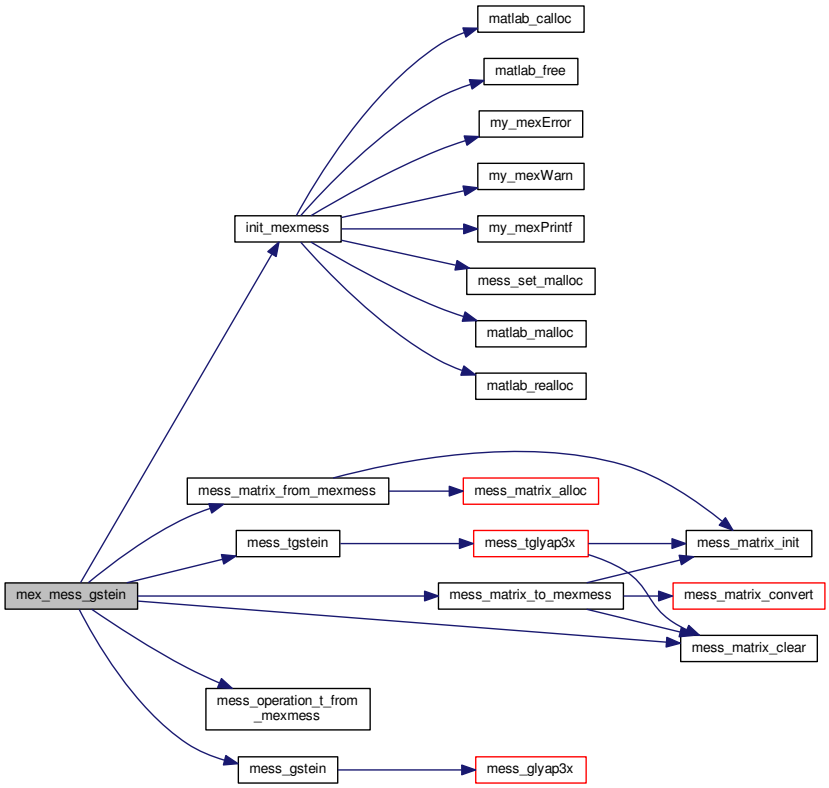
8.73.3.48 void mex_mess_glyap (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



8.73.3.49 void mex_mess_gstein (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



8.73.3.50 void mex_mess_have_amd (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.51 void mex_mess_have_arpack (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.52 void mex_mess_have_bzip2 (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.53 void mex_mess_have_cholmod (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.54 void mex_mess_have_colamd (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.55 void mex_mess_have_csparse (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.56 void mex_mess_have_matio (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.57 void mex_mess_have_mess64 (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.58 void mex_mess_have_mklpardiso (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.59 void mex_mess_have_openmp (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.60 void mex_mess_have_superlu (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

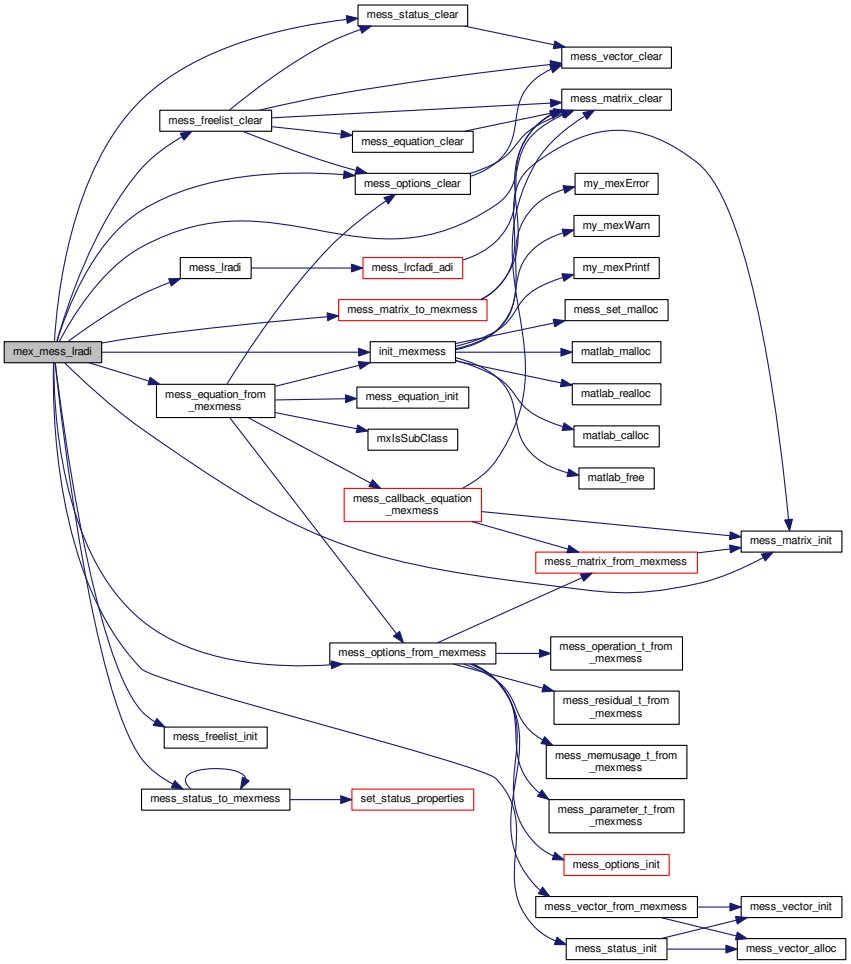
8.73.3.61 void mex_mess_have_umfpack (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.62 void mex_mess_have_zlib (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

8.73.3.63 void mex_mess_is_debug (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

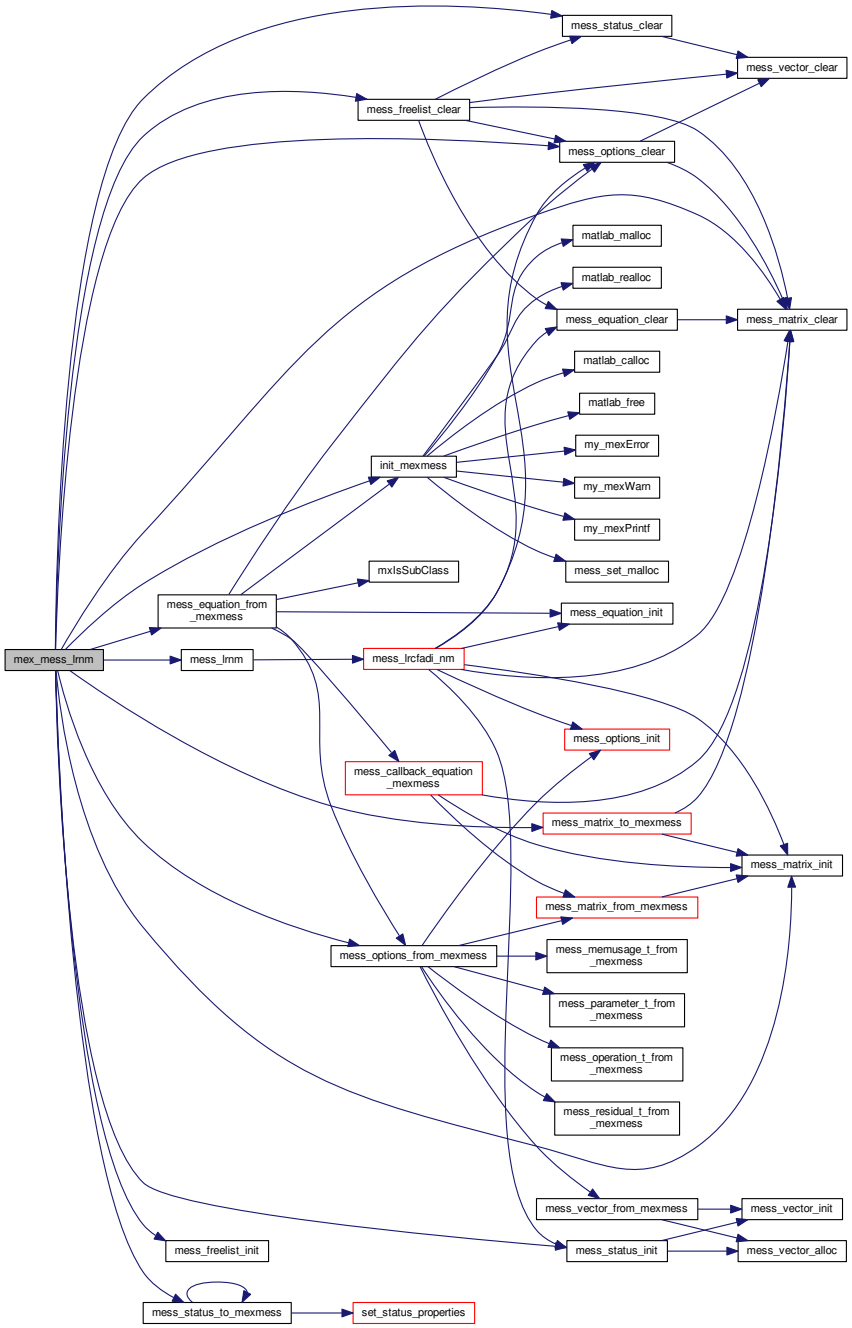
8.73.3.64 void mex_mess_lradi (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



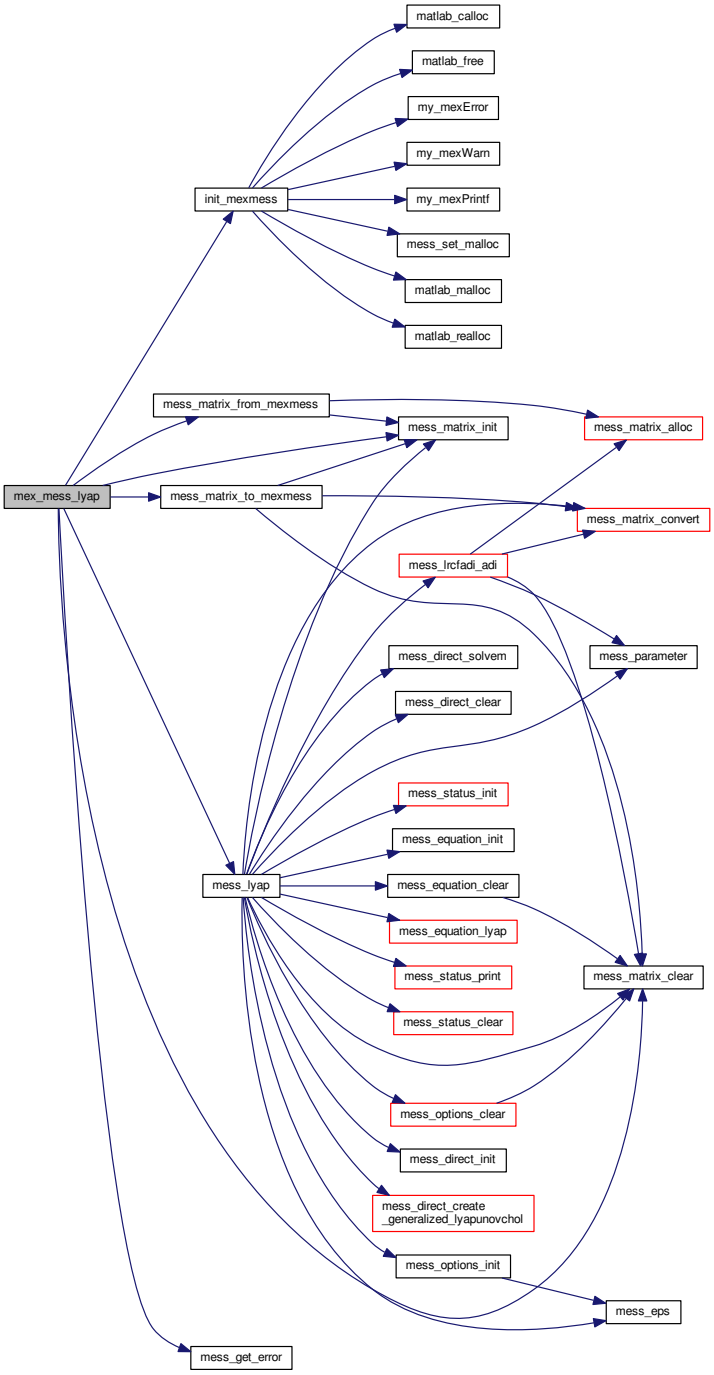
8.73.3.65 void mex_mess_lrnm (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



8.73.3.66 void mex_mess_lyap (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

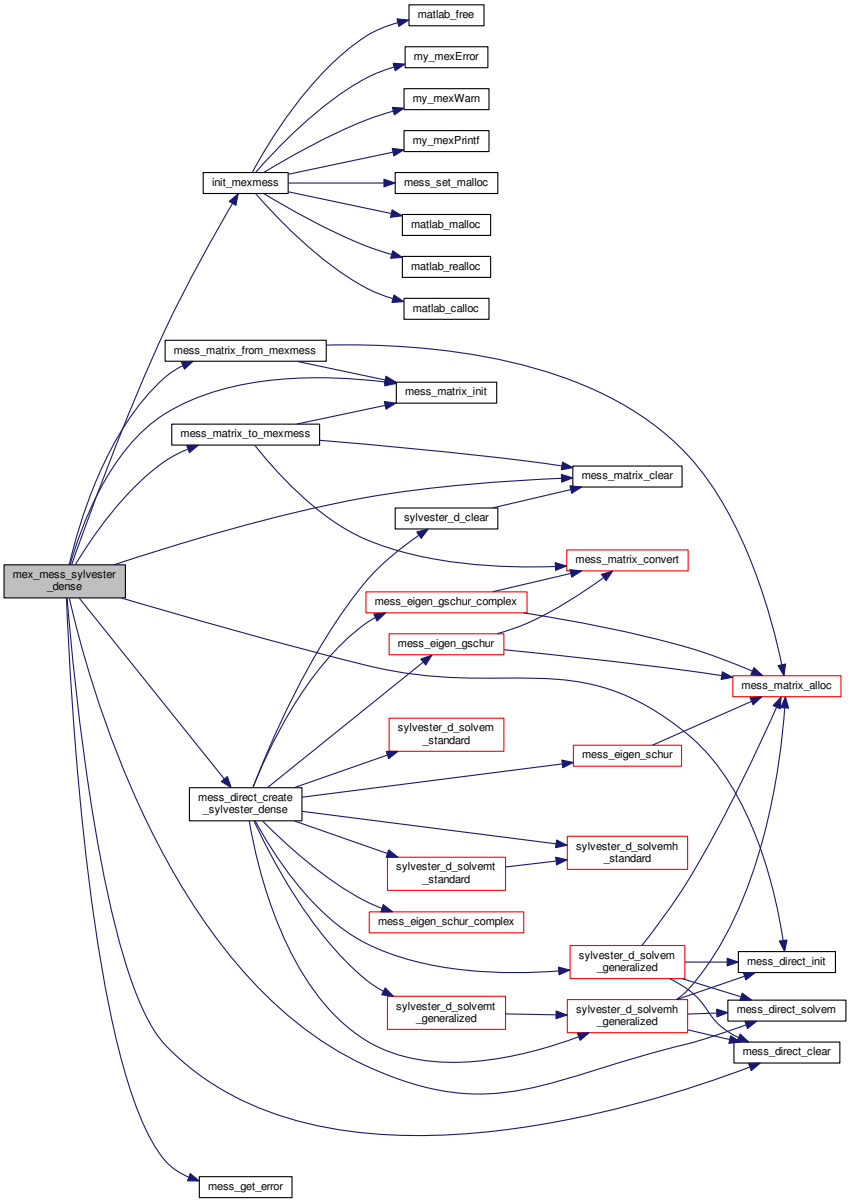
Here is the call graph for this function:



8.73.3.67 void mex_mess_multidirect_select (int *nlhs*, mxArray * *plhs[ ]*, int *nrsh*, const mxArray * *prhs[ ]*)

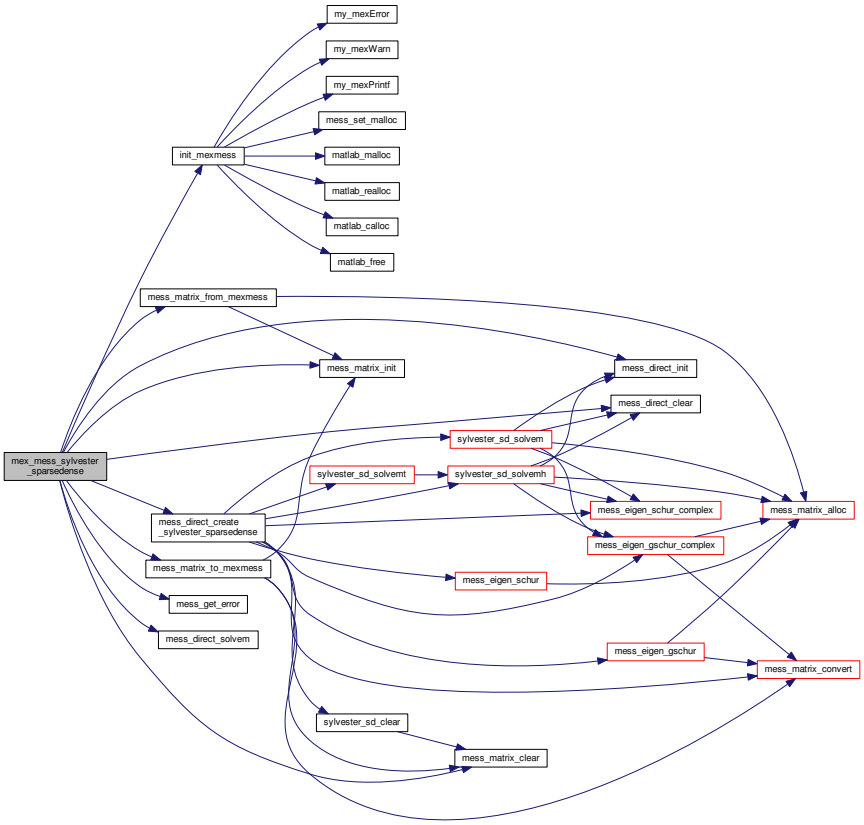
8.73.3.68 void mex_mess_sylvester_dense (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



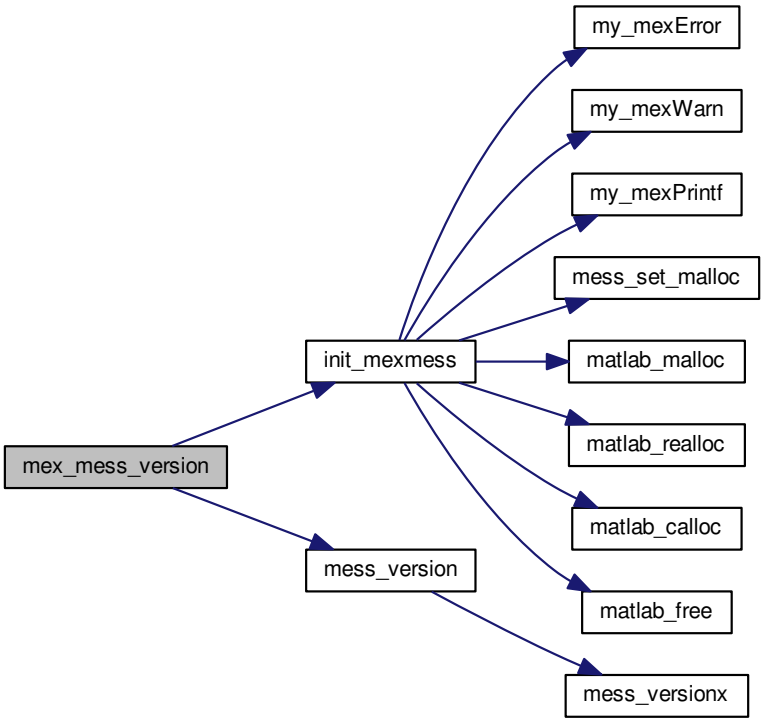
8.73.3.69 void mex_mess_sylvester_sparsedense (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



8.73.3.70 void mex_mess_version (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



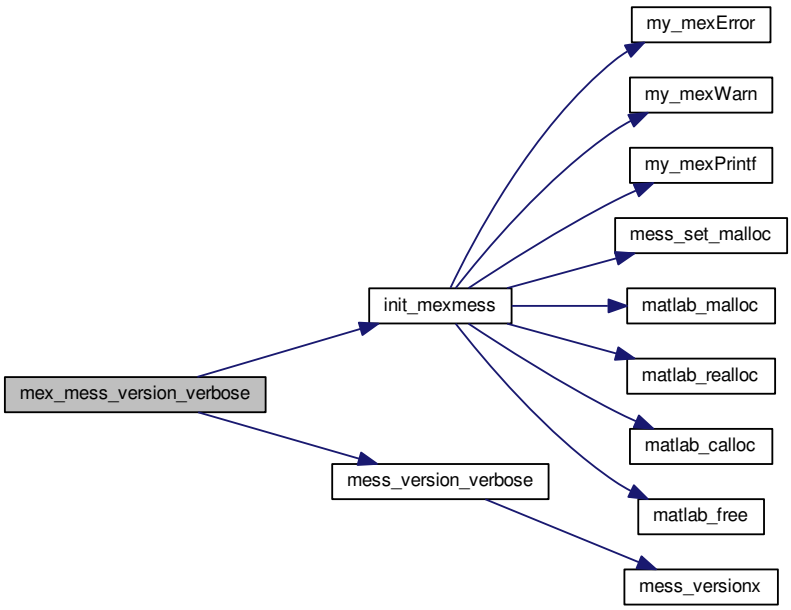
8.73.3.71 void mex_mess_version_major (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

8.73.3.72 void mex_mess_version_minor (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

8.73.3.73 void mex_mess_version_patch (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

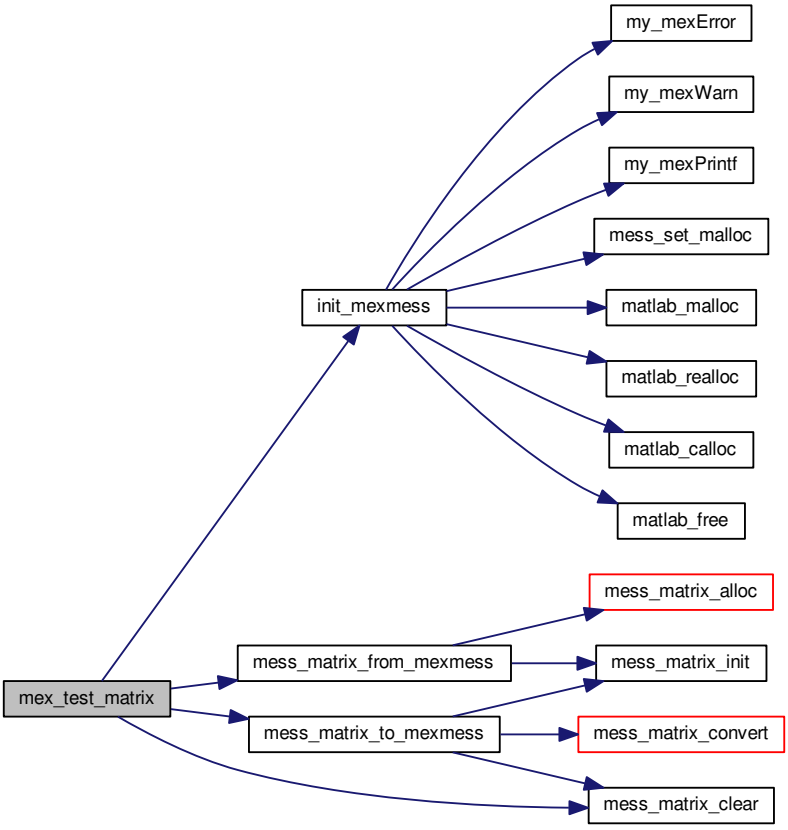
8.73.3.74 void mex_mess_version_verbose (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



8.73.3.75 void mex_test_matrix (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



8.73.3.76 void mex_test_mess_direct_cholpackage_t (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

8.73.3.77 void mex_test_mess_direct_lupackage_t (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

8.73.3.78 void mex_test_mess_equation_t (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

8.73.3.79 void mex_test_mess_memusage_t (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

8.73.3.80 void mex_test_mess_multidirect_t (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

8.73.3.81 void mex_test_mess_norm_t (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

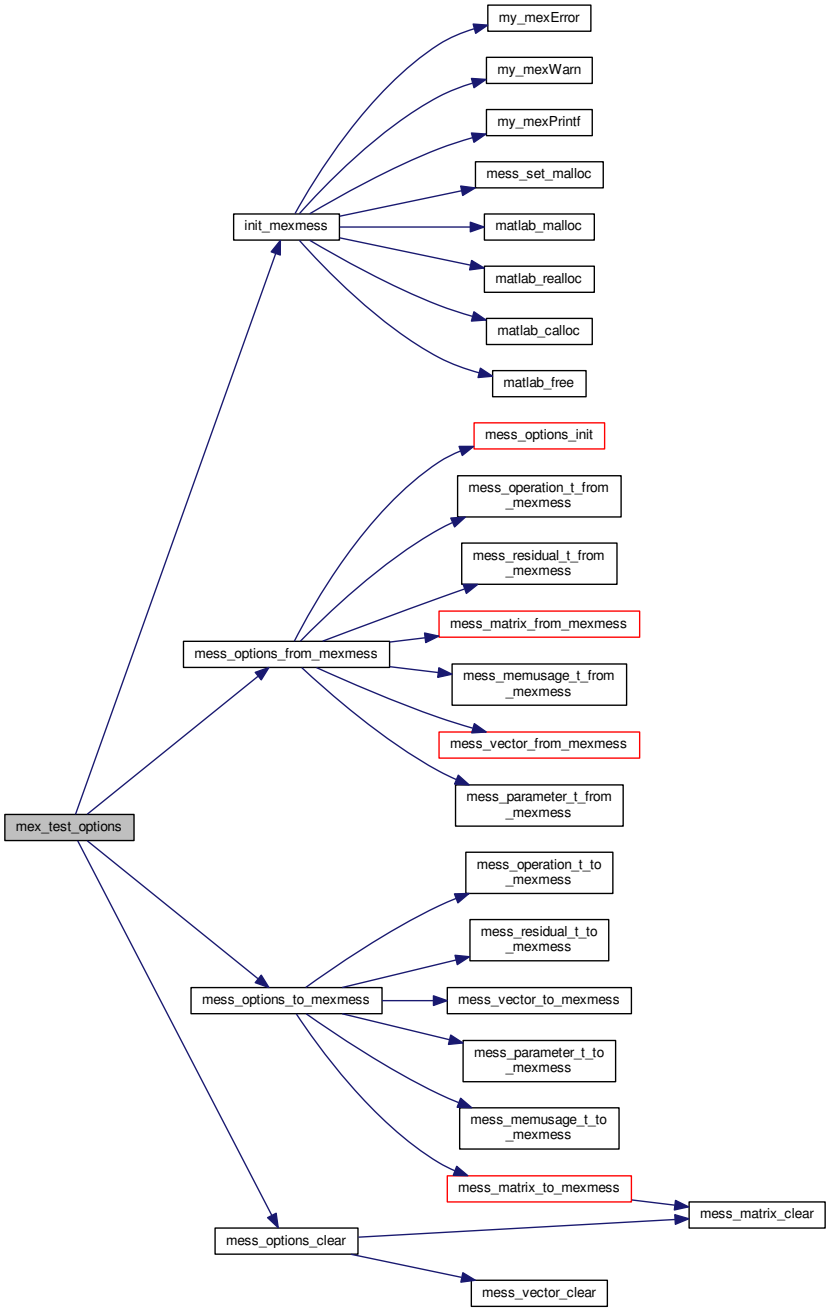
8.73.3.82 void mex_test_mess_operation_t (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

8.73.3.83 void mex_test_mess_parameter_t (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

8.73.3.84 void mex_test_mess_residual_t (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

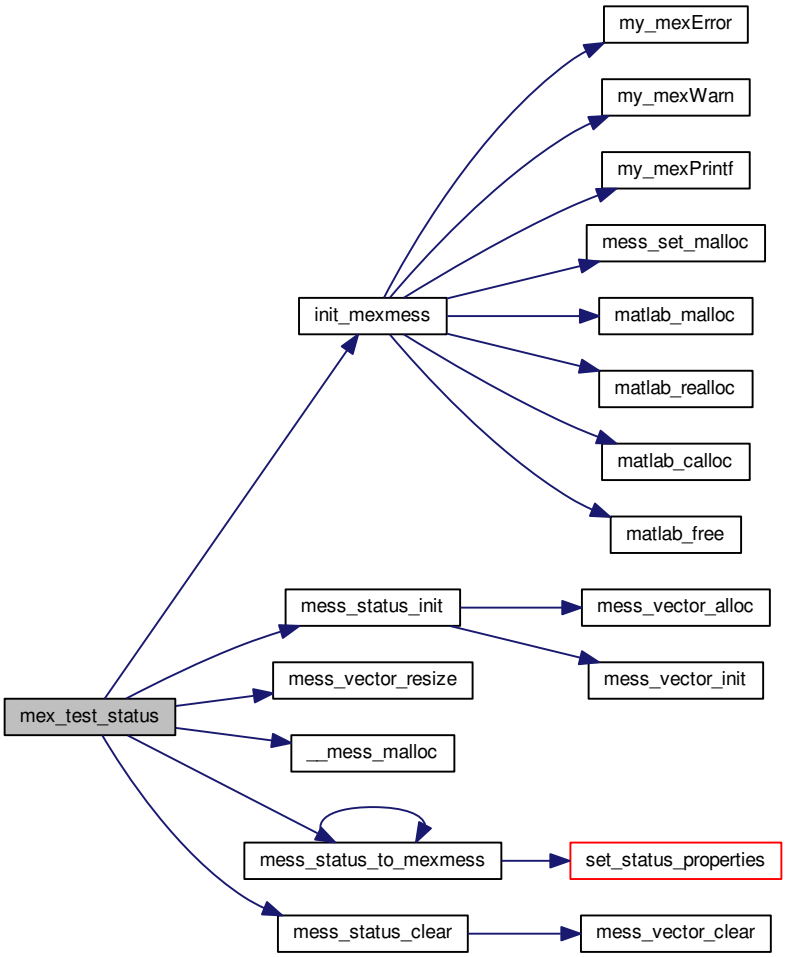
8.73.3.85 void mex_test_options (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



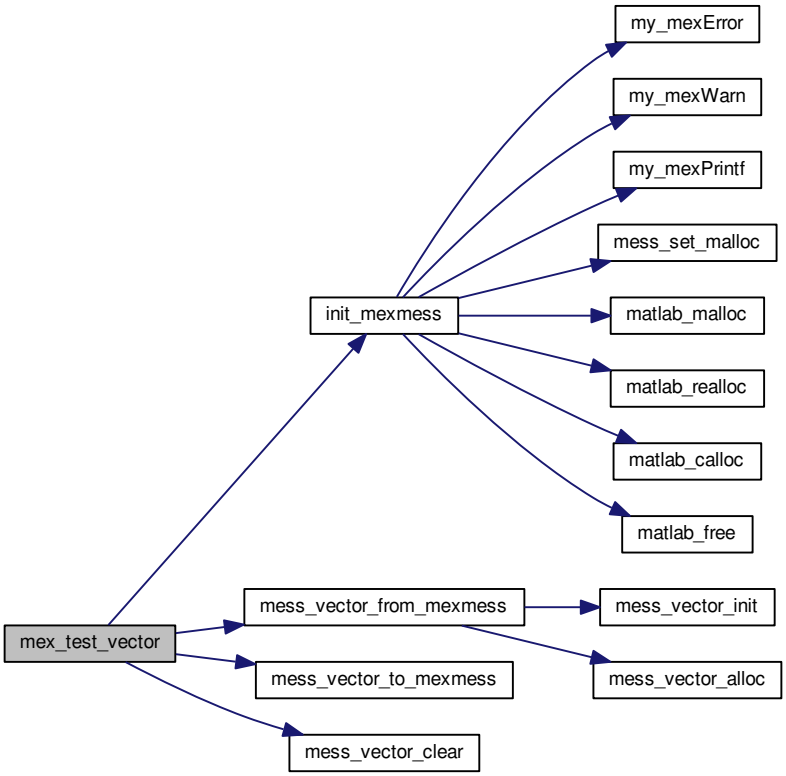
8.73.3.86 void mex_test_status (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



8.73.3.87 void mex_test_vector (int *nlhs*, mxArray * *plhs[ ]*, int *nrhs*, const mxArray * *prhs[ ]*)

Here is the call graph for this function:



8.73.3.88 mxArray* mxCreateComplexScalar (mess_double_cpx_t *s*)

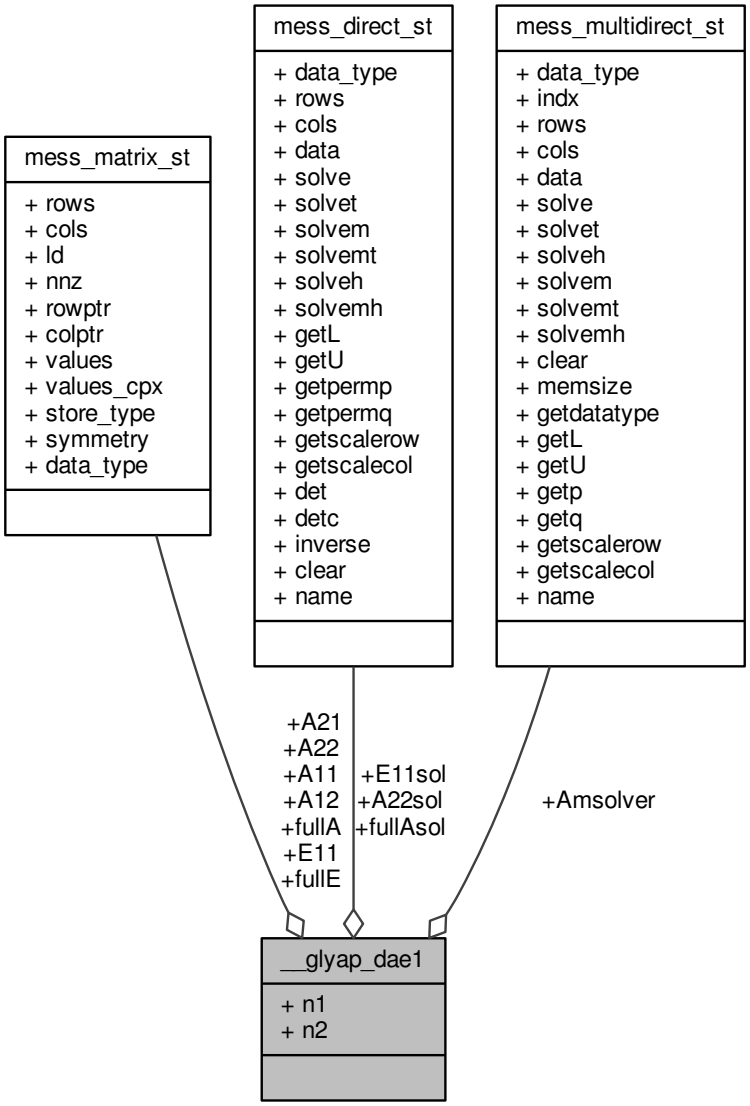
8.73.3.89 int mxIsSubClass (const mxArray * *instance*, char * *class*)

9 Data Structure Documentation

9.1 __glyap_dae1 Struct Reference

Contains matrices, solvers and additional data for a DAE 1 function handles.

Collaboration diagram for __glyap_dae1:



Data Fields

- mess_matrix E11
- mess_direct E11sol
- mess_matrix fullE

- mess_matrix A11
- mess_matrix A12
- mess_matrix A21
- mess_matrix A22
- mess_direct A22sol
- mess_matrix fullA
- mess_direct fullAsol
- mess_int_t n1
- mess_int_t n2
- mess_multidirect Amsolver

9.1.1 Detailed Description

The __glyap_dae1 structure contains all matrices, solvers and additional data for DAE 1 function handles.

Attention

Internal use only.

9.1.2 Field Documentation

9.1.2.1 mess_matrix A11

System matrix A_{11}.

9.1.2.2 mess_matrix A12

System matrix A_{12}.

9.1.2.3 mess_matrix A21

System matrix A_{21}.

9.1.2.4 mess_matrix A22

System matrix A_{22}.

9.1.2.5 mess_direct A22sol

Direct solver for A_{22}.

9.1.2.6 mess_multidirect Amsolver

Multidirect solver for the shifted system $\begin{bmatrix} A_{11} + pE_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$.

9.1.2.7 mess_matrix E11

System matrix E_{11}.

9.1.2.8 mess_direct E11sol

Direct solver for E_{11}.

9.1.2.9 mess_matrix fullA

System block matrix $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$.

9.1.2.10 mess_direct fullAsol

Direct solver fo system block matrix $\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$.

9.1.2.11 mess_matrix fullE

System matrix $\begin{bmatrix} E_{11} & 0 \\ 0 & 0 \end{bmatrix}$.

9.1.2.12 mess_int_t n1

Number of rows of A_{11}.

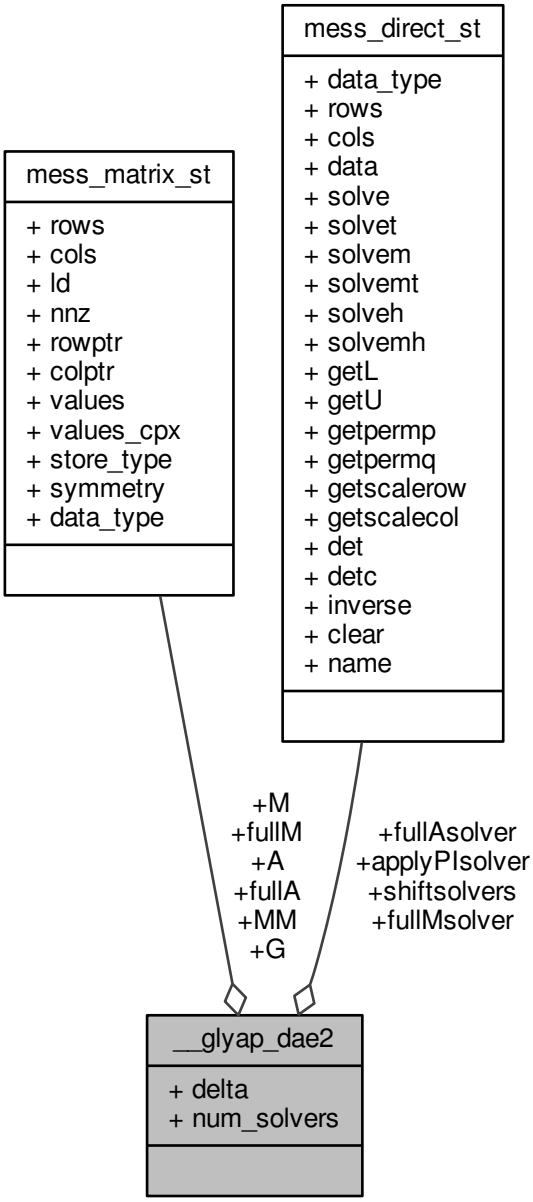
9.1.2.13 mess_int_t n2

Number of rows of A_{21}.

9.2 __glyap_dae2 Struct Reference

Contains matrices, solvers and additional data for a DAE 2 function handles.

Collaboration diagram for __glyap_dae2:



Data Fields

- mess_matrix M
- mess_matrix A
- mess_matrix G
- double delta
- mess_matrix fullA
- mess_direct fullAsolver
- mess_matrix fullM

- mess_direct fullMsolver
- mess_direct applyPIsolver
- mess_matrix MM
- mess_direct * shiftsolvers
- mess_int_t num_solvers

9.2.1 Detailed Description

The __glyap_dae2 structure contains all matrices, solvers and additional data for DAE 2 function handles.

Attention

Internal use only.

9.2.2 Field Documentation

9.2.2.1 mess_matrix A

System matrix A.

9.2.2.2 mess_direct applyPIsolver

Direct solver for $\begin{bmatrix} M & \delta G \\ \delta G^T & 0 \end{bmatrix}$.

9.2.2.3 double delta

Shift parameter to avoid infiite eigenvalues of the matrix pencil.

9.2.2.4 mess_matrix fullA

System block matrix $\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}$.

9.2.2.5 mess_direct fullAsolver

Direct solver for $\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}$.

9.2.2.6 mess_matrix fullM

System block matrix $\begin{bmatrix} M & \delta G \\ \delta G^T & 0 \end{bmatrix}$ for shift paramter computations.

9.2.2.7 mess_direct fullMsolver

Direct solver for $\begin{bmatrix} M & \delta G \\ \delta G^T & 0 \end{bmatrix}$ for shift paramter computations.

9.2.2.8 mess_matrix G

System matrix G.

9.2.2.9 mess_matrix M

System matrix M.

9.2.2.10 mess_matrix MM

System matrix $\begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}$.

9.2.2.11 mess_int_t num_solvers

Number of direct solvers in __glyap_dae2::shiftsolvers.

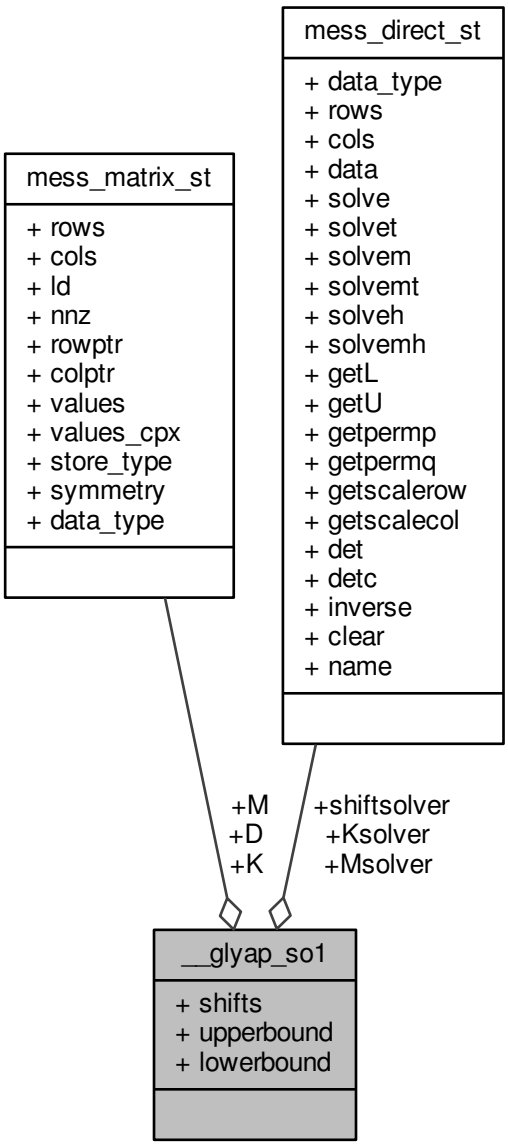
9.2.2.12 mess_direct* shiftsolvers

Array of direct solvers for shifted system $\begin{bmatrix} A + pM & G \\ G^T & 0 \end{bmatrix}$.

9.3 __glyap_so1 Struct Reference

Contains matrices, solvers and additional data for Second Order System function handles.

Collaboration diagram for __glyap_so1:



Data Fields

- mess_matrix M
- mess_matrix D
- mess_matrix K
- mess_direct Ksolver
- mess_direct Msolver
- mess_direct * shiftsolver
- mess_int_t shifts
- double upperbound
- double lowerbound

9.3.1 Detailed Description

The __glyap_so1 structure contains all matrices, solvers and additional data for Second Order System function handles.

Attention

Internal use only.

9.3.2 Field Documentation

9.3.2.1 mess_matrix D

System matrix D.

9.3.2.2 mess_matrix K

System matrix K.

9.3.2.3 mess_direct Ksolver

Direct solver fo K.

9.3.2.4 double lowerbound

lowerbound for absolute value of shift parameter.

9.3.2.5 mess_matrix M

System matrix M.

9.3.2.6 mess_direct Msolver

Direct solver fo M.

9.3.2.7 mess_int_t shifts

Number of direct solvers in __glyap_so1::shiftsolver.

9.3.2.8 mess_direct* shiftsolver

Array of direct solvers for shifted system $K - p * D + p^2 * M$.

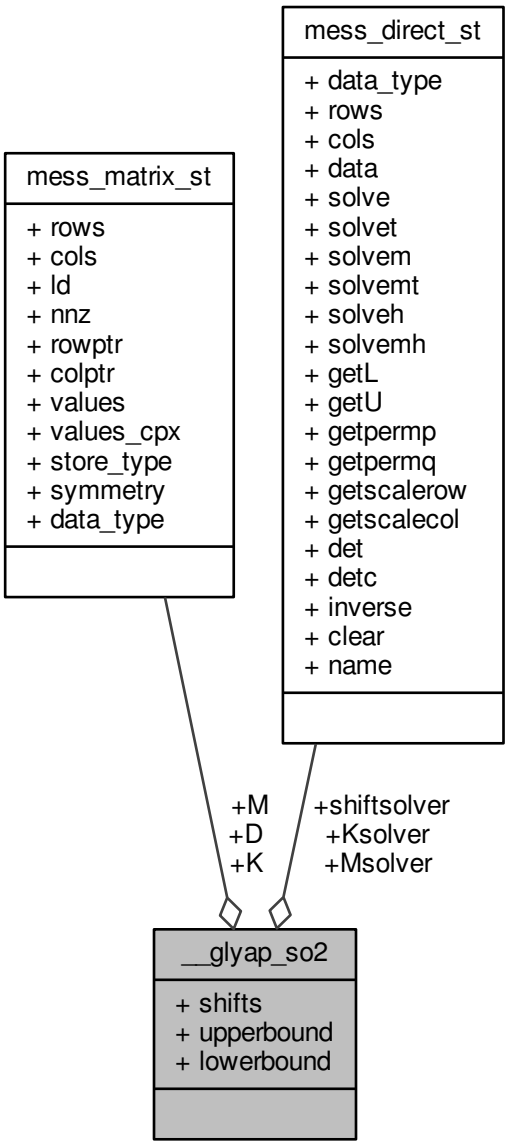
9.3.2.9 double upperbound

upperbound for absolute value of shift parameter.

9.4 __glyap_so2 Struct Reference

Contains matrices, solvers and additional data for Second Order System function handles.

Collaboration diagram for __glyap_so2:



Data Fields

- mess_matrix M
- mess_matrix D
- mess_matrix K
- mess_direct Ksolver

- mess_direct Msolver
- mess_direct * shiftsolver
- mess_int_t shifts
- double upperbound
- double lowerbound

9.4.1 Detailed Description

The __glyap_so2 structure contains all matrices, solvers and additional data for Second Order System function handles.

Attention

Internal use only.

9.4.2 Field Documentation

9.4.2.1 mess_matrix D

System matrix D.

9.4.2.2 mess_matrix K

System matrix K.

9.4.2.3 mess_direct Ksolver

Direct solver fo K.

9.4.2.4 double lowerbound

lowerbound for absolute value of shift parameter.

9.4.2.5 mess_matrix M

System matrix M.

9.4.2.6 mess_direct Msolver

Direct solver fo M.

9.4.2.7 mess_int_t shifts

Number of direct solvers in __glyap_so2::shiftsolver.

9.4.2.8 mess_direct* shiftsolver

Array of direct solvers for shifted system $K - p * D + p^2 * M$.

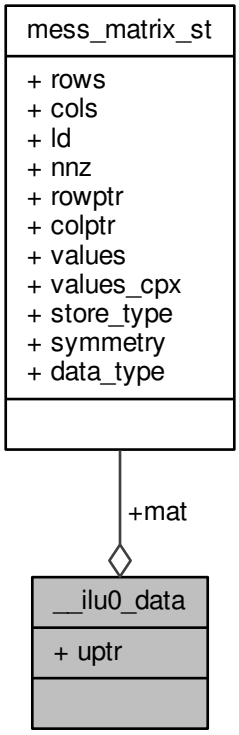
9.4.2.9 double upperbound

upperbound for absolute value of shift parameter.

9.5 __ilu0_data Struct Reference

Internal structure for $ILU(0)$ preconditioner. Internal structure for $ILU(0)$ preconditioner.

Collaboration diagram for __ilu0_data:



Data Fields

- mess_matrix mat
- mess_int_t $*$ uptr

9.5.1 Detailed Description

Attention

Interal use only.

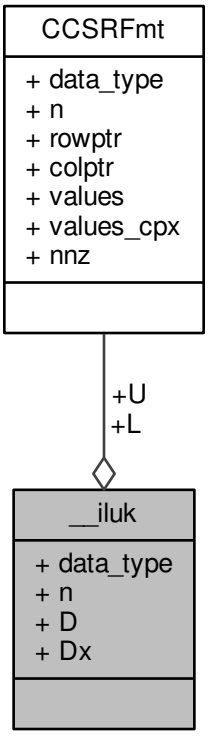
9.5.2 Field Documentation

9.5.2.1 mess_matrix mat

9.5.2.2 mess_int_t* uptr

9.6 __iluk Struct Reference

Collaboration diagram for __iluk:



Data Fields

- mess_datatype_t data_type
- mess_int_t n
- double * D
- mess_double_cpx_t * Dx
- ccsrptr L
- ccsrptr U

9.6.1 Field Documentation

9.6.1.1 double* D

9.6.1.2 mess_datatype_t data_type

9.6.1.3 mess_double_cpx_t* Dx

9.6.1.4 ccsrptr L

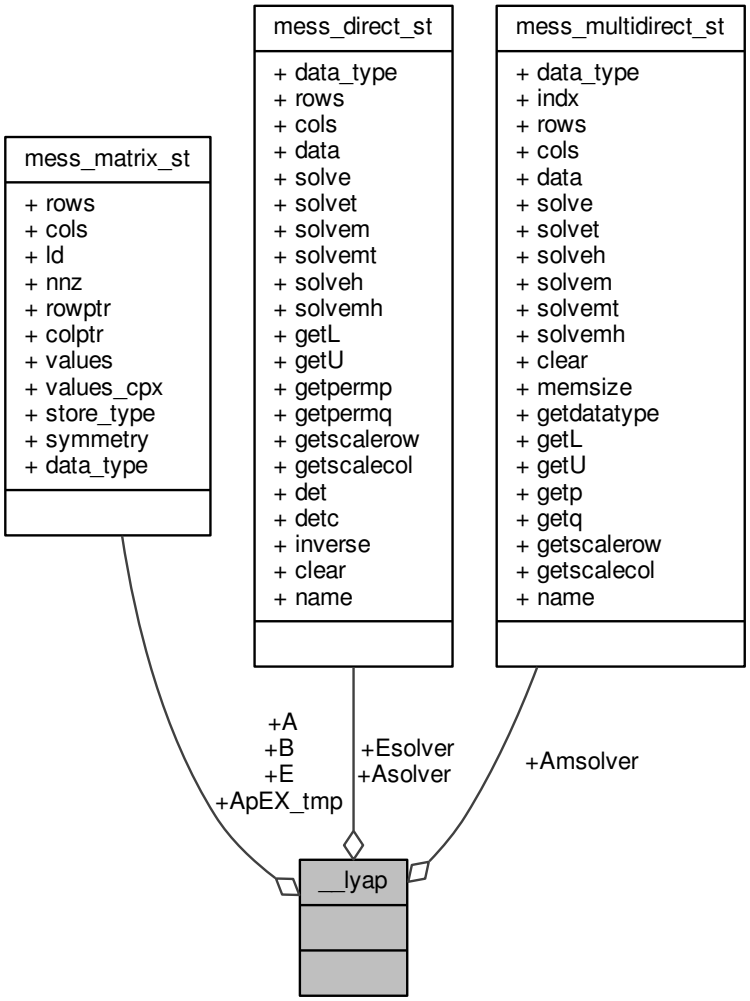
9.6.1.5 mess_int_t n

9.6.1.6 ccsrptr U

9.7 __lyap Struct Reference

Contains matrices, solvers and additional data for a generalized Lyapunov Equation.

Collaboration diagram for __lyap:



Data Fields

- mess_matrix A
- mess_matrix E
- mess_direct Asolver
- mess_direct Esolver
- mess_multidirect Amsolver
- mess_matrix ApEX_tmp
- mess_matrix B

9.7.1 Detailed Description

Contains matrices, solvers and additional data for a Lyapunov Equation.The __lyap structure contains all matrices, solvers and additional data for a generalized Lyapunov Equation

$$AXE^T + EAF^T = -BB^T$$

or

$$A^TXE + E^TXA = -B^TB.$$

Attention

Internal use only.

The __lyap structure contains all matrices, solvers and additional data for a Lyapunov Equation

$$AX + AF^T = -BB^T$$

or

$$A^TX + XA = -B^TB.$$

Attention

Internal use only.

9.7.2 Field Documentation

9.7.2.1 mess_matrix A

System matrix A of Lyapunov Equation.

9.7.2.2 mess_multidirect Amsolver

Multisolver for shifted matrix $A + pE$.

Multisolver for shifted matrix $A + pI$.

9.7.2.3 mess_matrix ApEX_tmp

Temporary shifted matrix $A + pI$.

Temporary shifted $A + pI$.

9.7.2.4 mess_direct Asolver

Direct solver for system matrix A.

Direct solver for system matrix A.

9.7.2.5 mess_matrix B

System matrix B of Lyapunov Equation.

9.7.2.6 mess_matrix E

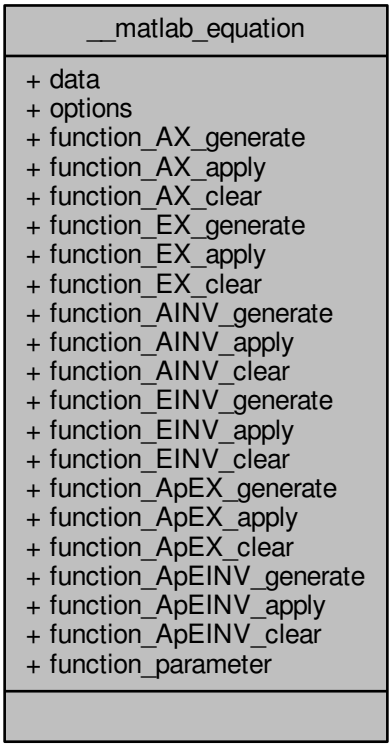
System matrix E of Lyapunov Equation.

9.7.2.7 mess_direct Esolver

Direct solver for system matrix E.

9.8 __matlab_equation Struct Reference

Collaboration diagram for __matlab_equation:



Data Fields

- mxArray * data
- mxArray * options
- mxArray * function_AX_generate
- mxArray * function_AX_apply
- mxArray * function_AX_clear
- mxArray * function_EX_generate
- mxArray * function_EX_apply
- mxArray * function_EX_clear
- mxArray * function_AINV_generate
- mxArray * function_AINV_apply
- mxArray * function_AINV_clear
- mxArray * function_EINV_generate
- mxArray * function_EINV_apply
- mxArray * function_EINV_clear
- mxArray * function_ApEX_generate
- mxArray * function_ApEX_apply
- mxArray * function_ApEX_clear
- mxArray * function_ApEINV_generate
- mxArray * function_ApEINV_apply
- mxArray * function_ApEINV_clear
- mxArray * function_parameter

9.8.1 Field Documentation

9.8.1.1 mxArray* data

9.8.1.2 mxArray* function_AINV_apply

9.8.1.3 mxArray* function_AINV_clear

9.8.1.4 mxArray* function_AINV_generate

9.8.1.5 mxArray* function_ApEINV_apply

9.8.1.6 mxArray* function_ApEINV_clear

9.8.1.7 mxArray* function_ApEINV_generate

9.8.1.8 mxArray* function_ApEX_apply

9.8.1.9 mxArray* function_ApEX_clear

9.8.1.10 mxArray* function_ApEX_generate

9.8.1.11 mxArray* function_AX_apply

9.8.1.12 mxArray* function_AX_clear

9.8.1.13 mxArray* function_AX_generate

9.8.1.14 mxArray* function_EINV_apply

9.8.1.15 mxArray* function_EINV_clear

9.8.1.16 mxArray* function_EINV_generate

9.8.1.17 mxArray* function_EX_apply

9.8.1.18 mxArray* function_EX_clear

9.8.1.19 mxArray* function_EX_generate

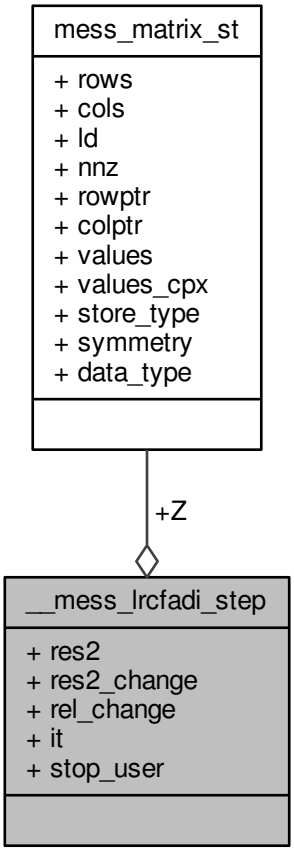
9.8.1.20 mxArray* function_parameter

9.8.1.21 mxArray* options

9.9 __mess_lrcfadi_step Struct Reference

Structure to pass information to the step function inside the LRCF iterations.

Collaboration diagram for __mess_lrcfadi_step:



Data Fields

- double res2
- double res2_change
- double rel_change
- mess_matrix Z
- mess_int_t it
- unsigned short stop_user

9.9.1 Detailed Description

The mess_lrcfadi_step structure is used to pass information to a step function inside the ADI iteration.
The step function, if set, is executed at the end of every ADI or Newton iteration.
This structure is passed to the step function to provide information about the current state of the iteration.
The stop_user component can be used to provide a user supplied cancellation criterion.

Attention

Interal use only.

9.9.2 Field Documentation

9.9.2.1 mess_int_t it

Current iteration number.

9.9.2.2 double rel_change

Current Frobenius norm change in the factor.

9.9.2.3 double res2

Current 2-norm residual.

9.9.2.4 double res2_change

Current 2-norm residual change.

9.9.2.5 unsigned short stop_user

Output parameter, flag to cancel the iteration.

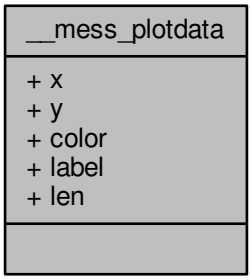
9.9.2.6 mess_matrix Z

Current solution factor Z.

9.10 __mess_plotdata Struct Reference

Structure for one data element in a plot.

Collaboration diagram for __mess_plotdata:



Data Fields

- double * x
- double * y
- char color [40]
- char label [40]
- mess_int_t len

9.10.1 Detailed Description

The mess_plotdata structure holds all information of a data element in a plot.

9.10.2 Field Documentation

9.10.2.1 char color[40]

Name of the plotting color

9.10.2.2 char label[40]

Label of the plot

9.10.2.3 mess_int_t len

Number of values

9.10.2.4 double∗ x

x values of the data element

9.10.2.5 double∗ y

y values of the data element

9.11 __mess_plotExportData Struct Reference

Structure for one data element in a plot.

Collaboration diagram for __mess_plotExportData:

Data Fields

- double ∗ x
- double ∗ y
- char color [40]
- char label [40]
- char type [40]
- mess_int_t len

9.11.1 Detailed Description

The mess_plotExportData structure holds all information of a data element in a plot.

Attention

Internal use only.

9.11.2 Field Documentation

9.11.2.1 char color[40]

Colorname to be used for the plot

9.11.2.2 char label[40]

Label of the plot

9.11.2.3 mess_int_t len

Number of values

9.11.2.4 char type[40]

Type of the plot

9.11.2.5 double* x

x values of the data element

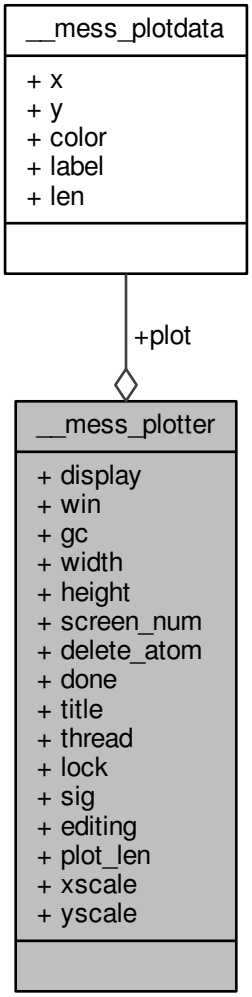
9.11.2.6 double* y

y values of the data element

9.12 __mess_plotter Struct Reference

Structure to hold a complete plotter.

Collaboration diagram for __mess_plotter:



Data Fields

- Display * display
- Window win
- GC gc
- int width
- int height
- int screen_num
- Atom delete_atom
- int done
- char * title
- pthread_t thread
- pthread_mutex_t lock
- pthread_cond_t sig
- int editing

- mess_plotdata * plot
- int plot_len
- int xscale
- int yscale

9.12.1 Detailed Description

The __mess_plotter is the basic structure to represent a complete plot.

Attention

Interal use only.

9.12.2 Field Documentation

9.12.2.1 Atom delete_atom

X11 destroy window atom

9.12.2.2 Display* display

X11 display

9.12.2.3 int done

Done flag

9.12.2.4 int editing

State variable for the lock

9.12.2.5 GC gc

X11 graphic context

9.12.2.6 int height

Height of the plot

9.12.2.7 pthread_mutex_t lock

Lock for the thread

9.12.2.8 mess_plotdata* plot

Data elements

9.12.2.9 int plot_len

Number of data elements

9.12.2.10 int screen_num

X11 screen number

9.12.2.11 pthread_cond_t sig

Signal for the lock

9.12.2.12 pthread_t thread

Worker thread for the plot

9.12.2.13 char* title

Title of the window

9.12.2.14 int width

Width of the plot

9.12.2.15 Window win

X11 window

9.12.2.16 int xscale

Scaling of the x axis

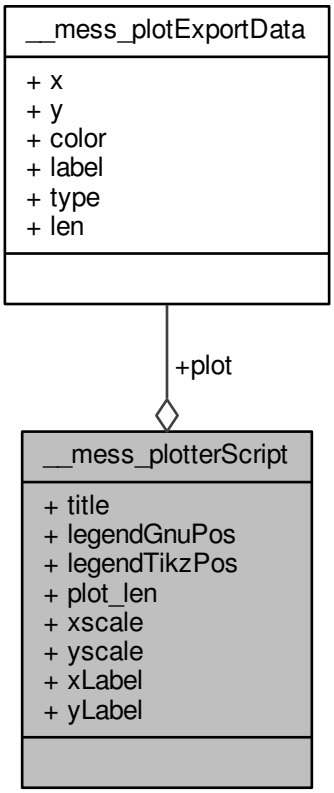
9.12.2.17 int yscale

Scaling of the y axis

9.13 __mess_plotterScript Struct Reference

Structure to hold a complete plotter.

Collaboration diagram for __mess_plotterScript:



Data Fields

- char * title
- char * legendGnuPos
- char * legendTikzPos
- mess_plotExportData * plot
- int plot_len
- mess_plot_axis_scale_t xscale
- mess_plot_axis_scale_t yscale
- char * xLabel
- char * yLabel

9.13.1 Detailed Description

The __mess_plotterScript is the basic structure to represent a complete plot.

Attention

Internal use only.

9.13.2 Field Documentation

9.13.2.1 char* legendGnuPos

Orientation of key-box for Gnu

9.13.2.2 char* legendTikzPos

Orientation of key-box for Tikz

9.13.2.3 mess_plotExportData* plot

Data elements

9.13.2.4 int plot_len

Number of data elements

9.13.2.5 char* title

Title of the window

9.13.2.6 char* xLabel

Label of the x axis

9.13.2.7 mess_plot_axis_scale_t xscale

Scaling of the x axis

9.13.2.8 char* yLabel

Label of the y axis

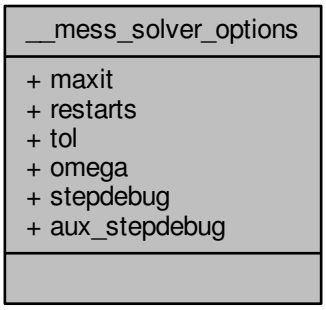
9.13.2.9 mess_plot_axis_scale_t yscale

Scaling of the y axis

9.14 __mess_solver_options Struct Reference

Representation of options for iterative solvers.

Collaboration diagram for __mess_solver_options:



Data Fields

- mess_int_t maxit
- mess_int_t restarts
- double tol
- double omega
- void(∗ stepdebug)(double res, double relres, mess_int_t it, void ∗opt)
- void ∗ aux_stepdebug

9.14.1 Detailed Description

The __mess_solver_options structure represents options for an iterative solver.

9.14.2 Field Documentation

9.14.2.1 void∗ aux_stepdebug

Auxilary data for stepdebug

9.14.2.2 mess_int_t maxit

Maximum number of iterations

9.14.2.3 double omega

Parameter for SOR and SSOR

9.14.2.4 mess_int_t restarts

Maximum number of restarts, if the value is -1 is runs until the algorithm converged.

9.14.2.5 void(∗ stepdebug) (double res, double relres, mess_int_t it, void ∗opt)

Step debug function, evaluated every step

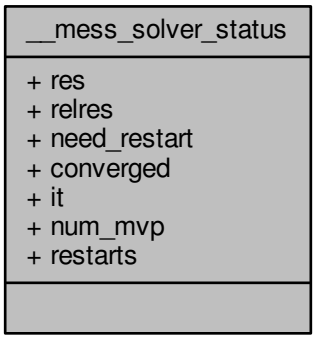
9.14.2.6 double tol

Relative tolerance

9.15 __mess_solver_status Struct Reference

Representation of status information of an iterative solvers.

Collaboration diagram for __mess_solver_status:



Data Fields

- double res
- double relres
- int need_restart
- int converged
- mess_int_t it
- mess_int_t num_mvp
- mess_int_t restarts

9.15.1 Detailed Description

The __mess_solver_status structure represents the status information about an iterative solver.

9.15.2 Field Documentation

9.15.2.1 int converged

True if the algorithm converged

9.15.2.2 mess_int_t it

Number of calculated iterations

9.15.2.3 int need_restart

True if a restart is needed

9.15.2.4 mess_int_t num_mvp

Number of computed matrix-vector products

9.15.2.5 double relres

Relative residual

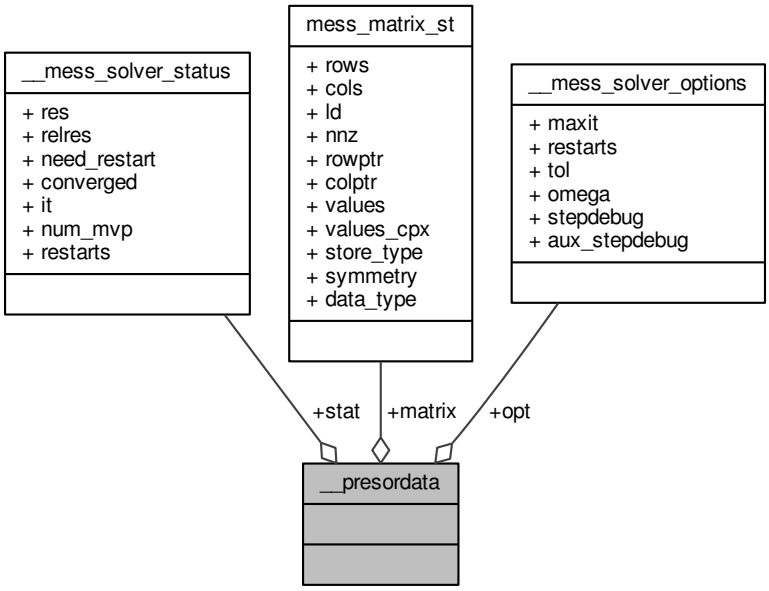
9.15.2.6 double res

Residual norm

9.15.2.7 mess_int_t restarts

9.16 __presordata Struct Reference

Collaboration diagram for __presordata:



Data Fields

- mess_solver_options opt
- mess_solver_status stat
- mess_matrix matrix

9.16.1 Field Documentation

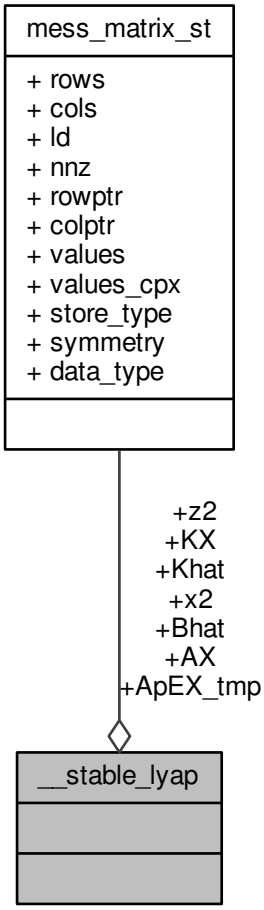
9.16.1.1 mess_matrix matrix

9.16.1.2 mess_solver_options opt

9.16.1.3 mess_solver_status stat

9.17 __stable_lyap Struct Reference

Collaboration diagram for __stable_lyap:



Data Fields

- mess_matrix ApEX_tmp
- mess_matrix AX
- mess_matrix KX
- mess_matrix x2
- mess_matrix z2
- mess_matrix Khat
- mess_matrix Bhat

9.17.1 Field Documentation

9.17.1.1 mess_matrix ApEX_tmp

9.17.1.2 mess_matrix AX

9.17.1.3 mess_matrix Bhat

9.17.1.4 mess_matrix Khat

9.17.1.5 mess_matrix KX

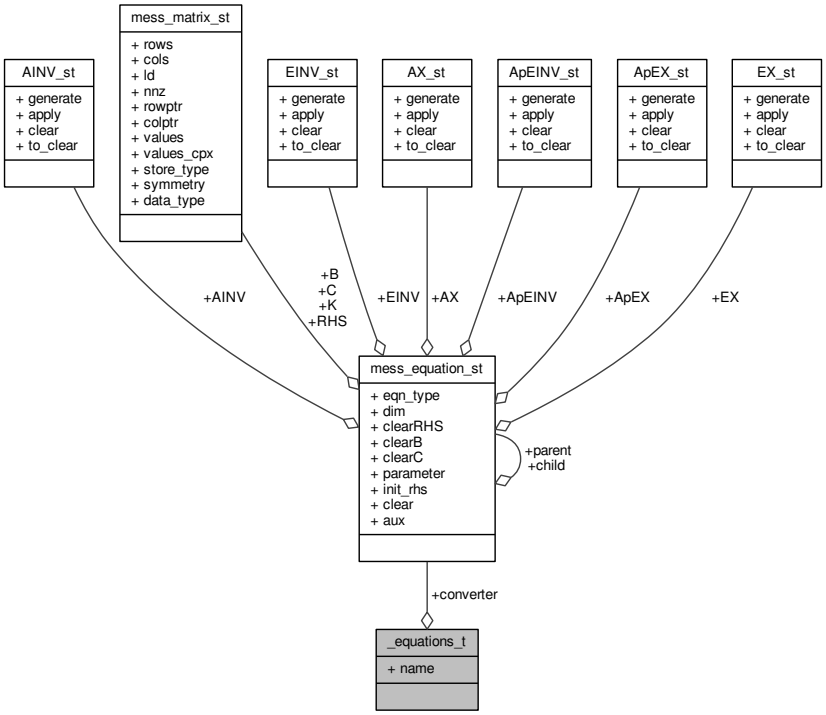
9.17.1.6 mess_matrix x2

9.17.1.7 mess_matrix z2

9.18 _equations_t Struct Reference

Structure for all **Py-M.E.S.S.** to **M.E.S.S.** converters. Structure for all **Py-M.E.S.S.** to **M.E.S.S.** converters.

Collaboration diagram for _equations_t:



Data Fields

- char * name
- eqn_converter converter

9.18.1 Detailed Description

Attention

Internal use only.

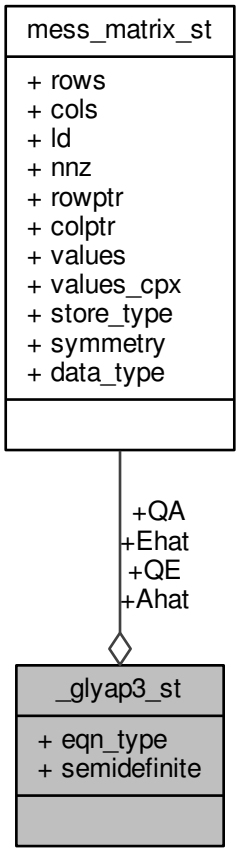
9.18.2 Field Documentation

9.18.2.1 eqn_converter converter

9.18.2.2 char * name

9.19 _glyap3_st Struct Reference

Collaboration diagram for _glyap3_st:



Data Fields

- mess_matrix Ahat
- mess_matrix QA
- mess_matrix Ehat
- mess_matrix QE
- _glyap3_eqn_t eqn_type
- int semidefinite

9.19.1 Detailed Description

Attention

Internal use only.

9.19.2 Field Documentation

9.19.2.1 mess_matrix Ahat

9.19.2.2 mess_matrix Ehat

Schur Transformation Matrix of Matrix A.

9.19.2.3 _glyap3_eqn_t eqn_type

Schur Transformation Matrix of Matrix E.

9.19.2.4 mess_matrix QA

Schur Form of Matrix A of Sylvester Equation.

9.19.2.5 mess_matrix QE

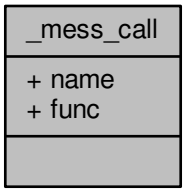
Schur Form of Matrix E of Sylvester Equation.

9.19.2.6 int semidefinite

Represents the Type of Equation.

9.20 _mess_call Struct Reference

Collaboration diagram for _mess_call:



Data Fields

- char * name
- mess_call_func func

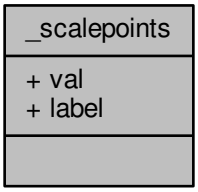
9.20.1 Field Documentation

9.20.1.1 mess_call_func func

9.20.1.2 char* name

9.21 _scalepoints Struct Reference

Collaboration diagram for _scalepoints:



Data Fields

- double val
- char label [11]

9.21.1 Detailed Description

Displaypoints for the X and Y axis

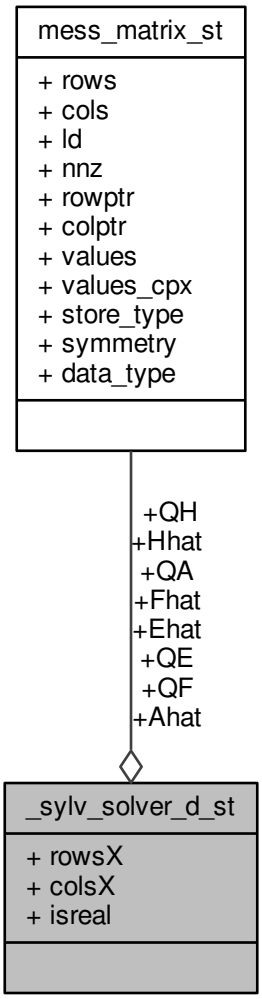
9.21.2 Field Documentation

9.21.2.1 char label[11]

9.21.2.2 double val

9.22 _sylv_solver_d_st Struct Reference

Collaboration diagram for _sylv_solver_d_st:



Data Fields

- mess_matrix Ahat
- mess_matrix QA
- mess_matrix Ehat
- mess_matrix QE
- mess_matrix Fhat
- mess_matrix QF
- mess_matrix Hhat
- mess_matrix QH
- mess_int_t rowsX
- mess_int_t colsX
- int isreal

9.22.1 Detailed Description

Attention

Internal use only.

9.22.2 Field Documentation

9.22.2.1 mess_matrix Ahat

9.22.2.2 mess_int_t colsX

Rows of Solution X of Sylvester Equation.

9.22.2.3 mess_matrix Ehat

Schur Transformation Matrix of Matrix A.

9.22.2.4 mess_matrix Fhat

Schur Transformation Matrix of Matrix E.

9.22.2.5 mess_matrix Hhat

Schur Transformation Matrix of Matrix F.

9.22.2.6 int isreal

Columns of Solution X of Sylvester Equation.

9.22.2.7 mess_matrix QA

Schur Form of Matrix A of Sylvester Equation.

9.22.2.8 mess_matrix QE

Schur Form of Matrix E of Sylvester Equation.

9.22.2.9 mess_matrix QF

Schur Form of Matrix F of Sylvester Equation.

9.22.2.10 mess_matrix QH

Schur Form of Matrix H of Sylvester Equation.

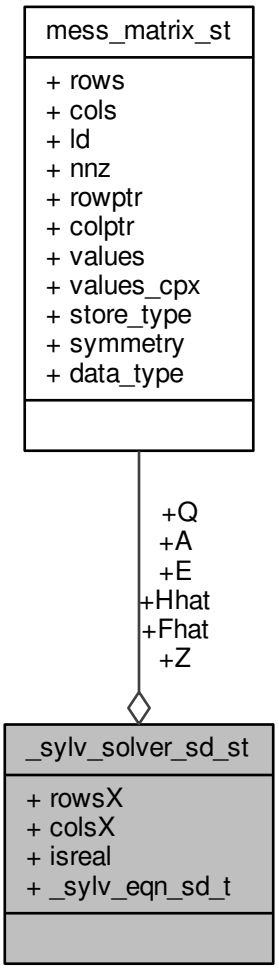
9.22.2.11 mess_int_t rowsX

Schur Transformation Matrix of Matrix H.

9.23 _sylv_solver_sd_st Struct Reference

Internal structure for sparse-dense Sylvester equation solver.

Collaboration diagram for _sylv_solver_sd_st:



Data Fields

- mess_matrix A
- mess_matrix Fhat
- mess_matrix Hhat
- mess_matrix Q
- mess_matrix Z
- mess_matrix E
- mess_int_t rowsX
- mess_int_t colsX
- int isreal
- _sylv_equation_sd_t _sylv_eqn_sd_t

9.23.1 Detailed Description

Attention

Internal use only.

9.23.2 Field Documentation

9.23.2.1 _sylv_equation_sd_t _sylv_eqn_sd_t

`1` if all Matrices of Sylvester Equation are real, `0` otherwise.

9.23.2.2 mess_matrix A

9.23.2.3 mess_int_t colsX

Rows of Solution X of Sylvester Equation.

9.23.2.4 mess_matrix E

Schur Transformation Matrix of Matrix Pair (F, H) of Sylvester Equation.

9.23.2.5 mess_matrix Fhat

Matrix A of Sylvester Equation.

9.23.2.6 mess_matrix Hhat

Schur Form of Matrix F of Sylvester Equation.

9.23.2.7 int isreal

Columns of Solution X of Sylvester Equation.

9.23.2.8 mess_matrix Q

Schur Form of Matrix H of Sylvester Equation.

9.23.2.9 mess_int_t rowsX

Matrix E of Sylvester Equation.

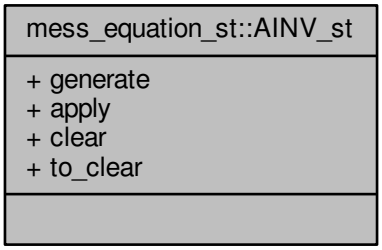
9.23.2.10 mess_matrix Z

Schur Transformation Matrix of Matrix Pair (F, H) of Sylvester Equation.

9.24 mess_equation_st::AINV_st Struct Reference

Structure defining $op(A)Y = X$.

Collaboration diagram for mess_equation_st::AINV_st:



Data Fields

- int(∗ generate)(struct mess_equation_st ∗eqn)
- int(∗ apply)(struct mess_equation_st ∗eqn, mess_operation_t op, mess_matrix in, mess_matrix out)
- int(∗ clear)(struct mess_equation_st ∗eqn)
- int to_clear

9.24.1 Detailed Description

Structure defining $op(A)Y = X$ where X and Y are matrices.

9.24.2 Field Documentation

9.24.2.1 int(∗ apply) (struct mess_equation_st ∗eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

Apply the inverse operator to a matrix. Compute $op(A)Y = X$.

9.24.2.2 int(∗ clear) (struct mess_equation_st ∗eqn)

Clear $AINV$ objects separately.

9.24.2.3 int(∗ generate) (struct mess_equation_st ∗eqn)

Post process function if $op(A)Y = X$ needs a global initialization.

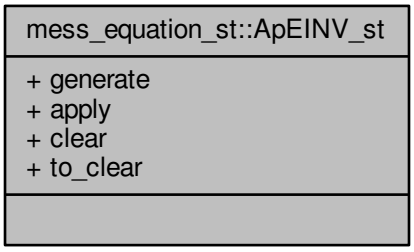
9.24.2.4 int to_clear

Indicator variable if the $AINV$ objects needs to be cleared separately.

9.25 mess_equation_st::ApEINV_st Struct Reference

Structure defining $op(A + pE)Y = X$.

Collaboration diagram for mess_equation_st::ApEINV_st:



Data Fields

- int(∗ generate)(struct mess_equation_st ∗eqn, mess_vector parameters)
- int(∗ apply)(struct mess_equation_st ∗eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)
- int(∗ clear)(struct mess_equation_st ∗eqn)
- int to_clear

9.25.1 Detailed Description

Structure defining $op(A + pE)Y = X$ where X and Y are vectors.

9.25.2 Field Documentation

9.25.2.1 int(∗ apply) (struct mess_equation_st ∗eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)

Apply the inverse of $op(A + pE)$ to a matrix. Compute $op(A + pE)Y = X$.

9.25.2.2 int(∗ clear) (struct mess_equation_st ∗eqn)

Clear $ApEINV$ objects separately.

9.25.2.3 int(∗ generate) (struct mess_equation_st ∗eqn, mess_vector parameters)

Post process function if $op(A + pE)Y = X$ needs a global initialization.

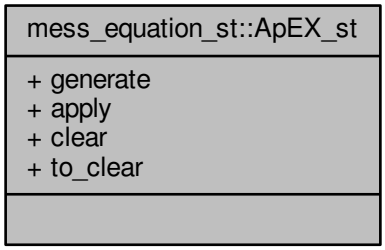
9.25.2.4 int to_clear

Indicator variable if the $ApEINV$ objects needs to be cleared separately.

9.26 mess_equation_st::ApEX_st Struct Reference

Structure defining $Y = op(A + \alpha pE)X$.

Collaboration diagram for mess_equation_st::ApEX_st:



Data Fields

- int(∗ generate)(struct mess_equation_st ∗eqn, mess_vector parameters)
- int(∗ apply)(struct mess_equation_st ∗eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)
- int(∗ clear)(struct mess_equation_st ∗eqn)
- int to_clear

9.26.1 Detailed Description

Structure defining $Y = op(A + \alpha pE)X$ where Y and X are matrices.

9.26.2 Field Documentation

9.26.2.1 int(∗ apply) (struct mess_equation_st ∗eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)

Apply $op(A + \alpha * p * E)$ to a matrix. Compute $y = op(A + \alpha pE)x$.

9.26.2.2 int(∗ clear) (struct mess_equation_st ∗eqn)

Clear $ApEX$ objects separately.

9.26.2.3 int(∗ generate) (struct mess_equation_st ∗eqn, mess_vector parameters)

Post process function if $Y = op(A + \alpha pE)X$ needs a global initialization.

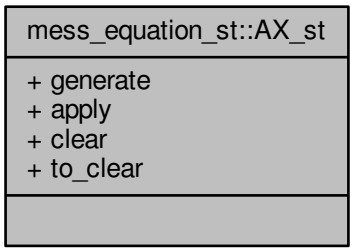
9.26.2.4 int to_clear

Indicator variable if the $ApEX$ objects needs to be cleared separately.

9.27 mess_equation_st::AX_st Struct Reference

Structure defining $Y = op(A)X$.

Collaboration diagram for mess_equation_st::AX_st:



Data Fields

- int(∗ generate)(struct mess_equation_st ∗eqn)
- int(∗ apply)(struct mess_equation_st ∗eqn, mess_operation_t op, mess_matrix in, mess_matrix out)
- int(∗ clear)(struct mess_equation_st ∗eqn)
- int to_clear

9.27.1 Detailed Description

Structure defining $Y = op(A)X$ where X and Y are matrices.

9.27.2 Field Documentation

9.27.2.1 int(∗ apply) (struct mess_equation_st ∗eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

Apply the operator to a matrix. Compute $Y = op(A)X$.

9.27.2.2 int(∗ clear) (struct mess_equation_st ∗eqn)

Clear AX objects separately.

9.27.2.3 int(∗ generate) (struct mess_equation_st ∗eqn)

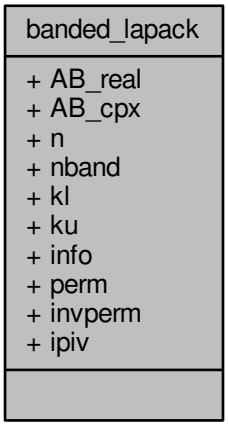
Post process function if $Y = AX$ needs a global initialization.

9.27.2.4 int to_clear

Indicator variable if the AX objects needs to be cleared separately.

9.28 banded_lapack Struct Reference

Collaboration diagram for banded_lapack:



Data Fields

- double ∗ AB_real
- mess_double_cpx_t ∗ AB_cpx
- mess_int_t n
- mess_int_t nband
- mess_int_t kl
- mess_int_t ku
- mess_int_t info
- mess_int_t ∗ perm
- mess_int_t ∗ invperm
- mess_int_t ∗ ipiv

9.28.1 Field Documentation

9.28.1.1 mess_double_cpx_t* AB_cpx

9.28.1.2 double* AB_real

9.28.1.3 mess_int_t info

9.28.1.4 mess_int_t* invperm

9.28.1.5 mess_int_t* ipiv

9.28.1.6 mess_int_t kl

9.28.1.7 mess_int_t ku

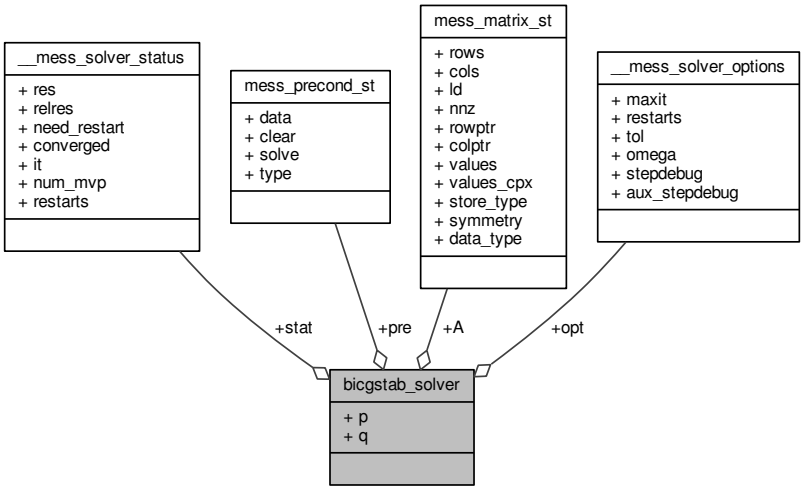
9.28.1.8 mess_int_t n

9.28.1.9 mess_int_t nband

9.28.1.10 mess_int_t* perm

9.29 bicgstab_solver Struct Reference

Collaboration diagram for bicgstab_solver:



Data Fields

- mess_matrix A
- mess_precond pre
- mess_solver_options opt
- mess_solver_status stat
- mess_int_t * p
- mess_int_t * q

9.29.1 Field Documentation

9.29.1.1 mess_matrix A

9.29.1.2 mess_solver_options opt

9.29.1.3 mess_int_t* p

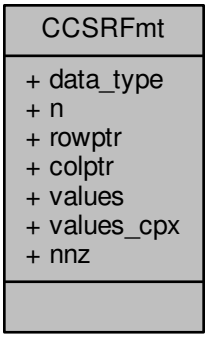
9.29.1.4 mess_precond pre

9.29.1.5 mess_int_t* q

9.29.1.6 mess_solver_status stat

9.30 CCSRFmt Struct Reference

Collaboration diagram for CCSRFmt:



Data Fields

- mess_datatype_t data_type
- mess_int_t n
- mess_int_t * rowptr
- mess_int_t * colptr
- double * values
- mess_double_cpx_t * values_cpx
- mess_int_t nnz

9.30.1 Field Documentation

9.30.1.1 mess_int_t* colptr

9.30.1.2 mess_datatype_t data_type

9.30.1.3 mess_int_t n

9.30.1.4 mess_int_t nnz

9.30.1.5 mess_int_t* rowptr

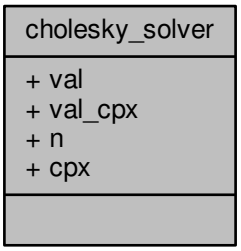
9.30.1.6 double* values

9.30.1.7 mess_double_cpx_t* values_cpx

9.31 cholesky_solver Struct Reference

Internal structure for the LAPACK based solver.

Collaboration diagram for cholesky_solver:



Data Fields

- double * val
- mess_double_cpx_t * val_cpx
- mess_int_t n
- unsigned short cpx

9.31.1 Field Documentation

9.31.1.1 unsigned short cpx

flag real/complex

9.31.1.2 mess_int_t n

dimension of the system

9.31.1.3 double* val

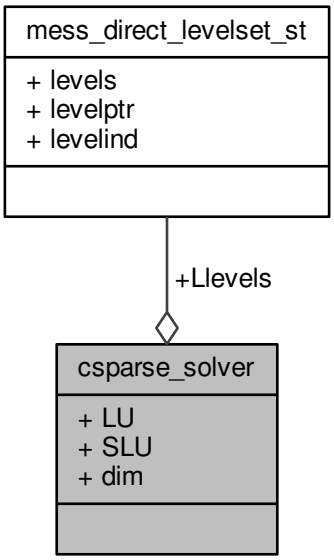
the real lu decomposed matrix

9.31.1.4 mess_double_cpx_t* val_cpx

the complex lu decomposed matrix

9.32 csparse_solver Struct Reference

Collaboration diagram for csparse_solver:



Data Fields

- cs_dln * LU
- cs_dls * SLU
- mess_int_t dim
- mess_direct_levelset Llevels

9.32.1 Detailed Description

Internal structure to save the factorization.

9.32.2 Field Documentation

9.32.2.1 mess_int_t dim

dimension

9.32.2.2 mess_direct_levelset Llevels

9.32.2.3 cs_dln * LU

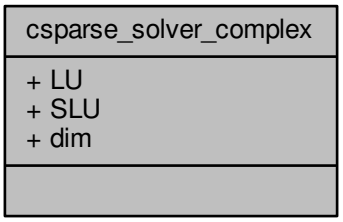
numerical part

9.32.2.4 cs_dls * SLU

symbolical part

9.33 csparse_solver_complex Struct Reference

Collaboration diagram for csparse_solver_complex:



Data Fields

- cs_cln * LU
- cs_cls * SLU
- mess_int_t dim

9.33.1 Detailed Description

Internal structure to save the factorization.

9.33.2 Field Documentation

9.33.2.1 mess_int_t dim

dimension

9.33.2.2 cs_cln ∗ LU

numerical part

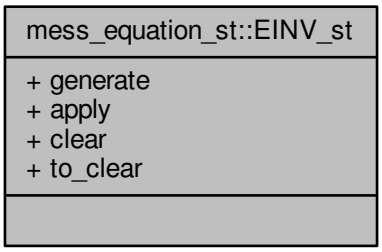
9.33.2.3 cs_cls ∗ SLU

symbolical part

9.34 mess_equation_st::EINV_st Struct Reference

Structure defining $op(E)Y = X$.

Collaboration diagram for mess_equation_st::EINV_st:



Data Fields

- int(∗ generate)(struct mess_equation_st ∗eqn)
- int(∗ apply)(struct mess_equation_st ∗eqn, mess_operation_t op, mess_matrix in, mess_matrix out)
- int(∗ clear)(struct mess_equation_st ∗eqn)
- int to_clear

9.34.1 Detailed Description

Structure defining $op(E)Y = X$ where X and Y are matrices.

9.34.2 Field Documentation

9.34.2.1 int(∗ apply) (struct mess_equation_st ∗eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

Apply the inverse mass matrix to a matrix. Compute $op(E)Y = X$.

9.34.2.2 int(∗ clear) (struct mess_equation_st ∗eqn)

Clear $EINV$ objects separately.

9.34.2.3 int(∗ generate) (struct mess_equation_st ∗eqn)

Post process function if $op(E)Y = X$ needs a global initialization.

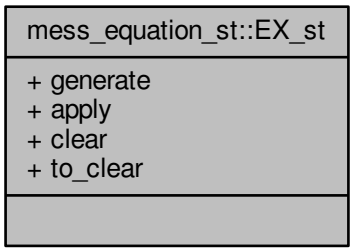
9.34.2.4 int to_clear

Indicator variable if the $EINV$ objects needs to be cleared separately.

9.35 mess_equation_st::EX_st Struct Reference

Structure defining $Y = op(E)X$.

Collaboration diagram for mess_equation_st::EX_st:



Data Fields

- int(∗ generate)(struct mess_equation_st ∗eqn)
- int(∗ apply)(struct mess_equation_st ∗eqn, mess_operation_t op, mess_matrix in, mess_matrix out)
- int(∗ clear)(struct mess_equation_st ∗eqn)
- int to_clear

9.35.1 Detailed Description

Structure defining $Y = op(E)X$ where X and Y are matrices.

9.35.2 Field Documentation

9.35.2.1 int(∗ apply) (struct mess_equation_st ∗eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

Apply the mass matrix to a matrix. Compute $Y = op(E)X$.

9.35.2.2 int(∗ clear) (struct mess_equation_st ∗eqn)

Clear EX objects separately.

9.35.2.3 int(∗ generate) (struct mess_equation_st ∗eqn)

Post process function if $Y = EX$ needs a global initialization.

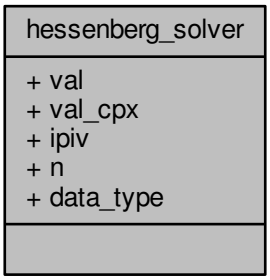
9.35.2.4 int to_clear

Indicator variable if the EX objects needs to be cleared separately.

9.36 hessenberg_solver Struct Reference

Internal structure for the LAPACK based solver.

Collaboration diagram for hessenberg_solver:



Data Fields

- double ∗ val
- mess_double_cpx_t ∗ val_cpx
- mess_int_t ∗ ipiv
- mess_int_t n
- mess_datatype_t data_type

9.36.1 Detailed Description

Attention

Internal use only.

9.36.2 Field Documentation

9.36.2.1 mess_datatype_t data_type

flag real/complex

9.36.2.2 mess_int_t* ipiv

row permutation

9.36.2.3 mess_int_t n

dimension of the system

9.36.2.4 double* val

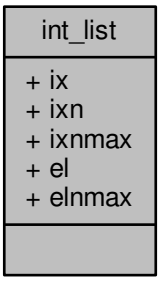
the real lu decomposed matrix

9.36.2.5 mess_double_cpx_t* val_cpx

the complex lu decomposed matrix

9.37 int_list Struct Reference

Collaboration diagram for int_list:



Data Fields

- mess_int_t * ix
- mess_int_t ixn
- mess_int_t ixnmax
- mess_int_t * el
- mess_int_t elnmax

9.37.1 Detailed Description

Data structure for adjency-, level- and similar lists of integers.

Attention

Internal use only.

9.37.2 Field Documentation

9.37.2.1 mess_int_t* el

list of elements, with j and k as described in the ix part, el[j+t] is the t-th element of the i-th part of the list, t<=k-j-1

9.37.2.2 mess_int_t elnmax

maximal length of el, therefore ix[ixn+1]-1<=elnmax has to be asured

9.37.2.3 mess_int_t* ix

index vector, j=ix[i] marks the begin of the i-th part of el, k=ix[i+1]-1 marks the end of the i-th part of el, i<ixn<=ixnmax

9.37.2.4 mess_int_t ixn

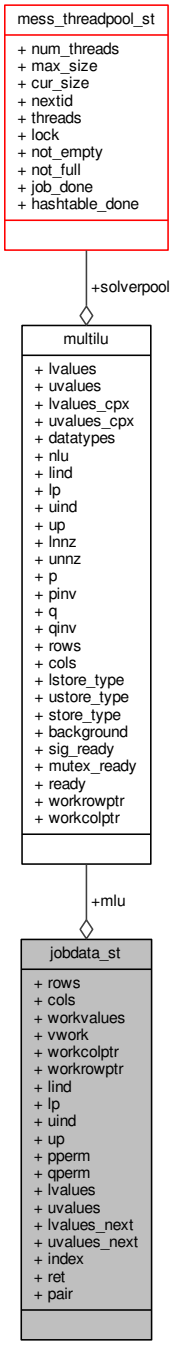
Actual length of the index vector ix

9.37.2.5 mess_int_t ixnmax

Maximal length of the index vector ix

9.38 jobdata_st Struct Reference

Collaboration diagram for jobdata_st:



Data Fields

- mess_int_t rows
- mess_int_t cols
- void * workvalues

- void * vwork
- mess_int_t * workcolptr
- mess_int_t * workrowptr
- mess_int_t * lind
- mess_int_t * lp
- mess_int_t * uind
- mess_int_t * up
- mess_int_t * pperm
- mess_int_t * qperm
- void * lvalues
- void * uvalues
- void * lvalues_next
- void * uvalues_next
- mess_int_t index
- int * ret
- struct multilu * mlu
- int pair

9.38.1 Field Documentation

9.38.1.1 mess_int_t cols

9.38.1.2 mess_int_t index

9.38.1.3 mess_int_t* lind

9.38.1.4 mess_int_t* lp

9.38.1.5 void* lvalues

9.38.1.6 void* lvalues_next

9.38.1.7 struct multilu* mlu

9.38.1.8 int pair

9.38.1.9 mess_int_t* pperm

9.38.1.10 mess_int_t* qperm

9.38.1.11 int* ret

9.38.1.12 mess_int_t rows

9.38.1.13 mess_int_t* uind

9.38.1.14 mess_int_t* up

9.38.1.15 void* uvalues

9.38.1.16 void∗ uvalues_next

9.38.1.17 void∗ vwork

9.38.1.18 mess_int_t∗ workcolptr

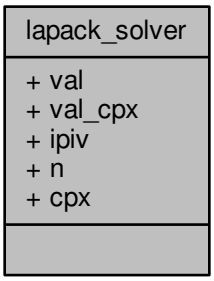
9.38.1.19 mess_int_t∗ workrowptr

9.38.1.20 void∗ workvalues

9.39 lapack_solver Struct Reference

Internal structure for the LAPACK based solver.

Collaboration diagram for lapack_solver:



Data Fields

- double ∗ val
- mess_double_cpx_t ∗ val_cpx
- mess_int_t ∗ ipiv
- mess_int_t n
- unsigned short cpx

9.39.1 Detailed Description

Attention

Internal use only.

9.39.2 Field Documentation

9.39.2.1 unsigned short cpx

flag real/complex

9.39.2.2 mess_int_t* ipiv

row permutation

9.39.2.3 mess_int_t n

dimension of the system

9.39.2.4 double* val

the real lu decomposed matrix

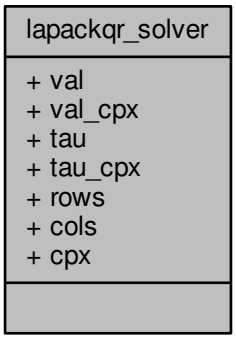
9.39.2.5 mess_double_cpx_t* val_cpx

the complex lu decomposed matrix

9.40 lapackqr_solver Struct Reference

Internal data structure. Internal data structure for QR/LQ decompostitions.

Collaboration diagram for lapackqr_solver:



Data Fields

- double * val
- mess_double_cpx_t * val_cpx
- double * tau
- mess_double_cpx_t * tau_cpx
- mess_int_t rows
- mess_int_t cols
- unsigned short cpx

9.40.1 Detailed Description

Attention

Internal use only.

9.40.2 Field Documentation

9.40.2.1 mess_int_t cols

number of cols of the matrix decomposed to.

9.40.2.2 unsigned short cpx

indicator for complex valued matrices

9.40.2.3 mess_int_t rows

number of rows of the matrix decomposed to.

9.40.2.4 double* tau

real scalar factors of elementary reflectors of QR/LQ decomposition.

9.40.2.5 mess_double_cpx_t* tau_cpx

complex scalar factors of elementary reflectors of QR/LQ decomposition.

9.40.2.6 double* val

real value pointer of a matrix in fortran indexing style

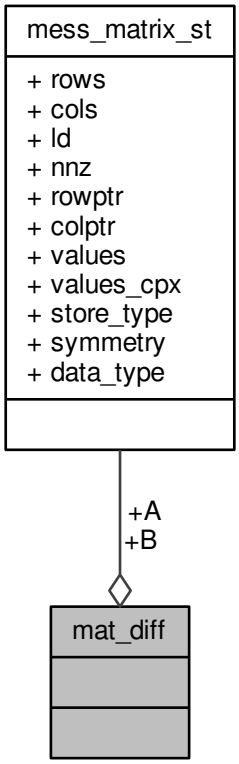
9.40.2.7 mess_double_cpx_t* val_cpx

complex value pointer of a matrix in fortran indexing style.

9.41 mat_diff Struct Reference

Helper structure for mat_diff_mvp.

Collaboration diagram for mat_diff:



Data Fields

- mess_matrix A
- mess_matrix B

9.41.1 Detailed Description

The mat_diff is a helper structure for holding two @ref mess_matrix structures A and B.

Attention

Internal use only.

9.41.2 Field Documentation

9.41.2.1 mess_matrix A

First matrix A

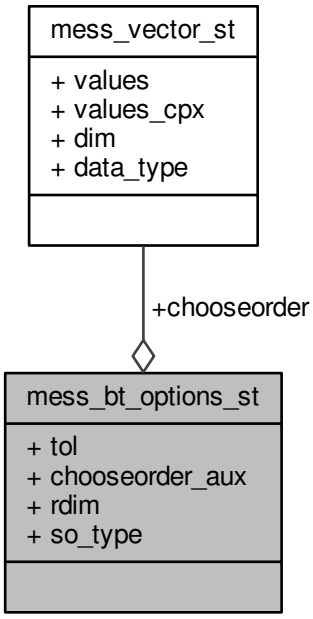
9.41.2.2 mess_matrix B

Second matrix B

9.42 mess_bt_options_st Struct Reference

Options structure for balanced truncation .

Collaboration diagram for mess_bt_options_st:



Data Fields

- double tol
- mess_bt_chooseorder_func chooseorder
- void * chooseorder_aux
- mess_int_t rdim
- unsigned short so_type

9.42.1 Detailed Description

This structure stores the information for the balanced truncation model order reduction algorithms.

9.42.2 Field Documentation

9.42.2.1 mess_bt_chooseorder_func chooseorder

Function to choose the reduced order in your own way

9.42.2.2 void* chooseorder_aux

Additional data for the chooseorder function.

9.42.2.3 mess_int_t rdim

Maximum dimension of the reduced order model

9.42.2.4 unsigned short so_type

Type of second order system

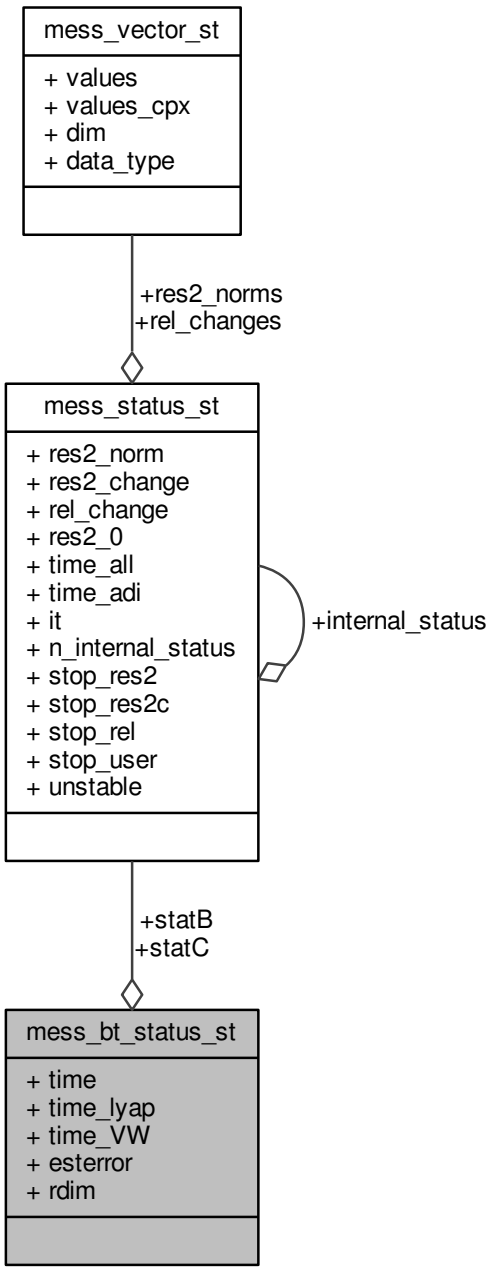
9.42.2.5 double tol

Cut off tolerance for Hankel singular values

9.43 mess_bt_status_st Struct Reference

Status structure for balanced truncation.

Collaboration diagram for mess_bt_status_st:



Data Fields

- double time
- double time_lyap
- double time_VW
- double esterror
- mess_status statB
- mess_status statC
- mess_int_t rdim

9.43.1 Detailed Description

This structure stores the information for the status of the balanced truncation algorithm.

9.43.2 Field Documentation

9.43.2.1 double esterror

Estimated error

9.43.2.2 mess_int_t rdim

Maximum dimension of the reduced order model

9.43.2.3 mess_status statB

LRCF-ADI status for $AX + XA^T = -BB^T$

9.43.2.4 mess_status statC

LRCF-ADI status for $A^TX + XA = -C^TC$

9.43.2.5 double time

Overall time

9.43.2.6 double time_lyap

Time spent on solving Lyapunov Equations

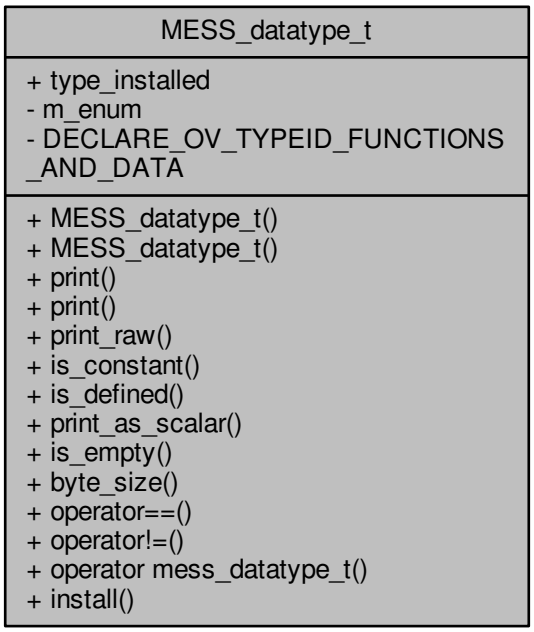
9.43.2.7 double time_VW

Time spent on computing V and W

9.44 MESS_datatype_t Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_datatype_t:



Public Member Functions

- MESS_datatype_t (void)
- MESS_datatype_t (mess_datatype_t enu)
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void)
- bool is_empty (void) const
- size_t byte_size (void) const
- bool operator== (const MESS_datatype_t &b) const
- bool operator!= (const MESS_datatype_t &b) const
- operator mess_datatype_t () const

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_datatype_t m_enum
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.44.1 Constructor & Destructor Documentation

9.44.1.1 MESS_datatype_t (void) [inline]

9.44.1.2 MESS_datatype_t (mess_datatype_t *enu*) [inline]

9.44.2 Member Function Documentation

9.44.2.1 size_t byte_size (void) const [inline]

9.44.2.2 void install (void) [static]

9.44.2.3 bool is_constant (void) const [inline]

9.44.2.4 bool is_defined (void) const [inline]

9.44.2.5 bool is_empty (void) const [inline]

9.44.2.6 operator mess_datatype_t ( ) const [inline]

9.44.2.7 bool operator!= (const MESS_datatype_t & *b*) const [inline]

9.44.2.8 bool operator== (const MESS_datatype_t & *b*) const [inline]

9.44.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const [inline]

9.44.2.10 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) [inline]

9.44.2.11 bool print_as_scalar (void) [inline]

9.44.2.12 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const [inline]

9.44.3 Field Documentation

9.44.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

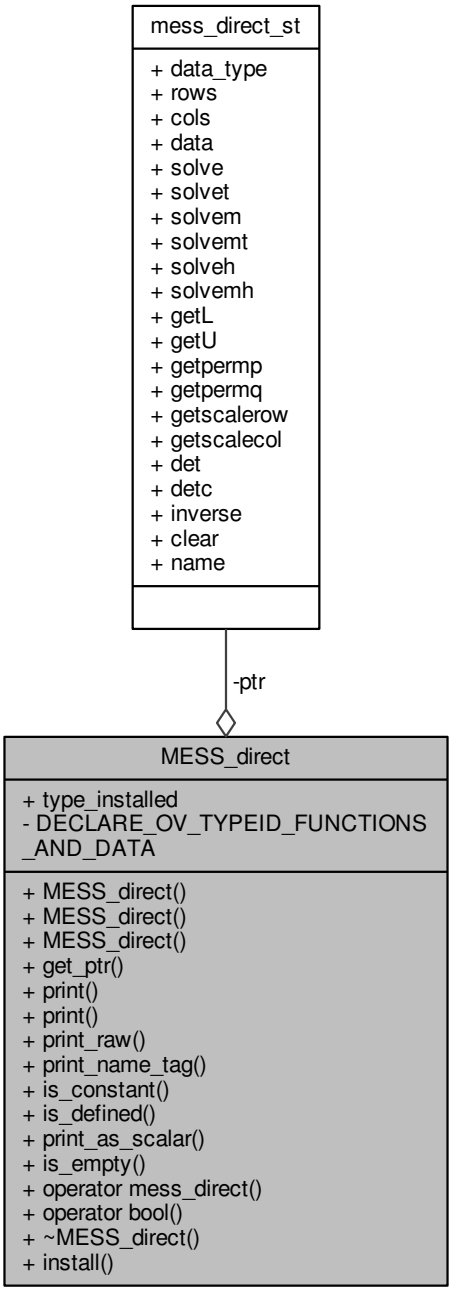
9.44.3.2 mess_datatype_t m_enum [private]

9.44.3.3 bool type_installed = false [static]

9.45 MESS_direct Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_direct:



Public Member Functions

- MESS_direct (void)

- MESS_direct (mess_direct s)
- MESS_direct (const MESS_direct &s)
- mess_direct get_ptr (void) const
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool print_name_tag (std::ostream &os, const std::string &name) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void) const
- bool is_empty (void) const
- operator mess_direct () const
- operator bool () const
- ~MESS_direct (void)

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_direct ptr
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.45.1 Constructor & Destructor Documentation

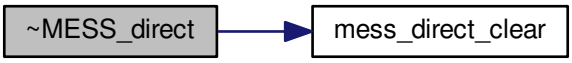
9.45.1.1 MESS_direct (void)

9.45.1.2 MESS_direct (mess_direct *s*)

9.45.1.3 MESS_direct (const MESS_direct & *s*)

9.45.1.4 ~MESS_direct (void)

Here is the call graph for this function:



9.45.2 Member Function Documentation

9.45.2.1 mess_direct get_ptr (void) const

9.45.2.2 void install (void) [static]

9.45.2.3 bool is_constant (void) const

9.45.2.4 bool is_defined (void) const

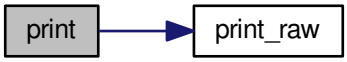
9.45.2.5 bool is_empty (void) const

9.45.2.6 operator bool (void) const

9.45.2.7 operator mess_direct (void) const

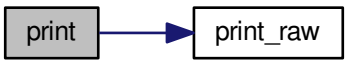
9.45.2.8 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const

Here is the call graph for this function:



9.45.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false)

Here is the call graph for this function:



9.45.2.10 bool print_as_scalar (void) const

9.45.2.11 bool print_name_tag (std::ostream & *os,* const std::string & *name*) const

9.45.2.12 void print_raw (std::ostream & *os,* bool *pr_as_read_syntax*) const

9.45.3 Field Documentation

9.45.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

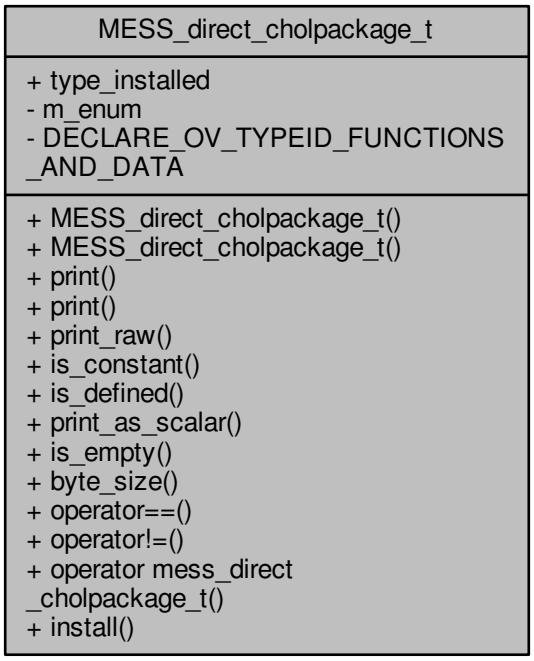
9.45.3.2 mess_direct ptr [private]

9.45.3.3 bool type_installed = false [static]

9.46 MESS_direct_cholpackage_t Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_direct_cholpackage_t:



Public Member Functions

- MESS_direct_cholpackage_t (void)
- MESS_direct_cholpackage_t (mess_direct_cholpackage_t enu)
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void)
- bool is_empty (void) const
- size_t byte_size (void) const
- bool operator== (const MESS_direct_cholpackage_t &b) const
- bool operator!= (const MESS_direct_cholpackage_t &b) const
- operator mess_direct_cholpackage_t () const

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_direct_cholpackage_t m_enum
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.46.1 Constructor & Destructor Documentation

9.46.1.1 MESS_direct_cholpackage_t (void) [inline]

9.46.1.2 MESS_direct_cholpackage_t (mess_direct_cholpackage_t *enu*) [inline]

9.46.2 Member Function Documentation

9.46.2.1 size_t byte_size (void) const [inline]

9.46.2.2 void install (void) [static]

9.46.2.3 bool is_constant (void) const [inline]

9.46.2.4 bool is_defined (void) const [inline]

9.46.2.5 bool is_empty (void) const [inline]

9.46.2.6 operator mess_direct_cholpackage_t ( ) const [inline]

9.46.2.7 bool operator!= (const MESS_direct_cholpackage_t & *b*) const [inline]

9.46.2.8 bool operator== (const MESS_direct_cholpackage_t & *b*) const [inline]

9.46.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const [inline]

9.46.2.10 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) [inline]

9.46.2.11 bool print_as_scalar (void) [inline]

9.46.2.12 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const [inline]

9.46.3 Field Documentation

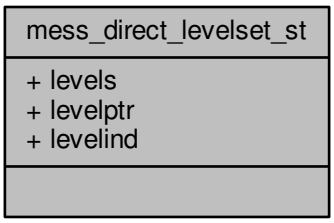
9.46.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

9.46.3.2 mess_direct_cholpackage_t m_enum [private]

9.46.3.3 bool type_installed = false [static]

9.47 mess_direct_levelset_st Struct Reference

Collaboration diagram for mess_direct_levelset_st:



Data Fields

- mess_int_t levels
- mess_int_t * levelptr
- mess_int_t * levelind

9.47.1 Field Documentation

9.47.1.1 mess_int_t* levelind

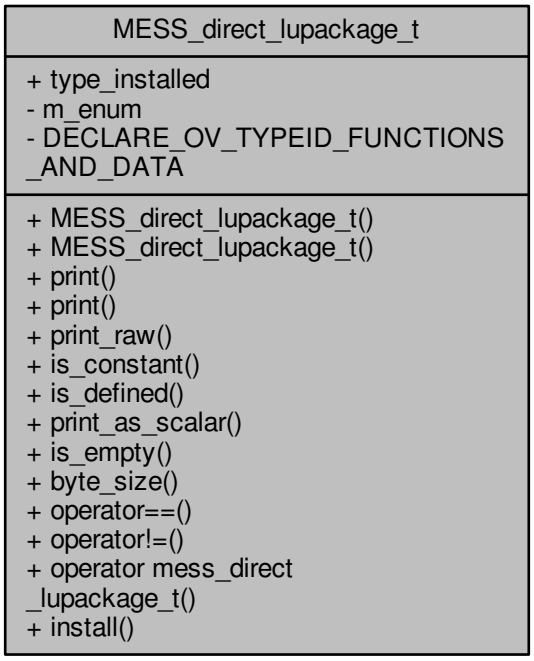
9.47.1.2 mess_int_t* levelptr

9.47.1.3 mess_int_t levels

9.48 MESS_direct_lupackage_t Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_direct_lupackage_t:



Public Member Functions

- MESS_direct_lupackage_t (void)
- MESS_direct_lupackage_t (mess_direct_lupackage_t enu)
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void)
- bool is_empty (void) const
- size_t byte_size (void) const
- bool operator== (const MESS_direct_lupackage_t &b) const
- bool operator!= (const MESS_direct_lupackage_t &b) const
- operator mess_direct_lupackage_t () const

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_direct_lupackage_t m_enum
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.48.1 Constructor & Destructor Documentation

9.48.1.1 MESS_direct_lupackage_t (void) [inline]

9.48.1.2 MESS_direct_lupackage_t (mess_direct_lupackage_t *enu*) [inline]

9.48.2 Member Function Documentation

9.48.2.1 size_t byte_size (void) const [inline]

9.48.2.2 void install (void) [static]

9.48.2.3 bool is_constant (void) const [inline]

9.48.2.4 bool is_defined (void) const [inline]

9.48.2.5 bool is_empty (void) const [inline]

9.48.2.6 operator mess_direct_lupackage_t ( ) const [inline]

9.48.2.7 bool operator!= (const MESS_direct_lupackage_t & *b*) const [inline]

9.48.2.8 bool operator== (const MESS_direct_lupackage_t & *b*) const [inline]

9.48.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const [inline]

9.48.2.10 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) [inline]

9.48.2.11 bool print_as_scalar (void) [inline]

9.48.2.12 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const [inline]

9.48.3 Field Documentation

9.48.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

9.48.3.2 mess_direct_lupackage_t m_enum [private]

9.48.3.3 bool type_installed = false [static]

9.49 mess_direct_st Struct Reference

Representation of a generic direct solver.

Collaboration diagram for mess_direct_st:

Data Fields

- mess_datatype_t data_type
- mess_int_t rows
- mess_int_t cols
- void * data
- int(* solve)(void *data, mess_vector b, mess_vector x)
- int(* solvet)(void *data, mess_vector b, mess_vector x)
- int(* solvem)(void *data, mess_matrix b, mess_matrix x)
- int(* solvemt)(void *data, mess_matrix b, mess_matrix x)
- int(* solveh)(void *data, mess_vector b, mess_vector x)
- int(* solvemh)(void *data, mess_matrix b, mess_matrix x)
- int(* getL)(void *data, mess_matrix L)

- int(∗ getU)(void ∗data, mess_matrix U)
- int(∗ getpermp)(void ∗data, mess_int_t ∗p)
- int(∗ getpermq)(void ∗data, mess_int_t ∗q)
- int(∗ getscalerow)(void ∗data, mess_vector r)
- int(∗ getscalecol)(void ∗data, mess_vector c)
- int(∗ det)(void ∗data, double ∗m, double ∗e)
- int(∗ detc)(void ∗data, double ∗mr, double ∗mi, double ∗e)
- int(∗ inverse)(void ∗data, mess_matrix inv)
- int(∗ clear)(void ∗myself)
- char ∗ name

9.49.1 Detailed Description

The mess_direct_st structure represents a generic direct solver.
Before it can be used it is necessary to initialize the object using mess_direct_init and to clean it up after usage by calling mess_direct_clear.
A full list of direct solvers is available in the Direct Solvers for Linear Systems section. Instead of using this structure directly please use the mess_direct type definition because this hides the pointer to this structure and it is used by all functions. Normally it is not necessary to access components of this structure directly. All operations and actions are performed by functions or macros of this section.

9.49.2 Field Documentation

9.49.2.1 int(∗ clear) (void ∗myself)

Clean up function

9.49.2.2 mess_int_t cols

Number of columns

9.49.2.3 void∗ data

Pointer to the data for the solver

9.49.2.4 mess_datatype_t data_type

Data type of the solver

9.49.2.5 int(∗ det) (void ∗data, double ∗m, double ∗e)

Function to get the real determinant

9.49.2.6 int(∗ detc) (void ∗data, double ∗mr, double ∗mi, double ∗e)

Function to get the complex determinant

9.49.2.7 int(∗ getL) (void ∗data, mess_matrix L)

Function to get a copy of the L factor

9.49.2.8 int(∗ getpermp) (void ∗data, mess_int_t ∗p)

Function to get a copy of the row permuatation

9.49.2.9 int(∗ getpermq) (void ∗data, mess_int_t ∗q)

Function to get a copy of the column permutation

9.49.2.10 int(∗ getscalecol) (void ∗data, mess_vector c)

Function to get a copy of a col scaling numbers

9.49.2.11 int(∗ getscalerow) (void ∗data, mess_vector r)

Function to get a copy of a row scaling numbers

9.49.2.12 int(∗ getU) (void ∗data, mess_matrix U)

Function to get a copy of the U factor

9.49.2.13 int(∗ inverse) (void ∗data, mess_matrix inv)

Function to compute the inverse

9.49.2.14 char∗ name

Name of the solver

9.49.2.15 mess_int_t rows

Number of rows

9.49.2.16 int(∗ solve) (void ∗data, mess_vector b, mess_vector x)

Function to solve $Ax = b$

9.49.2.17 int(∗ solveh) (void ∗data, mess_vector b, mess_vector x)

Function to solve $A^H x = b$

9.49.2.18 int(∗ solvem) (void ∗data, mess_matrix b, mess_matrix x)

Function to solve $AX = B$

9.49.2.19 int(∗ solvemh) (void ∗data, mess_matrix b, mess_matrix x)

Function to solve $A^H X = B$

9.49.2.20 int(∗ solvemt) (void ∗data, mess_matrix b, mess_matrix x)

Function to solve $A^T X = B$

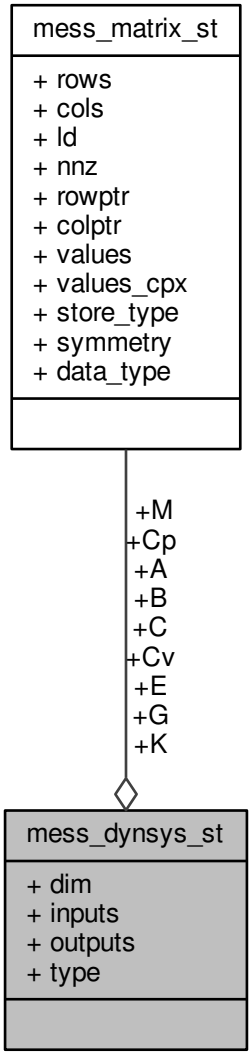
9.49.2.21 int(∗ solvet) (void ∗data, mess_vector b, mess_vector x)

Function to solve $A^T x = b$

9.50 mess_dynsys_st Struct Reference

Representation of a dynamical system.

Collaboration diagram for mess_dynsys_st:



Data Fields

- mess_matrix A
- mess_matrix B
- mess_matrix C
- mess_matrix E
- mess_matrix M
- mess_matrix G
- mess_matrix K
- mess_matrix Cp
- mess_matrix Cv
- mess_int_t dim
- mess_int_t inputs
- mess_int_t outputs
- unsigned short type

9.50.1 Detailed Description

The mess_dynsys_st structure contains all information to represent a dynamical system.

9.50.2 Field Documentation

9.50.2.1 mess_matrix A

System matrix A of a first order system

9.50.2.2 mess_matrix B

Input matrix B of a first and second order system

9.50.2.3 mess_matrix C

Output matrix C of a first order system

9.50.2.4 mess_matrix Cp

Postion output matrix C_p of a second order system

9.50.2.5 mess_matrix Cv

Velocity output matrix C_v of a second order system

9.50.2.6 mess_int_t dim

State space dimension of the system

9.50.2.7 mess_matrix E

Mass matrix E of a first order system

9.50.2.8 mess_matrix G

Matrix for the first derivative in a second order system

9.50.2.9 mess_int_t inputs

Number of inputs

9.50.2.10 mess_matrix K

Matrix for the state in a second order system

9.50.2.11 mess_matrix M

Matrix for the second derivative in a second order system

9.50.2.12 mess_int_t outputs

Number of outputs

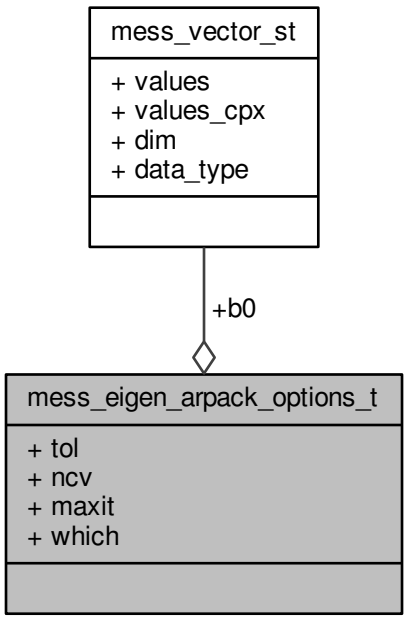
9.50.2.13 unsigned short type

Type of the dynamical system

9.51 mess_eigen_arpack_options_t Struct Reference

Options structure for `ARPACK` .

Collaboration diagram for mess_eigen_arpack_options_t:



Data Fields

- double tol
- mess_vector b0
- mess_int_t ncv
- mess_int_t maxit
- mess_eigen_arpack_which_t which

9.51.1 Detailed Description

The mess_eigen_arpack_options_t structure defines the behavior of ARPACK.
Using the MESS_EIGEN_ARPACK_DEFAULT marco the structure can be set to MATLAB defaults during initialization.
The structure should be initialized using

```
@ref mess_eigen_arpack_options_t arpack_opt =
      MESS_EIGEN_ARPACK_DEFAULT;
```

See also

mess_eigen_arpack
mess_eigen_arpack_which_t

Remarks

The structure is only available if the preprocessor macro MESS_HAVE_APRACK is defined and **M.E.S.S.** is compiled with ARPACK support.

9.51.2 Field Documentation

9.51.2.1 mess_vector b0

Starting vector for the Arnoldi process. If the vector is NULL a randomly chosen vector is used.

9.51.2.2 mess_int_t maxit

Maximum number of Arnoldi update iterations allowed.

9.51.2.3 mess_int_t ncv

Number of Arnoldi vectors to be generated. The default is MIN(MAX(NEV+10,50),N-1), where NEV is the number of desired eigenvalues and N the size of the eigenvalues problem.

9.51.2.4 double tol

Stopping criterion: The relative accuracy of the computed Ritz-values. The default is the machine precision mess_eps ().

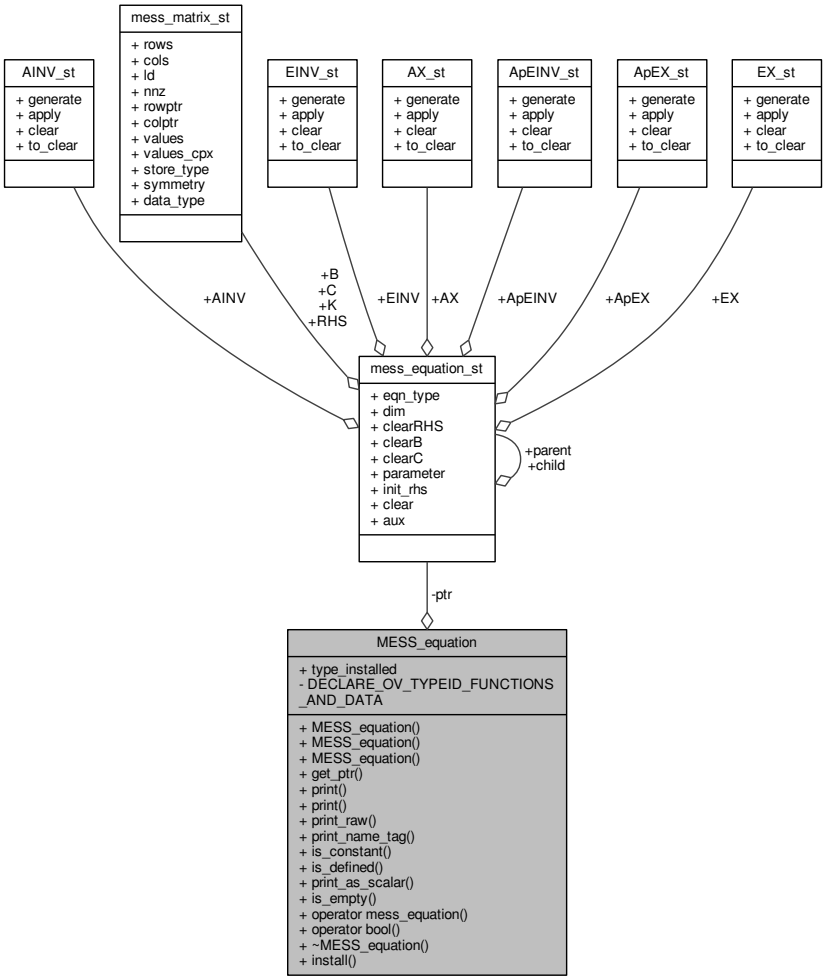
9.51.2.5 mess_eigen_arpack_which_t which

Kind of the desired eigenvalues. See mess_eigen_arpack_which_t for details.

9.52 MESS_equation Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_equation:



Public Member Functions

- MESS_equation (void)
- MESS_equation (mess_equation eqn)
- MESS_equation (const MESS_equation &eqn)
- mess_equation get_ptr (void) const
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)

- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool print_name_tag (std::ostream &os, const std::string &name) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void) const
- bool is_empty (void) const
- operator mess_equation () const
- operator bool () const
- ~MESS_equation (void)

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_equation ptr
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.52.1 Constructor & Destructor Documentation

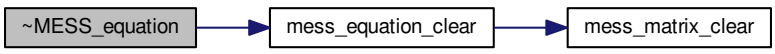
9.52.1.1 MESS_equation (void)

9.52.1.2 MESS_equation (mess_equation *eqn*)

9.52.1.3 MESS_equation (const MESS_equation & *eqn*)

9.52.1.4 ~MESS_equation (void)

Here is the call graph for this function:



9.52.2 Member Function Documentation

9.52.2.1 mess_equation get_ptr (void) const

9.52.2.2 void install (void) [static]

9.52.2.3 bool is_constant (void) const

9.52.2.4 bool is_defined (void) const

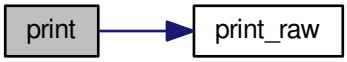
9.52.2.5 bool is_empty (void) const

9.52.2.6 operator bool (void) const

9.52.2.7 operator mess_equation (void) const

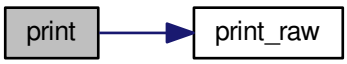
9.52.2.8 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const

Here is the call graph for this function:



9.52.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false)

Here is the call graph for this function:



9.52.2.10 bool print_as_scalar (void) const

9.52.2.11 bool print_name_tag (std::ostream & *os*, const std::string & *name*) const

9.52.2.12 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const

9.52.3 Field Documentation

9.52.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

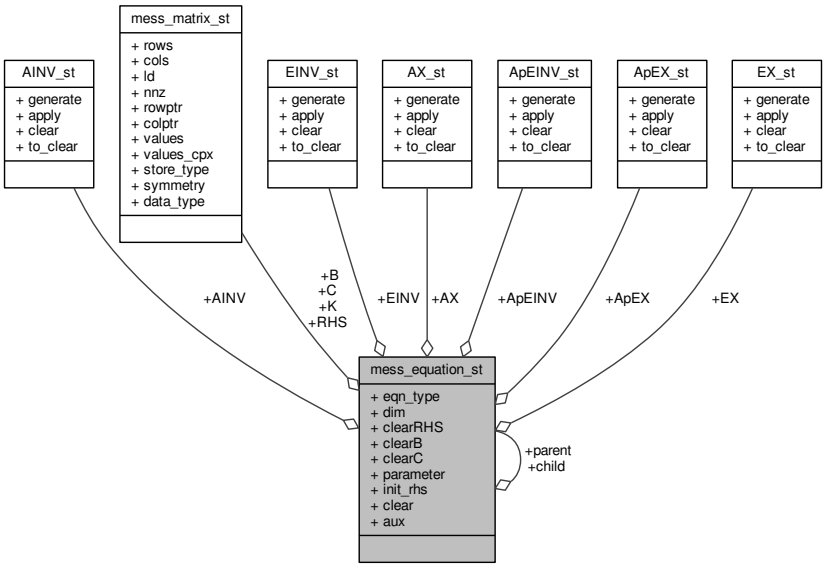
9.52.3.2 mess_equation ptr [private]

9.52.3.3 bool type_installed = false [static]

9.53 mess_equation_st Struct Reference

Structure defining an equation as a set of function pointers.

Collaboration diagram for mess_equation_st:



Data Structures

- struct AINV_st

  Structure defining $op(A)Y = X$.
- struct ApEINV_st

  Structure defining $op(A + pE)Y = X$.
- struct ApEX_st

  Structure defining $Y = op(A + \alpha pE)X$.
- struct AX_st

  Structure defining $Y = op(A)X$.
- struct EINV_st

  Structure defining $op(E)Y = X$.
- struct EX_st

  Structure defining $Y = op(E)X$.

Data Fields

- mess_equation child
- mess_equation parent
- mess_equation_t eqn_type
- mess_int_t dim
- mess_matrix B
- mess_matrix C
- mess_matrix K
- mess_matrix RHS
- mess_int_t clearRHS
- mess_int_t clearB
- mess_int_t clearC
- struct mess_equation_st::AX_st AX
- struct mess_equation_st::EX_st EX
- struct mess_equation_st::AINV_st AINV
- struct mess_equation_st::EINV_st EINV
- struct mess_equation_st::ApEX_st ApEX
- struct mess_equation_st::ApEINV_st ApEINV
- int(∗ parameter)(struct mess_equation_st ∗eqn, mess_options opt, mess_status stat)
- int(∗ init_rhs)(struct mess_equation_st ∗eqn, mess_options opt)
- int(∗ clear)(struct mess_equation_st ∗eqn)
- void ∗ aux

9.53.1 Detailed Description

The mess_equation_st structure contains all function pointers and data that is necessary to define a Lyapunov or Riccati Equation.
All operations depending on the operator are defined as function pointers.
The _generate function is used to initialize data structure once before _apply is called the first time.
The _clear function is called when the equation is destroyed.

9.53.2 Field Documentation

9.53.2.1 struct mess_equation_st::AINV_st AINV

9.53.2.2 struct mess_equation_st::ApEINV_st ApEINV

9.53.2.3 struct mess_equation_st::ApEX_st ApEX

9.53.2.4 void∗ aux

Pointer to the internal data structure.

9.53.2.5 struct mess_equation_st::AX_st AX

9.53.2.6 mess_matrix B

Matrix B of the operator.

9.53.2.7 mess_matrix C

Matrix C of the operator

9.53.2.8 mess_equation child

Pointer to child equation object.

9.53.2.9 int(∗ clear) (struct mess_equation_st ∗eqn)

Final clean up function.

9.53.2.10 mess_int_t clearB

True if B matrix needs to be cleared.

9.53.2.11 mess_int_t clearC

True if C matrix needs to be cleared.

9.53.2.12 mess_int_t clearRHS

True if the right hand side needs to be cleared.

9.53.2.13 mess_int_t dim

State space dimension of the equation.

9.53.2.14 struct mess_equation_st::EINV_st EINV

9.53.2.15 mess_equation_t eqn_type

Type of the equation (Lyapunov, Riccati, etc..).

9.53.2.16 struct mess_equation_st::EX_st EX

9.53.2.17 int(∗ init_rhs) (struct mess_equation_st ∗eqn, mess_options opt)

Initialize the right hand side eqn_parent/eqn structure, if eqn_parent is set to `NULL` eqn is used.

9.53.2.18 mess_matrix K

Matrix K of ther operator $(A - BK)$ or $(A - CK)$

9.53.2.19 int(∗ parameter) (struct mess_equation_st ∗eqn, mess_options opt, mess_status stat)

Get shift parameter function.

9.53.2.20 mess_equation parent

Pointer to parent equation object.

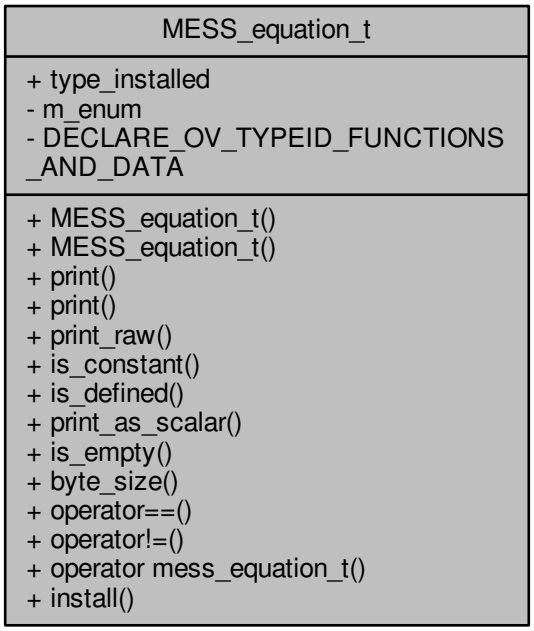
9.53.2.21 mess_matrix RHS

Right hand side for the current Lyapunov Equation, if `NULL` B is used.

9.54 MESS_equation_t Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_equation_t:



Public Member Functions

- MESS_equation_t (void)
- MESS_equation_t (mess_equation_t enu)
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void)
- bool is_empty (void) const
- size_t byte_size (void) const
- bool operator== (const MESS_equation_t &b) const
- bool operator!= (const MESS_equation_t &b) const
- operator mess_equation_t () const

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_equation_t m_enum
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.54.1 Constructor & Destructor Documentation

9.54.1.1 MESS_equation_t (void) [inline]

9.54.1.2 MESS_equation_t (mess_equation_t *enu*) [inline]

9.54.2 Member Function Documentation

9.54.2.1 size_t byte_size (void) const [inline]

9.54.2.2 void install (void) [static]

9.54.2.3 bool is_constant (void) const [inline]

9.54.2.4 bool is_defined (void) const [inline]

9.54.2.5 bool is_empty (void) const [inline]

9.54.2.6 operator mess_equation_t ( ) const [inline]

9.54.2.7 bool operator!= (const MESS_equation_t & *b*) const [inline]

9.54.2.8 bool operator== (const MESS_equation_t & *b*) const [inline]

9.54.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const [inline]

9.54.2.10 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) [inline]

9.54.2.11 bool print_as_scalar (void) [inline]

9.54.2.12 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const [inline]

9.54.3 Field Documentation

9.54.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

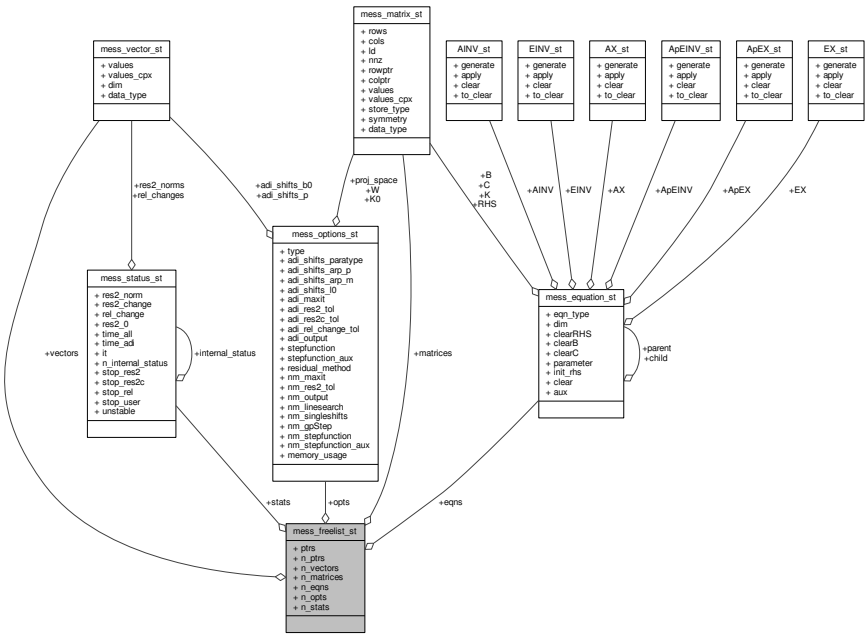
9.54.3.2 mess_equation_t m_enum [private]

9.54.3.3 bool type_installed = false [static]

9.55 mess_freelist_st Struct Reference

Structure containing a minimal memory manage in order to avoid memory leaks in the Python and **MEX-M.E.S.S.** interface.

Collaboration diagram for mess_freelist_st:



Data Fields

- void ∗∗ ptrs
- mess_vector ∗ vectors
- mess_matrix ∗ matrices
- mess_equation ∗ eqns
- mess_options ∗ opts
- mess_status ∗ stats
- size_t n_ptrs
- size_t n_vectors
- size_t n_matrices
- size_t n_eqns
- size_t n_opts
- size_t n_stats

9.55.1 Detailed Description

Due to the fact that C does not have a gargabe collector and **M.E.S.S.** does not implement reference counting we have to track the allocated data in some cases.

Attention

Interal use only.

9.55.2 Field Documentation

9.55.2.1 mess_equation* eqns

Array containing the mess_equation instances.

9.55.2.2 mess_matrix* matrices

Array containing the mess_matrix instances.

9.55.2.3 size_t n_eqns

Length of the eqns array.

9.55.2.4 size_t n_matrices

Length of the matrix array.

9.55.2.5 size_t n_opts

Length of the opts array.

9.55.2.6 size_t n_ptrs

Length of the ptrs array.

9.55.2.7 size_t n_stats

Length of the stats array.

9.55.2.8 size_t n_vectors

Length of the vector array.

9.55.2.9 mess_options* opts

Array containing the mess_options instances.

9.55.2.10 void** ptrs

Array containing arbitrary pointers.

9.55.2.11 mess_status* stats

Array containing the mess_status instances.

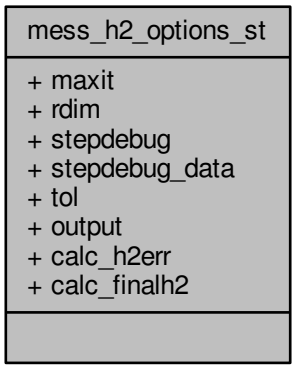
9.55.2.12 mess_vector* vectors

Array containing the mess_vector instances.

9.56 mess_h2_options_st Struct Reference

Structure to store $\mathcal{H}_2$ options.

Collaboration diagram for mess_h2_options_st:



Data Fields

- mess_int_t maxit
- mess_int_t rdim
- void(∗ stepdebug)(void ∗data, mess_int_t it, double sigmadiff, double h2err)
- void ∗ stepdebug_data
- double tol
- int output
- int calc_h2err
- int calc_finalh2

9.56.1 Detailed Description

This structure stores the information for the $\mathcal{H}_2$ model order reduction algorithms. It never be used directly but with the typedef mess_h2_options.

9.56.2 Field Documentation

9.56.2.1 int calc_finalh2

Should a final $\mathcal{H}_2$ error be computed.

9.56.2.2 int calc_h2err

Should the $\mathcal{H}_2$ error be computed during the iteration.

9.56.2.3 mess_int_t maxit

Maximum number of iteration

9.56.2.4 int output

Should output be created.

9.56.2.5 mess_int_t rdim

Dimension of the reduced order model.

9.56.2.6 void(∗ stepdebug) (void ∗data, mess_int_t it, double sigmadiff, double h2err)

Stepdebug function which is called every iteration, if wanted.

9.56.2.7 void∗ stepdebug_data

Auxillary data for the debug function

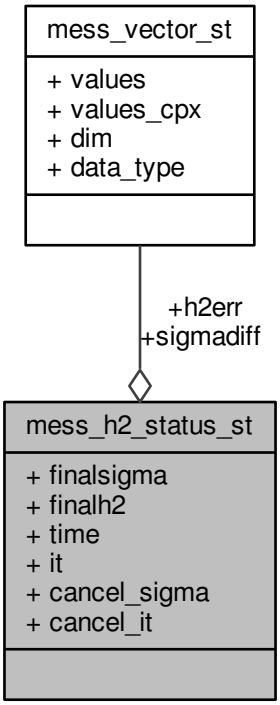
9.56.2.8 double tol

Cancellation tolerance

9.57 mess_h2_status_st Struct Reference

Structure to store $\mathcal{H}_2$ status.

Collaboration diagram for mess_h2_status_st:



Data Fields

- double finalsigma
- double finalh2
- double time
- mess_vector h2err
- mess_vector sigmadiff
- mess_int_t it
- int cancel_sigma
- int cancel_it

9.57.1 Detailed Description

This structure stores the information for the status of the $\mathcal{H}_2$ error, the difference between a system and its reduced one, the number of (needed) iterations and the time.

9.57.2 Field Documentation

9.57.2.1 int cancel_it

Number of iteration where the algorithm stops, 1 if tolerance is not reached.

9.57.2.2 int cancel_sigma

Boolean value, 1 if tolerance is reached.

9.57.2.3 double finalh2

$\mathcal{H}_2$-error in the end

9.57.2.4 double finalsigma

Difference between a system and its reduced one in the end

9.57.2.5 mess_vector h2err

$\mathcal{H}_2$-error

9.57.2.6 mess_int_t it

Number of iterations

9.57.2.7 mess_vector sigmadiff

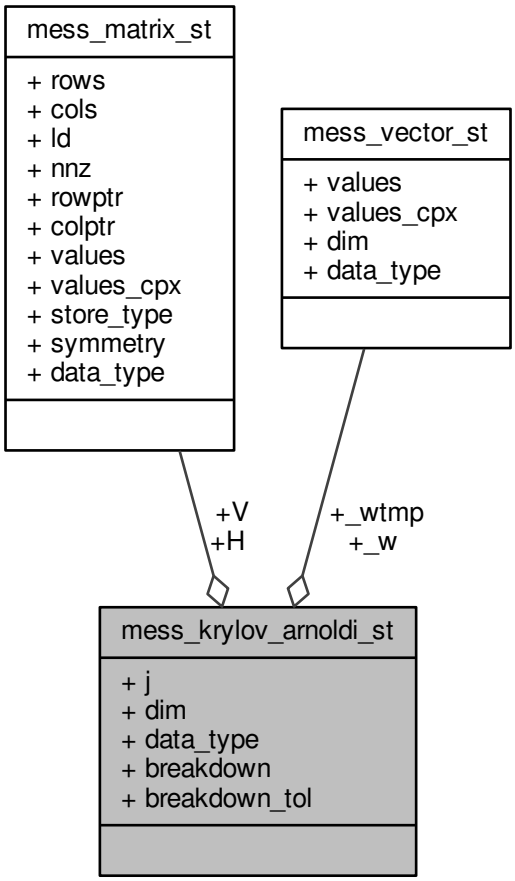
Difference between a system and its reduced one

9.57.2.8 double time

Required time

9.58 mess_krylov_arnoldi_st Struct Reference

Collaboration diagram for mess_krylov_arnoldi_st:



Data Fields

- mess_int_t j
- mess_int_t dim
- mess_datatype_t data_type
- mess_int_t breakdown
- double breakdown_tol
- mess_matrix V
- mess_matrix H
- mess_vector _w
- mess_vector _wtmp

9.58.1 Detailed Description

Internal structure holds data for Arnoldi process.

Attention

Internal use only.

9.58.2 Field Documentation

9.58.2.1 mess_vector _w

Upper Hessnberg Matrix.

9.58.2.2 mess_vector _wtmp

Cache.

9.58.2.3 mess_int_t breakdown

Data type of the operator used.

9.58.2.4 double breakdown_tol

If nonzero (early) breakdown occured during Arnoldi process.

9.58.2.5 mess_datatype_t data_type

Order of the operator used.

9.58.2.6 mess_int_t dim

Number of Arnoldi steps taken.

9.58.2.7 mess_matrix H

Arnoldi vectors as columns.

9.58.2.8 mess_int_t j

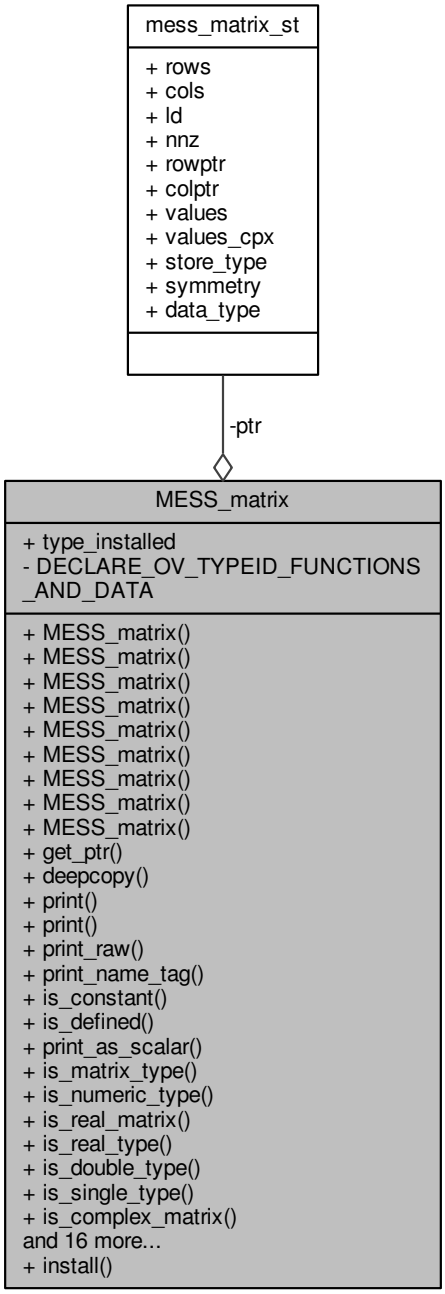
9.58.2.9 mess_matrix V

Tolerance for breakdown.

9.59 MESS_matrix Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_matrix:



Public Member Functions

- MESS_matrix (void)

- MESS_matrix (mess_matrix)
- MESS_matrix (Matrix m)
- MESS_matrix (ComplexMatrix m)
- MESS_matrix (SparseMatrix m)
- MESS_matrix (SparseComplexMatrix m)
- MESS_matrix (double m)
- MESS_matrix (Complex m)
- MESS_matrix (const MESS_matrix &m)
- mess_matrix get_ptr (void) const
- MESS_matrix deepcopy (void) const
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool print_name_tag (std::ostream &os, const std::string &name) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void) const
- bool is_matrix_type (void) const
- bool is_numeric_type (void) const
- bool is_real_matrix (void) const
- bool is_real_type (void) const
- bool is_double_type (void) const
- bool is_single_type (void) const
- bool is_complex_matrix (void) const
- bool is_complex_type (void) const
- bool is_sparse_type (void) const
- bool is_empty (void) const
- dim_vector dims (void) const
- int ndims (void) const
- size_t length (void) const
- size_t byte_size (void) const
- octave_value full_value (void) const
- octave_value map (unary_mapper_t umap) const
- Matrix matrix_value (bool=false) const
- ComplexMatrix complex_matrix_value (bool=false) const
- SparseMatrix sparse_matrix_value (bool=false) const
- SparseComplexMatrix sparse_complex_matrix_value (bool=false) const
- operator mess_matrix () const
- operator bool () const
- ~MESS_matrix (void)

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_matrix ptr
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

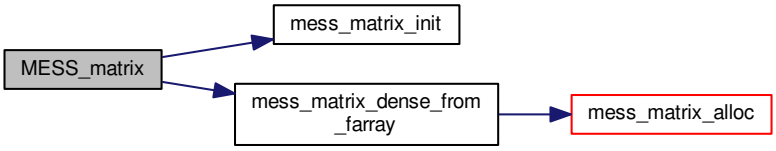
9.59.1 Constructor & Destructor Documentation

9.59.1.1 MESS_matrix (void)

9.59.1.2 MESS_matrix (mess_matrix *m*)

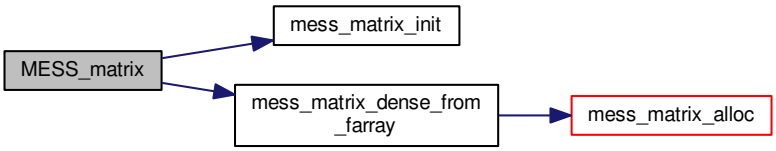
9.59.1.3 MESS_matrix (Matrix *m*)

Here is the call graph for this function:



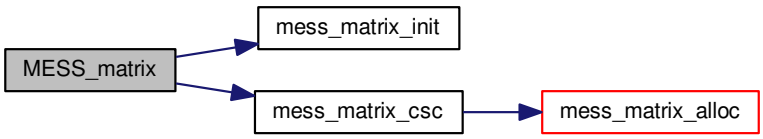
9.59.1.4 MESS_matrix (ComplexMatrix *m*)

Here is the call graph for this function:



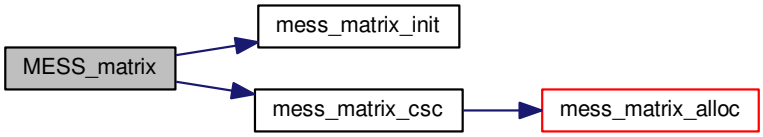
9.59.1.5 MESS_matrix (SparseMatrix *m*)

Here is the call graph for this function:



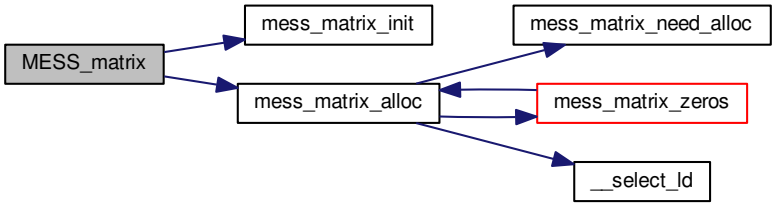
9.59.1.6 MESS_matrix (SparseComplexMatrix *m*)

Here is the call graph for this function:



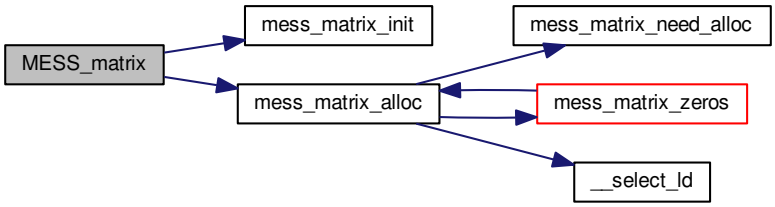
9.59.1.7 MESS_matrix (double *m*)

Here is the call graph for this function:



9.59.1.8 MESS_matrix (Complex *m*)

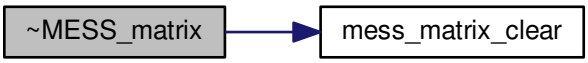
Here is the call graph for this function:



9.59.1.9 MESS_matrix (const MESS_matrix & *m*)

9.59.1.10 ~MESS_matrix (void)

Here is the call graph for this function:



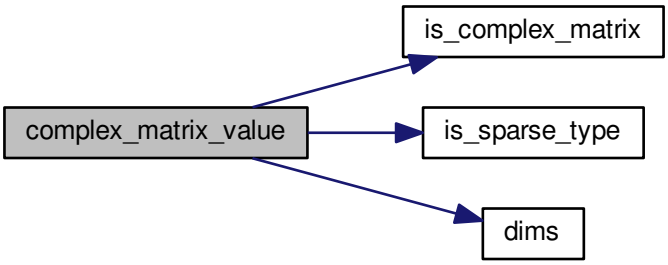
9.59.2 Member Function Documentation

9.59.2.1 size_t byte_size (void) const

Here is the call graph for this function:

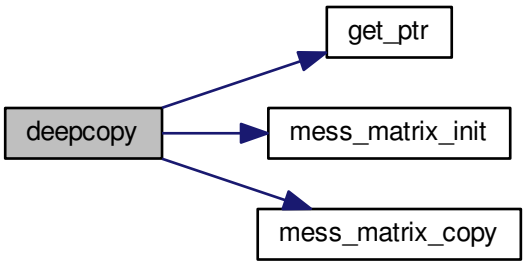
9.59.2.2 ComplexMatrix complex_matrix_value (bool = `false`) const

Here is the call graph for this function:



9.59.2.3 MESS_matrix deepcopy (void) const

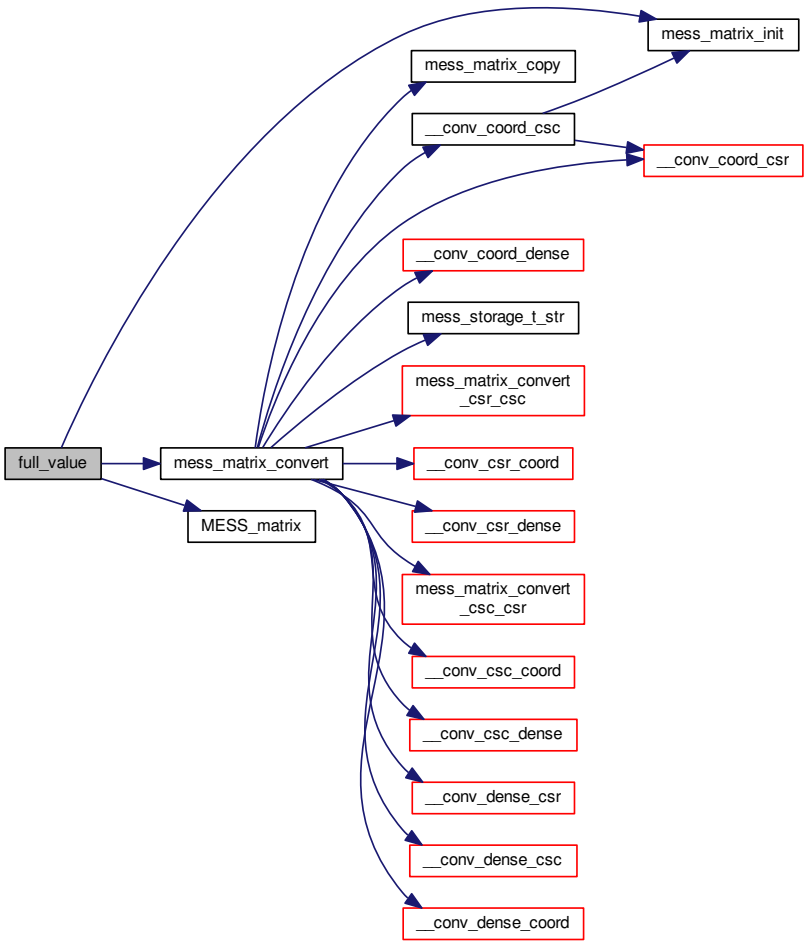
Here is the call graph for this function:



9.59.2.4 dim_vector dims (void) const

9.59.2.5 octave_value full_value (void) const

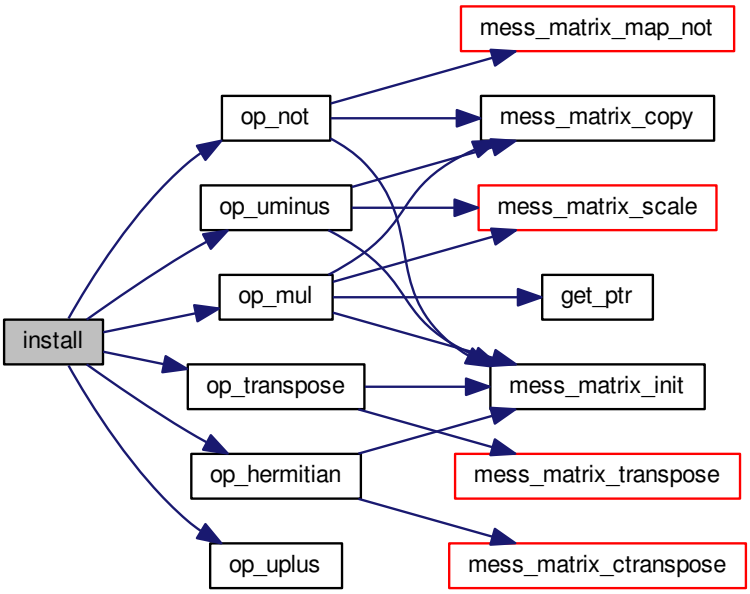
Here is the call graph for this function:



9.59.2.6 mess_matrix get_ptr (void) const

9.59.2.7 void install (void) [static]

Here is the call graph for this function:



9.59.2.8 bool is_complex_matrix (void) const

9.59.2.9 bool is_complex_type (void) const

9.59.2.10 bool is_constant (void) const

9.59.2.11 bool is_defined (void) const

9.59.2.12 bool is_double_type (void) const

9.59.2.13 bool is_empty (void) const

9.59.2.14 bool is_matrix_type (void) const

9.59.2.15 bool is_numeric_type (void) const

9.59.2.16 bool is_real_matrix (void) const

9.59.2.17 bool is_real_type (void) const

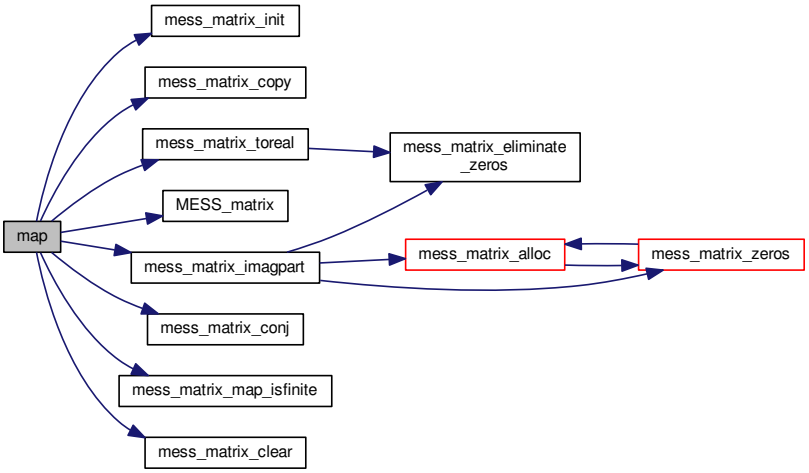
9.59.2.18 bool is_single_type (void) const

9.59.2.19 bool is_sparse_type (void) const

9.59.2.20 size_t length (void) const

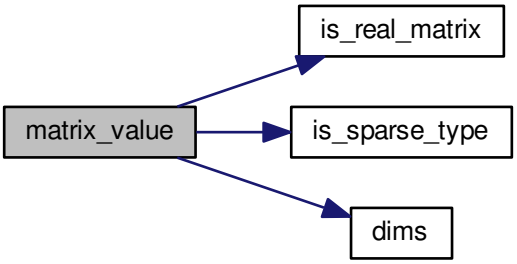
9.59.2.21 octave_value map (unary_mapper_t *umap*) const

Here is the call graph for this function:



9.59.2.22 Matrix matrix_value (bool = `false`) const

Here is the call graph for this function:



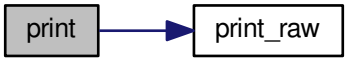
9.59.2.23 int ndims (void) const

9.59.2.24 operator bool (void) const

9.59.2.25 operator mess_matrix (void) const

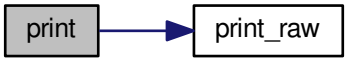
9.59.2.26 void print (std::ostream & *os*, bool *pr_as_read_syntax* = `false`) const

Here is the call graph for this function:



9.59.2.27 void print (std::ostream & *os*, bool *pr_as_read_syntax* = `false`)

Here is the call graph for this function:



9.59.2.28 bool print_as_scalar (void) const

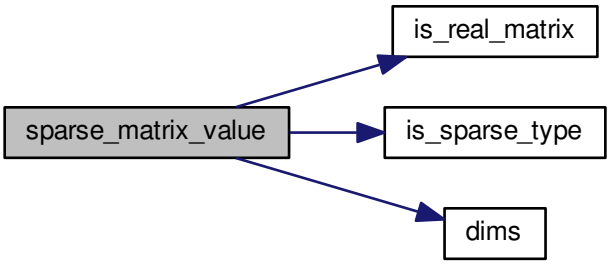
9.59.2.29 bool print_name_tag (std::ostream & *os*, const std::string & *name*) const

9.59.2.30 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const

9.59.2.31 SparseComplexMatrix sparse_complex_matrix_value (bool = `false`) const

9.59.2.32 SparseMatrix sparse_matrix_value (bool = `false`) const

Here is the call graph for this function:



9.59.3 Field Documentation

9.59.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA `[private]`

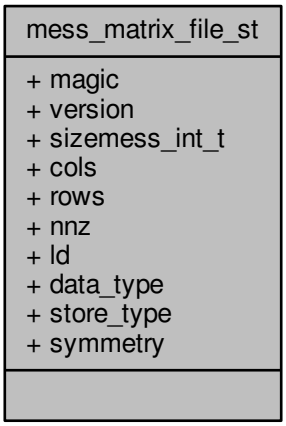
9.59.3.2 mess_matrix ptr `[private]`

9.59.3.3 bool type_installed = false `[static]`

9.60 mess_matrix_file_st Struct Reference

Type definition and structure for mess_matrix files.

Collaboration diagram for mess_matrix_file_st:



Data Fields

- int8_t magic [6]
- int8_t version
- int8_t sizemess_int_t
- int64_t cols
- int64_t rows
- int64_t nnz
- int64_t ld
- int8_t data_type
- int8_t store_type
- int8_t symmetry

9.60.1 Detailed Description

The mess_matrix_file type definition and structure defines a mess_matrix file.

9.60.2 Field Documentation

9.60.2.1 int64_t cols

Number of columns

9.60.2.2 int8_t data_type

Data type

9.60.2.3 int64_t ld

Number of leading dimension

9.60.2.4 int8_t magic[6]

Mess_matrix binary file

9.60.2.5 int64_t nnz

Number of non zero elements

9.60.2.6 int64_t rows

Number of rows

9.60.2.7 int8_t sizemess_int_t

Size of mess_int_t in file

9.60.2.8 int8_t store_type

Storage type

9.60.2.9 int8_t symmetry

Symmetry

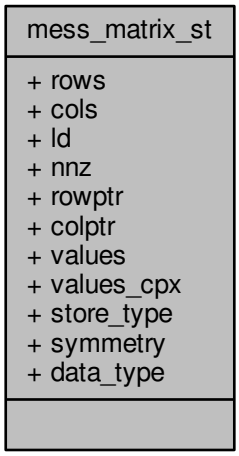
9.60.2.10 int8_t version

Binary format version

9.61 mess_matrix_st Struct Reference

The mess_matrix structure to handle dense and sparse matrices in a common context.

Collaboration diagram for mess_matrix_st:



Data Fields

- mess_int_t rows
- mess_int_t cols
- mess_int_t ld
- mess_int_t nnz
- mess_int_t ∗ rowptr
- mess_int_t ∗ colptr
- double ∗ values
- mess_double_cpx_t ∗ values_cpx
- mess_storage_t store_type
- mess_symmetry_t symmetry
- mess_datatype_t data_type

9.61.1 Detailed Description

This structure represents a matrix which is stored as compressed row, compressed column or dense storage. The used storage stored matrix is determined by the store_type property. In case of dense storage the matrix is stored in the Fortran format. Otherwise a 0 based indirect addressing format like MESS_CSR, MESS_CSC or MESS_COORD is used.

The data_type flag indicates the data type of the matrix. It is set to one of the values of the mess_datatype_t enumeration. If the matrix is real the values array is used to store the data. In case of a complex matrix the values_cpx array is used. The unused one of those two arrays should be set to `NULL`.

See also

mess_matrix_convert

Attention

The symmetry flag is supposed to indicate whether the matrix is symmetric or not. Unfortunately we do not use this at the moment. This is the reason why the symmetry is always set to MESS_GENERAL even if you read a symmetric matrix from a file.

9.61.2 Field Documentation

9.61.2.1 mess_int_t* colptr

Array to store the column pointer or column indices. `NULL` for dense matrices.

9.61.2.2 mess_int_t cols

Number of columns of the matrix.

9.61.2.3 mess_datatype_t data_type

Determine the data type of the matrix. This flag identifies whether values or values_cpx is used.

9.61.2.4 mess_int_t ld

Leading dimension of the matrix. This is only used for dense matrices.

9.61.2.5 mess_int_t nnz

Number of non zero elements in the matrix. This is only used for sparse matrices.

9.61.2.6 mess_int_t* rowptr

Array to store the row pointer or the row indices. `NULL` for dense matrices.

9.61.2.7 mess_int_t rows

Number of rows of the matrix.

9.61.2.8 mess_storage_t store_type

Identify the matrix storage type. This can be one of the mess_storage_t enumeration.

9.61.2.9 mess_symmetry_t symmetry

Symmetry flag. This flag is always set to MESS_GENERAL at the moment. Primarily used to support implicit symmetry.

See also

mess_symmetry_t

9.61.2.10 double* values

Array containing the real matrix entries.

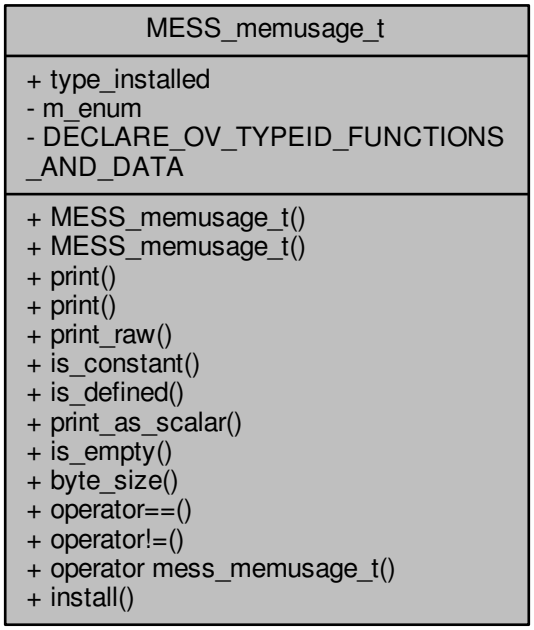
9.61.2.11 mess_double_cpx_t* values_cpx

Array containing the complex matrix entries.

9.62 MESS_memusage_t Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_memusage_t:



Public Member Functions

- MESS_memusage_t (void)
- MESS_memusage_t (mess_memusage_t enu)
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void)
- bool is_empty (void) const
- size_t byte_size (void) const
- bool operator== (const MESS_memusage_t &b) const
- bool operator!= (const MESS_memusage_t &b) const
- operator mess_memusage_t () const

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_memusage_t m_enum
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.62.1 Constructor & Destructor Documentation

9.62.1.1 MESS_memusage_t (void) [inline]

9.62.1.2 MESS_memusage_t (mess_memusage_t *enu*) [inline]

9.62.2 Member Function Documentation

9.62.2.1 size_t byte_size (void) const [inline]

9.62.2.2 void install (void) [static]

9.62.2.3 bool is_constant (void) const [inline]

9.62.2.4 bool is_defined (void) const [inline]

9.62.2.5 bool is_empty (void) const [inline]

9.62.2.6 operator mess_memusage_t ( ) const [inline]

9.62.2.7 bool operator!= (const MESS_memusage_t & *b*) const [inline]

9.62.2.8 bool operator== (const MESS_memusage_t & *b*) const [inline]

9.62.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const [inline]

9.62.2.10 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) [inline]

9.62.2.11 bool print_as_scalar (void) [inline]

9.62.2.12 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const [inline]

9.62.3 Field Documentation

9.62.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

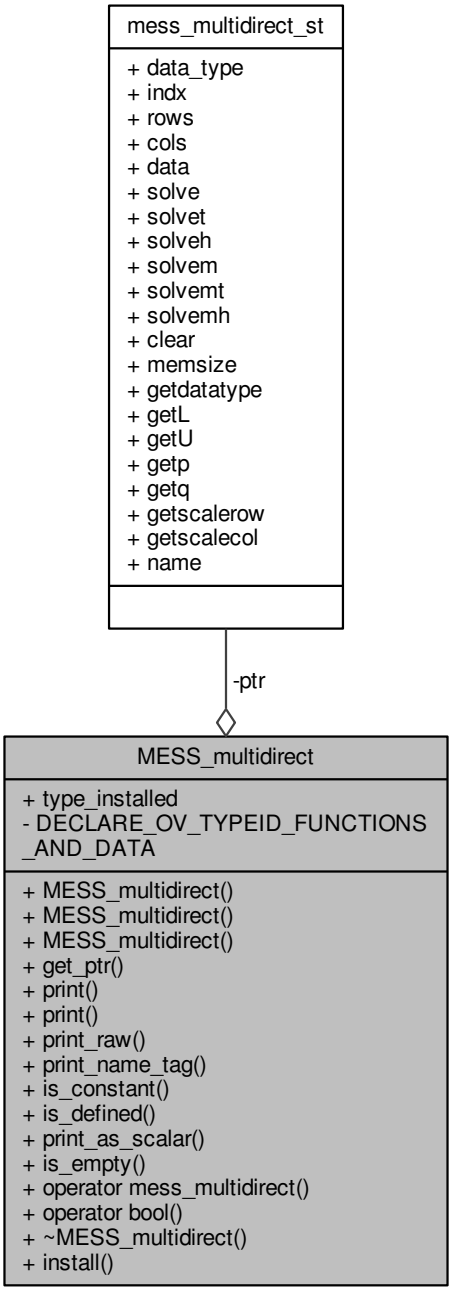
9.62.3.2 mess_memusage_t m_enum [private]

9.62.3.3 bool type_installed = false [static]

9.63 MESS_multidirect Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_multidirect:



Public Member Functions

- MESS_multidirect (void)
- MESS_multidirect (mess_multidirect s)
- MESS_multidirect (const MESS_multidirect &s)
- mess_multidirect get_ptr (void) const
- void print (std::ostream &os, bool pr_as_read_syntax=false) const

- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool print_name_tag (std::ostream &os, const std::string &name) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void) const
- bool is_empty (void) const
- operator mess_multidirect () const
- operator bool () const
- ~MESS_multidirect (void)

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_multidirect ptr
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.63.1 Constructor & Destructor Documentation

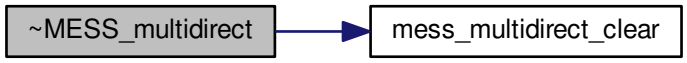
9.63.1.1 MESS_multidirect (void)

9.63.1.2 MESS_multidirect (mess_multidirect *s*)

9.63.1.3 MESS_multidirect (const MESS_multidirect & *s*)

9.63.1.4 ~MESS_multidirect (void)

Here is the call graph for this function:



9.63.2 Member Function Documentation

9.63.2.1 mess_multidirect get_ptr (void) const

9.63.2.2 void install (void) [static]

9.63.2.3 bool is_constant (void) const

9.63.2.4 bool is_defined (void) const

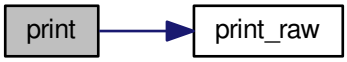
9.63.2.5 bool is_empty (void) const

9.63.2.6 operator bool (void) const

9.63.2.7 operator mess_multidirect (void) const

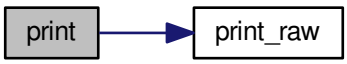
9.63.2.8 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const

Here is the call graph for this function:



9.63.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false)

Here is the call graph for this function:



9.63.2.10 bool print_as_scalar (void) const

9.63.2.11 bool print_name_tag (std::ostream & *os,* const std::string & *name*) const

9.63.2.12 void print_raw (std::ostream & *os,* bool *pr_as_read_syntax*) const

9.63.3 Field Documentation

9.63.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

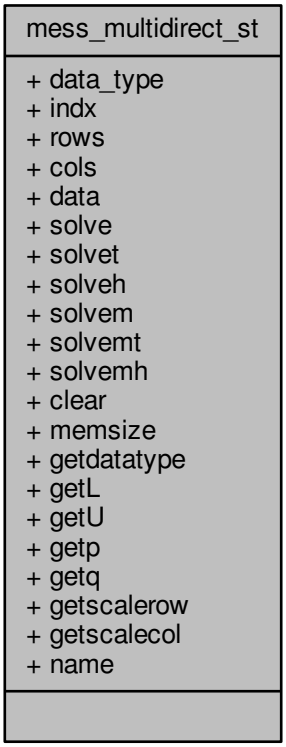
9.63.3.2 mess_multidirect ptr [private]

9.63.3.3 bool type_installed = false [static]

9.64 mess_multidirect_st Struct Reference

Representation of a generic direct solver.

Collaboration diagram for mess_multidirect_st:



Data Fields

- mess_datatype_t data_type
- mess_int_t indx
- mess_int_t rows
- mess_int_t cols
- void ∗ data
- int(∗ solve)(void ∗data, mess_int_t ind, mess_vector b, mess_vector x)
- int(∗ solvet)(void ∗data, mess_int_t ind, mess_vector b, mess_vector x)
- int(∗ solveh)(void ∗data, mess_int_t ind, mess_vector b, mess_vector x)
- int(∗ solvem)(void ∗data, mess_int_t ind, mess_matrix b, mess_matrix x)
- int(∗ solvemt)(void ∗data, mess_int_t ind, mess_matrix b, mess_matrix x)
- int(∗ solvemh)(void ∗data, mess_int_t ind, mess_matrix b, mess_matrix x)
- int(∗ clear)(void ∗myself)
- size_t(∗ memsize)(void ∗data)
- unsigned short(∗ getdatatype)(void ∗myself, mess_int_t ind)
- int(∗ getL)(void ∗data, mess_int_t ind, mess_matrix L)
- int(∗ getU)(void ∗data, mess_int_t ind, mess_matrix U)
- int(∗ getp)(void ∗data, mess_int_t ind, mess_int_t ∗p)
- int(∗ getq)(void ∗data, mess_int_t ind, mess_int_t ∗q)
- int(∗ getscalerow)(void ∗data, mess_int_t ind, mess_vector r)
- int(∗ getscalecol)(void ∗data, mess_int_t ind, mess_vector c)
- char ∗ name

9.64.1 Detailed Description

The concept of mess_multidirect solvers in **M.E.S.S.** represents a set of direct solvers of the structure

$$(t_i A + p_i E)x = b,$$

where t_i and p_i are a set of shift parameters.
The multi-direct solvers try to reuse as much information as possible to squeeze down the memory footprint.
This structure represents a generic interface for this class of solvers. The structure is used as a pointer and so the mess_multidirect type definition hides one pointer if this structure is used.
Accessing components of this structure is normally not necessary and all operations are covered by functions and macros of this section.

A list of possible mess_multidirect solvers is available in the Multisolvers section.

9.64.2 Field Documentation

9.64.2.1 int(∗ clear) (void ∗myself)

Function to clean up

9.64.2.2 mess_int_t cols

Number of columns

9.64.2.3 void∗ data

Pointer to the solver data

9.64.2.4 mess_datatype_t data_type

Data type of the solver

9.64.2.5 unsigned short(* getdatatype) (void *myself, mess_int_t ind)

Get the data type of a specified solver

9.64.2.6 int(* getL) (void *data, mess_int_t ind, mess_matrix L)

Get the L factor of a decomposition

9.64.2.7 int(* getp) (void *data, mess_int_t ind, mess_int_t *p)

Get the row permutation of a decomposition

9.64.2.8 int(* getq) (void *data, mess_int_t ind, mess_int_t *q)

Get the column permutation of a decomposition

9.64.2.9 int(* getscalecol) (void *data, mess_int_t ind, mess_vector c)

Function to get a copy of a col scaling numbers

9.64.2.10 int(* getscalerow) (void *data, mess_int_t ind, mess_vector r)

Function to get a copy of a row scaling numbers

9.64.2.11 int(* getU) (void *data, mess_int_t ind, mess_matrix U)

Get the U factor of a decomposition

9.64.2.12 mess_int_t indx

Number of different values

9.64.2.13 size_t(* memsize) (void *data)

Function to get the size of the data

9.64.2.14 char* name

Name of the solver

9.64.2.15 mess_int_t rows

Number of rows

9.64.2.16 int(∗ solve) (void ∗data, mess_int_t ind, mess_vector b, mess_vector x)

Function to solve $A(i)x = b$

9.64.2.17 int(∗ solveh) (void ∗data, mess_int_t ind, mess_vector b, mess_vector x)

Function to solve $A(i)^H x = b$

9.64.2.18 int(∗ solvem) (void ∗data, mess_int_t ind, mess_matrix b, mess_matrix x)

Function to solve $A(i)X = B$

9.64.2.19 int(∗ solvemh) (void ∗data, mess_int_t ind, mess_matrix b, mess_matrix x)

Function to solve $A(i)^H X = B$

9.64.2.20 int(∗ solvemt) (void ∗data, mess_int_t ind, mess_matrix b, mess_matrix x)

Function to solve $A(i)^T X = B$

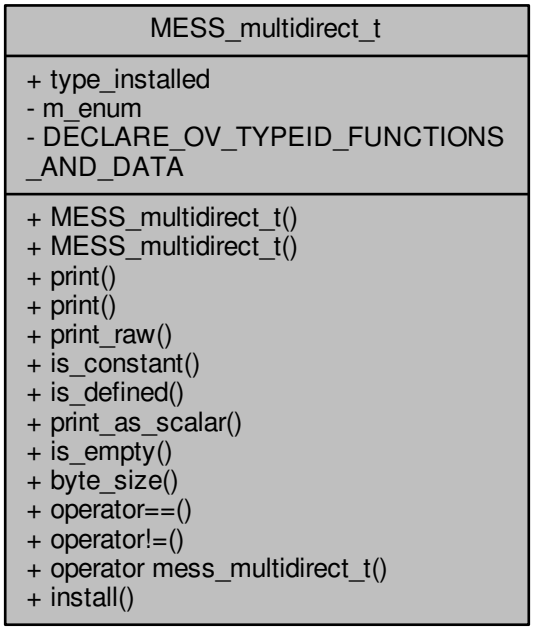
9.64.2.21 int(∗ solvet) (void ∗data, mess_int_t ind, mess_vector b, mess_vector x)

Function to solve $A(i)^T x = b$

9.65 MESS_multidirect_t Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_multidirect_t:



Public Member Functions

- MESS_multidirect_t (void)
- MESS_multidirect_t (mess_multidirect_t enu)
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void)
- bool is_empty (void) const
- size_t byte_size (void) const
- bool operator== (const MESS_multidirect_t &b) const
- bool operator!= (const MESS_multidirect_t &b) const
- operator mess_multidirect_t () const

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_multidirect_t m_enum
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.65.1 Constructor & Destructor Documentation

9.65.1.1 MESS_multidirect_t (void) [inline]

9.65.1.2 MESS_multidirect_t (mess_multidirect_t *enu*) [inline]

9.65.2 Member Function Documentation

9.65.2.1 size_t byte_size (void) const [inline]

9.65.2.2 void install (void) [static]

9.65.2.3 bool is_constant (void) const [inline]

9.65.2.4 bool is_defined (void) const [inline]

9.65.2.5 bool is_empty (void) const [inline]

9.65.2.6 operator mess_multidirect_t ( ) const [inline]

9.65.2.7 bool operator!= (const MESS_multidirect_t & *b*) const [inline]

9.65.2.8 bool operator== (const MESS_multidirect_t & *b*) const [inline]

9.65.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const [inline]

9.65.2.10 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) [inline]

9.65.2.11 bool print_as_scalar (void) [inline]

9.65.2.12 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const [inline]

9.65.3 Field Documentation

9.65.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

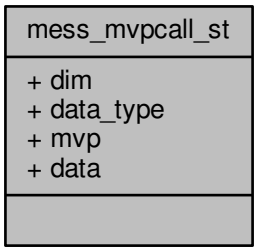
9.65.3.2 mess_multidirect_t m_enum [private]

9.65.3.3 bool type_installed = false [static]

9.66 mess_mvpcall_st Struct Reference

Represent a generic matrix-vector-product.

Collaboration diagram for mess_mvpcall_st:



Data Fields

- mess_int_t dim
- mess_datatype_t data_type
- int(∗ mvp)(void ∗data, mess_operation_t op, mess_vector x, mess_vector y)
- void ∗ data

9.66.1 Detailed Description

The mess_mvpcall object represents a generic matrix-vector product $y = Ax$ where A is not longer restricted to be a simple matrix.
This is necessary to use for example iterative solvers or iterative eigenvalue computations on more complex objects than a simple matrix.

See also

Sparse Eigenvalue Solvers
Solvers

9.66.2 Field Documentation

9.66.2.1 void∗ data

Pointer to the data needed by the matrix-vector product.

9.66.2.2 mess_datatype_t data_type

Data type of the matrix-vector product.

See also

mess_datatype_t.

9.66.2.3 mess_int_t dim

Dimension of the matrix-vector product.

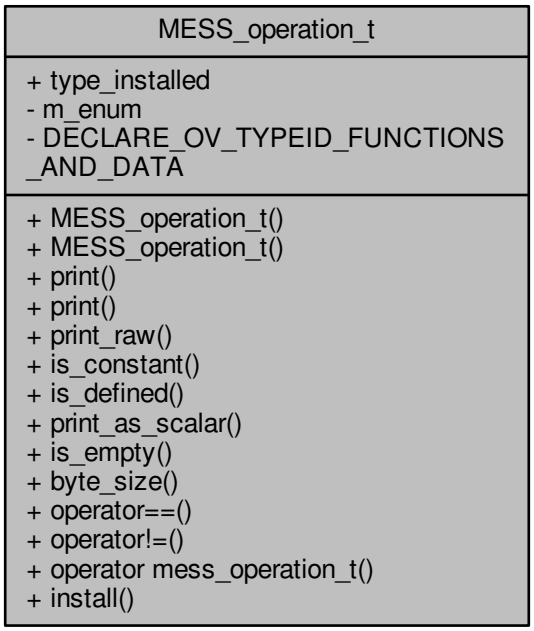
9.66.2.4 int(∗ mvp) (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)

Function pointer which points to the implementation of the matrix-vector product. It has to compute $y = Ax$.

9.67 MESS_operation_t Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_operation_t:



Public Member Functions

- MESS_operation_t (void)
- MESS_operation_t (mess_operation_t enu)
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void)
- bool is_empty (void) const
- size_t byte_size (void) const
- bool operator== (const MESS_operation_t &b) const
- bool operator!= (const MESS_operation_t &b) const
- operator mess_operation_t () const

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_operation_t m_enum
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.67.1 Constructor & Destructor Documentation

9.67.1.1 MESS_operation_t (void) [inline]

9.67.1.2 MESS_operation_t (mess_operation_t *enu*) [inline]

9.67.2 Member Function Documentation

9.67.2.1 size_t byte_size (void) const [inline]

9.67.2.2 void install (void) [static]

9.67.2.3 bool is_constant (void) const [inline]

9.67.2.4 bool is_defined (void) const [inline]

9.67.2.5 bool is_empty (void) const [inline]

9.67.2.6 operator mess_operation_t ( ) const [inline]

9.67.2.7 bool operator!= (const MESS_operation_t & *b*) const [inline]

9.67.2.8 bool operator== (const MESS_operation_t & *b*) const [inline]

9.67.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const [inline]

9.67.2.10 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) [inline]

9.67.2.11 bool print_as_scalar (void) [inline]

9.67.2.12 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const [inline]

9.67.3 Field Documentation

9.67.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

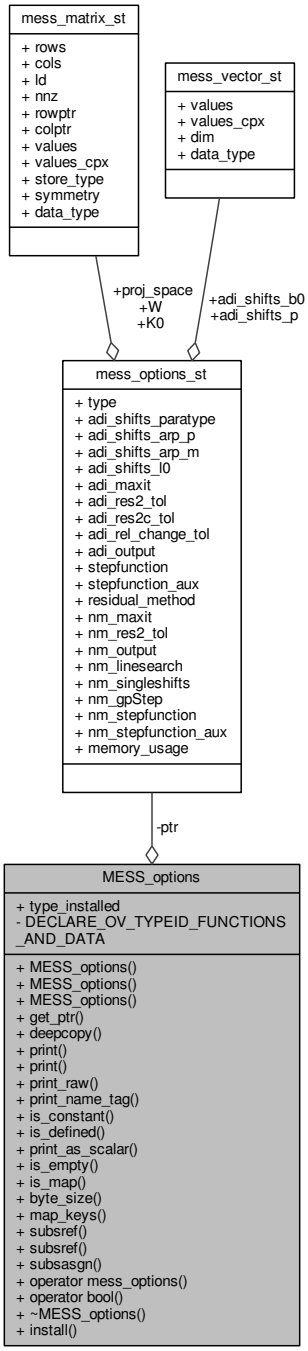
9.67.3.2 mess_operation_t m_enum [private]

9.67.3.3 bool type_installed = false [static]

9.68 MESS_options Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_options:



Public Member Functions

- MESS_options (void)
- MESS_options (mess_options opt)
- MESS_options (const MESS_options &opt)
- mess_options get_ptr (void) const
- MESS_options deepcopy (void) const

- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool print_name_tag (std::ostream &os, const std::string &name) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void) const
- bool is_empty (void) const
- bool is_map (void) const
- size_t byte_size (void) const
- string_vector map_keys (void) const
- octave_value_list subsref (const std::string &type, const std::list< octave_value_list > &idx, int k)
- octave_value subsref (const std::string &, const std::list< octave_value_list > &)
- octave_value subsasgn (const std::string &type, const std::list< octave_value_list > &idx, const octave_value &rhs)
- operator mess_options () const
- operator bool () const
- ~MESS_options (void)

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_options ptr
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.68.1 Constructor & Destructor Documentation

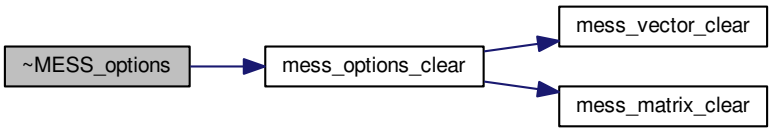
9.68.1.1 MESS_options (void)

9.68.1.2 MESS_options (mess_options *opt*)

9.68.1.3 MESS_options (const MESS_options & *opt*)

9.68.1.4 ~MESS_options (void)

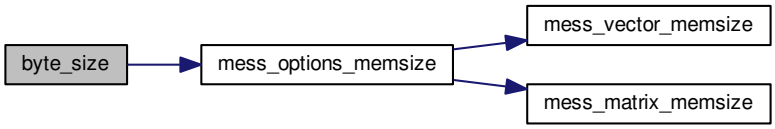
Here is the call graph for this function:



9.68.2 Member Function Documentation

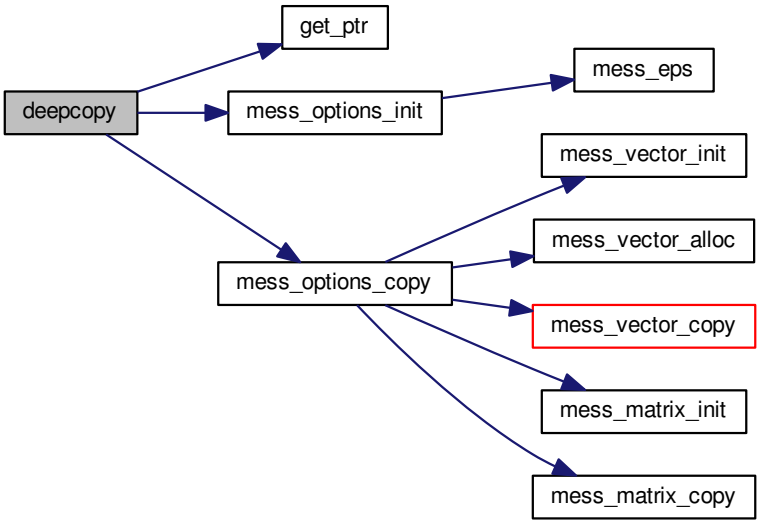
9.68.2.1 size_t byte_size (void) const

Here is the call graph for this function:



9.68.2.2 MESS_options deepcopy (void) const

Here is the call graph for this function:



9.68.2.3 mess_options get_ptr (void) const

9.68.2.4 void install (void) [static]

9.68.2.5 bool is_constant (void) const

9.68.2.6 bool is_defined (void) const

9.68.2.7 bool is_empty (void) const

9.68.2.8 bool is_map (void) const

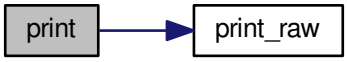
9.68.2.9 string_vector map_keys (void) const

9.68.2.10 operator bool (void) const

9.68.2.11 operator mess_options (void) const

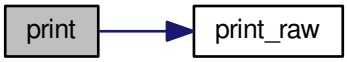
9.68.2.12 void print (std::ostream & *os*, bool *pr_as_read_syntax* = `false`) const

Here is the call graph for this function:



9.68.2.13 void print (std::ostream & *os*, bool *pr_as_read_syntax* = `false`)

Here is the call graph for this function:



9.68.2.14 bool print_as_scalar (void) const

9.68.2.15 bool print_name_tag (std::ostream & *os*, const std::string & *name*) const

9.68.2.16 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const

9.68.2.17 octave_value subsasgn (const std::string & *type*, const std::list< octave_value_list > & *idx*, const octave_value & *rhs*)

9.68.2.18 octave_value_list subsref (const std::string & *type*, const std::list< octave_value_list > & *idx*, int *k*)

9.68.2.19 octave_value subsref (const std::string & *type*, const std::list< octave_value_list > & *idx*)

Here is the call graph for this function:

9.68.3 Field Documentation

9.68.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

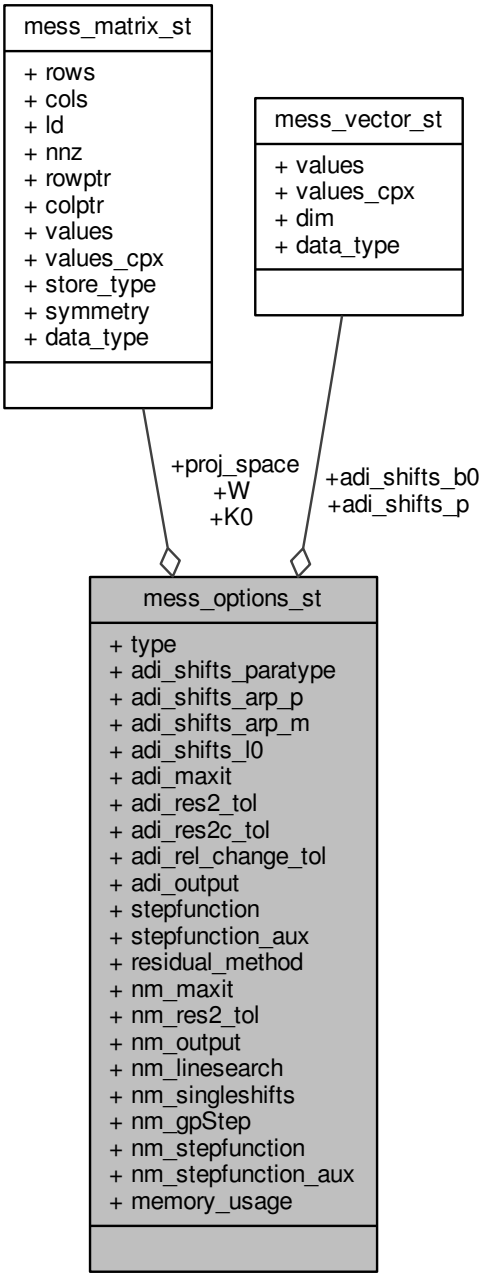
9.68.3.2 mess_options ptr [private]

9.68.3.3 bool type_installed = false [static]

9.69 mess_options_st Struct Reference

Options for the LRCF-ADI algorithm family.

Collaboration diagram for mess_options_st:



Data Fields

- mess_operation_t type
- mess_vector adi_shifts_p
- mess_parameter_t adi_shifts_paratype
- mess_int_t adi_shifts_arp_p
- mess_int_t adi_shifts_arp_m

- mess_int_t adi_shifts_l0
- mess_vector adi_shifts_b0
- mess_matrix proj_space
- mess_int_t adi_maxit
- double adi_res2_tol
- double adi_res2c_tol
- double adi_rel_change_tol
- mess_int_t adi_output
- void(∗ stepfunction)(void ∗aux, struct mess_equation_st ∗eqn, struct mess_options_st ∗opt, mess_lrcfadi↩_step ∗step)
- void ∗ stepfunction_aux
- mess_residual_t residual_method
- mess_int_t nm_maxit
- double nm_res2_tol
- mess_int_t nm_output
- mess_int_t nm_linesearch
- unsigned short nm_singleshifts
- mess_int_t nm_gpStep
- void(∗ nm_stepfunction)(void ∗aux, struct mess_equation_st ∗eqn, struct mess_equation_st ∗lyap, struct mess_options_st ∗opt, mess_lrcfadi_step ∗step)
- void ∗ nm_stepfunction_aux
- mess_memusage_t memory_usage
- mess_matrix K0
- mess_matrix W

9.69.1 Detailed Description

The mess_options_st structure is used to configure the LRCF-ADI based solvers.
It contains information about the cancellation criteria, maximum iteration numbers, the shift parameter computation and similar options. It is used by all LRCF related solvers.
The initialization is done using mess_options_init and it is freed after usage by mess_options_clear. An overview about the current setting can be printed to the standard output using mess_options_print.

See also

mess_lradi
mess_lrnm

9.69.2 Field Documentation

9.69.2.1 mess_int_t adi_maxit

Maximal number of iteration for the ADI process.

9.69.2.2 mess_int_t adi_output

True if you want some nice output during the iteration.

9.69.2.3 double adi_rel_change_tol

Stopping criteria: Frobenius-norm relative change in the ADI process.

9.69.2.4 double adi_res2_tol

relative 2-norm stopping tolerance in the ADI process.

9.69.2.5 double adi_res2c_tol

relative 2-norm change tolerance in the ADI process.

9.69.2.6 mess_int_t adi_shifts_arp_m

Number of steps in the Arnoldi process with respect to A^{-1} or $(A - UW)^{-1}$.

9.69.2.7 mess_int_t adi_shifts_arp_p

Number of steps in the Arnoldi process with respect to A or $(A - UW)$.

9.69.2.8 mess_vector adi_shifts_b0

Start vector for the Arnoldi process.

9.69.2.9 mess_int_t adi_shifts_l0

Number of shifts.

9.69.2.10 mess_vector adi_shifts_p

Precomputed shift parameters.

9.69.2.11 mess_parameter_t adi_shifts_paratype

Type of shift parameters.

9.69.2.12 mess_matrix K0

Matrix $K0$ initial feedback for LRNM.

9.69.2.13 mess_memusage_t memory_usage

control the usage of memory, 0 indicates the potential lowest memory usage and slowest performance

9.69.2.14 mess_int_t nm_gpStep

Galerkin projektion in LRCF-NM.

9.69.2.15 mess_int_t nm_linesearch

Turn line search on/off.

9.69.2.16 mess_int_t nm_maxit

Maximal number of iteration in the Newton method.

9.69.2.17 mess_int_t nm_output

Verbosity level for the LRNM.

9.69.2.18 double nm_res2_tol

Stopping criteria: 2-norm residual (Newton).

9.69.2.19 unsigned short nm_singleshifts

Set a single shift strategy in LRCF-NM.

9.69.2.20 void(∗ nm_stepfunction) (void ∗aux, struct mess_equation_st ∗eqn, struct mess_equation_st ∗lyap, struct mess_options_st ∗opt, mess_lrcfadi_step ∗step)

New iteration step function, called at the end of every full Newton step.

9.69.2.21 void∗ nm_stepfunction_aux

Auxillary data for the step function.

9.69.2.22 mess_matrix proj_space

Projection space given by the columns (only needed by MESS_LRCFADI_PARA_ADAPTIVE_V).

9.69.2.23 mess_residual_t residual_method

Set Method for Residual Computation.

9.69.2.24 void(∗ stepfunction) (void ∗aux, struct mess_equation_st ∗eqn, struct mess_options_st ∗opt, mess_lrcfadi_step ∗step)

New iteration step function, called at the end of every full ADI step.

9.69.2.25 void∗ stepfunction_aux

Auxillary data for the stepfunction.

9.69.2.26 mess_operation_t type

Type of the equation.

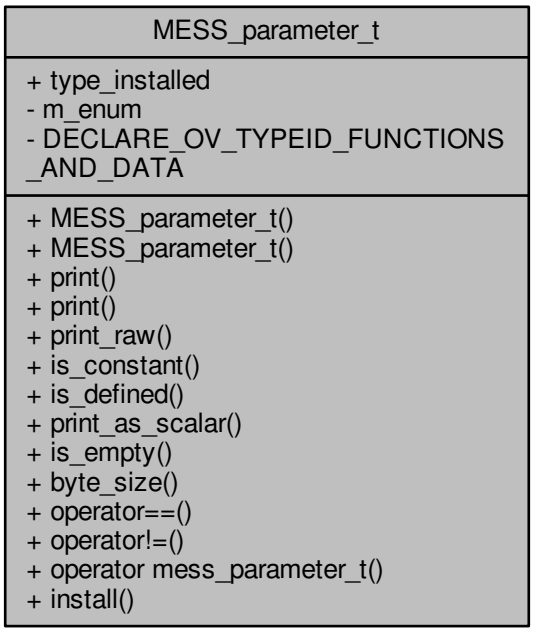
9.69.2.27 mess_matrix W

Matrix W low rank lyapunov residual from LRADI.

9.70 MESS_parameter_t Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_parameter_t:



Public Member Functions

- MESS_parameter_t (void)
- MESS_parameter_t (mess_parameter_t enu)
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void)
- bool is_empty (void) const
- size_t byte_size (void) const
- bool operator== (const MESS_parameter_t &b) const
- bool operator!= (const MESS_parameter_t &b) const
- operator mess_parameter_t () const

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_parameter_t m_enum
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.70.1 Constructor & Destructor Documentation

9.70.1.1 MESS_parameter_t (void) [inline]

9.70.1.2 MESS_parameter_t (mess_parameter_t *enu*) [inline]

9.70.2 Member Function Documentation

9.70.2.1 size_t byte_size (void) const [inline]

9.70.2.2 void install (void) [static]

9.70.2.3 bool is_constant (void) const [inline]

9.70.2.4 bool is_defined (void) const [inline]

9.70.2.5 bool is_empty (void) const [inline]

9.70.2.6 operator mess_parameter_t ( ) const [inline]

9.70.2.7 bool operator!= (const MESS_parameter_t & *b*) const [inline]

9.70.2.8 bool operator== (const MESS_parameter_t & *b*) const [inline]

9.70.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const [inline]

9.70.2.10 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) [inline]

9.70.2.11 bool print_as_scalar (void) [inline]

9.70.2.12 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const [inline]

9.70.3 Field Documentation

9.70.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

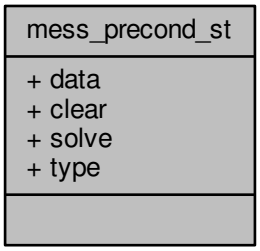
9.70.3.2 mess_parameter_t m_enum [private]

9.70.3.3 bool type_installed = false [static]

9.71 mess_precond_st Struct Reference

Representation of a preconditioner.

Collaboration diagram for mess_precond_st:



Data Fields

- void ∗ data
- int(∗ clear)(struct mess_precond_st ∗myself)
- int(∗ solve)(struct mess_precond_st ∗myself, mess_solver_options opt, mess_vector in, mess_vector out)
- int type

9.71.1 Detailed Description

The mess_precond_st structure represents a preconditioner.

9.71.2 Field Documentation

9.71.2.1 int(∗ clear) (struct mess_precond_st ∗myself)

9.71.2.2 void∗ data

Auxiliary data

9.71.2.3 int(∗ solve) (struct mess_precond_st ∗myself, mess_solver_options opt, mess_vector in, mess_vector out)

Solve $y = M^{-1}x$

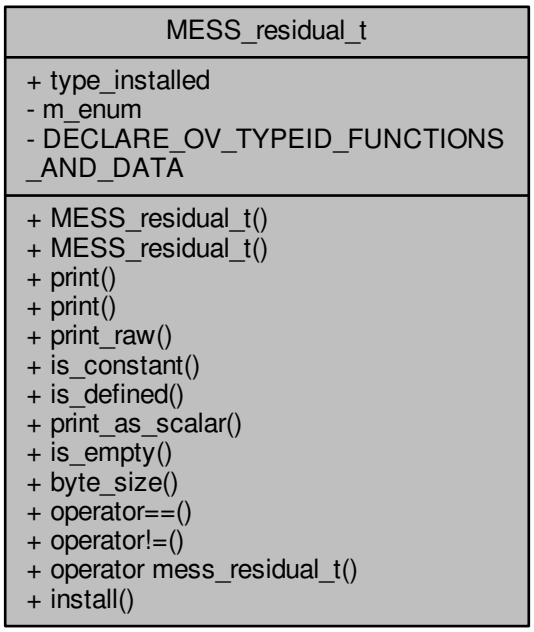
9.71.2.4 int type

Type or name of the precondtioner

9.72 MESS_residual_t Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_residual_t:



Public Member Functions

- MESS_residual_t (void)
- MESS_residual_t (mess_residual_t enu)
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void)
- bool is_empty (void) const
- size_t byte_size (void) const
- bool operator== (const MESS_residual_t &b) const
- bool operator!= (const MESS_residual_t &b) const
- operator mess_residual_t () const

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_residual_t m_enum
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.72.1 Constructor & Destructor Documentation

9.72.1.1 MESS_residual_t (void) [inline]

9.72.1.2 MESS_residual_t (mess_residual_t *enu*) [inline]

9.72.2 Member Function Documentation

9.72.2.1 size_t byte_size (void) const [inline]

9.72.2.2 void install (void) [static]

9.72.2.3 bool is_constant (void) const [inline]

9.72.2.4 bool is_defined (void) const [inline]

9.72.2.5 bool is_empty (void) const [inline]

9.72.2.6 operator mess_residual_t ( ) const [inline]

9.72.2.7 bool operator!= (const MESS_residual_t & *b*) const [inline]

9.72.2.8 bool operator== (const MESS_residual_t & *b*) const [inline]

9.72.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const [inline]

9.72.2.10 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) [inline]

9.72.2.11 bool print_as_scalar (void) [inline]

9.72.2.12 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const [inline]

9.72.3 Field Documentation

9.72.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

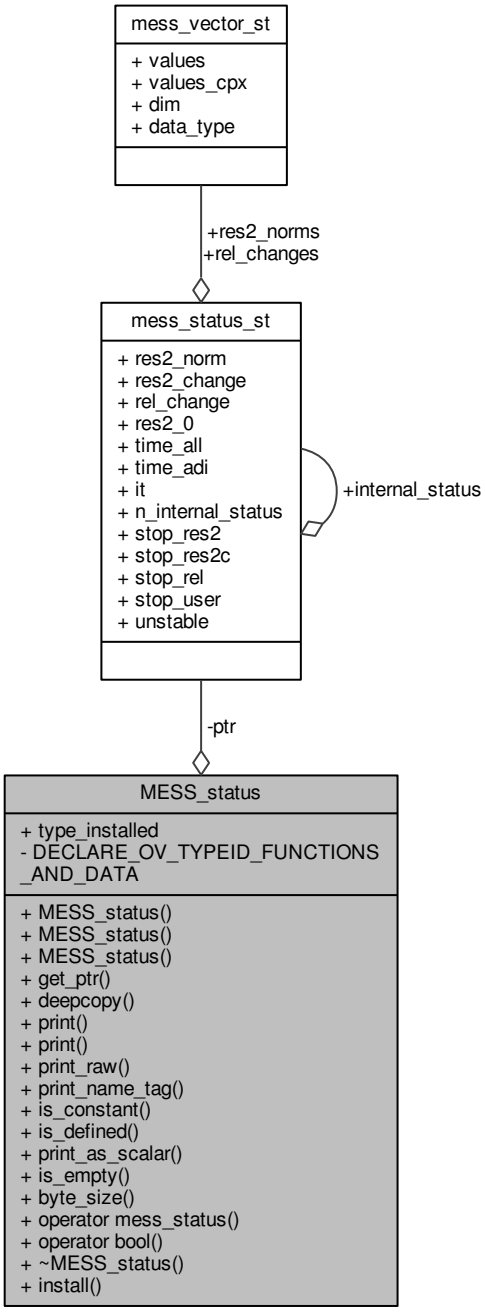
9.72.3.2 mess_residual_t m_enum [private]

9.72.3.3 bool type_installed = false [static]

9.73 MESS_status Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_status:



Public Member Functions

- MESS_status (void)

- MESS_status (mess_status s)
- MESS_status (const MESS_status &s)
- mess_status get_ptr (void) const
- MESS_status deepcopy (void) const
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool print_name_tag (std::ostream &os, const std::string &name) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void) const
- bool is_empty (void) const
- size_t byte_size (void) const
- operator mess_status () const
- operator bool () const
- ~MESS_status (void)

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_status ptr
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.73.1 Constructor & Destructor Documentation

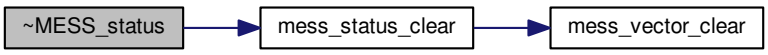
9.73.1.1 MESS_status (void)

9.73.1.2 MESS_status (mess_status *s*)

9.73.1.3 MESS_status (const MESS_status & *s*)

9.73.1.4 ~MESS_status (void)

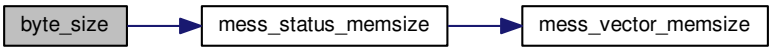
Here is the call graph for this function:



9.73.2 Member Function Documentation

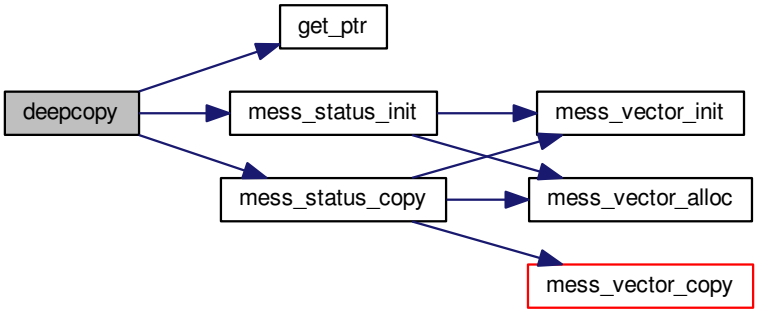
9.73.2.1 size_t byte_size (void) const

Here is the call graph for this function:



9.73.2.2 MESS_status deepcopy (void) const

Here is the call graph for this function:



9.73.2.3 mess_status get_ptr (void) const

9.73.2.4 void install (void) [static]

9.73.2.5 bool is_constant (void) const

9.73.2.6 bool is_defined (void) const

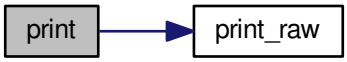
9.73.2.7 bool is_empty (void) const

9.73.2.8 operator bool (void) const

9.73.2.9 operator mess_status (void) const

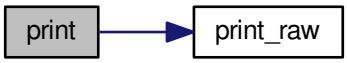
9.73.2.10 void print (std::ostream & *os*, bool *pr_as_read_syntax* = `false`) const

Here is the call graph for this function:



9.73.2.11 void print (std::ostream & *os*, bool *pr_as_read_syntax* = `false`)

Here is the call graph for this function:



9.73.2.12 bool print_as_scalar (void) const

9.73.2.13 bool print_name_tag (std::ostream & *os*, const std::string & *name*) const

9.73.2.14 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const

9.73.3 Field Documentation

9.73.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA `[private]`

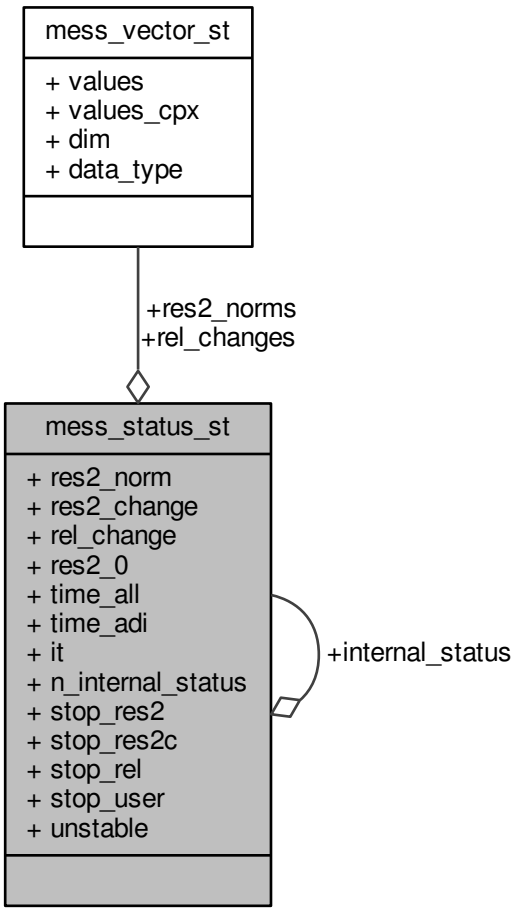
9.73.3.2 mess_status ptr `[private]`

9.73.3.3 bool type_installed = false `[static]`

9.74 mess_status_st Struct Reference

Structure/object to save the final status of a LRCF algorithm.

Collaboration diagram for mess_status_st:



Data Fields

- mess_vector res2_norms
- mess_vector rel_changes
- double res2_norm
- double res2_change
- double rel_change
- double res2_0
- double time_all
- double time_adi
- mess_int_t it
- mess_int_t n_internal_status
- struct mess_status_st ∗∗ internal_status
- unsigned short stop_res2
- unsigned short stop_res2c
- unsigned short stop_rel
- unsigned short stop_user
- unsigned short unstable

9.74.1 Field Documentation

9.74.1.1 struct mess_status_st** internal_status

Internal status at every iteration.

9.74.1.2 mess_int_t it

Number of iteration the algorithm took.

9.74.1.3 mess_int_t n_internal_status

Internal status to number of iterations.

9.74.1.4 double rel_change

Final relative change of Z.

9.74.1.5 mess_vector rel_changes

Relative change of the Iterate in every step in F-norm for lradi and in 2 norm for lrnm.

9.74.1.6 double res2_0

2-norm residual of the right hand side.

9.74.1.7 double res2_change

Final relative 2-norm change of the absolute residual.

9.74.1.8 double res2_norm

Final absoulute 2-norm residual.

9.74.1.9 mess_vector res2_norms

Absolute 2-norm in every step.

9.74.1.10 unsigned short stop_rel

Number of iteration the algorithm stops with respect to relative change in Z.

9.74.1.11 unsigned short stop_res2

Number of iteration the algorithm stops with respect to relative 2-norm residual.

9.74.1.12 unsigned short stop_res2c

Number of iteration the algorithm stops with respect to 2-norm relative change.

9.74.1.13 unsigned short stop_user

Flag to cancel the iteration.

9.74.1.14 double time_adi

Time for ADI.

9.74.1.15 double time_all

Overall time.

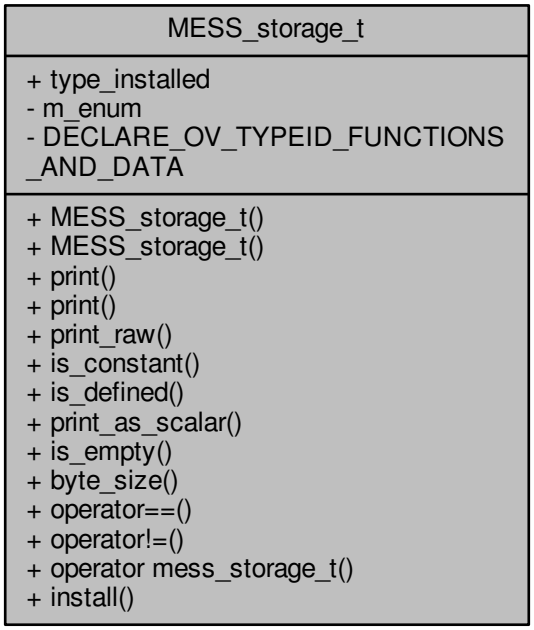
9.74.1.16 unsigned short unstable

True if there occur unstable Ritz values.

9.75 MESS_storage_t Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_storage_t:



Public Member Functions

- MESS_storage_t (void)
- MESS_storage_t (mess_storage_t enu)
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void)
- bool is_empty (void) const
- size_t byte_size (void) const
- bool operator== (const MESS_storage_t &b) const
- bool operator!= (const MESS_storage_t &b) const
- operator mess_storage_t () const

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_storage_t m_enum
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

9.75.1 Constructor & Destructor Documentation

9.75.1.1 MESS_storage_t (void) [inline]

9.75.1.2 MESS_storage_t (mess_storage_t *enu*) [inline]

9.75.2 Member Function Documentation

9.75.2.1 size_t byte_size (void) const [inline]

9.75.2.2 void install (void) [static]

9.75.2.3 bool is_constant (void) const [inline]

9.75.2.4 bool is_defined (void) const [inline]

9.75.2.5 bool is_empty (void) const [inline]

9.75.2.6 operator mess_storage_t ( ) const [inline]

9.75.2.7 bool operator!= (const MESS_storage_t & *b*) const [inline]

9.75.2.8 bool operator== (const MESS_storage_t & *b*) const [inline]

9.75.2.9 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) const [inline]

9.75.2.10 void print (std::ostream & *os*, bool *pr_as_read_syntax* = false) [inline]

9.75.2.11 bool print_as_scalar (void) [inline]

9.75.2.12 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const [inline]

9.75.3 Field Documentation

9.75.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA [private]

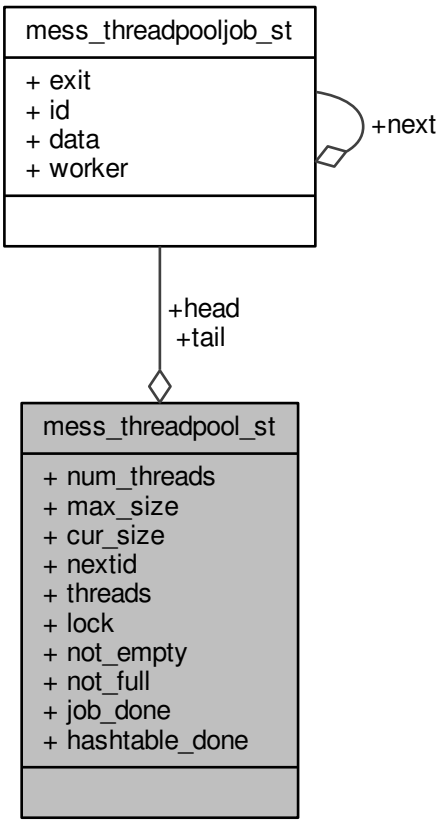
9.75.3.2 mess_storage_t m_enum [private]

9.75.3.3 bool type_installed = false [static]

9.76 mess_threadpool_st Struct Reference

Representation of a thread pool.

Collaboration diagram for mess_threadpool_st:



Data Fields

- int num_threads
- int max_size
- int cur_size
- mess_int_t nextid
- pthread_t * threads
- mess_threadpooljob head
- mess_threadpooljob tail
- pthread_mutex_t lock
- pthread_cond_t not_empty
- pthread_cond_t not_full
- pthread_cond_t job_done
- void * hashtable_done

9.76.1 Detailed Description

The mess_threadpool_st structure contains all information to represent a threadpool.

9.76.2 Field Documentation

9.76.2.1 int cur_size

Number of current elements in the queue holding the indices to load

9.76.2.2 void* hashtable_done

Pointer to a hashtable including done jobs.

9.76.2.3 mess_threadpooljob head

9.76.2.4 pthread_cond_t job_done

Condition variable if a thread is not empty, not full or the job is done

9.76.2.5 pthread_mutex_t lock

Mutex ensuring exclusive access to buffer

9.76.2.6 int max_size

Maximum size of the queue holding indices to load

9.76.2.7 mess_int_t nextid

Identification number of the next job

9.76.2.8 pthread_cond_t not_empty

9.76.2.9 pthread_cond_t not_full

9.76.2.10 int num_threads

Number of created threads

9.76.2.11 mess_threadpooljob tail

First and last job

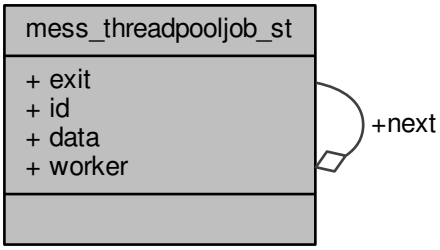
9.76.2.12 pthread_t* threads

Pointer to thread

9.77 mess_threadpooljob_st Struct Reference

Representation of a thread pool job.

Collaboration diagram for mess_threadpooljob_st:



Data Fields

- int exit
- mess_int_t id
- void * data
- void(* worker)(void *data)
- struct mess_threadpooljob_st * next

9.77.1 Detailed Description

The mess_threadpooljob_st structure contains all information to represent a job for the threadpool.

9.77.2 Field Documentation

9.77.2.1 void∗ data

Pointer to data of the job

9.77.2.2 int exit

Value to exit a job if it is done (-1)

9.77.2.3 mess_int_t id

Identification number of the job

9.77.2.4 struct mess_threadpooljob_st∗ next

Pointer to next job

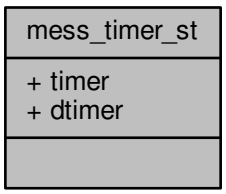
9.77.2.5 void(∗ worker) (void ∗data)

Pointer to the worker function.

9.78 mess_timer_st Struct Reference

Representation of a timer.

Collaboration diagram for mess_timer_st:



Data Fields

- struct timespec ∗ timer
- double dtimer

9.78.1 Detailed Description

The mess_timer_st structure represents a timer.

9.78.2 Field Documentation

9.78.2.1 double dtimer

Determined time

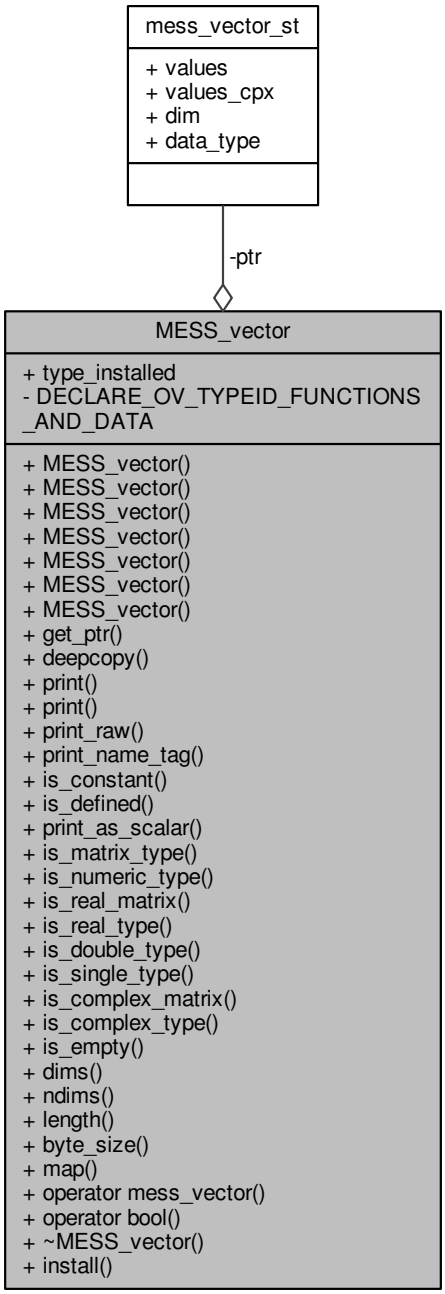
9.78.2.2 struct timespec* timer

Pointer to timer object

9.79 MESS_vector Class Reference

Inherits octave_base_value.

Collaboration diagram for MESS_vector:



Public Member Functions

- MESS_vector (void)
- MESS_vector (mess_vector v)
- MESS_vector (Matrix m)
- MESS_vector (ComplexMatrix m)
- MESS_vector (double m)

- MESS_vector (Complex m)
- MESS_vector (const MESS_vector &v)
- mess_vector get_ptr (void) const
- MESS_vector deepcopy (void) const
- void print (std::ostream &os, bool pr_as_read_syntax=false) const
- void print (std::ostream &os, bool pr_as_read_syntax=false)
- void print_raw (std::ostream &os, bool pr_as_read_syntax) const
- bool print_name_tag (std::ostream &os, const std::string &name) const
- bool is_constant (void) const
- bool is_defined (void) const
- bool print_as_scalar (void) const
- bool is_matrix_type (void) const
- bool is_numeric_type (void) const
- bool is_real_matrix (void) const
- bool is_real_type (void) const
- bool is_double_type (void) const
- bool is_single_type (void) const
- bool is_complex_matrix (void) const
- bool is_complex_type (void) const
- bool is_empty (void) const
- dim_vector dims (void) const
- int ndims (void) const
- size_t length (void) const
- size_t byte_size (void) const
- octave_value map (unary_mapper_t umap) const
- operator mess_vector () const
- operator bool () const
- ~MESS_vector (void)

Static Public Member Functions

- static void install (void)

Static Public Attributes

- static bool type_installed = false

Private Attributes

- mess_vector ptr
- DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA

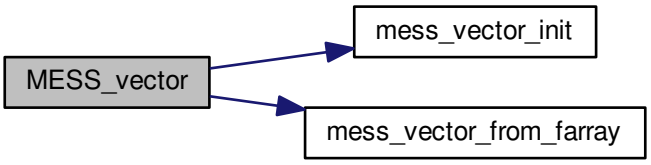
9.79.1 Constructor & Destructor Documentation

9.79.1.1 MESS_vector (void)

9.79.1.2 MESS_vector (mess_vector *v*)

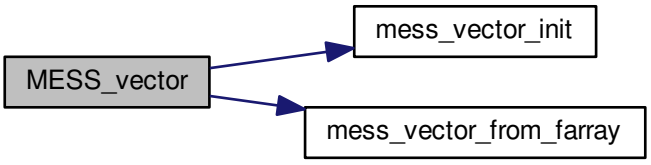
9.79.1.3 MESS_vector (Matrix *m*)

Here is the call graph for this function:



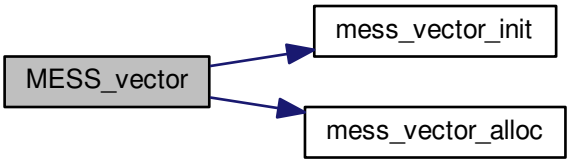
9.79.1.4 MESS_vector (ComplexMatrix *m*)

Here is the call graph for this function:



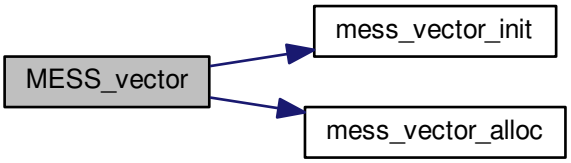
9.79.1.5 MESS_vector (double *m*)

Here is the call graph for this function:



9.79.1.6 MESS_vector (Complex *m*)

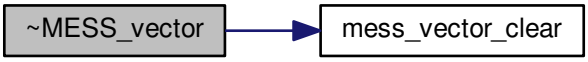
Here is the call graph for this function:



9.79.1.7 MESS_vector (const MESS_vector & *v*)

9.79.1.8 ~MESS_vector (void)

Here is the call graph for this function:



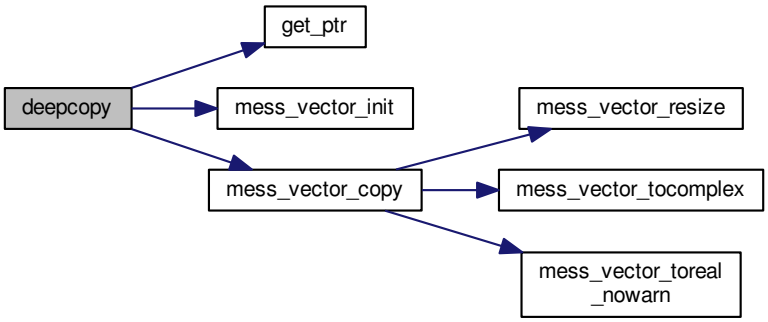
9.79.2 Member Function Documentation

9.79.2.1 size_t byte_size (void) const

Here is the call graph for this function:

9.79.2.2 MESS_vector deepcopy (void) const

Here is the call graph for this function:

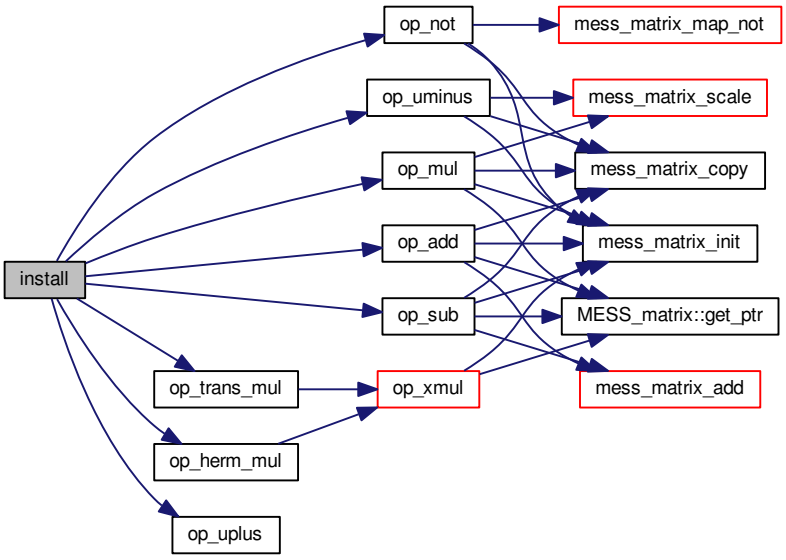


9.79.2.3 dim_vector dims (void) const

9.79.2.4 mess_vector get_ptr (void) const

9.79.2.5 void install (void) [static]

Here is the call graph for this function:



9.79.2.6 bool is_complex_matrix (void) const

9.79.2.7 bool is_complex_type (void) const

9.79.2.8 bool is_constant (void) const

9.79.2.9 bool is_defined (void) const

9.79.2.10 bool is_double_type (void) const

9.79.2.11 bool is_empty (void) const

9.79.2.12 bool is_matrix_type (void) const

9.79.2.13 bool is_numeric_type (void) const

9.79.2.14 bool is_real_matrix (void) const

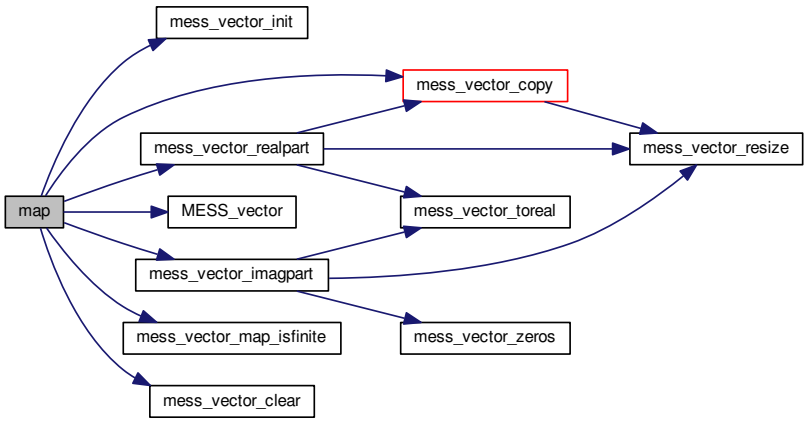
9.79.2.15 bool is_real_type (void) const

9.79.2.16 bool is_single_type (void) const

9.79.2.17 size_t length (void) const

9.79.2.18 octave_value map (unary_mapper_t *umap*) const

Here is the call graph for this function:



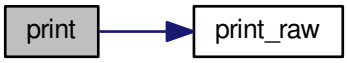
9.79.2.19 int ndims (void) const

9.79.2.20 operator bool (void) const

9.79.2.21 operator mess_vector (void) const

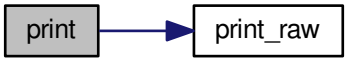
9.79.2.22 void print (std::ostream & *os*, bool *pr_as_read_syntax* = `false`) const

Here is the call graph for this function:



9.79.2.23 void print (std::ostream & *os*, bool *pr_as_read_syntax* = `false`)

Here is the call graph for this function:



9.79.2.24 bool print_as_scalar (void) const

9.79.2.25 bool print_name_tag (std::ostream & *os*, const std::string & *name*) const

9.79.2.26 void print_raw (std::ostream & *os*, bool *pr_as_read_syntax*) const

9.79.3 Field Documentation

9.79.3.1 DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA `[private]`

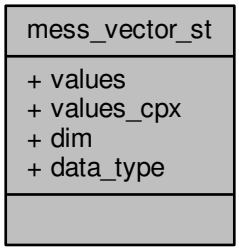
9.79.3.2 mess_vector ptr `[private]`

9.79.3.3 bool type_installed = false `[static]`

9.80 mess_vector_st Struct Reference

Representation of a real or complex vector.

Collaboration diagram for mess_vector_st:



Data Fields

- double * values
- mess_double_cpx_t * values_cpx
- mess_int_t dim
- mess_datatype_t data_type

9.80.1 Detailed Description

The mess_vector_st structure holds all information about a vector.

9.80.2 Field Documentation

9.80.2.1 mess_datatype_t data_type

Datatype of the values in a vector

9.80.2.2 mess_int_t dim

Dimension of a vector

9.80.2.3 double* values

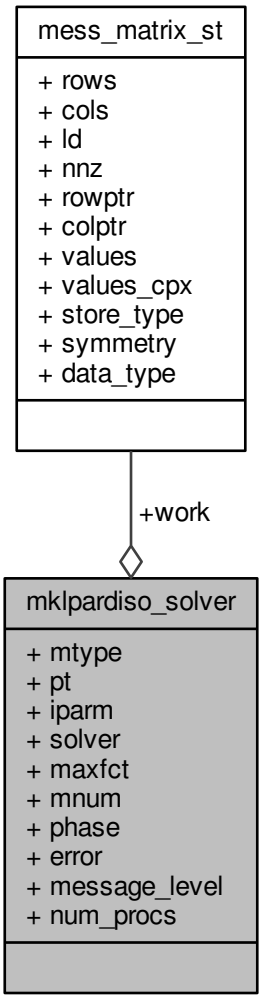
Real values of a vector

9.80.2.4 mess_double_cpx_t* values_cpx

Complex values of a vector

9.81 mklpardiso_solver Struct Reference

Collaboration diagram for mklpardiso_solver:



Data Fields

- mess_matrix work
- mess_int_t mtype
- void * pt [64]
- mess_int_t iparm [64]
- mess_int_t solver
- mess_int_t maxfct
- mess_int_t mnum
- mess_int_t phase
- mess_int_t error
- mess_int_t message_level
- mess_int_t num_procs

9.81.1 Field Documentation

9.81.1.1 mess_int_t error

9.81.1.2 mess_int_t iparm[64]

9.81.1.3 mess_int_t maxfct

9.81.1.4 mess_int_t message_level

9.81.1.5 mess_int_t mnum

9.81.1.6 mess_int_t mtype

9.81.1.7 mess_int_t num_procs

9.81.1.8 mess_int_t phase

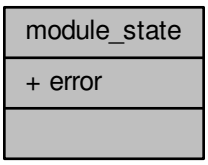
9.81.1.9 void∗ pt[64]

9.81.1.10 mess_int_t solver

9.81.1.11 mess_matrix work

9.82 module_state Struct Reference

Collaboration diagram for module_state:



Data Fields

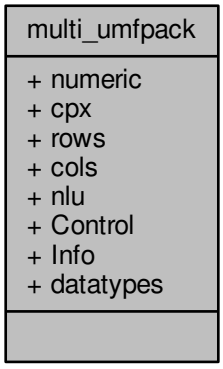
- PyObject ∗ error

9.82.1 Field Documentation

9.82.1.1 PyObject∗ error

9.83 multi_umfpack Struct Reference

Collaboration diagram for multi_umfpack:



Data Fields

- void ∗∗ numeric
- unsigned short cpx
- mess_int_t rows
- mess_int_t cols
- mess_int_t nlu
- double ∗∗ Control
- double ∗∗ Info
- unsigned short ∗ datatypes

9.83.1 Field Documentation

9.83.1.1 mess_int_t cols

9.83.1.2 double∗∗ Control

9.83.1.3 unsigned short cpx

9.83.1.4 unsigned short∗ datatypes

9.83.1.5 double∗∗ Info

9.83.1.6 mess_int_t nlu

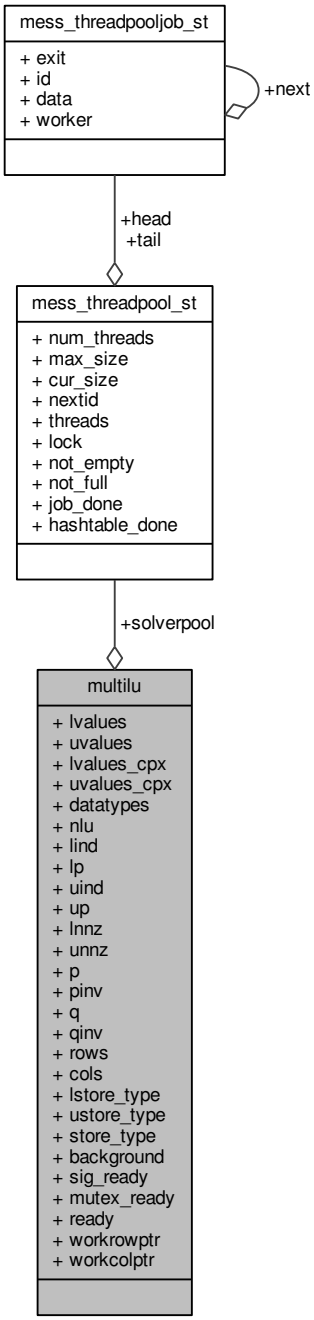
9.83.1.7 void** numeric

9.83.1.8 mess_int_t rows

9.84 multilu Struct Reference

Internal structure to store a multilu.

Collaboration diagram for multilu:



Data Fields

- double ** lvalues
- double ** uvalues
- mess_double_cpx_t ** lvalues_cpx
- mess_double_cpx_t ** uvalues_cpx
- unsigned short * datatypes
- mess_int_t nlu
- mess_int_t * lind
- mess_int_t * lp
- mess_int_t * uind
- mess_int_t * up
- mess_int_t lnnz
- mess_int_t unnz
- mess_int_t * p
- mess_int_t * pinv
- mess_int_t * q
- mess_int_t * qinv
- mess_int_t rows
- mess_int_t cols
- unsigned short lstore_type
- unsigned short ustore_type
- unsigned short store_type
- unsigned short background
- pthread_cond_t * sig_ready
- pthread_mutex_t * mutex_ready
- mess_threadpool solverpool
- mess_int_t * ready
- mess_int_t * workrowptr
- mess_int_t * workcolptr

9.84.1 Field Documentation

9.84.1.1 unsigned short background

9.84.1.2 mess_int_t cols

9.84.1.3 unsigned short* datatypes

9.84.1.4 mess_int_t* lind

9.84.1.5 mess_int_t lnnz

9.84.1.6 mess_int_t* lp

9.84.1.7 unsigned short lstore_type

9.84.1.8 double** lvalues

9.84.1.9 mess_double_cpx_t** lvalues_cpx

9.84.1.10 pthread_mutex_t* mutex_ready

9.84.1.11 mess_int_t nlu

9.84.1.12 mess_int_t* p

9.84.1.13 mess_int_t* pinv

9.84.1.14 mess_int_t* q

9.84.1.15 mess_int_t* qinv

9.84.1.16 mess_int_t* ready

9.84.1.17 mess_int_t rows

9.84.1.18 pthread_cond_t* sig_ready

9.84.1.19 mess_threadpool solverpool

9.84.1.20 unsigned short store_type

9.84.1.21 mess_int_t* uind

9.84.1.22 mess_int_t unnz

9.84.1.23 mess_int_t* up

9.84.1.24 unsigned short ustore_type

9.84.1.25 double** uvalues

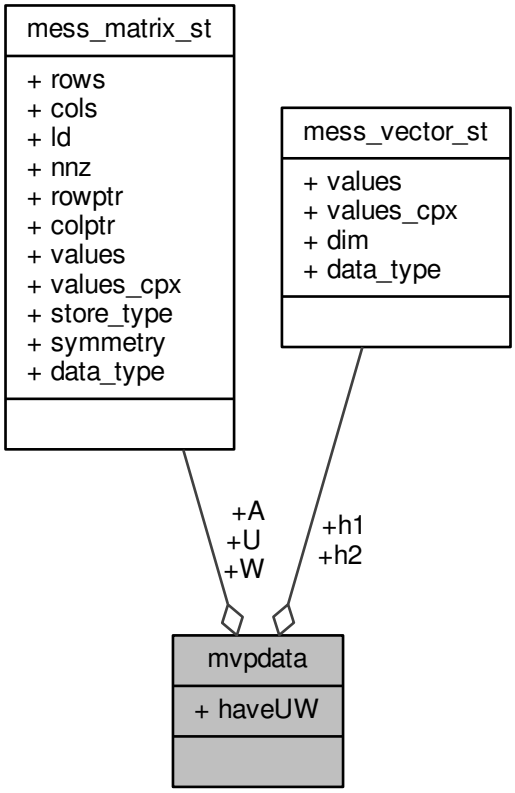
9.84.1.26 mess_double_cpx_t** uvalues_cpx

9.84.1.27 mess_int_t* workcolptr

9.84.1.28 mess_int_t* workrowptr

9.85 mvpdata Struct Reference

Collaboration diagram for mvpdata:



Data Fields

- mess_matrix A
- mess_matrix U
- mess_matrix W
- mess_vector h1
- mess_vector h2
- int haveUW

9.85.1 Field Documentation

9.85.1.1 mess_matrix A

9.85.1.2 mess_vector h1

9.85.1.3 mess_vector h2

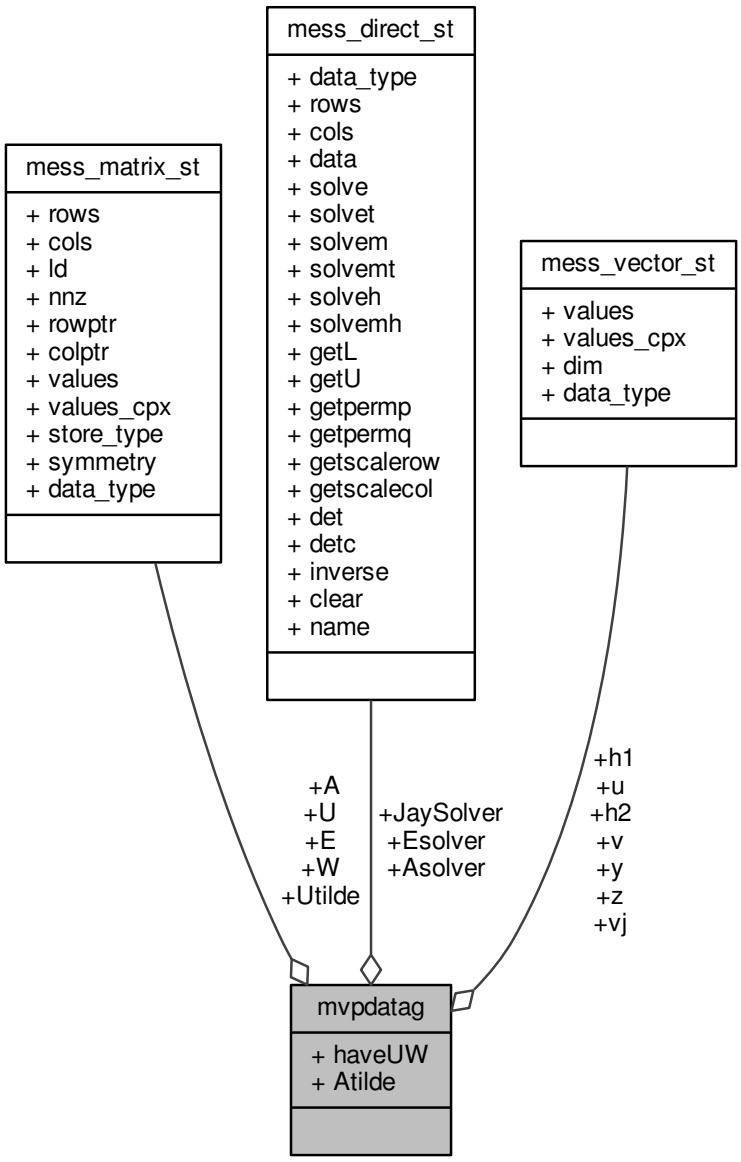
9.85.1.4 int haveUW

9.85.1.5 mess_matrix U

9.85.1.6 mess_matrix W

9.86 mvpdatag Struct Reference

Collaboration diagram for mvpdatag:



Data Fields

- mess_matrix A
- mess_matrix E
- mess_matrix U
- mess_matrix W
- mess_vector h1
- mess_vector h2
- mess_vector y
- mess_direct Esolver
- mess_direct Asolver
- mess_direct JaySolver
- mess_matrix Utilde
- mess_vector z
- mess_vector u
- mess_vector v
- mess_vector vj
- int haveUW
- int Atilde

9.86.1 Field Documentation

9.86.1.1 mess_matrix A

9.86.1.2 mess_direct Asolver

9.86.1.3 int Atilde

9.86.1.4 mess_matrix E

9.86.1.5 mess_direct Esolver

9.86.1.6 mess_vector h1

9.86.1.7 mess_vector h2

9.86.1.8 int haveUW

9.86.1.9 mess_direct JaySolver

9.86.1.10 mess_matrix U

9.86.1.11 mess_vector u

9.86.1.12 mess_matrix Utilde

9.86.1.13 mess_vector v

9.86.1.14 mess_vector vj

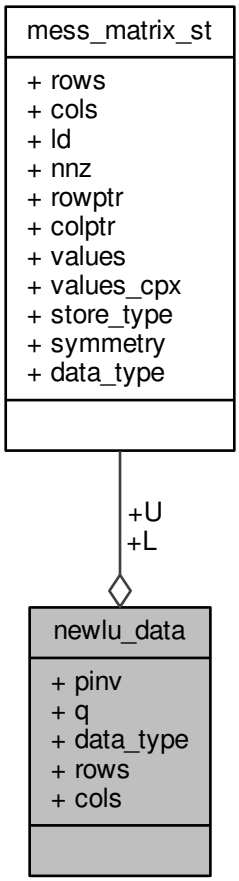
9.86.1.15 mess_matrix W

9.86.1.16 mess_vector y

9.86.1.17 mess_vector z

9.87 newlu_data Struct Reference

Collaboration diagram for newlu_data:



Data Fields

- mess_matrix L
- mess_matrix U
- mess_int_t * pinv
- mess_int_t * q
- mess_datatype_t data_type
- mess_int_t rows
- mess_int_t cols

9.87.1 Field Documentation

9.87.1.1 mess_int_t cols

9.87.1.2 mess_datatype_t data_type

9.87.1.3 mess_matrix L

9.87.1.4 mess_int_t* pinv

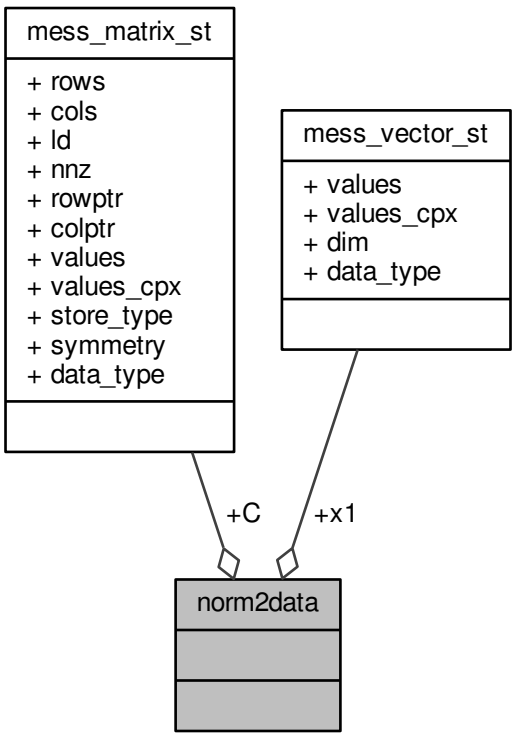
9.87.1.5 mess_int_t* q

9.87.1.6 mess_int_t rows

9.87.1.7 mess_matrix U

9.88 norm2data Struct Reference

Collaboration diagram for norm2data:



Data Fields

- mess_matrix C
- mess_vector x1

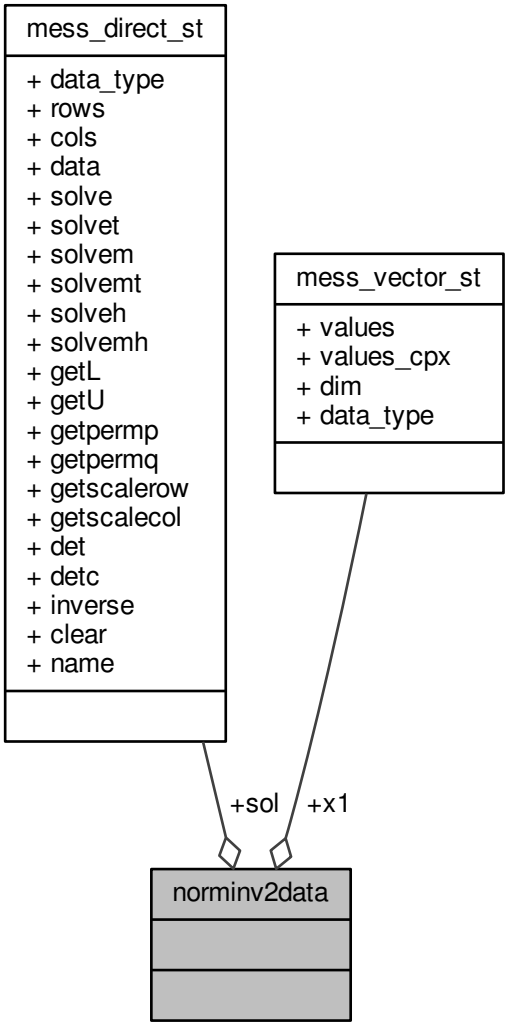
9.88.1 Field Documentation

9.88.1.1 mess_matrix C

9.88.1.2 mess_vector x1

9.89 norminv2data Struct Reference

Collaboration diagram for norminv2data:



Data Fields

- mess_vector x1
- mess_direct sol

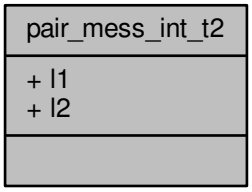
9.89.1 Field Documentation

9.89.1.1 mess_direct sol

9.89.1.2 mess_vector x1

9.90 pair_mess_int_t2 Struct Reference

Collaboration diagram for pair_mess_int_t2:



Data Fields

- mess_int_t l1
- mess_int_t l2

9.90.1 Detailed Description

structure to sort pairs (mess_int_t, mess_int_t)

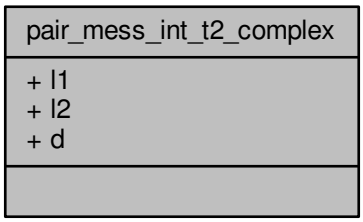
9.90.2 Field Documentation

9.90.2.1 mess_int_t l1

9.90.2.2 mess_int_t l2

9.91 pair_mess_int_t2_complex Struct Reference

Collaboration diagram for pair_mess_int_t2_complex:



Data Fields

- mess_int_t l1
- mess_int_t l2
- mess_double_cpx_t d

9.91.1 Detailed Description

structure to sort triples (mess_int_t, mess_int_t,`complex`)

9.91.2 Field Documentation

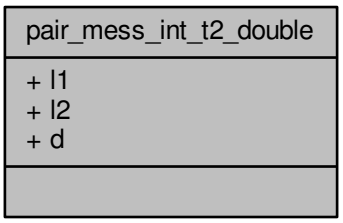
9.91.2.1 mess_double_cpx_t d

9.91.2.2 mess_int_t l1

9.91.2.3 mess_int_t l2

9.92 pair_mess_int_t2_double Struct Reference

Collaboration diagram for pair_mess_int_t2_double:



Data Fields

- mess_int_t l1
- mess_int_t l2
- double d

9.92.1 Detailed Description

structure to sort triples (mess_int_t, mess_int_t,`double`)

9.92.2 Field Documentation

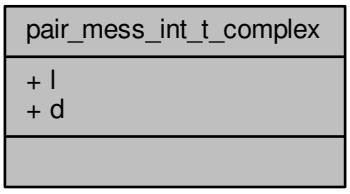
9.92.2.1 double d

9.92.2.2 mess_int_t l1

9.92.2.3 mess_int_t l2

9.93 pair_mess_int_t_complex Struct Reference

Collaboration diagram for pair_mess_int_t_complex:



Data Fields

- mess_int_t l
- mess_double_cpx_t d

9.93.1 Detailed Description

structure to sort pairs (mess_int_t, complex)

structure to sort pairs (mess_int_t,`double complex`)

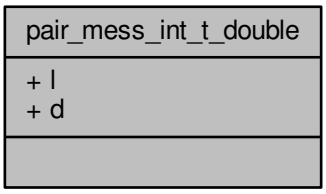
9.93.2 Field Documentation

9.93.2.1 mess_double_cpx_t d

9.93.2.2 mess_int_t l

9.94 pair_mess_int_t_double Struct Reference

Collaboration diagram for pair_mess_int_t_double:



Data Fields

- mess_int_t l
- double d

9.94.1 Detailed Description

structure to sort pairs (mess_int_t, double)

structure to sort pairs (mess_int_t,`double`)

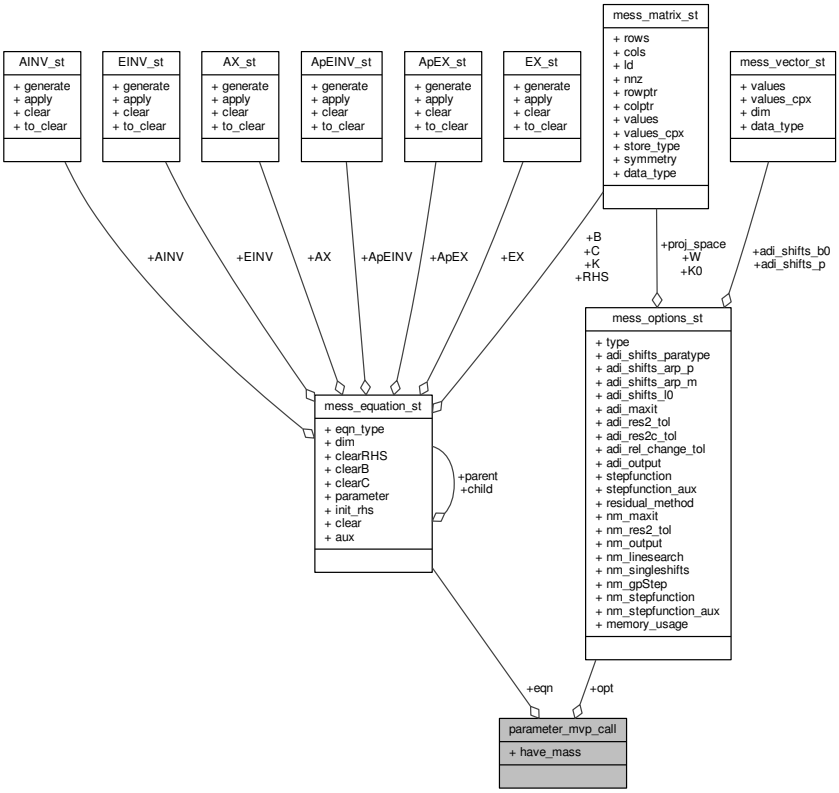
9.94.2 Field Documentation

9.94.2.1 double d

9.94.2.2 mess_int_t l

9.95 parameter_mvp_call Struct Reference

Collaboration diagram for parameter_mvp_call:



Data Fields

- mess_equation eqn
- mess_options opt
- int have_mass

9.95.1 Field Documentation

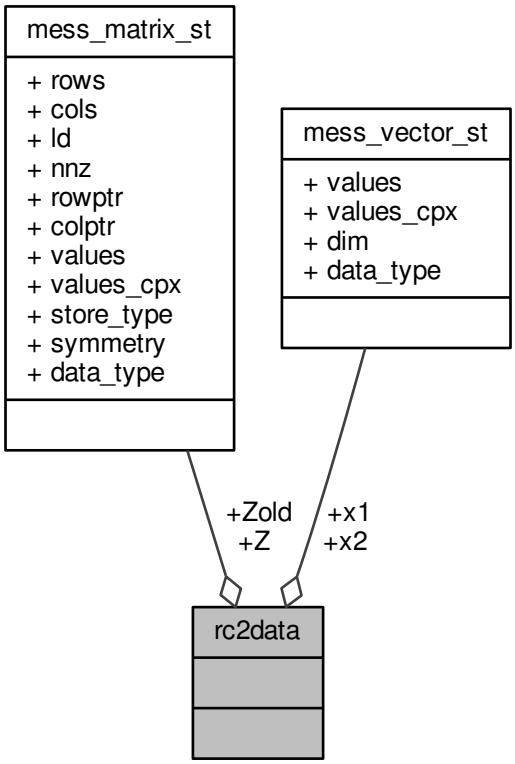
9.95.1.1 mess_equation eqn

9.95.1.2 int have_mass

9.95.1.3 mess_options opt

9.96 rc2data Struct Reference

Collaboration diagram for rc2data:



Data Fields

- mess_matrix Z
- mess_matrix Zold
- mess_vector x1
- mess_vector x2

9.96.1 Field Documentation

9.96.1.1 mess_vector x1

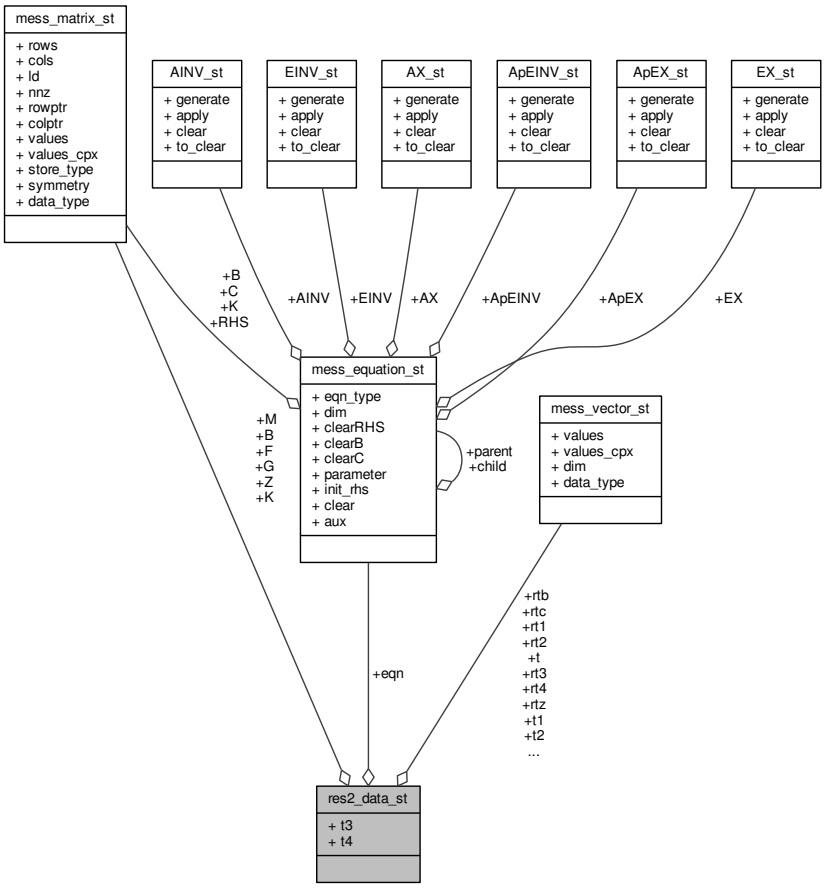
9.96.1.2 mess_vector x2

9.96.1.3 mess_matrix Z

9.96.1.4 mess_matrix Zold

9.97 res2_data_st Struct Reference

Collaboration diagram for res2_data_st:



Data Fields

- mess_matrix F
- mess_matrix G
- mess_matrix Z
- mess_matrix K
- mess_matrix B
- mess_matrix M
- mess_vector t
- mess_vector t1
- mess_vector t2
- mess_vector t3
- mess_vector t4
- mess_equation eqn
- mess_vector rtc
- mess_vector rtz
- mess_vector rt1
- mess_vector rt2
- mess_vector rtb
- mess_vector rt3
- mess_vector rt4

9.97.1 Field Documentation

9.97.1.1 mess_matrix B

9.97.1.2 mess_equation eqn

9.97.1.3 mess_matrix F

9.97.1.4 mess_matrix G

9.97.1.5 mess_matrix K

9.97.1.6 mess_matrix M

9.97.1.7 mess_vector rt1

9.97.1.8 mess_vector rt2

9.97.1.9 mess_vector rt3

9.97.1.10 mess_vector rt4

9.97.1.11 mess_vector rtb

9.97.1.12 mess_vector rtc

9.97.1.13 mess_vector rtz

9.97.1.14 mess_vector t

9.97.1.15 mess_vector t1

9.97.1.16 mess_vector t2

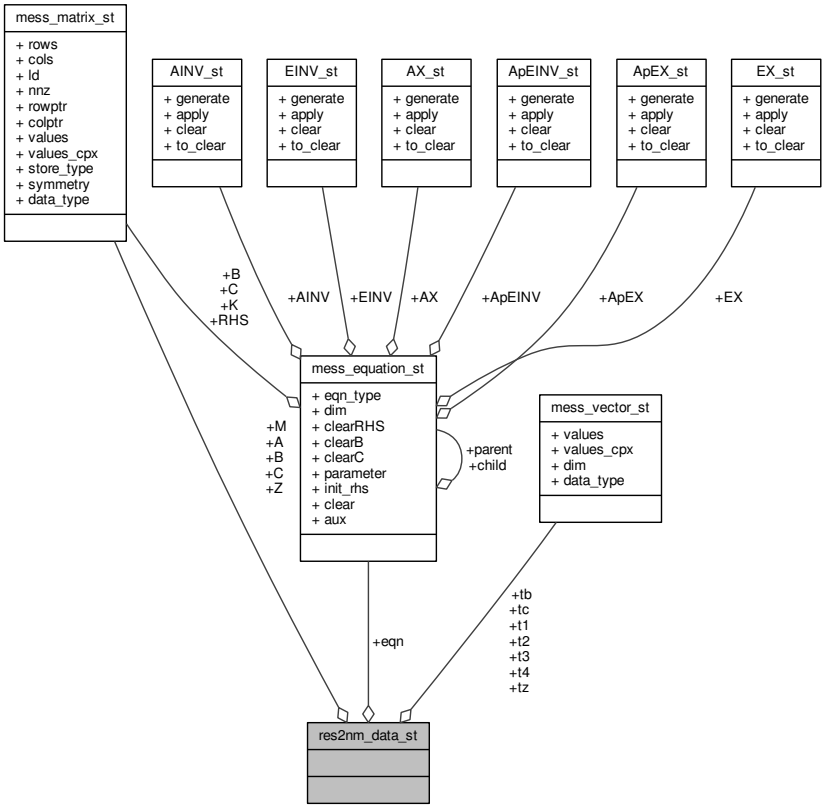
9.97.1.17 mess_vector t3

9.97.1.18 mess_vector t4

9.97.1.19 mess_matrix Z

9.98 res2nm_data_st Struct Reference

Collaboration diagram for res2nm_data_st:



Data Fields

- mess_matrix A
- mess_matrix B
- mess_matrix C
- mess_matrix Z
- mess_matrix M
- mess_vector tc
- mess_vector tz
- mess_vector t1
- mess_vector t2
- mess_vector tb
- mess_vector t3
- mess_vector t4
- mess_equation eqn

9.98.1 Field Documentation

9.98.1.1 mess_matrix A

9.98.1.2 mess_matrix B

9.98.1.3 mess_matrix C

9.98.1.4 mess_equation eqn

9.98.1.5 mess_matrix M

9.98.1.6 mess_vector t1

9.98.1.7 mess_vector t2

9.98.1.8 mess_vector t3

9.98.1.9 mess_vector t4

9.98.1.10 mess_vector tb

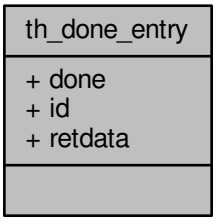
9.98.1.11 mess_vector tc

9.98.1.12 mess_vector tz

9.98.1.13 mess_matrix Z

9.99 th_done_entry Struct Reference

Collaboration diagram for th_done_entry:



Data Fields

- int done
- char * id
- void * retdata

9.99.1 Field Documentation

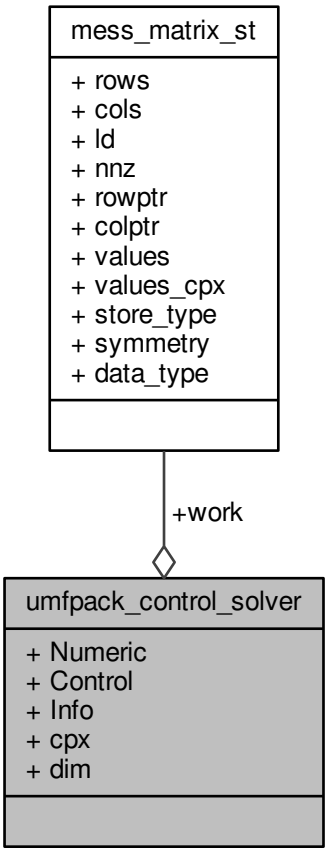
9.99.1.1 int done

9.99.1.2 char∗ id

9.99.1.3 void∗ retdata

9.100 umfpack_control_solver Struct Reference

Collaboration diagram for umfpack_control_solver:



Data Fields

- void ∗ Numeric
- double Control [UMFPACK_CONTROL]
- double Info [UMFPACK_INFO]
- unsigned short cpx
- mess_matrix work
- mess_int_t dim

9.100.1 Field Documentation

9.100.1.1 double Control[UMFPACK_CONTROL]

9.100.1.2 unsigned short cpx

9.100.1.3 mess_int_t dim

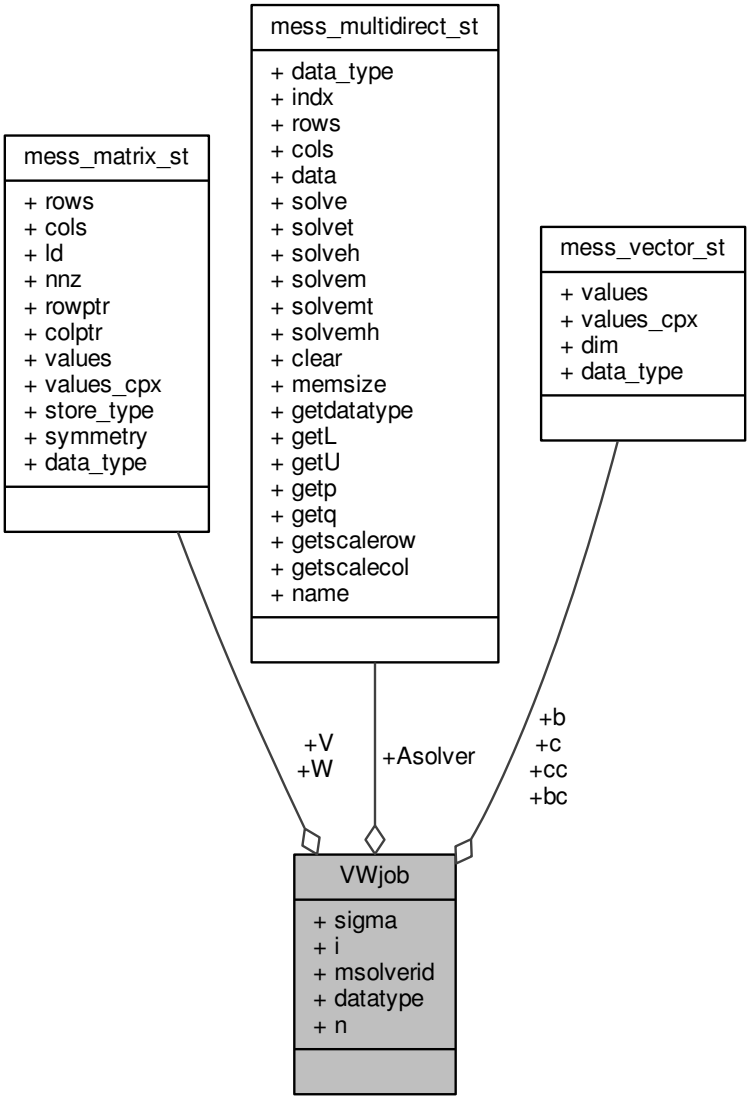
9.100.1.4 double Info[UMFPACK_INFO]

9.100.1.5 void* Numeric

9.100.1.6 mess_matrix work

9.101 VWjob Struct Reference

Collaboration diagram for VWjob:



Data Fields

- mess_double_cpx_t sigma
- mess_int_t i
- mess_multidirect Asolver
- mess_vector b
- mess_vector bc
- mess_vector c
- mess_vector cc
- mess_matrix V
- mess_matrix W
- mess_int_t msolverid
- mess_datatype_t datatype
- mess_int_t n

9.101.1 Field Documentation

9.101.1.1 mess_multidirect Asolver

9.101.1.2 mess_vector b

9.101.1.3 mess_vector bc

9.101.1.4 mess_vector c

9.101.1.5 mess_vector cc

9.101.1.6 mess_datatype_t datatype

9.101.1.7 mess_int_t i

9.101.1.8 mess_int_t msolverid

9.101.1.9 mess_int_t n

9.101.1.10 mess_double_cpx_t sigma

9.101.1.11 mess_matrix V

9.101.1.12 mess_matrix W

10 File Documentation

10.1 __my_module__my_sub.c File Reference

Functions

- const char * __my_module__my_sub (void)

10.1.1 Function Documentation

10.1.1.1 const char* __my_module__my_sub (void)

10.2 __my_module__my_sub.c File Reference

Functions

- const char * __my_module__my_sub (void)

10.2.1 Function Documentation

10.2.1.1 const char* __my_module__my_sub (void)

10.3 __my_module_MOD_my_sub.c File Reference

Functions

- const char * __my_module_MOD_my_sub (void)

10.3.1 Function Documentation

10.3.1.1 const char* __my_module_MOD_my_sub (void)

10.4 __my_module_MOD_my_sub.c File Reference

Functions

- const char * __my_module_MOD_my_sub (void)

10.4.1 Function Documentation

10.4.1.1 const char* __my_module_MOD_my_sub (void)

10.5 __my_module_NMOD_my_sub.c File Reference

Functions

- const char * __my_module_NMOD_my_sub (void)

10.5.1 Function Documentation

10.5.1.1 const char* __my_module_NMOD_my_sub (void)

10.6 __my_module_NMOD_my_sub.c File Reference

Functions

- const char * __my_module_NMOD_my_sub (void)

10.6.1 Function Documentation

10.6.1.1 const char* __my_module_NMOD_my_sub (void)

10.7 __mymodule__mysub.c File Reference

Functions

- const char * __mymodule__mysub (void)

10.7.1 Function Documentation

10.7.1.1 const char* __mymodule__mysub (void)

10.8 __mymodule__mysub.c File Reference

Functions

- const char * __mymodule__mysub (void)

10.8.1 Function Documentation

10.8.1.1 const char* __mymodule__mysub (void)

10.9 __mymodule_MOD_mysub.c File Reference

Functions

- const char * __mymodule_MOD_mysub (void)

10.9.1 Function Documentation

10.9.1.1 const char* __mymodule_MOD_mysub (void)

10.10 __mymodule_MOD_mysub.c File Reference

Functions

- const char * __mymodule_MOD_mysub (void)

10.10.1 Function Documentation

10.10.1.1 const char* __mymodule_MOD_mysub (void)

10.11 __mymodule_NMOD_mysub.c File Reference

Functions

- const char * __mymodule_NMOD_mysub (void)

10.11.1 Function Documentation

10.11.1.1 const char* __mymodule_NMOD_mysub (void)

10.12 __mymodule_NMOD_mysub.c File Reference

Functions

- const char * __mymodule_NMOD_mysub (void)

10.12.1 Function Documentation

10.12.1.1 const char* __mymodule_NMOD_mysub (void)

10.13 add.c File Reference

Functions

- static void add_kernel_real (mess_int_t n, double alpha, mess_int_t *Asptr, mess_int_t *Aiptr, double *Avalues, double beta, mess_int_t *Bsptr, mess_int_t *Biptr, double *Bvalues, mess_int_t *sptr, mess_int_t *iptr, double *values)

  Add a real matrix to a real one (Kernel for MESS_CSC and MESS_CSR matrices).
- static void add_kernel_complex (mess_int_t n, mess_double_cpx_t alpha, mess_int_t *Asptr, mess_int_t *Aiptr, mess_double_cpx_t *Avalues, mess_double_cpx_t beta, mess_int_t *Bsptr, mess_int_t *Biptr, mess_double_cpx_t *Bvalues, mess_int_t *sptr, mess_int_t *iptr, mess_double_cpx_t *values)

  Add a complex matrix to a complex one (Kernel for MESS_CSC and MESS_CSR matrices).
- static void add_kernel_complex_real (mess_int_t n, mess_double_cpx_t alpha, mess_int_t *Asptr, mess_int_t *Aiptr, mess_double_cpx_t *Avalues, mess_double_cpx_t beta, mess_int_t *Bsptr, mess_int_t *Biptr, double *Bvalues, mess_int_t *sptr, mess_int_t *iptr, mess_double_cpx_t *values)

  Add a complex matrix to a real one (Kernel for MESS_CSC and MESS_CSR matrices).
- static void add_kernel_real_complex (mess_int_t n, mess_double_cpx_t alpha, mess_int_t *Asptr, mess_int_t *Aiptr, double *Avalues, mess_double_cpx_t beta, mess_int_t *Bsptr, mess_int_t *Biptr, mess_double_cpx_t *Bvalues, mess_int_t *sptr, mess_int_t *iptr, mess_double_cpx_t *values)

  Add a real matrix to a complex one (Kernel for MESS_CSC and MESS_CSR matrices).
- static int __mess_matrix_add_bothsparse_csr (double alpha, mess_matrix A, double beta, mess_matrix B)

  Add two real matrices (kernel for MESS_CSR matrices).
- static int __mess_matrix_add_bothsparse_csc (double alpha, mess_matrix A, double beta, mess_matrix B)

  Add two real matrices (kernel for MESS_CSC matrices).
- static int __mess_matrix_add_bothsparse_complex_csr (mess_double_cpx_t alpha, mess_matrix A, mess_double_cpx_t beta, mess_matrix B)

  Add two matrices (kernel for MESS_CSR matrices), both complex version.
- static int __mess_matrix_add_bothsparse_complex_csc (mess_double_cpx_t alpha, mess_matrix A, mess_double_cpx_t beta, mess_matrix B)

  Add two matrices (kernel for MESS_CSC matrices), both complex version.
- static int __mess_matrix_add_bothsparse_real_complex_csr (mess_double_cpx_t alpha, mess_matrix A, mess_double_cpx_t beta, mess_matrix B)

  Add two matrices (kernel for MESS_CSR matrices), real - complex version.
- static int __mess_matrix_add_bothsparse_real_complex_csc (mess_double_cpx_t alpha, mess_matrix A, mess_double_cpx_t beta, mess_matrix B)

  Add two matrices (kernel for MESS_CSC matrices), real - complex version.

- static int __mess_matrix_add_bothsparse_complex_real_csr (mess_double_cpx_t alpha, mess_matrix A, mess_double_cpx_t beta, mess_matrix B)

  Add two matrices (kernel for MESS_CSR matrices), complex-real version.
- static int __mess_matrix_add_bothsparse_complex_real_csc (mess_double_cpx_t alpha, mess_matrix A, mess_double_cpx_t beta, mess_matrix B)

  Add two matrices (kernel for MESS_CSC matrices), complex-real version.
- int mess_matrix_add (double alpha, mess_matrix A, double beta, mess_matrix B)

  Add two real matrices.
- int mess_matrix_addc (mess_double_cpx_t alpha, mess_matrix A, mess_double_cpx_t beta, mess_matrix B)

  Add two matrices.

10.13.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file contains various functions to perform matrix-matrix additions and similar problems.

10.13.2 Function Documentation

10.13.2.1 static int __mess_matrix_add_bothsparse_complex_csc (mess_double_cpx_t *alpha*, mess_matrix *A*, mess_double_cpx_t *beta*, mess_matrix *B*) [static]

Parameters

in	*alpha*	input coefficient in front of A
in	*A*	input matrix A
in	*beta*	input coefficient in front of B
in,out	*B*	input/output matrix B

Returns

zero on success or a non-zero error value otherwise

The __mess_matrix_add_bothsparse_complex_csc function add $alpha$ times a matrix A to β times a matrix B and store it in B. This is only the kernel for MESS_CSC matrices which is wrapped by mess_matrix_add.

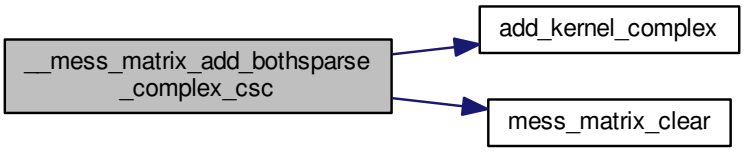
$$B \leftarrow \alpha A + \beta B$$

This function assumes that A and B are complex matrices.

Attention

Internal use only.

Here is the call graph for this function:



10.13.2.2 static int __mess_matrix_add_bothsparse_complex_csr (mess_double_cpx_t *alpha*, mess_matrix *A*, mess_double_cpx_t *beta*, mess_matrix *B*) [static]

Parameters

in	*alpha*	input coefficient in front of A
in	*A*	input matrix A
in	*beta*	input coefficient in front of B
in,out	*B*	input/output matrix B

Returns

zero on success or a non-zero error value otherwise

The __mess_matrix_add_bothsparse_complex_csr function add α times a matrix A to β times a matrix B and store it in B. This is only the kernel for MESS_CSR matrices which is wrapped by mess_matrix_add.

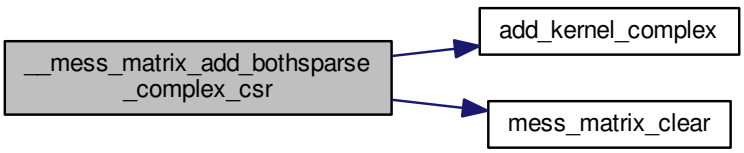
$$B \leftarrow \alpha A + \beta B$$

This function assumes that A and B are complex matrices.

Attention

Internal use only.

Here is the call graph for this function:



10.13.2.3 static int __mess_matrix_add_bothsparse_complex_real_csc (mess_double_cpx_t *alpha*, mess_matrix *A*, mess_double_cpx_t *beta*, mess_matrix *B*) [static]

Parameters

in	*alpha*	input coefficient in front of A
in	*A*	input matrix A
in	*beta*	input coefficient in front of B
in,out	*B*	input/output matrix B

Returns

zero on success or a non-zero error value otherwise

The __mess_matrix_add_bothsparse_complex_real_csc function add $alpha$ times a matrix A to β times a matrix B and store it in B. This is only the kernel for MESS_CSC matrices which is wrapped by mess_matrix_add.

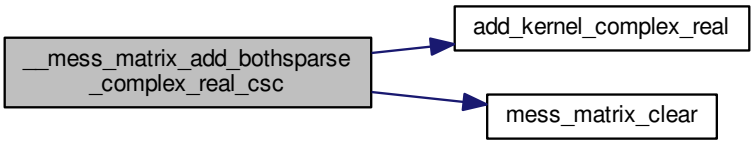
$$B \leftarrow \alpha A + \beta B$$

This function assumes that A is complex and B is real on input.

Attention

Internal use only.

Here is the call graph for this function:



10.13.2.4 static int __mess_matrix_add_bothsparse_complex_real_csr (mess_double_cpx_t *alpha*, mess_matrix *A*, mess_double_cpx_t *beta*, mess_matrix *B*) [static]

Parameters

in	*alpha*	input coefficient in front of A
in	*A*	input matrix A
in	*beta*	input coefficient in front of B
in,out	*B*	input/output matrix B

Returns

zero on success or a non-zero error value otherwise

The __mess_matrix_add_bothsparse_complex_real_csr function add $alpha$ times a matrix A to β times a matrix B and store it in B. This is only the kernel for MESS_CSR matrices which is wrapped by mess_matrix_add.

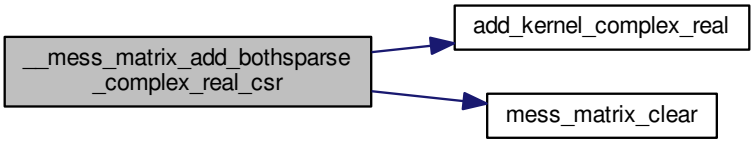
$$B \leftarrow \alpha A + \beta B$$

This function assumes that A is complex and B is real on input.

Attention

Internal use only.

Here is the call graph for this function:



10.13.2.5 static int __mess_matrix_add_bothsparse_csc (double *alpha*, mess_matrix *A*, double *beta*, mess_matrix *B*) [static]

Parameters

in	*alpha*	input coefficient in front of A
in	*A*	input matrix A
in	*beta*	input coefficient in front of B
in,out	*B*	input/output matrix B

Returns

zero on success or a non-zero error value otherwise

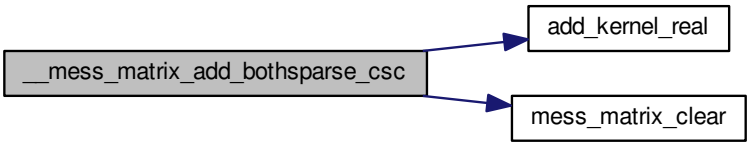
The __mess_matrix_add_bothsparse_csc function add α times a matrix A to β times a matrix B and store it in B. This is only the kernel for MESS_CSC matrices which is wrapped by mess_matrix_addc.

$$B \leftarrow \alpha A + \beta B$$

Attention

Internal use only.

Here is the call graph for this function:



10.13.2.6 static int __mess_matrix_add_bothsparse_csr (double *alpha*, mess_matrix *A*, double *beta*, mess_matrix *B*) [static]

Parameters

in	*alpha*	input coefficient in front of A
in	*A*	input matrix A
in	*beta*	input coefficient in front of B
in,out	*B*	input/output matrix B

Returns

zero on success or a non-zero error value otherwise

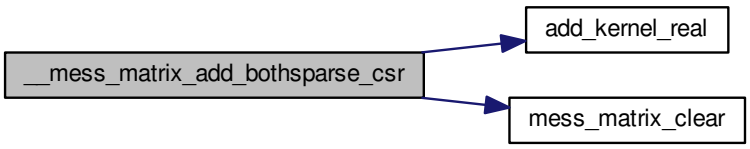
The __mess_matrix_add_bothsparse_csr function add α times a matrix A to β times a matrix B and store it in B. This is only the kernel for MESS_CSR matrices which is wrapped by mess_matrix_addc.

$$B \leftarrow \alpha A + \beta B$$

Attention

Internal use only.

Here is the call graph for this function:



10.13.2.7 static int __mess_matrix_add_bothsparse_real_complex_csc (mess_double_cpx_t *alpha*, mess_matrix *A*, mess_double_cpx_t *beta*, mess_matrix *B*) [static]

Parameters

in	*alpha*	input coefficient in front of A
in	*A*	input matrix A
in	*beta*	input coefficient in front of B
in,out	*B*	input/output matrix B

Returns

zero on success or a non-zero error value otherwise

The __mess_matrix_add_bothsparse_real_complex_csc function add $alpha$ times a matrix A to β times a matrix B and store it in B. This is only the kernel for MESS_CSC matrices which is wrapped by mess_matrix_add.

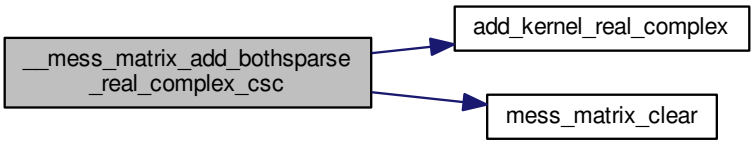
$$B \leftarrow \alpha A + \beta B$$

This function assumes that A is real and B is complex.

Attention

Internal use only.

Here is the call graph for this function:



10.13.2.8 static int __mess_matrix_add_bothsparse_real_complex_csr (mess_double_cpx_t *alpha*, mess_matrix *A*, mess_double_cpx_t *beta*, mess_matrix *B*) [static]

Parameters

in	*alpha*	input coefficient in front of A
in	*A*	input matrix A
in	*beta*	input coefficient in front of B
in,out	*B*	input/output matrix B

Returns

zero on success or a non-zero error value otherwise

The __mess_matrix_add_bothsparse_real_complex_csr function add $alpha$ times a matrix A to β times a matrix B and store it in B. This is only the kernel for MESS_CSR matrices which is wrapped by mess_matrix_add.

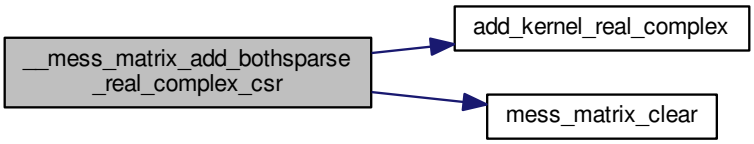
$$B \leftarrow \alpha A + \beta B$$

This function assumes that A is real and B is complex.

Attention

Internal use only.

Here is the call graph for this function:



10.13.2.9 static void add_kernel_complex (mess_int_t *n*, mess_double_cpx_t *alpha*, mess_int_t * *Asptr*, mess_int_t * *Aiptr*, mess_double_cpx_t * *Avalues*, mess_double_cpx_t *beta*, mess_int_t * *Bsptr*, mess_int_t * *Biptr*, mess_double_cpx_t * *Bvalues*, mess_int_t * *sptr*, mess_int_t * *iptr*, mess_double_cpx_t * *values*) [static]

Parameters

in	*n*	input dimension of the first indexpointer (row or column pointer)
in	*alpha*	input factor α
in	*Asptr*	input array with the start indices of the rows/columns in A
in	*Aiptr*	input array with the column/rows indices of the values of A
in	*Avalues*	input array containing the values of A
in	*beta*	input factor β
in	*Bsptr*	input array with the start indices of the rows/columns in B
in	*Biptr*	input array with the column/rows indices of the values of B
in	*Bvalues*	input array containing the values of B
out	*sptr*	output array with the start indices of the rows/columns in the result
out	*iptr*	output array with the column/row indices of the values in the result
out	*values*	output array with the values of the result

Returns

Nothing

The add_kernel_complex function adds two complex matrices stored whether in MESS_CSR or MESS_CSC storage.

Attention

Internal use only.

10.13.2.10 static void add_kernel_complex_real (mess_int_t *n*, mess_double_cpx_t *alpha*, mess_int_t * *Asptr*, mess_int_t * *Aiptr*, mess_double_cpx_t * *Avalues*, mess_double_cpx_t *beta*, mess_int_t * *Bsptr*, mess_int_t * *Biptr*, double * *Bvalues*, mess_int_t * *sptr*, mess_int_t * *iptr*, mess_double_cpx_t * *values*) [static]

Parameters

in	*n*	input dimension of the first index pointer (row or column pointer)
in	*alpha*	input factor α
in	*Asptr*	input array with the start indices of the rows/columns in A
in	*Aiptr*	input array with the column/rows indices of the values of A
in	*Avalues*	input array containing the values of A
in	*beta*	input factor β
in	*Bsptr*	input array with the start indices of the rows/columns in B
in	*Biptr*	input array with the column/rows indices of the values of B
in	*Bvalues*	input array containing the values of B
out	*sptr*	output array with the start indices of the rows/columns in the result
out	*iptr*	output array with the column/row indices of the values in the result
out	*values*	output array with the values of the result

Returns

Nothing

The add_kernel_complex_real function adds a complex matrix to a real one. The result will be complex again. The matrices stored whether in MESS_CSR or MESS_CSC storage.

Attention

Internal use only.

10.13.2.11 static void add_kernel_real (mess_int_t *n*, double *alpha*, mess_int_t * *Asptr*, mess_int_t * *Aiptr*, double * *Avalues*, double *beta*, mess_int_t * *Bsptr*, mess_int_t * *Biptr*, double * *Bvalues*, mess_int_t * *sptr*, mess_int_t * *iptr*, double * *values*) [static]

Parameters

in	*n*	input dimension of the first index pointer (row or column pointer)
in	*alpha*	input factor $alpha$
in	*Asptr*	input array with the start indices of the rows/columns in A
in	*Aiptr*	input array with the column/rows indices of the values of A
in	*Avalues*	input array containing the values of A
in	*beta*	input factor β
in	*Bsptr*	input array with the start indices of the rows/columns in B
in	*Biptr*	input array with the column/rows indices of the values of B
in	*Bvalues*	input array containing the values of B
out	*sptr*	output array with the start indices of the rows/columns in the result
out	*iptr*	output array with the column/row indices of the values in the result
out	*values*	output array with the values of the result

Returns

Nothing

The add_kernel_real function adds two real matrices stored whether in MESS_CSR or MESS_CSC storage.

Attention

Internal use only.

10.13.2.12 static void add_kernel_real_complex (mess_int_t *n*, mess_double_cpx_t *alpha*, mess_int_t * *Asptr*, mess_int_t * *Aiptr*, double * *Avalues*, mess_double_cpx_t *beta*, mess_int_t * *Bsptr*, mess_int_t * *Biptr*, mess_double_cpx_t * *Bvalues*, mess_int_t * *sptr*, mess_int_t * *iptr*, mess_double_cpx_t * *values*) [static]

Parameters

in	*n*	input dimension of the first index pointer (row or column pointer)
in	*alpha*	input factor α
in	*Asptr*	input array with the start indices of the rows/columns in A
in	*Aiptr*	input array with the column/rows indices of the values of A
in	*Avalues*	input array containing the values of A
in	*beta*	input factor β
in	*Bsptr*	input array with the start indices of the rows/columns in B
in	*Biptr*	input array with the column/rows indices of the values of B
in	*Bvalues*	input array containing the values of B
out	*sptr*	output array with the start indices of the rows/columns in the result
out	*iptr*	output array with the column/row indices of the values in the result
out	*values*	output array with the values of the result

Returns

Nothing

The add_kernel_real_complex function adds a real matrix to a complex one. The result will be complex again. The matrices stored whether in MESS_CSR or MESS_CSC storage.

Attention

Internal use only.

10.14 addcol.c File Reference

Functions

- int mess_matrix_addcols (mess_matrix matrix, mess_matrix toadd)

  Add columns to an existing matrix.
- int mess_matrix_addcols2p (mess_matrix Z, double s1, mess_matrix V1, double s2, mess_matrix V2)

  Add the sum of two column matrices to a real matrix.
- int mess_matrix_addcols1 (mess_matrix Z, double s1, mess_matrix V1)

  Add a scaled column to a real matrix.

10.14.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.15 alloc.c File Reference

Macros

- #define NEW_LD

Functions

- void __mess_matrix_alloc_setld (mess_int_t ld_m)

  Sets the multiplicity of the leading dimension for dense matrices.
- static mess_int_t __select_ld (mess_int_t rows, size_t es)
- int mess_matrix_need_alloc (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t nnz, mess_storage_t storetype, mess_datatype_t datatype)

  Check if a new allocation of a matrix is necessary.
- int mess_matrix_alloc (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t nnz, mess_storage_t storetype, mess_datatype_t datatype)

  Allocate a sparse or dense $(m \times n)$ *matrix.*
- int mess_matrix_resize (mess_matrix matrix, mess_int_t rows, mess_int_t cols)

  Wrapper for mess_matrix_realloc.
- int mess_matrix_realloc (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t nnz)

  Reallocate or resize a $(m \times n)$ *matrix.*
- int mess_matrix_reset (mess_matrix matrix)

  Resets a mess_matrix object.

Variables

- static mess_int_t ld_multiplicity = 1

10.15.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.15.2 Macro Definition Documentation

10.15.2.1 #define NEW_LD

10.15.3 Function Documentation

10.15.3.1 void __mess_matrix_alloc_setld (mess_int_t *ld_m*)

Parameters

in	*ld↩ _m*	Multiplicity of the leading dimension.

The __mess_matrix_alloc_setld function sets the multiplicity of the leading dimension of a matrix. Everytime a dense matrix gets allocated by mess_matrix_alloc the leading dimension is selected to be a multiple of `ld_m`. If ld_m <= 0 the multiplicity is set 1. Changing the multiplicity of the leading dimension may influence the behaviour of level-3 BLAS routines.

Attention

Internal use only.

10.15.3.2 static mess_int_t __select_ld (mess_int_t *rows*, size_t *es*) [static]

10.15.4 Variable Documentation

10.15.4.1 mess_int_t ld_multiplicity = 1 [static]

10.16 alloc.c File Reference

Functions

- int mess_vector_alloc (mess_vector vect, mess_int_t dim, mess_datatype_t dtype)

  Allocate a MESS_REAL or MESS_COMPLEX the mess_vector structure.
- int mess_vector_clear (mess_vector *vect)

  Clean up the mess_vector structure to use it again.
- int mess_vector_reset (mess_vector v)

  Resets a mess_vector object.

10.16.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.17 amd.c File Reference

Macros

- #define AMD_VALID amd_valid
- #define AMD_ORDER amd_order
- #define AMD_PINFO amd_info
- #define COLAMD_RECOMMENDED colamd_recommended
- #define COLAMD_SET_DEFAULTS colamd_set_defaults
- #define COLAMD colamd
- #define COLAMD_REPORT colamd_report

Functions

- int mess_matrix_reorder_amd (mess_matrix A, mess_int_t *p)

  Compute AMD *reordering of a matrix.*
- int mess_matrix_reorder_colamd (mess_matrix A, mess_int_t *p)

  Compute the COLAMD *reordering of a matrix.*

10.17.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.17.2 Macro Definition Documentation

10.17.2.1 #define AMD_ORDER amd_order

10.17.2.2 #define AMD_PINFO amd_info

10.17.2.3 #define AMD_VALID amd_valid

10.17.2.4 #define COLAMD colamd

10.17.2.5 #define COLAMD_RECOMMENDED colamd_recommended

10.17.2.6 #define COLAMD_REPORT colamd_report

10.17.2.7 #define COLAMD_SET_DEFAULTS colamd_set_defaults

10.18 any.c File Reference

Functions

- int mess_matrix_any (mess_matrix mat, mess_int_t(*f_real)(double), mess_int_t(*f_cpx)(mess_double_cpx_t), mess_int_t dim, mess_vector anyvec)

  Checks a matrix rowwise/columnwise if a given predicate is true.

10.18.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.19 any.c File Reference

Functions

- int mess_vector_any (mess_vector v, mess_int_t(*f_real)(double), mess_int_t(*f_cpx)(mess_double_cpx_t), mess_int_t *anyval)

  Checks a mess_vector if a given predicate is true.

10.19.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.20 arnoldi.c File Reference

Data Structures

- struct mvpdata
- struct mvpdatag

Functions

- static int mvp_arnoldi (void *data, mess_operation_t op, mess_vector in, mess_vector out)
- int mess_eigen_arnoldi (mess_matrix A, mess_matrix U, mess_matrix W, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method with respect to A or $(A - UW)$.
- static int mvp_arnoldig (void *data, mess_operation_t op, mess_vector in, mess_vector out)
- int mess_eigen_arnoldig (mess_matrix A, mess_matrix U, mess_matrix W, mess_direct E, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method with respect to $E^{-1}A$ or $E^{-1}(A - UW)$.
- static int mvp_arnoldi_inv (void *data, mess_operation_t op, mess_vector in, mess_vector out)
- int mess_eigen_arnoldi_inv (mess_direct A, mess_matrix U, mess_matrix W, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method on A^{-1} or $(A - UW)^{-1}$.
- static int mvp_arnoldig_inv (void *data, mess_operation_t op, mess_vector in, mess_vector out)
- int mess_eigen_arnoldig_inv (mess_direct A, mess_matrix U, mess_matrix W, mess_matrix E, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method on $A^{-1}E$ or $(A - UW)^{-1}E$.
- int mess_eigen_eigs (mess_matrix A, mess_int_t kp, mess_int_t km, mess_vector r, mess_vector ev)

  Compute some eigenvalues with the Arnoldi process.
- int mess_eigen_eiggs (mess_matrix A, mess_matrix E, mess_int_t kp, mess_int_t km, mess_vector r, mess_vector ev)

  Compute some generalized eigenvalues with the Arnoldi process.

10.20.1 Detailed Description

Author

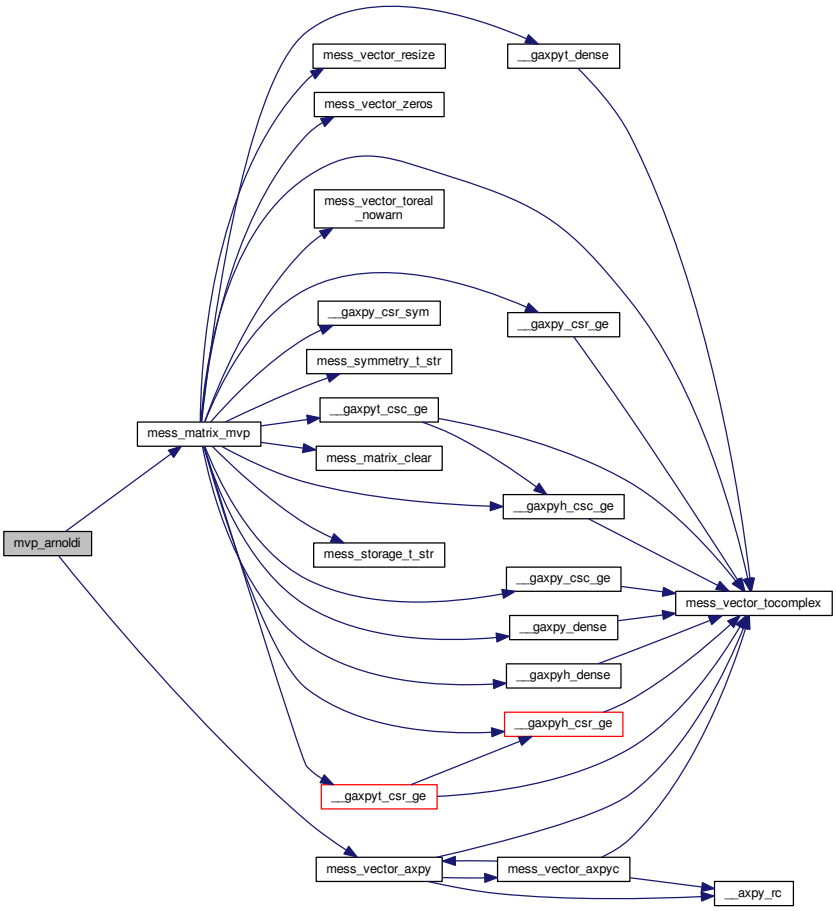
Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file contains different variants of the Arnoldi process.

10.20.2 Function Documentation

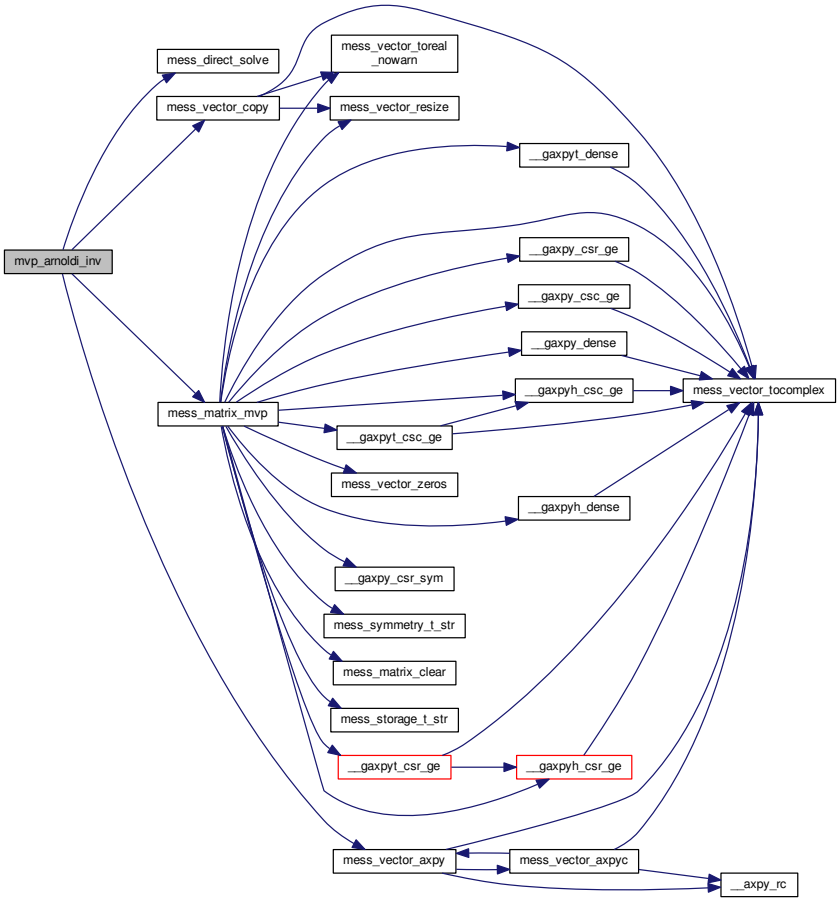
10.20.2.1 static int mvp_arnoldi (void * *data,* mess_operation_t *op,* mess_vector *in,* mess_vector *out*) [static]

Here is the call graph for this function:



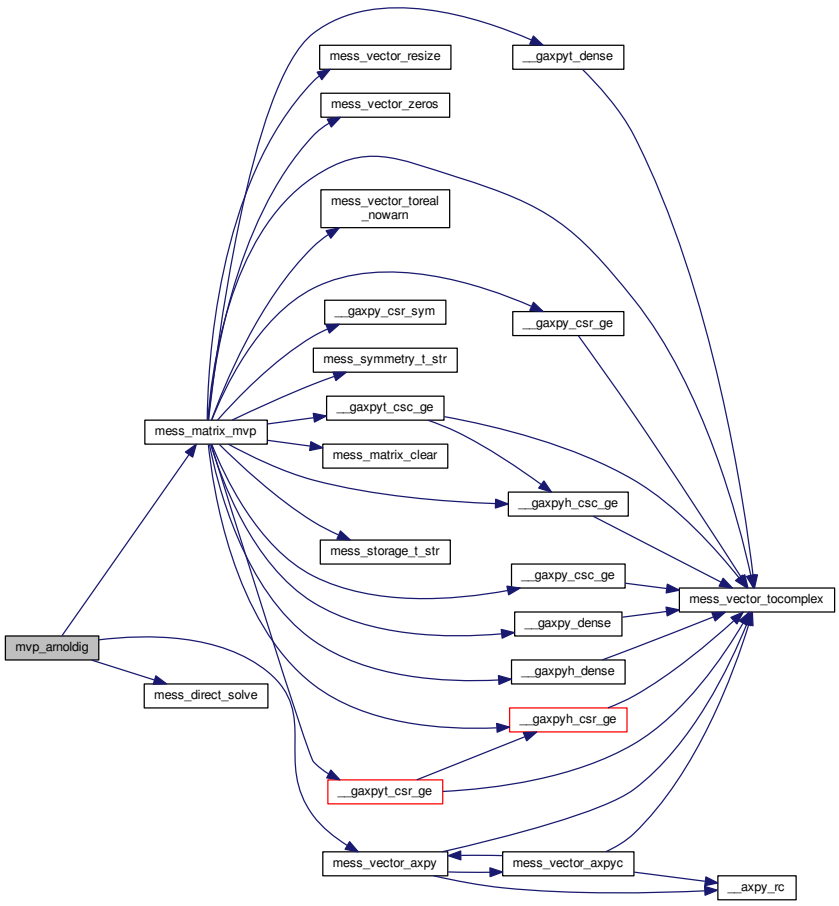
10.20.2.2 static int mvp_arnoldi_inv (void * *data,* mess_operation_t *op,* mess_vector *in,* mess_vector *out*) [static]

Here is the call graph for this function:



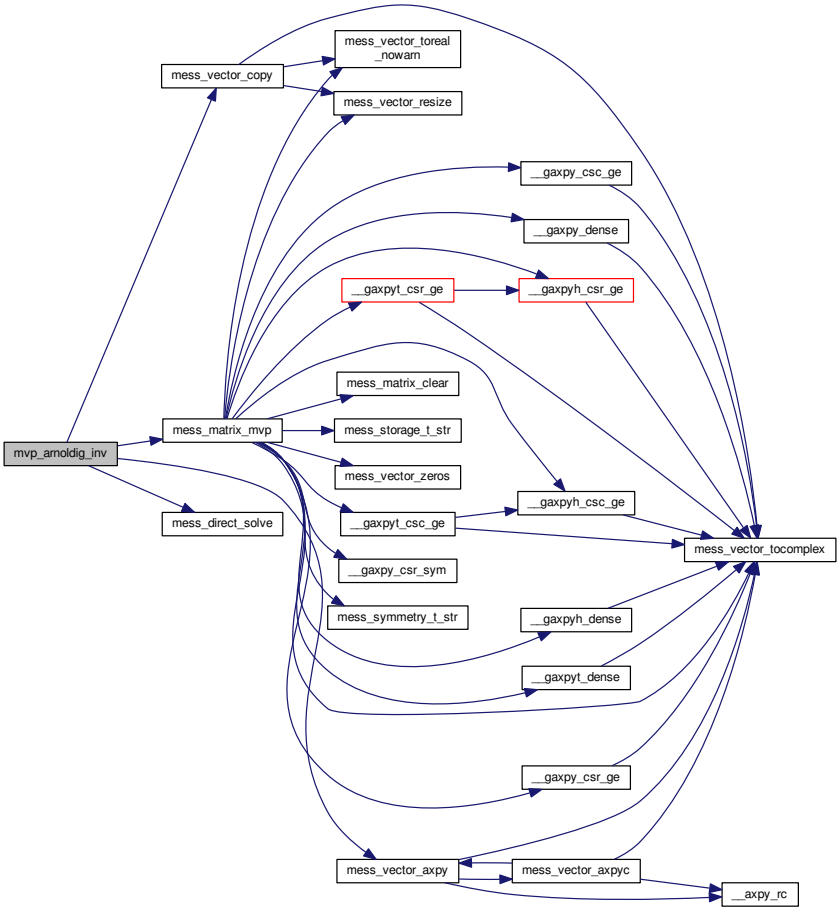
10.20.2.3 static int mvp_arnoldig (void * *data*, mess_operation_t *op*, mess_vector *in*, mess_vector *out*) [static]

Here is the call graph for this function:



10.20.2.4 static int mvp_arnoldig_inv (void * *data*, mess_operation_t *op*, mess_vector *in*, mess_vector *out*) [static]

Here is the call graph for this function:



10.21 arnoldi_template.c File Reference

Data Structures

- struct mess_krylov_arnoldi_st

Macros

- #define ARNOLDI_BREAKDOWN_TOL mess_eps()
- #define ARNOLDI_RESIZE_CHUNK 20

Typedefs

- typedef struct mess_krylov_arnoldi_st * mess_krylov_arnoldi

Functions

- int mess_krylov_arnoldi_init (mess_krylov_arnoldi *arn)

  Initialize a mess_krylov_arnoldi structure.
- int mess_krylov_arnoldi_clear (mess_krylov_arnoldi *arn)

  Clear a mess_krylov_arnoldi object.
- int mess_krylov_arnoldi_create (mess_krylov_arnoldi arn, mess_mvpcall A, mess_vector sv, mess_int_t k↩_est_max_steps)

  Create an Arnoldi process from mess_mvpcall.
- int mess_krylov_arnoldi_print (mess_krylov_arnoldi arn)

  Print fields of mess_krylov_arnoldi structure.
- int mess_krylov_arnoldi_step (mess_krylov_arnoldi arn, mess_mvpcall A)

  Perform exactly one steps of Arnoldi process.
- int mess_krylov_arnoldi_number_arnoldi_vectors (mess_krylov_arnoldi arn, mess_int_t *no_arnoldi_↩vectors)

  Return number of computed arnoldi vectors of mess_krylov_arnoldi.
- int mess_krylov_arnoldi_getV (mess_krylov_arnoldi arn, mess_matrix V)

  Return matrix which contains the Arnoldi vectors columnwise.
- int mess_krylov_arnoldi_getH (mess_krylov_arnoldi arn, mess_matrix H)

  Return Hessenberg matrix generated by Arnoldi process.
- int mess_krylov_arnoldi_H_largest_eigenvalue (mess_krylov_arnoldi arn, mess_double_cpx_t *lambda, double *err_est)

  Return largest eigenvalue of Hessenberg matrix and error estimation.
- int mess_eigen_arnoldi_template (mess_mvpcall A, mess_int_t k, mess_vector sv, mess_matrix H, mess_↩matrix V)

  Compute matrices H and V with Arnoldis method with respect to A.
- int mess_eigen_arnoldi_template_nrm (mess_mvpcall A, mess_int_t k, mess_vector sv, double *abseig)

  Compute absolut value of the largest eigenvalue using Arnoldi process.

10.21.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file contains the template for the arnoldi process.

10.21.2 Macro Definition Documentation

10.21.2.1 #define ARNOLDI_BREAKDOWN_TOL mess_eps()

10.21.2.2 #define ARNOLDI_RESIZE_CHUNK 20

10.21.3 Typedef Documentation

10.21.3.1 typedef struct mess_krylov_arnoldi_st* mess_krylov_arnoldi

Type definition for Arnoldi process.

Attention

Internal use only.

10.21.4 Function Documentation

10.21.4.1 int mess_krylov_arnoldi_clear (mess_krylov_arnoldi * *arn*)

Parameters

in,out	*arn*	pointer to a mess_krylov_arnoldi object

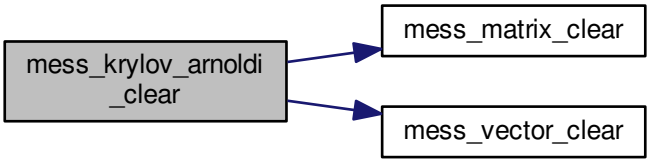
Returns

zero on success or a non-zero error value otherwise

The mess_krylov_arnoldi_clear function cleans up a mess_krylov_arnoldi object.

mess_krylov_arnoldi_init mess_krylov_arnoldi_create

Here is the call graph for this function:



10.21.4.2 int mess_krylov_arnoldi_create (mess_krylov_arnoldi *arn*, mess_mvpcall *A*, mess_vector *sv*, mess_int_t *k_est_max_steps*)

Parameters

in	*arn*	input, pointer to a mess_krylov_arnoldi object
in	*A*	input, represents the operator A in Arnoldi process.
in	*sv*	input, nonzero initial vector, must have same datatype as `mess_mvpcall`.
in	*k_est_max_steps*	input, number of arnoldi steps, positive and smaller than the order of the operator `A`.

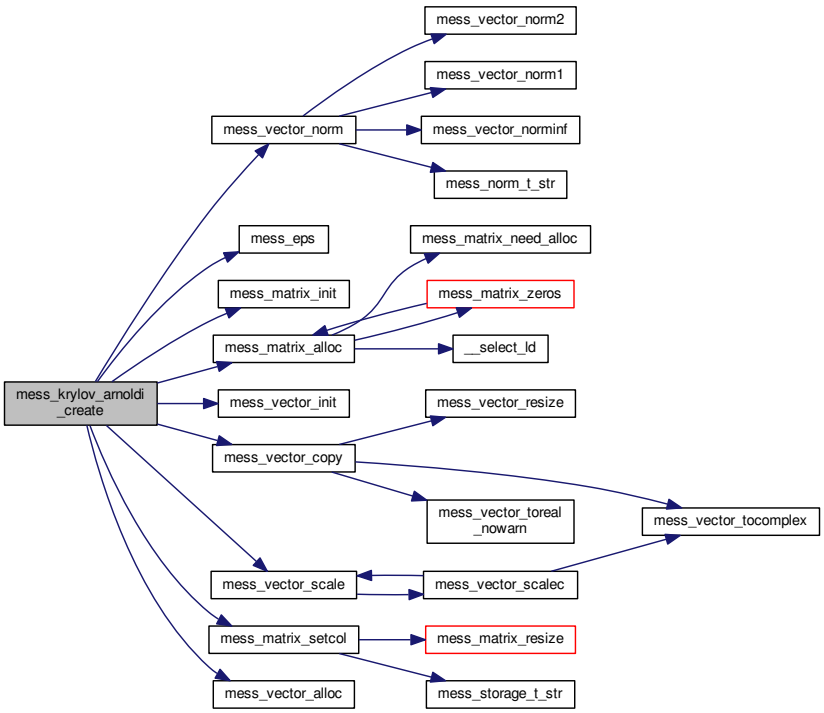
Returns

zero on success or a non-zero error value otherwise

The mess_krylov_arnoldi_create function prepares the Arnoldi process. The argument `k_est_max_steps` is an estimation of the maximal number of iterations needed. If more steps are needed, then memory reallocation occur for the Hessenberg matrix and matrix of Arnoldi vectors.

mess_krylov_arnoldi_init mess_krylov_arnoldi_clear

Here is the call graph for this function:



10.21.4.3 int mess_krylov_arnoldi_getH (mess_krylov_arnoldi *arn*, mess_matrix *H*)

Parameters

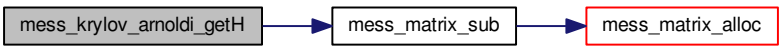
in	*arn*	input, mess_krylov_arnoldi process.
out	*H*	output, generated Hessenberg matrix

Returns

zero on success or a non-zero error value otherwise

The mess_krylov_arnoldi_getH function returns the Hessenberg Matrix generated by `arn`,

Here is the call graph for this function:



10.21.4.4 int mess_krylov_arnoldi_getV (mess_krylov_arnoldi *arn*, mess_matrix *V*)

Parameters

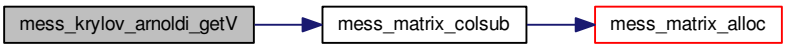
in	*arn*	input, mess_krylov_arnoldi process.
out	*V*	output, matrix of Arnoldi vectors columnwise

Returns

zero on success or a non-zero error value otherwise

The mess_krylov_arnoldi_getV function returns the Arnoldi vectors which `arn` currently holds in a matrix `V` columnwise.

Here is the call graph for this function:



10.21.4.5 int mess_krylov_arnoldi_H_largest_eigenvalue (mess_krylov_arnoldi *arn*, mess_double_cpx_t * *lambda*, double * *err_est*)

Parameters

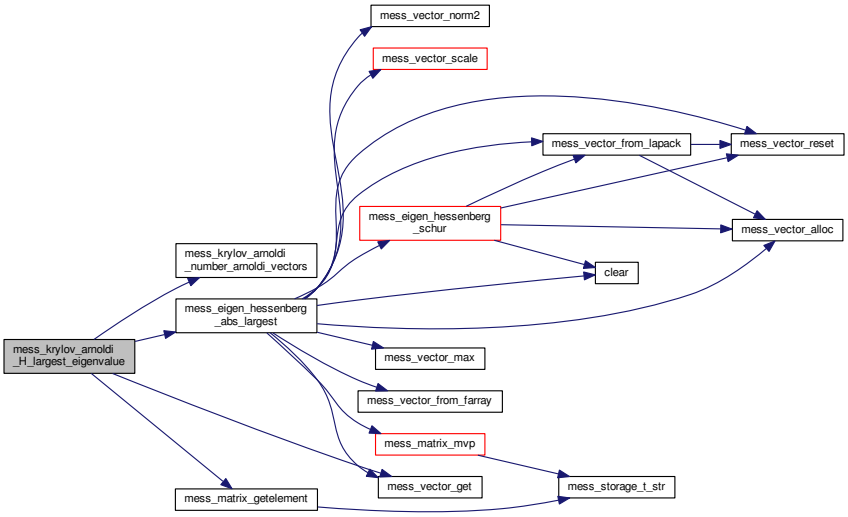
in	*arn*	input, mess_krylov_arnoldi process.
out	*lambda*	output, largest eigenvalue of Hessenberg matrix in magnitude
out	*err_est*	output, error estimation for eigen residual.

Returns

zero on success or a non-zero error value otherwise

The mess_krylov_arnoldi_H_largest_eigenvalue returns the largest eigenvalue in magnitude of the Hessenberg matrix generated by the Arnoldi process `arn`. `err_est` estimates the error of the eigenresidual $|Av - \lambda v|_2$. For more information read Proposition 6.8 of [26]. Matrix generated by `arn`,

Here is the call graph for this function:



10.21.4.6 int mess_krylov_arnoldi_init (mess_krylov_arnoldi * *arn*)

Parameters

in,out	*arn*	pointer to a mess_krylov_arnoldi.

Returns

zero on success or a non-zero error value otherwise

The mess_krylov_arnoldi_init function initializes a mess_krylov_arnoldi object.

mess_krylov_arnoldi_clear mess_krylov_arnoldi_create

10.21.4.7 int mess_krylov_arnoldi_number_arnoldi_vectors (mess_krylov_arnoldi *arn*, mess_int_t * *no_arnoldi_vectors*)

Parameters

in	*arn*	input, mess_krylov_arnoldi process.
out	*no_arnoldi_vectors*	output, on exit number of Arnoldi vectors of `arn`

Returns

zero on success or a non-zero error value otherwise

The mess_krylov_arnoldi_number_arnoldi_vectors function returns the number of Arnoldi vectors which `arn` holds.

10.21.4.8 int mess_krylov_arnoldi_print (mess_krylov_arnoldi *arn*)

Parameters

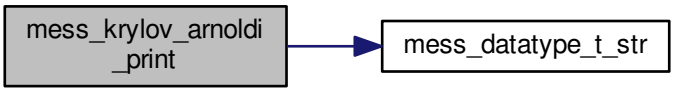
in	*arn*	input, pointer to a mess_krylov_arnoldi object

Returns

zero on success or a non-zero error value otherwise

The mess_krylov_arnoldi_print function prints a mess_krylov_arnoldi structure.

Here is the call graph for this function:



10.21.4.9 int mess_krylov_arnoldi_step (mess_krylov_arnoldi *arn*, mess_mvpcall *A*)

Parameters

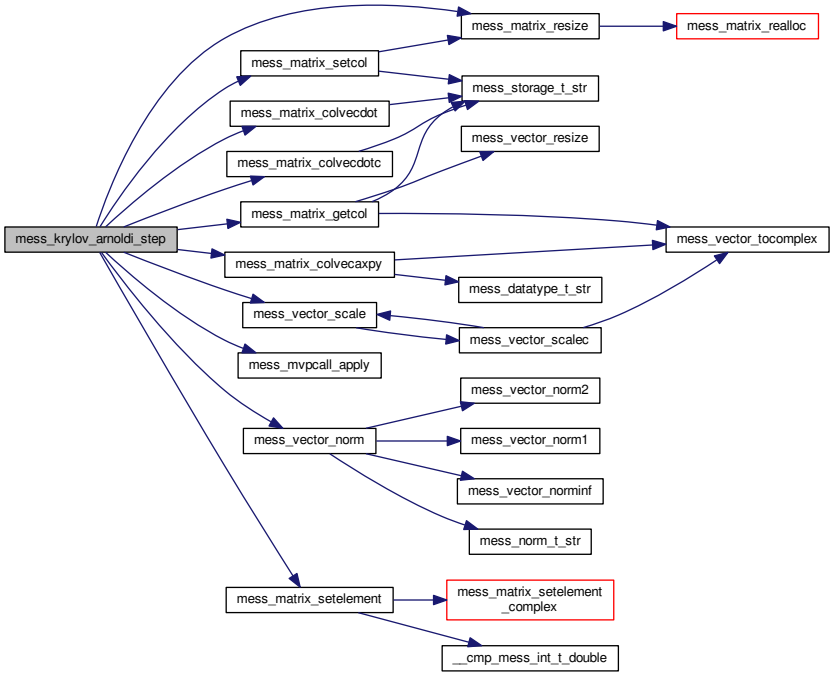
in	*arn*	input, mess_krylov_arnoldi process.
in	*A*	input, mess_mvpcall A

Returns

zero on success or a non-zero error value otherwise

The mess_krylov_arnoldi_step function performs exactly one step of Arnoldi process. If the number of computed arnoldi vectors is equal to the order of A an MESS_ERROR_CONVERGE is returned. If `arn` has already reached state of early breakdown MESS_ERROR_CONVERGE is also returned.

Here is the call graph for this function:



10.22 arpack.c File Reference

Macros

- #define ARPACK_IPARAM_SIZE 11
- #define ARPACK_IPNTR_SIZE 14

Functions

- void F77_GLOBAL (dnaupd, DNAUPD)
- void arpack_print_iparam (mess_int_t *iparam)

  Print `ARPACK` *iparam.*
- static int arpack_arnoldi_real (mess_mvpcall A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)
- static int arpack_arnoldi_complex (mess_mvpcall A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)
- static int arpack_lanczos_real (mess_mvpcall A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)
- int mess_eigen_arpack (mess_matrix A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)

  An interface to the Arnoldi process from `ARPACK`.
- int mess_eigen_arpack_template (mess_mvpcall A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)

  An interface to the Arnoldi process from `ARPACK` *(template version).*
- int mess_eigen_arpack_lanczos (mess_matrix A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)

An interface to the Lanczos process from ARPACK.

- int mess_eigen_arpack_lanczos_template (mess_mvpcall A, mess_int_t nev, mess_eigen_arpack_options↩_t opt, mess_vector ev, mess_matrix V)

  An interface to the Lanczos process from ARPACK *(template version).*

10.22.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file implements an interface to ARPACK.

10.22.2 Macro Definition Documentation

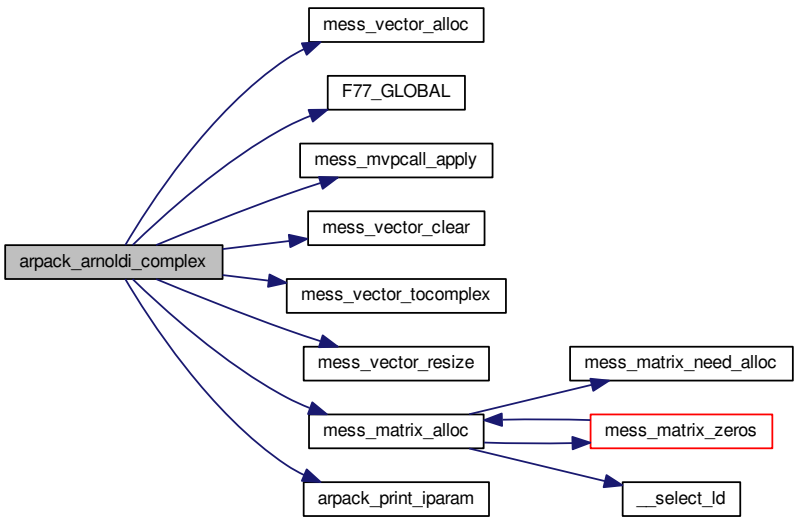
10.22.2.1 #define ARPACK_IPARAM_SIZE 11

10.22.2.2 #define ARPACK_IPNTR_SIZE 14

10.22.3 Function Documentation

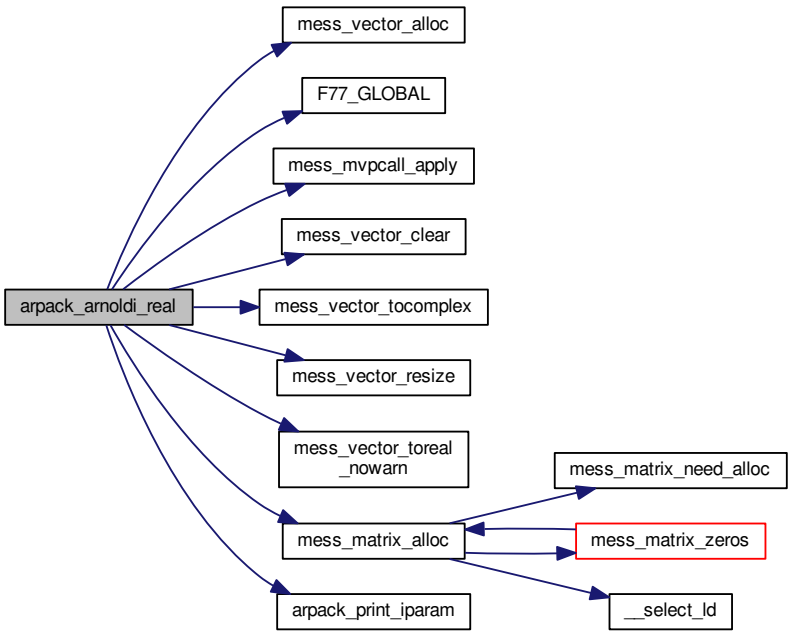
10.22.3.1 static int arpack_arnoldi_complex (mess_mvpcall *A*, mess_int_t *nev*, mess_eigen_arpack_options_t *opt*, mess_vector *ev*, mess_matrix *V*) [static]

Here is the call graph for this function:



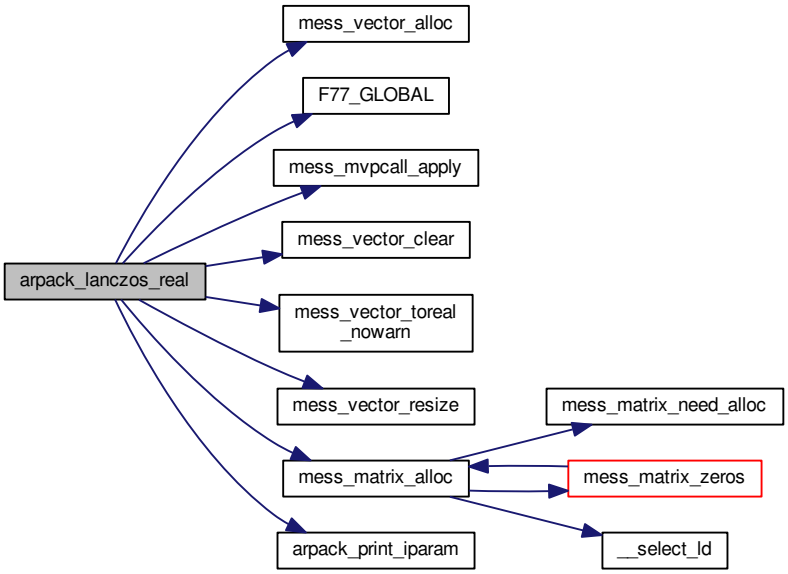
10.22.3.2 static int arpack_arnoldi_real (mess_mvpcall *A*, mess_int_t *nev*, mess_eigen_arpack_options_t *opt*, mess_vector *ev*, mess_matrix *V*) [static]

Here is the call graph for this function:



10.22.3.3 static int arpack_lanczos_real (mess_mvpcall *A*, mess_int_t *nev*, mess_eigen_arpack_options_t *opt*, mess_vector *ev*, mess_matrix *V*) [static]

Here is the call graph for this function:



10.22.3.4 void arpack_print_iparam (mess_int_t * *iparam*)

Parameters

in	*iparam*	input pointer to different shifts

The arpack_print_iparam function prints all shifts to stderr.

10.22.3.5 void F77_GLOBAL (dnaupd , DNAUPD)

10.23 axpy.c File Reference

Functions

- static int __axpy_rc (mess_double_cpx_t a, mess_vector x, mess_vector y)

  Internal axpy for real/complex mixing.
- int mess_vector_axpy (double a, mess_vector x, mess_vector y)

  Compute $y \leftarrow ax + y$ *(real).*
- int mess_vector_axpyc (mess_double_cpx_t a, mess_vector x, mess_vector y)

  Compute $y \leftarrow ax + y$ *(complex).*

Variables

- static mess_int_t __ONE = 1

10.23.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.23.2 Function Documentation

10.23.2.1 static int __axpy_rc (mess_double_cpx_t *a*, mess_vector *x*, mess_vector *y*) [static]

Parameters

in	*a*	input mess_double_cpx_t calar
in	*x*	input first vector
in,out	*y*	input/output second vector

Returns

zero on success or a non-zero error value otherwise

The __axpy_rc function computes

$$y \leftarrow ax + y$$

with

$$x$$

real and

$$y$$

complex.

Attention

Internal use only.

10.23.3 Variable Documentation

10.23.3.1 mess_int_t __ONE = 1 [static]

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

10.24 backslash.c File Reference

Functions

- int mess_matrix_backslash (mess_operation_t op, mess_matrix A, mess_vector b, mess_vector x)

  Solve $op(A)x = b$ with b and x vectors.
- int mess_matrix_backslashm (mess_operation_t op, mess_matrix A, mess_matrix b, mess_matrix x)

  Solve $op(A)x = b$ with b and x matrices.

10.24.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.25 banded.c File Reference

Data Structures

- struct banded_lapack

Macros

- #define umfpack_zl_defaults umfpack_zi_defaults
- #define umfpack_dl_defaults umfpack_di_defaults
- #define BANDWIDHT_SWITCH_THRESHOLD 0.3
- #define IDX2F(i, j, ld) (((((j)-1))∗(ld))+ (((i)-1)))
- #define LAPACK_ERROR(INFO)

Functions

- static int banded_lapack_solve (void ∗data, mess_vector b, mess_vector x)

  Solve $Ax = b$.
- static int banded_lapack_solvet (void ∗data, mess_vector b, mess_vector x)

  Solve $A^T x = b$.
- static int banded_lapack_solveh (void ∗data, mess_vector b, mess_vector x)

  Solve $A^H x = b$.
- static int banded_lapack_solvem (void ∗data, mess_matrix b, mess_matrix x)

  Solve $AX = B$.
- static int banded_lapack_solvemt (void ∗data, mess_matrix b, mess_matrix x)

  Solve $A^T X = B$.
- static int banded_lapack_solvemh (void ∗data, mess_matrix b, mess_matrix x)

  Solve $A^H X = B$.
- static int banded_lapack_clear (void ∗data)

  Clear the banded solver.
- int mess_direct_create_banded (mess_matrix matrix, mess_direct solver)

  Generate a direct linear banded solver for standard linear systems $Ax = b$ *with LAPACK.*

10.25.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.25.2 Macro Definition Documentation

10.25.2.1 #define BANDWIDHT_SWITCH_THRESHOLD 0.3

10.25.2.2 #define IDX2F(*i*, *j*, *ld*) (((((j)-1))∗(ld))+ (((i)-1)))

10.25.2.3 #define LAPACK_ERROR(INFO)

Value:

```
if ( (INFO) < 0) {
          \
        MSG_ERROR("error calling DGBTRS/DZGBTRS/ZGBTRS. Invalid argument: "
      MESS_PRINTF_INT "\n", -(INFO));      \
        return (int) (INFO);
              \
    }
```

10.25.2.4 #define umfpack_dl_defaults umfpack_di_defaults

10.25.2.5 #define umfpack_zl_defaults umfpack_zi_defaults

10.25.3 Function Documentation

10.25.3.1 static int banded_lapack_clear (void ∗ *data*) [static]

Parameters

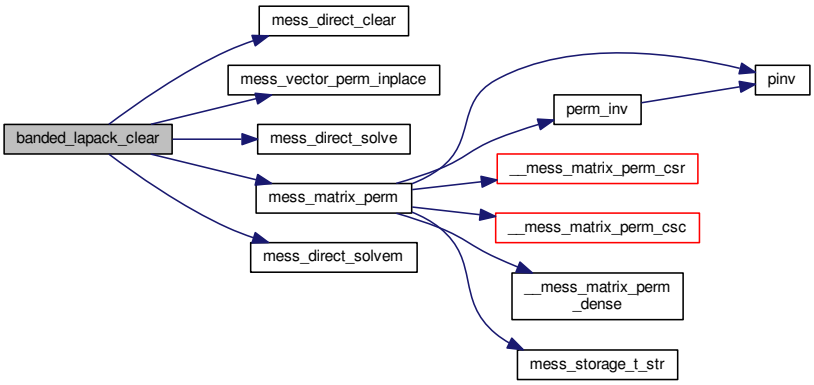
in,out	*data*	pointer to the internal data

Returns

always zero

The banded_lapack_clear function clears a banded solver.

Here is the call graph for this function:



10.25.3.2 static int banded_lapack_solve (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

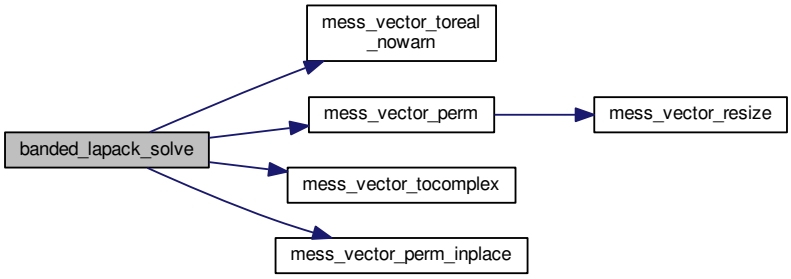
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The banded_lapack_solve function solves $Ax = b$.

Here is the call graph for this function:



10.25.3.3 static int banded_lapack_solveh (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

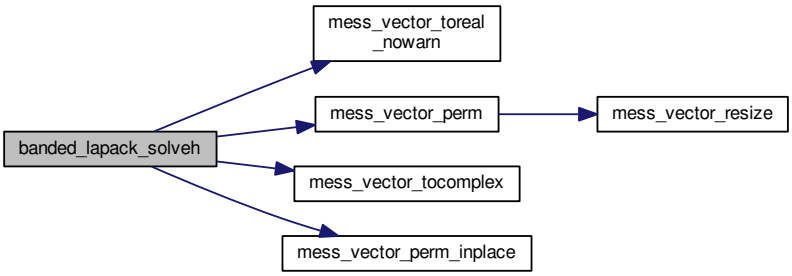
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The banded_lapack_solveh function solves $A^H x = b$.

Here is the call graph for this function:



10.25.3.4 static int banded_lapack_solvem (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

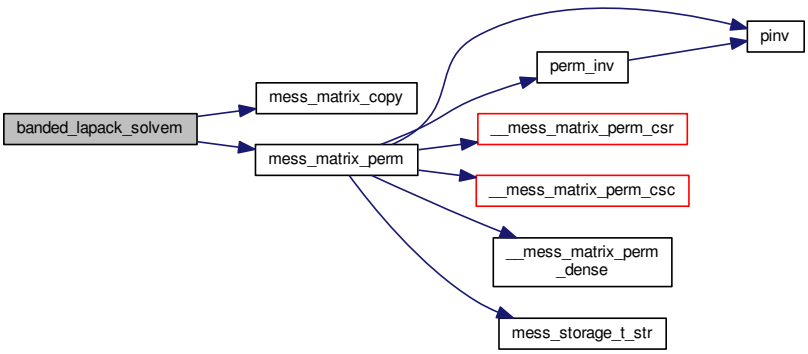
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The banded_lapack_solvem function solves $AX = B$.

Here is the call graph for this function:



10.25.3.5 static int banded_lapack_solvemh (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

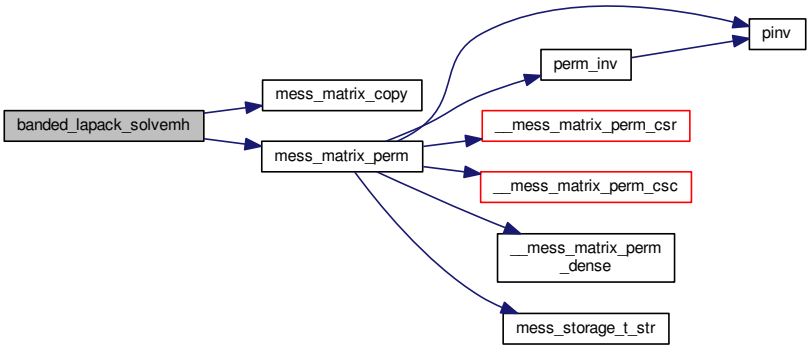
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The banded_lapack_solvemh function solves $A^H X = B$.

Here is the call graph for this function:



10.25.3.6 static int banded_lapack_solvemt (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

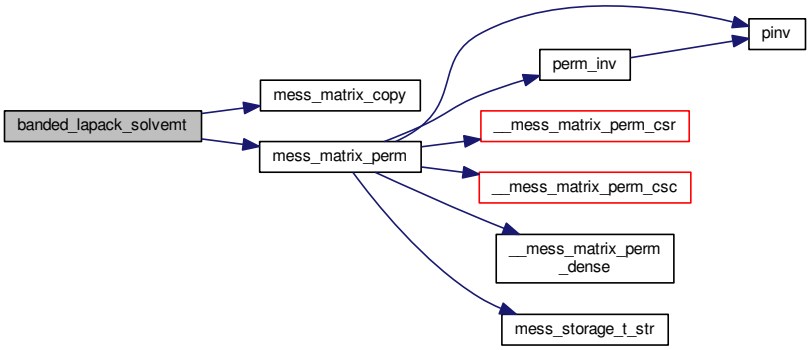
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The banded_lapack_solvemt function solves $A^T X = B$.

Here is the call graph for this function:



10.25.3.7 static int banded_lapack_solvet (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

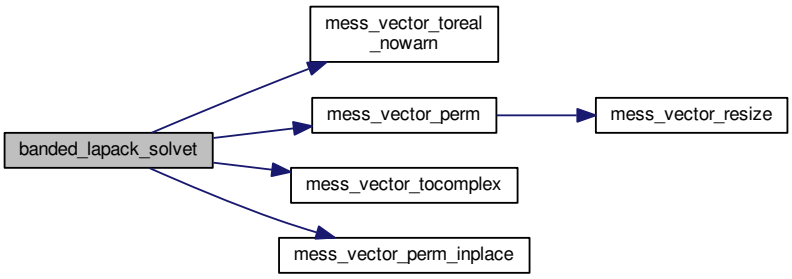
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The banded_lapack_solvet function solves $A^T x = b$.

Here is the call graph for this function:



10.26 bandwidth.c File Reference

Functions

- int mess_matrix_bandwidth (mess_matrix matrix, mess_int_t *kl, mess_int_t *ku)

  Determine the bandwidth of a matrix.

10.26.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.27 ber_sgn_fac.c File Reference

Functions

- int mess_lrcf_signber (mess_matrix A, mess_matrix B, mess_int_t *maxit, double *tol, mess_matrix Z, mess_sign_scale_t scale)

  Solve a standard Bernoulli equation $A^T X + XA - XB^T BX = 0$ *via sign function iteration.*
- int mess_lrcf_gsignber (mess_matrix A, mess_matrix E, mess_matrix B, mess_int_t *maxit, double *tol, mess_matrix Z, mess_sign_scale_t scale)

  Solve a generalized Bernoulli equation $A^T XE + E^T XA - E^T XB^T BXE = 0$ *via sign function iteration.*

10.27.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.28 bicgstab.c File Reference

Data Structures

- struct bicgstab_solver

Macros

- #define ILU_LEVEL 3

Functions

- static int bicgstab_clear (void *data)
- static int bicgstab_solve (void *data, mess_vector b, mess_vector x)
- static int bicgstab_solvem (void *data, mess_matrix B, mess_matrix X)
- int mess_direct_create_bicgstab (mess_matrix matrix, mess_direct solver)

  Generate a bicgstab solver in order to use it as direct solver.

10.28.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

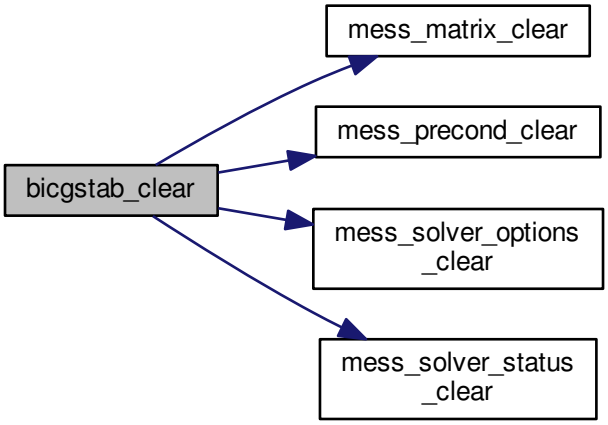
10.28.2 Macro Definition Documentation

10.28.2.1 #define ILU_LEVEL 3

10.28.3 Function Documentation

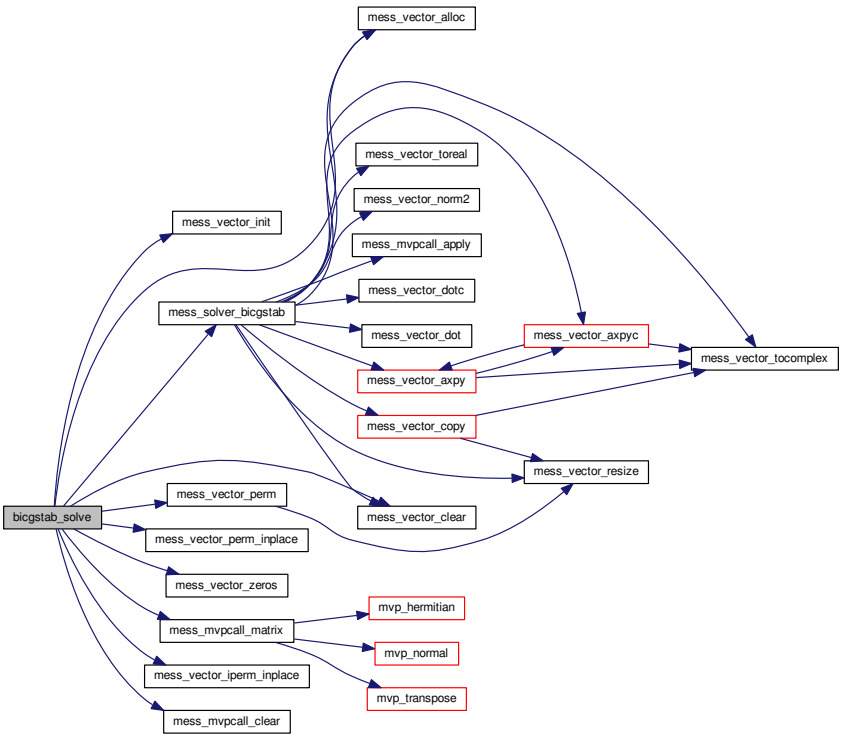
10.28.3.1 static int bicgstab_clear (void * *data*) [static]

Here is the call graph for this function:



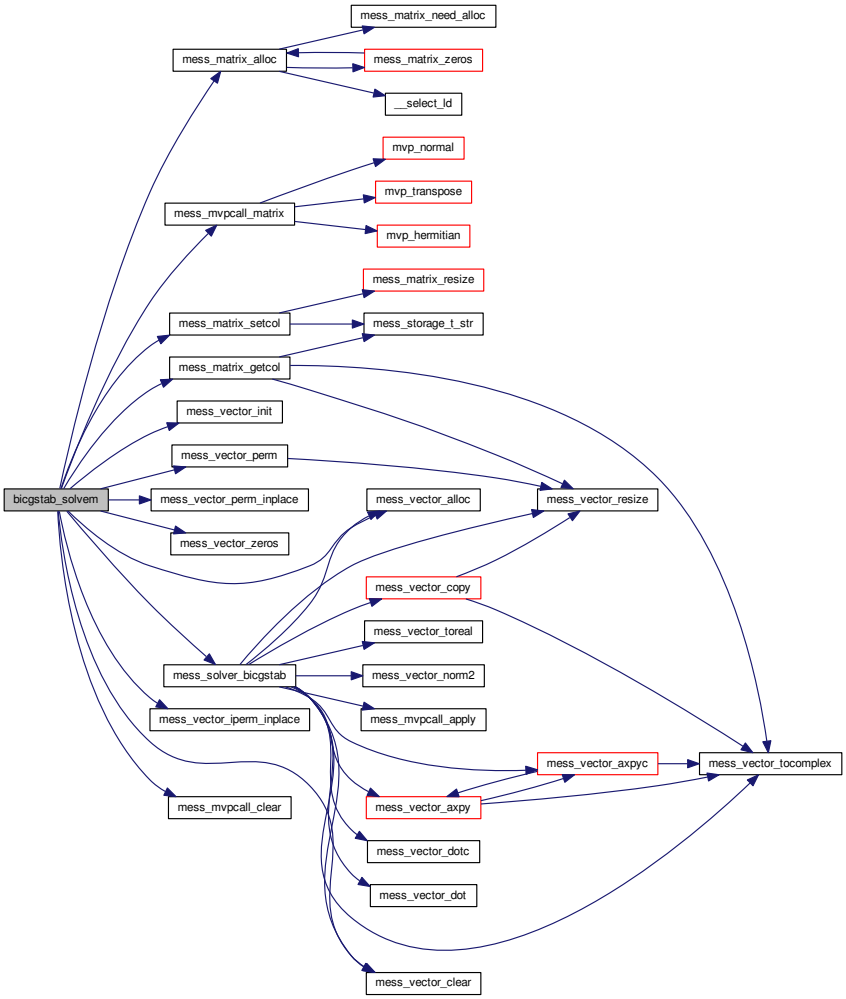
10.28.3.2 static int bicgstab_solve (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Here is the call graph for this function:



10.28.3.3 static int bicgstab_solvem (void * *data*, mess_matrix *B*, mess_matrix *X*) [static]

Here is the call graph for this function:



10.29 bicgstab.c File Reference

Functions

- int mess_solver_bicgstab (mess_mvpcall matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with BICGSTAB algorithm.

10.29.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.30 biorth.c File Reference

Functions

- int mess_matrix_biorth (mess_matrix Vin, mess_matrix Win, mess_matrix Vout, mess_matrix Wout)

  Construct two biorthonormal matrices from two given matrices.
- int mess_matrix_gbiorth (mess_matrix E, mess_matrix Vin, mess_matrix Win, mess_matrix Vout, mess_matrix Wout)

  Construct two biorthonormal matrices from two given matrices with respect to a weighted scalar product.

10.30.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.31 blas_defs.h File Reference

Typedefs

- typedef mess_int_t MESS_LAPACK_LOGICAL

  Datatype to represent Fortran Logical.

Functions

- mess_double_cpx_t F77_GLOBAL (zdotc, ZDOTC)(mess_int_t *N
- mess_double_cpx_t F77_GLOBAL (zdotu, ZDOTU)(mess_int_t *N
- double F77_GLOBAL (ddot, DDOT)(mess_int_t *N
- double F77_GLOBAL (dnrm2, DNRM2)(mess_int_t *N
- double F77_GLOBAL (dznrm2, DZNRM2)(mess_int_t *N
- void F77_GLOBAL (daxpy, DAXPY)(mess_int_t *N
- void F77_GLOBAL (dcopy, DCOPY)(mess_int_t *N
- void F77_GLOBAL (dgemm, DGEMM)(char *transA
- void F77_GLOBAL (dgemv, DGEMV)(char *trans
- void F77_GLOBAL (dger, DGER)(mess_int_t *M
- void F77_GLOBAL (dscal, DSCAL)(mess_int_t *N
- void F77_GLOBAL (dswap, DSWAP)(mess_int_t *N
- void F77_GLOBAL (dtrmm, DTRMM)(char *side
- void F77_GLOBAL (dtrsm, DTRSM)(char *side
- void F77_GLOBAL (dtrevc, DTREVC)(char *SIDE
- void F77_GLOBAL (zaxpy, ZAXPY)(mess_int_t *N
- void F77_GLOBAL (zcopy, ZCOPY)(mess_int_t *N
- void F77_GLOBAL (zgemm, ZGEMM)(char *transA
- void F77_GLOBAL (zgemv, ZGEMV)(char *trans
- void F77_GLOBAL (zgeru, ZGERU)(mess_int_t *M
- void F77_GLOBAL (zscal, ZSCAL)(mess_int_t *N
- void F77_GLOBAL (zswap, ZSWAP)(mess_int_t *N
- void F77_GLOBAL (ztrevc, ZTREVC)(char *SIDE
- void F77_GLOBAL (dgeadd, DGEADD)(mess_int_t *M
- void F77_GLOBAL (zgeadd, ZGEADD)(mess_int_t *M
- void F77_GLOBAL (dzgeadd, DZGEADD)(mess_int_t *M

- void F77_GLOBAL (dzgemmx, DZGEMMX)(char *transA
- void F77_GLOBAL (zdgemmx, ZDGEMMX)(char *transA
- mess_double_cpx_t F77_GLOBAL (dzdotu, DZDOTU)(mess_int_t *N
- mess_double_cpx_t F77_GLOBAL (zddotc, ZDDOTC)(mess_int_t *N
- mess_double_cpx_t F77_GLOBAL (zddotu, ZDDOTU)(mess_int_t *N
- double F77_GLOBAL (dlange, DLANGE)(char *NORM
- double F77_GLOBAL (zlange, ZLANGE)(char *NORM
- void F77_GLOBAL (dgbtrf, DGBTRF)(mess_int_t *M
- void F77_GLOBAL (dgbtrs, DGBTRS)(char *TRANS
- void F77_GLOBAL (dgees, DGEES)(char *JOBVS
- void F77_GLOBAL (dgeev, DGEEV)(const char *JOBVL
- void F77_GLOBAL (dgelqf, DGELQF)(mess_int_t *M
- void F77_GLOBAL (dgels, DGELS)(char *TRANS
- void F77_GLOBAL (dgeqp3, DGEQP3)(mess_int_t *M
- void F77_GLOBAL (dgeqrf, DGEQRF)(mess_int_t *M
- void F77_GLOBAL (dgesvd, DGESVD)(char *JOBU
- void F77_GLOBAL (dgetrf, DGETRF)(mess_int_t *M
- void F77_GLOBAL (dgetri, DGETRI)(mess_int_t *N
- void F77_GLOBAL (dgetrs, DGETRS)(char *TRANS
- void F77_GLOBAL (dgges, DGGES)(char *JOBVSL
- void F77_GLOBAL (dggev, DGGEV)(char *JOBVL
- void F77_GLOBAL (dlacpy, DLACPY)(char *uplo
- void F77_GLOBAL (dlapmt, DLAPMT)(mess_int_t *FORWARD
- void F77_GLOBAL (dlarnv, DLARNV)(mess_int_t *type
- void F77_GLOBAL (dhseqr, DHSEQR)(char *JOB
- void F77_GLOBAL (dhsein, DHSEIN)(char *SIDE
- void F77_GLOBAL (dorglq, DORGLQ)(mess_int_t *M
- void F77_GLOBAL (dorgqr, DORGQR)(mess_int_t *M
- void F77_GLOBAL (dormlq, DORMLQ)(char *SIDE
- void F77_GLOBAL (dormqr, DORMQR)(char *SIDE
- void F77_GLOBAL (dpotrf, DPOTRF)(const char *UPLO
- void F77_GLOBAL (dpotri, DPOTRI)(const char *UPLO
- void F77_GLOBAL (dpotrs, DPOTRS)(const char *uplo
- void F77_GLOBAL (dstev, DSTEV)(const char *JOBZ
- void F77_GLOBAL (dsyev, DSYEV)(char *JOBZ
- void F77_GLOBAL (dtgevc, DTGEVC)(char *side
- void F77_GLOBAL (dtgsyl, DTGSYL)(char *TRANS
- void F77_GLOBAL (dtrsyl, DTRSYL)(char *transA
- void F77_GLOBAL (dtrtrs, DTRTRS)(char *UPLO
- void F77_GLOBAL (dzgbtrs, DZGBTRS)(char *TRANS
- void F77_GLOBAL (dzgetrs, DZGETRS)(char *TRANS
- void F77_GLOBAL (dzpotrs, DZPOTRS)(const char *uplo
- void F77_GLOBAL (zgbtrf, ZGBTRF)(mess_int_t *M
- void F77_GLOBAL (zgbtrs, ZGBTRS)(char *TRANS
- void F77_GLOBAL (zgees, ZGEES)(char *JOBVS
- void F77_GLOBAL (zgeev, ZGEEV)(const char *JOBVL
- void F77_GLOBAL (zgelqf, ZGELQF)(mess_int_t *M
- void F77_GLOBAL (zgels, ZGELS)(char *TRANS
- void F77_GLOBAL (zgeqrf, ZGEQRF)(mess_int_t *M
- void F77_GLOBAL (zgesvd, ZGESVD)(char *JOBU
- void F77_GLOBAL (zgetrf, ZGETRF)(mess_int_t *M
- void F77_GLOBAL (zgetri, ZGETRI)(mess_int_t *N
- void F77_GLOBAL (zgetrs, ZGETRS)(char *TRANS
- void F77_GLOBAL (zgges, ZGGES)(char *JOBVSL
- void F77_GLOBAL (zggev, ZGGEV)(const char *JOBL

- void F77_GLOBAL (zhseqr, ZHSEQR)(char *JOB
- void F77_GLOBAL (zhsein, zHSEIN)(char *SIDE
- void F77_GLOBAL (zlacpy, ZLACPY)(char *uplo
- void F77_GLOBAL (zlapmt, ZLAPMT)(mess_int_t *FORWARD
- void F77_GLOBAL (zpotrf, ZPOTRF)(const char *UPLO
- void F77_GLOBAL (zpotri, ZPOTRI)(const char *UPLO
- void F77_GLOBAL (zpotrs, ZPOTRS)(const char *UPLO
- void F77_GLOBAL (ztrsm, ZTRSM)(char *SIDE
- void F77_GLOBAL (ztrsyl, ZTRSYL)(char *transA
- void F77_GLOBAL (ztgsyl, ZTGSYL)(char *TRANS
- void F77_GLOBAL (ztrtrs, ZTRTRS)(char *UPLO
- void F77_GLOBAL (zunglq, ZUNGLQ)(mess_int_t *M
- void F77_GLOBAL (zungqr, ZUNGQR)(mess_int_t *M
- void F77_GLOBAL (zunmlq, ZUNMLQ)(char *SIDE
- void F77_GLOBAL (zunmqr, ZUNMQR)(char *SIDE
- void F77_GLOBAL (dgglyp, DGGLYP)(char *DICO
- void F77_GLOBAL (dgelyp, DGELYP)(char *DICO

Variables

- mess_double_cpx_t mess_double_cpx_t * X
- mess_double_cpx_t mess_double_cpx_t mess_int_t * incx
- mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t * y
- mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t * incy
- void double * alpha
- void double double * x
- void char * transB
- void char mess_int_t * m
- void char mess_int_t mess_int_t * n
- void char mess_int_t mess_int_t mess_int_t * k
- void char mess_int_t mess_int_t mess_int_t double double * A
- void char mess_int_t mess_int_t mess_int_t double double mess_int_t * lda
- void char mess_int_t mess_int_t mess_int_t double double mess_int_t double * B
- void char mess_int_t mess_int_t mess_int_t double double mess_int_t double mess_int_t * ldb
- void char mess_int_t mess_int_t mess_int_t double double mess_int_t double mess_int_t double * beta
- void char mess_int_t mess_int_t mess_int_t double double mess_int_t double mess_int_t double double * C
- void char mess_int_t mess_int_t mess_int_t double double mess_int_t double mess_int_t double double mess_int_t * ldc
- void mess_int_t mess_int_t double double * a
- void mess_int_t * N
- void double * DX
- void double mess_int_t double * DY
- void char * uplo
- void char char * transA
- void char char char * diag
- void char * HOWMNY
- void char MESS_LAPACK_LOGICAL * SELECT
- void char MESS_LAPACK_LOGICAL mess_int_t double * T
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t * LDT
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double * VL
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t * LDVL
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t double * VR
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t double mess_int_t * LDVR

- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t double mess_int_t mess_int_t * MM
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t double mess_int_t mess_int_t mess_int_t * M
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t double mess_int_t mess_int_t mess_int_t double * WORK
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t double mess_int_t mess_int_t mess_int_t double mess_int_t * INFO
- void char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t mess_int_t mess_double_cpx_t double * RWORK
- void mess_int_t double double mess_int_t * LDA
- void mess_int_t double double mess_int_t double double mess_int_t * LDB
- mess_double_cpx_t double * ZX
- mess_double_cpx_t double mess_int_t * INCX
- mess_double_cpx_t double mess_int_t mess_double_cpx_t * ZY
- mess_double_cpx_t double mess_int_t mess_double_cpx_t mess_int_t * INCY
- void mess_int_t mess_int_t * KL
- void mess_int_t mess_int_t mess_int_t * KU
- void mess_int_t mess_int_t mess_int_t double * AB
- void mess_int_t mess_int_t mess_int_t double mess_int_t * LDAB
- void mess_int_t mess_int_t mess_int_t double mess_int_t mess_int_t * IPIV
- void mess_int_t mess_int_t mess_int_t mess_int_t * NRHS
- void char * SORT
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t * SDA
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t mess_int_t * SDIM
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t mess_int_t double * WR
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t mess_int_t double double * WI
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t mess_int_t double double double * VS
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t mess_int_t double double double mess_int_t * LVDS
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t mess_int_t double double double mess_int_t double * work
- void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t mess_int_t double double double mess_int_t double mess_int_t mess_int_t * BWORK
- void const char * JOBVR
- void const char mess_int_t double mess_int_t double double double mess_int_t double mess_int_t double mess_int_t * LWORK
- void mess_int_t double mess_int_t double * TAU
- void mess_int_t double mess_int_t mess_int_t * JPVT
- void mess_int_t double mess_int_t mess_int_t double double mess_int_t mess_int_t * info
- void char * JOBVT
- void char mess_int_t mess_int_t double mess_int_t double * S
- void char mess_int_t mess_int_t double mess_int_t double double * U
- void char mess_int_t mess_int_t double mess_int_t double double mess_int_t * LDU
- void char mess_int_t mess_int_t double mess_int_t double double mess_int_t double * VT
- void char mess_int_t mess_int_t double mess_int_t double double mess_int_t double mess_int_t * LDVT
- void mess_int_t double mess_int_t mess_int_t * ipiv
- void char * JOBVSR
- void char char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t mess_int_t double * ALPHAR
- void char char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t mess_int_t double double * ALPHAI
- void char char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t mess_int_t double double double * BETA
- void char char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t mess_int_t double double double double * VSL

- void char char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t mess_int_t double double double double mess_int_t * LDVSL
- void char char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t mess_int_t double double double double mess_int_t double * VSR
- void char char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double mess_int_t mess_int_t double double double double mess_int_t double mess_int_t * LDVSR
- void mess_int_t mess_int_t double mess_int_t * LDX
- void mess_int_t mess_int_t double mess_int_t mess_int_t * K
- void mess_int_t * seed
- void mess_int_t mess_int_t * n2
- void mess_int_t mess_int_t double * values
- void char * COMPZ
- void char mess_int_t mess_int_t * ILO
- void char mess_int_t mess_int_t mess_int_t * IHI
- void char mess_int_t mess_int_t mess_int_t double * H
- void char mess_int_t mess_int_t mess_int_t double mess_int_t * LDH
- void char mess_int_t mess_int_t mess_int_t double mess_int_t double double double * Z
- void char mess_int_t mess_int_t mess_int_t double mess_int_t double double double mess_int_t * LDZ
- void char * EIGSRC
- void char char * INITV
- void char char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double double double mess_int_t double mess_int_t mess_int_t mess_int_t double mess_int_t * IFAILL
- void char char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t double double double mess_int_t double mess_int_t mess_int_t mess_int_t double mess_int_t mess_int_t * IFAILR
- void char * TRANS
- void char mess_int_t mess_int_t mess_int_t double mess_int_t double double mess_int_t * LDC
- void char * Trans
- void mess_int_t mess_int_t * nrhs
- void mess_int_t mess_int_t double mess_int_t double * b
- void mess_int_t double * D
- void mess_int_t double double * E
- void char * UPLO
- void char mess_int_t double mess_int_t double * W
- void char * how
- void mess_int_t * ijob
- void mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t * LDD
- void mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t * LDE
- void mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double * F
- void mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t * LDF
- void mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double * SCALE
- void mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double double * DIFF
- void mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double mess_int_t * LDWORK
- void mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double mess_int_t mess_int_t * lwork
- void char char * TRANSA
- void char char char * UNIT
- void char mess_int_t * ISGN
- void char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double * scale

- void char * trans
- void char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t * LDVS
- void mess_double_cpx_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t * lwork
- void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t double * rwork
- void const char * JOBR
- void char * JOB
- void char char * FACT
- void char char char char mess_int_t * ISOLVE
- void char char char char mess_int_t mess_int_t * NB
- void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double * Q
- void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t * LDQ
- void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double double * sep
- void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double * ferr
- void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double double double double mess_int_t * IWORK
- void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double double double double mess_int_t double * DWORK
- void char char char mess_int_t double mess_int_t double mess_int_t double mess_int_t double double * SEP
- void char char char mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double * FERR

10.31.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.31.2 Typedef Documentation

10.31.2.1 typedef mess_int_t MESS_LAPACK_LOGICAL

The MESS_LAPACK_LOGICAL represents the `LOGICAL` fortran type.

10.31.3 Function Documentation

10.31.3.1 mess_double_cpx_t F77_GLOBAL (zdotc , ZDOTC)

10.31.3.2 mess_double_cpx_t F77_GLOBAL (zdotu , ZDOTU)

10.31.3.3 double F77_GLOBAL (ddot , DDOT)

10.31.3.4 double F77_GLOBAL (dnrm2 , DNRM2)

10.31.3.5 double F77_GLOBAL (dznrm2 , DZNRM2)

10.31.3.6 void F77_GLOBAL (daxpy , DAXPY)

10.31.3.7 void F77_GLOBAL (dcopy , DCOPY)

10.31.3.8 void F77_GLOBAL (dgemm , DGEMM)

10.31.3.9 void F77_GLOBAL (dgemv , DGEMV)

10.31.3.10 void F77_GLOBAL (dger , DGER)

10.31.3.11 void F77_GLOBAL (dscal , DSCAL)

10.31.3.12 void F77_GLOBAL (dswap , DSWAP)

10.31.3.13 void F77_GLOBAL (dtrmm , DTRMM)

10.31.3.14 void F77_GLOBAL (dtrsm , DTRSM)

10.31.3.15 void F77_GLOBAL (dtrevc , DTREVC)

10.31.3.16 void F77_GLOBAL (zaxpy , ZAXPY)

10.31.3.17 void F77_GLOBAL (zcopy , ZCOPY)

10.31.3.18 void F77_GLOBAL (zgemm , ZGEMM)

10.31.3.19 void F77_GLOBAL (zgemv , ZGEMV)

10.31.3.20 void F77_GLOBAL (zgeru , ZGERU)

10.31.3.21 void F77_GLOBAL (zscal , ZSCAL)

10.31.3.22 void F77_GLOBAL (zswap , ZSWAP)

10.31.3.23 void F77_GLOBAL (ztrevc , ZTREVC)

10.31.3.24 void F77_GLOBAL (dgeadd , DGEADD)

10.31.3.25 void F77_GLOBAL (zgeadd , ZGEADD)

10.31.3.26 void F77_GLOBAL (dzgeadd , DZGEADD)

10.31.3.27 void F77_GLOBAL (dzgemmx , DZGEMMX)

10.31.3.28 void F77_GLOBAL (zdgemmx , ZDGEMMX)

10.31.3.29 mess_double_cpx_t F77_GLOBAL (dzdotu , DZDOTU)

10.31.3.30 mess_double_cpx_t F77_GLOBAL (zddotc , ZDDOTC)

10.31.3.31 mess_double_cpx_t F77_GLOBAL (zddotu , ZDDOTU)

10.31.3.32 double F77_GLOBAL (dlange , DLANGE)

10.31.3.33 double F77_GLOBAL (zlange , ZLANGE)

10.31.3.34 void F77_GLOBAL (dgbtrf , DGBTRF)

10.31.3.35 void F77_GLOBAL (dgbtrs , DGBTRS)

10.31.3.36 void F77_GLOBAL (dgees , DGEES)

10.31.3.37 void F77_GLOBAL (dgeev , DGEEV) const

10.31.3.38 void F77_GLOBAL (dgelqf , DGELQF)

10.31.3.39 void F77_GLOBAL (dgels , DGELS)

10.31.3.40 void F77_GLOBAL (dgeqp3 , DGEQP3)

10.31.3.41 void F77_GLOBAL (dgeqrf , DGEQRF)

10.31.3.42 void F77_GLOBAL (dgesvd , DGESVD)

10.31.3.43 void F77_GLOBAL (dgetrf , DGETRF)

10.31.3.44 void F77_GLOBAL (dgetri , DGETRI)

10.31.3.45 void F77_GLOBAL (dgetrs , DGETRS)

10.31.3.46 void F77_GLOBAL (dgges , DGGES)

10.31.3.47 void F77_GLOBAL (dggev , DGGEV)

10.31.3.48 void F77_GLOBAL (dlacpy , DLACPY)

10.31.3.49 void F77_GLOBAL (dlapmt , DLAPMT)

10.31.3.50 void F77_GLOBAL (dlarnv , DLARNV)

10.31.3.51 void F77_GLOBAL (dhseqr , DHSEQR)

10.31.3.52 void F77_GLOBAL (dhsein , DHSEIN)

10.31.3.53 void F77_GLOBAL (dorglq , DORGLQ)

10.31.3.54 void F77_GLOBAL (dorgqr , DORGQR)

10.31.3.55 void F77_GLOBAL (dormlq , DORMLQ)

10.31.3.56 void F77_GLOBAL (dormqr , DORMQR)

10.31.3.57 void F77_GLOBAL (dpotrf , DPOTRF) const

10.31.3.58 void F77_GLOBAL (dpotri , DPOTRI) const

10.31.3.59 void F77_GLOBAL (dpotrs , DPOTRS) const

10.31.3.60 void F77_GLOBAL (dstev , DSTEV) const

10.31.3.61 void F77_GLOBAL (dsyev , DSYEV)

10.31.3.62 void F77_GLOBAL (dtgevc , DTGEVC)

10.31.3.63 void F77_GLOBAL (dtgsyl , DTGSYL)

10.31.3.64 void F77_GLOBAL (dtrsyl , DTRSYL)

10.31.3.65 void F77_GLOBAL (dtrtrs , DTRTRS)

10.31.3.66 void F77_GLOBAL (dzgbtrs , DZGBTRS)

10.31.3.67 void F77_GLOBAL (dzgetrs , DZGETRS)

10.31.3.68 void F77_GLOBAL (dzpotrs , DZPOTRS) const

10.31.3.69 void F77_GLOBAL (zgbtrf , ZGBTRF)

10.31.3.70 void F77_GLOBAL (zgbtrs , ZGBTRS)

10.31.3.71 void F77_GLOBAL (zgees , ZGEES)

10.31.3.72 void F77_GLOBAL (zgeev , ZGEEV) const

10.31.3.73 void F77_GLOBAL (zgelqf , ZGELQF)

10.31.3.74 void F77_GLOBAL (zgels , ZGELS)

10.31.3.75 void F77_GLOBAL (zgeqrf , ZGEQRF)

10.31.3.76 void F77_GLOBAL (zgesvd , ZGESVD)

10.31.3.77 void F77_GLOBAL (zgetrf , ZGETRF)

10.31.3.78 void F77_GLOBAL (zgetri , ZGETRI)

10.31.3.79 void F77_GLOBAL (zgetrs , ZGETRS)

10.31.3.80 void F77_GLOBAL (zgges , ZGGES)

10.31.3.81 void F77_GLOBAL (zggev , ZGGEV) const

10.31.3.82 void F77_GLOBAL (zhseqr , ZHSEQR)

10.31.3.83 void F77_GLOBAL (zhsein , zHSEIN)

10.31.3.84 void F77_GLOBAL (zlacpy , ZLACPY)

10.31.3.85 void F77_GLOBAL (zlapmt , ZLAPMT)

10.31.3.86 void F77_GLOBAL (zpotrf , ZPOTRF) const

10.31.3.87 void F77_GLOBAL (zpotri , ZPOTRI) const

10.31.3.88 void F77_GLOBAL (zpotrs , ZPOTRS) const

10.31.3.89 void F77_GLOBAL (ztrsm , ZTRSM)

10.31.3.90 void F77_GLOBAL (ztrsyl , ZTRSYL)

10.31.3.91 void F77_GLOBAL (ztgsyl , ZTGSYL)

10.31.3.92 void F77_GLOBAL (ztrtrs , ZTRTRS)

10.31.3.93 void F77_GLOBAL (zunglq , ZUNGLQ)

10.31.3.94 void F77_GLOBAL (zungqr , ZUNGQR)

10.31.3.95 void F77_GLOBAL (zunmlq , ZUNMLQ)

10.31.3.96 void F77_GLOBAL (zunmqr , ZUNMQR)

10.31.3.97 void F77_GLOBAL (dgglyp , DGGLYP)

10.31.3.98 void F77_GLOBAL (dgelyp , DGELYP)

10.31.4 Variable Documentation

10.31.4.1 void char char char mess_int_t double * A

10.31.4.2 void mess_int_t mess_int_t double * a

10.31.4.3 void mess_int_t mess_int_t mess_int_t mess_int_t mess_double_cpx_t * AB

10.31.4.4 void char char char mess_int_t mess_int_t mess_double_cpx_t * alpha

10.31.4.5 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double double double * ALPHAI

10.31.4.6 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double double * ALPHAR

10.31.4.7 void char char mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t * B

10.31.4.8 void mess_int_t mess_int_t double mess_int_t mess_double_cpx_t * b

10.31.4.9 void const char mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t * beta

10.31.4.10 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double double double double * BETA

10.31.4.11 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t double mess_int_t * BWORK

10.31.4.12 void char char char mess_int_t double mess_int_t double mess_int_t double * C

10.31.4.13 void char * COMPZ

10.31.4.14 void mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t * D

10.31.4.15 void char char * diag

10.31.4.16 void mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t double double * DIFF

10.31.4.17 void char char char mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double double double double mess_int_t double * DWORK

10.31.4.18 void mess_double_cpx_t * DX

10.31.4.19 void mess_double_cpx_t mess_int_t mess_double_cpx_t * DY

10.31.4.20 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double * E

10.31.4.21 void char * EIGSRC

10.31.4.22 void mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t * F

10.31.4.23 void char char * FACT

10.31.4.24 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double∗ ferr

10.31.4.25 void char char char mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double∗ FERR

10.31.4.26 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t ∗ H

10.31.4.27 void char∗ how

10.31.4.28 void char ∗ HOWMNY

10.31.4.29 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t ∗ IFAILL

10.31.4.30 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t ∗ IFAILR

10.31.4.31 void char mess_int_t mess_int_t mess_int_t ∗ IHI

10.31.4.32 void mess_int_t ∗ ijob

10.31.4.33 void char mess_int_t mess_int_t ∗ ILO

10.31.4.34 void mess_double_cpx_t mess_int_t ∗ incx

10.31.4.35 mess_double_cpx_t mess_double_cpx_t mess_int_t ∗ INCX

10.31.4.36 void mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t ∗ incy

10.31.4.37 mess_double_cpx_t mess_double_cpx_t mess_int_t double mess_int_t ∗ INCY

10.31.4.38 void char char char mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double double double double mess_int_t double mess_int_t mess_int_t ∗ INFO

10.31.4.39 void mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t double double mess_double_cpx_t mess_int_t mess_int_t mess_int_t ∗ info

10.31.4.40 void char char ∗ INITV

10.31.4.41 void mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t ∗ IPIV

10.31.4.42 void mess_int_t mess_double_cpx_t mess_int_t mess_int_t ∗ ipiv

10.31.4.43 void char mess_int_t ∗ ISGN

10.31.4.44 void char char char char mess_int_t∗ ISOLVE

10.31.4.45 void mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t double double mess_double_cpx_t mess_int_t mess_int_t * lwork

10.31.4.46 void char char char mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double double double mess_int_t * IWORK

10.31.4.47 void char * JOB

10.31.4.48 void const char* JOBR

10.31.4.49 void const char * JOBVR

10.31.4.50 void char * JOBVSR

10.31.4.51 void char * JOBVT

10.31.4.52 void mess_int_t double mess_int_t mess_int_t* JPVT

10.31.4.53 void char mess_int_t mess_int_t mess_int_t * k

10.31.4.54 void char mess_int_t mess_int_t mess_int_t * K

10.31.4.55 void mess_int_t mess_int_t * KL

10.31.4.56 void mess_int_t mess_int_t mess_int_t * KU

10.31.4.57 void mess_int_t mess_int_t double mess_int_t * lda

10.31.4.58 void char char char mess_int_t double mess_int_t * LDA

10.31.4.59 void mess_int_t mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t * LDAB

10.31.4.60 void char char char mess_int_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t * ldb

10.31.4.61 void char char mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t * LDB

10.31.4.62 void char mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t double mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t * ldc

10.31.4.63 void char char char mess_int_t double mess_int_t double mess_int_t double mess_int_t * LDC

10.31.4.64 void mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t * LDD

10.31.4.65 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t * LDE

10.31.4.66 void mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t * LDF

10.31.4.67 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t * LDH

10.31.4.68 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t* LDQ

10.31.4.69 void char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t * LDT

10.31.4.70 void char char char mess_int_t double mess_int_t double mess_int_t * LDU

10.31.4.71 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t * LDVL

10.31.4.72 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t * LDVR

10.31.4.73 void char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t* LDVS

10.31.4.74 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_double_cpx_t mess_double_cpx_t mess_int_t * LDVSL

10.31.4.75 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t * LDVSR

10.31.4.76 void char mess_int_t mess_int_t mess_double_cpx_t mess_int_t double mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t * LDVT

10.31.4.77 void char char char mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double double double mess_int_t double mess_int_t * LDWORK

10.31.4.78 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t * LDX

10.31.4.79 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t * LDZ

10.31.4.80 void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t mess_int_t double double double mess_int_t* LVDS

10.31.4.81 void char mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t * LWORK

10.31.4.82 void char mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t * lwork

10.31.4.83 void char mess_int_t* m

10.31.4.84 void char mess_int_t * M

10.31.4.85 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t * MM

10.31.4.86 void mess_int_t * n

10.31.4.87 void char char char mess_int_t * N

10.31.4.88 void mess_int_t mess_int_t* n2

10.31.4.89 void char char char char mess_int_t mess_int_t* NB

10.31.4.90 void char char mess_int_t mess_int_t * NRHS

10.31.4.91 void mess_int_t mess_int_t * nrhs

10.31.4.92 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double* Q

10.31.4.93 void char mess_int_t mess_int_t mess_double_cpx_t mess_int_t double mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t double * RWORK

10.31.4.94 void const char mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t double * rwork

10.31.4.95 void char mess_int_t mess_int_t mess_double_cpx_t mess_int_t double * S

10.31.4.96 void char char char mess_int_t double mess_int_t double mess_int_t double mess_int_t double * SCALE

10.31.4.97 void char mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t double * scale

10.31.4.98 void char MESS_LAPACK_LOGICAL mess_int_t double mess_int_t* SDA

10.31.4.99 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t * SDIM

10.31.4.100 void mess_int_t* seed

10.31.4.101 void char char MESS_LAPACK_LOGICAL * SELECT

10.31.4.102 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double double* sep

10.31.4.103 void char char char mess_int_t double mess_int_t double mess_int_t double mess_int_t double double* SEP

10.31.4.104 void char char * SORT

10.31.4.105 void char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t * T

10.31.4.106 void char mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t * TAU

10.31.4.107 void char char char * TRANS

10.31.4.108 void char * Trans

10.31.4.109 void char* trans

10.31.4.110 void char char * transA

10.31.4.111 void char char char * TRANSA

10.31.4.112 void char * transB

10.31.4.113 void char char char mess_int_t double mess_int_t double * U

10.31.4.114 void char char char * UNIT

10.31.4.115 void char * uplo

10.31.4.116 void char char char char * UPLO

10.31.4.117 void mess_int_t mess_int_t double* values

10.31.4.118 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t * VL

10.31.4.119 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t * VR

10.31.4.120 void char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_int_t mess_double_cpx_t mess_double_cpx_t * VS

10.31.4.121 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_double_cpx_t * VSL

10.31.4.122 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_int_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t * VSR

10.31.4.123 void char mess_int_t mess_int_t mess_double_cpx_t mess_int_t double mess_double_cpx_t mess_int_t mess_double_cpx_t * VT

10.31.4.124 void char char MESS_LAPACK_LOGICAL mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t * W

10.31.4.125 void char char char mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double double double double ∗ WI

10.31.4.126 void char mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t ∗ WORK

10.31.4.127 void char mess_int_t mess_int_t mess_int_t mess_double_cpx_t mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t ∗ work

10.31.4.128 void char char char mess_int_t double mess_int_t double mess_int_t double mess_int_t double double double double double ∗ WR

10.31.4.129 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double mess_int_t double ∗ X

10.31.4.130 void mess_int_t mess_int_t mess_double_cpx_t ∗ x

10.31.4.131 void mess_int_t mess_double_cpx_t mess_double_cpx_t mess_int_t mess_double_cpx_t ∗ y

10.31.4.132 void char char char char mess_int_t mess_int_t mess_int_t double mess_int_t double mess_int_t double mess_int_t double ∗ Z

10.31.4.133 mess_double_cpx_t mess_double_cpx_t ∗ ZX

10.31.4.134 mess_double_cpx_t mess_double_cpx_t mess_int_t double ∗ ZY

10.32 blockpart.c File Reference

Functions

- static int lblock_dfs (mess_matrix G, mess_int_t startnode, mess_int_t ∗stack, mess_int_t ∗marked, mess_int_t marker, mess_int_t ∗sT)

  Depth-First-Search on a lower triangular matrix.
- static void swap (mess_int_t ∗V, mess_int_t i, mess_int_t j)

  Inline function to swap to elements in an integer array.
- int mess_graph_lblockpart (mess_matrix G, mess_int_t ∗perm, mess_int_t ∗nblocks, mess_int_t ∗blocks)

  Compute a lower triangular block partition of a matrix.

10.32.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.32.2 Function Documentation

10.32.2.1 static int lblock_dfs (mess_matrix *G*, mess_int_t *startnode*, mess_int_t ∗ *stack*, mess_int_t ∗ *marked*, mess_int_t *marker*, mess_int_t ∗ *sT*) [static]

Parameters

in	*G*	input matrix representing the directed graph
in	*startnode*	input start node for the DFS
in,out	*stack*	recursion stack to work on
in,out	*marked*	array with the node markers
in	*marker*	input value to mark nodes
out	*sT*	top of the stack where the visited nodes are

Returns

0 on success or a non zero error code otherwise

The lblock_dfs function performs a Depth-First-Search of a graph beginning at the startnode.
The result is returned as a part of the $stack$ array. The visited nodes are contained in $stack[st, ..., G->rows-1]$.

10.32.2.2 static void swap (mess_int_t * *V*, mess_int_t *i*, mess_int_t *j*) [inline],[static]

Parameters

in,out	*V*	input/output array
in	*i*	input index one
in	*j*	input index two

The swap function swaps $V[i] \leftrightarrow V[j]$.

10.33 call_macro.h File Reference

Macros

- #define CALL(X) ret = (X) ; if (ret !=0) { fprintf(stderr,"%s: %d : %s returned with error: %d - %s\n", __FIL↩E__,__LINE__, #X,ret, mess_get_error(ret)); return 1; }

10.33.1 Macro Definition Documentation

10.33.1.1 #define CALL(X) ret = (X) ; if (ret !=0) { fprintf(stderr,"%s: %d : %s returned with error: %d - %s\n", __FILE__,__LINE__, #X,ret, mess_get_error(ret)); return 1; }

10.34 callback_equation.c File Reference

Data Structures

- struct __matlab_equation

Macros

- #define GENERATE_FUNCTION(name, component_extern, component_intern)
- #define CLEAR_FUNCTION(name, component_extern, component_intern)
- #define APPLY_FUNCTION(name, component_extern, component_intern)
- #define GENERATE_FUNCTION2(name, component_extern, component_intern)
- #define APPLY_FUNCTION2(name, component_extern, component_intern)

Functions

- static int clear (mess_equation e)
- static int AX_generate (mess_equation eqn)
- static int AINV_generate (mess_equation eqn)
- static int EX_generate (mess_equation eqn)
- static int EINV_generate (mess_equation eqn)
- static int AX_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int EX_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int AINV_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int EINV_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int AX_clear (mess_equation eqn)
- static int AINV_clear (mess_equation eqn)
- static int EX_clear (mess_equation eqn)
- static int EINV_clear (mess_equation eqn)
- static int ApEX_generate (mess_equation eqn, mess_vector p)
- static int ApEINV_generate (mess_equation eqn, mess_vector p)
- static int ApEX_apply (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)
- static int ApEINV_apply (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)
- static int ApEX_clear (mess_equation eqn)
- static int ApEINV_clear (mess_equation eqn)
- static int parameter (mess_equation eqn, mess_options opt, mess_status stat)
- int mess_callback_equation_mexmess (mess_equation eqn, mess_options opt, const mxArray *meqn)

10.34.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.34.2 Macro Definition Documentation

10.34.2.1 #define APPLY_FUNCTION(*name, component_extern, component_intern*)

10.34.2.2 #define APPLY_FUNCTION2(*name, component_extern, component_intern*)

10.34.2.3 #define CLEAR_FUNCTION(*name, component_extern, component_intern*)

Value:

```
static int name ( mess_equation eqn) { \
    MSG_FNAME(__func__); \
    mess_check_nullpointer(eqn); \
    mxArray *prhs[2];  \
    __matlab_equation *data = (__matlab_equation *) eqn->aux; \
    int ret = 0; \
    mess_check_nullpointer(data); \
    if ( eqn->component_extern.to_clear == 1  ) { return 0; } \
    eqn->component_extern.to_clear = 1; \
    prhs[0] = mxCreateString(#name); \
    prhs[1] = data->data; \
    ret = mexCallMATLAB(0, NULL, 2, prhs, "feval"); \
    if ( ret != 0 ) {\
        csc_error_message("Exection of " #name "failed.\n"); \
        return -1; \
    }\
    return 0; \
}
```

10.34.2.4 #define GENERATE_FUNCTION(*name, component_extern, component_intern*)

Value:

```
static int name ( mess_equation eqn) { \
    MSG_FNAME(__func__); \
    mess_check_nullpointer(eqn); \
    mxArray *prhs[3];  \
    __matlab_equation *data = (__matlab_equation *) eqn->aux; \
    int ret = 0; \
    mess_check_nullpointer(data); \
    if ( eqn->component_extern.to_clear == 1  ) { return 0; } \
    eqn->component_extern.to_clear = 1; \
    prhs[0] = mxCreateString(#name); \
    prhs[1] = data->data; \
    prhs[2] = data->options;\
    ret = mexCallMATLAB(0, NULL, 3, prhs, "feval"); \
    if ( ret != 0 ) {\
        csc_error_message("Exection of " #name "failed.\n"); \
        return -1; \
    }\
    return 0; \
}
```

10.34.2.5 #define GENERATE_FUNCTION2(*name, component_extern, component_intern*)

Value:

```
static int name ( mess_equation eqn, mess_vector p) { \
    MSG_FNAME(__func__); \
    mess_check_nullpointer(eqn); \
    mxArray *prhs[4];  \
    __matlab_equation *data = (__matlab_equation *) eqn->aux; \
    int ret = 0; \
    mess_check_nullpointer(data); \
    if ( eqn->component_extern.to_clear == 1  ) { return 0; } \
    eqn->component_extern.to_clear = 1; \
    prhs[0] = mxCreateString(#name); \
    prhs[1] = data->data; \
    prhs[2] = data->options;\
    prhs[3] = mess_vector_to_mexmess(p);\
    ret = mexCallMATLAB(0, NULL, 4, prhs, "feval"); \
    mxDestroyArray(prhs[3]); \
    if ( ret != 0 ) {\
        csc_error_message("Exection of " #name "failed.\n"); \
        return -1; \
    }\
    return 0; \
}
```

10.34.3 Function Documentation

10.34.3.1 static int AINV_apply (mess_equation *eqn,* mess_operation_t *op,* mess_matrix *in,* mess_matrix *out*) [static]

10.34.3.2 static int AINV_clear (mess_equation *eqn*) [static]

10.34.3.3 static int AINV_generate (mess_equation *eqn*) [static]

10.34.3.4 static int ApEINV_apply (mess_equation *eqn,* mess_operation_t *op,* mess_double_cpx_t *p,* mess_int_t *idx_p,* mess_matrix *in,* mess_matrix *out*) [static]

10.34.3.5 static int ApEINV_clear (mess_equation *eqn*) [static]

10.34.3.6 static int ApEINV_generate (mess_equation *eqn,* mess_vector *p*) [static]

10.34.3.7 static int ApEX_apply (mess_equation *eqn,* mess_operation_t *op,* mess_double_cpx_t *p,* mess_int_t *idx_p,* mess_matrix *in,* mess_matrix *out*) [static]

10.34.3.8 static int ApEX_clear (mess_equation *eqn*) [static]

10.34.3.9 static int ApEX_generate (mess_equation *eqn,* mess_vector *p*) [static]

10.34.3.10 static int AX_apply (mess_equation *eqn,* mess_operation_t *op,* mess_matrix *in,* mess_matrix *out*) [static]

10.34.3.11 static int AX_clear (mess_equation *eqn*) [static]

10.34.3.12 static int AX_generate (mess_equation *eqn*) [static]

10.34.3.13 static int clear (mess_equation *e*) [static]

10.34.3.14 static int EINV_apply (mess_equation *eqn,* mess_operation_t *op,* mess_matrix *in,* mess_matrix *out*) [static]

10.34.3.15 static int EINV_clear (mess_equation *eqn*) [static]

10.34.3.16 static int EINV_generate (mess_equation *eqn*) [static]

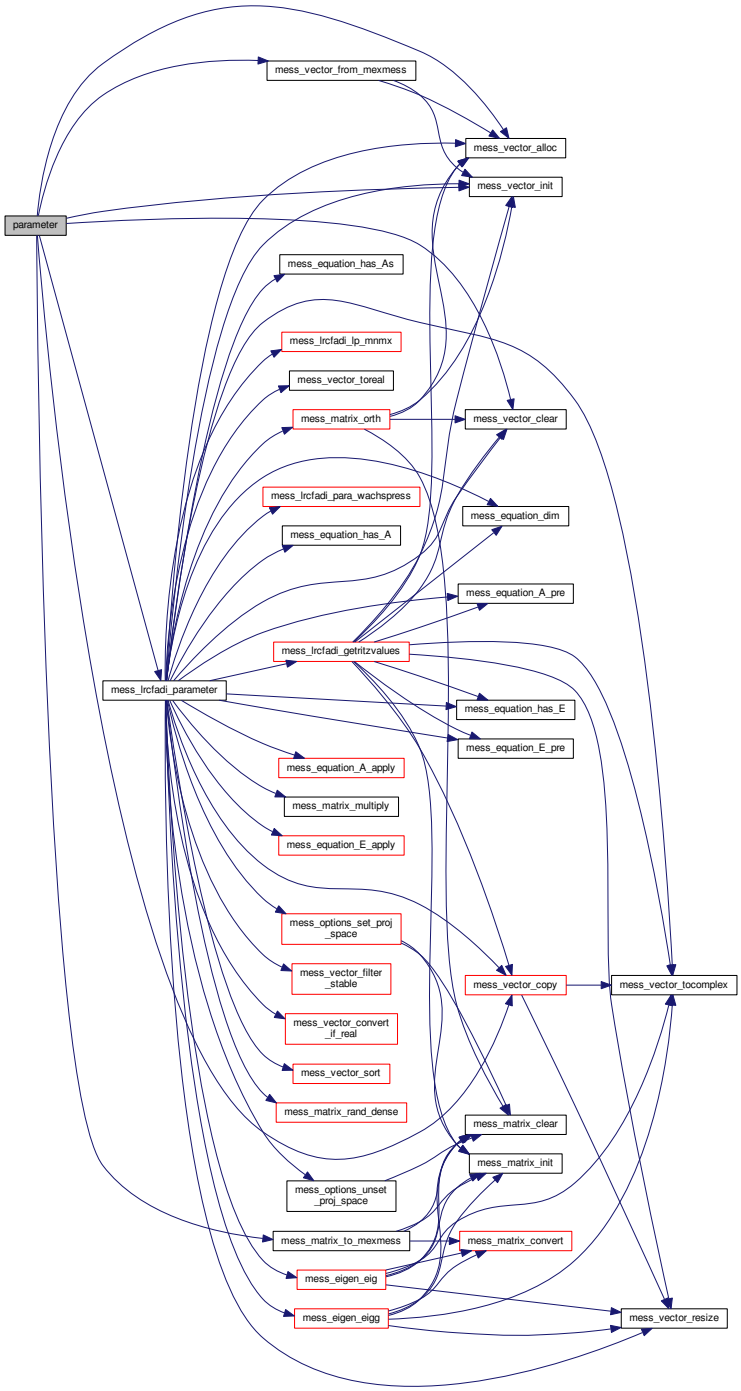
10.34.3.17 static int EX_apply (mess_equation *eqn,* mess_operation_t *op,* mess_matrix *in,* mess_matrix *out*) [static]

10.34.3.18 static int EX_clear (mess_equation *eqn*) [static]

10.34.3.19 static int EX_generate (mess_equation *eqn*) [static]

10.34.3.20 static int parameter (mess_equation *eqn*, mess_options *opt*, mess_status *stat*) [static]

Here is the call graph for this function:



10.35 care.c File Reference

Functions

- int mess_direct_care (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X)

Compute the solution of a (generalized) Riccati Equation using a dense newton method.

- int mess_direct_care_res2 (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X, double *res2)

  Compute the 2-norm residual of the algebraic (generalized) Riccati equation.

10.35.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.36 care.c File Reference

Functions

- int mess_care (mess_matrix A, mess_matrix E, mess_matrix B, mess_matrix C, mess_matrix Z)

  Frontend to compute a factorized solution of a (generalized) Riccati Equation.

10.37 care.c File Reference

Functions

- PyObject * pymess_care (PyObject *self, PyObject *args)

  Wrapper around the easy to use interface mess_care for use with Python.

10.37.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Nitin Prasad (nitin.prasad93@gmail.com, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.38 cat.c File Reference

Functions

- static int __mess_matrix_cat_diag (mess_matrix A, mess_matrix D, unsigned short otype, mess_matrix O)

  Concatenate two matrices in diagonal.
- static int __mess_matrix_cat_antidiag (mess_matrix B, mess_matrix C, unsigned short otype, mess_matrix O)

  Concatenate two matrices on the anti diagonal.
- static int __mess_matrix_cat_firstcol (mess_matrix A, mess_matrix C, unsigned short otype, mess_matrix O)

  Concatenate two matrices in a block column.
- static int __mess_matrix_cat_firstrow (mess_matrix A, mess_matrix B, unsigned short otype, mess_matrix O)

  Concatenates two matrices in the first block row.

- static int __mess_matrix_cat_full (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix D, unsigned short otype, mess_matrix O)

  Concatenates four matrices.
- static int __mess_matrix_cat_LABC (mess_matrix A, mess_matrix B, mess_matrix C, unsigned short otype, mess_matrix O)

  Concatenate three matrices as upper left triangle.
- static int __mess_matrix_cat_RBCD (mess_matrix B, mess_matrix C, mess_matrix D, unsigned short otype, mess_matrix O)

  Concatenate three matrices as lower right triangle.
- static int __mess_matrix_cat_RABD (mess_matrix A, mess_matrix B, mess_matrix D, unsigned short otype, mess_matrix O)

  Concatenate three matrices as upper right triangle.
- static int __mess_matrix_cat_LACD (mess_matrix A, mess_matrix C, mess_matrix D, unsigned short otype, mess_matrix O)

  Concatenate three matrices as lower left triangle.
- int mess_matrix_cat (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix D, mess_storage_t otype, mess_matrix O)

  Concatenate up to four matrices.

10.38.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.38.2 Function Documentation

10.38.2.1 static int __mess_matrix_cat_antidiag (mess_matrix *B*, mess_matrix *C*, unsigned short *otype*, mess_matrix *O*) [static]

Parameters

in	*B*	input upper right matrix
in	*C*	input lower left matrix
in	*otype*	input output storage type
out	*O*	output matrix

Returns

zero on success or a non-zero error value otherwise

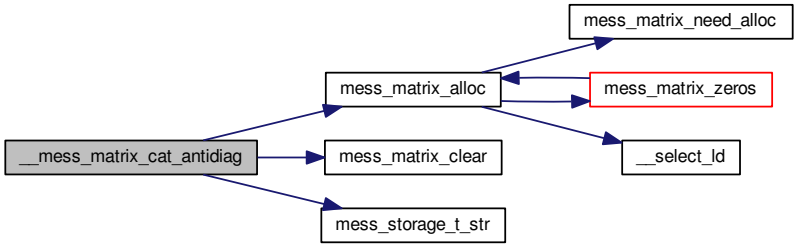
The __mess_matrix_cat_antidiag function concatenates two matrices in the following way:

```
O = [0, B; C, 0];
```

Attention

Internal use only.

Here is the call graph for this function:



10.38.2.2 static int __mess_matrix_cat_diag (mess_matrix *A*, mess_matrix *D*, unsigned short *otype*, mess_matrix *O*) [static]

Parameters

in	*A*	input upper left matrix
in	*D*	input lower right matrix
in	*otype*	input output storage type
out	*O*	input output matrix

Returns

zero on success or a non-zero error value otherwise

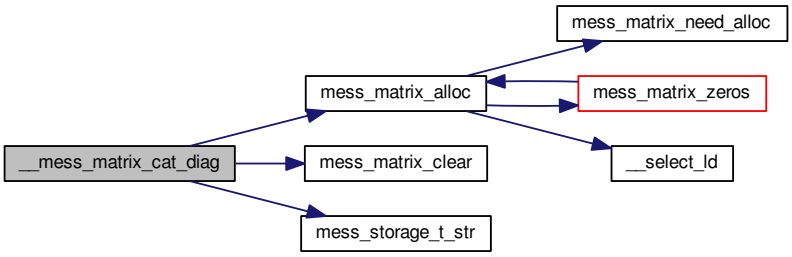
The __mess_matrix_cat_diag function concatenates two matrices in the following way:

```
O = [A, 0; 0, D];
```

Attention

Internal use only.

Here is the call graph for this function:



10.38.2.3 static int __mess_matrix_cat_firstcol (mess_matrix *A*, mess_matrix *C*, unsigned short *otype*, mess_matrix *O*) [static]

Parameters

in	*A*	input upper left matrix
in	*C*	input lower left matrix
in	*otype*	input output storage type
out	*O*	output matrix

Returns

zero on success or a non-zero error value otherwise

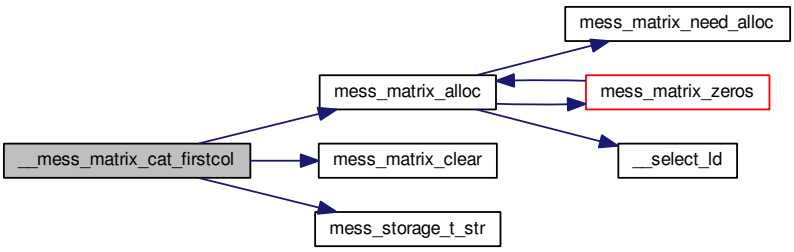
The __mess_matrix_cat_firstcol function concatenates two matrices in the following way:

```
O = [A; C];
```

Attention

Internal use only.

Here is the call graph for this function:



10.38.2.4 static int __mess_matrix_cat_firstrow (mess_matrix *A*, mess_matrix *B*, unsigned short *otype*, mess_matrix *O*) [static]

Parameters

in	*A*	input upper left matrix
in	*B*	input upper right matrix
in	*otype*	input output storage type
out	*O*	output matrix

Returns

zero on success or a non-zero error value otherwise

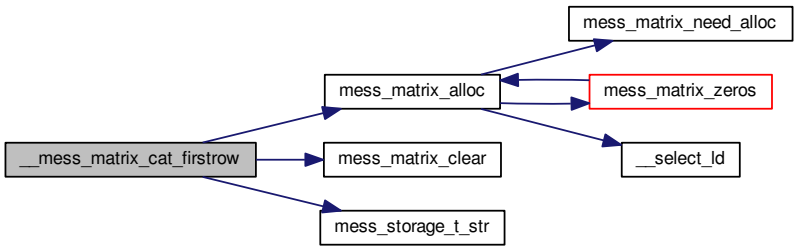
The __mess_matrix_cat_firstrow function concatenates two matrices in the following way:

```
O = [A,B];
```

Attention

Internal use only.

Here is the call graph for this function:



10.38.2.5 static int __mess_matrix_cat_full (mess_matrix *A*, mess_matrix *B*, mess_matrix *C*, mess_matrix *D*, unsigned short *otype*, mess_matrix *O*) [static]

Parameters

in	*A*	input upper left matrix
in	*B*	input upper right matrix
in	*C*	input lower left matrix
in	*D*	input lower right matrix
in	*otype*	input output storage type
out	*O*	output matrix

Returns

zero on success or a non-zero error value otherwise

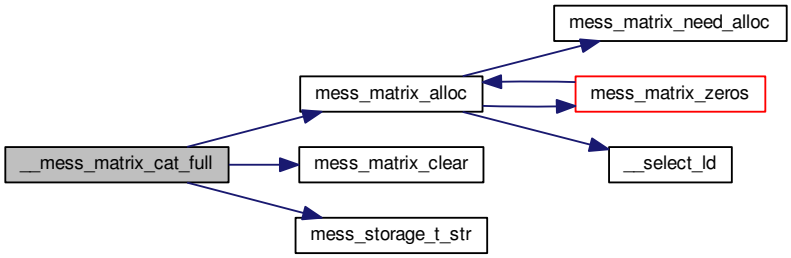
The __mess_matrix_cat_full function concatenates 4 matrices in a square:

```
O = [A,B; C,D];
```

Attention

Internal use only.

Here is the call graph for this function:



10.38.2.6 static int __mess_matrix_cat_LABC (mess_matrix *A*, mess_matrix *B*, mess_matrix *C*, unsigned short *otype*, mess_matrix *O*) [static]

Parameters

in	*A*	input upper left matrix
in	*B*	input upper right matrix
in	*C*	input lower left matrix
in	*otype*	input output storage type
out	*O*	output matrix

Returns

zero on success or a non-zero error value otherwise

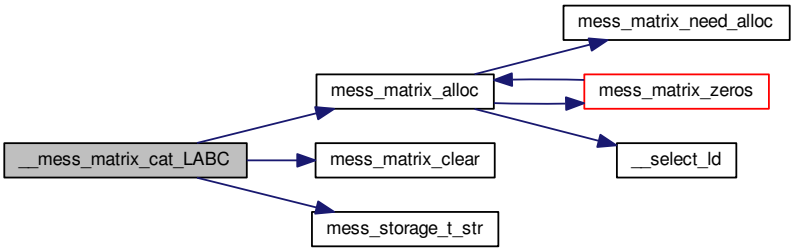
The __mess_matrix_cat_LABC function concatenates three matrices like:

```
O = [A,B;C,0];
```

Attention

Internal use only.

Here is the call graph for this function:



10.38.2.7 static int __mess_matrix_cat_LACD (mess_matrix *A*, mess_matrix *C*, mess_matrix *D*, unsigned short *otype*, mess_matrix *O*) [static]

Parameters

in	*A*	input upper left matrix
in	*C*	input lower left matrix
in	*D*	input lower right matrix
in	*otype*	input output storage type
out	*O*	output matrix

Returns

zero on success or a non-zero error value otherwise

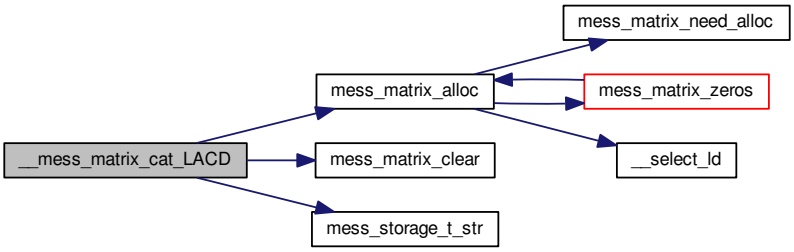
The __mess_matrix_cat_LACD function concatenates three matrices like:

```
O = [A,0;C,D];
```

Attention

Internal use only.

Here is the call graph for this function:



10.38.2.8 static int __mess_matrix_cat_RABD (mess_matrix *A*, mess_matrix *B*, mess_matrix *D*, unsigned short *otype*, mess_matrix *O*) [static]

Parameters

in	*A*	input upper left matrix
in	*B*	input upper right matrix
in	*D*	input lower right matrix
in	*otype*	input output storage type
out	*O*	output matrix

Returns

zero on success or a non-zero error value otherwise

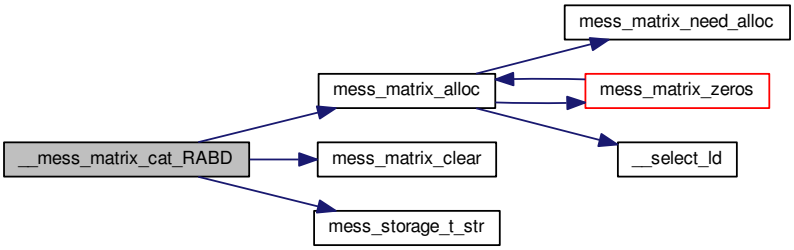
The __mess_matrix_cat_RABD function concatenates three matrices like:

```
O = [A,B;0,D];
```

Attention

Internal use only.

Here is the call graph for this function:



10.38.2.9 static int __mess_matrix_cat_RBCD (mess_matrix *B*, mess_matrix *C*, mess_matrix *D*, unsigned short *otype*, mess_matrix *O*) [static]

Parameters

in	*B*	input upper right matrix
in	*C*	input lower left matrix
in	*D*	input lower right matrix
in	*otype*	input output storage type
out	*O*	output matrix

Returns

zero on success or a non-zero error value otherwise

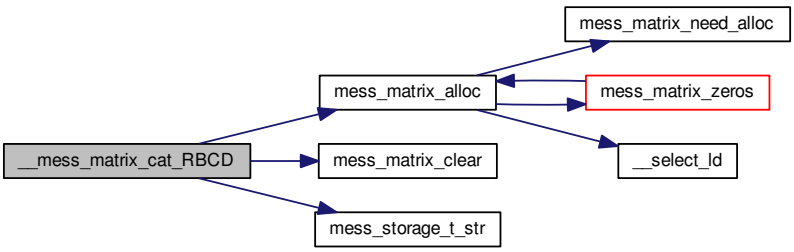
The __mess_matrix_cat_RBCD function concatenates three matrices like:

```
O = [0,B;C,D];
```

Attention

Internal use only.

Here is the call graph for this function:



10.39 cat.c File Reference

Functions

- int mess_vector_resize (mess_vector v, mess_int_t dim)

  Resize a vector.
- int mess_vector_cat (mess_vector x1, mess_vector x2, mess_vector x)

  Concatenate two vectors.
- int mess_vector_split (mess_vector input, mess_int_t n, mess_vector x1, mess_vector x2)

  Split a vector after the n-th entry.
- int mess_vector_lift (mess_vector in, mess_int_t n, mess_vector out)

  Add n zeros to vector.

Variables

- static mess_int_t __ONE = 1

10.39.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.39.2 Variable Documentation

10.39.2.1 mess_int_t __ONE = 1 [static]

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

10.40 category.dox File Reference

10.41 cg.c File Reference

Functions

- int mess_solver_cg (mess_mvpcall matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations using conjugate gradients.

10.41.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.42 check_add.c File Reference

Macros

- #define CHECK_ADD(TOL, P, ROWS, COLS, ALPHA, BETA, DTA, DTB, STA, STB)

Functions

- int main (int argc, char ∗argv[ ])

10.42.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

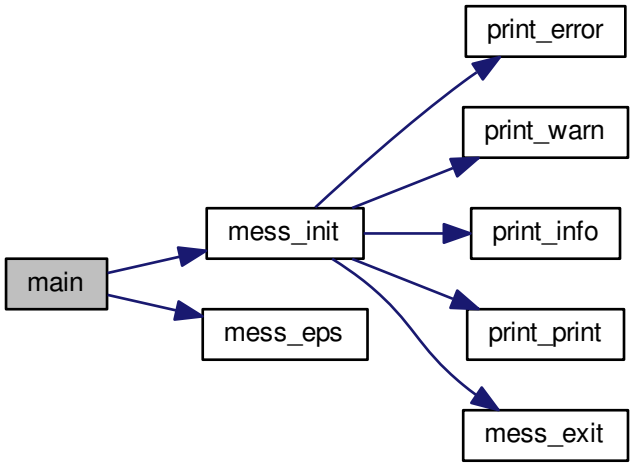
10.42.2 Macro Definition Documentation

10.42.2.1 #define CHECK_ADD(*TOL, P, ROWS, COLS, ALPHA,* BETA, *DTA, DTB, STA, STB*)

10.42.3 Function Documentation

10.42.3.1 int main (int *argc,* char ∗ *argv[ ]*)

Here is the call graph for this function:



10.43 check_addcol2p.c File Reference

Macros

- #define CHECKADDCOL2P(Z, S1, V1, S2, V2, TEMP1, TEMP2, TEMP3, ERR, EPS, DIFF)

Functions

- int main (int argc, char **argv)

10.43.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.43.2 Macro Definition Documentation

10.43.2.1 #define CHECKADDCOL2P(Z, S1, V1, S2, V2, TEMP1, TEMP2, TEMP3, ERR, EPS, DIFF)

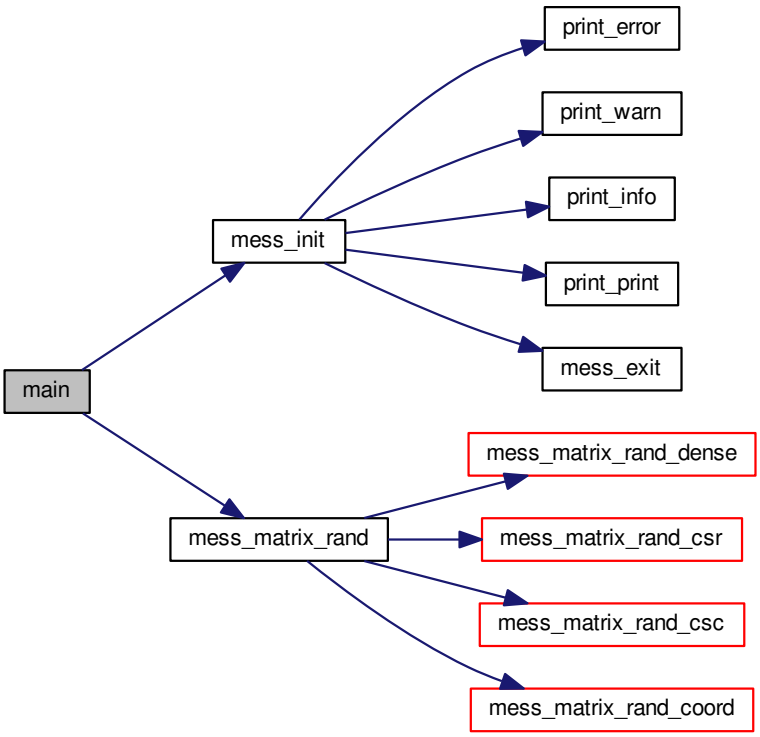
Value:

```
{         \
    CALL(mess_matrix_copy(Z,TEMP1));
      \
    CALL(mess_matrix_copy(V1,TEMP2));
      \
    CALL(mess_matrix_add(S2,V2,S1,TEMP2));
      \
    CALL(mess_matrix_cat(TEMP1,TEMP2,NULL,NULL,MESS_DENSE,TEMP3));
      \
    CALL(mess_matrix_addcols2p(Z,S1,V1,S2,V2));
      \
    CALL(mess_matrix_diffnorm(Z,TEMP3,&DIFF));
        \
    if(DIFF>EPS){                                                    \
        printf("Failed\n");                                          \
        printf("Diff:%e\n",diff);                                    \
        ++ERR;                                                       \
    }                                                                \
}
```

10.43.3 Function Documentation

10.43.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.44 check_any.c File Reference

Macros

- #define CHECK_ANY(TOL, ROWS, COLS, DTA, STA, DIM)

Functions

- static mess_int_t f_real (double val)
- static mess_int_t f_cpx (mess_double_cpx_t val)
- int main (int argc, char *argv[ ])

10.44.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.44.2 Macro Definition Documentation

10.44.2.1 #define CHECK_ANY(*TOL, ROWS, COLS, DTA, STA, DIM*)

Value:

```
{                                   \
    int ret;                                                                \
    double diff;                                                            \
    mess_matrix A, TMPA;
      \
    MESS_INIT_MATRICES(&A,&TMPA);                                           \
    \
    mess_vector v, tmpv;
      \
    MESS_INIT_VECTORS(&v,&tmpv);
      \
    CALL(mess_vector_alloc(v,0,MESS_REAL));
         \
    CALL(mess_vector_alloc(tmpv,0,MESS_REAL));
         \
    \
    /*generate random matrix and create reference solution*/                \
    CALL(mess_matrix_rand(A,ROWS,COLS,MESS_DENSE,DTA,0.5));
         \
    CALL(mess_matrix_convert(A,TMPA,STA));
      \
    CALL(mess_matrix_any(A,f_real,f_cpx,DIM,v));
         \
    \
    /* compute solution and compare */                                      \
    CALL(mess_matrix_any(TMPA,f_real,f_cpx,DIM,tmpv));
         \
    CALL(mess_vector_diffnorminf(v,tmpv,&diff));
      \
    \
    if(diff>TOL){                                                           \
        printf("FAILED\n");                                                 \
        printf("Matrix\n");                                                 \
        mess_matrix_printinfo(TMPA);                                        \
        return 1;                                                           \
    }                                                                       \
    MESS_CLEAR_MATRICES(&A,&TMPA);
      \
    MESS_CLEAR_VECTORS(&v,&tmpv);                                           \
}
```

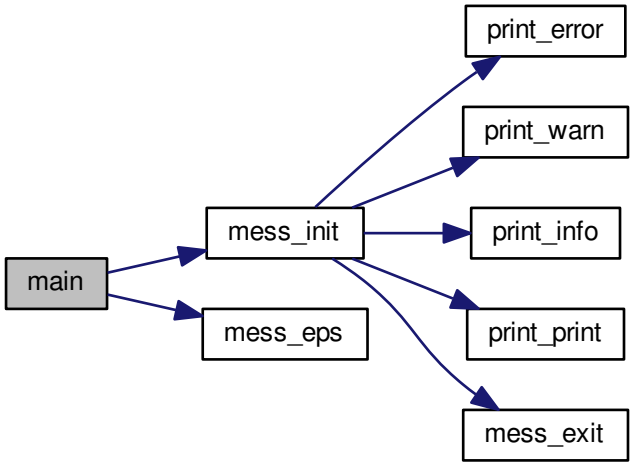
10.44.3 Function Documentation

10.44.3.1 static mess_int_t f_cpx (mess_double_cpx_t *val*) [static]

10.44.3.2 static mess_int_t f_real (double *val*) [static]

10.44.3.3 int main (int *argc*, char ∗ *argv[ ]*)

Here is the call graph for this function:



10.45 check_arnoldi_template.c File Reference

Macros

- #define CHECK_ARNOLDI(K, A, MVPA, SV, H, V)

Functions

- int main (int argc, char ∗∗argv)

10.45.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

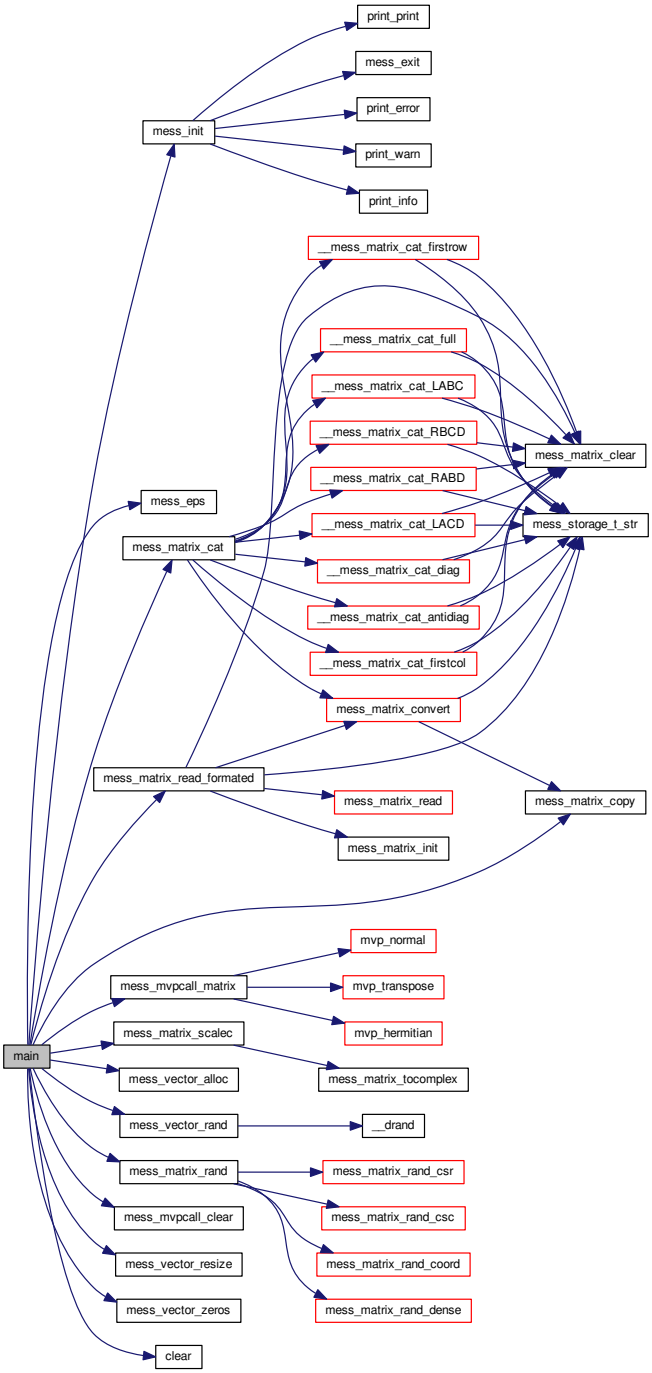
10.45.2 Macro Definition Documentation

10.45.2.1 #define CHECK_ARNOLDI(K, A, *MVPA*, *SV*, H, *V*)

10.45.3 Function Documentation

10.45.3.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.46 check_arnoldi_template_nrm.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.46.1 Detailed Description

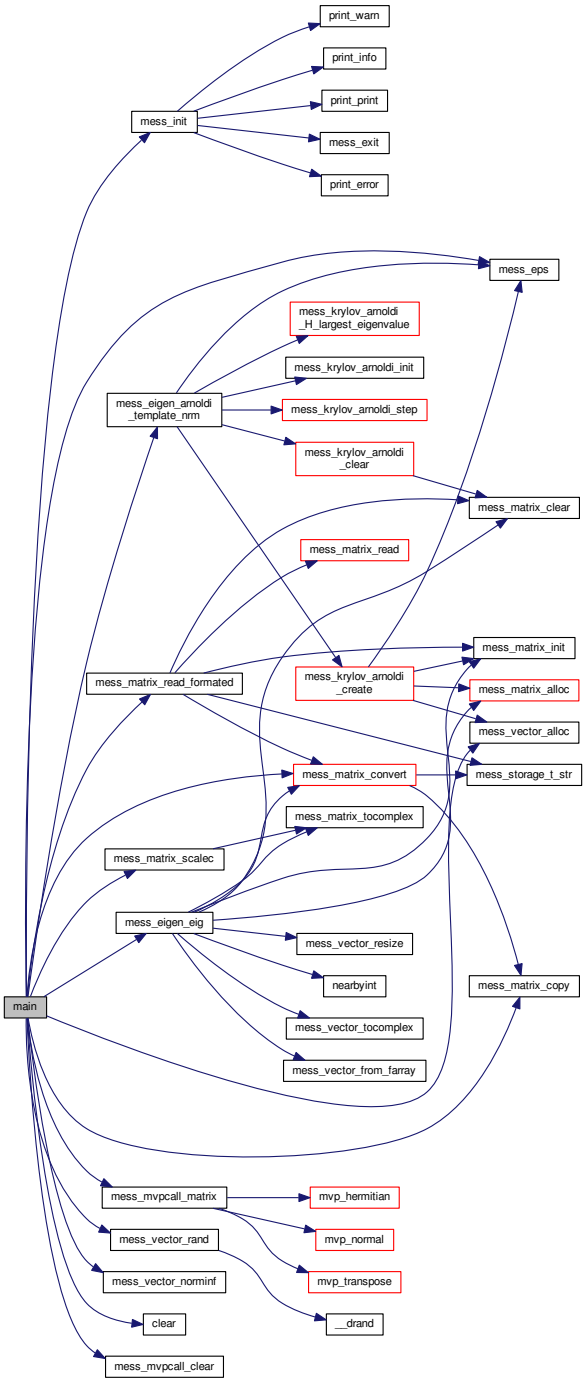
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.46.2 Function Documentation

10.46.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.47 check_arpack.c File Reference

Functions

- int check_ev (mess_matrix A, mess_vector ev, mess_matrix V)

  Check if the eigenvalue problem is solved correctly.
- int call_arnoldi (mess_matrix A, mess_int_t nev, mess_eigen_arpack_which_t which)

  Check if the eigenvalue problem is solved correctly.
- int main (int argc, char ∗argv[ ])

10.47.1 Detailed Description

10.47.2 Function Documentation

10.47.2.1 int call_arnoldi (mess_matrix *A*, mess_int_t *nev*, mess_eigen_arpack_which_t *which*)

Parameters

in	*A*	input matrix to compute its eigenvalues from
in	*nev*	input number of desired eigenvalues
in	*which*	input eigenvalues to select in the Arnoldi process

Returns

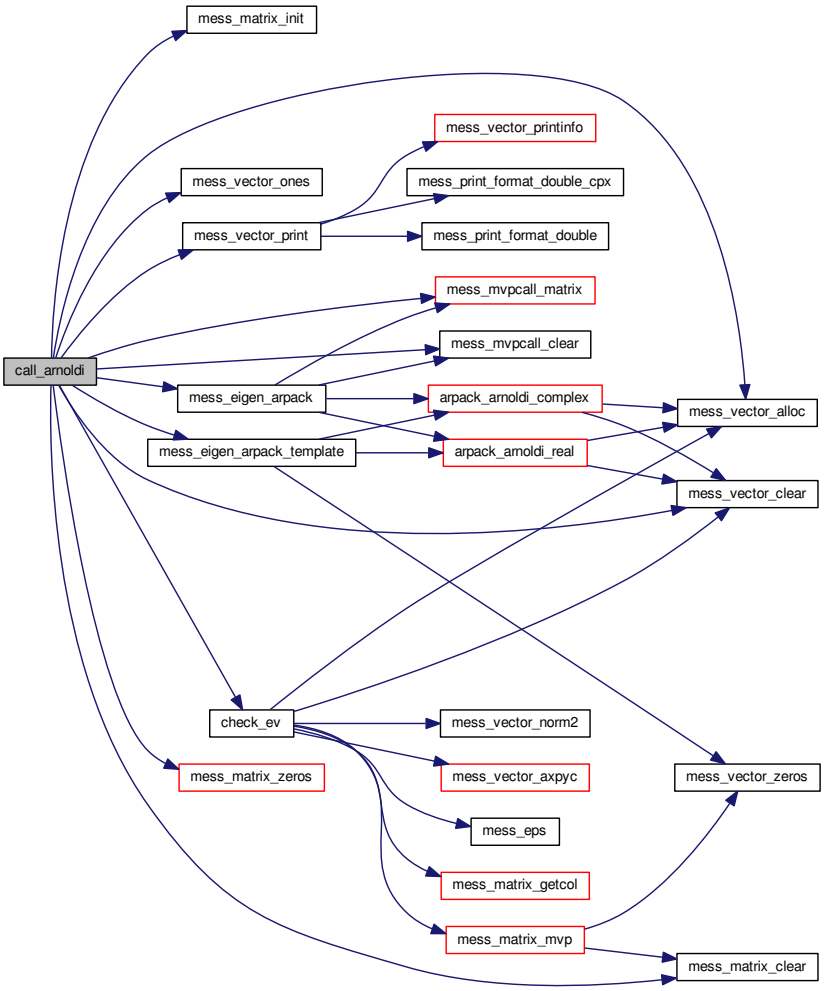
zero on success or a non-zero error value otherwise

The call_arnoldi function computes the eigenvalues and eigenvector for the eigenvalue problem

$$Ax = \lambda x$$

and checks if the eigenvalue problem is fulfilled.

Here is the call graph for this function:



10.47.2.2 int check_ev (mess_matrix *A*, mess_vector *ev*, mess_matrix *V*)

Parameters

in	*A*	input matrix
in	*ev*	input vector containing eigenvalues
in	*V*	input matrix containing eigenvectors

Returns

zero on success or a non-zero error value otherwise

The check_ev function checks if the eigenvalue problem

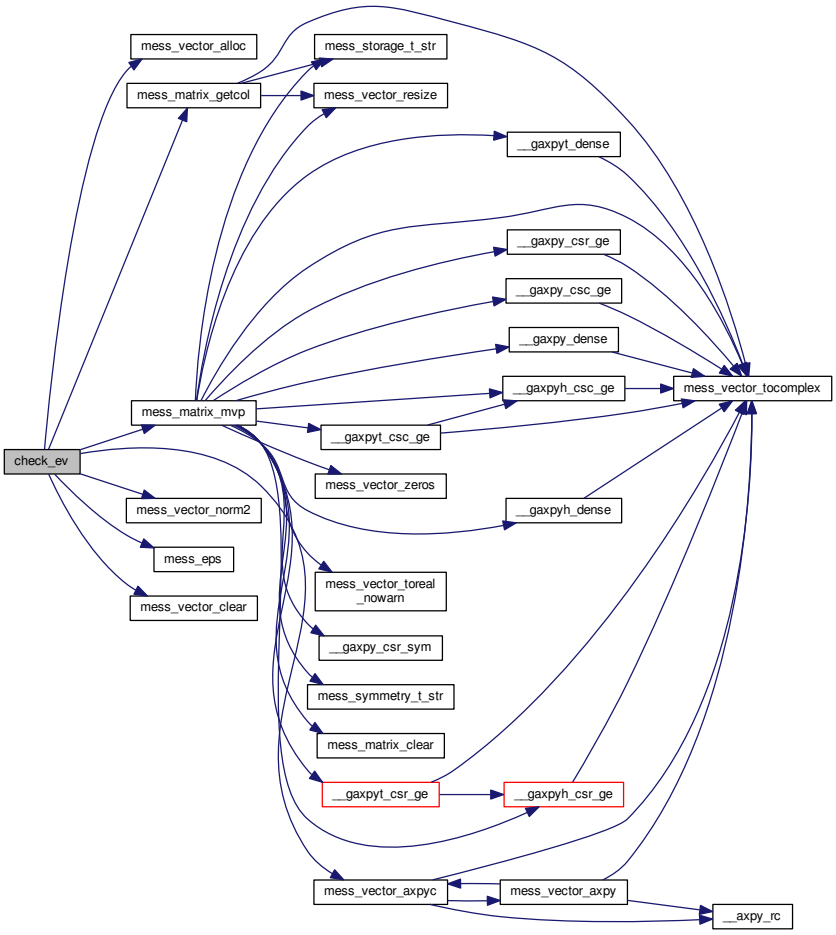
$$AV(:,i) = ev(i)V(:,i)$$

is solved correctly for $i = 1, \cdots, dim(ev)$.
If

$$\frac{\|AV(:,i) - ev(i)V(:,i)\|_2}{\|V(:,i)\|_2} > \sqrt{\epsilon n}$$

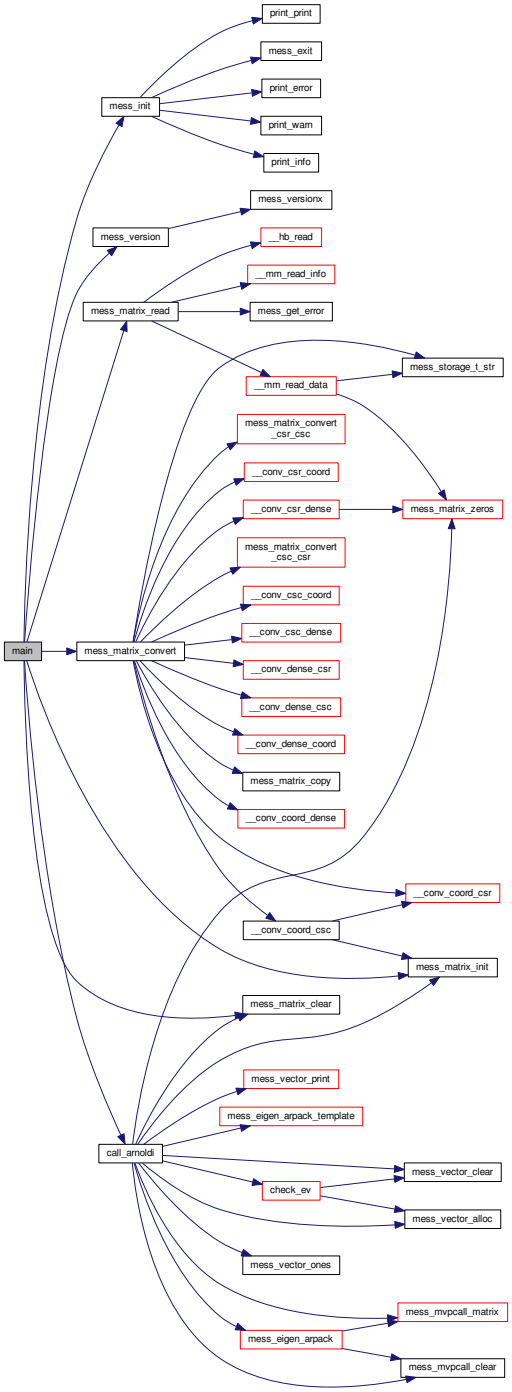
the error value is increased by 1.
n is the number of rows of V.

Here is the call graph for this function:



10.47.2.3 int main (int *argc*, char ∗ *argv[ ]*)

Here is the call graph for this function:



10.48 check_arpack_lanczos.c File Reference

Functions

- int mvp (void ∗data, mess_vector x, mess_vector y)

Check if the eigenvalue problem is solved correctly.

- int check_ev (mess_matrix A, mess_vector ev, mess_matrix V)

  Check if the eigenvalue problem is solved correctly.

- int call_lanczos (mess_matrix A, mess_int_t nev, mess_eigen_arpack_which_t which)

  Check if the eigenvalue problem is solved correctly.

- int main (int argc, char ∗argv[ ])

10.48.1 Detailed Description

10.48.2 Function Documentation

10.48.2.1 int call_lanczos (mess_matrix *A*, mess_int_t *nev*, mess_eigen_arpack_which_t *which*)

Parameters

in	*A*	input matrix to compute its eigenvalues from
in	*nev*	input number of desired eigenvalues
in	*which*	input eigenvalues to select in the Lanczos process

Returns

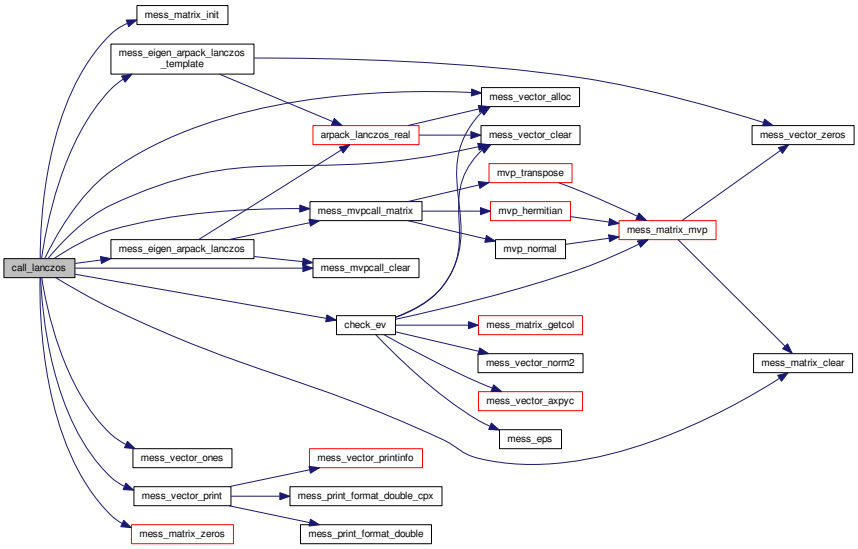
zero on success or a non-zero error value otherwise

The call_lanczos function computes the eigenvalues and eigenvector for the eigenvalue problem

$$Ax = \lambda x$$

and checks if the eigenvalue problem is fulfilled.

Here is the call graph for this function:



10.48.2.2 int check_ev (mess_matrix *A*, mess_vector *ev*, mess_matrix *V*)

Parameters

in	*A*	input matrix
in	*ev*	input vector containing eigenvalues
in	*V*	input matrix containing eigenvectors

Returns

zero on success or a non-zero error value otherwise

The check_ev function checks if the eigenvalue problem

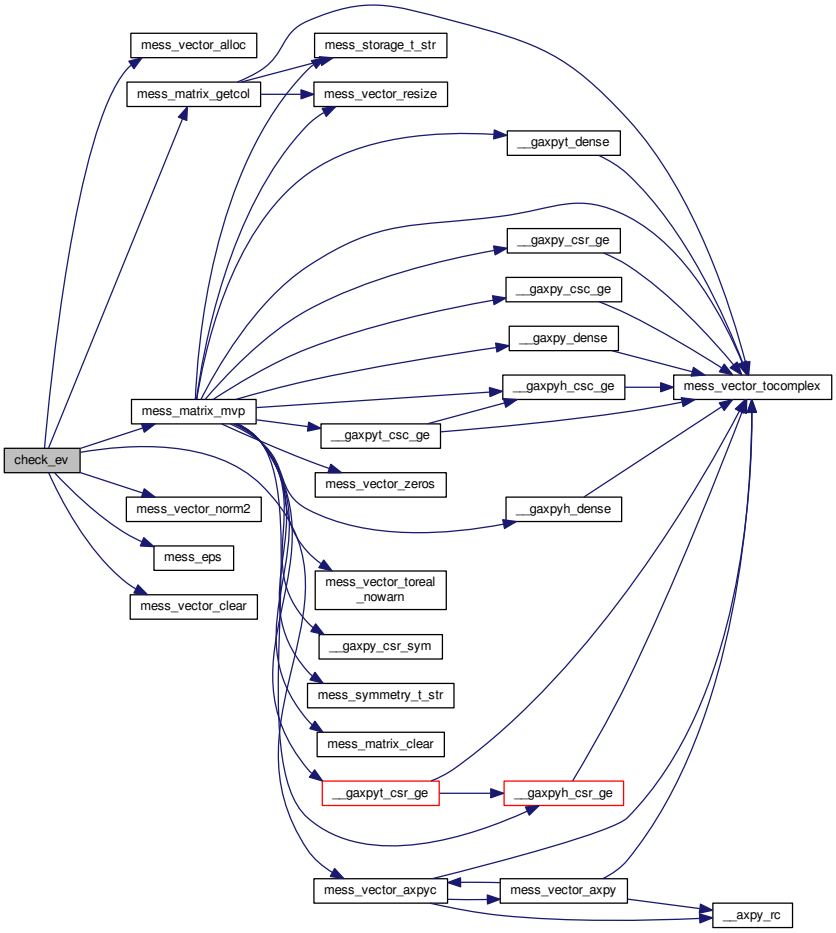
$$AV(:,i) = ev(i)V(:,i)$$

is solved correctly for $i = 1, \cdots, dim(ev)$.
If

$$\frac{\|AV(:,i) - ev(i)V(:,i)\|_2}{\|V(:,i)\|_2} > \sqrt{\epsilon n}$$

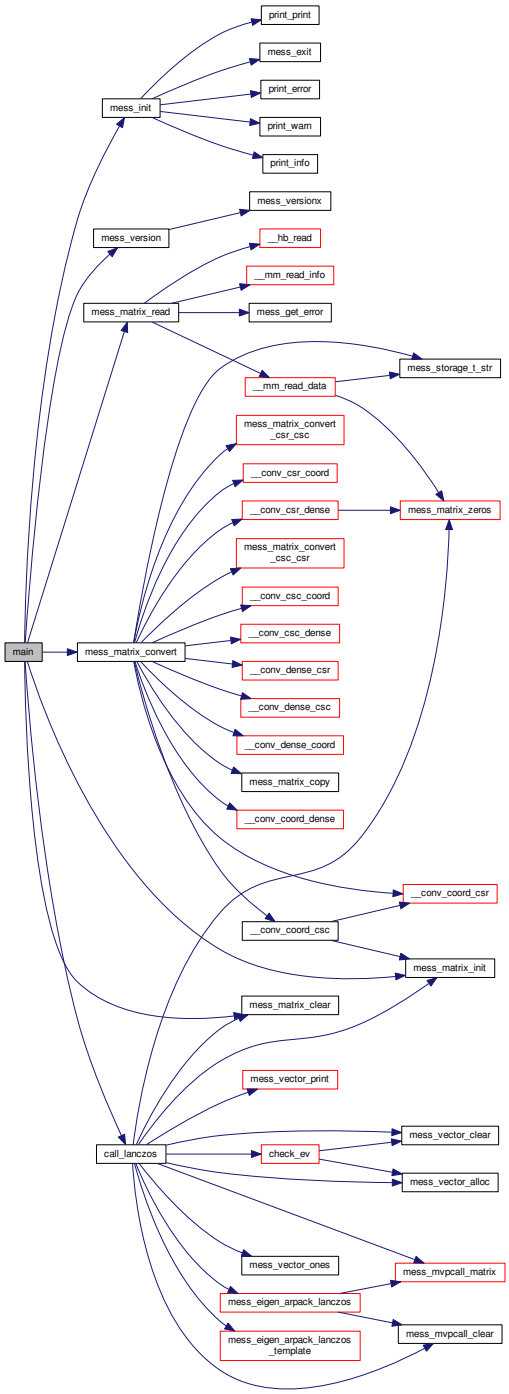
the error value is increased by 1.
n is the number of rows of V.

Here is the call graph for this function:



10.48.2.3 int main (int *argc*, char ∗ *argv[ ]*)

Here is the call graph for this function:



10.48.2.4 int mvp (void ∗ *data*, mess_vector *x*, mess_vector *y*)

Parameters

in	*data*	input data of a matrix

Parameters

in	*x*	input vector
out	*y*	output vectorvector

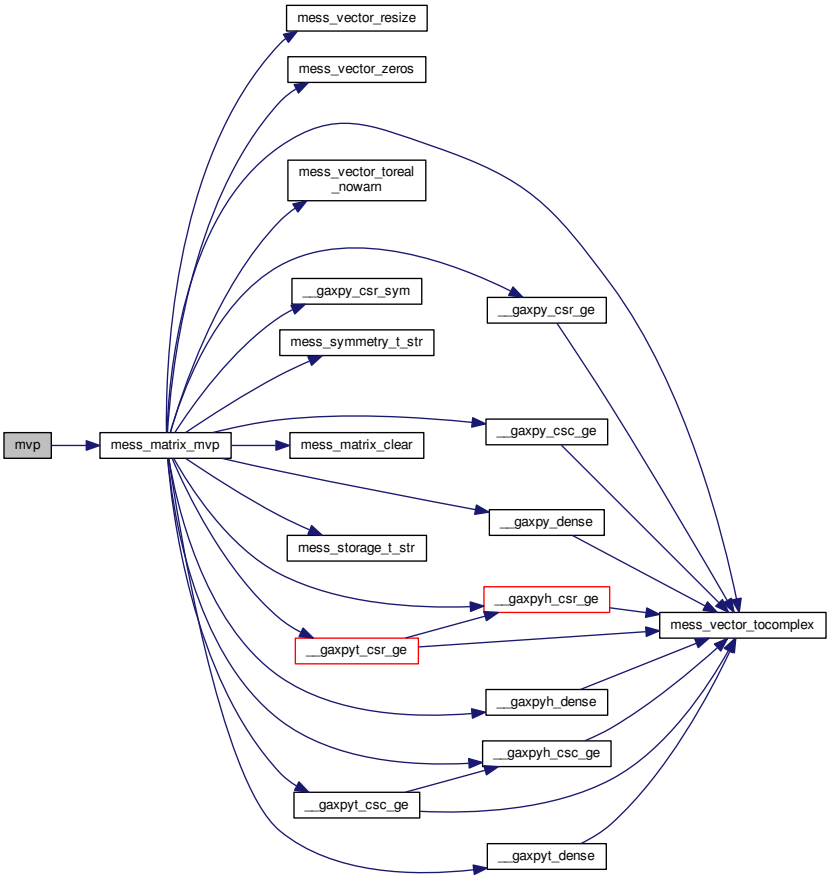
Returns

zero on success or a non-zero error value otherwise

The mvp function determines a matrix A out of $data$ and computes the matrix-vector product

$$y = Ax.$$

Here is the call graph for this function:



10.49 check_backslash.c File Reference

Macros

- #define CHECK_BACKSLASH(OP, A, B, X, TEMP, ERR, EPS, DIFF)
- #define CHECK_BACKSLASHM(OP, A, B, X, TEMP, ERR, EPS, DIFF)

Functions

- int main (int argc, char **argv)

10.49.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_matrix_backslash and mess_matrix_backslashm functions defined in backslash.c that means it checks if the solution of

$$op(A)x = b$$

and

$$op(A)X = B$$

is computed correctly for given matrices A and B and given vector b.
Operations $op(.)$ on A can be

- MESS_OP_NONE ($op(A) = A$),
- MESS_OP_TRANSPOSE ($op(A) = A^T$),
- MESS_OP_HERMITIAN ($op(A) = A^H$).

10.49.2 Macro Definition Documentation

10.49.2.1 #define CHECK_BACKSLASH(*OP*, A, B, X, *TEMP*, *ERR*, *EPS*, DIFF)

Value:

```
{                                   \
    CALL(mess_matrix_backslash(OP,A,B,X));
       \
    CALL(mess_matrix_mvp(OP,A,X,TEMP));
      \
    CALL(mess_vector_diffnorm(B,TEMP,&DIFF));
        \
    if(DIFF>EPS){                                                       \
        printf("Failed:\n");                                            \
        printf("OP: %s\n",#OP);                                         \
        CALL(mess_matrix_printinfo(A));
         \
        CALL(mess_vector_printinfo(B));
         \
        printf("Diff: %e\n",DIFF);                                      \
        ++ERR;                                                          \
    }                                                                   \
}
```

10.49.2.2 #define CHECK_BACKSLASHM(*OP*, A, B, X, *TEMP*, *ERR*, *EPS*, DIFF)

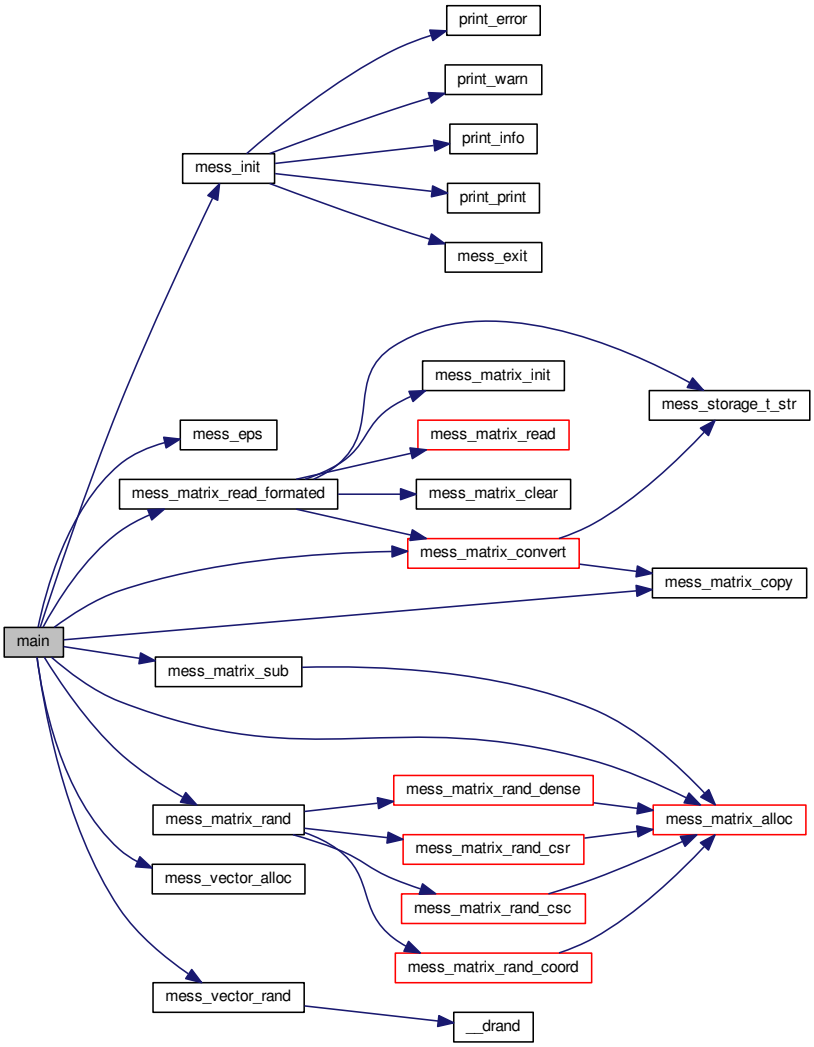
Value:

```
{                                   \
    CALL(mess_matrix_backslashm(OP,A,B,X));
        \
    CALL( mess_matrix_multiply(OP, A, MESS_OP_NONE,
      X, TEMP));                    \
    CALL(mess_matrix_diffnorm(B,TEMP,&DIFF));
        \
    if(DIFF>EPS){                                                       \
        printf("BACKSLASHM Failed:\n");                                 \
        printf("OP: %s\n",#OP);                                         \
        CALL(mess_matrix_printinfo(A));
         \
        CALL(mess_matrix_printinfo(B));
         \
        printf("Diff: %e\n",DIFF);                                      \
        ++ERR;                                                          \
    }                                                                   \
}
```

10.49.3 Function Documentation

10.49.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.50 check_bandwidth.c File Reference

Functions

- int main (int argc, char **argv)

10.50.1 Detailed Description

Author

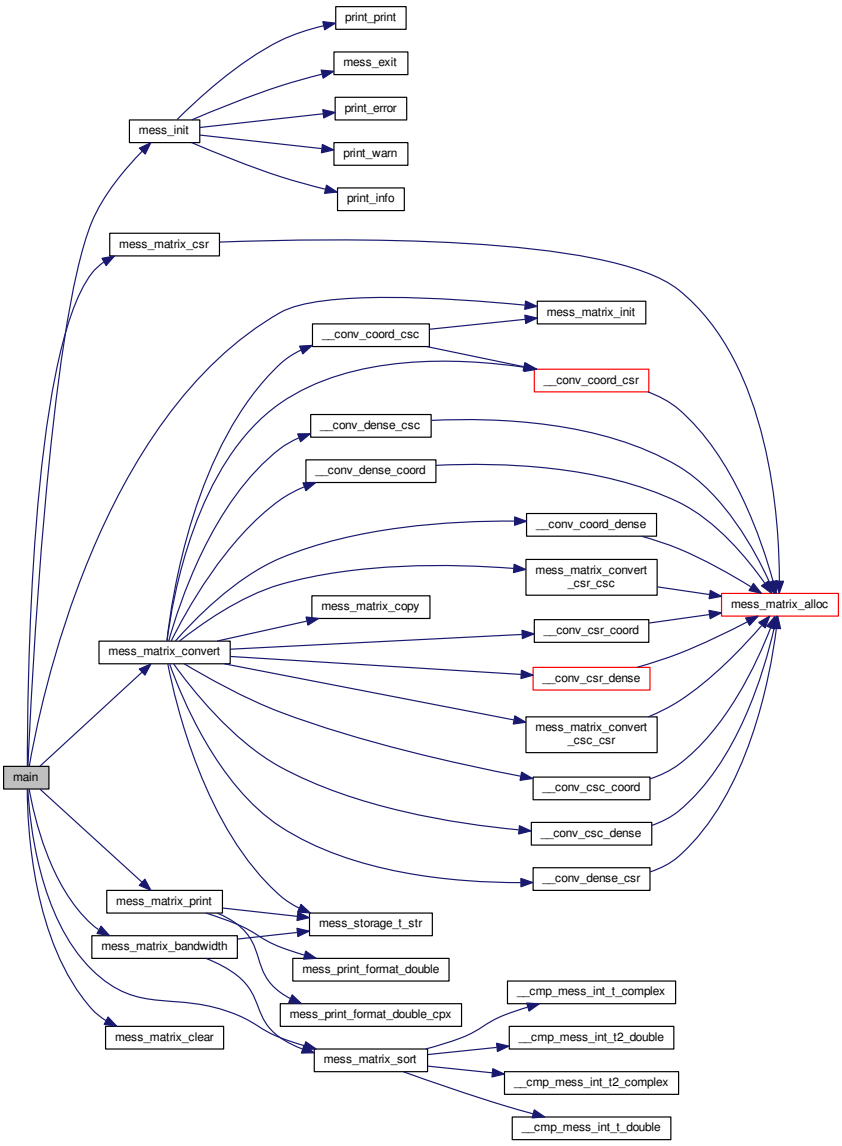
Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks if mess_matrix_bandwidth works correctly.

10.50.2 Function Documentation

10.50.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.51 check_ber_sign_fac.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.51.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the solution of the generalized Bernoulli equation

$$AXE^T + EXA^T - EXBB^TXE^T = 0$$

and the Bernoulli equations

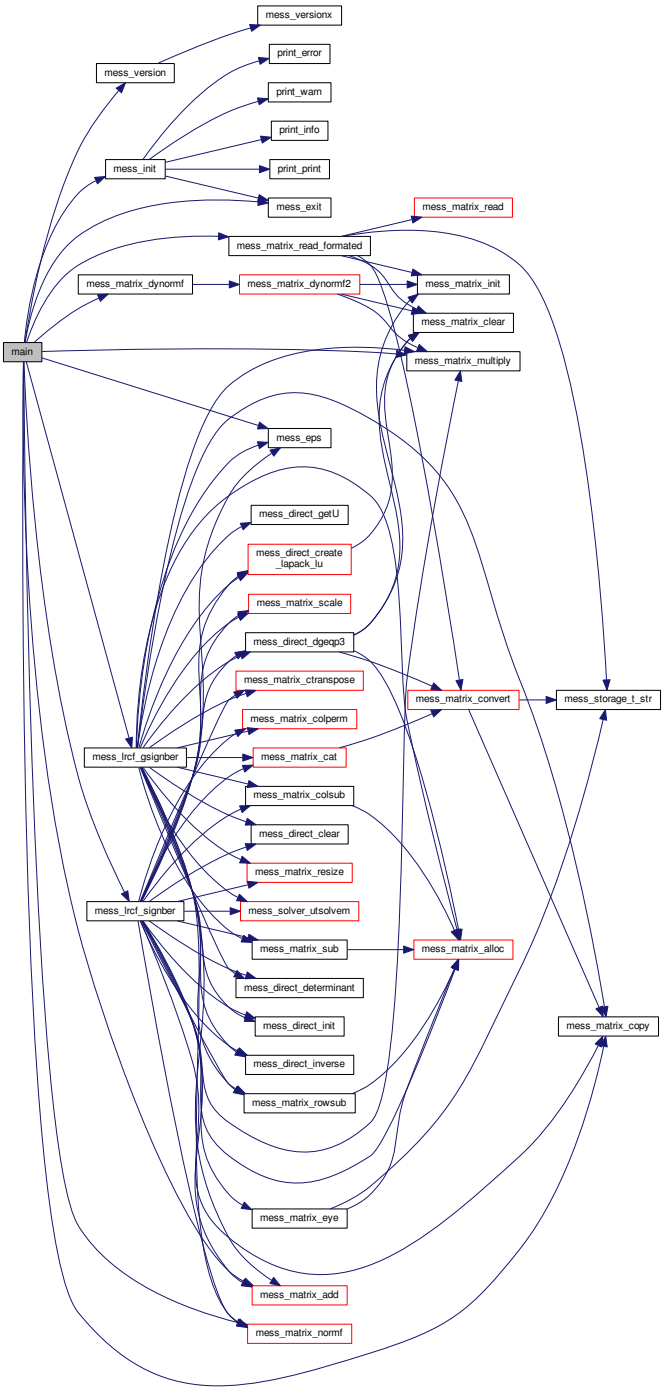
$$A^TXE + E^TXA - E^TXBB^TXE = 0$$

by computing the corresponding resiudal.
Time is measured for computing the solution of one Bernoulli equation and its residual.

10.51.2 Function Documentation

10.51.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.52 check_biorth.c File Reference

Functions

- int main (int argc, char ∗argv[ ])

10.52.1 Detailed Description

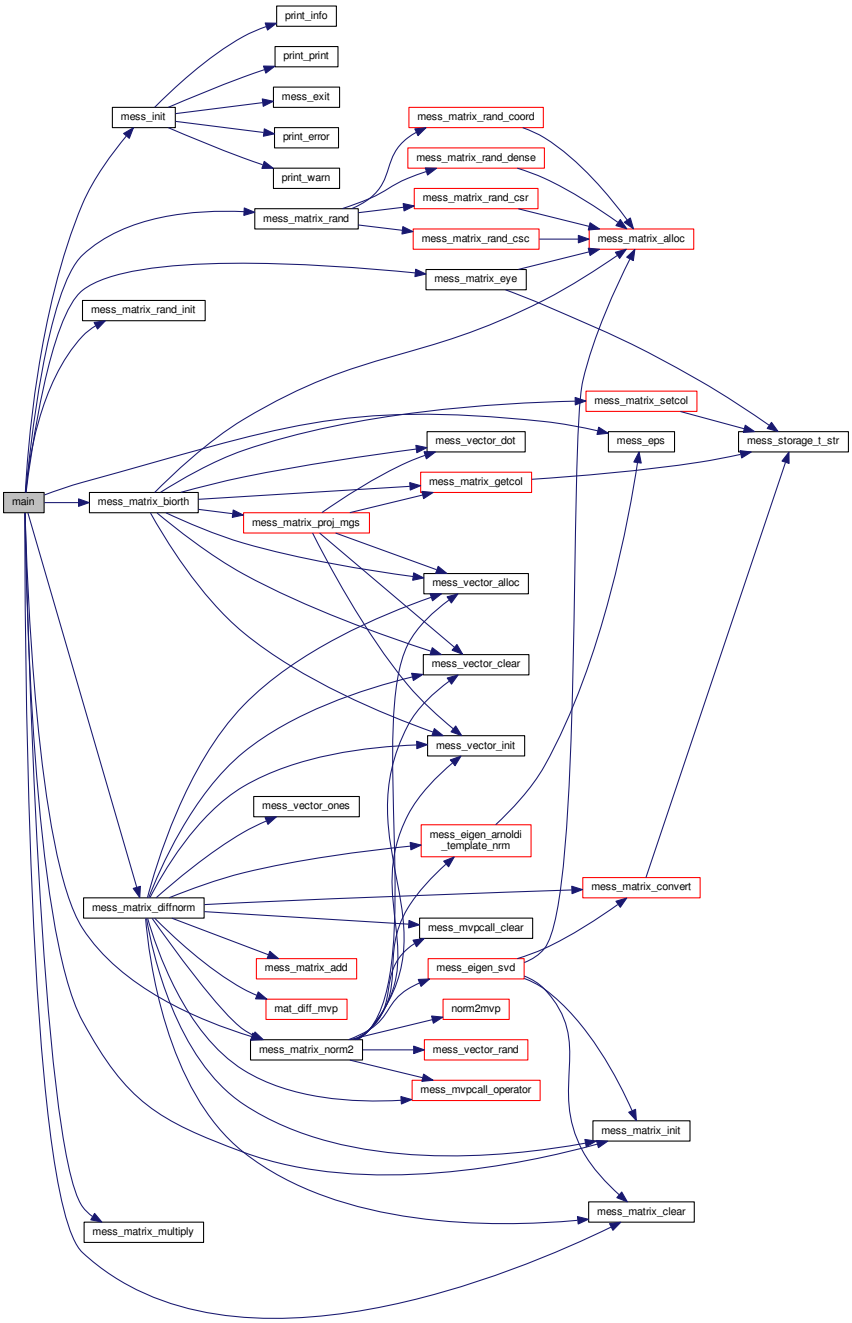
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.52.2 Function Documentation

10.52.2.1 int main (int *argc*, char ∗ *argv[ ]*)

Here is the call graph for this function:



10.53 check_cat.c File Reference

Macros

- #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed =================\n", #X); } else {fprintf(stderr, "test %35s passed.\n",#X); }

Functions

- int check_cat (mess_matrix A, mess_storage_t storeA, mess_matrix B, mess_storage_t storeB, mess_matrix C, mess_storage_t storeC, mess_matrix D, mess_storage_t storeD, mess_storage_t outtype, mess_matrix ref)
- int main (int argc, char **argv)

Variables

- double data_A [9] = { 11,21,31,12,22,32,13,23,33}
- double data_B [9] = { 14,24,34,15,25,35,16,26,36}
- double data_C [9] = { 41,51,61,42,52,62,43,53,63}
- double data_D [9] = { 44,54,64,45,55,65,46,56,66}
- double data_AB [18] = { 1.100000e+01,2.100000e+01,3.100000e+01,1.200000e+01,2.200000e+01,3.200000e+01,1.300000e+01,2.300000e+01,3.300000e+01,1.400000e+01,2.400000e+01,3.400000e+01,1.500000e+01,2.500000e+01,3.500000e+01,1.600000e+01,2.600000e+01,3.600000e+01}
- double data_AC [18] = { 1.100000e+01,2.100000e+01,3.100000e+01,4.100000e+01,5.100000e+01,6.100000e+01,1.200000e+01,2.200000e+01,3.200000e+01,4.200000e+01,5.200000e+01,6.200000e+01,1.300000e+01,2.300000e+01,3.300000e+01,4.300000e+01,5.300000e+01,6.300000e+01}
- double data_AD [36] = { 1.100000000000000e+01, 2.100000000000000e+01, 3.100000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 1.200000000000000e+01, 2.200000000000000e+01, 3.200000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 1.300000000000000e+01, 2.300000000000000e+01, 3.300000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.400000000000000e+01, 5.400000000000000e+01, 6.400000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.500000000000000e+01, 5.500000000000000e+01, 6.500000000000000e+01,0.000000000000000e+00,0.000000000000000e+00,0.000000000000000e+00,4.600000000000000e+01,5.600000000000000e+01,6.600000000000000e+01}
- double data_BC [36] = { 0,0,0,41,51,61,0,0,0,42,52,62,0,0,0,43,53,63,14,24,34,0,0,0,15,25,35,0,0,0,16,26,36,0,0,0}
- double data_ABC [36] ={1.100000000000000e+01,2.100000000000000e+01,3.100000000000000e+01,4.100000000000000e+01,5.100000000000000e+01,6.100000000000000e+01,1.200000000000000e+01,2.200000000000000e+01,3.200000000000000e+01,4.200000000000000e+01,5.200000000000000e+01,6.200000000000000e+01,1.300000000000000e+01,2.300000000000000e+01,3.300000000000000e+01,4.300000000000000e+01,5.300000000000000e+01,6.300000000000000e+01,1.400000000000000e+01,2.400000000000000e+01,3.400000000000000e+01,0.000000000000000e+00,0.000000000000000e+00,0.000000000000000e+00,1.500000000000000e+01,2.500000000000000e+01,3.500000000000000e+01,0.000000000000000e+00,0.000000000000000e+00,1.600000000000000e+01,2.600000000000000e+01,3.600000000000000e+01,0.000000000000000e+00,0.000000000000000e+00,0.000000000000000e+00}
- double data_ACD [36] ={ 1.100000000000000e+01, 2.100000000000000e+01, 3.100000000000000e+01, 4.100000000000000e+01, 5.100000000000000e+01, 6.100000000000000e+01, 1.200000000000000e+01, 2.200000000000000e+01, 3.200000000000000e+01, 4.200000000000000e+01, 5.200000000000000e+01, 6.200000000000000e+01, 1.300000000000000e+01, 2.300000000000000e+01, 3.300000000000000e+01, 4.300000000000000e+01, 5.300000000000000e+01, 6.300000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.400000000000000e+01, 5.400000000000000e+01, 6.400000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.500000000000000e+01,5.500000000000000e+01,6.500000000000000e+01,0.000000000000000e+00,0.000000000000000e+00,0.000000000000000e+00,4.600000000000000e+01,5.600000000000000e+01,6.600000000000000e+01}

- double data_ABD [36] = {1.100000000000000e+01,2.100000000000000e+01,3.100000000000000e+01,0.↩000000000000000e+00,0.000000000000000e+00,0.000000000000000e+00,1.200000000000000e+01,2.↩200000000000000e+01,3.200000000000000e+01,0.000000000000000e+00,0.000000000000000e+00,0.↩000000000000000e+00,1.300000000000000e+01,2.300000000000000e+01,3.300000000000000e+01,0.↩000000000000000e+00,0.000000000000000e+00,0.000000000000000e+00,1.400000000000000e+01,2.↩400000000000000e+01,3.400000000000000e+01,4.400000000000000e+01,5.400000000000000e+01,6.↩400000000000000e+01,1.500000000000000e+01,2.500000000000000e+01,3.500000000000000e+01,4.↩500000000000000e+01,5.500000000000000e+01,6.500000000000000e+01,1.600000000000000e+01,2.↩600000000000000e+01,3.600000000000000e+01,4.600000000000000e+01,5.600000000000000e+01,6.↩600000000000000e+01}
- double data_BCD [36] = {0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.100000000000000e+01, 5.100000000000000e+01, 6.100000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.200000000000000e+01, 5.200000000000000e+01, 6.200000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.300000000000000e+01, 5.300000000000000e+01, 6.300000000000000e+01, 1.400000000000000e+01, 2.400000000000000e+01, 3.400000000000000e+01, 4.400000000000000e+01, 5.400000000000000e+01, 6.400000000000000e+01, 1.500000000000000e+01, 2.500000000000000e+01, 3.500000000000000e+01, 4.500000000000000e+01, 5.500000000000000e+01, 6.500000000000000e+01, 1.600000000000000e+01, 2.600000000000000e+01, 3.600000000000000e+01, 4.600000000000000e+01,5.600000000000000e+01,6.↩600000000000000e+01}
- double data_ABCD [36] ={1.100000e+01, 2.100000e+01, 3.100000e+01, 4.100000e+01, 5.100000e+01, 6.100000e+01, 1.200000e+01, 2.200000e+01, 3.200000e+01, 4.200000e+01, 5.200000e+01, 6.↩200000e+01, 1.300000e+01, 2.300000e+01, 3.300000e+01, 4.300000e+01, 5.300000e+01, 6.300000e+01, 1.400000e+01, 2.400000e+01, 3.400000e+01, 4.400000e+01, 5.400000e+01, 6.400000e+01, 1.↩500000e+01, 2.500000e+01, 3.500000e+01, 4.500000e+01, 5.500000e+01, 6.500000e+01, 1.600000e+01, 2.600000e+01, 3.600000e+01, 4.600000e+01, 5.600000e+01, 6.600000e+01}

10.53.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

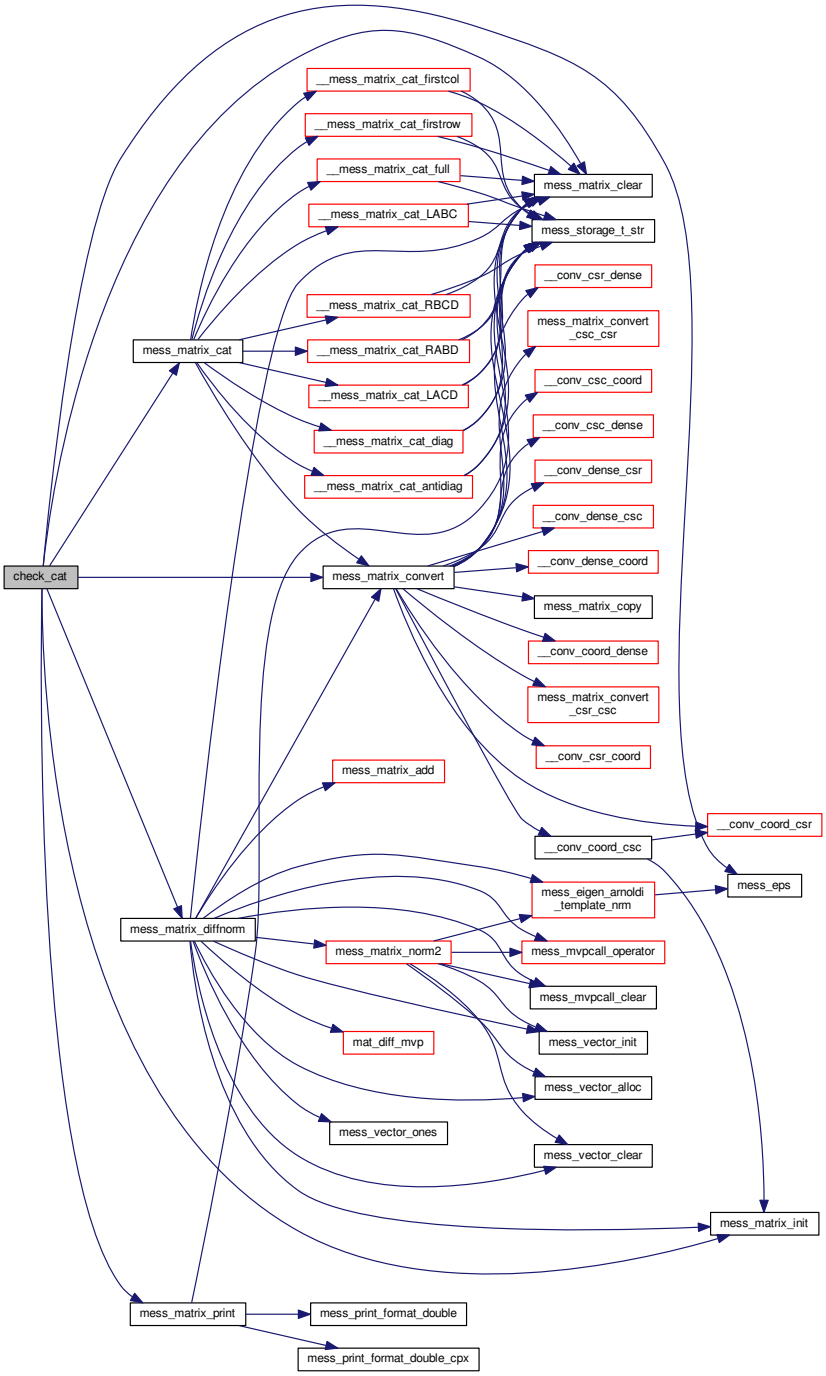
10.53.2 Macro Definition Documentation

10.53.2.1 #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed ================\n", #X); } else {fprintf(stderr, "test %35s passed.\n",#X); }

10.53.3 Function Documentation

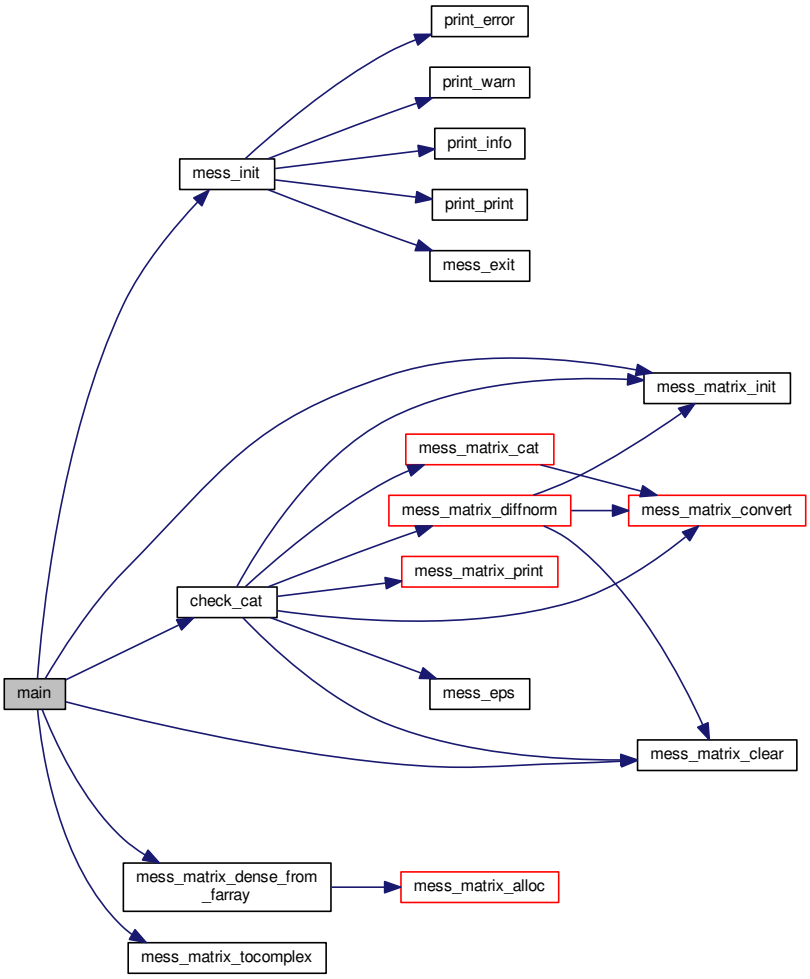
10.53.3.1 int check_cat (mess_matrix *A*, mess_storage_t *storeA*, mess_matrix *B*, mess_storage_t *storeB*, mess_matrix *C*, mess_storage_t *storeC*, mess_matrix *D*, mess_storage_t *storeD*, mess_storage_t *outtype*, mess_matrix *ref*)

Here is the call graph for this function:



10.53.3.2 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.53.4 Variable Documentation

10.53.4.1 double data_A[9] = { 11,21,31,12,22,32,13,23,33}

10.53.4.2 double data_AB[18] = { 1.100000e+01,2.100000e+01,3.100000e+01,1.200000e+01,2.200000e+01,3.200000e+01,1.↩ 300000e+01,2.300000e+01,3.300000e+01,1.400000e+01,2.400000e+01,3.400000e+01,1.500000e+01,2.500000e+01,3.↩ 500000e+01,1.600000e+01,2.600000e+01,3.600000e+01}

10.53.4.3 double data_ABC[36] ={1.100000000000000e+01,2.100000000000000e+01,3.100000000000000e+01,4.↩ 100000000000000e+01,5.100000000000000e+01,6.100000000000000e+01,1.200000000000000e+01,2.↩ 200000000000000e+01,3.200000000000000e+01,4.200000000000000e+01,5.200000000000000e+01,6.↩ 200000000000000e+01,1.300000000000000e+01,2.300000000000000e+01,3.300000000000000e+01,4.↩ 300000000000000e+01,5.300000000000000e+01,6.300000000000000e+01,1.400000000000000e+01,2.↩ 400000000000000e+01,3.400000000000000e+01,0.000000000000000e+00,0.000000000000000e+00,0.↩ 000000000000000e+00,1.500000000000000e+01,2.500000000000000e+01,3.500000000000000e+01,0.↩ 000000000000000e+00,0.000000000000000e+00,0.000000000000000e+00,1.600000000000000e+01,2.↩ 600000000000000e+01,3.600000000000000e+01,0.000000000000000e+00,0.000000000000000e+00,0.↩ 000000000000000e+00}

10.53.4.4 double data_ABCD[36] ={1.100000e+01, 2.100000e+01, 3.100000e+01, 4.100000e+01, 5.100000e+01, 6.100000e+01, 1.200000e+01, 2.200000e+01, 3.200000e+01, 4.200000e+01, 5.200000e+01, 6.200000e+01, 1.300000e+01, 2.300000e+01, 3.300000e+01, 4.300000e+01, 5.300000e+01, 6.300000e+01, 1.400000e+01, 2.400000e+01, 3.400000e+01, 4.400000e+01, 5.400000e+01, 6.400000e+01, 1.500000e+01, 2.500000e+01, 3.500000e+01, 4.500000e+01, 5.500000e+01, 6.500000e+01, 1.600000e+01, 2.600000e+01, 3.600000e+01, 4.600000e+01, 5.600000e+01, 6.600000e+01}

10.53.4.5 double data_ABD[36] = {1.100000000000000e+01,2.100000000000000e+01,3.100000000000000e+01,0.↩
000000000000000e+00,0.000000000000000e+00,0.000000000000000e+00,1.200000000000000e+01,2.↩
200000000000000e+01,3.200000000000000e+01,0.000000000000000e+00,0.000000000000000e+00,0.↩
000000000000000e+00,1.300000000000000e+01,2.300000000000000e+01,3.300000000000000e+01,0.↩
000000000000000e+00,0.000000000000000e+00,0.000000000000000e+00,1.400000000000000e+01,2.↩
400000000000000e+01,3.400000000000000e+01,4.400000000000000e+01,5.400000000000000e+01,6.↩
400000000000000e+01,1.500000000000000e+01,2.500000000000000e+01,3.500000000000000e+01,4.↩
500000000000000e+01,5.500000000000000e+01,6.500000000000000e+01,1.600000000000000e+01,2.↩
600000000000000e+01,3.600000000000000e+01,4.600000000000000e+01,5.600000000000000e+01,6.↩
600000000000000e+01}

10.53.4.6 double data_AC[18] = { 1.100000e+01,2.100000e+01,3.100000e+01,4.100000e+01,5.100000e+01,6.100000e+01,1.↩
200000e+01,2.200000e+01,3.200000e+01,4.200000e+01,5.200000e+01,6.200000e+01,1.300000e+01,2.300000e+01,3.↩
300000e+01,4.300000e+01,5.300000e+01,6.300000e+01}

10.53.4.7 double data_ACD[36] ={ 1.100000000000000e+01, 2.100000000000000e+01, 3.100000000000000e+01, 4.100000000000000e+01, 5.100000000000000e+01, 6.100000000000000e+01, 1.200000000000000e+01, 2.200000000000000e+01, 3.200000000000000e+01, 4.200000000000000e+01, 5.200000000000000e+01, 6.200000000000000e+01, 1.300000000000000e+01, 2.300000000000000e+01, 3.300000000000000e+01, 4.300000000000000e+01, 5.300000000000000e+01, 6.300000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.400000000000000e+01, 5.400000000000000e+01, 6.400000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.500000000000000e+01,5.500000000000000e+01,6.500000000000000e+01,0.000000000000000e+00,0.↩
000000000000000e+00,0.000000000000000e+00,4.600000000000000e+01,5.600000000000000e+01,6.↩
600000000000000e+01}

10.53.4.8 double data_AD[36] = { 1.100000000000000e+01, 2.100000000000000e+01, 3.100000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 1.200000000000000e+01, 2.200000000000000e+01, 3.200000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 1.300000000000000e+01, 2.300000000000000e+01, 3.300000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.400000000000000e+01, 5.400000000000000e+01, 6.400000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.500000000000000e+01, 5.500000000000000e+01, 6.500000000000000e+01,0.000000000000000e+00,0.↩
000000000000000e+00,0.000000000000000e+00,4.600000000000000e+01,5.600000000000000e+01,6.↩
600000000000000e+01}

10.53.4.9 double data_B[9] = { 14,24,34,15,25,35,16,26,36}

10.53.4.10 double data_BC[36] = { 0,0,0,41,51,61,0,0,0,42,52,62,0,0,0,43,53,63,14,24,34,0,0,0,15,25,35,0,0,0,16,26,36,0,0,0}

10.53.4.11 double data_BCD[36] = {0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.100000000000000e+01, 5.100000000000000e+01, 6.100000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.200000000000000e+01, 5.200000000000000e+01, 6.200000000000000e+01, 0.000000000000000e+00, 0.000000000000000e+00, 0.000000000000000e+00, 4.300000000000000e+01, 5.300000000000000e+01, 6.300000000000000e+01, 1.400000000000000e+01, 2.400000000000000e+01, 3.400000000000000e+01, 4.400000000000000e+01, 5.400000000000000e+01, 6.400000000000000e+01, 1.500000000000000e+01, 2.500000000000000e+01, 3.500000000000000e+01, 4.500000000000000e+01, 5.500000000000000e+01, 6.500000000000000e+01, 1.600000000000000e+01, 2.600000000000000e+01, 3.600000000000000e+01, 4.600000000000000e+01,5.600000000000000e+01,6.↩
600000000000000e+01}

10.53.4.12 double data_C[9] = { 41,51,61,42,52,62,43,53,63}

10.53.4.13 double data_D[9] = { 44,54,64,45,55,65,46,56,66}

10.54 check_cholmod.c File Reference

Macros

- #define CHECK_CONVERTCHOLMOD_DENSEMATRIX(A, A_CHOLMOD, TMP, C, ERR, EPS, DIFF)
- #define CHECK_CONVERTCHOLMOD_VECTOR(V, V_CHOLMOD, TMP, C, ERR, EPS, DIFF)
- #define CHECK_CONVERTCHOLMOD_SPARSEMATRIX(A, A_CHOLMOD, TMP, C, ERR, EPS, DIFF)

Functions

- int cholmod_print_dense_all (cholmod_dense *A)
- int cholmod_print_sparse_all (cholmod_sparse *A)
- int main ()

10.54.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks all function defined in cholmod.c that means it checks if a `cholmod_dense` or `cholmod_sparse` datastructure can be converted in a MESS_DENSE mess_matrix or a MESS_CSC mess_matrix structure, respectively and vice versa.
It also checks if a `cholmod_dense` datastructure can be converted to a mess_vector.

10.54.2 Macro Definition Documentation

10.54.2.1 #define CHECK_CONVERTCHOLMOD_DENSEMATRIX(A, *A_CHOLMOD, TMP,* C, *ERR, EPS,* DIFF)

Value:

```
{   \
    CALL(mess_matrix_convert_dense_to_cholmod(
      A,&A_CHOLMOD,&C));        \
    CALL(mess_matrix_convert_cholmod_to_dense(A_CHOLMOD, TMP, &
      C));     \
    cholmod_l_free_dense(&A_CHOLMOD,&C);                                \
    CALL(mess_matrix_diffnorm(A,TMP,&DIFF));                            \
    if(DIFF>EPS){                                                       \
        printf("CHECK_CONVERTCHOLMOD_DENSEMATRIX failed:\n");           \
        CALL(mess_matrix_printinfo(A));
      \
        CALL(mess_matrix_printshort(A));
      \
        ERR++;                                                          \
    }                                                                   \
}
```

10.54.2.2 #define CHECK_CONVERTCHOLMOD_SPARSEMATRIX(A, *A_CHOLMOD, TMP,* C, *ERR, EPS,* DIFF)

Value:

```
{  \
    CALL(mess_matrix_convert_csc_to_cholmod(
      A,&A_CHOLMOD,&C));          \
    CALL(mess_matrix_convert_cholmod_to_csc(A_CHOLMOD, TMP, &
      C));       \
    cholmod_l_free_sparse(&A_CHOLMOD,&C);                                \
    CALL(mess_matrix_add(1,A,-1,TMP));
      \
    CALL(mess_matrix_normf(TMP,&DIFF));                                  \
    if(DIFF>EPS){                                                        \
        printf("CONVERTCHOLMOD_SPARSEMATRIX failed:\n");                 \
        CALL(mess_matrix_printinfo(A));
      \
        CALL(mess_matrix_printshort(A));
      \
        ERR++;                                                           \
    }                                                                    \
}
```

10.54.2.3 #define CHECK_CONVERTCHOLMOD_VECTOR(*V, V_CHOLMOD, TMP,* C, *ERR, EPS,* DIFF)

Value:

```
{        \
    CALL(mess_vector_convert_dense_to_cholmod(V,&V_CHOLMOD,&
      C));        \
    CALL(mess_vector_convert_cholmod_to_dense(V_CHOLMOD, TMP, &
      C));     \
    cholmod_l_free_dense(&V_CHOLMOD,&C);                                 \
    CALL(mess_vector_diffnorm(V,TMP,&DIFF));                             \
    if(DIFF>EPS){                                                        \
        printf("CHECK_CONVERTCHOLMOD_VECTOR failed:\n");                 \
        CALL(mess_vector_printinfo(V));
      \
        CALL(mess_vector_printshort(V));                                 \
        ERR++;                                                           \
    }                                                                    \
}
```

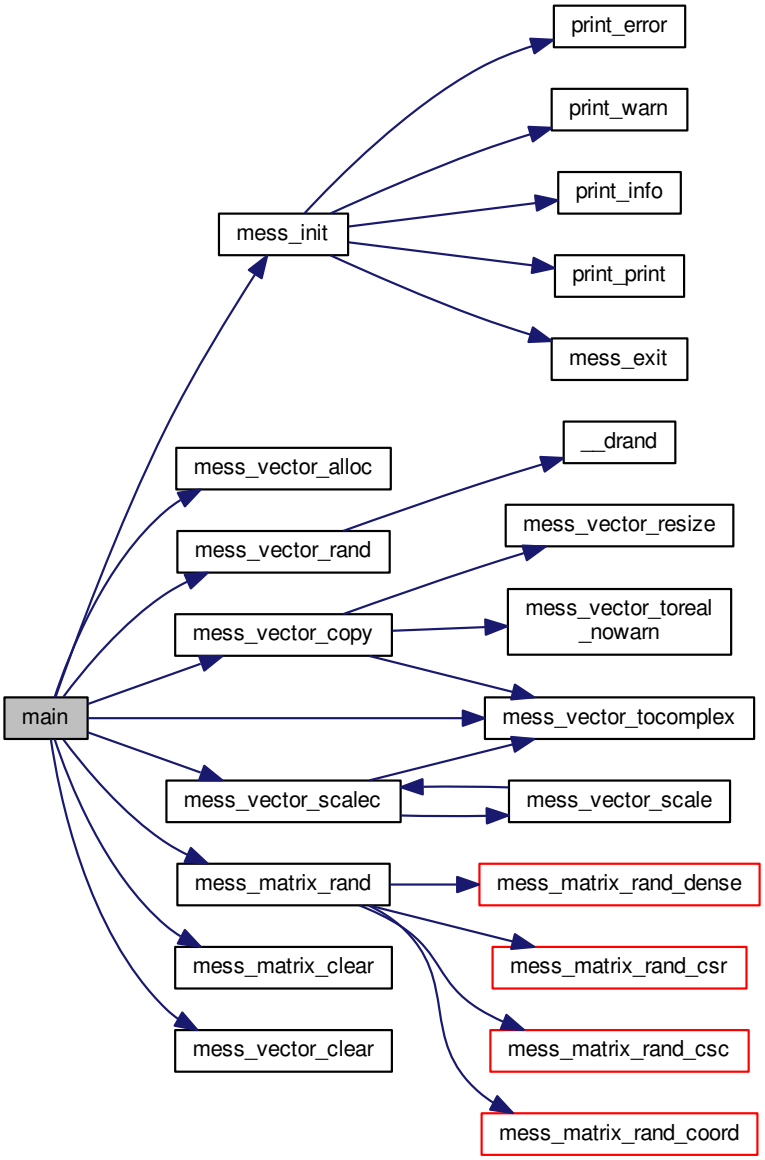
10.54.3 Function Documentation

10.54.3.1 int cholmod_print_dense_all (cholmod_dense ∗ *A*)

10.54.3.2 int cholmod_print_sparse_all (cholmod_sparse ∗ *A*)

10.54.3.3 int main (void)

Here is the call graph for this function:



10.55 check_cimport.c File Reference

Functions

- int main (int argc, char ∗argv[ ])

10.55.1 Detailed Description

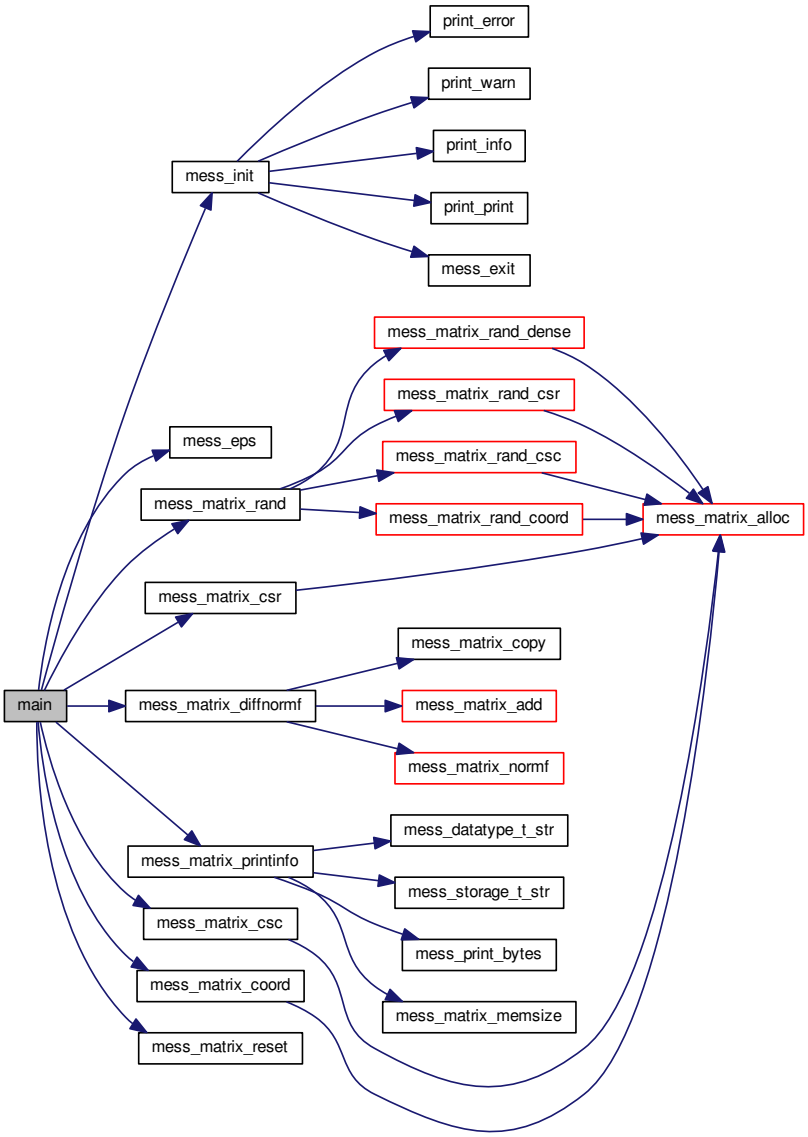
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.55.2 Function Documentation

10.55.2.1 int main (int *argc*, char * *argv[ ]*)

Here is the call graph for this function:



10.56 check_cimport_c2d.c File Reference

Functions

- int check_real ()

  Check the creation of a dense matrix from a real C 2 dimensional array.
- int check_complex ()

  Check the creation of a dense matrix from a complex C 2 dimensional array.
- int main (int argc, char **argv)

10.56.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.56.2 Function Documentation

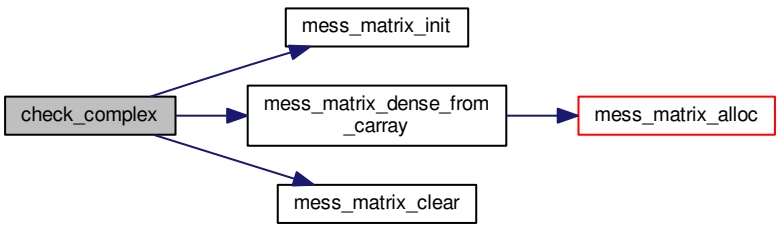
10.56.2.1 int check_complex ( )

Returns

zero on success or a non zero error code

The check_complex function creates a dense mess_matrix from a given complex C 2 dimensional array and checks if this matrix is in the desired storage format.

Here is the call graph for this function:



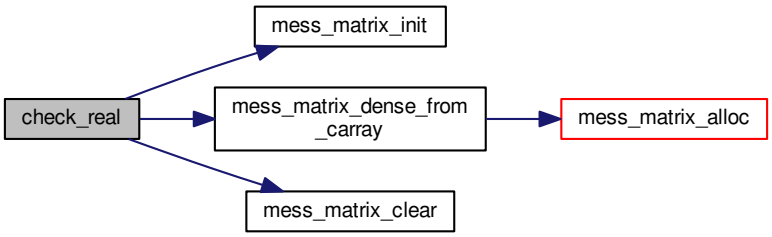
10.56.2.2 int check_real ( )

Returns

zero on success or a non zero error code

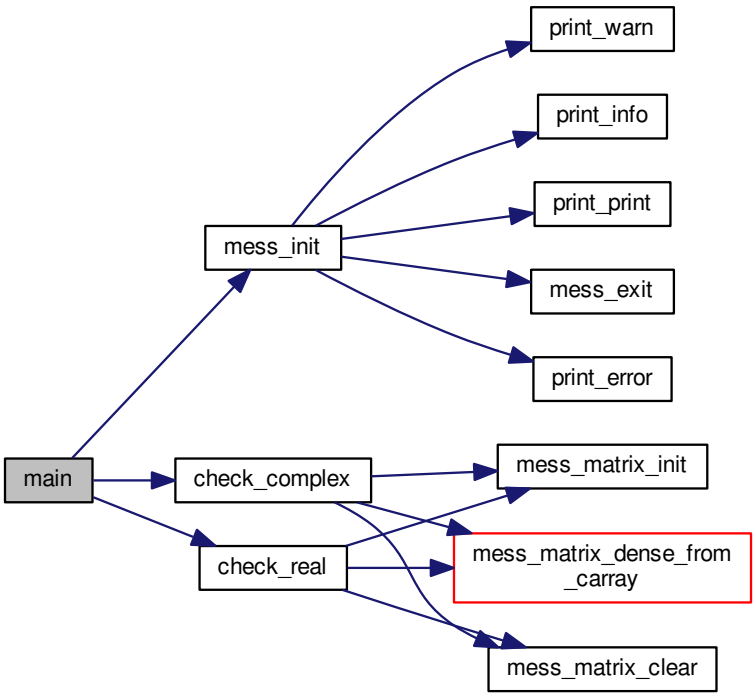
The check_real function creates a dense mess_matrix from a given real C 2 dimensional array and checks if this matrix is in the desired storage format.

Here is the call graph for this function:



10.56.2.3 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.57 check_colaxpy2.c File Reference

Functions

- int main (int argc, char **argv)

10.57.1 Detailed Description

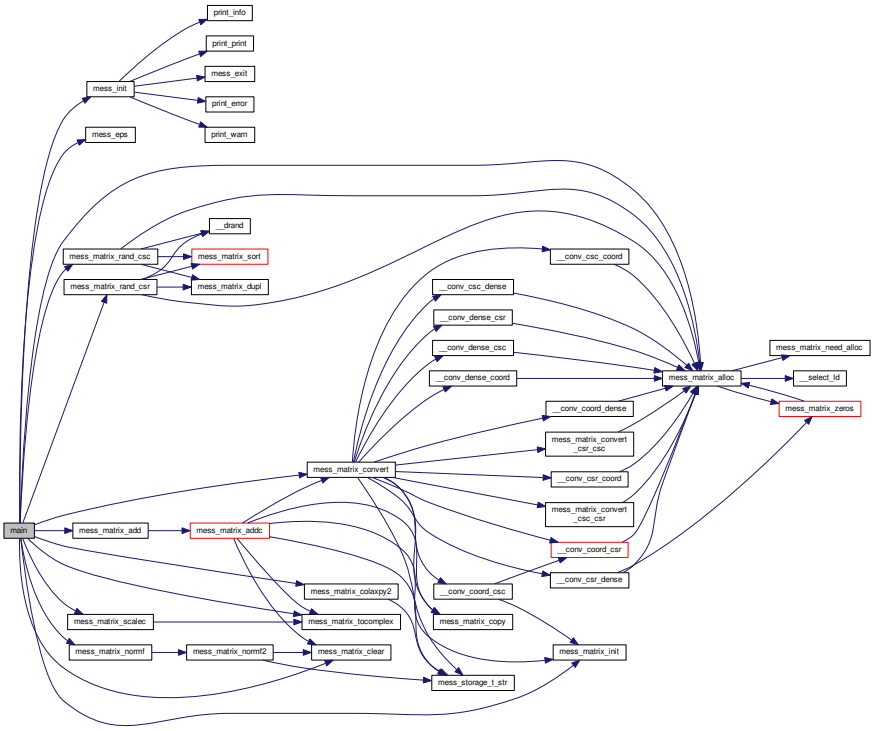
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.57.2 Function Documentation

10.57.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.58 check_coldot.c File Reference

Macros

- #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed ================\n", #X); } else {fprintf(stderr, "test %35s passed.\n",#X); }

Functions

- int main (int argc, char **argv)

Variables

- double matrix_data [16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}

10.58.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

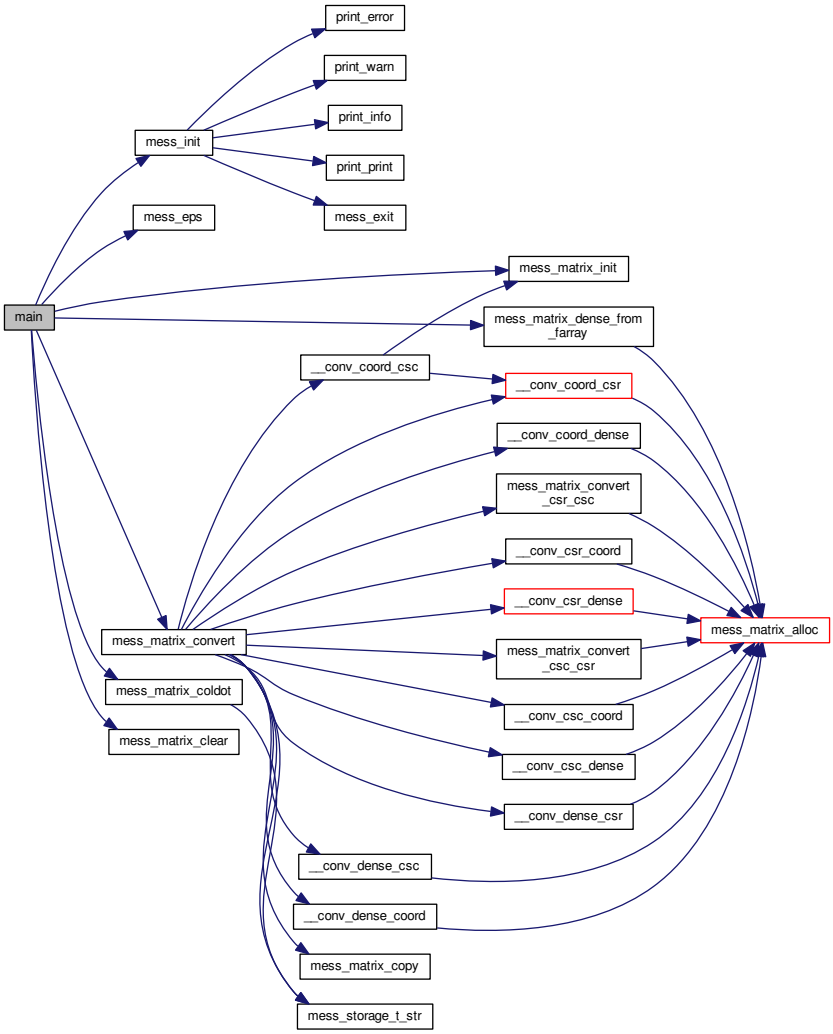
10.58.2 Macro Definition Documentation

10.58.2.1 #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed ================\n", #X); } else {fprintf(stderr, "test %35s passed.\n",#X); }

10.58.3 Function Documentation

10.58.3.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.58.4 Variable Documentation

10.58.4.1 double matrix_data[16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}

10.59 check_coldotc.c File Reference

Functions

- int main (int argc, char ∗∗argv)

Variables

- mess_double_cpx_t matrix_data [16] = {11,0,31,0,12-3∗I,22+I,0,42,0,0,33,43,14,24,0,0}

10.59.1 Detailed Description

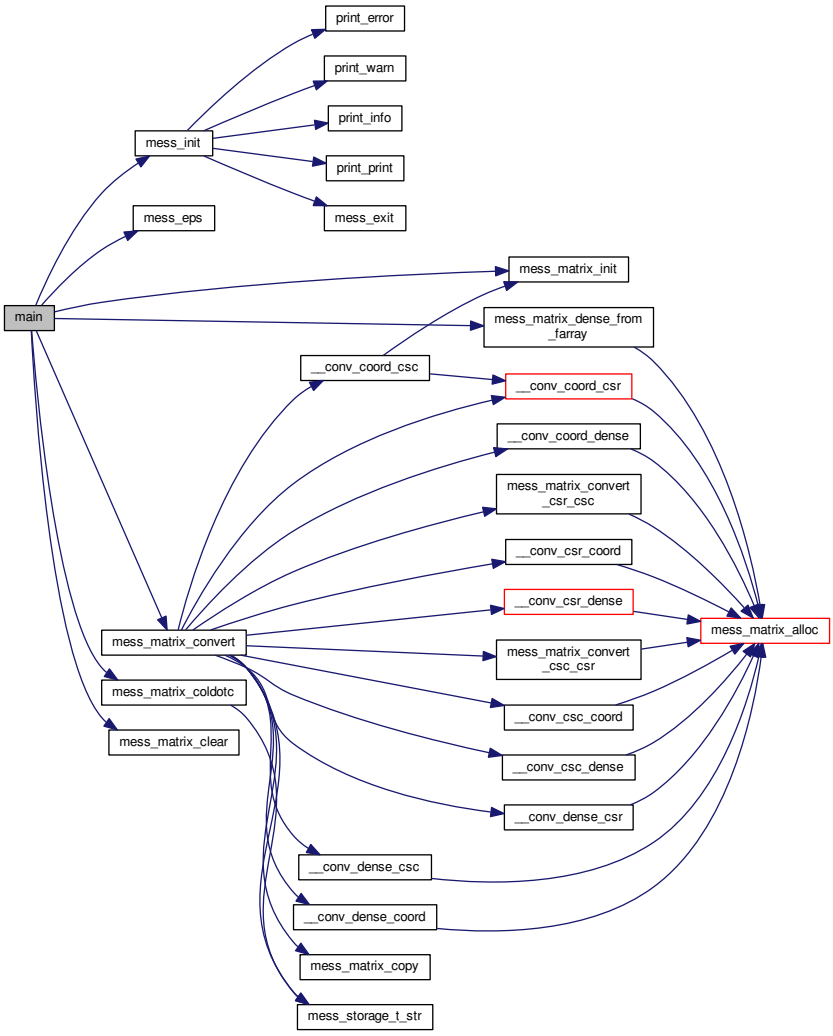
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.59.2 Function Documentation

10.59.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.59.3 Variable Documentation

10.59.3.1 mess_double_cpx_t matrix_data[16] = {11,0,31,0,12-3∗I,22+I,0,42,0,0,33,43,14,24,0,0}

10.60 check_coldotcE.c File Reference

Macros

- #define CHECKCOLDOTCE(Q, E, COL1, COL2, SOL, DOT, ERR, EPS, DIFF)

Functions

- int main (int argc, char **argv)

10.60.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.60.2 Macro Definition Documentation

10.60.2.1 #define CHECKCOLDOTCE(Q, E, COL1, COL2, SOL, DOT, ERR, EPS, DIFF)

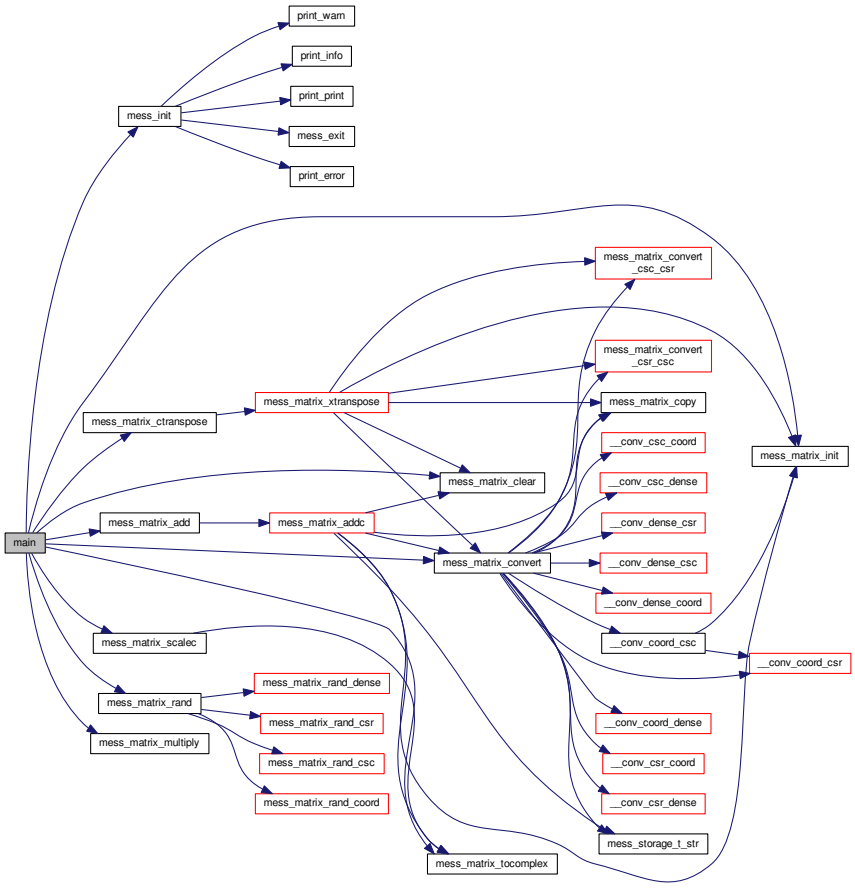
Value:

```
{                   \
    CALL(mess_matrix_coldotcE(Q, E, COL1, COL2, &DOT ))
      \
    DIFF=cabs(SOL->values_cpx[COL1*SOL->ld+COL2]-DOT);                  \
    if(DIFF>EPS){++ERR;                                                 \
        printf("Dot: %e\t + I*%e\n",creal(DOT), cimag(DOT));                \
        printf("Diff%e\n\n",DIFF);                                          \
    }                                                                       \
}
```

10.60.3 Function Documentation

10.60.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.61 check_coldotE.c File Reference

Macros

- #define CHECKCOLVDOTE(Q, E, COL1, COL2, SOL, DOT, ERR, EPS, DIFF)

Functions

- int main (int argc, char **argv)

10.61.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.61.2 Macro Definition Documentation

10.61.2.1 #define CHECKCOLVDOTE(Q, E, *COL1*, *COL2*, *SOL*, *DOT*, *ERR*, *EPS*, DIFF)

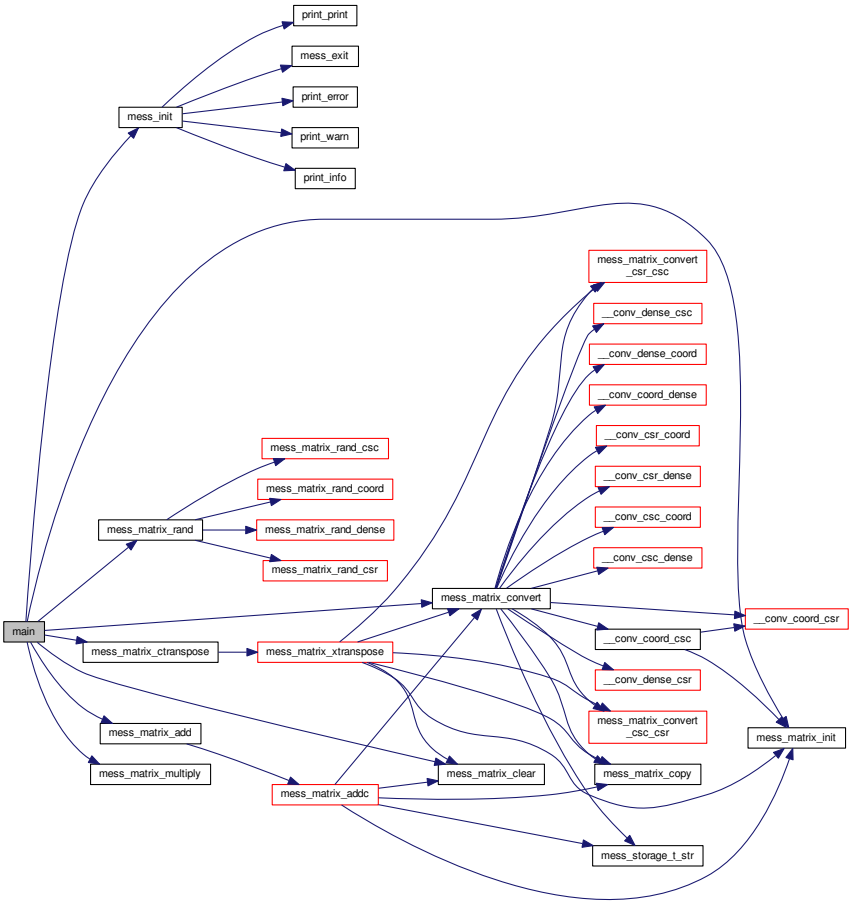
Value:

```
{                    \
    CALL(mess_matrix_coldotE(Q, E, COL1, COL2, &DOT ))
      \
    DIFF=fabs(SOL->values[COL1*SOL->ld+COL2]-DOT);                    \
    printf("Dot: %e\n",DOT);                                              \
    printf("Diff%e\n\n",DIFF);                                        \
    if(DIFF>EPS){++ERR;}                                              \
}
```

10.61.3 Function Documentation

10.61.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.62 check_colnorm.c File Reference

Macros

- #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"========== test %35s failed ==========\n", #X); } else {fprintf(stderr, "test %35s passed.\n",#X); }

Functions

- int main (int argc, char **argv)

Variables

- double matrix_data [16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}

10.62.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

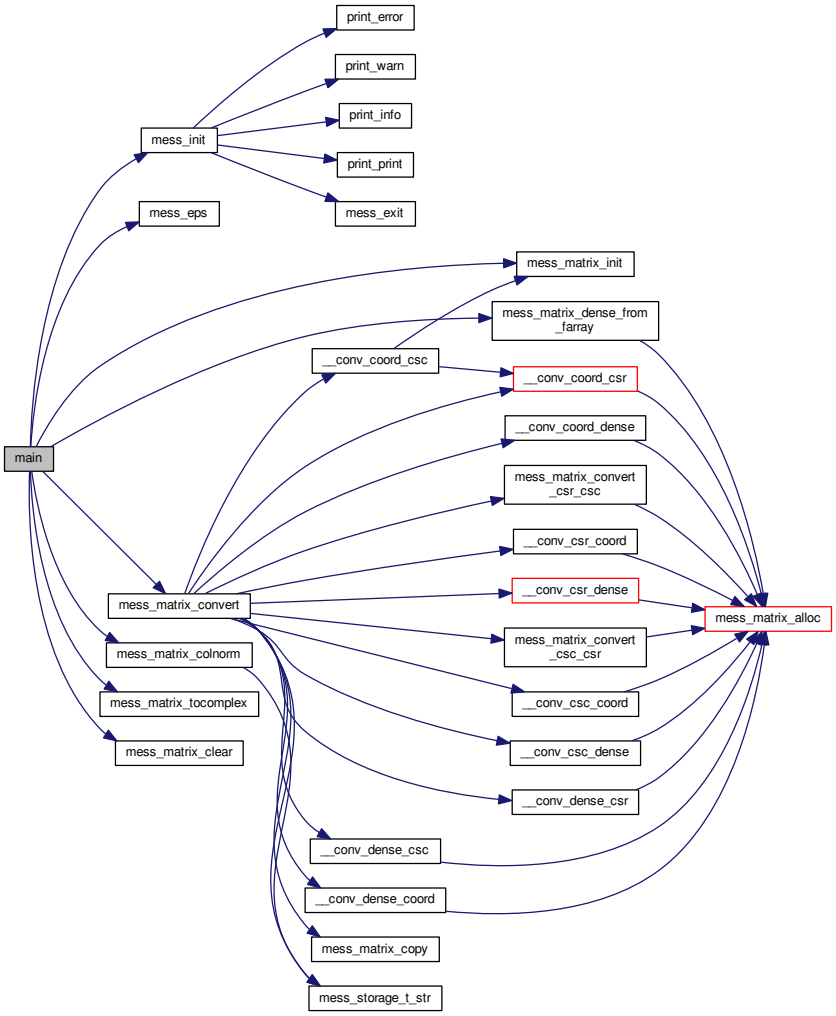
10.62.2 Macro Definition Documentation

10.62.2.1 #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed ================\n", #X); } else {fprintf(stderr, "test %35s passed.\n",#X); }

10.62.3 Function Documentation

10.62.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.62.4 Variable Documentation

10.62.4.1 double matrix_data[16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}

10.63 check_colscale.c File Reference

Macros

- #define CHECKCOLSCALE(A, COL, SCALAR, DIFF, EPS, ERR)

Functions

- int main (int argc, char **argv)

Variables

- double real_matrix_data [9] = {1,2,3,4,5,6,7,8,9}
- mess_double_cpx_t complex_matrix_data [9] = {1,2∗I,3,4∗I,5,6∗I,7,8∗I,9}

10.63.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.63.2 Macro Definition Documentation

10.63.2.1 #define CHECKCOLSCALE(A, COL, SCALAR, DIFF, EPS, ERR)

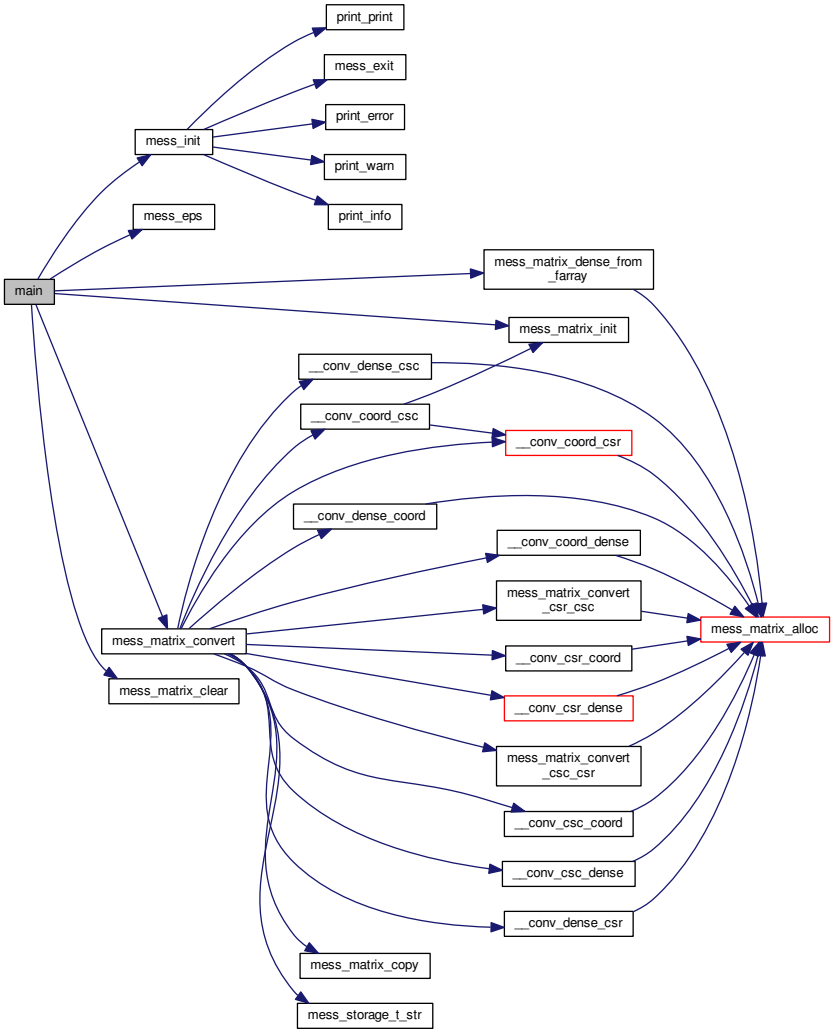
Value:

```
{                                   \
    mess_vector check1, check2;
      \
    MESS_INIT_VECTORS(&check1,&check2);
      \
    mess_vector_alloc(check1, (A)->cols,(A)->data_type);
      \
    mess_vector_alloc(check2, (A)->cols,(A)->data_type);
      \
    mess_matrix_getcol((A),(COL),check1);                                   \
    if(MESS_IS_REAL(check1)){
      \
        mess_vector_scale((SCALAR),check1);                                     \
    }else{                                                                  \
        mess_vector_scalec((SCALAR),check1);
         \
    }                                                                       \
    mess_matrix_colscale((A),(COL),(SCALAR));
      \
    mess_matrix_getcol((A),(COL),check2);
      \
    mess_vector_diffnorm(check1,check2,&(DIFF));                            \
    if((DIFF)>10*(EPS)){++(ERR); printf("diff = %lg\n", DIFF);};
                \
    mess_vector_clear(&check1);
      \
    mess_vector_clear(&check2);                                             \
}
```

10.63.3 Function Documentation

10.63.3.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.63.4 Variable Documentation

10.63.4.1 mess_double_cpx_t complex_matrix_data[9] = {1,2∗I,3,4∗I,5,6∗I,7,8∗I,9}

10.63.4.2 double real_matrix_data[9] = {1,2,3,4,5,6,7,8,9}

10.64 check_colsub.c File Reference

Macros

- #define CHECKROWSUB(A, TEMP, ROW1, ROW2, SOL, ERR, EPS, DIFF)

Functions

- int main (int argc, char **argv)

10.64.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.64.2 Macro Definition Documentation

10.64.2.1 #define CHECKROWSUB(A, *TEMP, ROW1, ROW2, SOL, ERR, EPS,* DIFF)

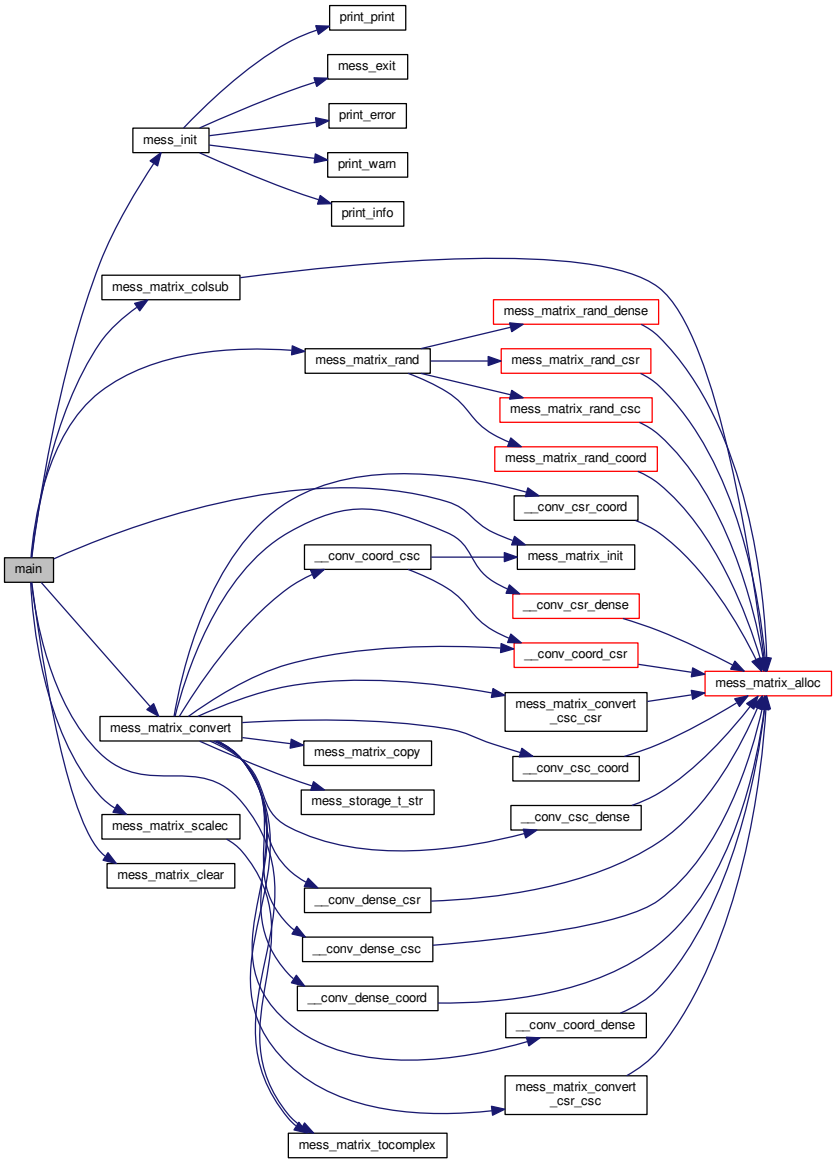
Value:

```
{                                   \
    mess_matrix_colsub(A, ROW1, ROW2, TEMP);
      \
    mess_matrix_diffnorm(SOL,TEMP,&DIFF);
        \
    /*mess_matrix_print(temp);*/                                        \
    printf("%e\n",DIFF);                                                \
    if(DIFF>EPS){++ERR;}                                                \
}
```

10.64.3 Function Documentation

10.64.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.65 check_colvecaxpy.c File Reference

Macros

- #define COMPUTESOL(A, COL, COEFF, V, VSOL, VTEMP)
- #define CHECKCOLVCAXPY(A, COL, COEFF, VCHECK, V, VSOL, NRM, ERR, EPS)

Functions

- int main (int argc, char **argv)

10.65.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.65.2 Macro Definition Documentation

10.65.2.1 #define CHECKCOLVCAXPY(A, COL, COEFF, VCHECK, V, VSOL, NRM, ERR, EPS)

Value:

```
{   \
    CALL(mess_vector_copy(V,VCHECK))
      \
    CALL(mess_matrix_colvecaxpy(COEFF,COL,A,VCHECK));                    \
    /*mess_vector_print(VCHECK);*/                                        \
    /*mess_vector_print(VSOL);*/                                          \
    CALL(mess_vector_diffnorm(VCHECK,VSOL,&NRM));                         \
    printf("%e\n",NRM);                                                   \
    if(NRM>10*EPS){++ERR;}                                                \
}
```

10.65.2.2 #define COMPUTESOL(A, COL, COEFF, V, VSOL, VTEMP)

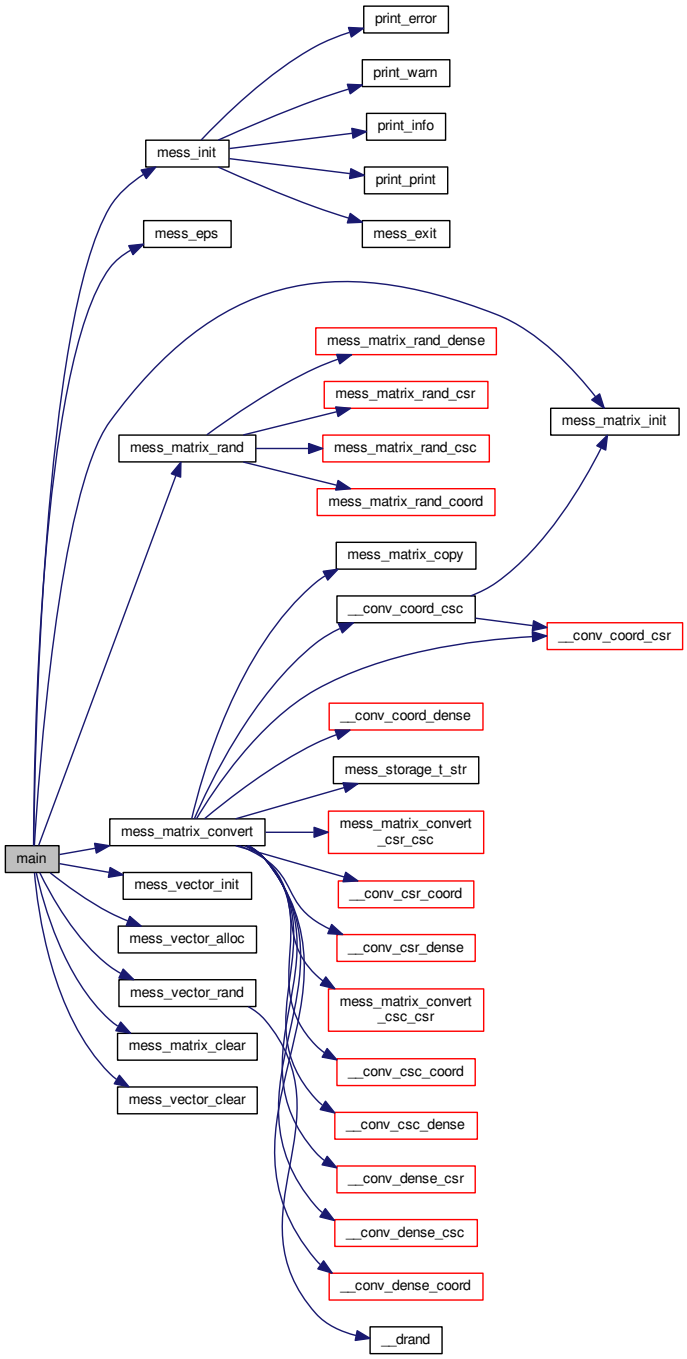
Value:

```
{                                 \
    CALL(mess_vector_copy(V,VSOL));
      \
    CALL(mess_matrix_getcol(A,COL,VTEMP));
      \
    CALL(mess_vector_tocomplex(VTEMP));
      \
    CALL(mess_vector_tocomplex(VSOL));
      \
    CALL(mess_vector_axpyc(COEFF,VTEMP,VSOL));                            \
}
```

10.65.3 Function Documentation

10.65.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.66 check_colvecdot.c File Reference

Functions

- int main (int argc, char **argv)

Variables

- double matrix_data [16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}

10.66.1 Detailed Description

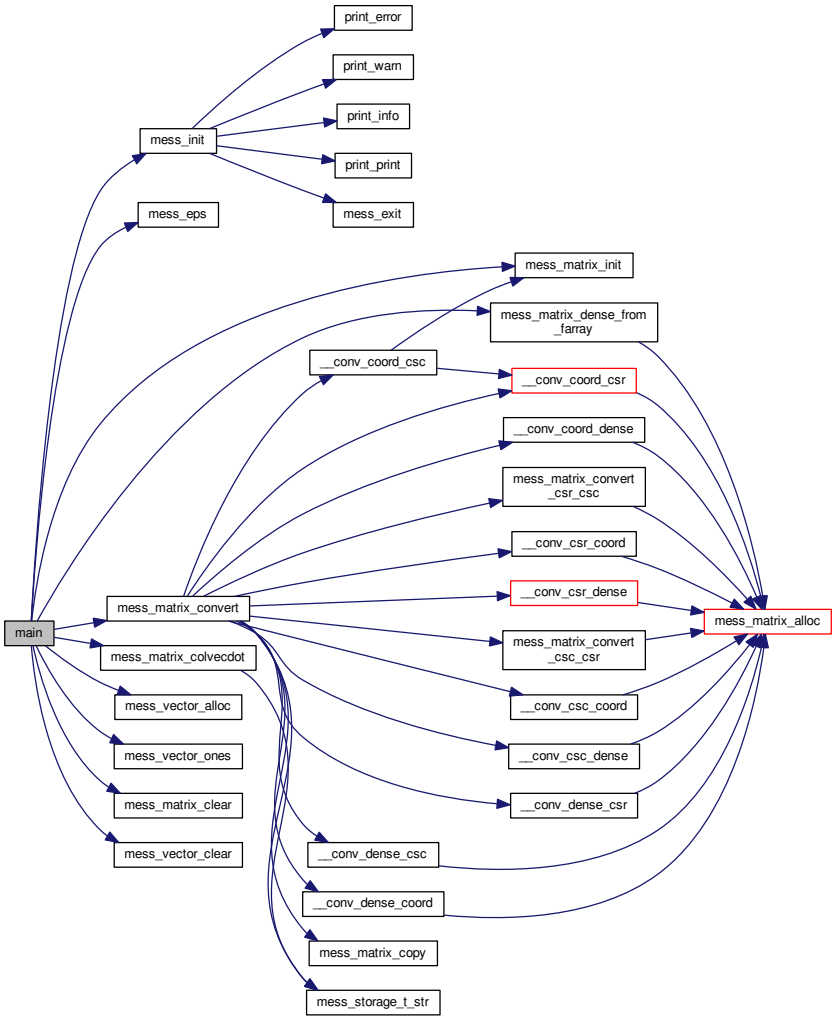
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.66.2 Function Documentation

10.66.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.66.3 Variable Documentation

10.66.3.1 double matrix_data[16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}

10.67 check_colvecdotc.c File Reference

Functions

- int main (int argc, char **argv)

Variables

- mess_double_cpx_t matrix_datac [16] = {11,0,31,0,12,22,0,42,0,0,33,43+40*I,14,24,0,0}
- double matrix_datar [16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}

10.67.1 Detailed Description

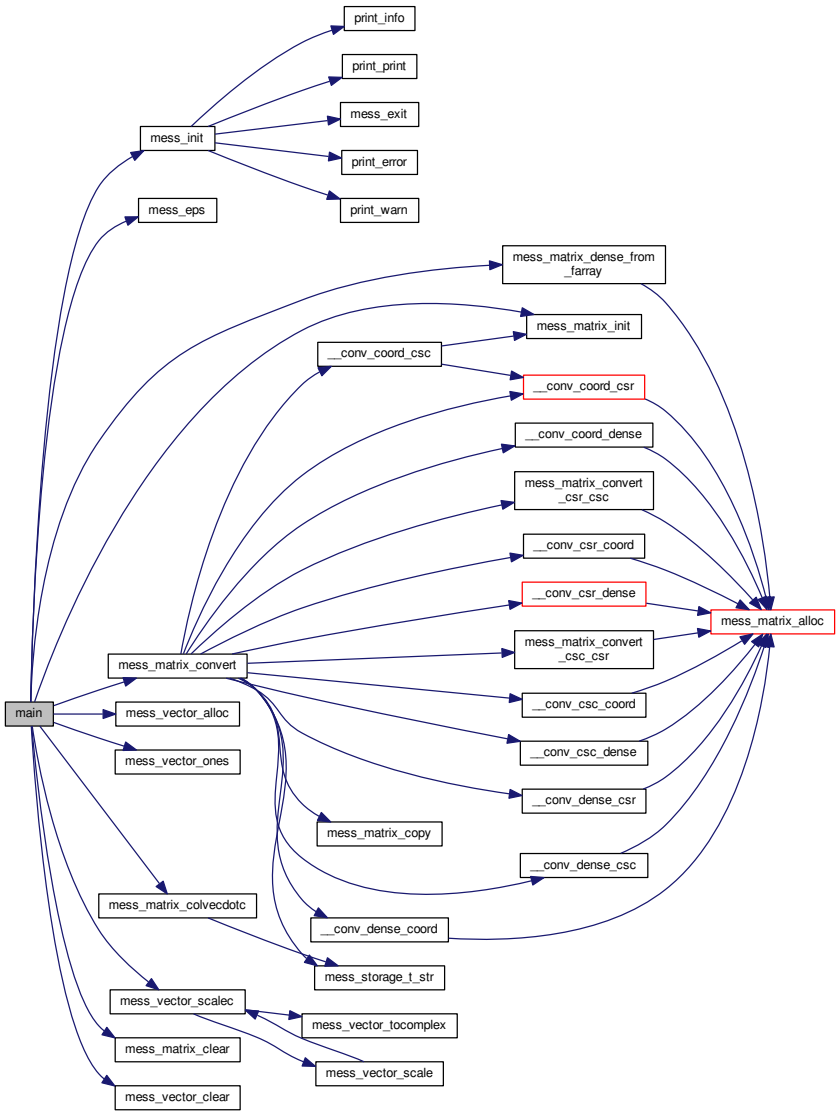
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.67.2 Function Documentation

10.67.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.67.3 Variable Documentation

10.67.3.1 mess_double_cpx_t matrix_datac[16] = {11,0,31,0,12,22,0,42,0,0,33,43+40∗I,14,24,0,0}

10.67.3.2 double matrix_datar[16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}

10.68 check_condest.c File Reference

Macros

- #define CHECKCONDEST(A, NRM, SOL, ERR, EPS)

Functions

- int main (int argc, char **argv)

Variables

- double rand50x50condest = 1.962107663241329e+03

10.68.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.68.2 Macro Definition Documentation

10.68.2.1 #define CHECKCONDEST(A, NRM, SOL, ERR, EPS)

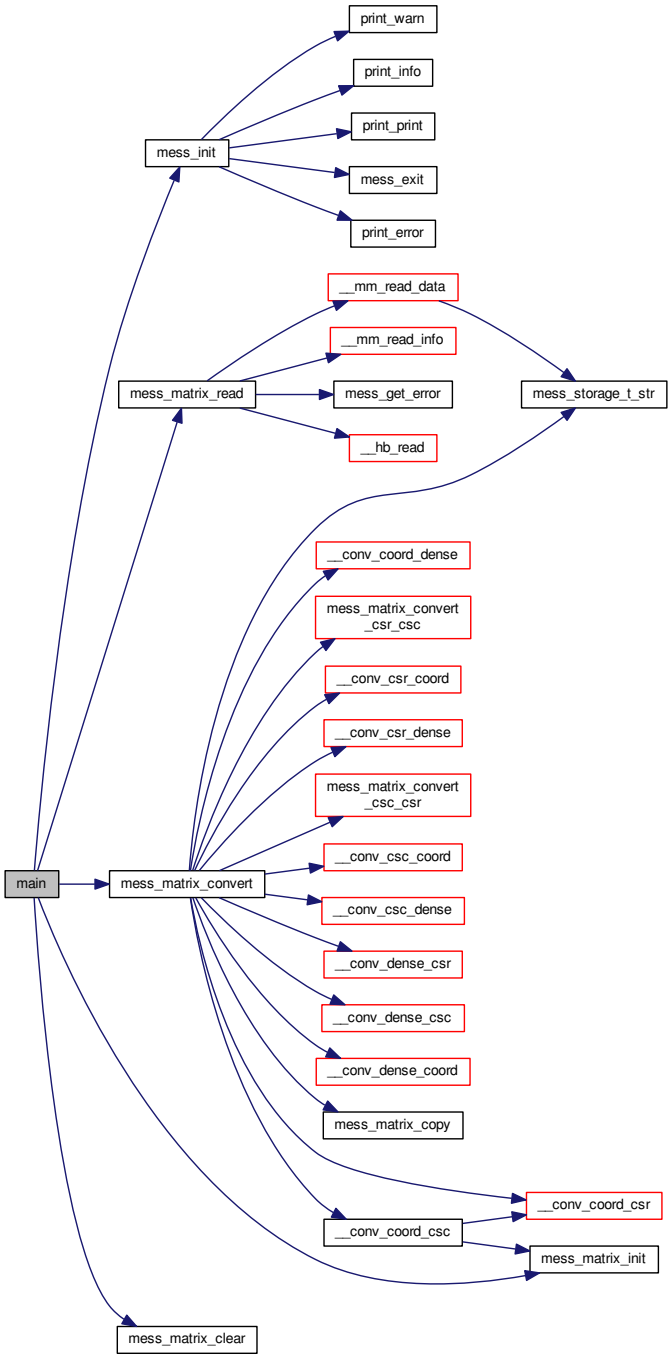
Value:

```
{                                                   \
    CALL(mess_matrix_condest(A,&NRM));
            \
    if(fabs(NRM-SOL)>EPS){                                                          \
        printf("Failed Nrm: %e \t Sol: %e \t Err: %e\n",NRM,SOL,fabs(NRM-SOL));     \
        ++ERR;                                                                      \
    }                                                                               \
}
```

10.68.3 Function Documentation

10.68.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.68.4 Variable Documentation

10.68.4.1 double rand50x50condest = 1.962107663241329e+03

10.69 check_convert.c File Reference

Functions

- int check (mess_matrix A, unsigned short from, unsigned short to)

  Check the conversion function.
- int main (int argc, char **argv)

10.69.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

- CSR → CSR
- CSR → CSC
- CSR → DENSE
- CSR → COORD
- CSC → CSC
- CSC → CSR
- CSC → DENSE
- CSC → COORD
- COORD → COORD
- COORD → CSR
- COORD → CSC
- COORD → DENSE
- DENSE → DENSE
- DENSE → CSR
- DENSE → CSC
- DENSE → COORD

10.69.2 Function Documentation

10.69.2.1 int check (mess_matrix *A*, unsigned short *from*, unsigned short *to*)

Parameters

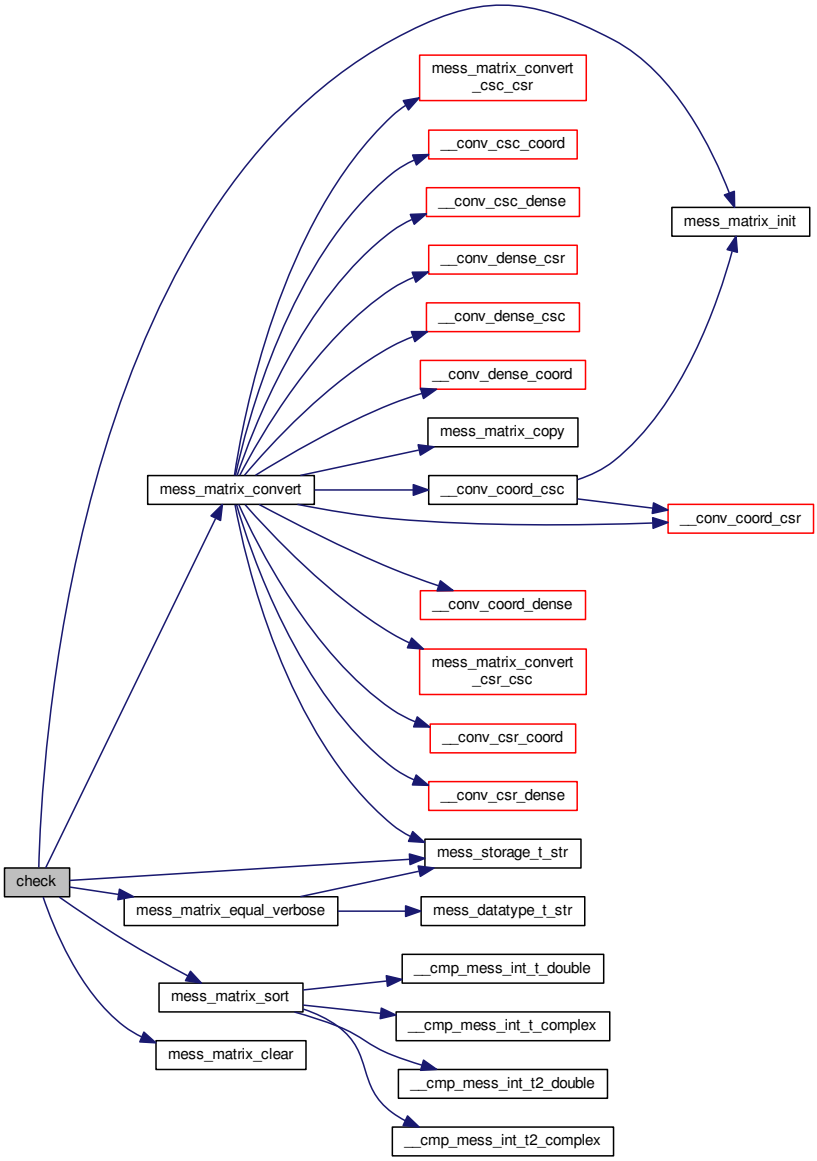
in	*A*	input matrix
in	*from*	input type
in	*to*	input output type

Returns

zero on success or a non zero error code otherwise

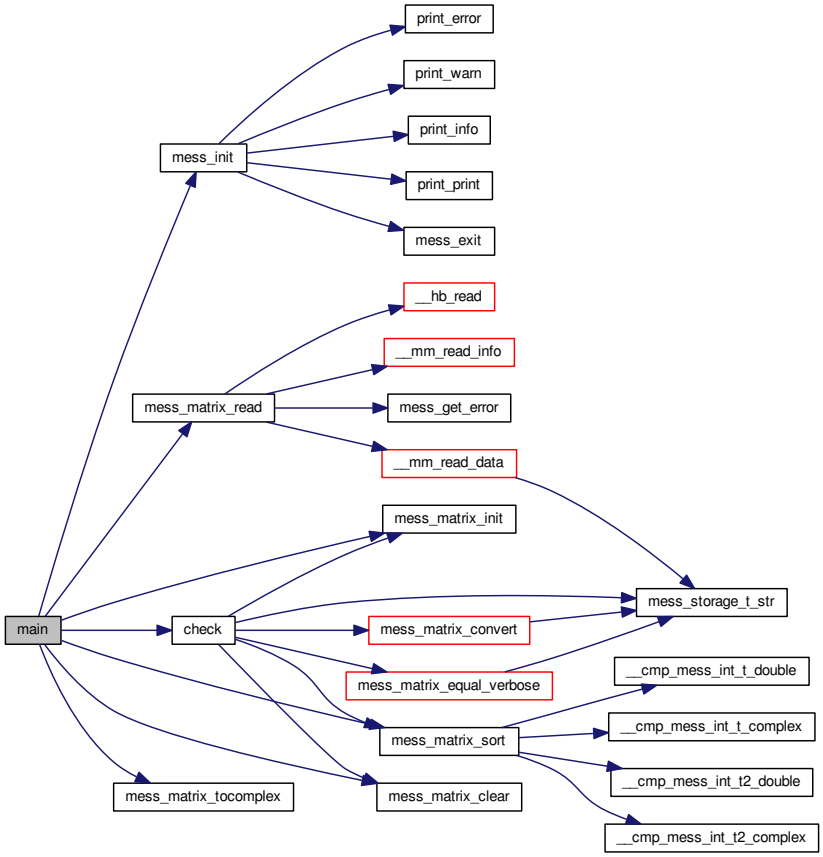
The check function converts a matrix A to the desired output type to using mess_matrix_convert and checks if the computed matrix is in the desired storage format.

Here is the call graph for this function:



10.69.2.2 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.70 check_dae2_handles_adi.c File Reference

Macros

- #define CHECK_FUNCTION(INFO, FNAME, DIFFFUNC, Y1, Y2, DIFF, ERR, EPS)

Functions

- int main (int argc, char **argv)

10.70.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

See also

mess_equation_glyap_dae2

10.70.2 Macro Definition Documentation

10.70.2.1 #define CHECK_FUNCTION(INFO, *FNAME*, *DIFFFUNC*, *Y1*, *Y2*, DIFF, *ERR*, *EPS*)

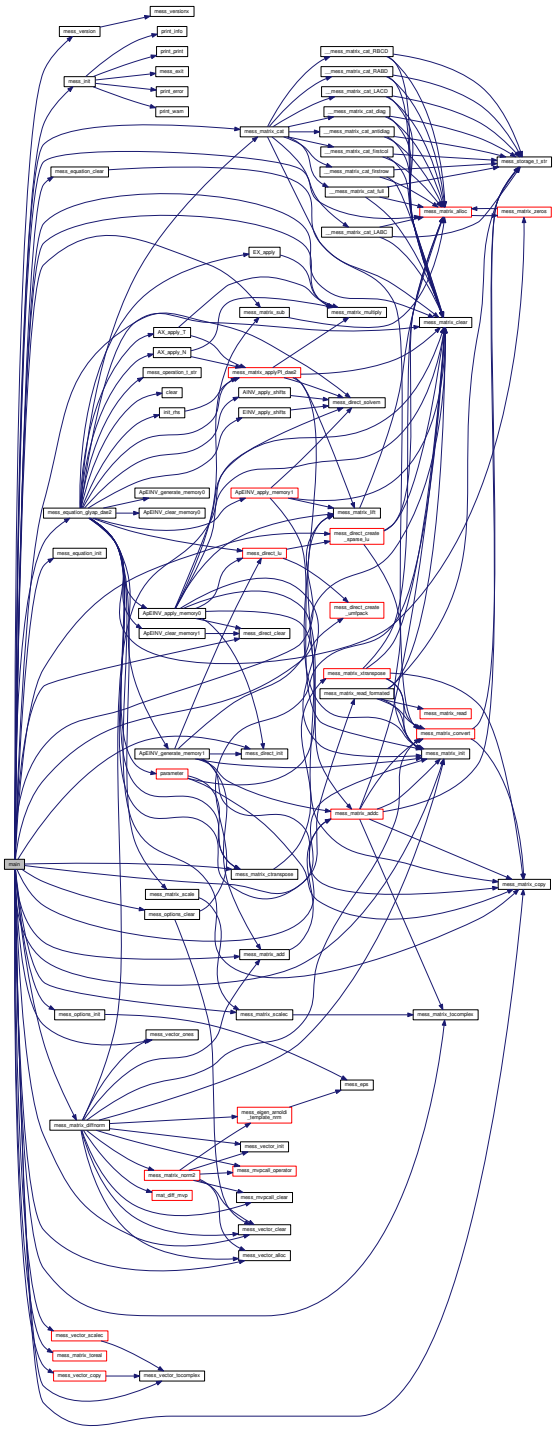
Value:

```
CALL(DIFFFUNC(Y1,Y2,&DIFF));                                    \
        if(DIFF>EPS){                                           \
            printf(INFO " " #FNAME " failed\tdiff=%e\n",diff);  \
            ERR++;                                              \
        }else{                                                  \
            printf(INFO " " #FNAME " passed with diff=%e.\n",diff);  \
        }                                                       \
```

10.70.3 Function Documentation

10.70.3.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.71 check_dae2_handles_para.c File Reference

Macros

- #define CHECK_FUNCTION(INFO, FNAME, DIFFFUNC, Y1, Y2, DIFF, ERR, EPS)

Functions

- int main (int argc, char **argv)

10.71.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

See also

mess_equation_glyap_dae2

10.71.2 Macro Definition Documentation

10.71.2.1 #define CHECK_FUNCTION(INFO, *FNAME*, *DIFFFUNC*, *Y1*, *Y2*, DIFF, *ERR*, *EPS*)

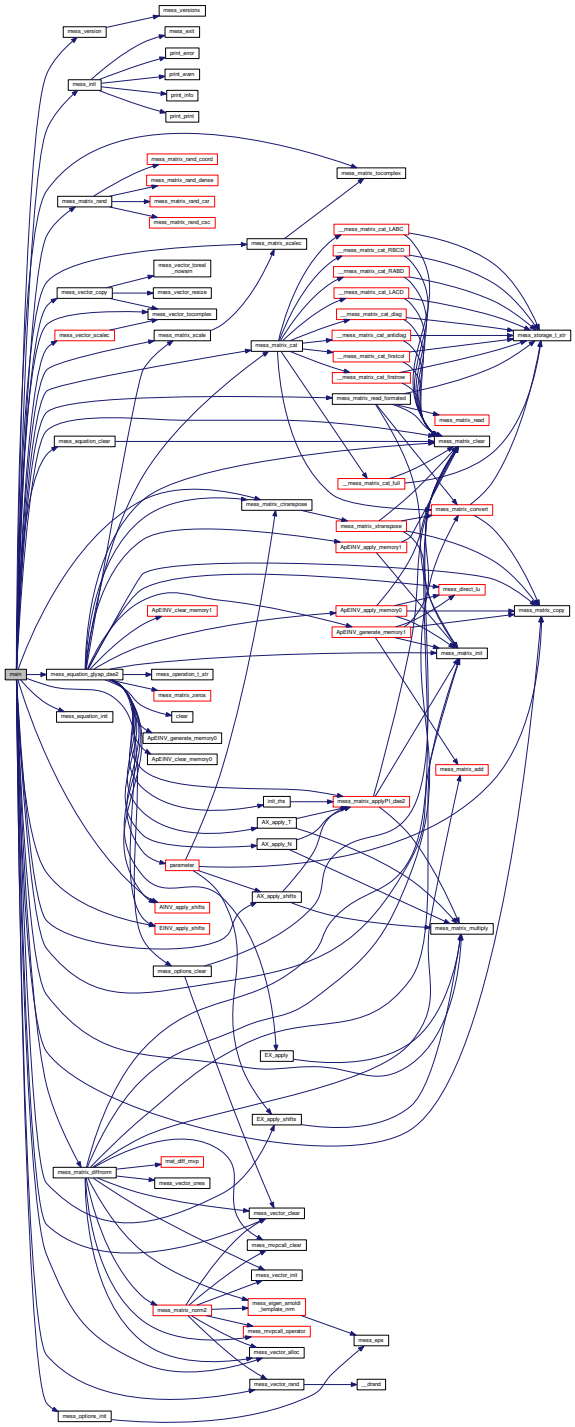
Value:

```
CALL(DIFFFUNC(Y1,Y2,&DIFF));                                    \
if(DIFF>EPS){                                                   \
    printf(INFO " " #FNAME " failed\tdiff=%e\n",diff);          \
    ERR++;                                                      \
}else{                                                          \
    printf(INFO " " #FNAME  " passed with diff=%e.\n",diff);    \
}                                                               \
```

10.71.3 Function Documentation

10.71.3.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.72 check_decomp.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.72.1 Detailed Description

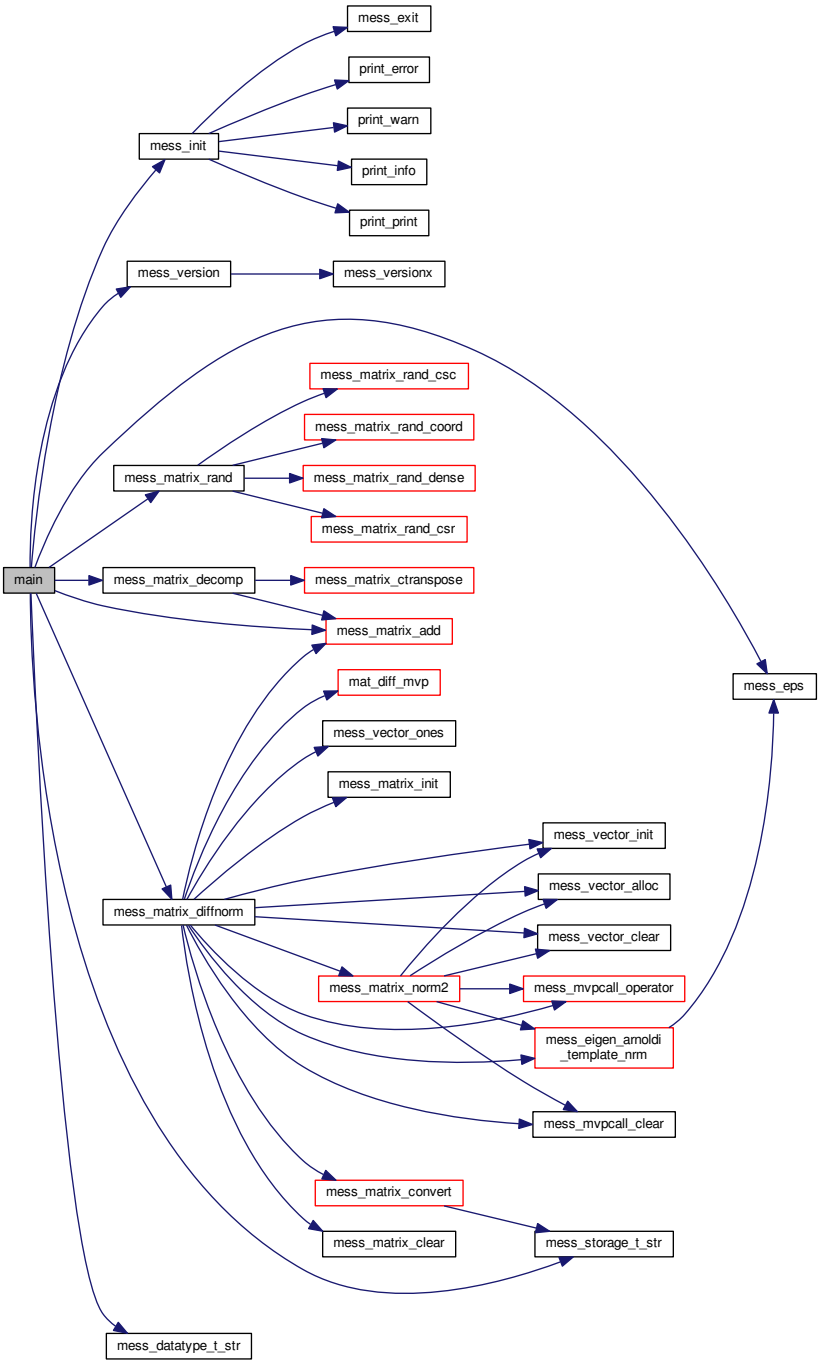
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.72.2 Function Documentation

10.72.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.73 check_dense_nm.c File Reference

Functions

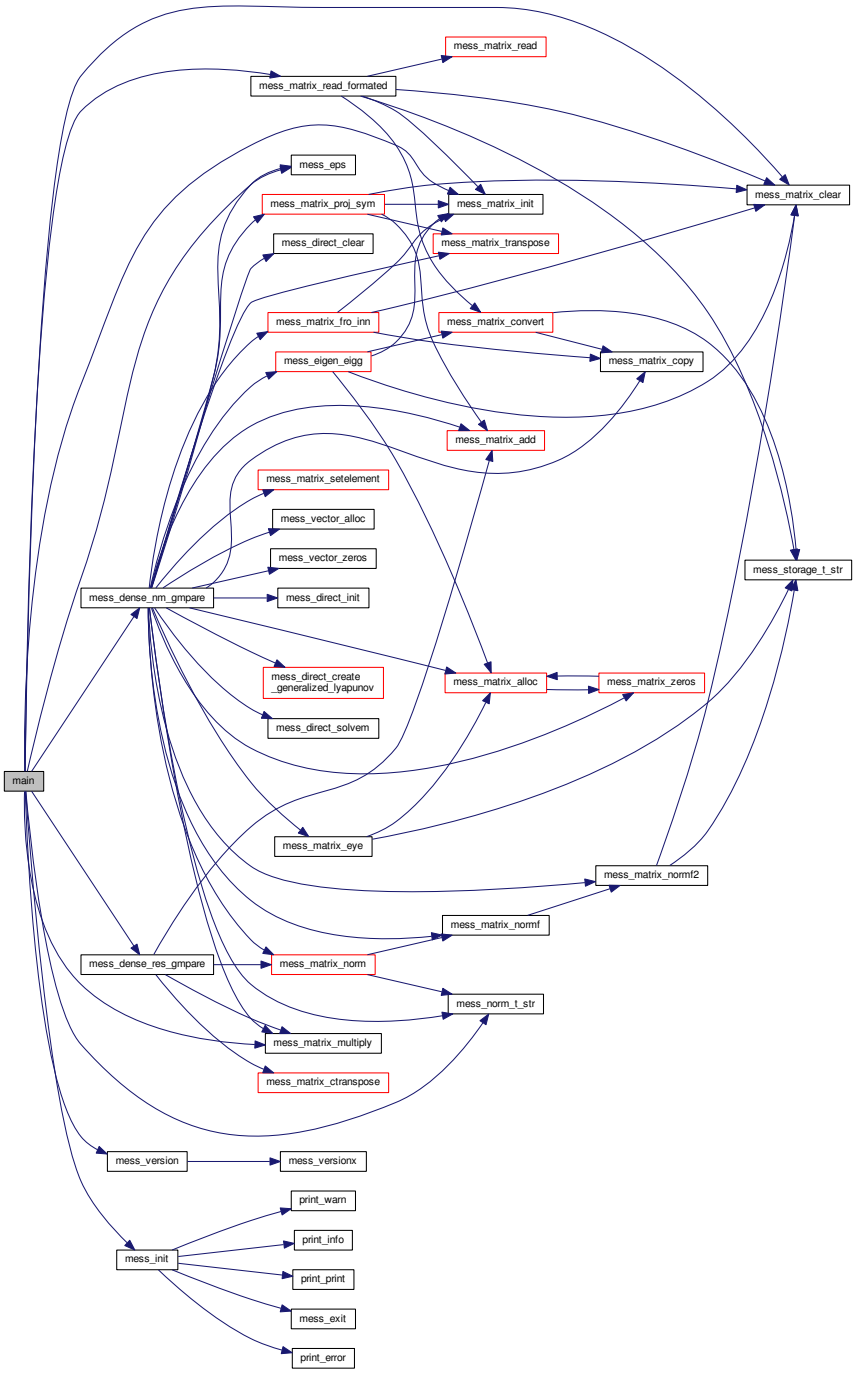
- int main (int argc, char **argv)

10.73.1 Detailed Description

10.73.2 Function Documentation

10.73.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.74 check_dense_nm_tridiag.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.74.1 Detailed Description

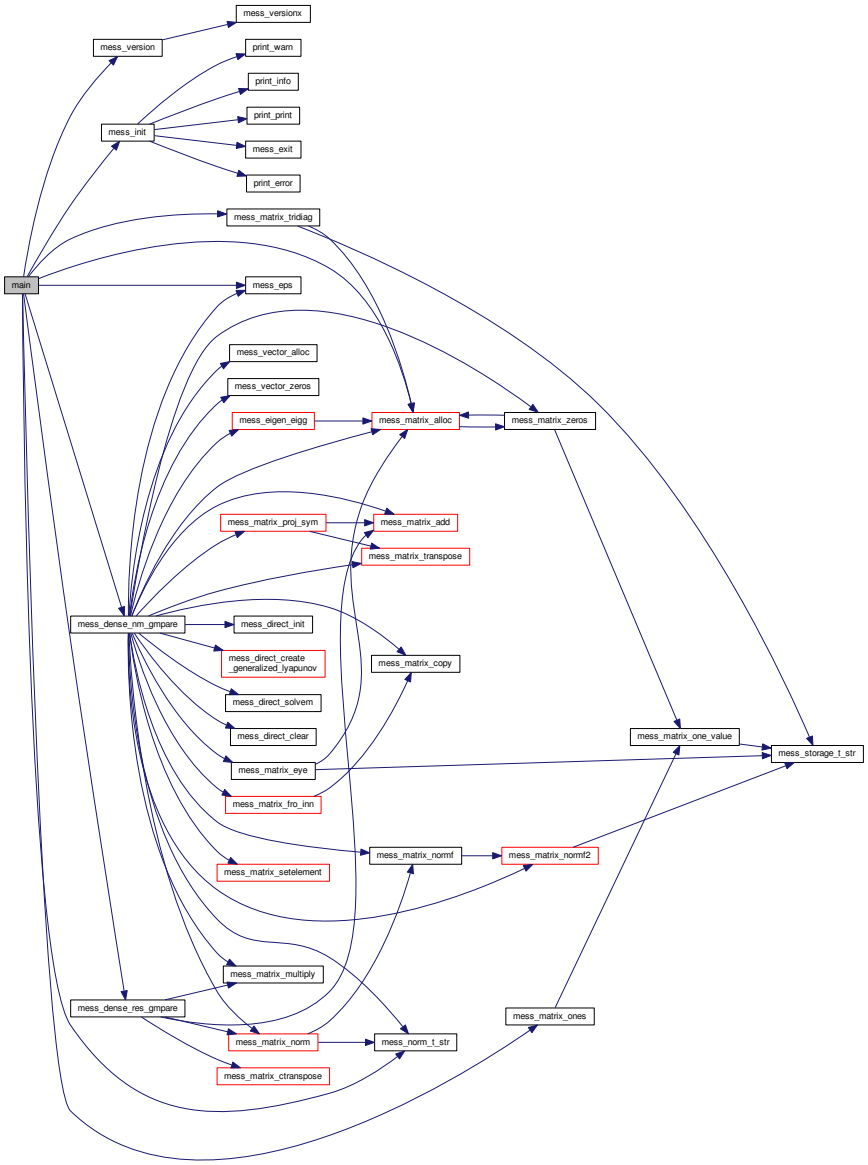
Attention

The Newton method does not converge for this instance using the positive Riccati equation.

10.74.2 Function Documentation

10.74.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.75 check_det.c File Reference

Functions

- int test_det (mess_direct sol, double ∗det_check, mess_double_cpx_t ∗detc_check)
- int main (int argc, char ∗∗argv)

10.75.1 Detailed Description

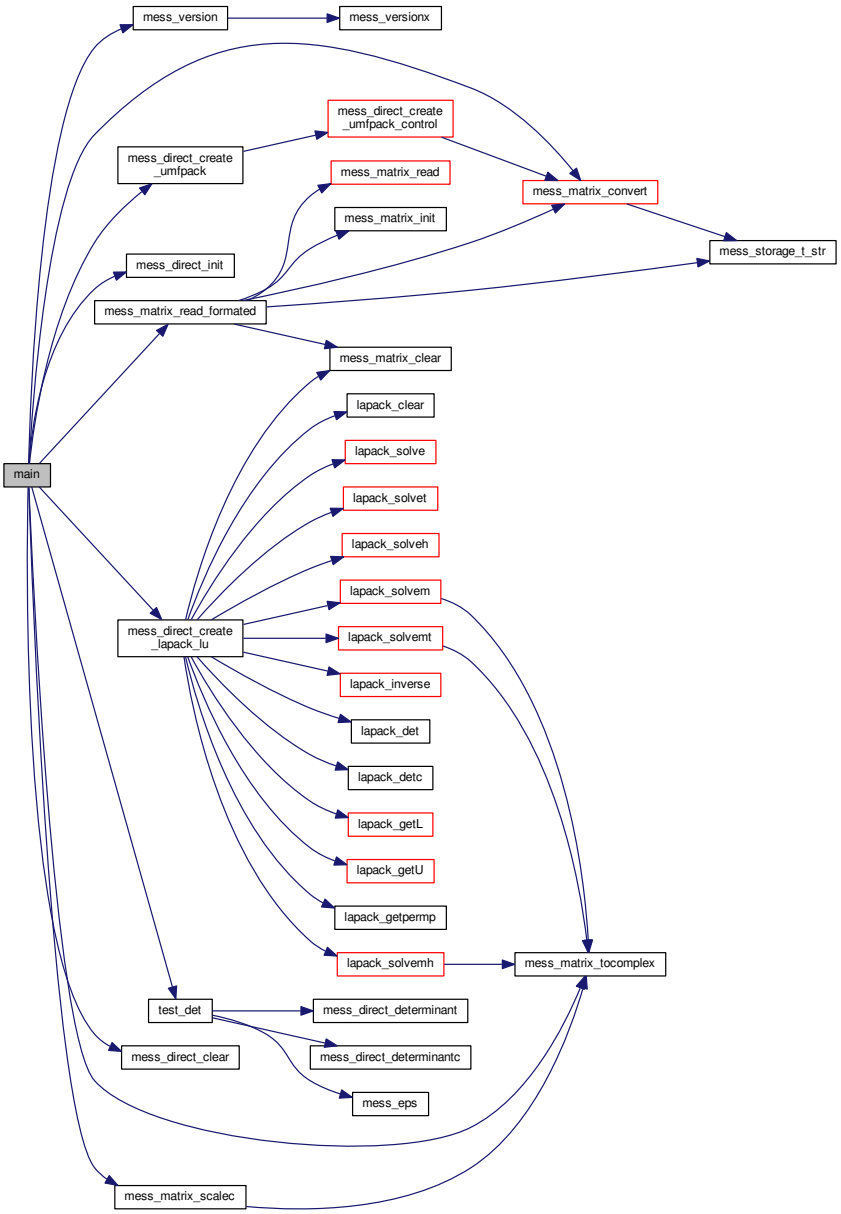
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.75.2 Function Documentation

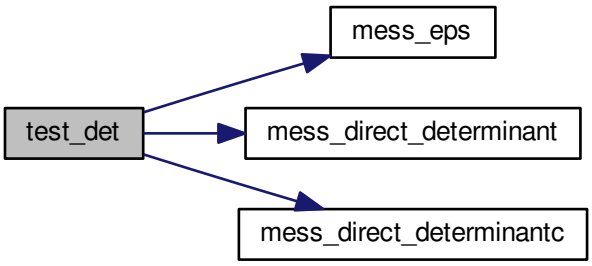
10.75.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.75.2.2 int test_det (mess_direct *sol*, double * *det_check*, mess_double_cpx_t * *detc_check*)

Here is the call graph for this function:



10.76 check_dgeqp3.c File Reference

Functions

- int main (int argc, char **argv)

10.76.1 Detailed Description

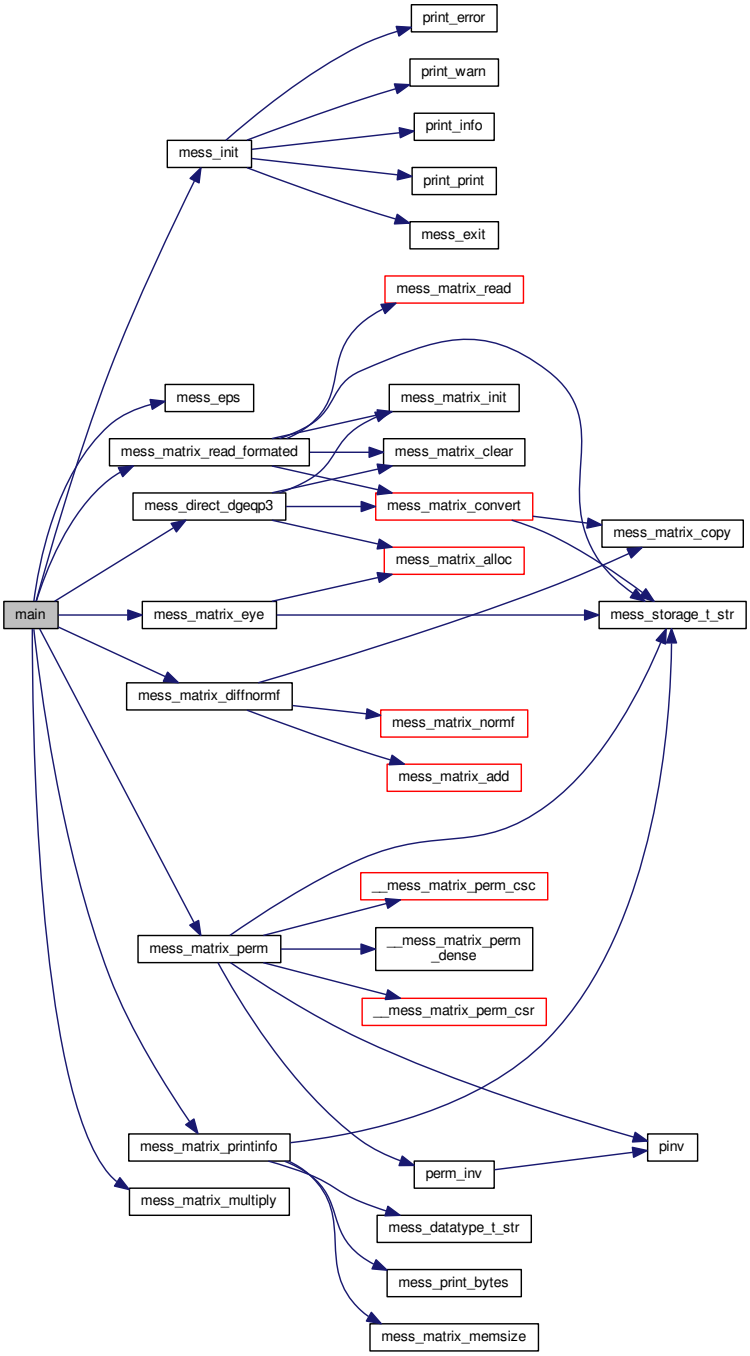
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.76.2 Function Documentation

10.76.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.77 check_diag.c File Reference

Macros

- #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed ================\n", #X); } else {fprintf(stderr, "test %35s passed.\n",#X); }

Functions

- int main (int argc, char **argv)

Variables

- double matrix_data [16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}
- mess_double_cpx_t matrix_datac [16] = {11,0,31,0,12-3*I,22+I,0,42,0,0,33,43,14,24,0,0}

10.77.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

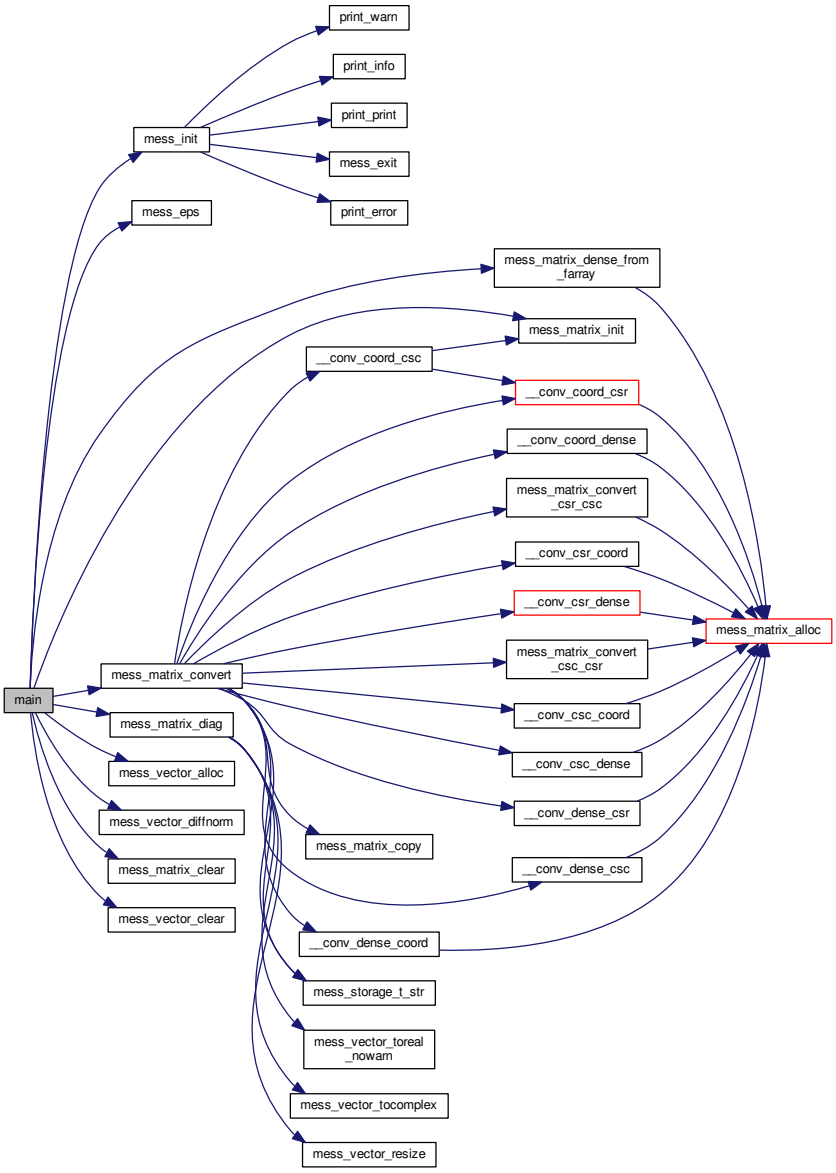
10.77.2 Macro Definition Documentation

10.77.2.1 #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed ================\n", #X); } else {fprintf(stderr, "test %35s passed.\n",#X); }

10.77.3 Function Documentation

10.77.3.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.77.4 Variable Documentation

10.77.4.1 double matrix_data[16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}

10.77.4.2 mess_double_cpx_t matrix_datac[16] = {11,0,31,0,12-3∗I,22+I,0,42,0,0,33,43,14,24,0,0}

10.78 check_direct_care.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.78.1 Detailed Description

Author

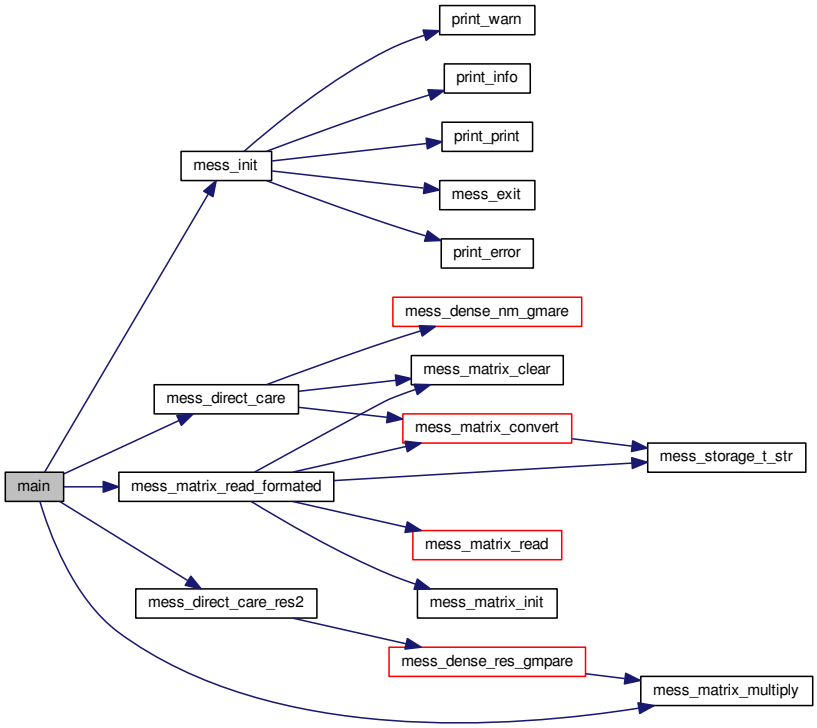
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Checks the mess_direct_care and mess_direct_care_res2 function.

10.78.2 Function Documentation

10.78.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.79 check_dynorm2.c File Reference

Functions

- double fdmcol (double x, double y)

  Determine values used in a FDM matrix.
- int main (int argc, char **argv)

10.79.1 Detailed Description

10.79.2 Function Documentation

10.79.2.1 double fdmcol (double *x*, double *y*)

Parameters

in	*x*	input space variable x
in	*y*	input space variable y

Returns

1 if x is between 0.1 and 0.4, otherwise 0

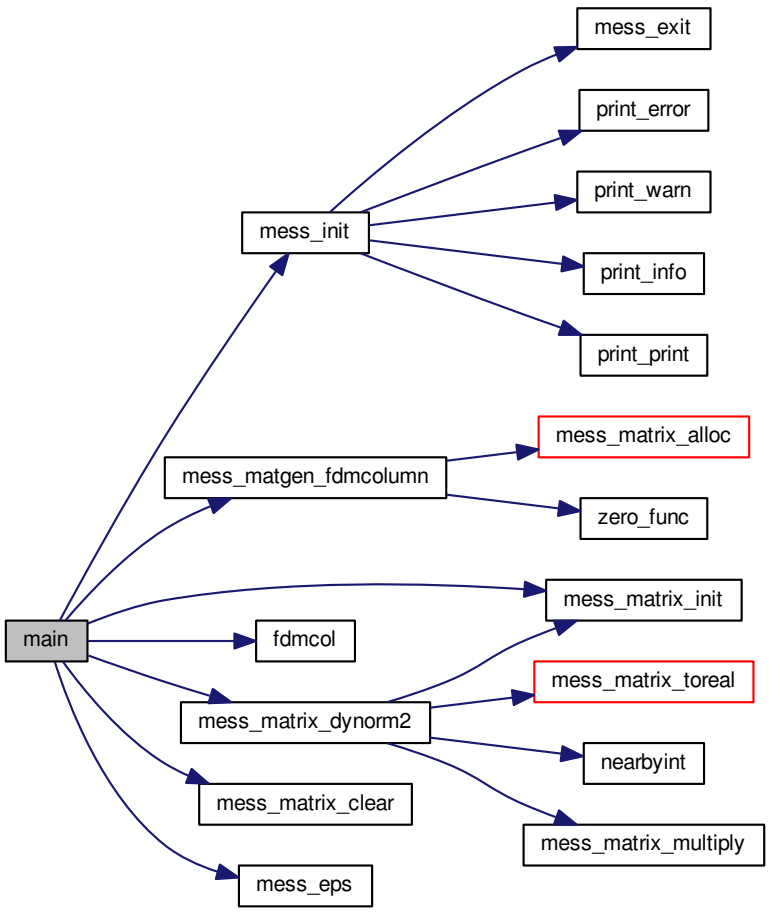
The fdmcol function returns 1 if the space variable x fullfills

$$0.1 < x < 0.4,$$

otherwise 0.

10.79.2.2 int main (int *argc*, char $**$ *argv*)

Here is the call graph for this function:



10.80 check_dynorm2t.c File Reference

Functions

- double fdmcol (double x, double y)

  Determine values used in a FDM matrix.
- int main (int argc, char $**$argv)

10.80.1 Detailed Description

10.80.2 Function Documentation

10.80.2.1 double fdmcol (double *x*, double *y*)

Parameters

in	*x*	input space variable x
in	*y*	input space variable y

Returns

1 if x is between 0.1 and 0.4, otherwise 0

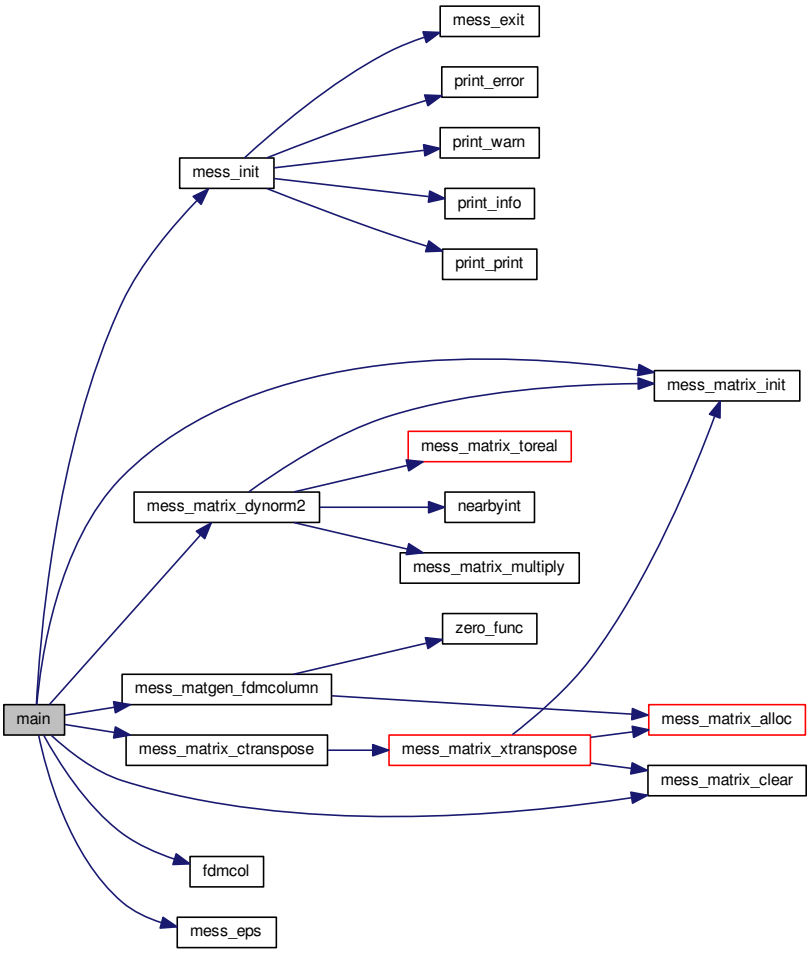
The fdmcol function returns 1 if the space variable x fullfills

$$0.1 < x < 0.4,$$

otherwise 0.

10.80.2.2 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.81 check_dynorm_indefinite2.c File Reference

Functions

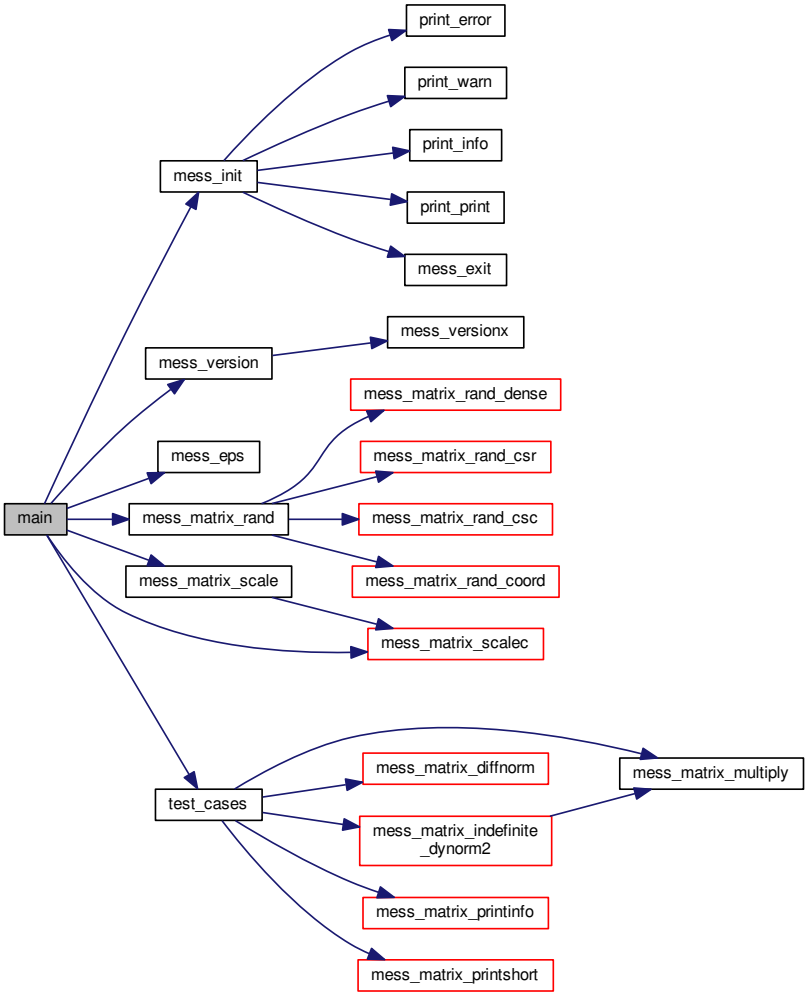
- static int test_cases (mess_matrix W, mess_matrix K, double tol)
- int main (int argc, char ∗∗argv)

10.81.1 Detailed Description

10.81.2 Function Documentation

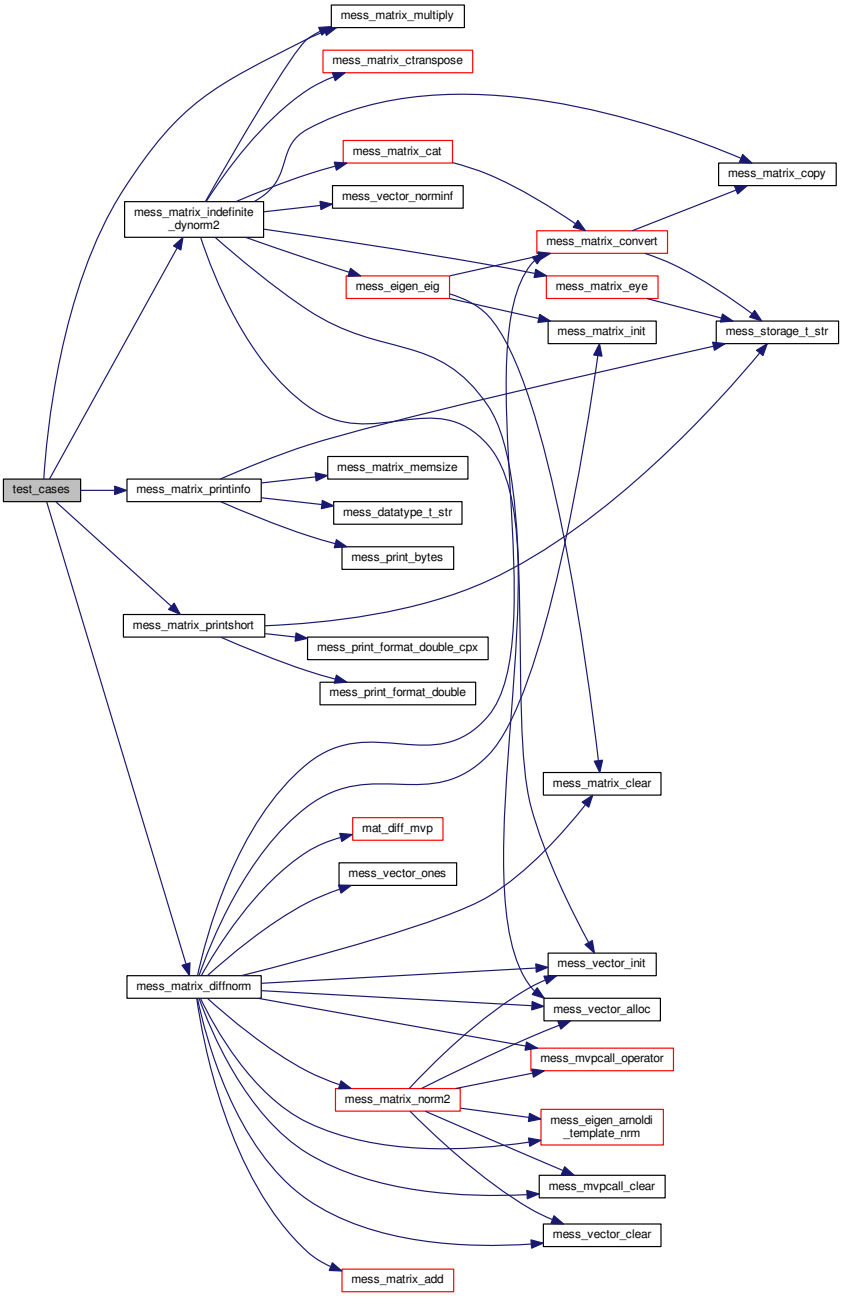
10.81.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.81.2.2 static int test_cases (mess_matrix *W*, mess_matrix *K*, double *tol*) [static]

Here is the call graph for this function:



10.82 check_dynorm_indefinitef.c File Reference

Functions

- static int test_cases (mess_matrix W, mess_matrix K, double tol)
- int main (int argc, char ∗∗argv)

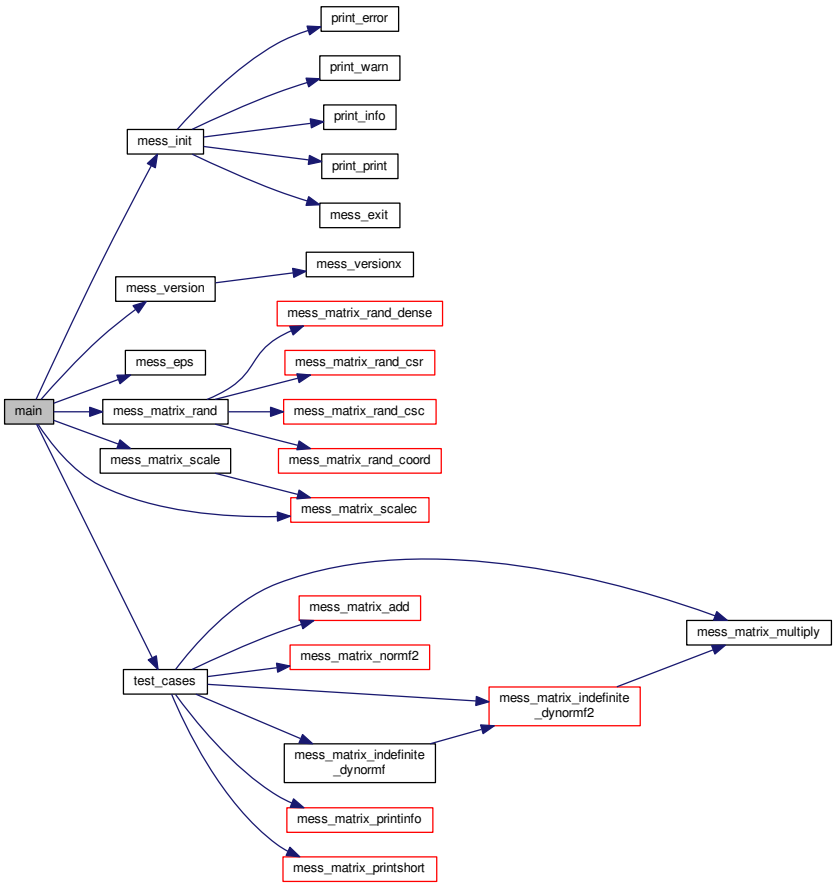
10.82.1 Detailed Description

This function checks the mess_matrix_indefinite_dynormf and mess_matrix_indefinite_dynormf2 functions defined in dynorm.c.

10.82.2 Function Documentation

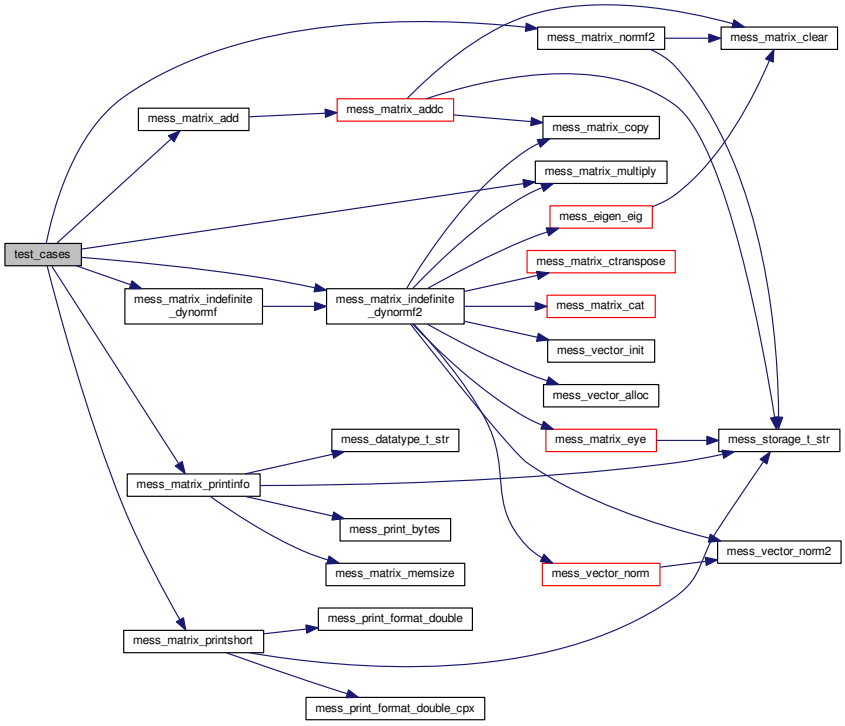
10.82.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.82.2.2 static int test_cases (mess_matrix *W*, mess_matrix *K*, double *tol*) [static]

Here is the call graph for this function:



10.83 check_dynormf.c File Reference

Functions

- double fdmcol (double x, double y)

  Determine values used in a FDM matrix.
- int main (int argc, char $**$argv)

10.83.1 Detailed Description

10.83.2 Function Documentation

10.83.2.1 double fdmcol (double *x*, double *y*)

Parameters

in	*x*	input space variable x
in	*y*	input space variable y

Returns

1 if x is between 0.1 and 0.4, otherwise 0

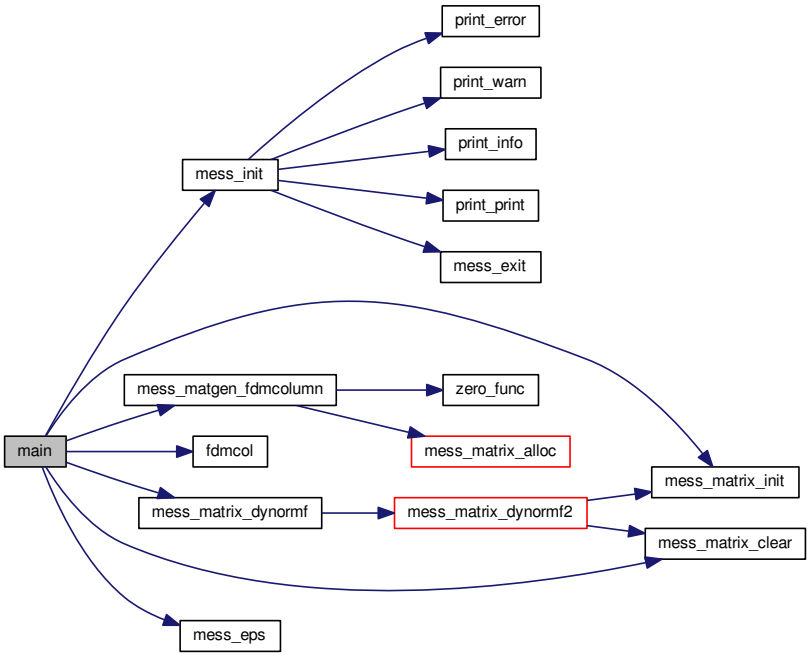
The fdmcol function returns 1 if the space variable x fullfills

$$0.1 < x < 0.4,$$

otherwise 0.

10.83.2.2 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.84 check_dynormft.c File Reference

Functions

- double fdmcol (double x, double y)

  Determine values used in a FDM matrix.
- int main (int argc, char **argv)

10.84.1 Detailed Description

10.84.2 Function Documentation

10.84.2.1 double fdmcol (double *x*, double *y*)

Parameters

in	*x*	input space variable x
in	*y*	input space variable y

Returns

1 if x is between 0.1 and 0.4, otherwise 0

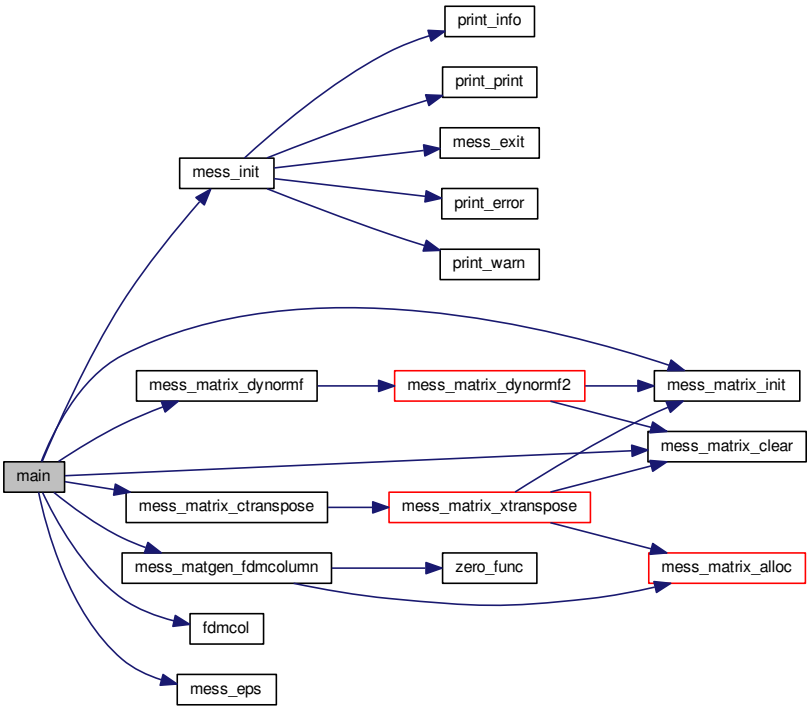
The fdmcol function returns 1 if the space variable x fullfills

$$0.1 < x < 0.4,$$

otherwise 0.

10.84.2.2 int main (int *argc*, char $**$ *argv*)

Here is the call graph for this function:



10.85 check_eig.c File Reference

Functions

- int check_eigen_vector (mess_matrix A, mess_matrix Z, double ev, mess_int_t col, double tol)

  Check if the eigenvalue problem is solved correctly.
- int main (int argc, char $**$argv)

10.85.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.85.2 Function Documentation

10.85.2.1 int check_eigen_vector (mess_matrix *A*, mess_matrix *Z*, double *ev*, mess_int_t *col*, double *tol*)

Parameters

in	*A*	input matrix
in	*Z*	input matrix containing eigenvectors
in	*ev*	input eigenvalue
in	*col*	input column of eigenvector matrix to check
in	*tol*	tolerance for error

Returns

diff > tol

The check_eigen_vector function checks if the eigenvalue problem

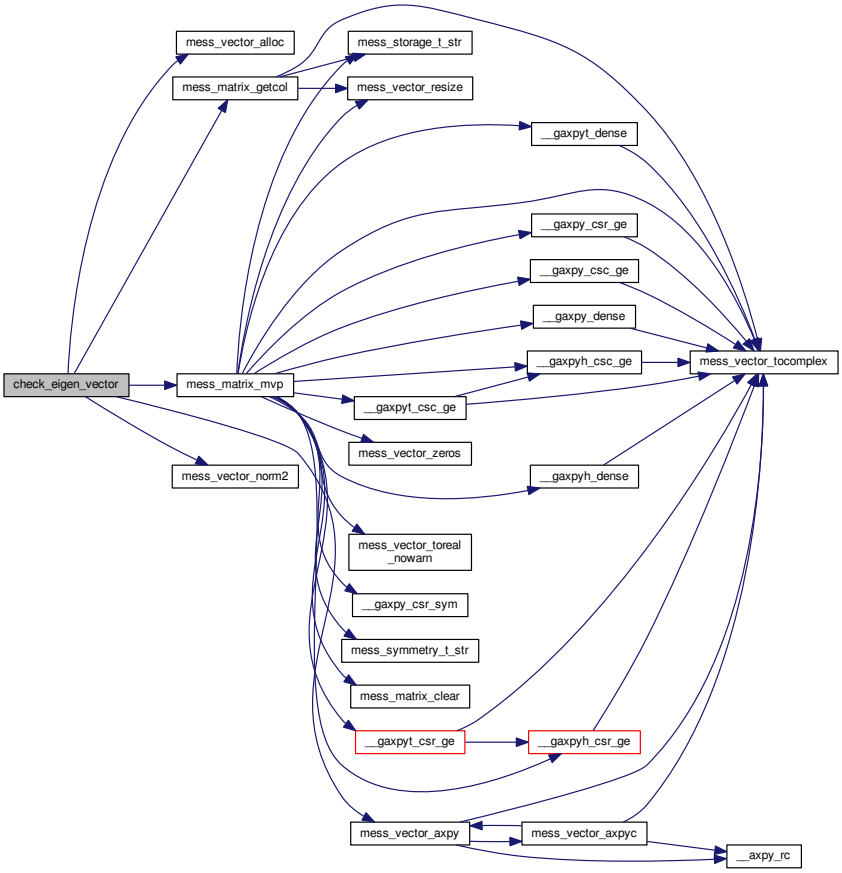
$$AV(:, col) = evV(:, col)$$

is solved correctly for a given eigenvalue ev and an eigenvector $Z(:, col)$.
If

$$\|AV(:, col) - evV(:, col)\|_2 > \sqrt{(\epsilon)}$$

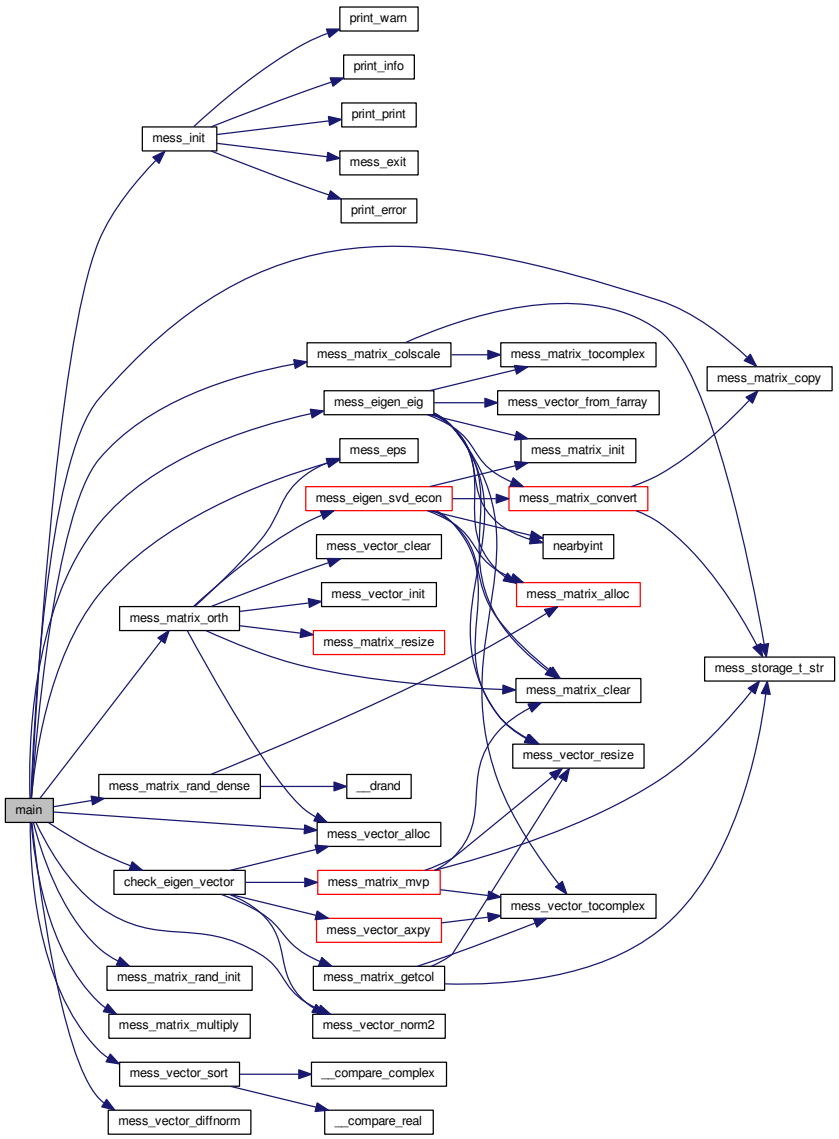
the error value is increased by 1.
n is the number of rows of Z.

Here is the call graph for this function:



10.85.2.2 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.86 check_eig2.c File Reference

Functions

- int check_eigen_vector (mess_matrix A, mess_matrix Z, mess_double_cpx_t ev, mess_int_t col, double tol)

  Check if the eigenvalue problem is solved correctly.
- int main (int argc, char ∗∗argv)

10.86.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_eigen_eig function defined in eig.c that means it checks if the eigenvalue problem

$$Ax = \lambda x$$

is solved correctly .

10.86.2 Function Documentation

10.86.2.1 int check_eigen_vector (mess_matrix *A*, mess_matrix *Z*, mess_double_cpx_t *ev*, mess_int_t *col*, double *tol*)

Parameters

in	*A*	input matrix
in	*Z*	input matrix containing eigenvectors
in	*ev*	input eigenvalue
in	*col*	input column of eigenvector matrix to check
in	*tol*	tolerance for error

Returns

diff > tol

The check_eigen_vector function checks if the eigenvalue problem

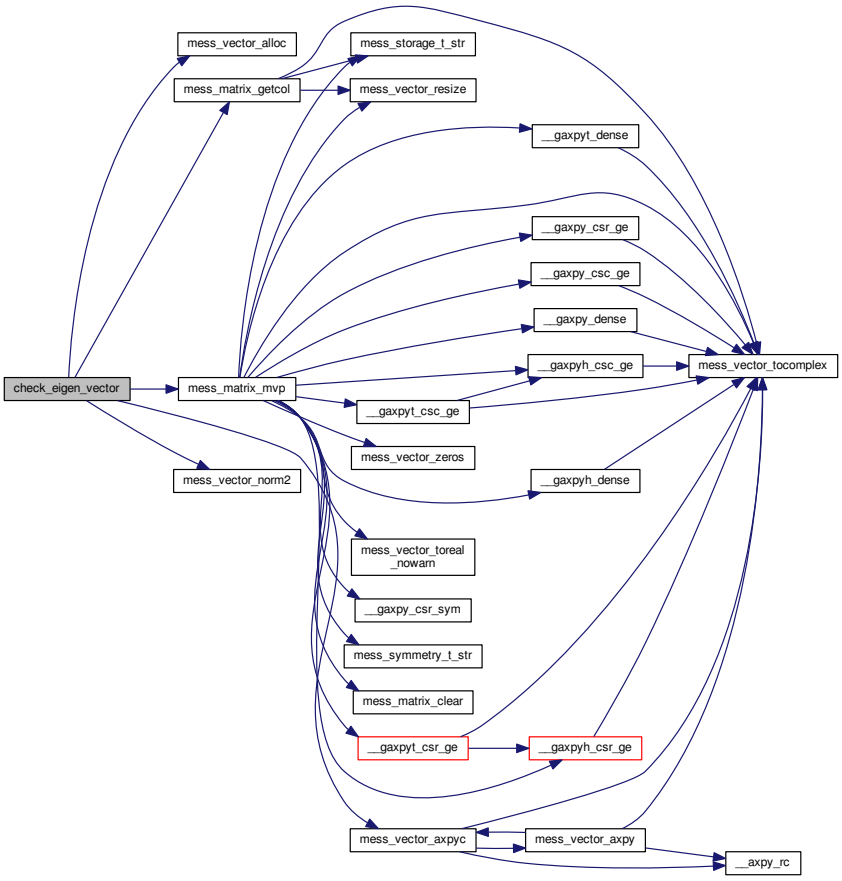
$$AV(:, col) = evV(:, col)$$

is solved correctly for a given eigenvalue ev and an eigenvector $Z(:, col)$.
If

$$\|AV(:, col) - evV(:, col)\|_2 > \sqrt(\epsilon)$$

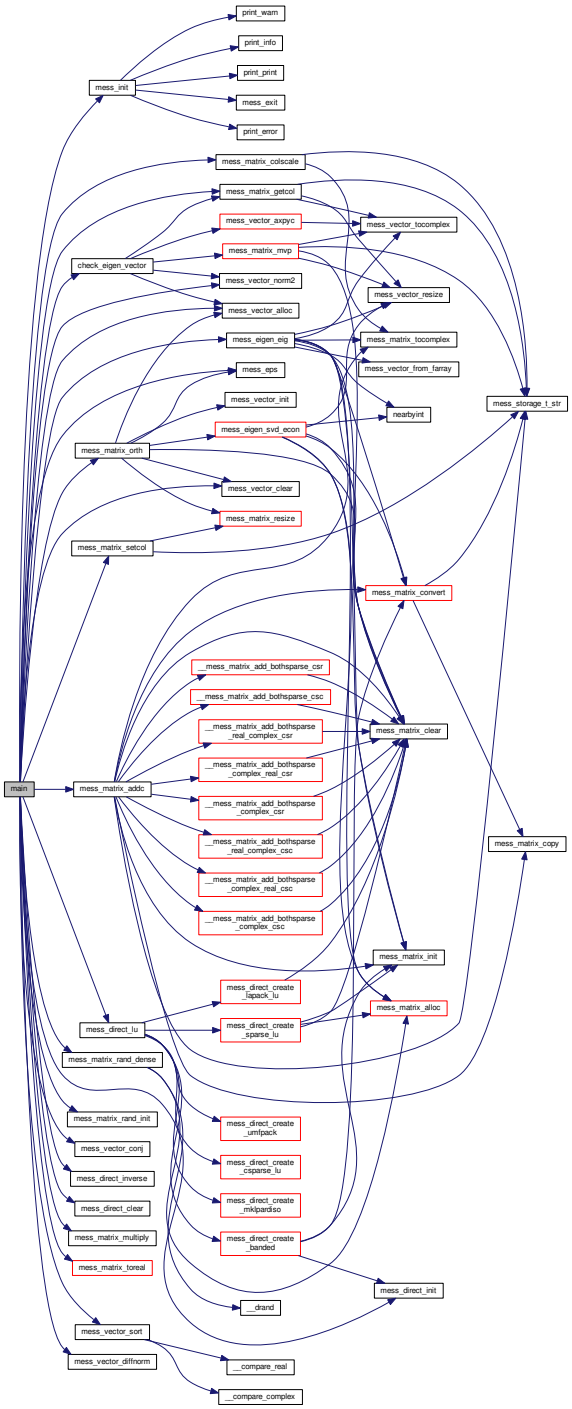
the error value is increased by 1.
n is the number of rows of Z.

Here is the call graph for this function:



10.86.2.2 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.87 check_eig_complex.c File Reference

Functions

- int check_eigen_vector (mess_matrix A, mess_matrix Z, mess_double_cpx_t ev, mess_int_t col, double tol)
  Check if the eigenvalue problem is solved correctly.
- int main (int argc, char ∗∗argv)

10.87.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.87.2 Function Documentation

10.87.2.1 int check_eigen_vector (mess_matrix *A*, mess_matrix *Z*, mess_double_cpx_t *ev*, mess_int_t *col*, double *tol*)

Parameters

in	*A*	input matrix
in	*Z*	input matrix containing eigenvectors
in	*ev*	input eigenvalue
in	*col*	input column of eigenvector matrix to check
in	*tol*	tolerance for error

Returns

diff > tol

The check_eigen_vector function checks if the eigenvalue problem

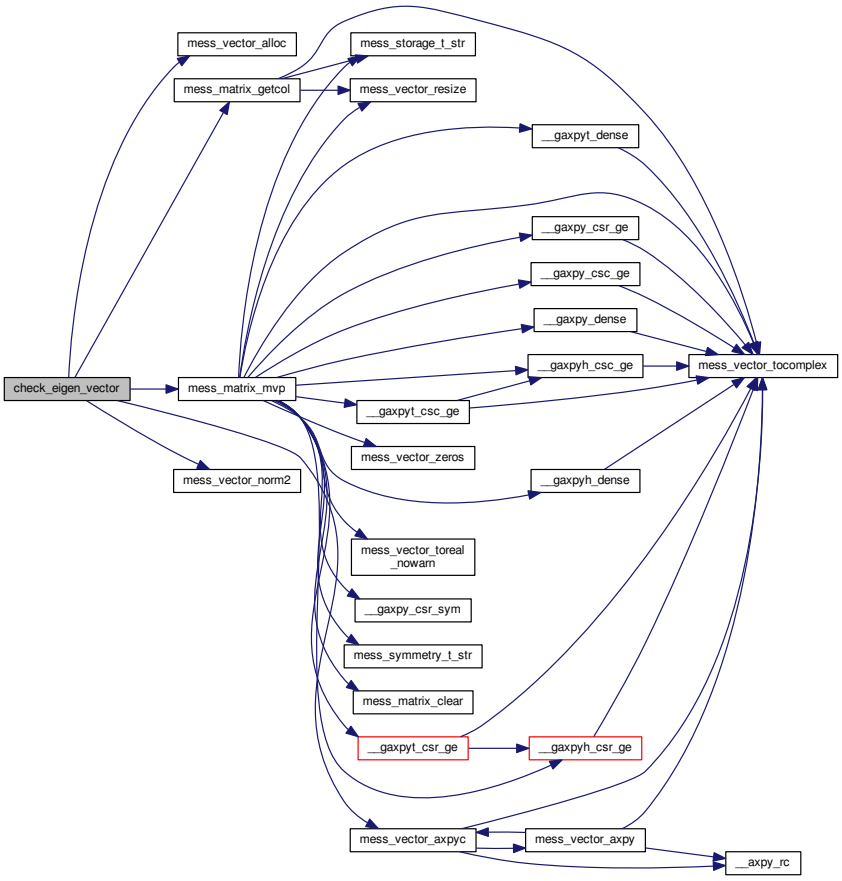
$$AV(:, col) = evV(:, col)$$

is solved correctly for a given eigenvalue ev and an eigenvector $Z(:, col)$.
If

$$\|AV(:, col) - evV(:, col)\|_2 > \sqrt{(\epsilon)}$$

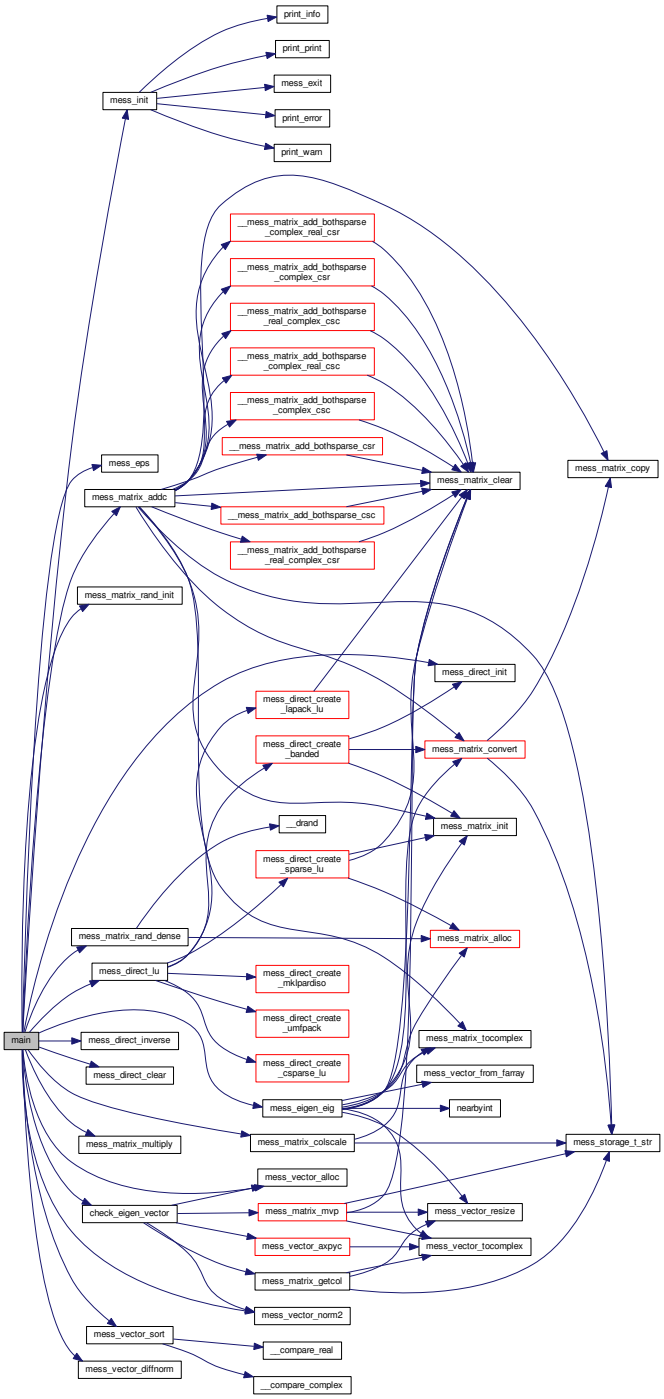
the error value is increased by 1.
n is the number of rows of Z.

Here is the call graph for this function:



10.87.2.2 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.88 check_eig_hessenberg_schur.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.88.1 Detailed Description

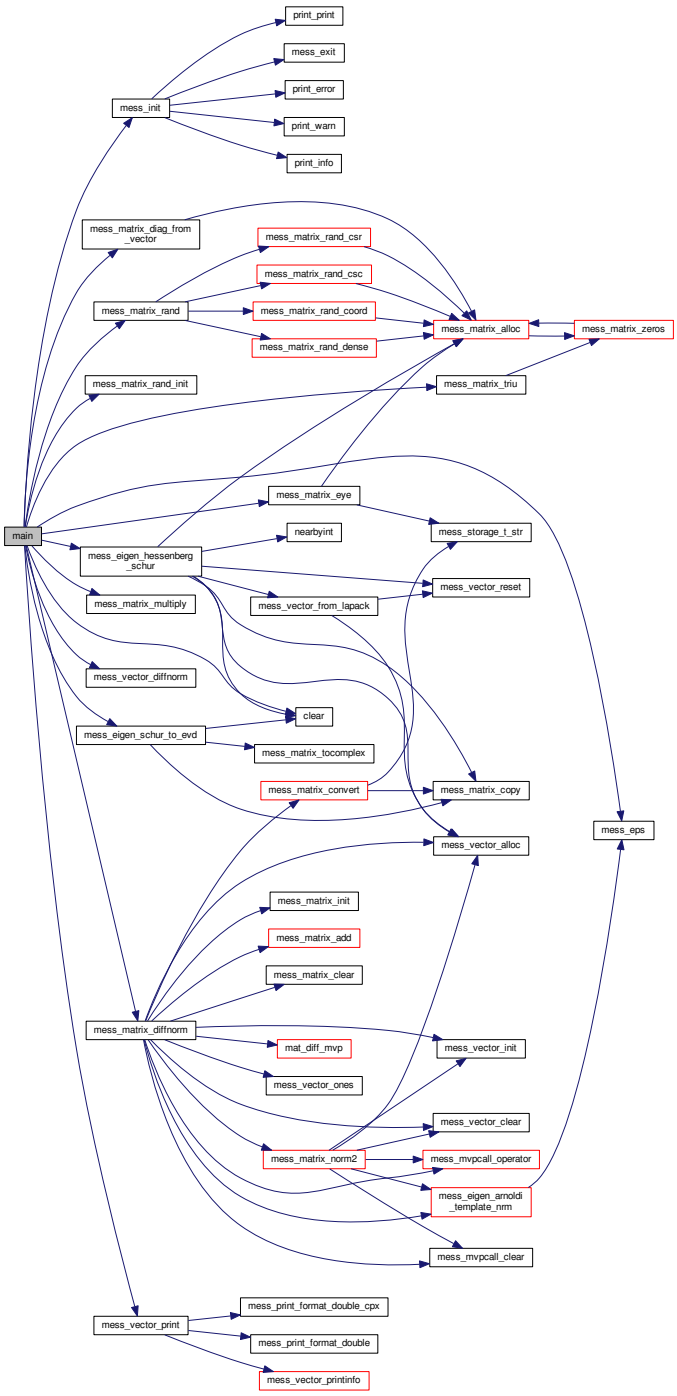
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.88.2 Function Documentation

10.88.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.89 check_eig_hessenberg_schur_largest_ev.c File Reference

Functions

- int main (int argc, char **argv)

10.89.1 Detailed Description

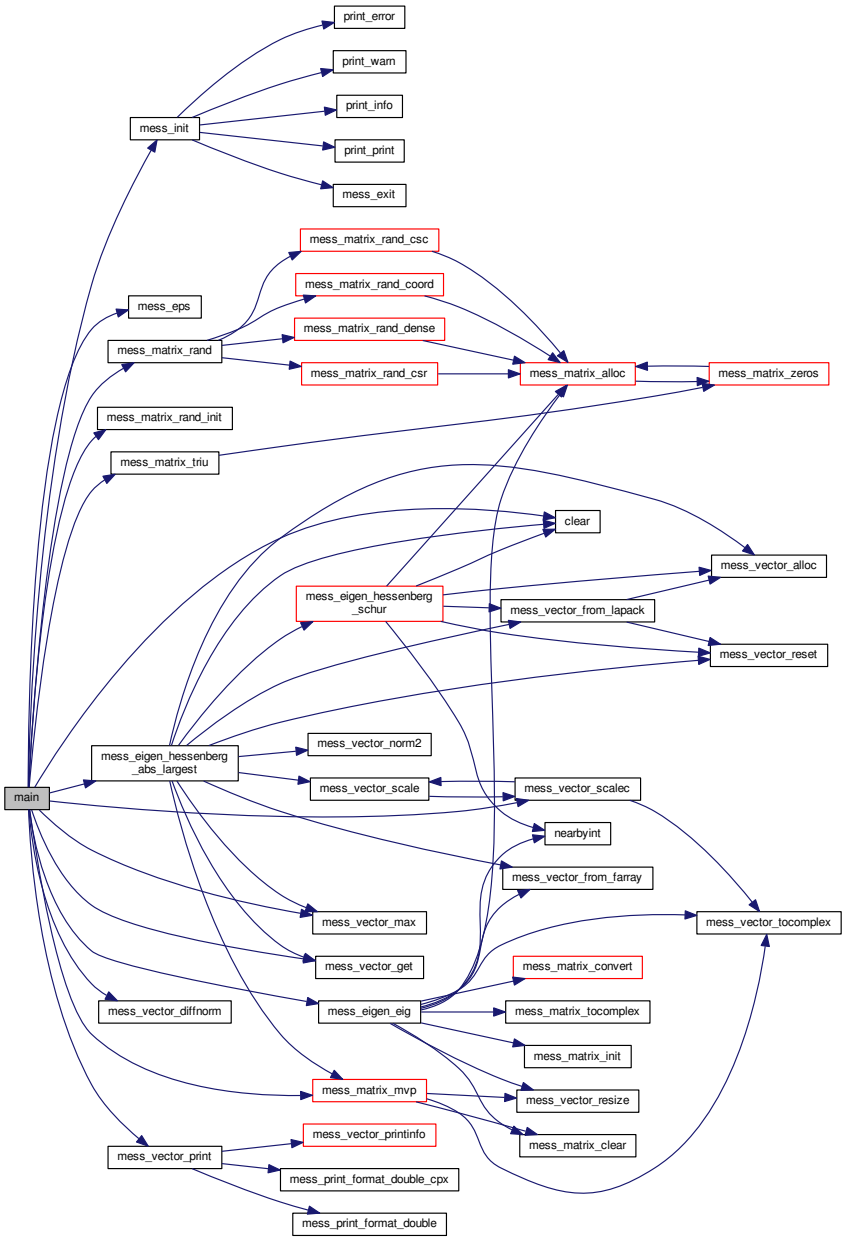
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.89.2 Function Documentation

10.89.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.90 check_eliminate_zeros.c File Reference

Functions

- static double f_real (double val)
- static mess_double_cpx_t f_cpx (mess_double_cpx_t val)
- static mess_int_t iszero_real (double val)
- static mess_int_t iszero_cpx (mess_double_cpx_t val)
- int main (int argc, char ∗∗argv)

10.90.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.90.2 Function Documentation

10.90.2.1 static mess_double_cpx_t f_cpx (mess_double_cpx_t *val*) [static]

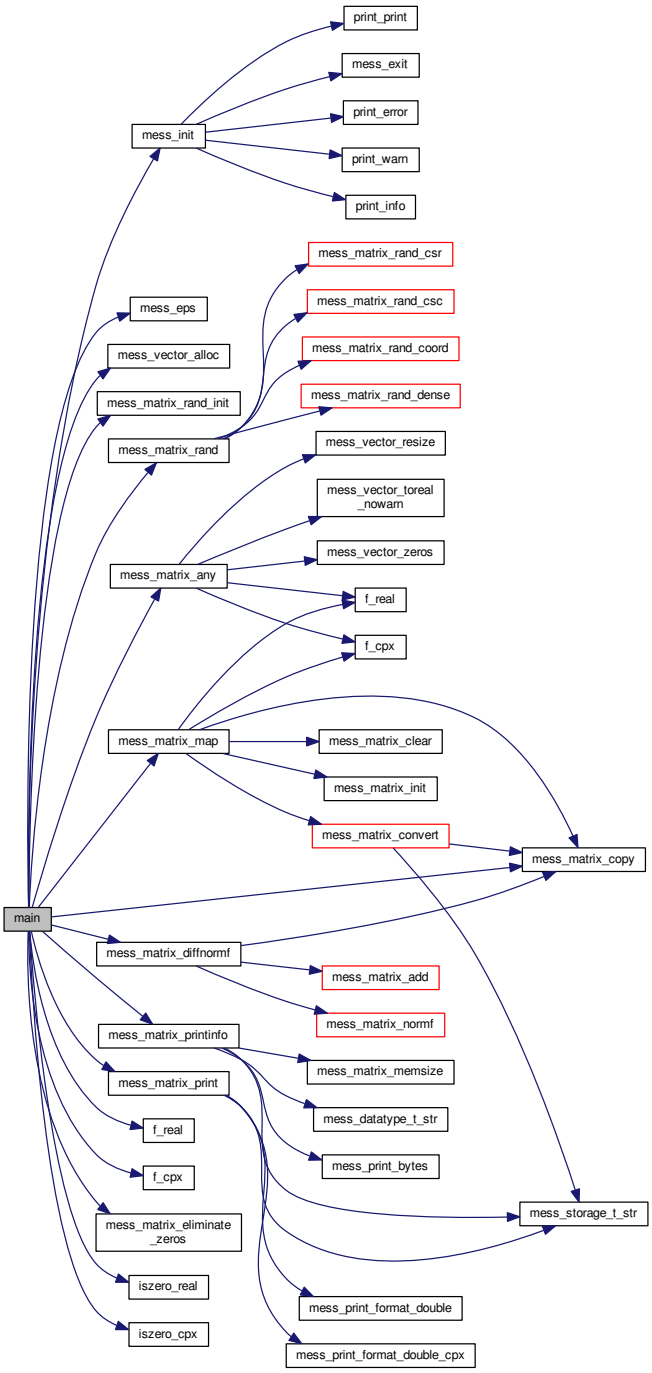
10.90.2.2 static double f_real (double *val*) [static]

10.90.2.3 static mess_int_t iszero_cpx (mess_double_cpx_t *val*) [static]

10.90.2.4 static mess_int_t iszero_real (double *val*) [static]

10.90.2.5 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.91 check_eps.c File Reference

Functions

- int main (int argc, char **argv)

10.91.1 Detailed Description

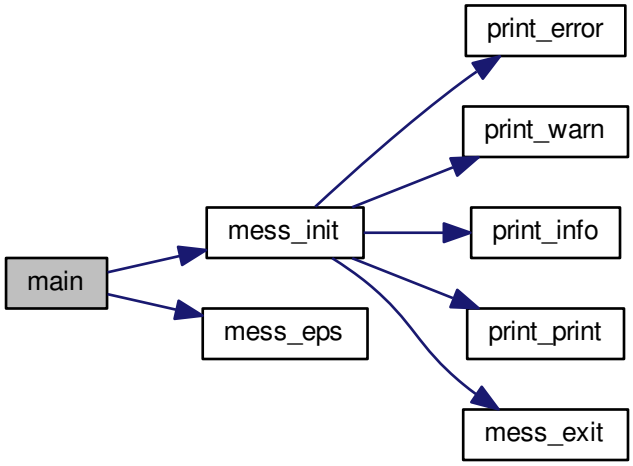
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.91.2 Function Documentation

10.91.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.92 check_fputs_block.c File Reference

Functions

- int main (int argc, char **argv)

10.92.1 Detailed Description

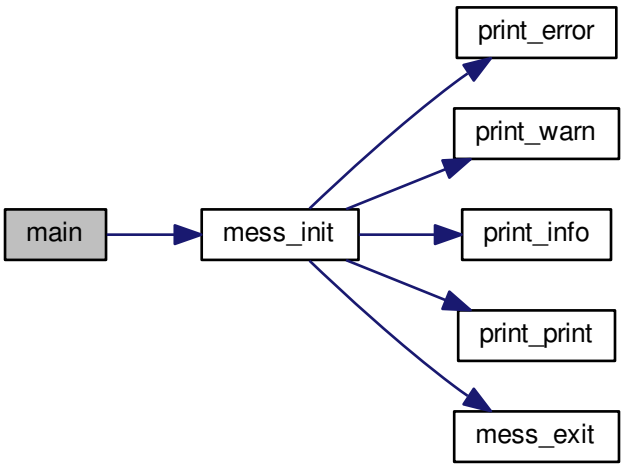
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.92.2 Function Documentation

10.92.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.93 check_freelist.c File Reference

Functions

- int main (int argc, char **argv)

10.93.1 Detailed Description

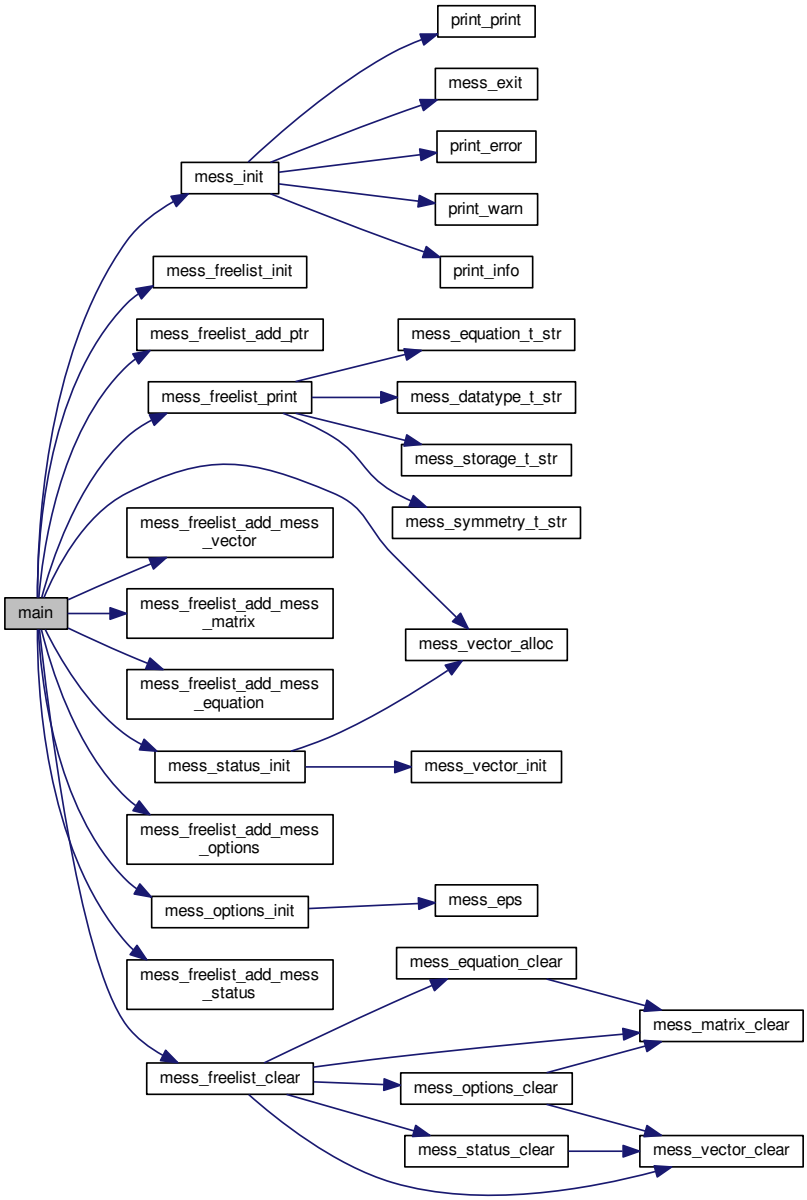
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.93.2 Function Documentation

10.93.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.94 check_fro_inn.c File Reference

Functions

- int main (int argc, char ∗∗argv)

Variables

- double matrix_data1 [16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}
- double matrix_data2 [16] = {-12,0,1,-7,-5,3,42,2,1,23,2,1,-4,0,1,1337}

10.94.1 Detailed Description

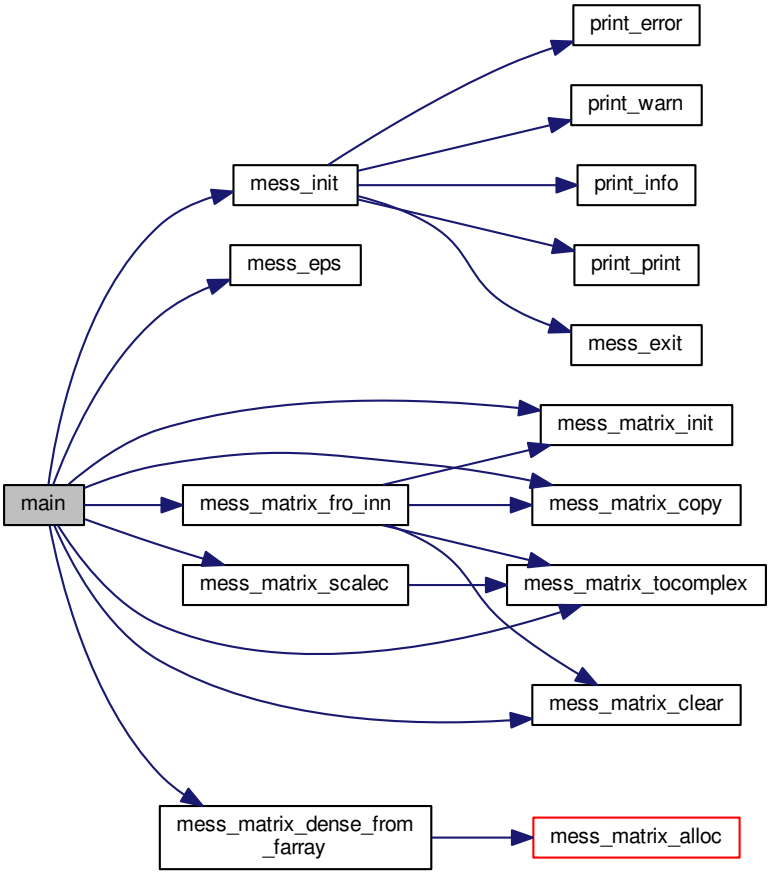
Author

Robert Dykstra (dykstra@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.94.2 Function Documentation

10.94.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.94.3 Variable Documentation

10.94.3.1 double matrix_data1[16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}

10.94.3.2 double matrix_data2[16] = {-12,0,1,-7,-5,3,42,2,1,23,2,1,-4,0,1,1337}

10.95 check_gaxpy.c File Reference

Macros

- #define CHECKGAXPY(ADENSE, TEMP, STORETYPEMAT, OP, VIN, VTEMP1, VTEMP2, VTEMP3, DAT↩ ATYPEOUT, DIFF, ERR, EPS)

Functions

- int main (int argc, char **argv)

10.95.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.95.2 Macro Definition Documentation

10.95.2.1 #define CHECKGAXPY(*ADENSE, TEMP, STORETYPEMAT, OP, VIN, VTEMP1, VTEMP2, VTEMP3, DATATYPEOUT,* DIFF, *ERR, EPS*)

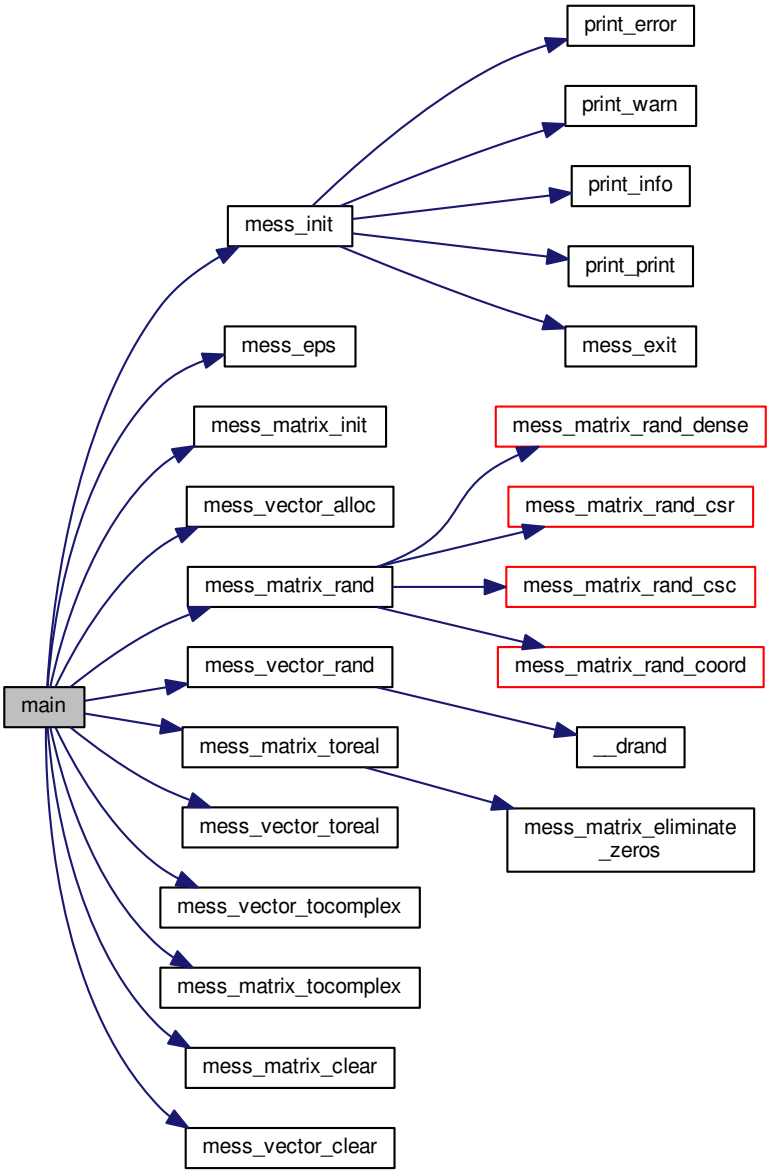
Value:

```
{                                   \
    CALL(mess_vector_totype(VTEMP2,DATATYPEOUT));
                                                                    \
    CALL(mess_vector_totype(VTEMP3,DATATYPEOUT));                   \
    CALL(mess_vector_copy(VIN,VTEMP1));
                                                                  \
    CALL(mess_matrix_convert(ADENSE, TEMP, STORETYPEMAT));          \
    CALL(mess_matrix_gaxpy(OP, ADENSE, VTEMP1, VTEMP2)); /*solution*/
                                                                    \
    CALL(mess_vector_copy(VIN,VTEMP1));
                                                                  \
    CALL(mess_matrix_gaxpy(OP,TEMP,VTEMP1,VTEMP3));
                                                                    \
    /* @ref mess_vector_print(VTEMP2); mess_vector_print(VTEMP3); */
                                        \
    CALL(mess_vector_diffnorm(VTEMP2,VTEMP3,&DIFF));
                                                                        \
    if(DIFF>10*EPS){++ERR;
                                        \
        printf("Diff: %e\n",DIFF);
                                            \
    };/*reset data*/
                                      \
    CALL(mess_vector_zeros(VTEMP1));
                                                                    \
    CALL(mess_vector_zeros(VTEMP2));                                \
    CALL(mess_vector_zeros(VTEMP3));
                                                                    \
}
```

10.95.3 Function Documentation

10.95.3.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.96 check_getelement.c File Reference

Functions

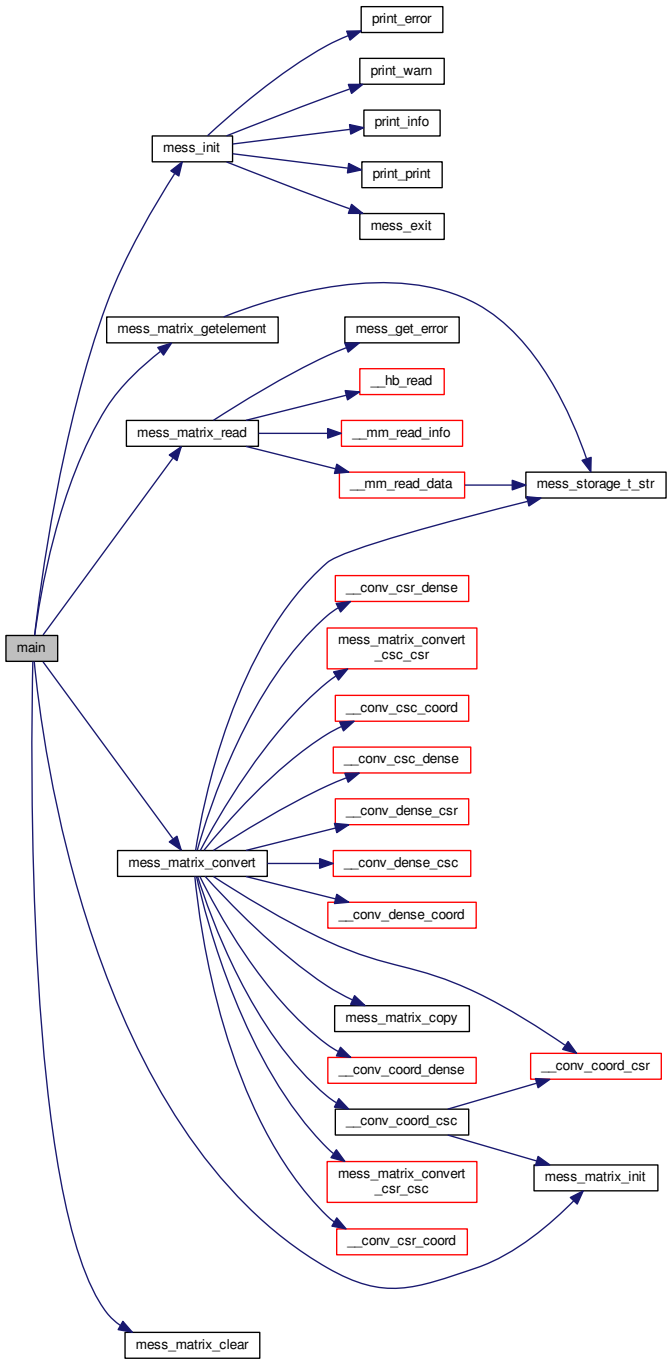
- int main (int argc, char ∗argv[ ])

10.96.1 Detailed Description

10.96.2 Function Documentation

10.96.2.1 int main (int *argc*, char ∗ *argv[ ]*)

Here is the call graph for this function:



10.97 check_getLU.c File Reference

Functions

- int select_solver (int i, mess_matrix A, mess_direct sol)
- int testmat (int i, mess_matrix A)
- int main (int argc, char **argv)

10.97.1 Detailed Description

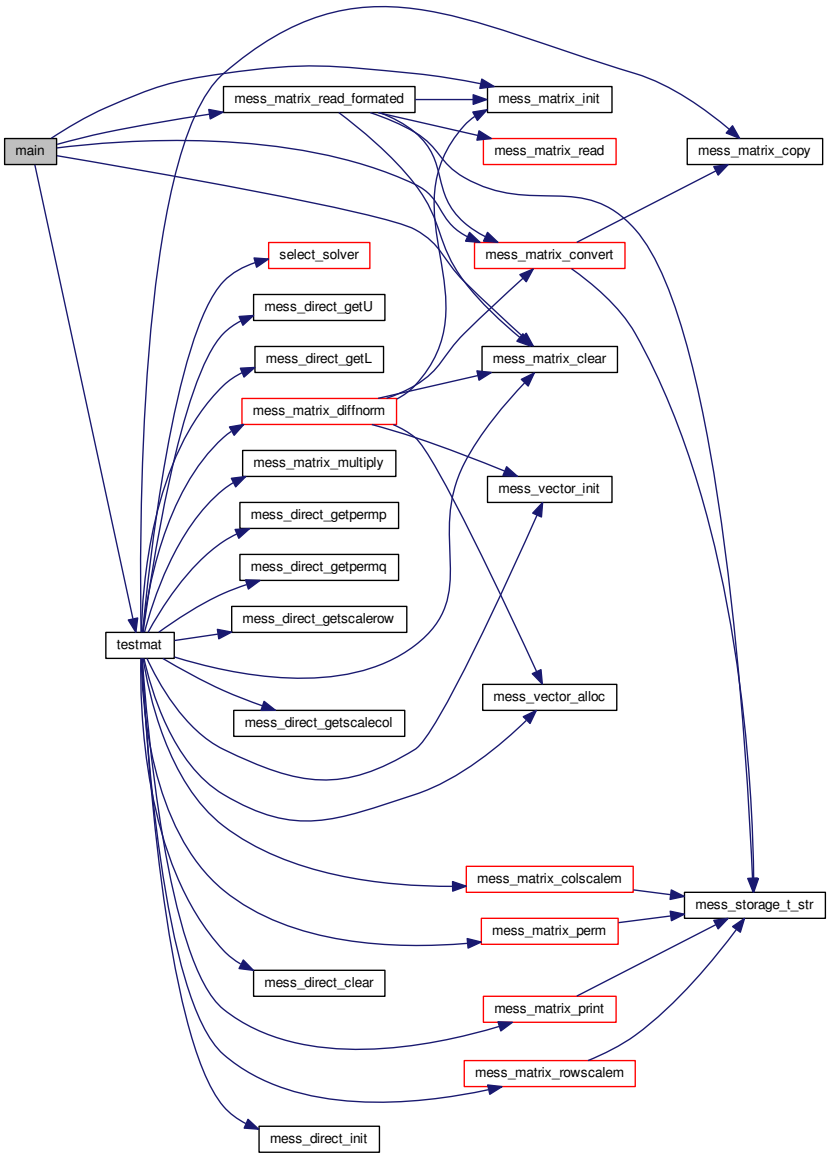
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.97.2 Function Documentation

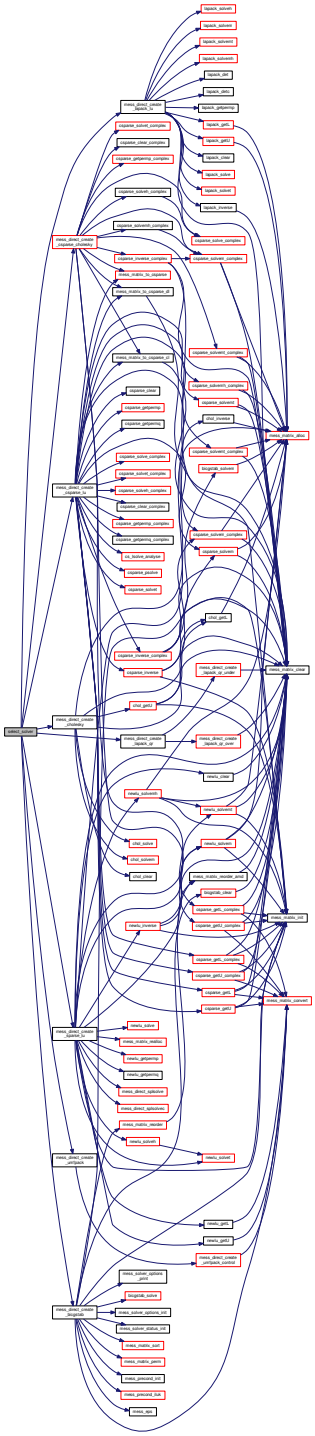
10.97.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



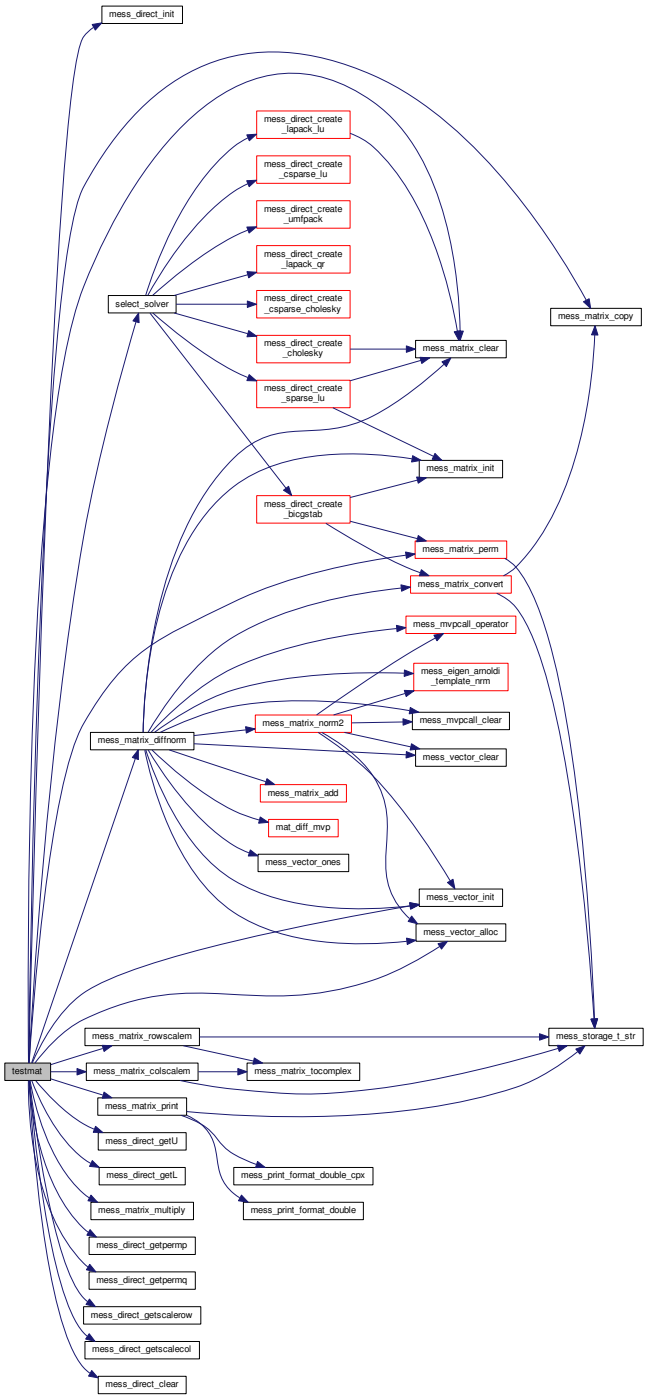
10.97.2.2 int select_solver (int *i*, mess_matrix *A*, mess_direct *sol*)

Here is the call graph for this function:



10.97.2.3 int testmat (int *i*, mess_matrix *A*)

Here is the call graph for this function:



10.98 check_gsign.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.98.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_eigen_gsign function defined in sign.c that means it checks if the sign of a matrix is computed correctly, by the identity

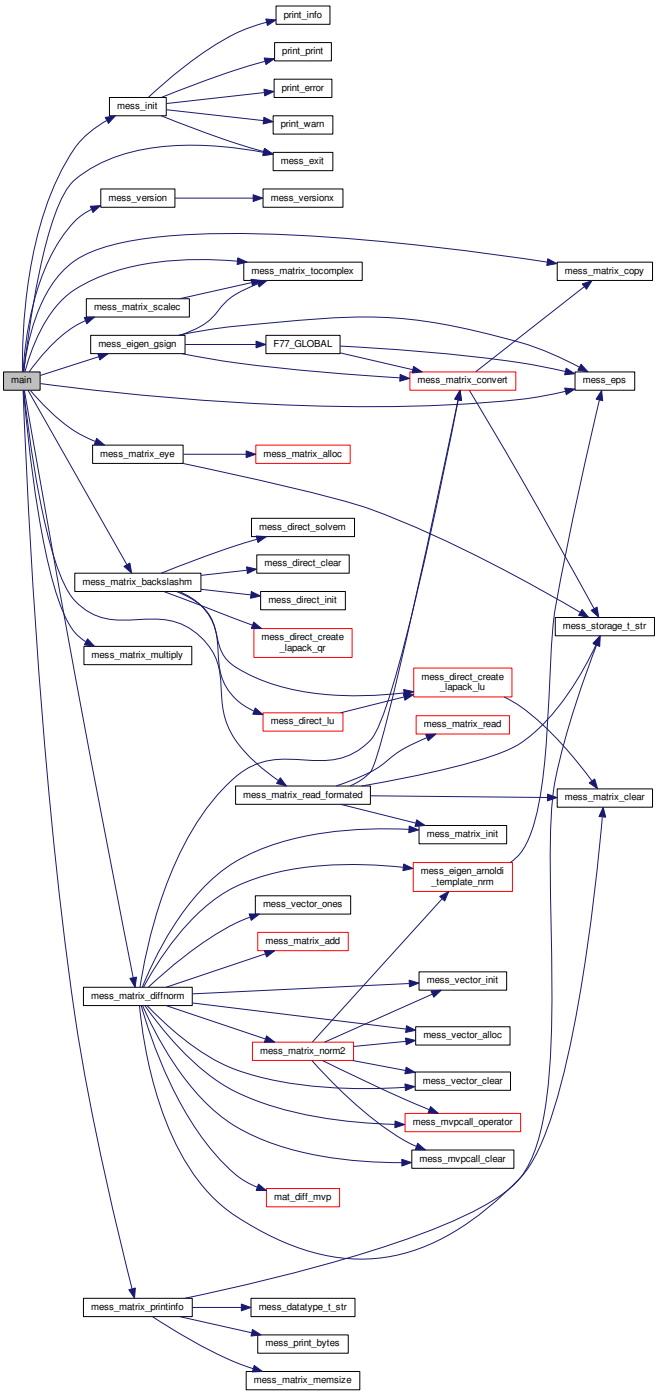
$$(E^{-1} sign(A, B))^2 = I_n$$

is computed correctly.

10.98.2 Function Documentation

10.98.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.99 check_hessenberg.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.99.1 Detailed Description

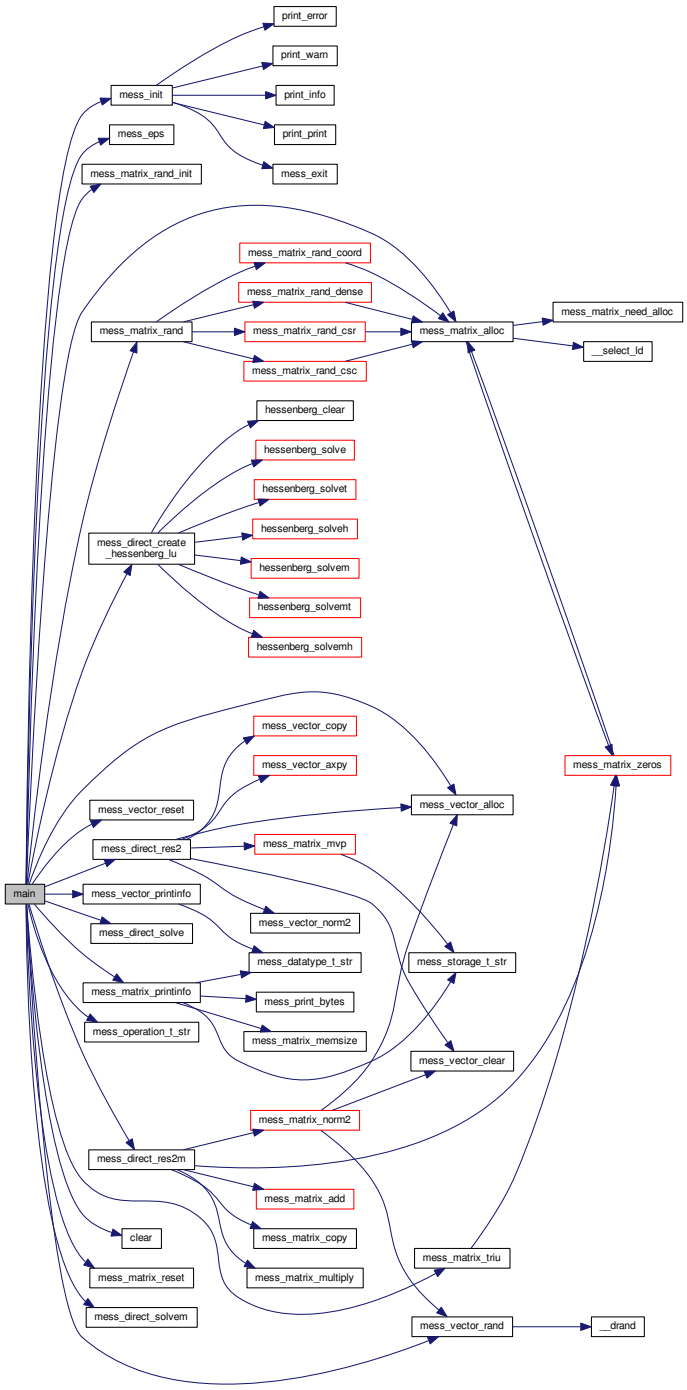
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.99.2 Function Documentation

10.99.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.100 check_itsolver.c File Reference

Functions

- int check_solver (const char ∗name, mess_matrix A, mess_int_t maxit, double tol)

  Check different iterative solvers.
- int main (int argc, const char ∗∗argv)

10.100.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.100.2 Function Documentation

10.100.2.1 int check_solver (const char ∗ *name*, mess_matrix *A*, mess_int_t *maxit*, double *tol*)

Parameters

in	*name*	input name of solver
in	*A*	input matrix
in	*maxit*	input maximum number of iteration
in	*tol*	input tolerance for the iteration

Returns

zero on success or a non zero error code.

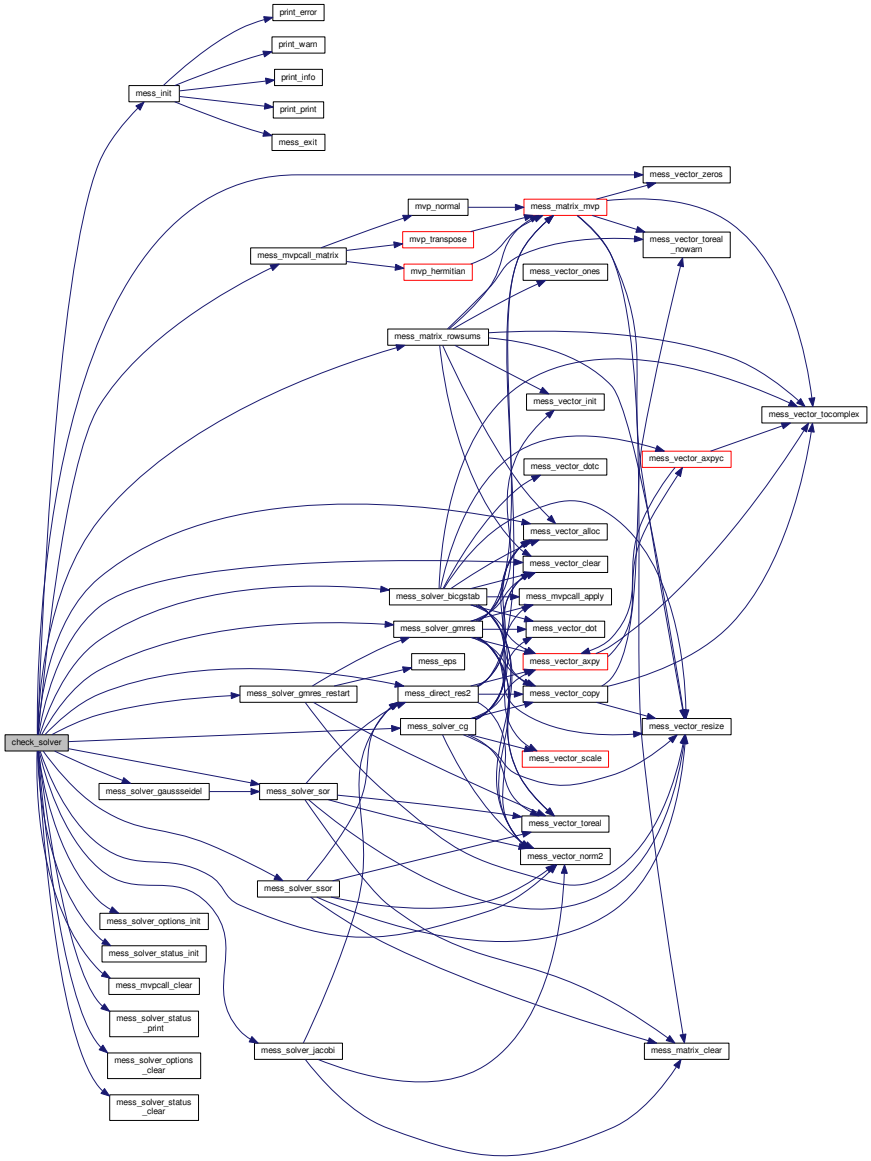
The check_solver function selects a solver by the $name$, solves the linear system of equations

$$Ax = b$$

and checks if the solution is computed correctly.
The entries of the right hand side b are computed by the sum of each row of a A.
Possible solver names are:

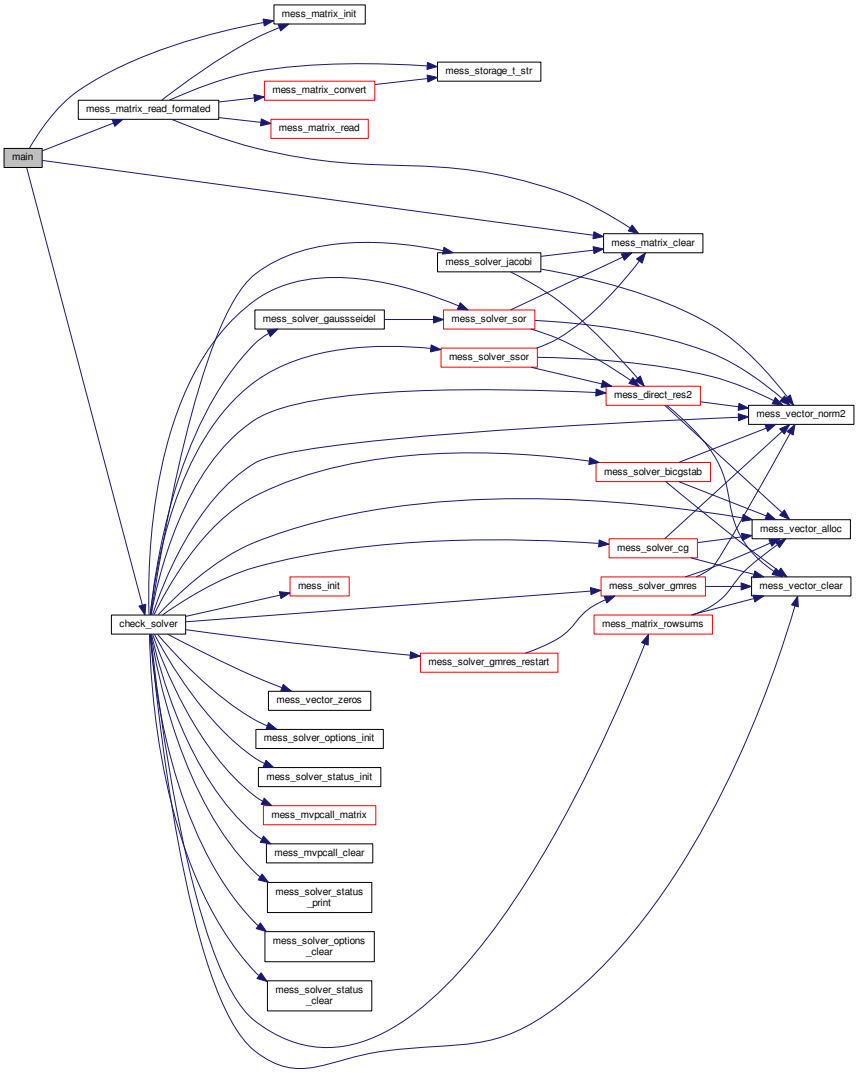
- "BICGSTAB",
- "CG",
- "GMRES",
- "GS",
- "JACOBI",
- "RGMRES",
- "SOR",
- "SSOR".

Here is the call graph for this function:



10.100.2.2 int main (int *argc*, const char ** *argv*)

Here is the call graph for this function:



10.101 check_kron.c File Reference

Macros

- #define CHECK_KRON(V1, V2, V3, VTEST, DIFF, ERR, EPS)

Functions

- int testkron (mess_vector v1, mess_vector v2, mess_vector out)
- int main (int argc, char **argv)

10.101.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.101.2 Macro Definition Documentation

10.101.2.1 #define CHECK_KRON(*V1, V2, V3, VTEST,* DIFF, *ERR, EPS*)

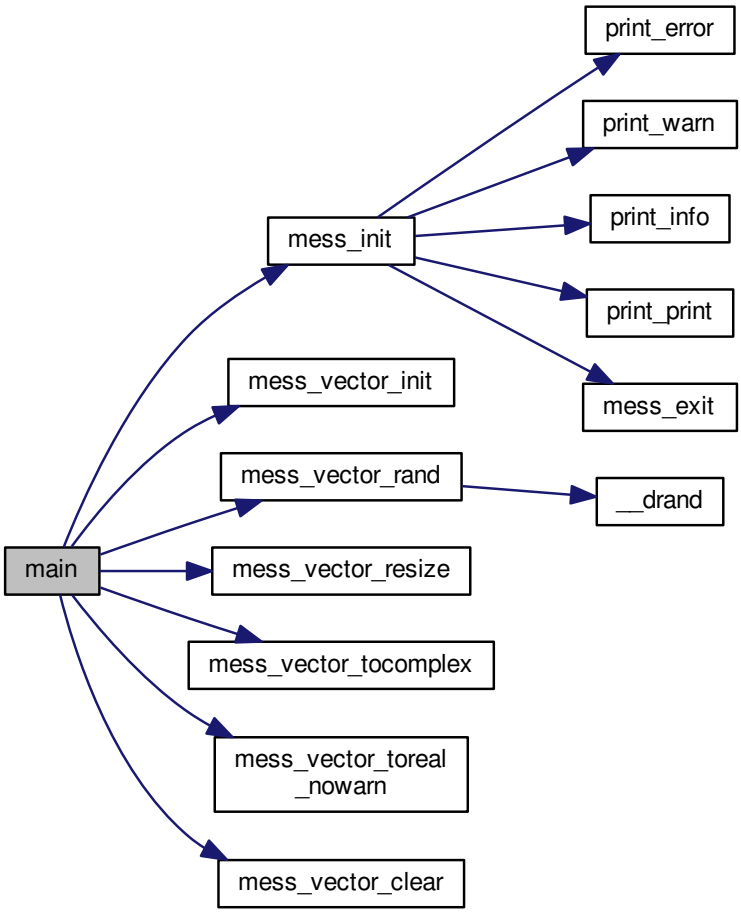
Value:

```
CALL(mess_vector_kron(V1,V2,V3));
      \
CALL(testkron(V1,V2,VTEST));                                    \
CALL(mess_vector_diffnorm(V3,VTEST,&DIFF));                     \
if(DIFF>EPS){                                                   \
    printf("failed\tdiff=%e\n",diff);                           \
    mess_vector_printinfo(V1);
      \
    mess_vector_printinfo(V2);
      \
    mess_vector_printinfo(V3);                                  \
    ERR++;                                                      \
}else{                                                          \
    printf(" passed with diff=%e.\n",diff);                     \
}                                                               \
```

10.101.3 Function Documentation

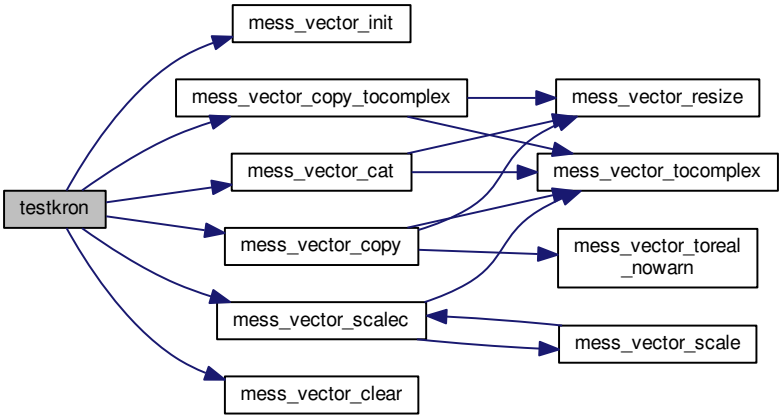
10.101.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.101.3.2 int testkron (mess_vector *v1*, mess_vector *v2*, mess_vector *out*)

Here is the call graph for this function:



10.102 check_lapackqr.c File Reference

Functions

- double check_getL_getU (mess_direct solver, mess_matrix A)

  Checks mess_direct_getL and mess_direct_getU function of mess_direct_create_lapack_qr solver.
- double check_solve (mess_direct solver, mess_matrix A, mess_int_t cpx, mess_operation_t op)

  Checks mess_direct_solve function of mess_direct_create_lapack_qr solver.
- double check_solvem (mess_direct solver, mess_matrix A, mess_int_t cpx, mess_operation_t op)

  Checks mess_direct_solvem function of mess_direct_create_lapack_qr solver.
- int main (int argc, char $**$argv)

10.102.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_matrix_pinv function defined in pinv.c. It checks if

- $@refmess_direct_solve$
- $@refmess_direct_solvem$
- $@refmess_direct_getL$
- $@refmess_direct_getU$ for MESS_REAL and MESS_COMPLEX MESS_DENSE matrices.

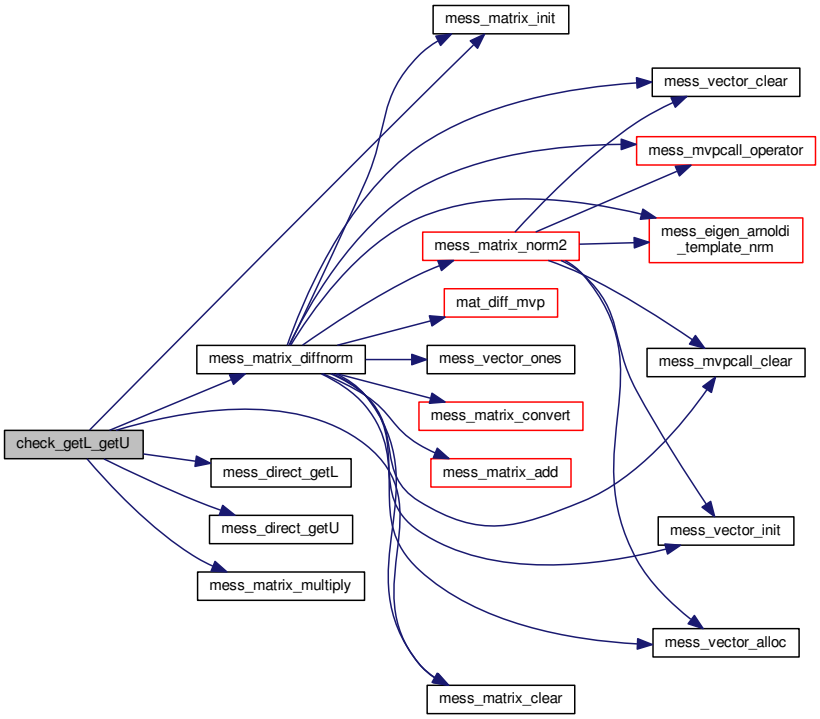
10.102.2 Function Documentation

10.102.2.1 double check_getL_getU (mess_direct *solver,* mess_matrix *A*)

Attention

Internal use only.

Here is the call graph for this function:



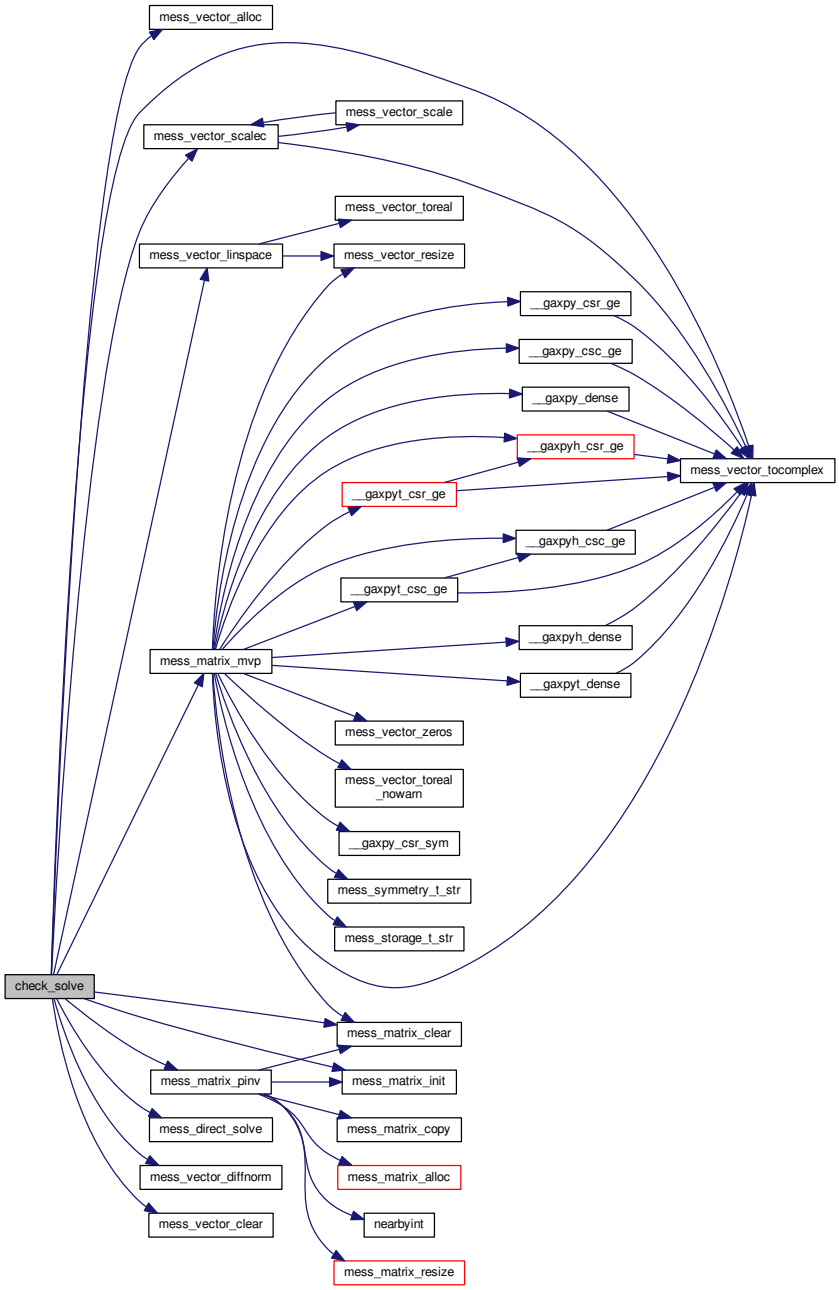
10.102.2.2 double check_solve (mess_direct *solver,* mess_matrix *A,* mess_int_t *cpx,* mess_operation_t *op*)

Internal the functions compares the solution, with the one generated by the pseudoinverse.

Attention

Internal use only.

Here is the call graph for this function:



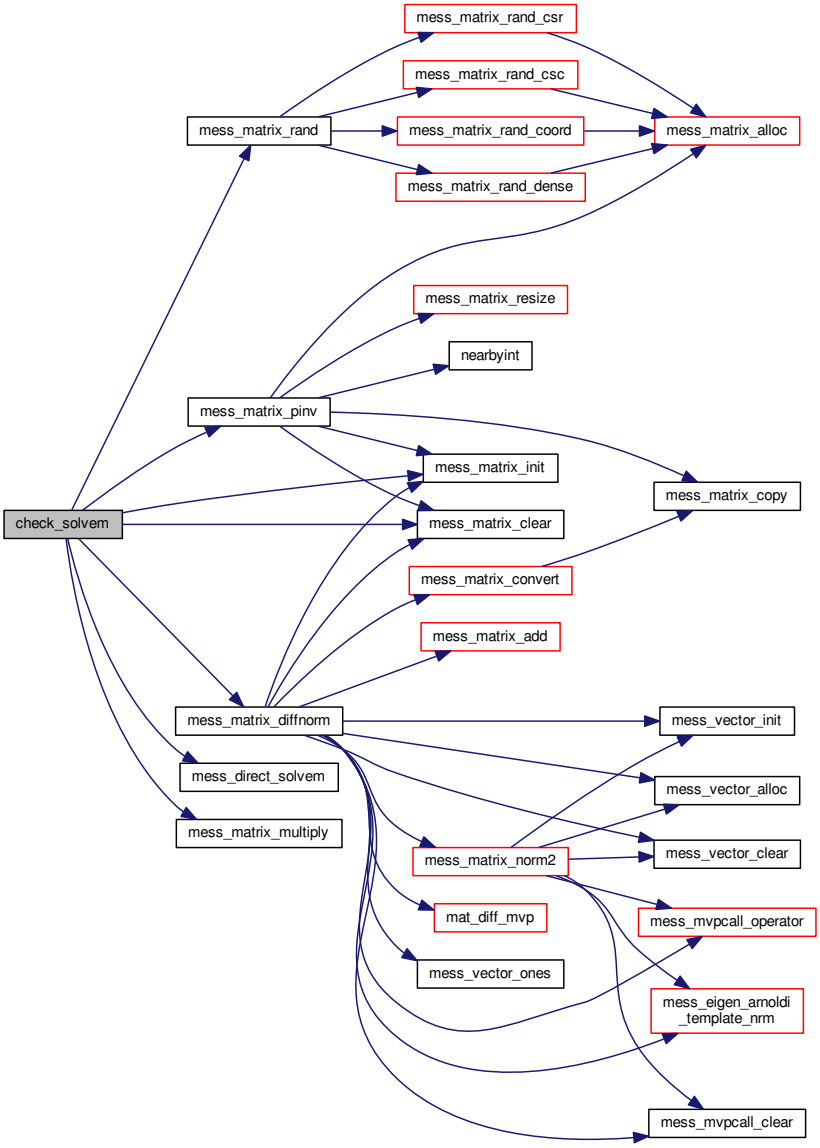
10.102.2.3 double check_solvem (mess_direct *solver,* mess_matrix *A,* mess_int_t *cpx,* mess_operation_t *op*)

Internal the functions compares the solution, with the one generated by the pseudoinverse.

Attention

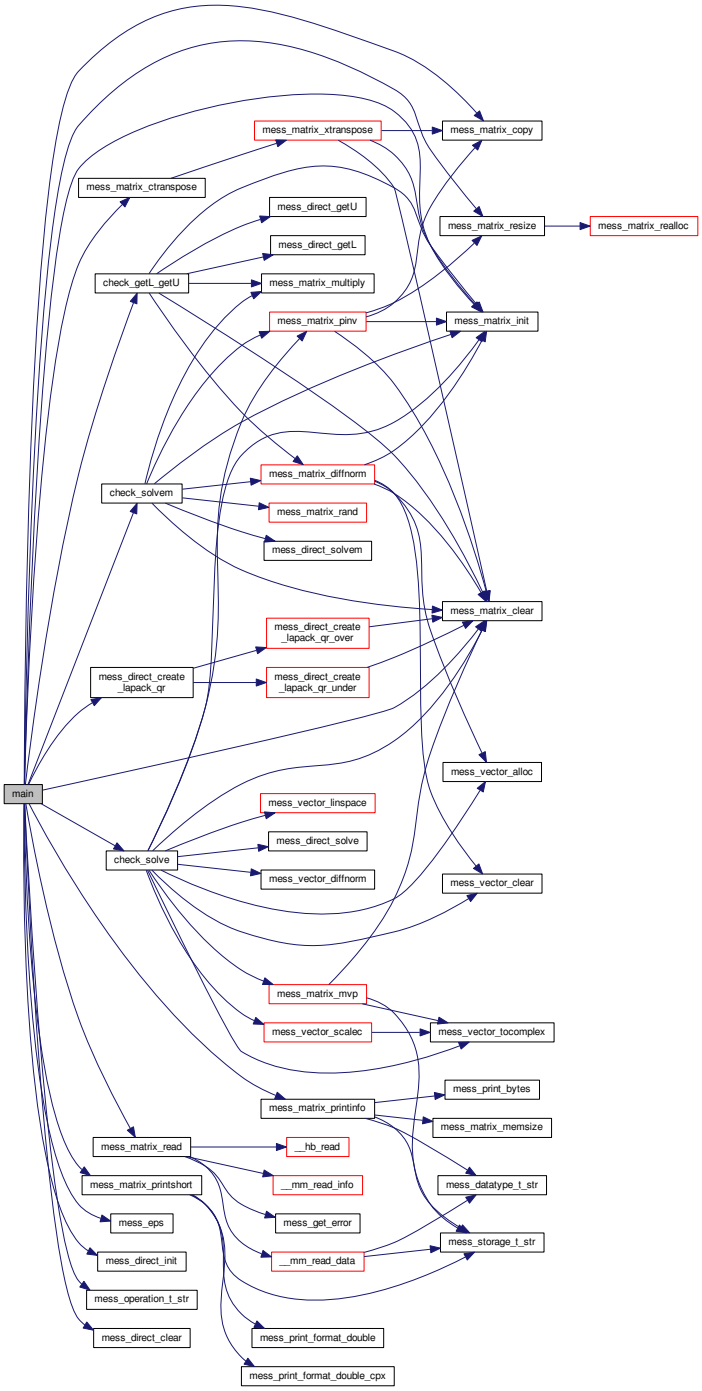
Internal use only.

Here is the call graph for this function:



10.102.2.4 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.103 check_lift.c File Reference

Macros

- #define CHECKVECTORLIFT(X, LIFT, XLIFT, XZEROS, XTMP, DIFF, EPS, ERR)
- #define CHECKMATRIXLIFT(A, LIFT, ALIFT, AZEROS, ATMP, DIFF, EPS, ERR)

Functions

- int main (int argc, char **argv)

10.103.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.103.2 Macro Definition Documentation

10.103.2.1 #define CHECKMATRIXLIFT(A, LIFT, ALIFT, AZEROS, ATMP, DIFF, EPS, ERR)

Value:

```
CALL(mess_matrix_resize((AZEROS),(LIFT),cols));
      \
CALL(mess_matrix_cat((A),NULL,(AZEROS),NULL,MESS_DENSE,(ALIFT)));
      \
CALL(mess_matrix_lift((A),(LIFT), (ATMP)));
      \
mess_matrix_printinfo(A);
      \
mess_matrix_print(A);
      \
mess_matrix_printinfo(AZEROS);
      \
mess_matrix_print(AZEROS);
      \
mess_matrix_printinfo(ALIFT);
      \
mess_matrix_print(ALIFT);
      \
mess_matrix_print(ATMP);
      \
CALL(mess_matrix_add(-1,(ALIFT),1,(ATMP)));
      \
CALL(mess_matrix_normf((ATMP),&(DIFF)));                          \
if((DIFF)>(EPS)){                                                 \
    printf("mess_matrix_lift failed with n=%d\n",(LIFT));         \
    CALL(mess_matrix_printinfo(A));                               \
    printf("diff=%e\n",(DIFF));                                   \
    ++(ERR);                                                      \
}
```

10.103.2.2 #define CHECKVECTORLIFT(X, LIFT, XLIFT, XZEROS, XTMP, DIFF, EPS, ERR)

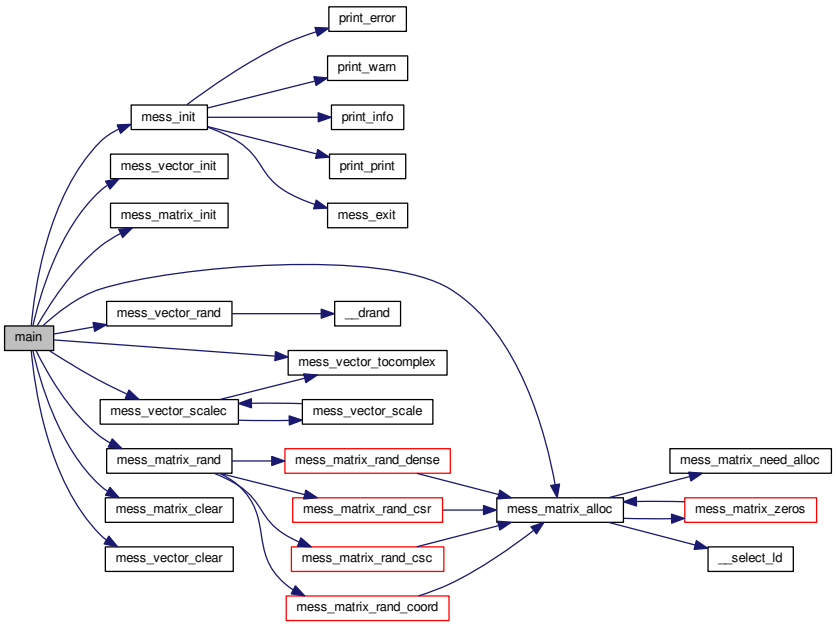
Value:

```
CALL(mess_vector_lift((X),(LIFT),(XLIFT)));
      \
CALL(mess_vector_resize((XZEROS),(LIFT)));
      \
CALL(mess_vector_zeros((XZEROS)));
      \
CALL(mess_vector_cat((X),(XZEROS),(XTMP)));
      \
CALL(mess_vector_diffnorm((XLIFT),(XTMP),&(DIFF)));             \
if((DIFF)>(EPS)){                                               \
    printf("mess_vector_lift failed with n=%d\n",(LIFT));       \
    CALL(mess_vector_printinfo((X)));                           \
    ++(ERR);                                                    \
}
```

10.103.3 Function Documentation

10.103.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.104 check_linesearch.c File Reference

Macros

- #define SCALAR_ERROR_MACRO(NAME, SCALAR, MYSCALAR, ABS_ERR, REL_ERR, TOL, ERR)

Functions

- int main (int argc, char **argv)

10.104.1 Detailed Description

This function checks the mess_lrcfadi_nm_linesearch_poly and mess_lrcfadi_nm_linesearch_lambda functions defined in lrnm_linesearch.c that means it checks if the scalars of the polynomial are computed correctly and the minimizer $lambda$ of the polynomial was found correct. The check is done with tests data generated from the matlab implementation.

10.104.2 Macro Definition Documentation

10.104.2.1 #define SCALAR_ERROR_MACRO(*NAME, SCALAR, MYSCALAR, ABS_ERR, REL_ERR, TOL, ERR*)

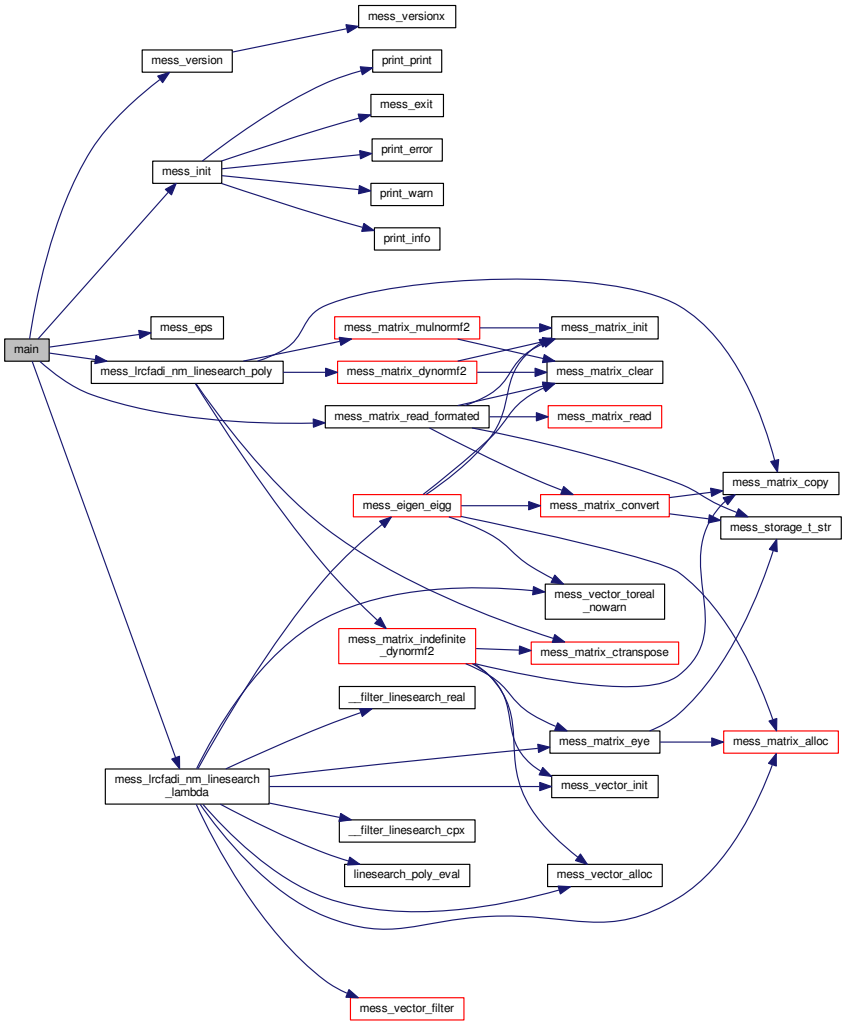
Value:

```
(ABS_ERR) = fabs((SCALAR)-(MYSCALAR));
                    \
    (REL_ERR) = (ABS_ERR)/fabs(SCALAR);
                          \
    printf(#NAME "\n");
                          \
    printf("correct=%+.10e\t computed=%+.10e\t abs_err=%+.10e\t rel_err=%+.10e\n",(SCALAR),(MYSCALAR),(
      ABS_ERR),(REL_ERR)); \
    if((REL_ERR)>(TOL)){printf("failed for: " #NAME"\n"); ++(ERR);}
                          \
    printf("----------------------\n");
```

10.104.3 Function Documentation

10.104.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.105 check_lradi_dae1.c File Reference

Functions

- int main (int argc, char **argv)

10.105.1 Detailed Description

Author

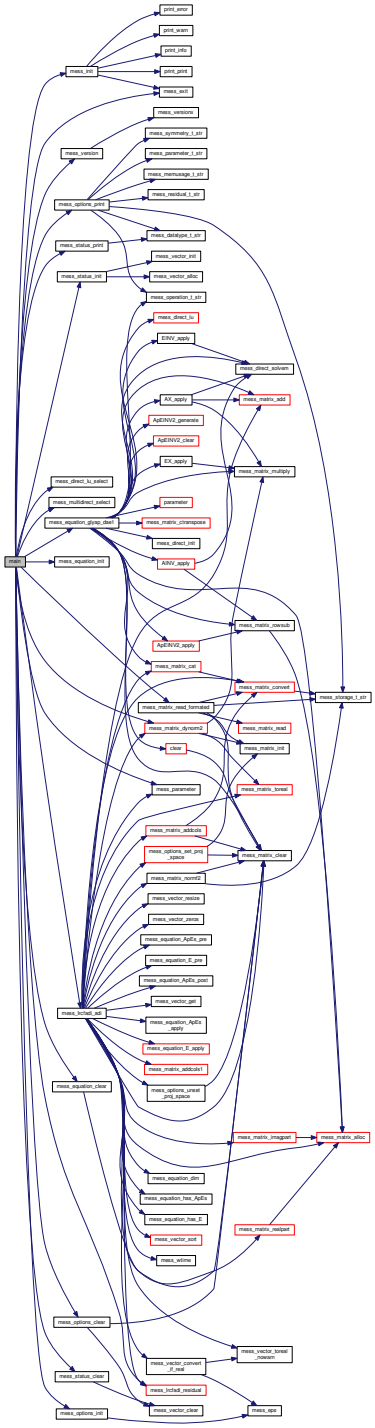
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_lrcfadi_adi function for the DAE 1 function handles defined in lrcf_adi/equation_glyap_dae1.c.

10.105.2 Function Documentation

10.105.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.106 check_lradi_dae2.c File Reference

Functions

- int main (int argc, char ∗∗argv)

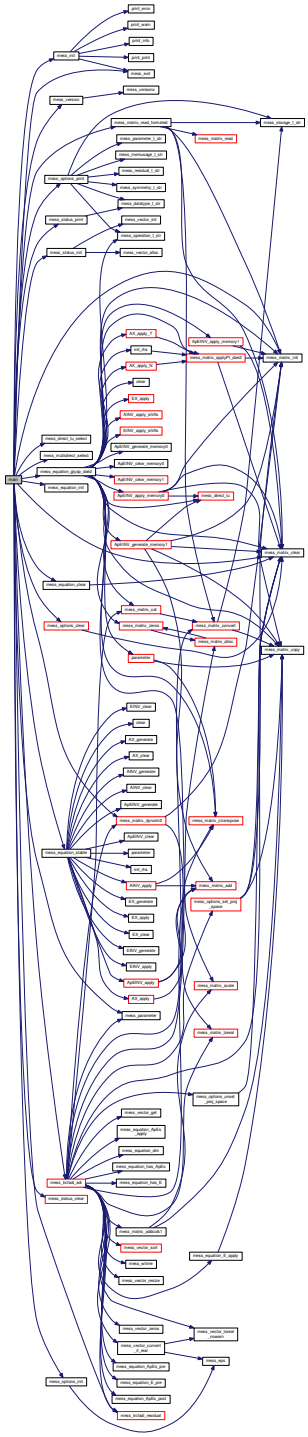
10.106.1 Detailed Description

This function checks the mess_lrcfadi_adi function defined in lrcf_adi/equation_glyap_dae2.c.

10.106.2 Function Documentation

10.106.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.107 check_lradi_glyap.c File Reference

Functions

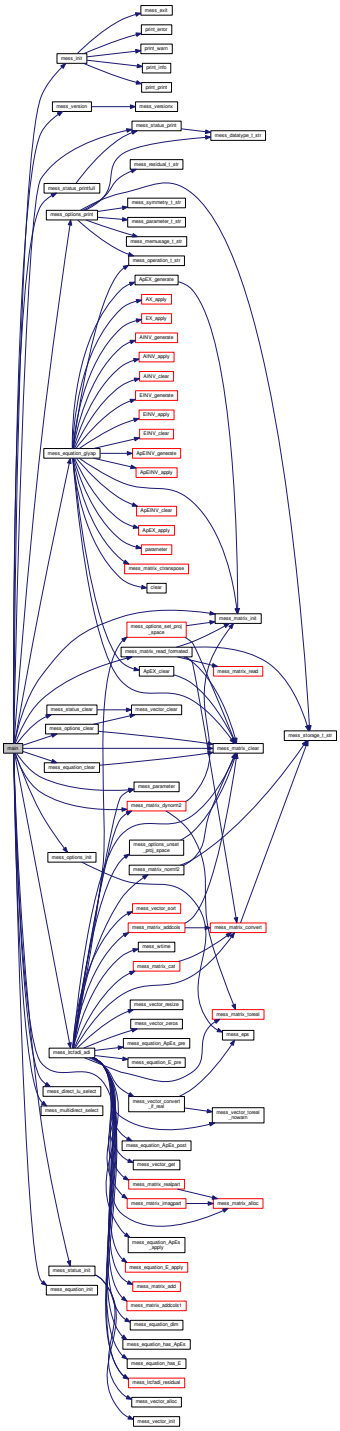
- int main (int argc, char **argv)

10.107.1 Detailed Description

10.107.2 Function Documentation

10.107.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.108 check_lradi_so.c File Reference

Functions

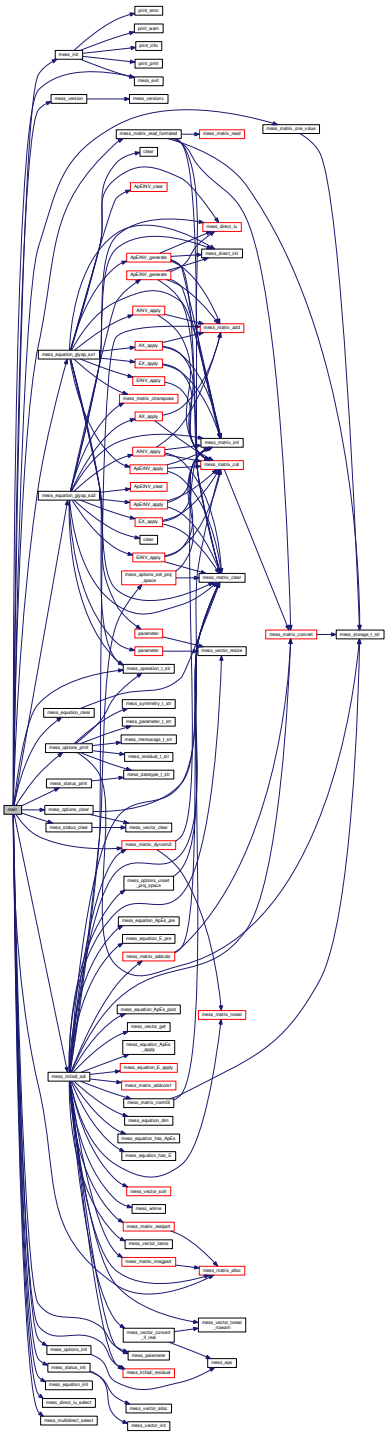
- int main (int argc, char ∗∗argv)

10.108.1 Detailed Description

10.108.2 Function Documentation

10.108.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.109 check_lradi_unsym.c File Reference

Functions

- double fdmcol (double x, double y)

  Determine values used in a FDM matrix.
- double fx (double x, double y)

  Determine values used in a FDM matrix.
- double fy (double x, double y)

  Determine values used in a FDM matrix.
- int main (int argc, char $**$argv)

10.109.1 Detailed Description

10.109.2 Function Documentation

10.109.2.1 double fdmcol (double *x*, double *y*)

Parameters

in	*x*	input space variable x
in	*y*	input space variable y

Returns

1 if x is between 0.1 and 0.4, otherwise 0

The fdmcol function returns 1 if the space variable x fullfills

$$0.1 < x < 0.4,$$

otherwise 0.

10.109.2.2 double fx (double *x*, double *y*)

Parameters

in	*x*	input space variable x
in	*y*	input space variable y

Returns

$400x$

The fx function always returns $400x$.

10.109.2.3 double fy (double *x*, double *y*)

Parameters

in	*x*	input space variable x
in	*y*	input space variable y

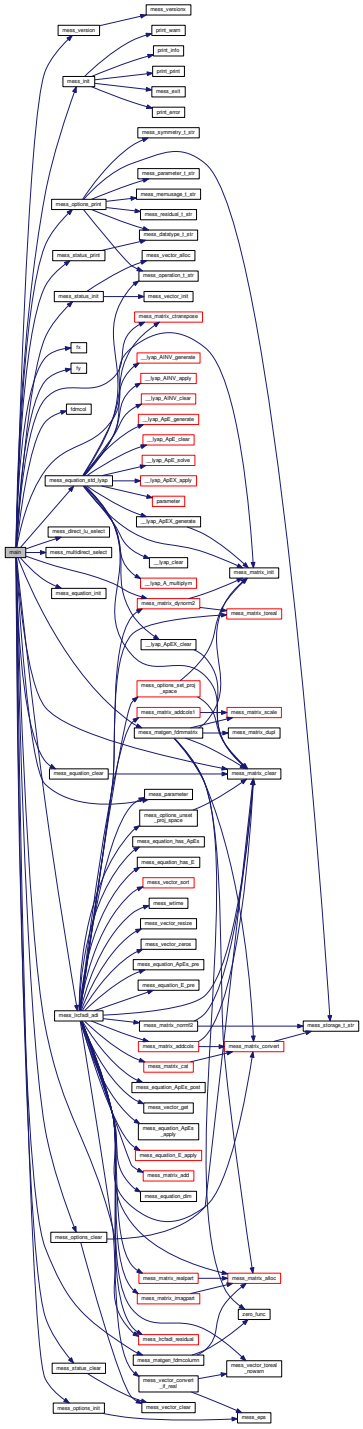
Returns

$10\sin(y)$

The fy function always returns $10\sin(y)$.

10.109.2.4 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.110 check_lrnm_dae1.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.110.1 Detailed Description

Author

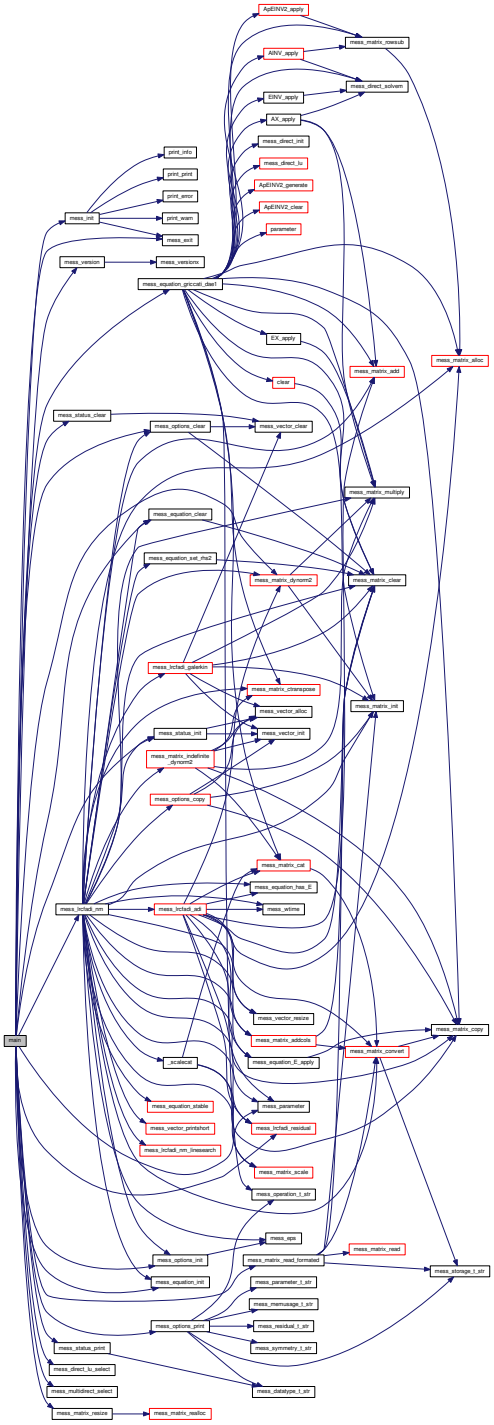
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_lrcfadi_nm function for the DAE 1 function handles defined in lrcf_adi/equation_glyap_dae1.c.

10.110.2 Function Documentation

10.110.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.111 check_lrnm_dae2.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.111.1 Detailed Description

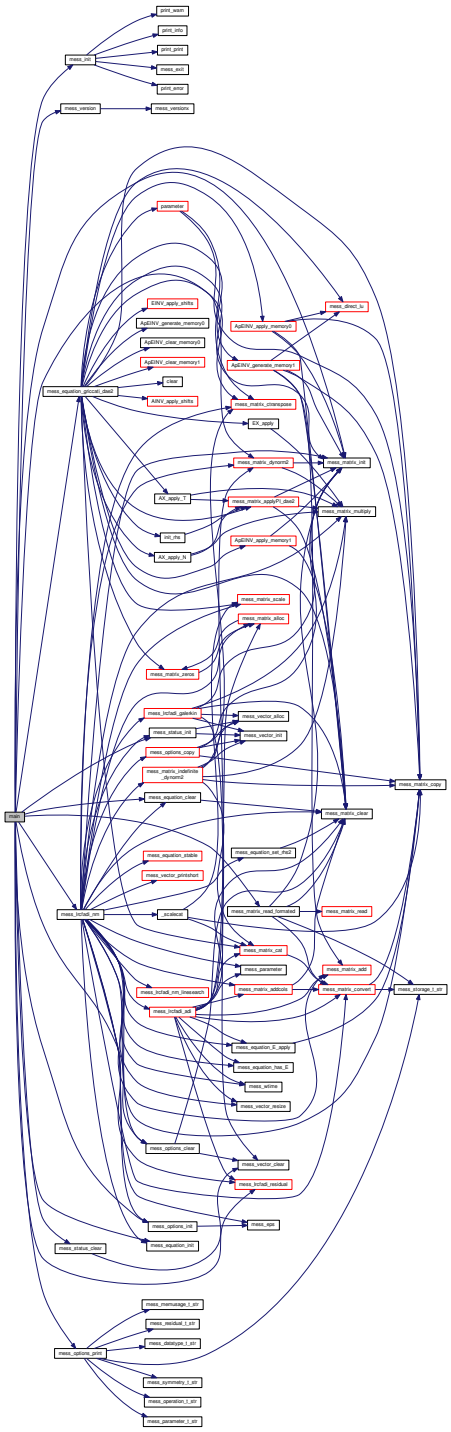
This function uses the mess_equation_griccati_dae2 function defined in equation_glyap_dae2.c.

Model description available under [2].

10.111.2 Function Documentation

10.111.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.112 check_lrnm_g.c File Reference

Functions

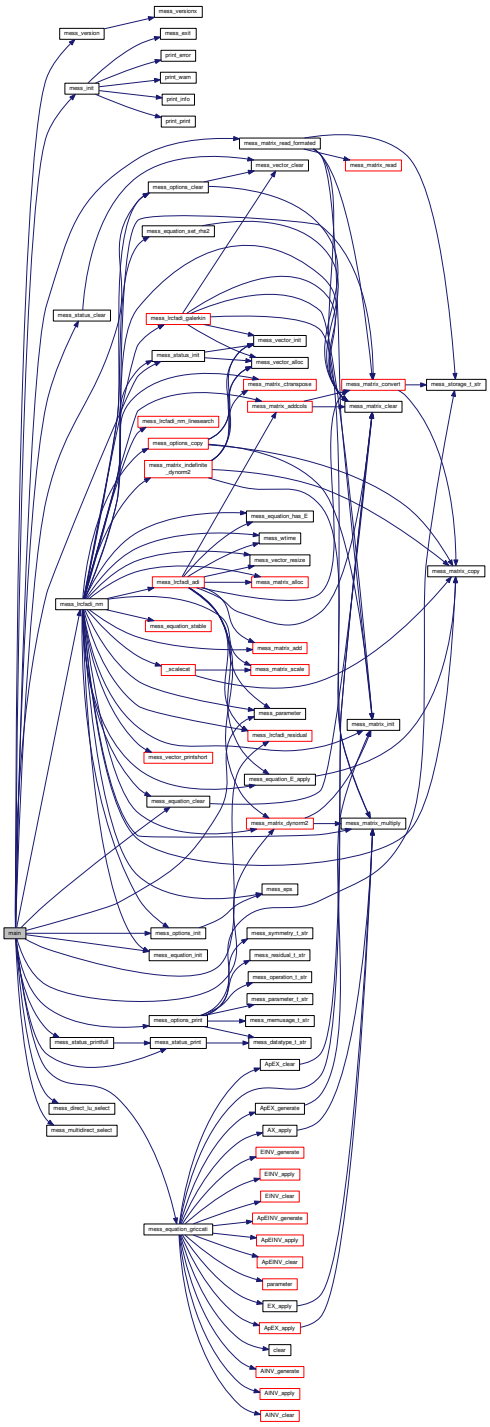
- int main (int argc, char **argv)

10.112.1 Detailed Description

10.112.2 Function Documentation

10.112.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.113 check_lrnm_so.c File Reference

Functions

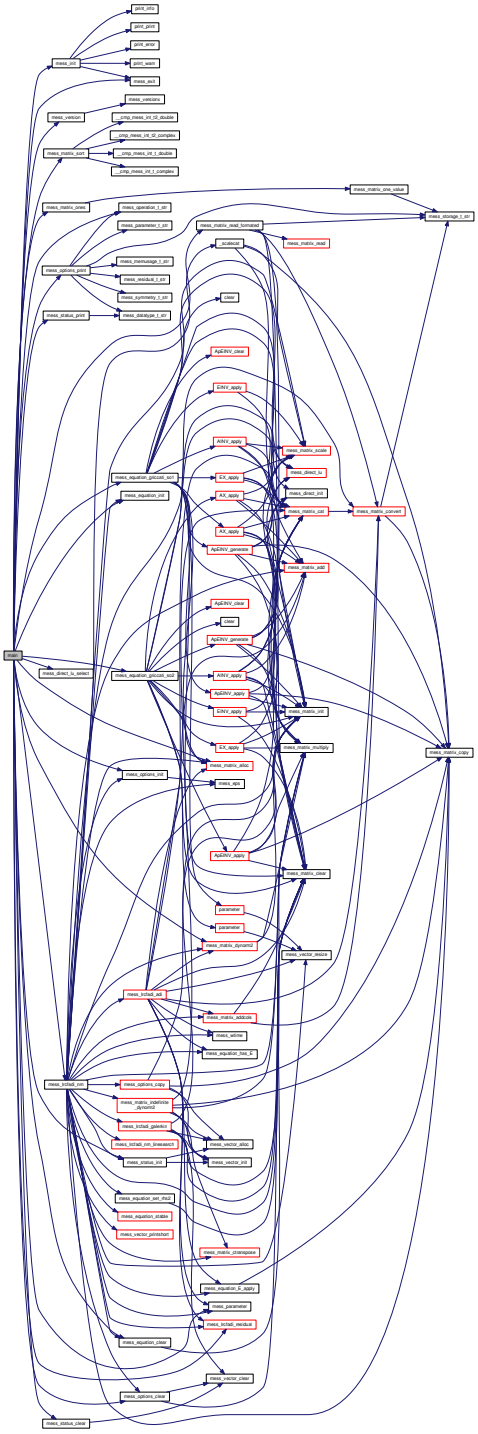
- int main (int argc, char ∗∗argv)

10.113.1 Detailed Description

10.113.2 Function Documentation

10.113.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.114 check_lrnm_unsym.c File Reference

Functions

- double fdmcol (double x, double y)

  Determine values used in a FDM matrix.
- double fdmcol2 (double x, double y)

  Determine values used in a FDM matrix.
- double fx (double x, double y)

  Determine values used in a FDM matrix.
- double fy (double x, double y)

  Determine values used in a FDM matrix.
- int main (int argc, char **argv)

10.114.1 Detailed Description

10.114.2 Function Documentation

10.114.2.1 double fdmcol (double *x*, double *y*)

Parameters

in	*x*	input space variable x
in	*y*	input space variable y

Returns

1 if x is between 0.1 and 0.4, otherwise 0

The fdmcol function returns 1 if the space variable x fullfills

$$0.1 < x < 0.4,$$

otherwise 0.

10.114.2.2 double fdmcol2 (double *x*, double *y*)

Parameters

in	*x*	input space variable x
in	*y*	input space variable y

Returns

1 if x is between 0.4 and 0.7, otherwise 0

The fdmcol2 function returns 1 if the space variable x fullfills

$$0.4 < x < 0.7,$$

otherwise 0.

10.114.2.3 double fx (double *x*, double *y*)

Parameters

in	*x*	input space variable x
in	*y*	input space variable y

Returns

$400x$

The fx function always returns $400x$.

10.114.2.4 double fy (double *x*, double *y*)

Parameters

in	*x*	input space variable x
in	*y*	input space variable y

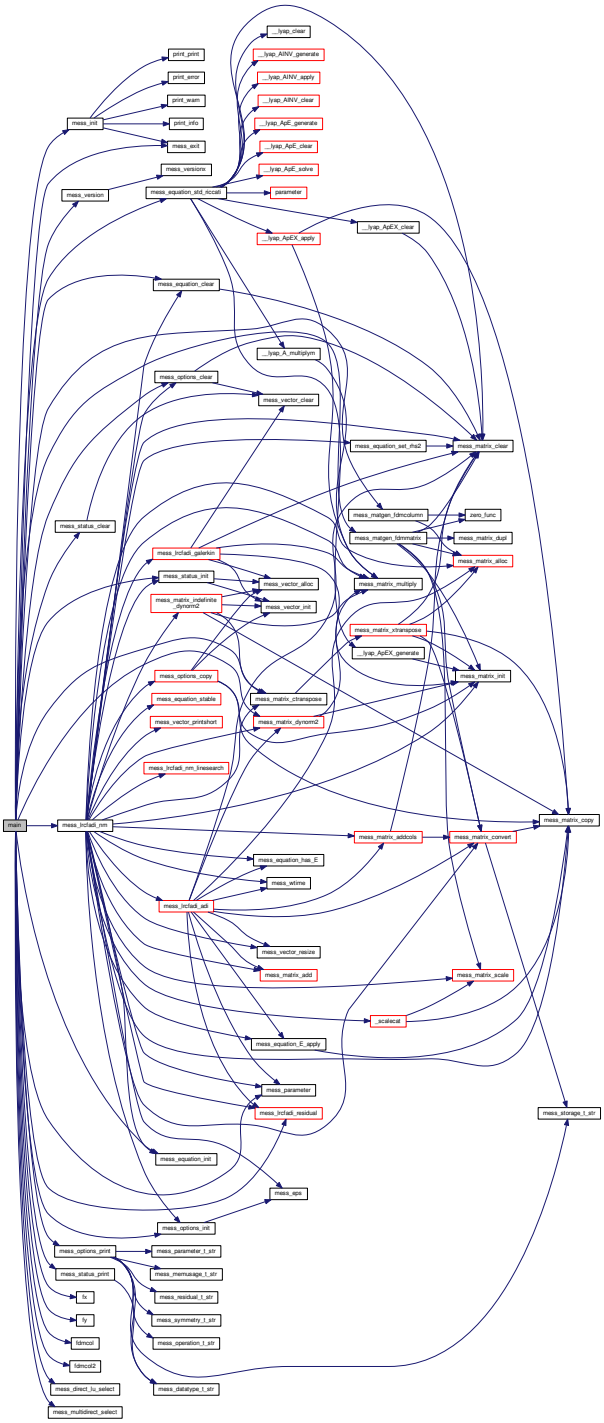
Returns

$10\sin(y)$

The fy function always returns $10\sin(y)$.

10.114.2.5 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.115 check_lyap_sign_fac.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.115.1 Detailed Description

This function checks the solution of the generalized Lyapunov Equation

$$AXE^T + EXA^T + BB^T = 0$$

and the Lyapunov Equations

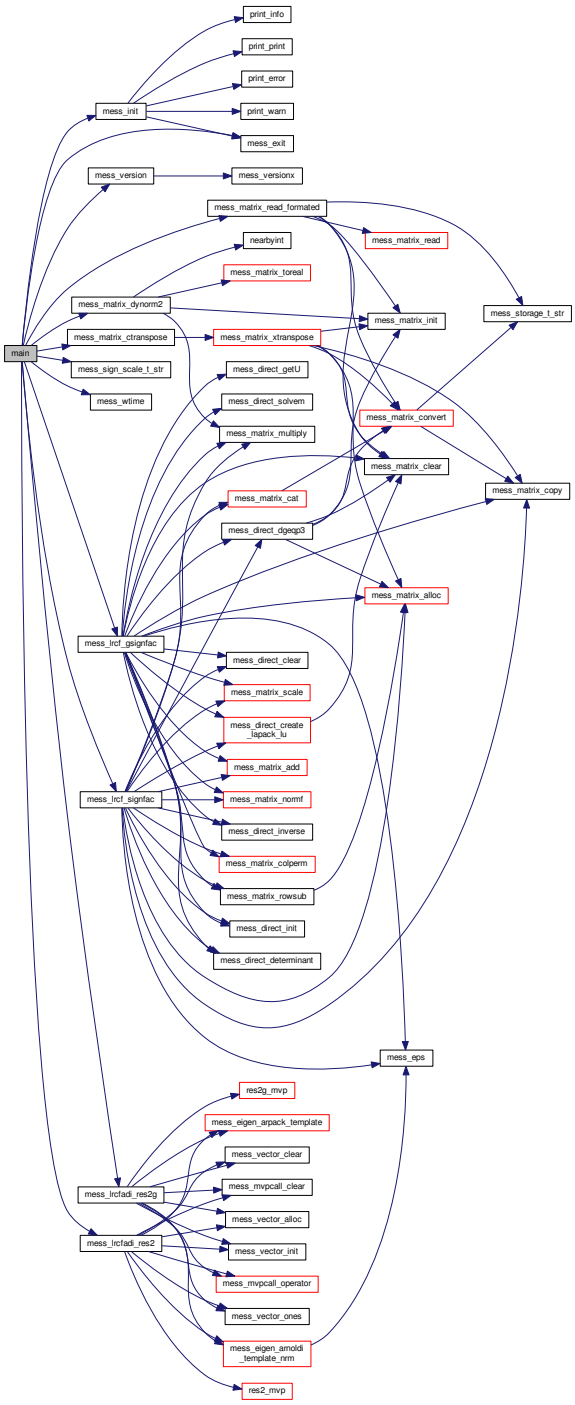
$$AX + XA^T + BB^T = 0$$

by computing the corresponding resiudal.
Time is measured for computing the solution of one Lyapunov Equation and its residual.

10.115.2 Function Documentation

10.115.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.116 check_matrix_biorth.c File Reference

Macros

- #define CHECKMARTIXBIORTH(VIN, WIN, VOUT, WOUT, EYE, TEMP1, ERR, EPS, DIFF)

Functions

- int main (int argc, char **argv)

10.116.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.116.2 Macro Definition Documentation

10.116.2.1 #define CHECKMARTIXBIORTH(*VIN, WIN, VOUT, WOUT, EYE, TEMP1, ERR, EPS,* DIFF)

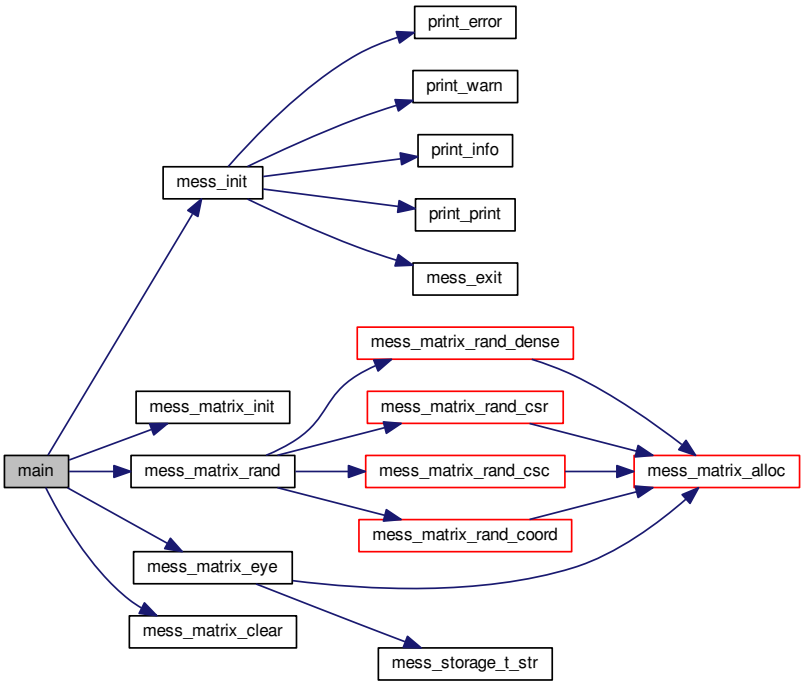
Value:

```
{                                   \
    CALL(mess_matrix_biorth(VIN,WIN,VOUT,WOUT))
                                    \
    CALL( mess_matrix_multiply(MESS_OP_TRANSPOSE, WOUT,
      MESS_OP_NONE, VOUT, TEMP1));                  \
    CALL(mess_matrix_diffnorm(TEMP1,EYE,&DIFF));
                                    \
    if(DIFF>EPS){                                                                   \
        ++ERR;                                                                      \
        printf("mess_matrix_mgsadd failed \n");                                     \
        printf("Error:%e\n",DIFF);                                                  \
    }                                                                               \
}
```

10.116.3 Function Documentation

10.116.3.1 int main (int *argc,* char ** *argv*)

Here is the call graph for this function:



10.117 check_matrix_colaxpy.c File Reference

Macros

- #define CHECKMATRIXOCLAXPY(Q, TEMP, COL, COEFF, TYPE, V1, VTEMP1, VTEMP2, VSOL, ERR, EPS, DIFF)

Functions

- int main (int argc, char **argv)

10.117.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.117.2 Macro Definition Documentation

10.117.2.1 #define CHECKMATRIXOCLAXPY(Q, TEMP, COL, COEFF, TYPE, V1, VTEMP1, VTEMP2, VSOL, ERR, EPS, DIFF)

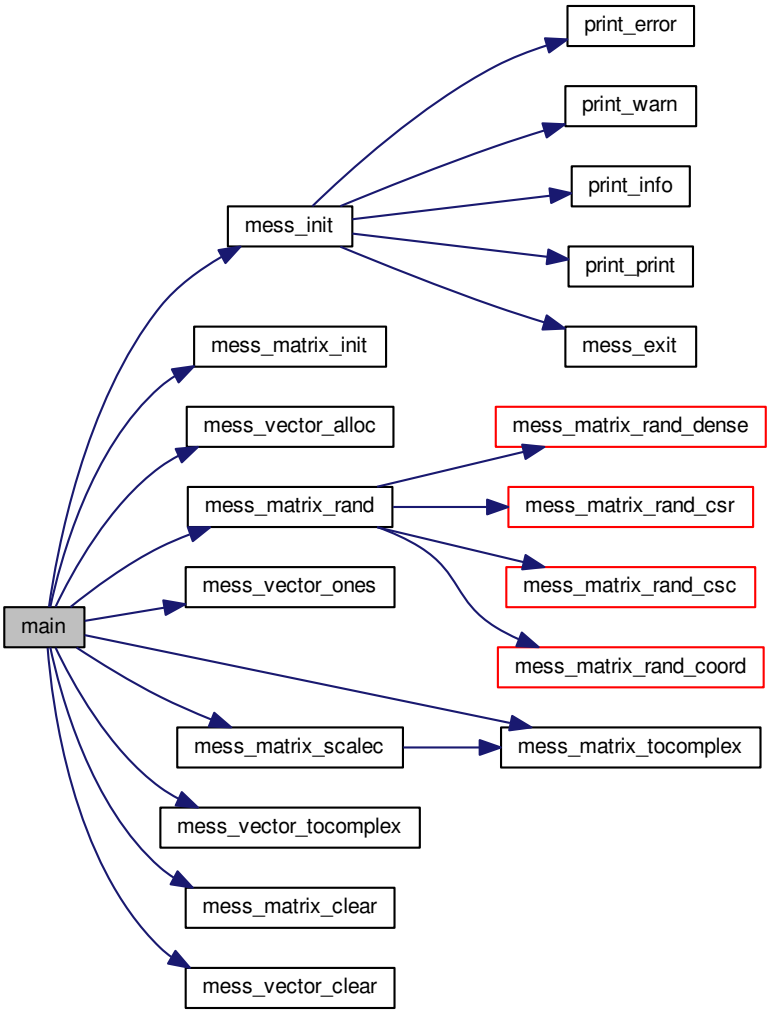
Value:

```
{           \
    CALL(mess_matrix_copy(Q,TEMP));
                \
    CALL(mess_vector_copy(V1,VTEMP1));
            \
    CALL(mess_matrix_getcol(TEMP,COL,VSOL));
                \
    if(MESS_REAL==TYPE){
      \
        CALL(mess_vector_axpy(COEFF,VTEMP1,VSOL));
                        \
    }else{
      \
        CALL(mess_vector_tocomplex(VTEMP1));
                                \
        CALL(mess_vector_tocomplex(VSOL));
                                \
        CALL(mess_vector_axpyc(COEFF,VTEMP1,VSOL));
                        \
    }
      \
    CALL(mess_matrix_colaxpy(COEFF,VTEMP1,COL,TEMP));
                \
    CALL(mess_matrix_getcol(TEMP,COL,VTEMP2));
                \
    CALL(mess_vector_diffnorm(VTEMP2,VSOL,&DIFF));
                        \
    if(DIFF>EPS){                                                                   \
        printf("Failed diff:%e\n",DIFF);                                            \
        ++ERR;                                                                      \
    }                                                                               \
}
```

10.117.3 Function Documentation

10.117.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.118 check_matrix_gbiorth.c File Reference

Macros

- #define CHECKMARTIXGBIORTH(E, VIN, WIN, VOUT, WOUT, EYE, TEMP1, TEMP2, ERR, EPS, DIFF)

Functions

- void generatespd (mess_matrix E, int dim)
- int main (int argc, char **argv)

10.118.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.118.2 Macro Definition Documentation

10.118.2.1 #define CHECKMARTIXGBIORTH(E, *VIN*, *WIN*, *VOUT*, *WOUT*, *EYE*, *TEMP1*, *TEMP2*, *ERR*, *EPS*, DIFF)

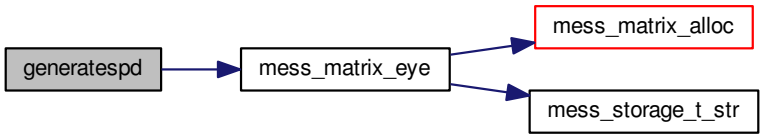
Value:

```
{                    \
    CALL(mess_matrix_gbiorth(E,VIN,WIN,VOUT,WOUT))
                \
    CALL( mess_matrix_multiply(MESS_OP_TRANSPOSE, WOUT,
      MESS_OP_NONE, E, TEMP1));           \
    CALL( mess_matrix_multiply(MESS_OP_NONE, TEMP1,
      MESS_OP_NONE, VOUT, TEMP2));                \
    CALL(mess_matrix_diffnorm(TEMP2,EYE,&DIFF));
                        \
    if(DIFF>EPS){                                                                  \
        ++ERR;                                                                     \
        printf("mess_matrix_mgsadd failed \n");                                    \
        printf("Error:%e\n",DIFF);                                                 \
    }                                                                              \
}
```

10.118.3 Function Documentation

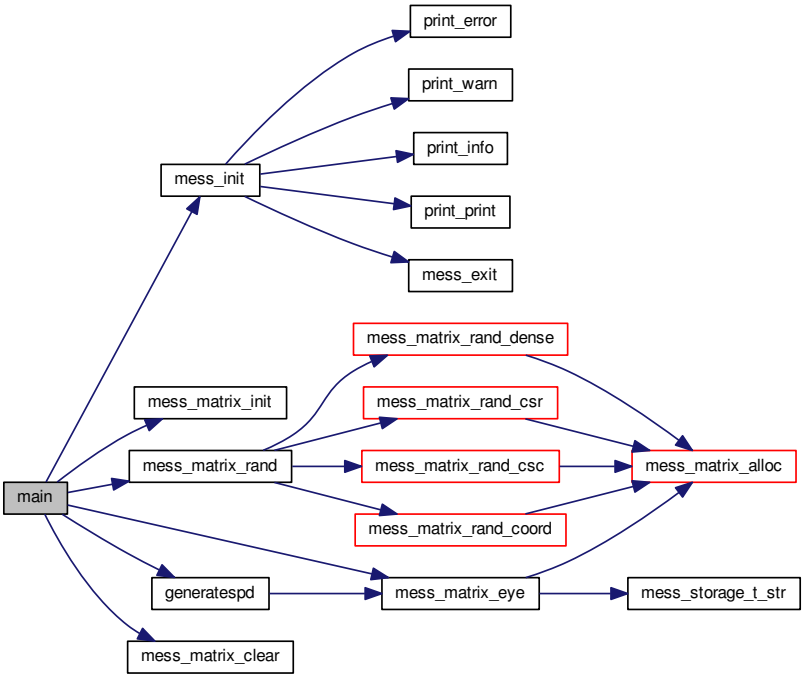
10.118.3.1 void generatespd (mess_matrix *E*, int *dim*)

Here is the call graph for this function:



10.118.3.2 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.119 check_matrix_null.c File Reference

Macros

- #define CHECK_MESS_MATRIX_NULL(Q, ZNULL, ZORTH, TEMP1, TEMP2, EYE, RANK, NRM, ERR, E↩ PS)

Functions

- int main (int argc, char **argv)

10.119.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

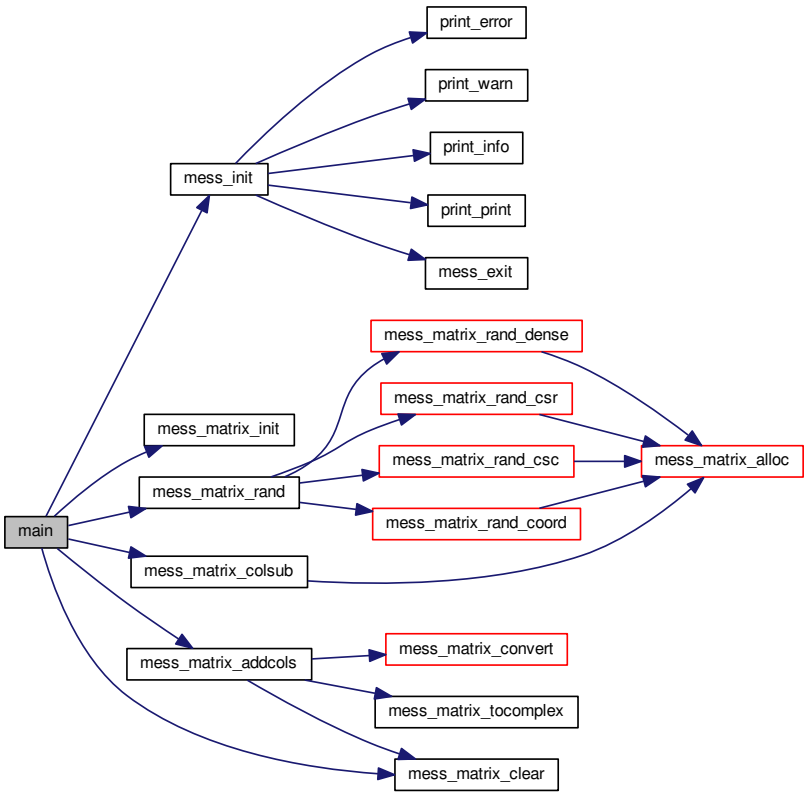
10.119.2 Macro Definition Documentation

10.119.2.1 #define CHECK_MESS_MATRIX_NULL(Q, ZNULL, ZORTH, TEMP1, TEMP2, EYE, RANK, NRM, ERR, EPS)

10.119.3 Function Documentation

10.119.3.1 int main (int argc, char ** argv)

Here is the call graph for this function:



10.120 check_matrix_sum.c File Reference

Macros

- #define CHECKMATRIXSUM(A, TEMP, VSUM1, VSUM2, ERR, EPS, DIFF)

Functions

- int main (int argc, char **argv)

10.120.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.120.2 Macro Definition Documentation

10.120.2.1 #define CHECKMATRIXSUM(A, *TEMP, VSUM1, VSUM2, ERR, EPS,* DIFF)

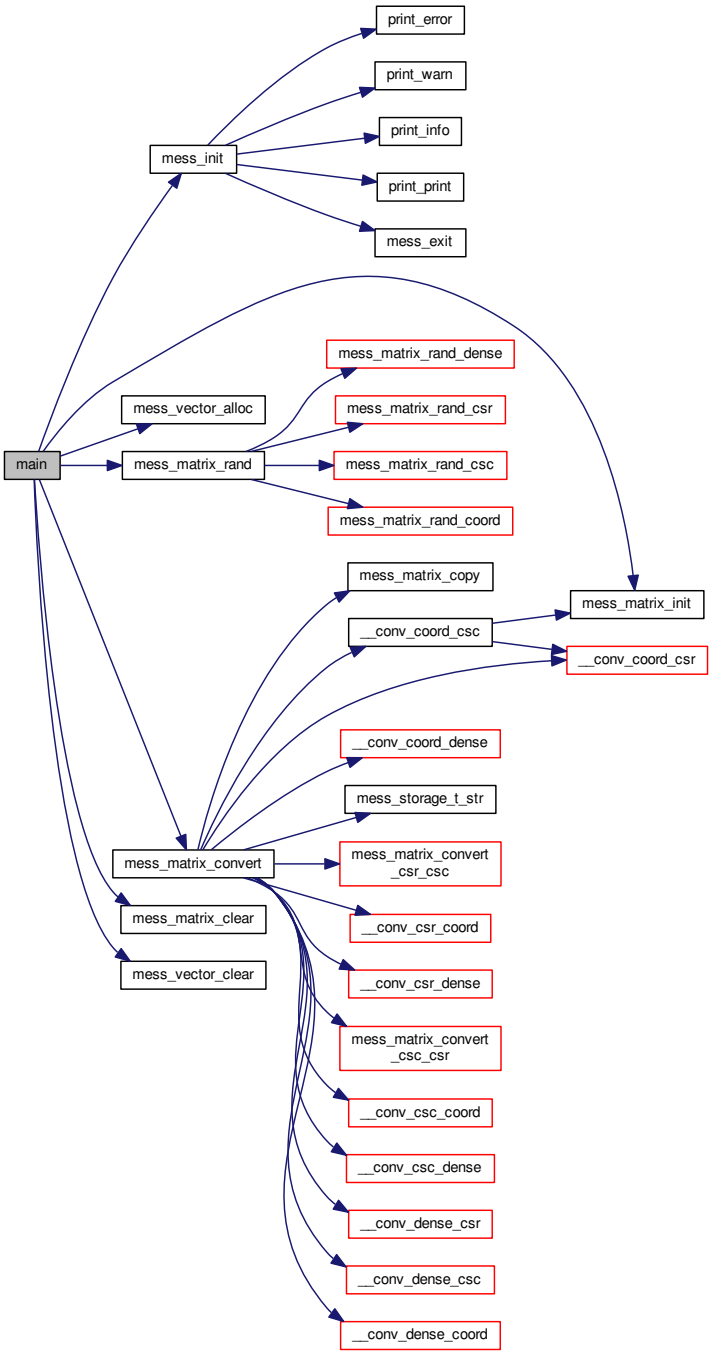
Value:

```
{                                   \
    CALL(mess_matrix_rowsums(A,VSUM1));
      \
    CALL(mess_matrix_ctranspose(A,TEMP));
      \
    CALL(mess_matrix_colsums(TEMP,VSUM2));
      \
    CALL(mess_vector_diffnorm(VSUM1,VSUM2,&DIFF));                    \
    if(DIFF>10*EPS){                                                  \
        printf("Failed:\n");                                          \
        printf("A: %s\n", #A);                                        \
        printf("TEMP: %s\n", #TEMP);                                  \
        printf("VSUM1: %s\n", #VSUM1);                                \
        printf("VSUM2: %s\n", #VSUM2);                                \
        printf("Diff: %e\n",DIFF);                                    \
        ++ERR;                                                        \
    }                                                                 \
}
```

10.120.3 Function Documentation

10.120.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.121 check_matrixio.c File Reference

Functions

- int main (int argc, char **argv)

10.121.1 Detailed Description

Author

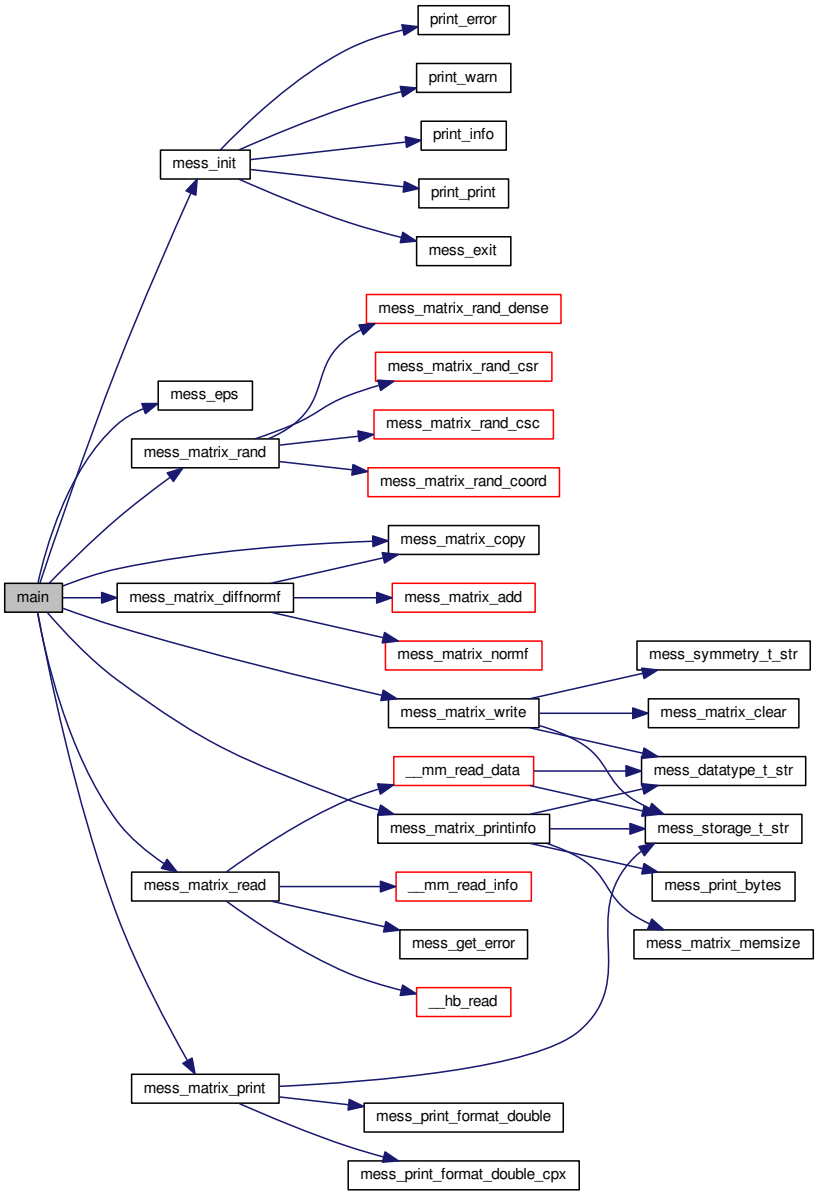
Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_matrix_read and mess_matrix_write functions defined in write.c and read.c that means it checks if the reading and writing functions are working for vectors.

10.121.2 Function Documentation

10.121.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.122 check_mgs.c File Reference

Macros

- #define CHECKMGS(A, Q, R, EYE, TEMP1, TEMP2, ERR, EPS, DIFF)

Functions

- int main (int argc, char **argv)

10.122.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.122.2 Macro Definition Documentation

10.122.2.1 #define CHECKMGS(A, Q, R, EYE, TEMP1, TEMP2, ERR, EPS, DIFF)

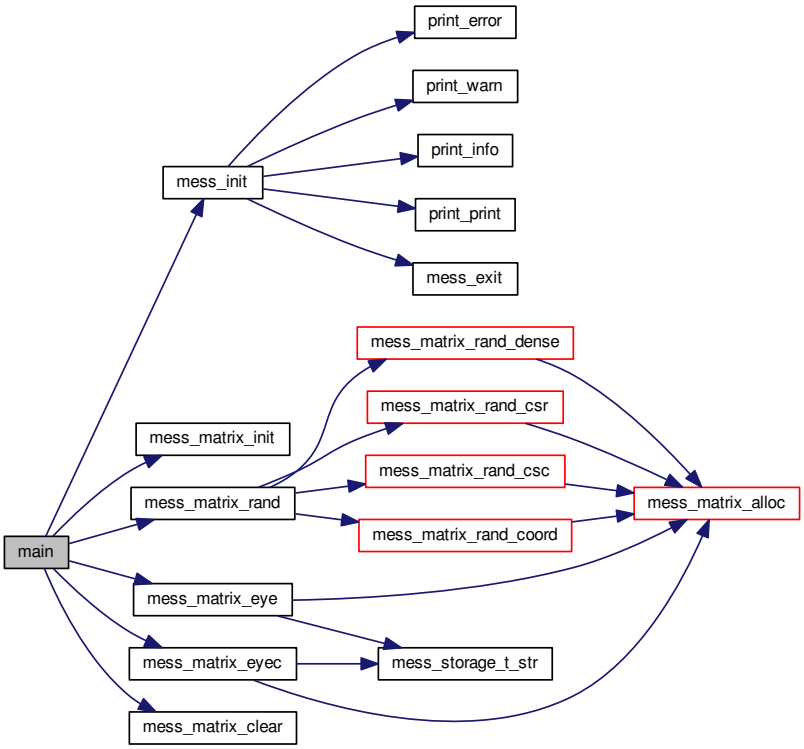
Value:

```
{                                   \
    mess_matrix_mgs (A, Q, R);                                          \
    if(R!=NULL){                                                        \
        mess_matrix_multiply(MESS_OP_NONE,
      Q, MESS_OP_NONE, R, TEMP1);                 \
        mess_matrix_diffnorm(TEMP1,A,&DIFF);
                \
        if(DIFF>EPS){++ERR;}                                                \
    }                                                                   \
    mess_matrix_copy(Q,TEMP1);
      \
    mess_matrix_multiply(MESS_OP_HERMITIAN,
      Q, MESS_OP_NONE, TEMP1, TEMP2); \
    mess_matrix_diffnorm(TEMP2,EYE,&DIFF);
       \
    if(DIFF>eps){++ERR;}                                                \
}
```

10.122.3 Function Documentation

10.122.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.123 check_mgs_inplace.c File Reference

Macros

- #define CHECKMGSINPLACE(Q, QCOPY, EYE, TEMP1, TEMP2, ERR, EPS, DIFF)

Functions

- int main (int argc, char **argv)

10.123.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.123.2 Macro Definition Documentation

10.123.2.1 #define CHECKMGSINPLACE(Q, *QCOPY, EYE, TEMP1, TEMP2, ERR, EPS,* DIFF)

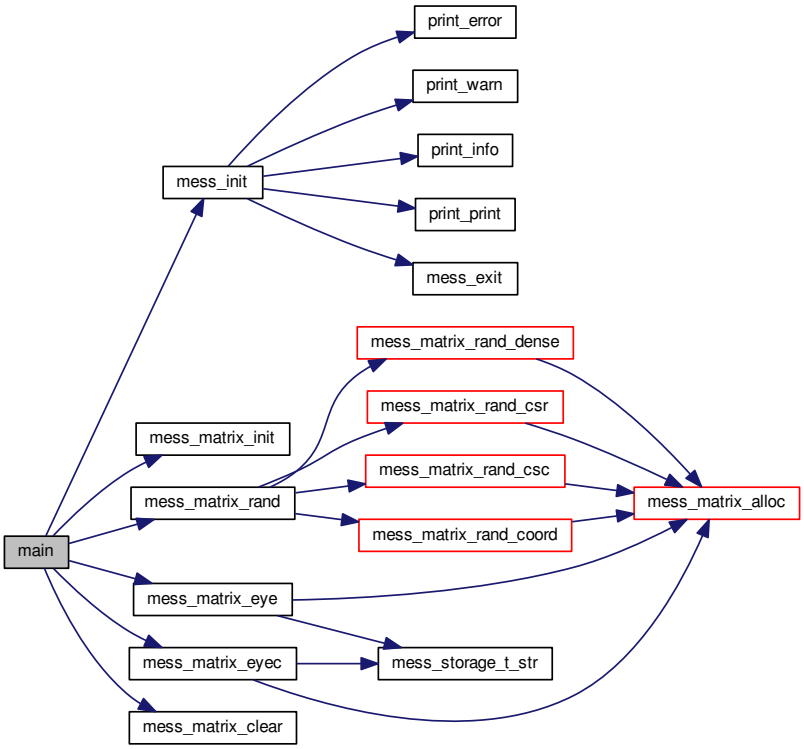
Value:

```
{                \
    mess_matrix_copy(QCOPY,Q);
      \
    mess_matrix_mgs_inplace (Q);
      \
    mess_matrix_copy(Q,TEMP1);
      \
    mess_matrix_multiply(MESS_OP_HERMITIAN,
      Q, MESS_OP_NONE, TEMP1, TEMP2); \
    mess_matrix_diffnorm(TEMP2,EYE,&DIFF);
       \
    if(DIFF>eps){++ERR;}                                          \
}
```

10.123.3 Function Documentation

10.123.3.1 int main (int *argc,* char ** *argv*)

Here is the call graph for this function:



10.124 check_mgsadd.c File Reference

Macros

- #define CHECKMGSADD(Z, V, EYE, TEMP1, TEMP2, TEMP3, ERR, EPS, DIFF)
- #define CHECKMGSADDE(Z, V, EYE, E, CHOLSOLVER, TEMP1, TEMP2, TEMP3, ERR, EPS, DIFF)

Functions

- void generatespd (mess_matrix E, int dim)
- int main (int argc, char **argv)

10.124.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.124.2 Macro Definition Documentation

10.124.2.1 #define CHECKMGSADD(Z, V, EYE, TEMP1, TEMP2, TEMP3, ERR, EPS, DIFF)

Value:

```
{                                   \
        CALL(mess_matrix_mgs(Z,TEMP1,NULL));
                                    \
        CALL(mess_matrix_mgs_add (TEMP1, V, NULL ));
                                    \
        CALL(mess_matrix_copy(TEMP1,TEMP2));
                                    \
        if(MESS_IS_COMPLEX(Z)||MESS_IS_COMPLEX(V)){
                                    \
        CALL( mess_matrix_multiply(MESS_OP_NONE, TEMP2,
      MESS_OP_HERMITIAN, TEMP1, TEMP3));      \
        }else{
      \
            CALL( mess_matrix_multiply(
      MESS_OP_NONE, TEMP2, MESS_OP_TRANSPOSE, TEMP1, TEMP3));  \
        }
      \
        CALL(mess_matrix_diffnorm(TEMP3,EYE,&
      DIFF));                                         \
        if(DIFF>EPS){                                                                   \
            ++ERR;                                                                      \
            printf("mess_matrix_mgsadd failed \n");                                     \
            printf("Error:%e\n",DIFF);                                                  \
        }                                                                               \
}
```

10.124.2.2 #define CHECKMGSADDE(Z, V, EYE, E, CHOLSOLVER, TEMP1, TEMP2, TEMP3, ERR, EPS, DIFF)

Value:

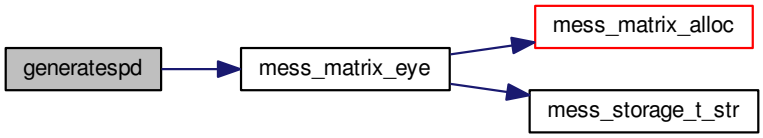
```
{                   \
        CALL(mess_direct_init(&CHOLSOLVER));
                    \
        CALL(mess_direct_create_cholesky(
      E,CHOLSOLVER));                                   \
        CALL(mess_direct_getL(CHOLSOLVER,TEMP1));
                    \
        CALL( mess_matrix_multiply(
      MESS_OP_TRANSPOSE, TEMP1, MESS_OP_NONE, Z, TEMP2));
      \
        CALL(mess_matrix_mgs(TEMP2,TEMP3,NULL));
                    \
        CALL(mess_matrix_backslashm(
      MESS_OP_TRANSPOSE,TEMP1,TEMP3,TEMP2));                \
        CALL(mess_matrix_mgs_add(TEMP2,V,E));
                    \
        CALL(mess_matrix_copy(TEMP2,TEMP1));
                    \
        if(MESS_IS_COMPLEX(Z)||MESS_IS_COMPLEX(V)){
                    \
        CALL( mess_matrix_multiply(
```

```
      MESS_OP_HERMITIAN, TEMP1, MESS_OP_NONE, E, TEMP3));          \
        }else{
      \
        CALL( mess_matrix_multiply(
      MESS_OP_TRANSPOSE, TEMP1, MESS_OP_NONE, E, TEMP3));          \
        }
      \
        CALL( mess_matrix_multiply(MESS_OP_NONE, TEMP3,
      MESS_OP_NONE, TEMP2, TEMP1));          \
        CALL(mess_matrix_diffnorm(TEMP1,EYE,&
      DIFF));                                          \
        if(DIFF>EPS){                                                                    \
            ++ERR;                                                                       \
            printf("mess_matrix_mgsadd failed \n");                                      \
            printf("Error:%e\n",DIFF);                                                   \
        }
      \
        CALL(mess_direct_clear(&CHOLSOLVER));
                  \
}
```

10.124.3 Function Documentation

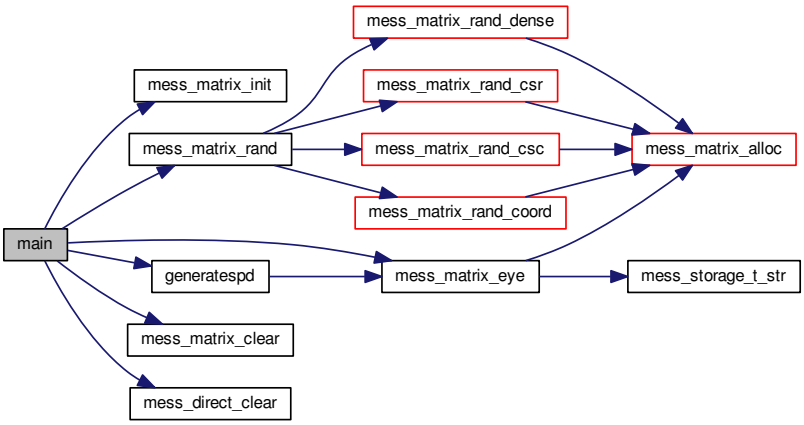
10.124.3.1 void generatespd (mess_matrix *E*, int *dim*)

Here is the call graph for this function:



10.124.3.2 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.125 check_mul.c File Reference

Macros

- #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed =================\n", #X); } else {fprintf(stderr, "test %35s passed.\n",#X); }

Functions

- int check_mul (mess_matrix A, mess_operation_t opA, mess_matrix B, mess_operation_t opB, mess_matrix C, mess_matrix Csoll)
- int main (int argc, char **argv)

Variables

- double matrix_data [16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}
- double matrix_NN [16] = {121,0,1364,1333,984,1492,372,924,602,1032,1089,1419,442,528,434,1008}
- double matrix_NT [16] = {461,600,341,504,600,1060,0,924,341,0,2050,1419,504,924,1419,3613}
- double matrix_TN [16] = { 1082,132,1023,154,132,2392,1806,696,1023,1806,2938,0,154,696,0,772}
- double matrix_TT [16] = {121,984,602,442,0,1492,1032,528,1364,372,1089,434,1333,924,1419,1008}
- mess_double_cpx_t matrixi_data [16] ={11+11*I,0+12*I,31,14*I,12,22+22*I,0,42+24*I,31*I,0,33+33*I,43,14,24+42*I,43*I,I}
- mess_double_cpx_t matrix_NNi [16]
- mess_double_cpx_t matrix_NTi [16]
- mess_double_cpx_t matrix_NHi [16]
- mess_double_cpx_t matrix_TNi [16]
- mess_double_cpx_t matrix_TTi [16]
- mess_double_cpx_t matrix_THi [16]
- mess_double_cpx_t matrix_NiN [16]
- mess_double_cpx_t matrix_NiT [16]
- mess_double_cpx_t matrix_TiN [16]
- mess_double_cpx_t matrix_TiT [16]
- mess_double_cpx_t matrix_HiN [16]
- mess_double_cpx_t matrix_HiT [16]
- mess_double_cpx_t matrix_NiNi [16]
- mess_double_cpx_t matrix_NiTi [16]
- mess_double_cpx_t matrix_NiHi [16]
- mess_double_cpx_t matrix_TiNi [16]
- mess_double_cpx_t matrix_TiTi [16]
- mess_double_cpx_t matrix_TiHi [16]
- mess_double_cpx_t matrix_HiNi [16]
- mess_double_cpx_t matrix_HiTi [16]
- mess_double_cpx_t matrix_HiHi [16]

10.125.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

- MESS_OP_NONE ($op(X) = X$),
- MESS_OP_TRANSPOSE ($op(X) = X^T$),
- MESS_OP_HERMITIAN ($op(X) = X^H$).

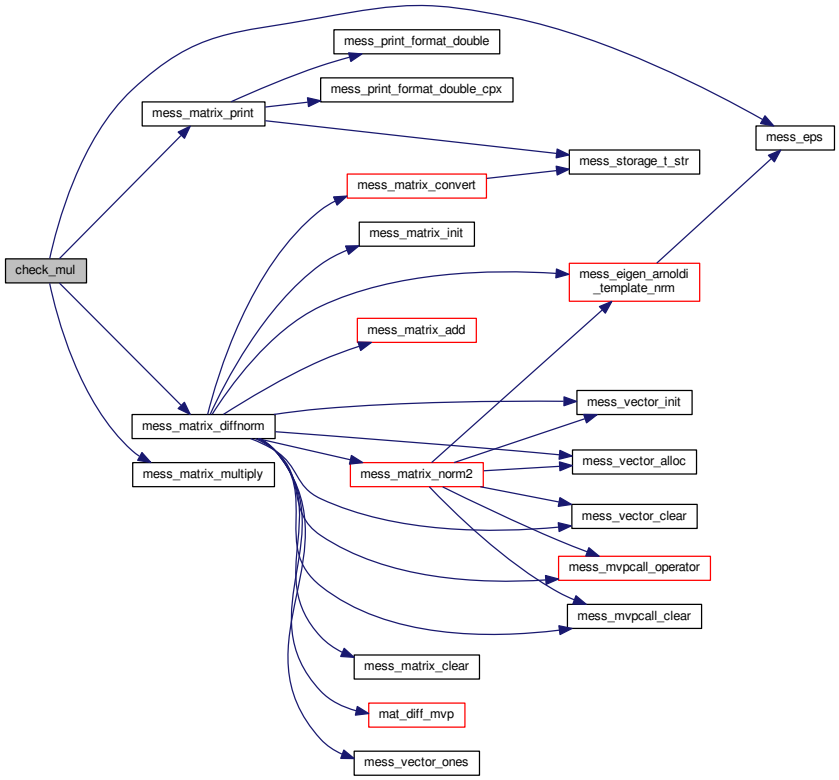
10.125.2 Macro Definition Documentation

10.125.2.1 #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed ================\n", #X); } else {fprintf(stderr, "test %35s passed.\n",#X); }

10.125.3 Function Documentation

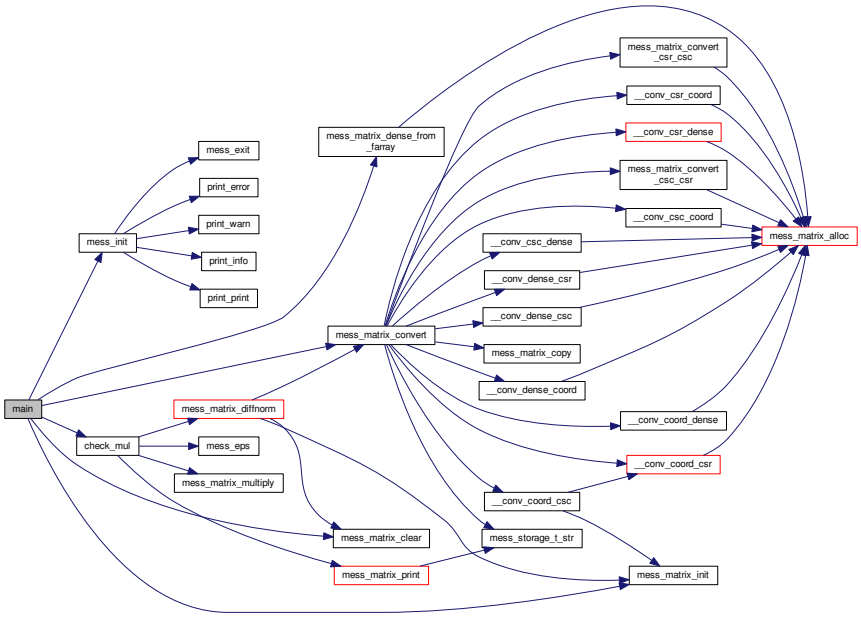
10.125.3.1 int check_mul (mess_matrix *A*, mess_operation_t *opA*, mess_matrix *B*, mess_operation_t *opB*, mess_matrix *C*, mess_matrix *Csoll*)

Here is the call graph for this function:



10.125.3.2 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.125.4 Variable Documentation

10.125.4.1 double matrix_data[16] = {11,0,31,0,12,22,0,42,0,0,33,43,14,24,0,0}

10.125.4.2 mess_double_cpx_t matrix_HiHi[16]

Initial value:

```
= {  0 -        1543*I, 984 -        732*I,-762 -        1364*I, -891 -        672*I,
    -984 -        732*I,  0 -        3452*I,  660 -        1806*I,  -438 -        1644*I,
    762 -        1364*I,  -660 -        1806*I,  0 -        4988*I, -1028 -        1419*I,
    877 -        658*I,  372 -        1662*I, 985 -        1462*I,   -1 -        4385*I}
```

10.125.4.3 mess_double_cpx_t matrix_HiN[16]

Initial value:

```
={ 1082 -        121*I, 132              ,1023 -        1364*I, 154 -        1333*I,
    132 -        984*I, 2392 -        1492*I,   1806 -        372*I, 696 -        966*I,
    1023 -        602*I,1806 -        1032*I, 2938 -        1089*I, 0 -        1462*I,
    154 -        442*I, 696 -        528*I,  0 -        434*I,  772 -        1008*I}
```

10.125.4.4 mess_double_cpx_t matrix_HiNi[16]

Initial value:

```
= {  1543, 732 +        984*I, 1364 -        762*I, 672 -        891*I,
    732 -        984*I,   3452, 1806 +        660*I,1644 -        438*I,
    1364 +        762*I, 1806 -        660*I, 4988, 1419 -        1028*I,
    672 +        891*I, 1644 +        438*I, 1419 +        1028*I,  4386     }
```

10.125.4.5 mess_double_cpx_t matrix_HiT[16]

Initial value:

```
={ 121 -        461*I, 984 -        600*I,  602 -        341*I,442 -        518*I,
    0 -        600*I,  1492 -       1060*I, 1032 , 528 -        948*I,
    1364 -        341*I,  372              ,    1089 -       2050*I,    434 -       1419*I,
    1333 -        504*I,  924 -        924*I,   1419 -       1419*I,  1008 -       3613*I }
```

10.125.4.6 mess_double_cpx_t matrix_HiTi[16]

Initial value:

```
= {   242 +        621*I,  984 -        468*I,  1966 +        682*I, 1775 -        364*I,
    984 -        468*I,   2984 +       1332*I,  1404 +       1806*I, 1494 -        252*I,
    1966 +        682*I, 1404 +       1806*I,2178 +        888*I, 1896 -       1419*I,
    1789 -        350*I, 1476 -        186*I,1853 -       1376*I, 2017 -       2841*I}
```

10.125.4.7 mess_double_cpx_t matrix_NHi[16]

Initial value:

```
={  461 -        121*I,       600              ,           341 -       1364*I,        504 -       1333*I,
    600 -        984*I,         1060 -       1492*I,     0 -        372*I,     924 -        924*I,
    341 -        602*I,      0 -       1032*I,     2050 -       1089*I,      1419 -       1419*I,
    504 -        456*I,      924 -        552*I,  1419 -        434*I,  3613 -       1008*I }
```

10.125.4.8 mess_double_cpx_t matrix_NiHi[16]

Initial value:

```
= {  1543,    732 +        984*I,        1364 -        762*I,     658 -        877*I,
    732 -        984*I,   3452 ,  1806 +        660*I,      1662 -        372*I,
    1364 +        762*I,  1806 -        660*I,  4988,  1462 -        985*I,
    658 +        877*I,   1662 +        372*I,  1462 +        985*I, 4386  }
```

10.125.4.9 mess_double_cpx_t matrix_NiN[16]

Initial value:

```
={   121 +       1082*I, 0 +        132*I, 1364 +       1023*I, 1333 +        154*I,
    984 +        132*I, 1492 +       2392*I, 372 +       1806*I, 924 +        738*I,
    602 +       1023*I, 1032 +       1806*I, 1089 +       2938*I, 1419 +         43*I,
    442 +        154*I, 528 +        696*I, 434              ,  1008 +        772*I}
```

10.125.4.10 mess_double_cpx_t matrix_NiNi[16]

Initial value:

```
= {   0 +       1543*I, -984 +        732*I,  762 +       1364*I,  877 +        658*I,
    984 +        732*I, 0 +       3452*I,   -660 +       1806*I,  372 +       1662*I,
    -762 +       1364*I, 660 +       1806*I, 0 +       4988*I,      985 +       1462*I,
    -891 +        672*I,-438 +       1644*I, -1028 +       1419*I, -1 +       4385*I, }
```

10.125.4.11 mess_double_cpx_t matrix_NiT[16]

Initial value:

```
={  461 +        121*I, 600 +        984*I, 341 +        602*I, 504 +        456*I,
    600              ,     1060 +       1492*I,    0 +       1032*I, 924 +        552*I,
    341 +       1364*I,   0 +        372*I, 2050 +       1089*I,  1419 +        434*I,
    504 +       1333*I,  924 +        924*I, 1419 +       1419*I,3613 +       1008*I}
```

10.125.4.12 mess_double_cpx_t matrix_NiTi[16]

Initial value:

```
= {   -621 +        242*I, 468 +        984*I, -682 +       1966*I,  350 +       1789*I,
    468 +        984*I, -1332 +       2984*I, -1806 +       1404*I,  186 +       1476*I,
    -682 +       1966*I, -1806 +       1404*I, -888 +       2178*I,  1376 +       1853*I,
    350 +       1789*I,  186 +       1476*I, 1376 +       1853*I,   2840 +       2016*I }
```

10.125.4.13 double matrix_NN[16] = {121,0,1364,1333,984,1492,372,924,602,1032,1089,1419,442,528,434,1008}

10.125.4.14 mess_double_cpx_t matrix_NNi[16]

Initial value:

```
={121 +        461*I, 0 +        600*I, 1364 +        341*I, 1333 +        504*I,
    984 +        600*I, 1492 +       1060*I, 372 ,  924 +        924*I,
    602 +        341*I, 1032,   1089 +       2050*I, 1419 +       1419*I,
    442 +        518*I, 528 +        948*I,  434 +       1419*I, 1008 +       3613*I}
```

10.125.4.15 double matrix_NT[16] = {461,600,341,504,600,1060,0,924,341,0,2050,1419,504,924,1419,3613}

10.125.4.16 mess_double_cpx_t matrix_NTi[16]

Initial value:

```
={ 461 +        121*I, 600              ,     341 +       1364*I, 504 +       1333*I,
    600 +        984*I, 1060 +       1492*I,  0 +        372*I, 924 +        924*I,
    341 +        602*I, 0 +       1032*I, 2050 +       1089*I, 1419 +       1419*I,
    504 +        456*I,    924 +        552*I,    1419 +        434*I,    3613 +       1008*I}
```

10.125.4.17 mess_double_cpx_t matrix_THi[16]

Initial value:

```
= { 121 -       1082*I,      984 -        132*I,    602 -       1023*I,    442 -        154*I,
    0 -        132*I,  1492 -       2392*I,    1032 -       1806*I,    528 -        696*I,
    1364 -       1023*I, 372 -       1806*I,    1089 -       2938*I,    434              ,
    1333 -        154*I, 924 -        738*I,    1419 -         43*I,  1008 -        772*I}
```

10.125.4.18 mess_double_cpx_t matrix_TiHi[16]

Initial value:

```
= {  242 -        621*I, 984 +        468*I, 1966 -        682*I,1775 +        364*I,
    984 +        468*I, 2984 -       1332*I, 1404 -       1806*I, 1494 +        252*I,
    1966 -        682*I, 1404 -       1806*I,  2178 -        888*I, 1896 +       1419*I,
    1789 +        350*I, 1476 +        186*I,  1853 +       1376*I,  2017 +       2841*I}
```

10.125.4.19 mess_double_cpx_t matrix_TiN[16]

Initial value:

```
={ 1082 +        121*I,132              ,  1023 +       1364*I, 154 +       1333*I,
    132 +        984*I, 2392 +       1492*I, 1806 +        372*I, 696 +        966*I,
    1023 +        602*I, 1806 +       1032*I,2938 +       1089*I, 0 +       1462*I,
    154 +        442*I,  696 +        528*I, 0 +        434*I,772 +       1008*I}
```

10.125.4.20 mess_double_cpx_t matrix_TiNi[16]

Initial value:

```
= { 621 +        242*I,   -468 +        984*I,  682 +       1966*I,  -364 +       1775*I,
    -468 +        984*I,  1332 +       2984*I,  1806 +       1404*I, -252 +       1494*I,
    682 +       1966*I,  1806 +       1404*I,   888 +       2178*I,  -1419 +       1896*I,
    -364 +       1775*I, -252 +       1494*I,  -1419 +       1896*I, -2842 +       2016*I }
```

10.125.4.21 mess_double_cpx_t matrix_TiT[16]

Initial value:

```
={ 121 +        461*I,  984 +        600*I, 602 +        341*I,442 +        518*I,
    0 +        600*I,  1492 +       1060*I,  1032     , 528 +        948*I,
    1364 +        341*I, 372              , 1089 +       2050*I, 434 +       1419*I,
    1333 +        504*I, 924 +        924*I, 1419 +       1419*I, 1008 +       3613*I}
```

10.125.4.22 mess_double_cpx_t matrix_TiTi[16]

Initial value:

```
= { 0 +       1543*I,   984 +        732*I, -762 +       1364*I, -891 +        672*I,
    -984 +        732*I,  0 +       3452*I,  660 +       1806*I, -438 +       1644*I,
    762 +       1364*I, -660 +       1806*I, 0 +       4988*I,  -1028 +       1419*I,
    877 +        658*I,  372 +       1662*I, 985 +       1462*I,  -1 +       4385*I}
```

10.125.4.23 double matrix_TN[16] = { 1082,132,1023,154,132,2392,1806,696,1023,1806,2938,0,154,696,0,772}

10.125.4.24 mess_double_cpx_t matrix_TNi[16]

Initial value:

```
= { 1082 +        121*I,  132 +        984*I, 1023 +        602*I, 154 +        442*I,
    132   ,  2392 +       1492*I,  1806 +       1032*I,  696 +        528*I,
    1023 +       1364*I,     1806 +        372*I,          2938 +       1089*I,     0 +        434*I,
    154 +       1333*I,  696 +        966*I,  0 +       1462*I,  772 +       1008*I }
```

10.125.4.25 double matrix_TT[16] = {121,984,602,442,0,1492,1032,528,1364,372,1089,434,1333,924,1419,1008}

10.125.4.26 mess_double_cpx_t matrix_TTi[16]

Initial value:

```
= { 121 +       1082*I, 984 +        132*I, 602 +       1023*I, 442 +        154*I,
    0 +        132*I,  1492 +       2392*I, 1032 +       1806*I, 528 +        696*I,
    1364 +       1023*I, 372 +       1806*I, 1089 +       2938*I, 434               ,
    1333 +        154*I,924 +        738*I,  1419 +         43*I, 1008 +        772*I }
```

10.125.4.27 mess_double_cpx_t matrixi_data[16] ={11+11∗I,0+12∗I,31,14∗I,12,22+22∗I,0,42+24∗I,31∗I,0,33+33∗I,43,14,24+42∗I,43∗I,I}

10.126 check_mul_small.c File Reference

Macros

- #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed ================\n", #X); /∗ abort(); ∗/ } else {fprintf(stderr, "test %35s passed.\n",#X); }

Functions

- int check_mul (mess_matrix A, mess_operation_t opA, mess_matrix B, mess_operation_t opB, mess_matrix C, mess_matrix Csoll)
- int main (int argc, char ∗∗argv)

Variables

- double matrix_B [4 ∗2] = { 11, 21, 0, 41, 0, 22, 32, 42}
- double matrix_A [2 ∗4] = { 11, 21,0,22,0,23,14,24}
- double matrix_ANBN [4] = { 695, 1677, 588, 2228 }
- double matrix_ANAT [4] = { 317, 567, 567, 2030 }
- double matrix_ANAH [4] = { 317, 567, 567, 2030 }
- double matrix_BTBN [4] = { 2243, 2184, 2184, 3272 }
- double matrix_BHBN [4] = { 2243, 2184, 2184, 3272 }
- double matrix_BTAT [4] = { 695, 588, 1677, 2228 }
- double matrix_BHAT [4] = { 695, 588, 1677, 2228 }
- double matrix_BTAH [4] = { 695, 588, 1677, 2228 }
- double matrix_BHAH [4] = { 695, 588, 1677, 2228 }
- mess_double_cpx_t matrix_Bi [8] = { 11 + 11∗I, 21 + 0∗I, 0 + 0∗I, 41 + 14∗I, 0 + 21∗I, 22 + 22∗I, 32 + 23∗I, 42 + 24∗I }
- mess_double_cpx_t matrix_Ai [8] = { 11 + -22∗I, 21 + -0∗I, 0 + -42∗I, 22 + -44∗I, 0 + -0∗I, 23 + -64∗I, 14 + -82∗I, 24 + -84∗I }
- mess_double_cpx_t matrix_AiNBN [4] = { 695 + -4486∗I, 1677 + -4368∗I, 588 + -4368∗I, 2228 + -6544∗I }
- mess_double_cpx_t matrix_AiNAT [4] = { 317 + -1390∗I, 567 + -1176∗I, 567 + -3354∗I, 2030 + -4456∗I }
- mess_double_cpx_t matrix_AiNAH [4] = { 317 + -1390∗I, 567 + -1176∗I, 567 + -3354∗I, 2030 + -4456∗I }
- mess_double_cpx_t matrix_BiTBN [4] = { 2243 + 695∗I, 2184 + 1677∗I, 2184 + 588∗I, 3272 + 2228∗I }
- mess_double_cpx_t matrix_BiTAT [4] = { 695 + 317∗I, 588 + 567∗I, 1677 + 567∗I, 2228 + 2030∗I }
- mess_double_cpx_t matrix_BiTAH [4] = { 695 + 317∗I, 588 + 567∗I, 1677 + 567∗I, 2228 + 2030∗I }
- mess_double_cpx_t matrix_BiHBN [4] = { 2243 + -695∗I, 2184 + -1677∗I, 2184 + -588∗I, 3272 + -2228∗I }
- mess_double_cpx_t matrix_BiHAT [4] = { 695 + -317∗I, 588 + -567∗I, 1677 + -567∗I, 2228 + -2030∗I }

- mess_double_cpx_t matrix_BiHAH [4] = { 695 + -317∗I, 588 + -567∗I, 1677 + -567∗I, 2228 + -2030∗I }
- mess_double_cpx_t matrix_ANBiN [4] = { 695 + 317∗I, 1677 + 567∗I, 588 + 567∗I, 2228 + 2030∗I }
- mess_double_cpx_t matrix_ANAiT [4] = { 317 + -1390∗I, 567 + -3354∗I, 567 + -1176∗I, 2030 + -4456∗I }
- mess_double_cpx_t matrix_ANAiH [4] = { 317 + 1390∗I, 567 + 3354∗I, 567 + 1176∗I, 2030 + 4456∗I }
- mess_double_cpx_t matrix_BTBiN [4] = { 2243 + 695∗I, 2184 + 588∗I, 2184 + 1677∗I, 3272 + 2228∗I }
- mess_double_cpx_t matrix_BTAiT [4] = { 695 + -4486∗I, 588 + -4368∗I, 1677 + -4368∗I, 2228 + -6544∗I }
- mess_double_cpx_t matrix_BTAiH [4] = { 695 + 4486∗I, 588 + 4368∗I, 1677 + 4368∗I, 2228 + 6544∗I }
- mess_double_cpx_t matrix_BHBiN [4] = { 2243 + 695∗I, 2184 + 588∗I, 2184 + 1677∗I, 3272 + 2228∗I }
- mess_double_cpx_t matrix_BHAiT [4] = { 695 + -4486∗I, 588 + -4368∗I, 1677 + -4368∗I, 2228 + -6544∗I }
- mess_double_cpx_t matrix_BHAiH [4] = { 695 + 4486∗I, 588 + 4368∗I, 1677 + 4368∗I, 2228 + 6544∗I }
- mess_double_cpx_t matrix_AiNBiN [4] = { 2085 + -4169∗I, 2853 + -3801∗I, 3942 + -3801∗I, 6684 + -4514∗I }
- mess_double_cpx_t matrix_AiNAiT [4] = { -8655 + -2780∗I, -8169 + -4530∗I, -8169 + -4530∗I, -11058 + -8912∗I }
- mess_double_cpx_t matrix_AiNAiH [4] = { 9289 + 0∗I, 9303 + 2178∗I, 9303 + -2178∗I, 15118 + 0∗I }
- mess_double_cpx_t matrix_BiTBiN [4] = { 1926 + 1390∗I, 1617 + 2265∗I, 1617 + 2265∗I, 1242 + 4456∗I }
- mess_double_cpx_t matrix_BiTAiT [4] = { 2085 + -4169∗I, 3942 + -3801∗I, 2853 + -3801∗I, 6684 + -4514∗I }
- mess_double_cpx_t matrix_BiTAiH [4] = { -695 + 4803∗I, -2766 + 4935∗I, 501 + 4935∗I, -2228 + 8574∗I }
- mess_double_cpx_t matrix_BiHBiN [4] = { 2560 + 0∗I, 2751 + -1089∗I, 2751 + 1089∗I, 5302 + 0∗I }
- mess_double_cpx_t matrix_BiHAiT [4] = { -695 + -4803∗I, -2766 + -4935∗I, 501 + -4935∗I, -2228 + -8574∗I }
- mess_double_cpx_t matrix_BiHAiH [4] = { 2085 + 4169∗I, 3942 + 3801∗I, 2853 + 3801∗I, 6684 + 4514∗I }

10.126.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

- MESS_OP_NONE ($op(X) = X$),
- MESS_OP_TRANSPOSE ($op(X) = X^T$),
- MESS_OP_HERMITIAN ($op(X) = X^H$).

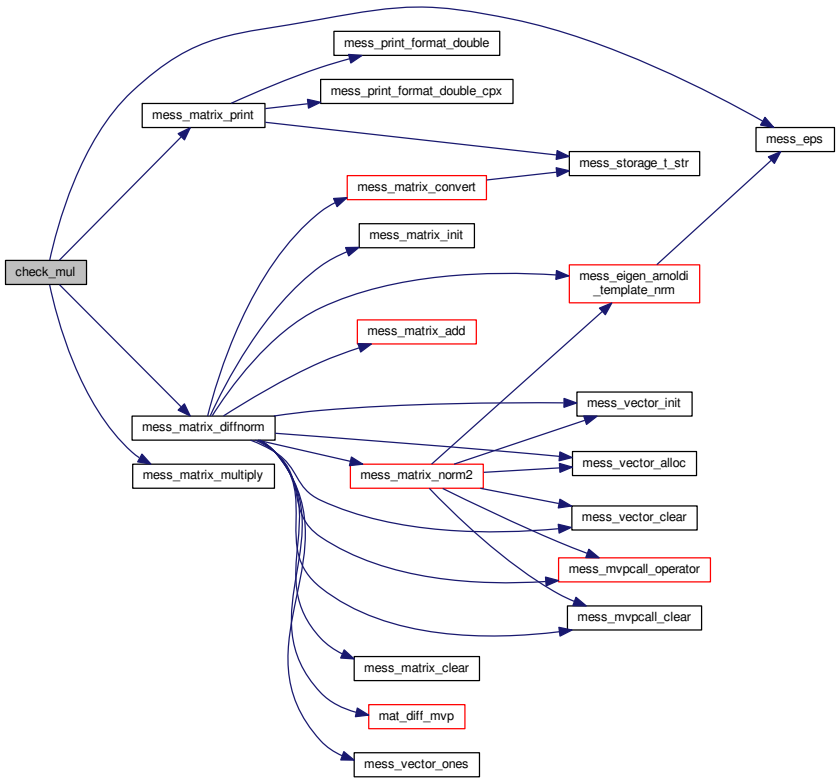
10.126.2 Macro Definition Documentation

10.126.2.1 #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed ================\n", #X); /∗ abort(); ∗/ } else {fprintf(stderr, "test %35s passed.\n",#X); }

10.126.3 Function Documentation

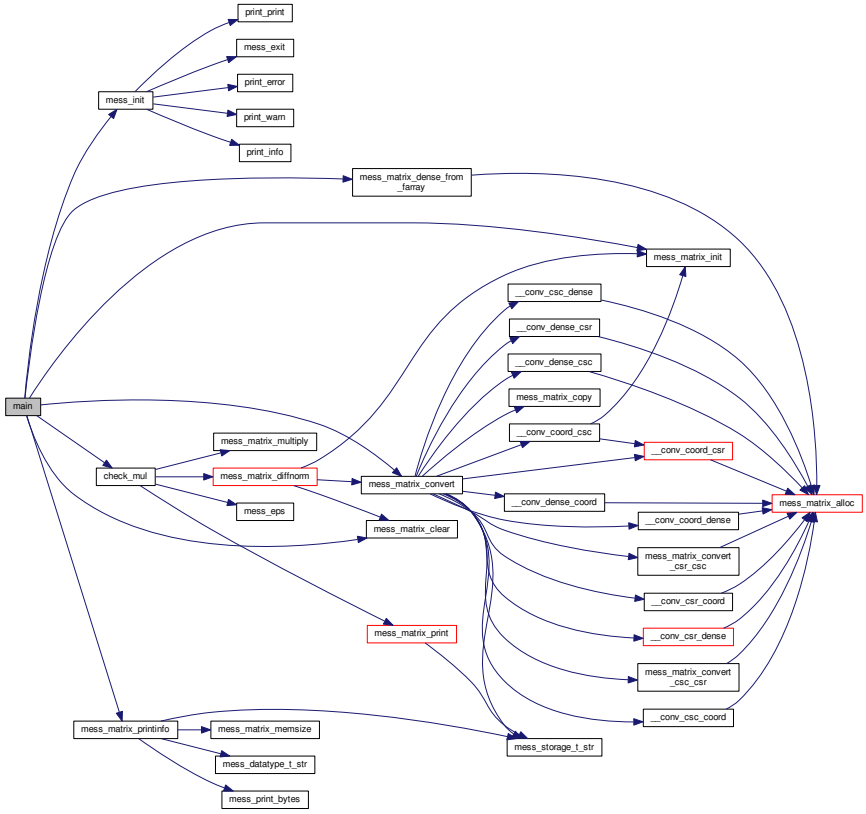
10.126.3.1 int check_mul (mess_matrix *A*, mess_operation_t *opA*, mess_matrix *B*, mess_operation_t *opB*, mess_matrix *C*, mess_matrix *Csoll*)

Here is the call graph for this function:



10.126.3.2 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.126.4 Variable Documentation

10.126.4.1 double matrix_A[2 ∗4] = { 11, 21,0,22,0,23,14,24}

10.126.4.2 mess_double_cpx_t matrix_Ai[8] = { 11 + -22∗I, 21 + -0∗I, 0 + -42∗I, 22 + -44∗I, 0 + -0∗I, 23 + -64∗I, 14 + -82∗I, 24 + -84∗I }

10.126.4.3 mess_double_cpx_t matrix_AiNAH[4] = { 317 + -1390∗I, 567 + -1176∗I, 567 + -3354∗I, 2030 + -4456∗I }

10.126.4.4 mess_double_cpx_t matrix_AiNAiH[4] = { 9289 + 0∗I, 9303 + 2178∗I, 9303 + -2178∗I, 15118 + 0∗I }

10.126.4.5 mess_double_cpx_t matrix_AiNAiT[4] = { -8655 + -2780∗I, -8169 + -4530∗I, -8169 + -4530∗I, -11058 + -8912∗I }

10.126.4.6 mess_double_cpx_t matrix_AiNAT[4] = { 317 + -1390∗I, 567 + -1176∗I, 567 + -3354∗I, 2030 + -4456∗I }

10.126.4.7 mess_double_cpx_t matrix_AiNBiN[4] = { 2085 + -4169∗I, 2853 + -3801∗I, 3942 + -3801∗I, 6684 + -4514∗I }

10.126.4.8 mess_double_cpx_t matrix_AiNBN[4] = { 695 + -4486∗I, 1677 + -4368∗I, 588 + -4368∗I, 2228 + -6544∗I }

10.126.4.9 double matrix_ANAH[4] = { 317, 567, 567, 2030 }

10.126.4.10 mess_double_cpx_t matrix_ANAiH[4] = { 317 + 1390∗I, 567 + 3354∗I, 567 + 1176∗I, 2030 + 4456∗I }

10.126.4.11 mess_double_cpx_t matrix_ANAiT[4] = { 317 + -1390∗I, 567 + -3354∗I, 567 + -1176∗I, 2030 + -4456∗I }

10.126.4.12 double matrix_ANAT[4] = { 317, 567, 567, 2030 }

10.126.4.13 mess_double_cpx_t matrix_ANBiN[4] = { 695 + 317∗I, 1677 + 567∗I, 588 + 567∗I, 2228 + 2030∗I }

10.126.4.14 double matrix_ANBN[4] = { 695, 1677, 588, 2228 }

10.126.4.15 double matrix_B[4 ∗2] = { 11, 21, 0, 41, 0, 22, 32, 42}

10.126.4.16 double matrix_BHAH[4] = { 695, 588, 1677, 2228 }

10.126.4.17 mess_double_cpx_t matrix_BHAiH[4] = { 695 + 4486∗I, 588 + 4368∗I, 1677 + 4368∗I, 2228 + 6544∗I }

10.126.4.18 mess_double_cpx_t matrix_BHAiT[4] = { 695 + -4486∗I, 588 + -4368∗I, 1677 + -4368∗I, 2228 + -6544∗I }

10.126.4.19 double matrix_BHAT[4] = { 695, 588, 1677, 2228 }

10.126.4.20 mess_double_cpx_t matrix_BHBiN[4] = { 2243 + 695∗I, 2184 + 588∗I, 2184 + 1677∗I, 3272 + 2228∗I }

10.126.4.21 double matrix_BHBN[4] = { 2243, 2184, 2184, 3272 }

10.126.4.22 mess_double_cpx_t matrix_Bi[8] = { 11 + 11∗I, 21 + 0∗I, 0 + 0∗I, 41 + 14∗I, 0 + 21∗I, 22 + 22∗I, 32 + 23∗I, 42 + 24∗I }

10.126.4.23 mess_double_cpx_t matrix_BiHAH[4] = { 695 + -317∗I, 588 + -567∗I, 1677 + -567∗I, 2228 + -2030∗I }

10.126.4.24 mess_double_cpx_t matrix_BiHAiH[4] = { 2085 + 4169∗I, 3942 + 3801∗I, 2853 + 3801∗I, 6684 + 4514∗I }

10.126.4.25 mess_double_cpx_t matrix_BiHAiT[4] = { -695 + -4803∗I, -2766 + -4935∗I, 501 + -4935∗I, -2228 + -8574∗I }

10.126.4.26 mess_double_cpx_t matrix_BiHAT[4] = { 695 + -317∗I, 588 + -567∗I, 1677 + -567∗I, 2228 + -2030∗I }

10.126.4.27 mess_double_cpx_t matrix_BiHBiN[4] = { 2560 + 0∗I, 2751 + -1089∗I, 2751 + 1089∗I, 5302 + 0∗I }

10.126.4.28 mess_double_cpx_t matrix_BiHBN[4] = { 2243 + -695∗I, 2184 + -1677∗I, 2184 + -588∗I, 3272 + -2228∗I }

10.126.4.29 mess_double_cpx_t matrix_BiTAH[4] = { 695 + 317∗I, 588 + 567∗I, 1677 + 567∗I, 2228 + 2030∗I }

10.126.4.30 mess_double_cpx_t matrix_BiTAiH[4] = { -695 + 4803∗I, -2766 + 4935∗I, 501 + 4935∗I, -2228 + 8574∗I }

10.126.4.31 mess_double_cpx_t matrix_BiTAiT[4] = { 2085 + -4169∗I, 3942 + -3801∗I, 2853 + -3801∗I, 6684 + -4514∗I }

10.126.4.32 mess_double_cpx_t matrix_BiTAT[4] = { 695 + 317∗I, 588 + 567∗I, 1677 + 567∗I, 2228 + 2030∗I }

10.126.4.33 mess_double_cpx_t matrix_BiTBiN[4] = { 1926 + 1390∗I, 1617 + 2265∗I, 1617 + 2265∗I, 1242 + 4456∗I }

10.126.4.34 mess_double_cpx_t matrix_BiTBN[4] = { 2243 + 695∗I, 2184 + 1677∗I, 2184 + 588∗I, 3272 + 2228∗I }

10.126.4.35 double matrix_BTAH[4] = { 695, 588, 1677, 2228 }

10.126.4.36 mess_double_cpx_t matrix_BTAiH[4] = { 695 + 4486∗I, 588 + 4368∗I, 1677 + 4368∗I, 2228 + 6544∗I }

10.126.4.37 mess_double_cpx_t matrix_BTAiT[4] = { 695 + -4486∗I, 588 + -4368∗I, 1677 + -4368∗I, 2228 + -6544∗I }

10.126.4.38 double matrix_BTAT[4] = { 695, 588, 1677, 2228 }

10.126.4.39 mess_double_cpx_t matrix_BTBiN[4] = { 2243 + 695∗I, 2184 + 588∗I, 2184 + 1677∗I, 3272 + 2228∗I }

10.126.4.40 double matrix_BTBN[4] = { 2243, 2184, 2184, 3272 }

10.127 check_mul_tall.c File Reference

Macros

- #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed ================\n", #X); abort(); } else {fprintf(stderr, "test %35s passed.\n",#X); }

Functions

- int check_mul (mess_matrix A, mess_operation_t opA, mess_matrix B, mess_operation_t opB, mess_matrix C, mess_matrix Csoll)
- int main (int argc, char ∗∗argv)

Variables

- double matrix_A [4 ∗2] = { 11, 21, 0, 41, 0, 22, 32, 42}
- double matrix_B [2 ∗4] = { 11, 21,0,22,0,23,14,24}
- double matrix_ANBN [16] = { 121, 693, 672, 1333, 0, 484, 704, 924, 0, 506, 736, 966, 154, 822, 768, 1582 }
- double matrix_ANAT [16] = { 121, 231, 0, 451, 231, 925, 704, 1785, 0, 704, 1024, 1344, 451, 1785, 1344, 3445 }
- double matrix_BTBN [16] = { 562, 462, 483, 658, 462, 484, 506, 528, 483, 506, 529, 552, 658, 528, 552, 772 }
- double matrix_BTAT [16] = { 121, 0, 0, 154, 693, 484, 506, 822, 672, 704, 736, 768, 1333, 924, 966, 1582 }
- mess_double_cpx_t matrix_Ai [8] = { 11 + 11∗I, 21 + 0∗I, 0 + 0∗I, 41 + 14∗I, 0 + 21∗I, 22 + 22∗I, 32 + 23∗I, 42 + 24∗I }
- mess_double_cpx_t matrix_Bi [8] = { 11 + -22∗I, 21 + -0∗I, 0 + -42∗I, 22 + -44∗I, 0 + -0∗I, 23 + -64∗I, 14 + -82∗I, 24 + -84∗I }
- mess_double_cpx_t matrix_AiNBN [16] = { 121 + 562∗I, 693 + 462∗I, 672 + 483∗I, 1333 + 658∗I, 0 + 462∗I, 484 + 484∗I, 704 + 506∗I, 924 + 528∗I, 0 + 483∗I, 506 + 506∗I, 736 + 529∗I, 966 + 552∗I, 154 + 658∗I, 822 + 528∗I, 768 + 552∗I, 1582 + 772∗I }
- mess_double_cpx_t matrix_AiNAT [16] = { 121 + 121∗I, 231 + 0∗I, 0 + 0∗I, 451 + 154∗I, 231 + 693∗I, 925 + 484∗I, 704 + 506∗I, 1785 + 822∗I, 0 + 672∗I, 704 + 704∗I, 1024 + 736∗I, 1344 + 768∗I, 451 + 1333∗I, 1785 + 924∗I, 1344 + 966∗I, 3445 + 1582∗I }
- mess_double_cpx_t matrix_AiNAH [16] = { 121 + 121∗I, 231 + 0∗I, 0 + 0∗I, 451 + 154∗I, 231 + 693∗I, 925 + 484∗I, 704 + 506∗I, 1785 + 822∗I, 0 + 672∗I, 704 + 704∗I, 1024 + 736∗I, 1344 + 768∗I, 451 + 1333∗I, 1785 + 924∗I, 1344 + 966∗I, 3445 + 1582∗I }

- mess_double_cpx_t matrix_BiTBN [16] = { 562 + -242∗I, 462 + -1386∗I, 483 + -1344∗I, 658 + -2666∗I, 462 + 0∗I, 484 + -968∗I, 506 + -1408∗I, 528 + -1848∗I, 483 + 0∗I, 506 + -1012∗I, 529 + -1472∗I, 552 + -1932∗I, 658 + -308∗I, 528 + -1644∗I, 552 + -1536∗I, 772 + -3164∗I }
- mess_double_cpx_t matrix_BiTAT [16] = { 121 + -242∗I, 0 + -462∗I, 0 + 0∗I, 154 + -902∗I, 693 + -462∗I, 484 + -1850∗I, 506 + -1408∗I, 822 + -3570∗I, 672 + 0∗I, 704 + -1408∗I, 736 + -2048∗I, 768 + -2688∗I, 1333 + -902∗I, 924 + -3570∗I, 966 + -2688∗I, 1582 + -6890∗I }
- mess_double_cpx_t matrix_BiTAH [16] = { 121 + -242∗I, 0 + -462∗I, 0 + 0∗I, 154 + -902∗I, 693 + -462∗I, 484 + -1850∗I, 506 + -1408∗I, 822 + -3570∗I, 672 + 0∗I, 704 + -1408∗I, 736 + -2048∗I, 768 + -2688∗I, 1333 + -902∗I, 924 + -3570∗I, 966 + -2688∗I, 1582 + -6890∗I }
- mess_double_cpx_t matrix_BiHBN [16] = { 562 + 242∗I, 462 + 1386∗I, 483 + 1344∗I, 658 + 2666∗I, 462 + 0∗I, 484 + 968∗I, 506 + 1408∗I, 528 + 1848∗I, 483 + 0∗I, 506 + 1012∗I, 529 + 1472∗I, 552 + 1932∗I, 658 + 308∗I, 528 + 1644∗I, 552 + 1536∗I, 772 + 3164∗I }
- mess_double_cpx_t matrix_BiHAT [16] = { 121 + 242∗I, 0 + 462∗I, 0 + 0∗I, 154 + 902∗I, 693 + 462∗I, 484 + 1850∗I, 506 + 1408∗I, 822 + 3570∗I, 672 + 0∗I, 704 + 1408∗I, 736 + 2048∗I, 768 + 2688∗I, 1333 + 902∗I, 924 + 3570∗I, 966 + 2688∗I, 1582 + 6890∗I }
- mess_double_cpx_t matrix_BiHAH [16] = { 121 + 242∗I, 0 + 462∗I, 0 + 0∗I, 154 + 902∗I, 693 + 462∗I, 484 + 1850∗I, 506 + 1408∗I, 822 + 3570∗I, 672 + 0∗I, 704 + 1408∗I, 736 + 2048∗I, 768 + 2688∗I, 1333 + 902∗I, 924 + 3570∗I, 966 + 2688∗I, 1582 + 6890∗I }
- mess_double_cpx_t matrix_ANBiN [16] = { 121 + -242∗I, 693 + -462∗I, 672 + 0∗I, 1333 + -902∗I, 0 + -462∗I, 484 + -1850∗I, 704 + -1408∗I, 924 + -3570∗I, 0 + 0∗I, 506 + -1408∗I, 736 + -2048∗I, 966 + -2688∗I, 154 + -902∗I, 822 + -3570∗I, 768 + -2688∗I, 1582 + -6890∗I }
- mess_double_cpx_t matrix_ANAiT [16] = { 121 + 121∗I, 231 + 693∗I, 0 + 672∗I, 451 + 1333∗I, 231 + 0∗I, 925 + 484∗I, 704 + 704∗I, 1785 + 924∗I, 0 + 0∗I, 704 + 506∗I, 1024 + 736∗I, 1344 + 966∗I, 451 + 154∗I, 1785 + 822∗I, 1344 + 768∗I, 3445 + 1582∗I }
- mess_double_cpx_t matrix_ANAiH [16] = { 121 + -121∗I, 231 + -693∗I, 0 + -672∗I, 451 + -1333∗I, 231 + 0∗I, 925 + -484∗I, 704 + -704∗I, 1785 + -924∗I, 0 + 0∗I, 704 + -506∗I, 1024 + -736∗I, 1344 + -966∗I, 451 + -154∗I, 1785 + -822∗I, 1344 + -768∗I, 3445 + -1582∗I }
- mess_double_cpx_t matrix_BTBiN [16] = { 562 + -242∗I, 462 + 0∗I, 483 + 0∗I, 658 + -308∗I, 462 + -1386∗I, 484 + -968∗I, 506 + -1012∗I, 528 + -1644∗I, 483 + -1344∗I, 506 + -1408∗I, 529 + -1472∗I, 552 + -1536∗I, 658 + -2666∗I, 528 + -1848∗I, 552 + -1932∗I, 772 + -3164∗I }
- mess_double_cpx_t matrix_BTAiT [16] = { 121 + 562∗I, 0 + 462∗I, 0 + 483∗I, 154 + 658∗I, 693 + 462∗I, 484 + 484∗I, 506 + 506∗I, 822 + 528∗I, 672 + 483∗I, 704 + 506∗I, 736 + 529∗I, 768 + 552∗I, 1333 + 658∗I, 924 + 528∗I, 966 + 552∗I, 1582 + 772∗I }
- mess_double_cpx_t matrix_BTAiH [16] = { 121 + -562∗I, 0 + -462∗I, 0 + -483∗I, 154 + -658∗I, 693 + -462∗I, 484 + -484∗I, 506 + -506∗I, 822 + -528∗I, 672 + -483∗I, 704 + -506∗I, 736 + -529∗I, 768 + -552∗I, 1333 + -658∗I, 924 + -528∗I, 966 + -552∗I, 1582 + -772∗I }
- mess_double_cpx_t matrix_BHBiN [16] = { 562 + -242∗I, 462 + 0∗I, 483 + 0∗I, 658 + -308∗I, 462 + -1386∗I, 484 + -968∗I, 506 + -1012∗I, 528 + -1644∗I, 483 + -1344∗I, 506 + -1408∗I, 529 + -1472∗I, 552 + -1536∗I, 658 + -2666∗I, 528 + -1848∗I, 552 + -1932∗I, 772 + -3164∗I }
- mess_double_cpx_t matrix_BHAiT [16] = { 121 + 562∗I, 0 + 462∗I, 0 + 483∗I, 154 + 658∗I, 693 + 462∗I, 484 + 484∗I, 506 + 506∗I, 822 + 528∗I, 672 + 483∗I, 704 + 506∗I, 736 + 529∗I, 768 + 552∗I, 1333 + 658∗I, 924 + 528∗I, 966 + 552∗I, 1582 + 772∗I }
- mess_double_cpx_t matrix_BHAiH [16] = { 121 + -562∗I, 0 + -462∗I, 0 + -483∗I, 154 + -658∗I, 693 + -462∗I, 484 + -484∗I, 506 + -506∗I, 822 + -528∗I, 672 + -483∗I, 704 + -506∗I, 736 + -529∗I, 768 + -552∗I, 1333 + -658∗I, 924 + -528∗I, 966 + -552∗I, 1582 + -772∗I }
- mess_double_cpx_t matrix_AiNBiN [16] = { 363 + 320∗I, 693 + 0∗I, 672 + 483∗I, 1641 + -244∗I, 1386 + 0∗I, 1452 + -1366∗I, 1716 + -902∗I, 2568 + -3042∗I, 1344 + 483∗I, 1914 + -902∗I, 2208 + -1519∗I, 2502 + -2136∗I, 2820 + -244∗I, 2670 + -3042∗I, 2700 + -2136∗I, 4746 + -6118∗I }
- mess_double_cpx_t matrix_AiNAiT [16] = { -441 + 242∗I, -231 + 693∗I, -483 + 672∗I, -207 + 1487∗I, -231 + 693∗I, 441 + 968∗I, 198 + 1210∗I, 1257 + 1746∗I, -483 + 672∗I, 198 + 1210∗I, 495 + 1472∗I, 792 + 1734∗I, -207 + 1487∗I, 1257 + 1746∗I, 792 + 1734∗I, 2673 + 3164∗I }
- mess_double_cpx_t matrix_AiNAiH [16] = { 683 + 0∗I, 693 + -693∗I, 483 + -672∗I, 1109 + -1179∗I, 693 + 693∗I, 1409 + 0∗I, 1210 + -198∗I, 2313 + -102∗I, 483 + 672∗I, 1210 + 198∗I, 1553 + 0∗I, 1896 + -198∗I, 1109 + 1179∗I, 2313 + 102∗I, 1896 + 198∗I, 4217 + 0∗I }
- mess_double_cpx_t matrix_BiTBiN [16] = { 78 + -484∗I, -462 + -1386∗I, 483 + -1344∗I, -1146 + -2974∗I, -462 + -1386∗I, -3216 + -1936∗I, -2310 + -2420∗I, -6612 + -3492∗I, 483 + -1344∗I, -2310 + -2420∗I, -3567 + -2944∗I, -4824 + -3468∗I, -1146 + -2974∗I, -6612 + -3492∗I, -4824 + -3468∗I, -13008 + -6328∗I }

- mess_double_cpx_t matrix_BiTAiT [16] = { 363 + 320∗I, 1386 + 0∗I, 1344 + 483∗I, 2820 + -244∗I, 693 + 0∗I, 1452 + -1366∗I, 1914 + -902∗I, 2670 + -3042∗I, 672 + 483∗I, 1716 + -902∗I, 2208 + -1519∗I, 2700 + -2136∗I, 1641 + -244∗I, 2568 + -3042∗I, 2502 + -2136∗I, 4746 + -6118∗I }
- mess_double_cpx_t matrix_BiTAiH [16] = { -121 + -804∗I, -1386 + -924∗I, -1344 + -483∗I, -2512 + -1560∗I, 693 + -924∗I, -484 + -2334∗I, -902 + -1914∗I, -1026 + -4098∗I, 672 + -483∗I, -308 + -1914∗I, -736 + -2577∗I, -1164 + -3240∗I, 1025 + -1560∗I, -720 + -4098∗I, -570 + -3240∗I, -1582 + -7662∗I }
- mess_double_cpx_t matrix_BiHBiN [16] = { 1046 + 0∗I, 1386 + 1386∗I, 483 + 1344∗I, 2462 + 2358∗I, 1386 + -1386∗I, 4184 + 0∗I, 3322 + 396∗I, 7668 + 204∗I, 483 + -1344∗I, 3322 + -396∗I, 4625 + 0∗I, 5928 + 396∗I, 2462 + -2358∗I, 7668 + -204∗I, 5928 + -396∗I, 14552 + 0∗I }
- mess_double_cpx_t matrix_BiHAiT [16] = { -121 + 804∗I, -1386 + 924∗I, -1344 + 483∗I, -2512 + 1560∗I, 693 + 924∗I, -484 + 2334∗I, -902 + 1914∗I, -1026 + 4098∗I, 672 + 483∗I, -308 + 1914∗I, -736 + 2577∗I, -1164 + 3240∗I, 1025 + 1560∗I, -720 + 4098∗I, -570 + 3240∗I, -1582 + 7662∗I }
- mess_double_cpx_t matrix_BiHAiH [16] = { 363 + -320∗I, 1386 + 0∗I, 1344 + -483∗I, 2820 + 244∗I, 693 + 0∗I, 1452 + 1366∗I, 1914 + 902∗I, 2670 + 3042∗I, 672 + -483∗I, 1716 + 902∗I, 2208 + 1519∗I, 2700 + 2136∗I, 1641 + 244∗I, 2568 + 3042∗I, 2502 + 2136∗I, 4746 + 6118∗I }

10.127.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

- MESS_OP_NONE ($op(X) = X$),
- MESS_OP_TRANSPOSE ($op(X) = X^T$),
- MESS_OP_HERMITIAN ($op(X) = X^H$).

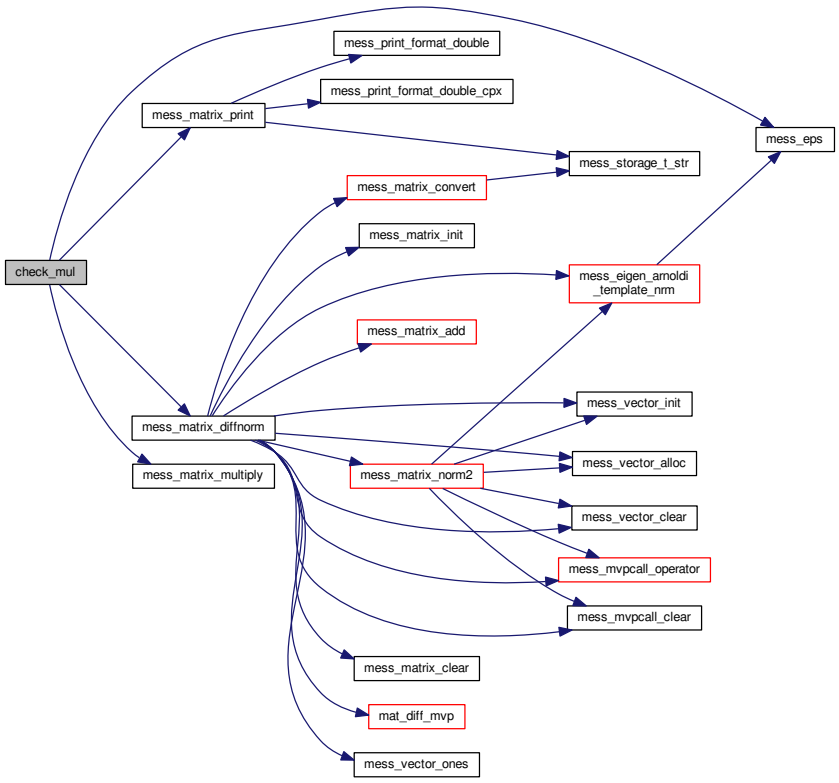
10.127.2 Macro Definition Documentation

10.127.2.1 #define CALL2(X) ret = (X); tests++; if (ret == 1) { err ++; fprintf(stderr,"================ test %35s failed ================\n", #X); abort(); } else {fprintf(stderr, "test %35s passed.\n",#X); }

10.127.3 Function Documentation

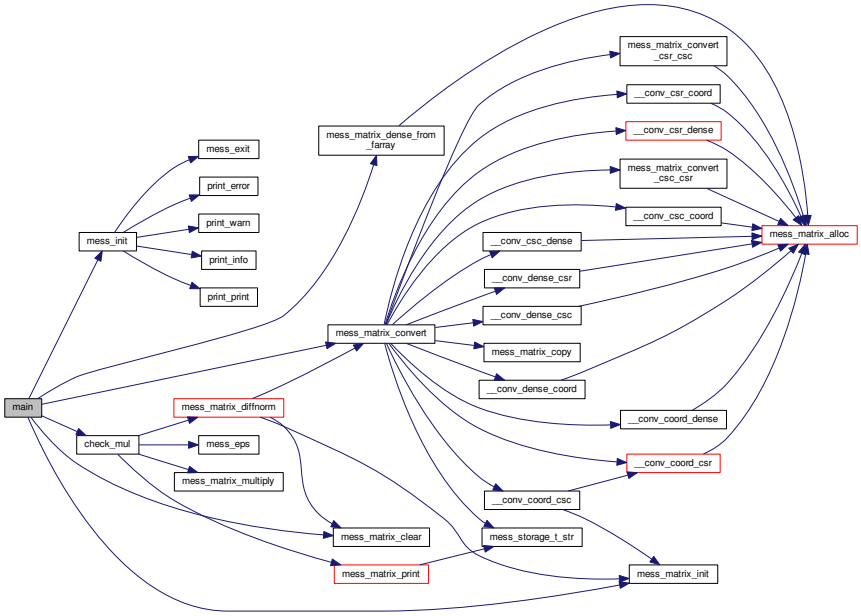
10.127.3.1 int check_mul (mess_matrix *A*, mess_operation_t *opA*, mess_matrix *B*, mess_operation_t *opB*, mess_matrix *C*, mess_matrix *Csoll*)

Here is the call graph for this function:



10.127.3.2 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.127.4 Variable Documentation

10.127.4.1 double matrix_A[4 ∗2] = { 11, 21, 0, 41, 0, 22, 32, 42}

10.127.4.2 mess_double_cpx_t matrix_Ai[8] = { 11 + 11∗I, 21 + 0∗I, 0 + 0∗I, 41 + 14∗I, 0 + 21∗I, 22 + 22∗I, 32 + 23∗I, 42 + 24∗I }

10.127.4.3 mess_double_cpx_t matrix_AiNAH[16] = { 121 + 121∗I, 231 + 0∗I, 0 + 0∗I, 451 + 154∗I, 231 + 693∗I, 925 + 484∗I, 704 + 506∗I, 1785 + 822∗I, 0 + 672∗I, 704 + 704∗I, 1024 + 736∗I, 1344 + 768∗I, 451 + 1333∗I, 1785 + 924∗I, 1344 + 966∗I, 3445 + 1582∗I }

10.127.4.4 mess_double_cpx_t matrix_AiNAiH[16] = { 683 + 0∗I, 693 + -693∗I, 483 + -672∗I, 1109 + -1179∗I, 693 + 693∗I, 1409 + 0∗I, 1210 + -198∗I, 2313 + -102∗I, 483 + 672∗I, 1210 + 198∗I, 1553 + 0∗I, 1896 + -198∗I, 1109 + 1179∗I, 2313 + 102∗I, 1896 + 198∗I, 4217 + 0∗I }

10.127.4.5 mess_double_cpx_t matrix_AiNAiT[16] = { -441 + 242∗I, -231 + 693∗I, -483 + 672∗I, -207 + 1487∗I, -231 + 693∗I, 441 + 968∗I, 198 + 1210∗I, 1257 + 1746∗I, -483 + 672∗I, 198 + 1210∗I, 495 + 1472∗I, 792 + 1734∗I, -207 + 1487∗I, 1257 + 1746∗I, 792 + 1734∗I, 2673 + 3164∗I }

10.127.4.6 mess_double_cpx_t matrix_AiNAT[16] = { 121 + 121∗I, 231 + 0∗I, 0 + 0∗I, 451 + 154∗I, 231 + 693∗I, 925 + 484∗I, 704 + 506∗I, 1785 + 822∗I, 0 + 672∗I, 704 + 704∗I, 1024 + 736∗I, 1344 + 768∗I, 451 + 1333∗I, 1785 + 924∗I, 1344 + 966∗I, 3445 + 1582∗I }

10.127.4.7 mess_double_cpx_t matrix_AiNBiN[16] = { 363 + 320∗I, 693 + 0∗I, 672 + 483∗I, 1641 + -244∗I, 1386 + 0∗I, 1452 + -1366∗I, 1716 + -902∗I, 2568 + -3042∗I, 1344 + 483∗I, 1914 + -902∗I, 2208 + -1519∗I, 2502 + -2136∗I, 2820 + -244∗I, 2670 + -3042∗I, 2700 + -2136∗I, 4746 + -6118∗I }

10.127.4.8 **mess_double_cpx_t** matrix_AiNBN[16] = { 121 + 562∗I, 693 + 462∗I, 672 + 483∗I, 1333 + 658∗I, 0 + 462∗I, 484 + 484∗I, 704 + 506∗I, 924 + 528∗I, 0 + 483∗I, 506 + 506∗I, 736 + 529∗I, 966 + 552∗I, 154 + 658∗I, 822 + 528∗I, 768 + 552∗I, 1582 + 772∗I }

10.127.4.9 **mess_double_cpx_t** matrix_ANAiH[16] = { 121 + -121∗I, 231 + -693∗I, 0 + -672∗I, 451 + -1333∗I, 231 + 0∗I, 925 + -484∗I, 704 + -704∗I, 1785 + -924∗I, 0 + 0∗I, 704 + -506∗I, 1024 + -736∗I, 1344 + -966∗I, 451 + -154∗I, 1785 + -822∗I, 1344 + -768∗I, 3445 + -1582∗I }

10.127.4.10 **mess_double_cpx_t** matrix_ANAiT[16] = { 121 + 121∗I, 231 + 693∗I, 0 + 672∗I, 451 + 1333∗I, 231 + 0∗I, 925 + 484∗I, 704 + 704∗I, 1785 + 924∗I, 0 + 0∗I, 704 + 506∗I, 1024 + 736∗I, 1344 + 966∗I, 451 + 154∗I, 1785 + 822∗I, 1344 + 768∗I, 3445 + 1582∗I }

10.127.4.11 double matrix_ANAT[16] = { 121, 231, 0, 451, 231, 925, 704, 1785, 0, 704, 1024, 1344, 451, 1785, 1344, 3445 }

10.127.4.12 **mess_double_cpx_t** matrix_ANBiN[16] = { 121 + -242∗I, 693 + -462∗I, 672 + 0∗I, 1333 + -902∗I, 0 + -462∗I, 484 + -1850∗I, 704 + -1408∗I, 924 + -3570∗I, 0 + 0∗I, 506 + -1408∗I, 736 + -2048∗I, 966 + -2688∗I, 154 + -902∗I, 822 + -3570∗I, 768 + -2688∗I, 1582 + -6890∗I }

10.127.4.13 double matrix_ANBN[16] = { 121, 693, 672, 1333, 0, 484, 704, 924, 0, 506, 736, 966, 154, 822, 768, 1582 }

10.127.4.14 double matrix_B[2 ∗4] = { 11, 21,0,22,0,23,14,24}

10.127.4.15 **mess_double_cpx_t** matrix_BHAiH[16] = { 121 + -562∗I, 0 + -462∗I, 0 + -483∗I, 154 + -658∗I, 693 + -462∗I, 484 + -484∗I, 506 + -506∗I, 822 + -528∗I, 672 + -483∗I, 704 + -506∗I, 736 + -529∗I, 768 + -552∗I, 1333 + -658∗I, 924 + -528∗I, 966 + -552∗I, 1582 + -772∗I }

10.127.4.16 **mess_double_cpx_t** matrix_BHAiT[16] = { 121 + 562∗I, 0 + 462∗I, 0 + 483∗I, 154 + 658∗I, 693 + 462∗I, 484 + 484∗I, 506 + 506∗I, 822 + 528∗I, 672 + 483∗I, 704 + 506∗I, 736 + 529∗I, 768 + 552∗I, 1333 + 658∗I, 924 + 528∗I, 966 + 552∗I, 1582 + 772∗I }

10.127.4.17 **mess_double_cpx_t** matrix_BHBiN[16] = { 562 + -242∗I, 462 + 0∗I, 483 + 0∗I, 658 + -308∗I, 462 + -1386∗I, 484 + -968∗I, 506 + -1012∗I, 528 + -1644∗I, 483 + -1344∗I, 506 + -1408∗I, 529 + -1472∗I, 552 + -1536∗I, 658 + -2666∗I, 528 + -1848∗I, 552 + -1932∗I, 772 + -3164∗I }

10.127.4.18 **mess_double_cpx_t** matrix_Bi[8] = { 11 + -22∗I, 21 + -0∗I, 0 + -42∗I, 22 + -44∗I, 0 + -0∗I, 23 + -64∗I, 14 + -82∗I, 24 + -84∗I }

10.127.4.19 **mess_double_cpx_t** matrix_BiHAH[16] = { 121 + 242∗I, 0 + 462∗I, 0 + 0∗I, 154 + 902∗I, 693 + 462∗I, 484 + 1850∗I, 506 + 1408∗I, 822 + 3570∗I, 672 + 0∗I, 704 + 1408∗I, 736 + 2048∗I, 768 + 2688∗I, 1333 + 902∗I, 924 + 3570∗I, 966 + 2688∗I, 1582 + 6890∗I }

10.127.4.20 **mess_double_cpx_t** matrix_BiHAiH[16] = { 363 + -320∗I, 1386 + 0∗I, 1344 + -483∗I, 2820 + 244∗I, 693 + 0∗I, 1452 + 1366∗I, 1914 + 902∗I, 2670 + 3042∗I, 672 + -483∗I, 1716 + 902∗I, 2208 + 1519∗I, 2700 + 2136∗I, 1641 + 244∗I, 2568 + 3042∗I, 2502 + 2136∗I, 4746 + 6118∗I }

10.127.4.21 **mess_double_cpx_t** matrix_BiHAiT[16] = { -121 + 804∗I, -1386 + 924∗I, -1344 + 483∗I, -2512 + 1560∗I, 693 + 924∗I, -484 + 2334∗I, -902 + 1914∗I, -1026 + 4098∗I, 672 + 483∗I, -308 + 1914∗I, -736 + 2577∗I, -1164 + 3240∗I, 1025 + 1560∗I, -720 + 4098∗I, -570 + 3240∗I, -1582 + 7662∗I }

10.127.4.22 **mess_double_cpx_t** matrix_BiHAT[16] = { 121 + 242∗I, 0 + 462∗I, 0 + 0∗I, 154 + 902∗I, 693 + 462∗I, 484 + 1850∗I, 506 + 1408∗I, 822 + 3570∗I, 672 + 0∗I, 704 + 1408∗I, 736 + 2048∗I, 768 + 2688∗I, 1333 + 902∗I, 924 + 3570∗I, 966 + 2688∗I, 1582 + 6890∗I }

10.127.4.23 **mess_double_cpx_t matrix_BiHBiN[16]** = { 1046 + 0∗I, 1386 + 1386∗I, 483 + 1344∗I, 2462 + 2358∗I, 1386 + -1386∗I, 4184 + 0∗I, 3322 + 396∗I, 7668 + 204∗I, 483 + -1344∗I, 3322 + -396∗I, 4625 + 0∗I, 5928 + 396∗I, 2462 + -2358∗I, 7668 + -204∗I, 5928 + -396∗I, 14552 + 0∗I }

10.127.4.24 **mess_double_cpx_t matrix_BiHBN[16]** = { 562 + 242∗I, 462 + 1386∗I, 483 + 1344∗I, 658 + 2666∗I, 462 + 0∗I, 484 + 968∗I, 506 + 1408∗I, 528 + 1848∗I, 483 + 0∗I, 506 + 1012∗I, 529 + 1472∗I, 552 + 1932∗I, 658 + 308∗I, 528 + 1644∗I, 552 + 1536∗I, 772 + 3164∗I }

10.127.4.25 **mess_double_cpx_t matrix_BiTAH[16]** = { 121 + -242∗I, 0 + -462∗I, 0 + 0∗I, 154 + -902∗I, 693 + -462∗I, 484 + -1850∗I, 506 + -1408∗I, 822 + -3570∗I, 672 + 0∗I, 704 + -1408∗I, 736 + -2048∗I, 768 + -2688∗I, 1333 + -902∗I, 924 + -3570∗I, 966 + -2688∗I, 1582 + -6890∗I }

10.127.4.26 **mess_double_cpx_t matrix_BiTAiH[16]** = { -121 + -804∗I, -1386 + -924∗I, -1344 + -483∗I, -2512 + -1560∗I, 693 + -924∗I, -484 + -2334∗I, -902 + -1914∗I, -1026 + -4098∗I, 672 + -483∗I, -308 + -1914∗I, -736 + -2577∗I, -1164 + -3240∗I, 1025 + -1560∗I, -720 + -4098∗I, -570 + -3240∗I, -1582 + -7662∗I }

10.127.4.27 **mess_double_cpx_t matrix_BiTAiT[16]** = { 363 + 320∗I, 1386 + 0∗I, 1344 + 483∗I, 2820 + -244∗I, 693 + 0∗I, 1452 + -1366∗I, 1914 + -902∗I, 2670 + -3042∗I, 672 + 483∗I, 1716 + -902∗I, 2208 + -1519∗I, 2700 + -2136∗I, 1641 + -244∗I, 2568 + -3042∗I, 2502 + -2136∗I, 4746 + -6118∗I }

10.127.4.28 **mess_double_cpx_t matrix_BiTAT[16]** = { 121 + -242∗I, 0 + -462∗I, 0 + 0∗I, 154 + -902∗I, 693 + -462∗I, 484 + -1850∗I, 506 + -1408∗I, 822 + -3570∗I, 672 + 0∗I, 704 + -1408∗I, 736 + -2048∗I, 768 + -2688∗I, 1333 + -902∗I, 924 + -3570∗I, 966 + -2688∗I, 1582 + -6890∗I }

10.127.4.29 **mess_double_cpx_t matrix_BiTBiN[16]** = { 78 + -484∗I, -462 + -1386∗I, 483 + -1344∗I, -1146 + -2974∗I, -462 + -1386∗I, -3216 + -1936∗I, -2310 + -2420∗I, -6612 + -3492∗I, 483 + -1344∗I, -2310 + -2420∗I, -3567 + -2944∗I, -4824 + -3468∗I, -1146 + -2974∗I, -6612 + -3492∗I, -4824 + -3468∗I, -13008 + -6328∗I }

10.127.4.30 **mess_double_cpx_t matrix_BiTBN[16]** = { 562 + -242∗I, 462 + -1386∗I, 483 + -1344∗I, 658 + -2666∗I, 462 + 0∗I, 484 + -968∗I, 506 + -1408∗I, 528 + -1848∗I, 483 + 0∗I, 506 + -1012∗I, 529 + -1472∗I, 552 + -1932∗I, 658 + -308∗I, 528 + -1644∗I, 552 + -1536∗I, 772 + -3164∗I }

10.127.4.31 **mess_double_cpx_t matrix_BTAiH[16]** = { 121 + -562∗I, 0 + -462∗I, 0 + -483∗I, 154 + -658∗I, 693 + -462∗I, 484 + -484∗I, 506 + -506∗I, 822 + -528∗I, 672 + -483∗I, 704 + -506∗I, 736 + -529∗I, 768 + -552∗I, 1333 + -658∗I, 924 + -528∗I, 966 + -552∗I, 1582 + -772∗I }

10.127.4.32 **mess_double_cpx_t matrix_BTAiT[16]** = { 121 + 562∗I, 0 + 462∗I, 0 + 483∗I, 154 + 658∗I, 693 + 462∗I, 484 + 484∗I, 506 + 506∗I, 822 + 528∗I, 672 + 483∗I, 704 + 506∗I, 736 + 529∗I, 768 + 552∗I, 1333 + 658∗I, 924 + 528∗I, 966 + 552∗I, 1582 + 772∗I }

10.127.4.33 **double matrix_BTAT[16]** = { 121, 0, 0, 154, 693, 484, 506, 822, 672, 704, 736, 768, 1333, 924, 966, 1582 }

10.127.4.34 **mess_double_cpx_t matrix_BTBiN[16]** = { 562 + -242∗I, 462 + 0∗I, 483 + 0∗I, 658 + -308∗I, 462 + -1386∗I, 484 + -968∗I, 506 + -1012∗I, 528 + -1644∗I, 483 + -1344∗I, 506 + -1408∗I, 529 + -1472∗I, 552 + -1536∗I, 658 + -2666∗I, 528 + -1848∗I, 552 + -1932∗I, 772 + -3164∗I }

10.127.4.35 **double matrix_BTBN[16]** = { 562, 462, 483, 658, 462, 484, 506, 528, 483, 506, 529, 552, 658, 528, 552, 772 }

10.128 check_mulnormf2.c File Reference

Functions

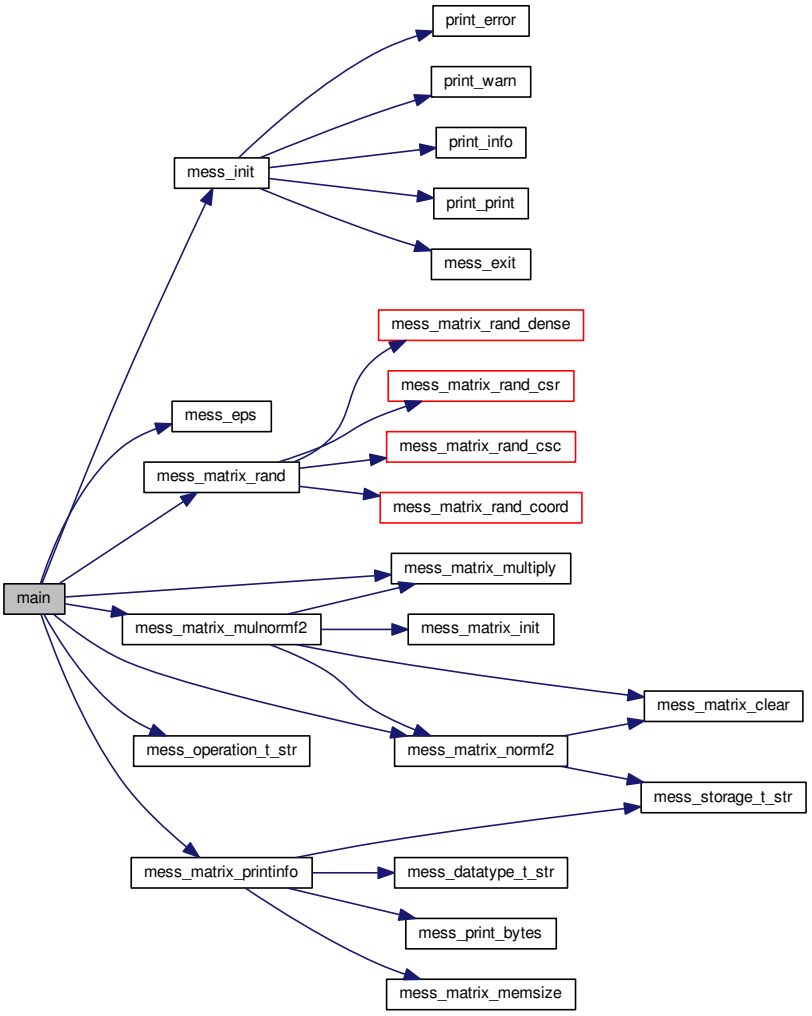
- int main (int argc, char ∗∗argv)

10.128.1 Detailed Description

10.128.2 Function Documentation

10.128.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.129 check_mvp.c File Reference

Functions

- int main (int argc, char ∗argv[ ])

10.129.1 Detailed Description

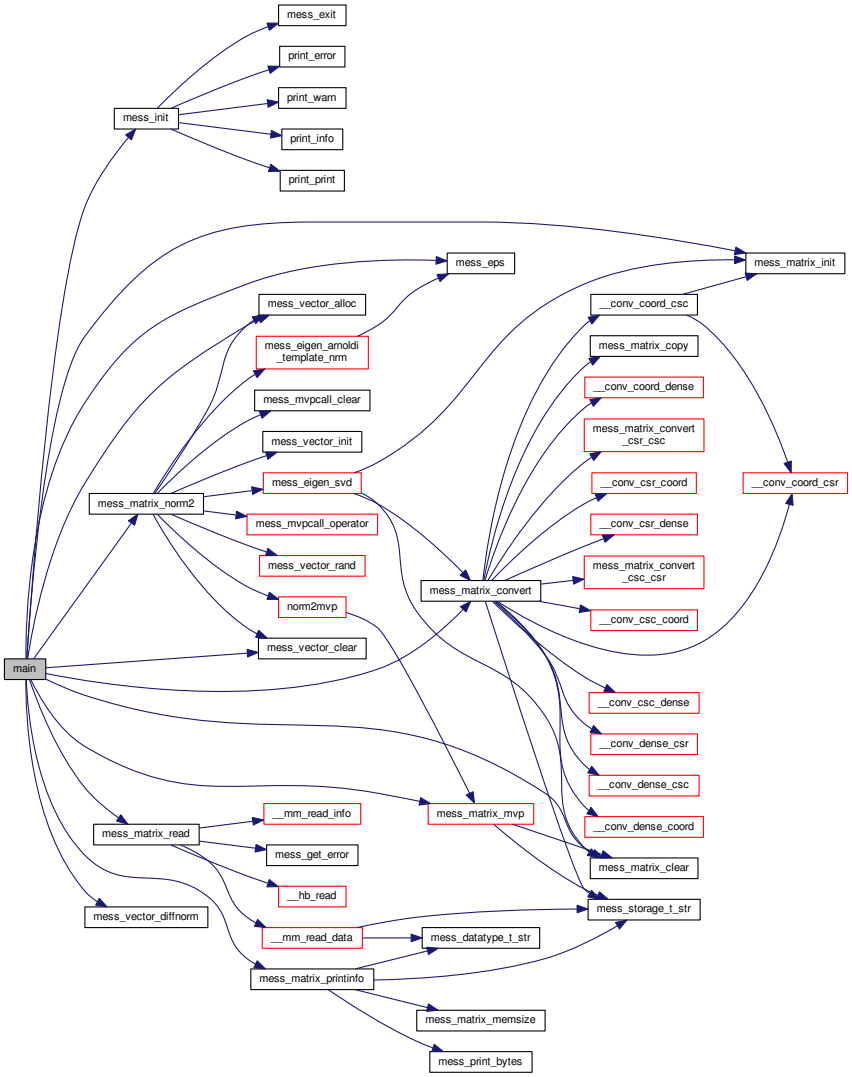
- 0: MESS_OP_NONE ($op(A) = A$),
- 1: MESS_OP_TRANSPOSE ($op(A) = A^T$),
- 2: MESS_OP_HERMITIAN ($op(A) = A^H$).

1 is default value.

10.129.2 Function Documentation

10.129.2.1 int main (int *argc*, char $*$ *argv[ ]*)

Here is the call graph for this function:



10.130 check_mvpcall_matrix.c File Reference

Functions

- int check_mvp_call_matrix (mess_matrix mat, mess_operation_t op1, mess_operation_t op2, mess_vector v1, mess_vector v2)
- int main (int argc, char **argv)

10.130.1 Detailed Description

Author

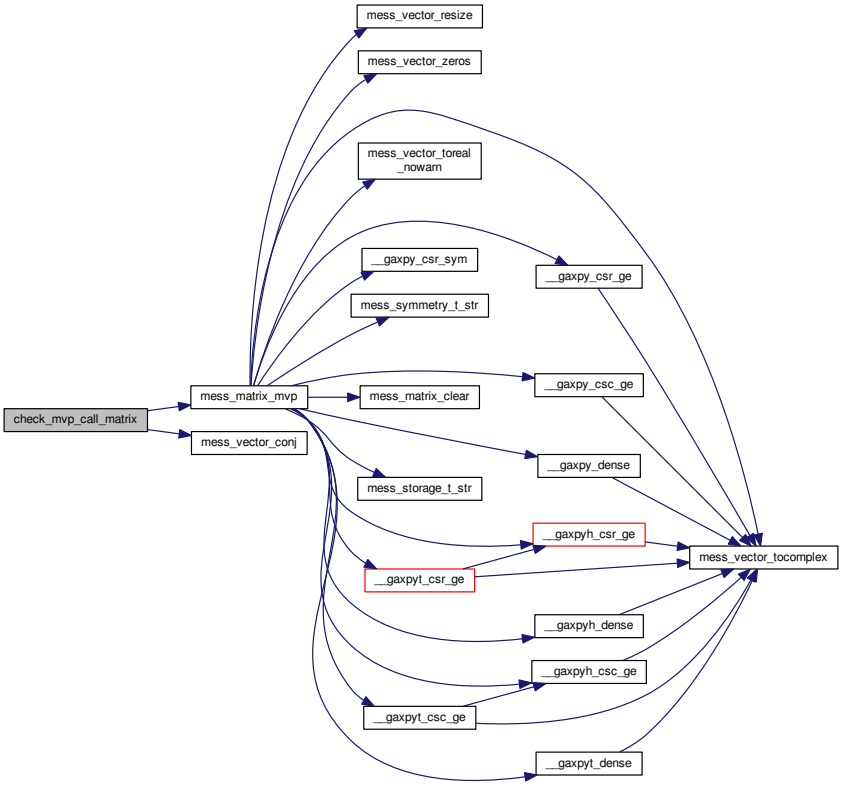
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_mvpcall_matrix function defined in mvpcall.c that means it uses mess_matrix_mvp to compare the results.

10.130.2 Function Documentation

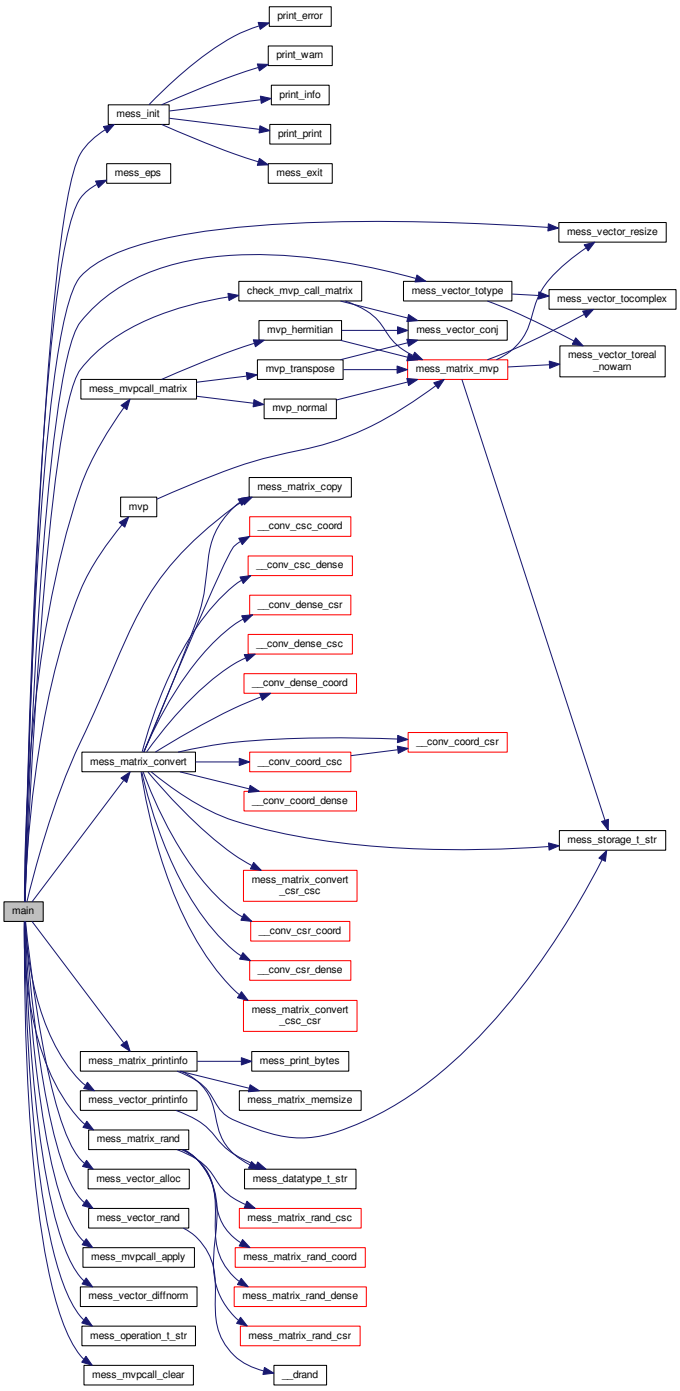
10.130.2.1 int check_mvp_call_matrix (mess_matrix *mat*, mess_operation_t *op1*, mess_operation_t *op2*, mess_vector *v1*, mess_vector *v2*)

Here is the call graph for this function:



10.130.2.2 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.131 check_norm.c File Reference

Macros

- #define CHECKNORM(NORMFUNC, A, NRM, SOL, EPS, ERR)

Functions

- int main (int argc, char **argv)

Variables

- const double rand50x50norm1 = 29.8018246341451
- const double rand50x50norminf = 28.9667636414232
- const double rand50x50normf = 28.8202850526756
- const double rand50x50norm2 = 25.2249050294185
- const double rand50x50norm2inv = 29.5091653482019
- const double randc50x50norm1 = 41.6678409812798
- const double randc50x50norminf = 43.6532647822569
- const double randc50x50normf = 40.8594662754399
- const double randc50x50norm2 = 35.6331007126333
- const double randc50x50norm2inv = 12.4776860845471
- const double issAnorm2inv = 2.572750289476989

10.131.1 Detailed Description

10.131.2 Macro Definition Documentation

10.131.2.1 #define CHECKNORM(*NORMFUNC*, **A**, *NRM*, *SOL*, *EPS*, *ERR*)

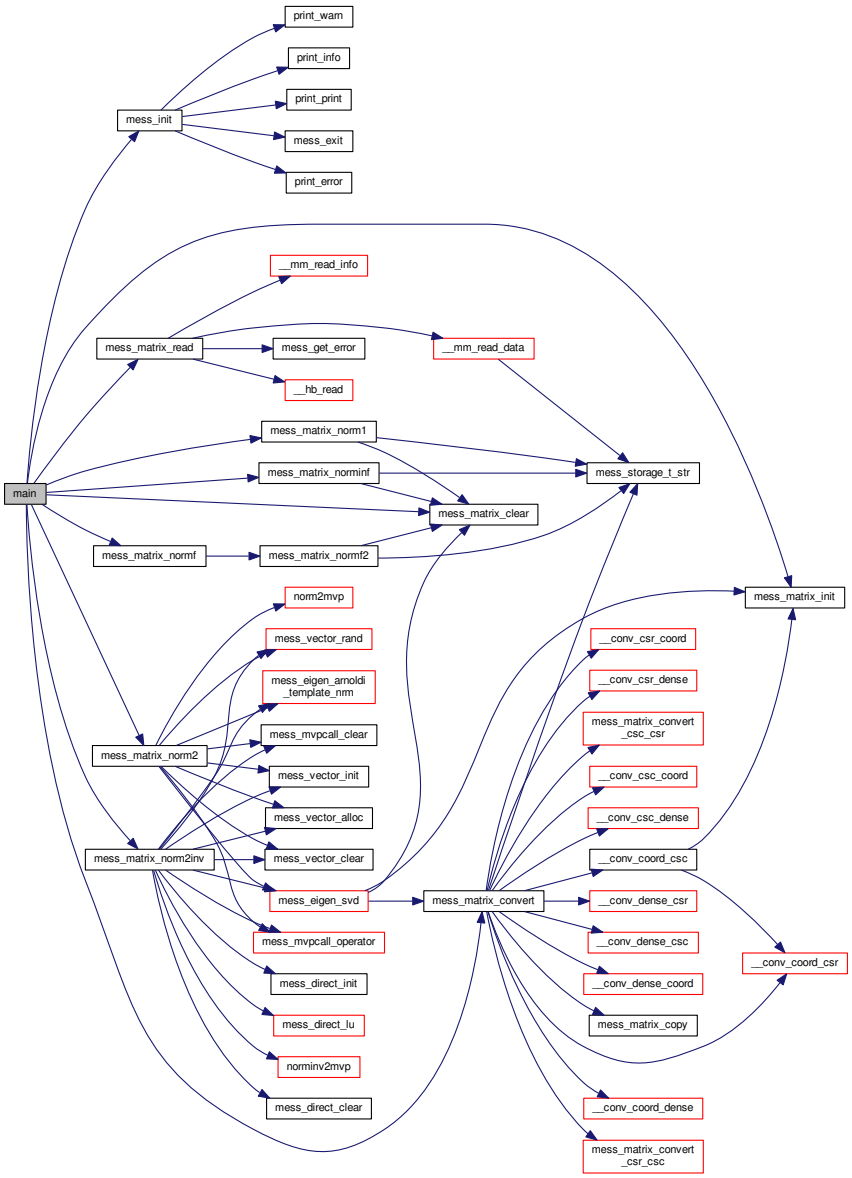
Value:

```
{  \
    NORMFUNC(A,&NRM);                                \
    if(fabs(NRM-SOL)>EPS){++ERR;                     \
        printf("%s\n",#NORMFUNC);                        \
        printf("Matrix:%s\n",#A);                        \
        printf("Norm:%e\n",NRM);                         \
        printf("Sol:%e\n",SOL);                          \
        printf("Diff:%e\n",fabs(NRM-SOL));               \
        printf("-----------------------\n");             \
    };                                               \
}
```

10.131.3 Function Documentation

10.131.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.131.4 Variable Documentation

10.131.4.1 const double issAnorm2inv = 2.572750289476989

10.131.4.2 const double rand50x50norm1 = 29.8018246341451

10.131.4.3 const double rand50x50norm2 = 25.2249050294185

10.131.4.4 const double rand50x50norm2inv = 29.5091653482019

10.131.4.5 const double rand50x50normf = 28.8202850526756

10.131.4.6 const double rand50x50norminf = 28.9667636414232

10.131.4.7 const double randc50x50norm1 = 41.6678409812798

10.131.4.8 const double randc50x50norm2 = 35.6331007126333

10.131.4.9 const double randc50x50norm2inv = 12.4776860845471

10.131.4.10 const double randc50x50normf = 40.8594662754399

10.131.4.11 const double randc50x50norminf = 43.6532647822569

10.132 check_one_value.c File Reference

Macros

- #define VAL 2.1

Functions

- static double f_one_value (double val)
- static mess_double_cpx_t f_one_valuec (mess_double_cpx_t valc)
- int main (int argc, char **argv)

10.132.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.132.2 Macro Definition Documentation

10.132.2.1 #define VAL 2.1

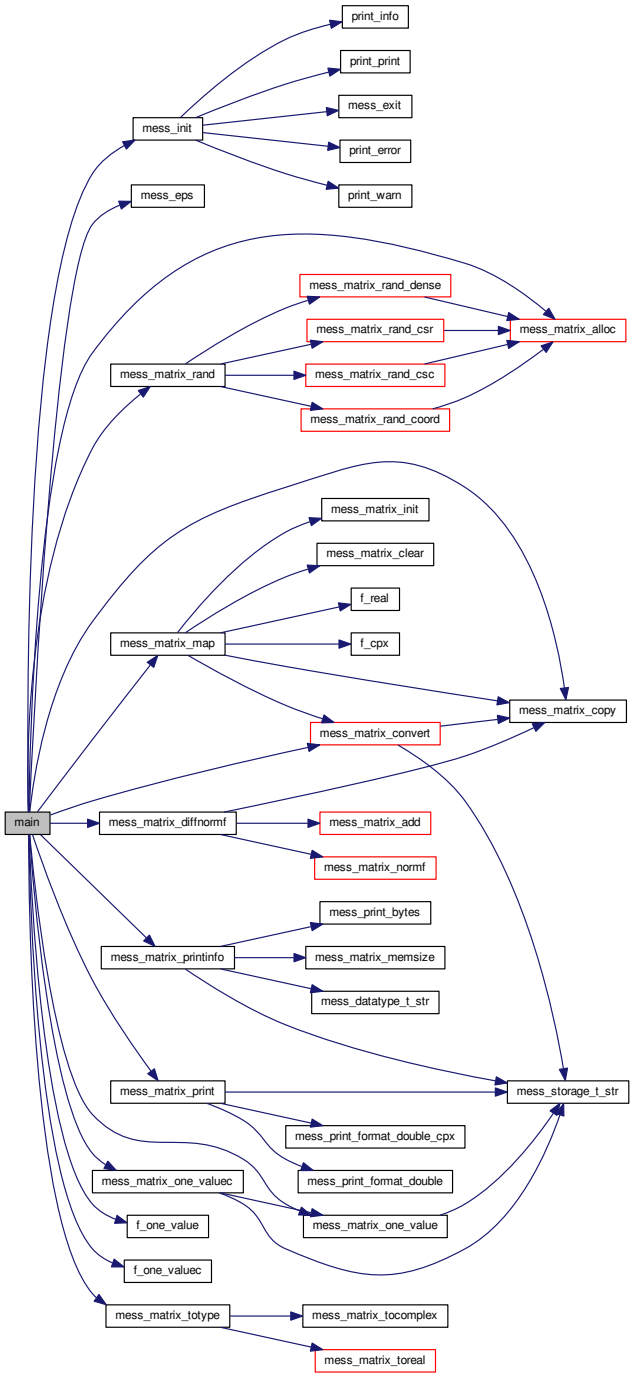
10.132.3 Function Documentation

10.132.3.1 static double f_one_value (double *val*) [static]

10.132.3.2 static mess_double_cpx_t f_one_valuec (mess_double_cpx_t *valc*) [static]

10.132.3.3 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.133 check_orth.c File Reference

Macros

- #define CHECKORTH(MAT, Q, EYE, TEMP1, TEMP2, ERR, EPS, DIFF)

Functions

- int main (int argc, char **argv)

10.133.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.133.2 Macro Definition Documentation

10.133.2.1 #define CHECKORTH(*MAT*, Q, *EYE*, *TEMP1*, *TEMP2*, *ERR*, *EPS*, DIFF)

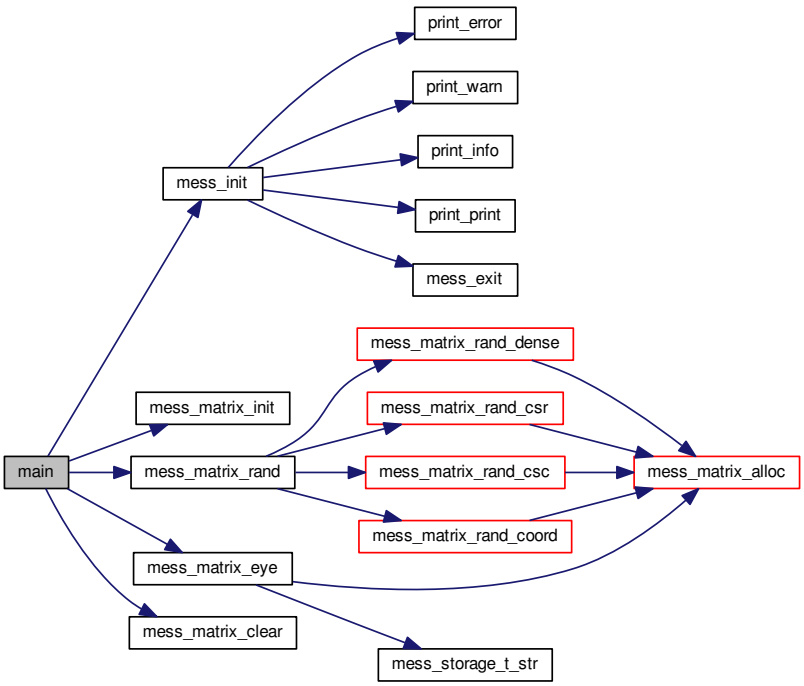
Value:

```
{                                       \
    mess_matrix_orth(MAT,Q);
        \
    mess_matrix_copy(Q,TEMP1);
        \
    if(MESS_IS_COMPLEX(MAT)){
      \
        CALL( mess_matrix_multiply(
      MESS_OP_HERMITIAN, TEMP1, MESS_OP_NONE, Q, TEMP2));           \
    }else{
      \
        CALL( mess_matrix_multiply(
      MESS_OP_HERMITIAN, TEMP1, MESS_OP_NONE, Q, TEMP2));           \
    }
      \
    CALL(mess_matrix_diffnorm(TEMP2,EYE,&DIFF));
                        \
    if(DIFF>EPS){                                                                   \
        ++ERR;                                                                      \
        printf("mess_matrix_mgsadd failed \n");                                     \
        printf("Error:%e\n",DIFF);                                                  \
    }                                                                               \
}
```

10.133.3 Function Documentation

10.133.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.134 check_perm.c File Reference

Macros

- #define CHECKPERM(A1, A2, DIFF, ERR, EPS)

Functions

- mess_int_t * permute (mess_int_t dim)
- int main ()

10.134.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.134.2 Macro Definition Documentation

10.134.2.1 #define CHECKPERM(*A1*, *A2*, DIFF, *ERR*, *EPS*)

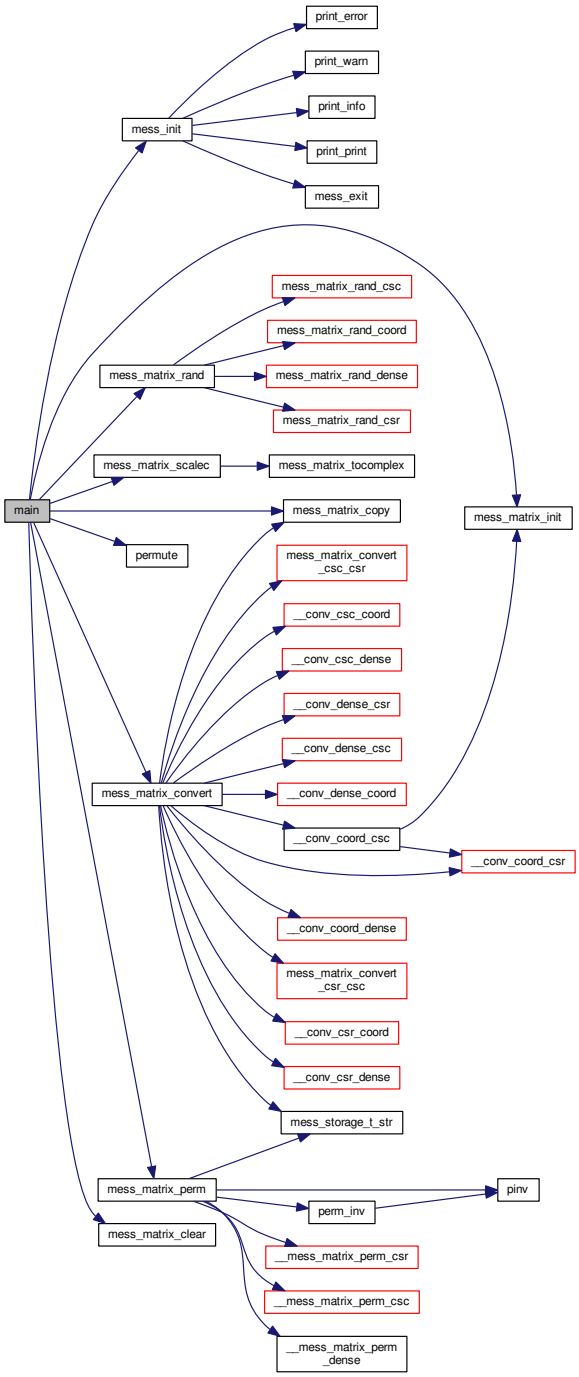
Value:

```
CALL(mess_matrix_diffnorm(A1,A2,&DIFF));                    \
printf("diff=%e\n",DIFF);                                   \
if(DIFF>EPS){                                               \
    printf("Failed with diff=%e\n",DIFF);                   \
    mess_matrix_printinfo(A1);
      \
    mess_matrix_printinfo(A2);                              \
}
```

10.134.3 Function Documentation

10.134.3.1 int main (void)

Here is the call graph for this function:



10.134.3.2 mess_int_t* permute (mess_int_t *dim*)

10.135 check_pinv.c File Reference

Functions

- double check_pseudo1 (mess_matrix A, mess_matrix Pinv)

  Check if pseudoinverse P_{inv} of A fullfills $AP_{inv}A = A$.

- double check_pseudo2 (mess_matrix A, mess_matrix Pinv)

  Check if pseudoinverse P_{inv} of A fullfills $P_{inv}AP_{inv} = P_{inv}$.

- double check_pseudo3 (mess_matrix A, mess_matrix Pinv)

  Check if pseudoinverse P_{inv} of A fullfills $(AP_{inv})^H = AP_{inv}$.

- double check_pseudo4 (mess_matrix A, mess_matrix Pinv)

  Check if pseudoinverse P_{inv} of A fullfills $(AP_{inv})^H = AP_{inv}$.

- int main (int argc, char $**$argv)

10.135.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

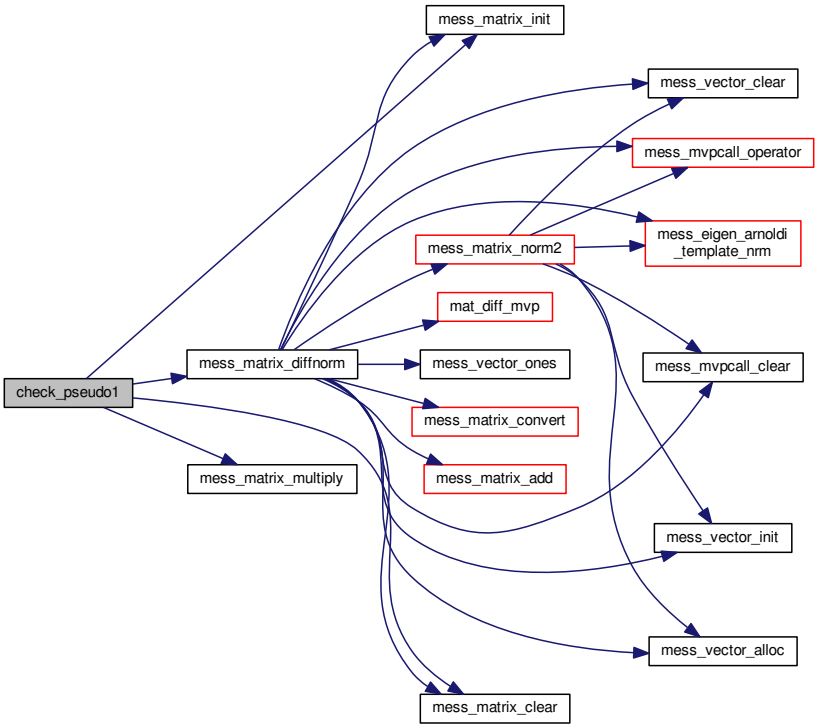
This function checks the mess_matrix_pinv function defined in pinv.c. It checks if

- $AP_{inv}A = A$
- $P_{inv}AP_{inv} = P_{inv}$
- $(AP_{inv})^H = AP_{inv}$
- $(P_{inv}A)^H = P_{inv}A$ where P_{inv} is the pseudoinverse of A.

10.135.2 Function Documentation

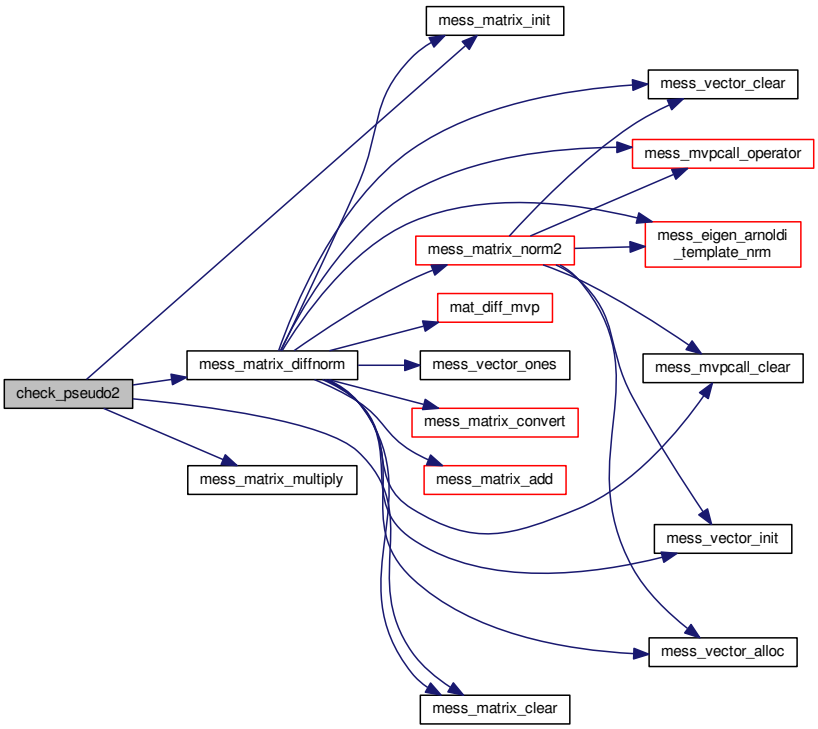
10.135.2.1 double check_pseudo1 (mess_matrix *A*, mess_matrix *Pinv*)

Here is the call graph for this function:



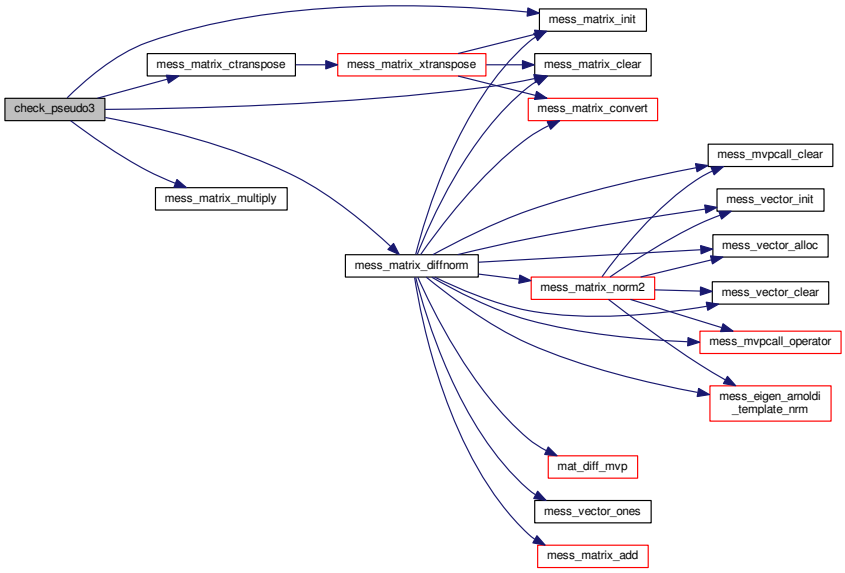
10.135.2.2 double check_pseudo2 (mess_matrix *A*, mess_matrix *Pinv*)

Here is the call graph for this function:



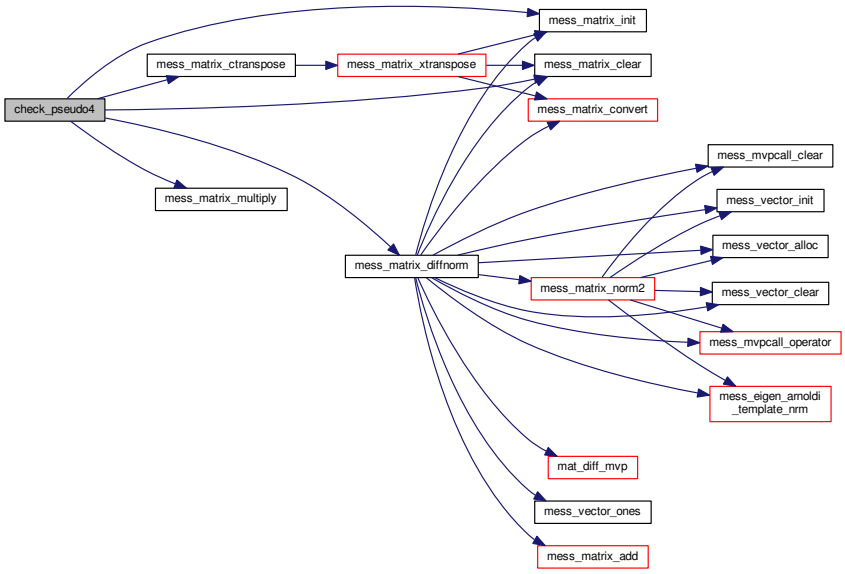
10.135.2.3 double check_pseudo3 (mess_matrix *A*, mess_matrix *Pinv*)

Here is the call graph for this function:



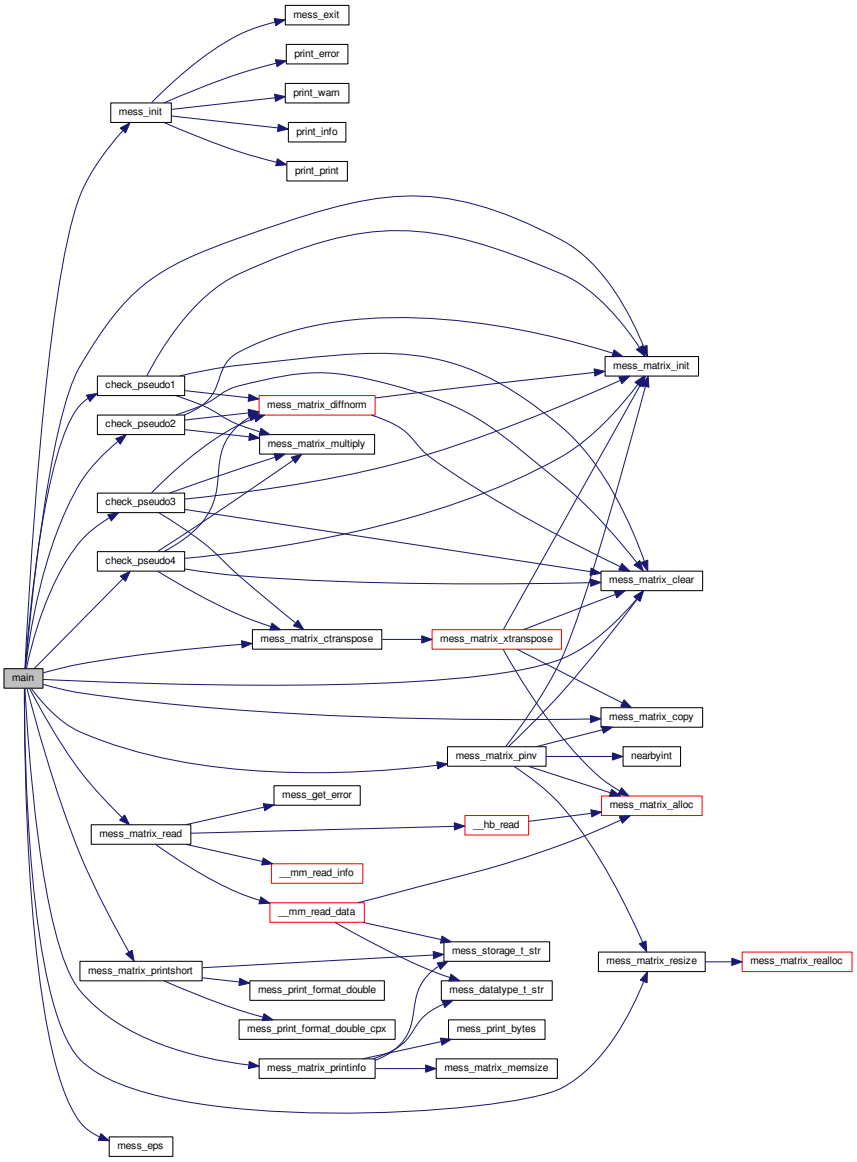
10.135.2.4 double check_pseudo4 (mess_matrix *A*, mess_matrix *Pinv*)

Here is the call graph for this function:



10.135.2.5 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.136 check_precond.c File Reference

Functions

- int check_precond (const char ∗name, mess_matrix A, mess_int_t maxit, double tol)

  Check different preconditioner.
- int main (int argc, const char ∗∗argv)

10.136.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

- "DIAG",
- "ILU0",
- "ILUK0",
- "ILUK1",
- "ILUK2",
- "SOR",
- "SSOR".

that should be put in the second argument of the second input.

10.136.2 Function Documentation

10.136.2.1 int check_precond (const char * *name*, mess_matrix *A*, mess_int_t *maxit*, double *tol*)

Parameters

in	*name*	input name of preconditioner
in	*A*	input matrix
in	*maxit*	input maximum number of iteration
in	*tol*	input tolerance for the iteration

Returns

zero on success or a non zero error code.

The check_precond function selects a preconditioner by the $name$, solves the linear system of equations

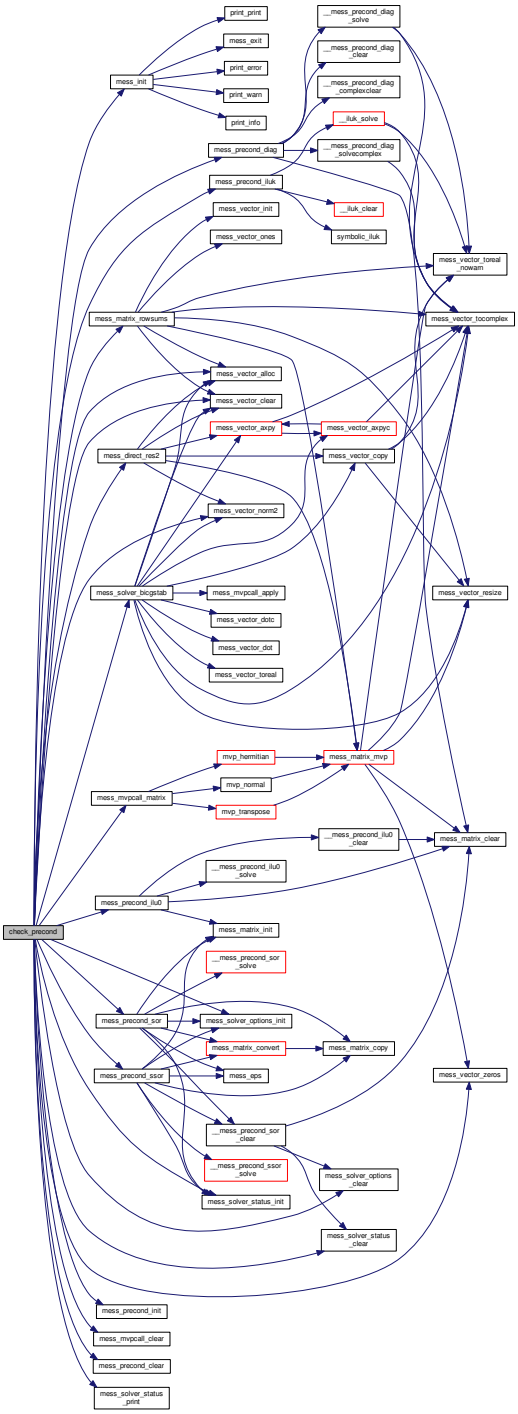
$$Ax = b$$

using the computed preconditioner and checks if the solution is computed correctly.
The entries of the right hand side b are computed by the sum of each row of a A.
Possible preconditioner names are:

- "DIAG",
- "ILU0",
- "ILUK0",
- "ILUK1",
- "ILUK2",
- "SOR",

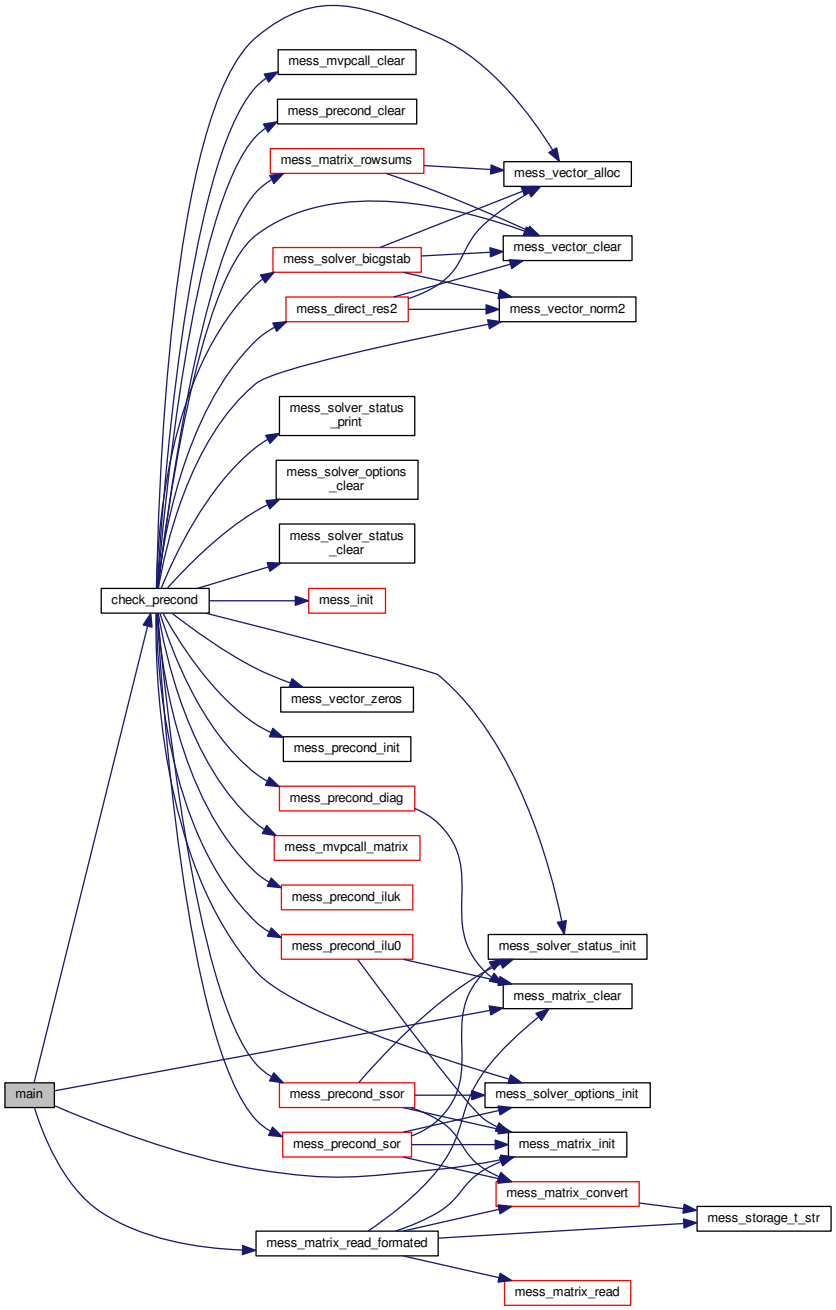
- "SSOR".

Here is the call graph for this function:



10.136.2.2 int main (int *argc*, const char ** *argv*)

Here is the call graph for this function:



10.137 check_proj_sym.c File Reference

Macros

- #define CHECKPROSYM(A, SOL, DIFF, EPS, ERR)

Functions

- int main ()

Variables

- double matrix_A [9] = { 1, 2, 3, 4, 5, 6, 7, 8, 9}
- double matrix_AS [9] = { 1, 3, 5, 3, 5, 7, 5, 7, 9}

10.137.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.137.2 Macro Definition Documentation

10.137.2.1 #define CHECKPROSYM(A, SOL, DIFF, EPS, ERR)

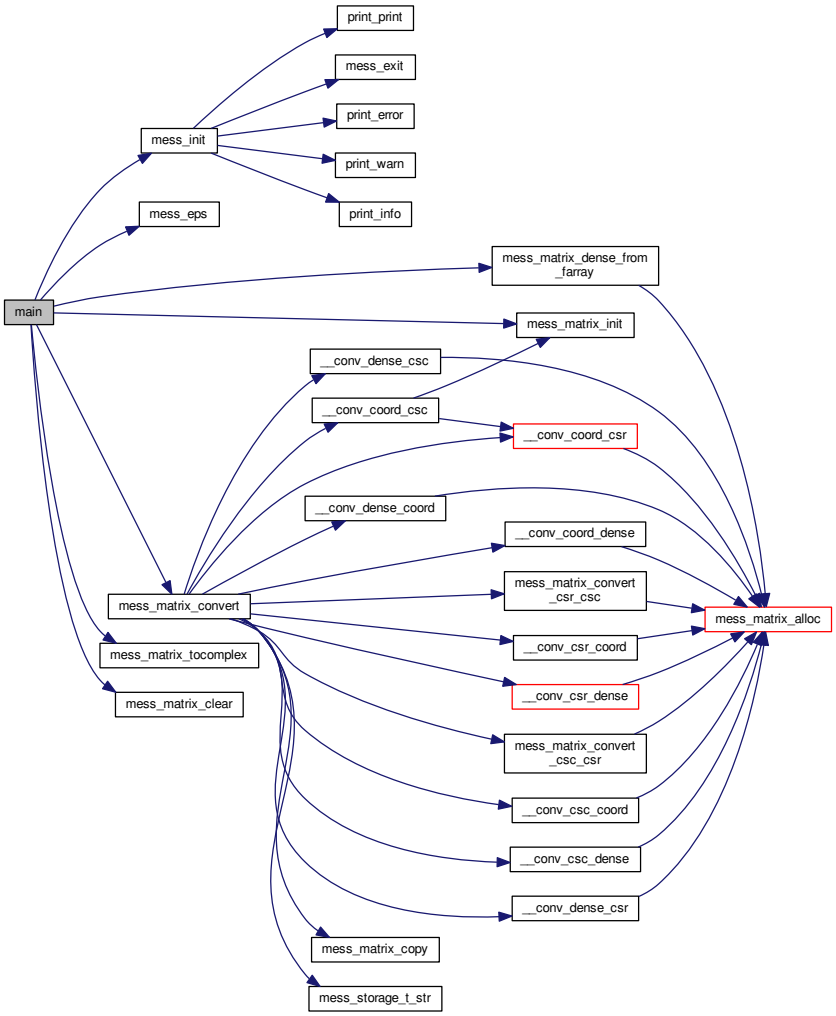
Value:

```
{                                   \
    mess_matrix_proj_sym(A);
      \
    mess_matrix_diffnorm(A,SOL,&(DIFF));                                \
    if((DIFF)>(EPS))     {                                              \
        ++(ERR);                                                        \
        mess_matrix_printinfo(A);                                       \
    }                                                                   \
}
```

10.137.3 Function Documentation

10.137.3.1 int main (void)

Here is the call graph for this function:



10.137.4 Variable Documentation

10.137.4.1 double matrix_A[9] = { 1, 2, 3, 4, 5, 6, 7, 8, 9}

10.137.4.2 double matrix_AS[9] = { 1, 3, 5, 3, 5, 7, 5, 7, 9}

10.138 check_rank.c File Reference

Functions

- int main (int argc, char **argv)

10.138.1 Detailed Description

Author

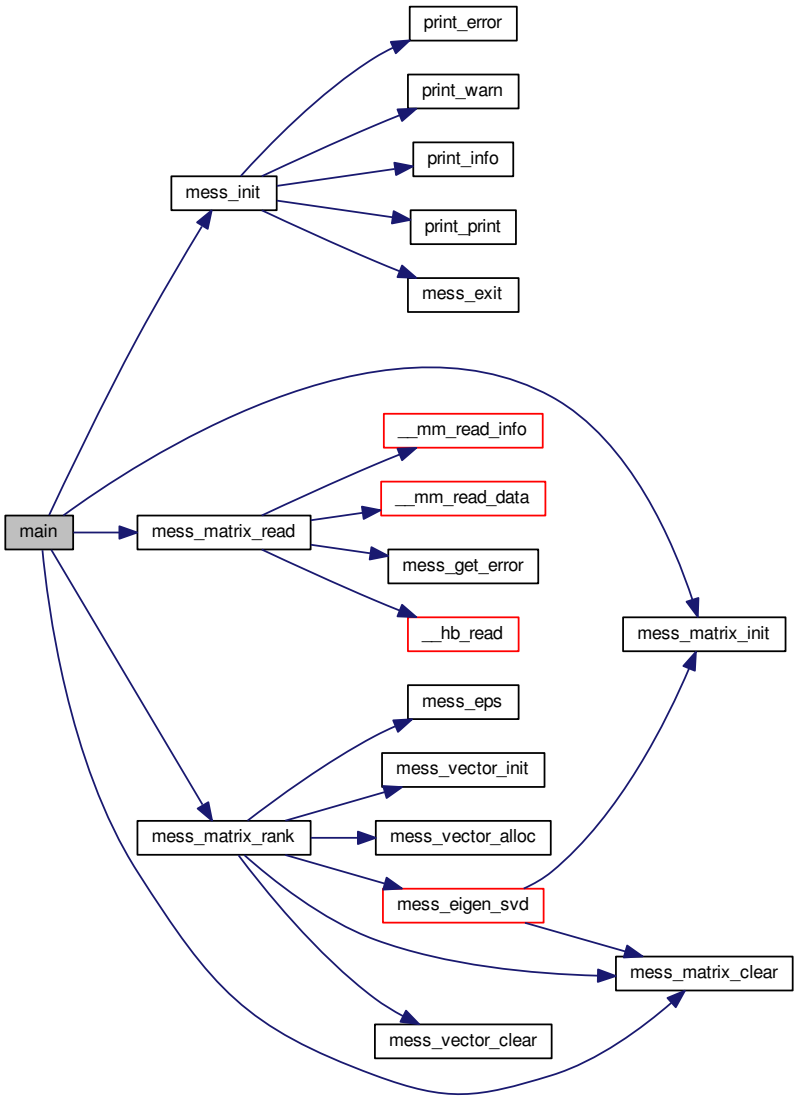
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_matrix_rank function defined in rank.c that means it checks if the rank of a matrix is computed correctly.

10.138.2 Function Documentation

10.138.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.139 check_real_imag_part.c File Reference

Functions

- int main (int argc, char **argv)

10.139.1 Detailed Description

Author

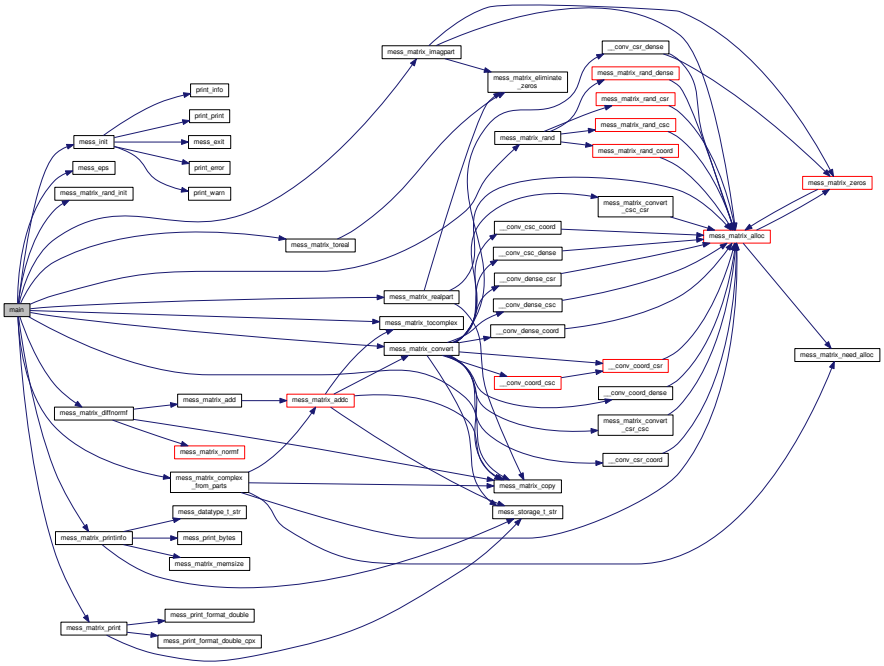
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_matrix_realpart mess_matrix_imagpart mess_matrix_complex_from_parts function defined in realimag.c that means it uses these functions together to tests against each other.

10.139.2 Function Documentation

10.139.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.140 check_rowsub.c File Reference

Macros

- #define CHECKROWSUB(A, TEMP, COL1, COL2, SOL, ERR, EPS, DIFF)

Functions

- int main ()

10.140.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.140.2 Macro Definition Documentation

10.140.2.1 #define CHECKROWSUB(A, TEMP, COL1, COL2, SOL, ERR, EPS, DIFF)

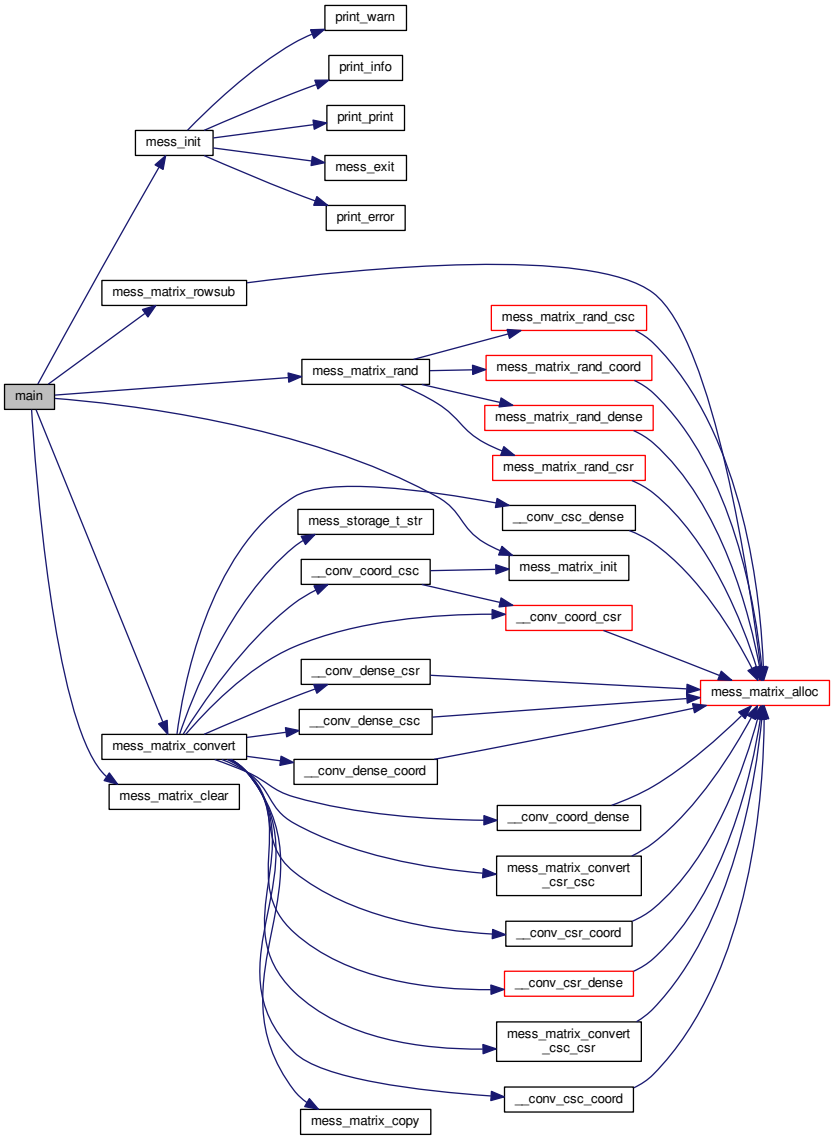
Value:

```
{                                   \
    mess_matrix_rowsub(A, COL1, COL2, TEMP);
      \
    mess_matrix_diffnorm(SOL,TEMP,&DIFF);                                   \
    /*mess_matrix_print(TEMP);*/                                            \
    printf("%e\n",DIFF);                                                    \
    if(DIFF>EPS){++ERR;}                                                    \
}
```

10.140.3 Function Documentation

10.140.3.1 int main (void)

Here is the call graph for this function:



10.141 check_scalem.c File Reference

Functions

- int main (int argc, char ∗∗argv)

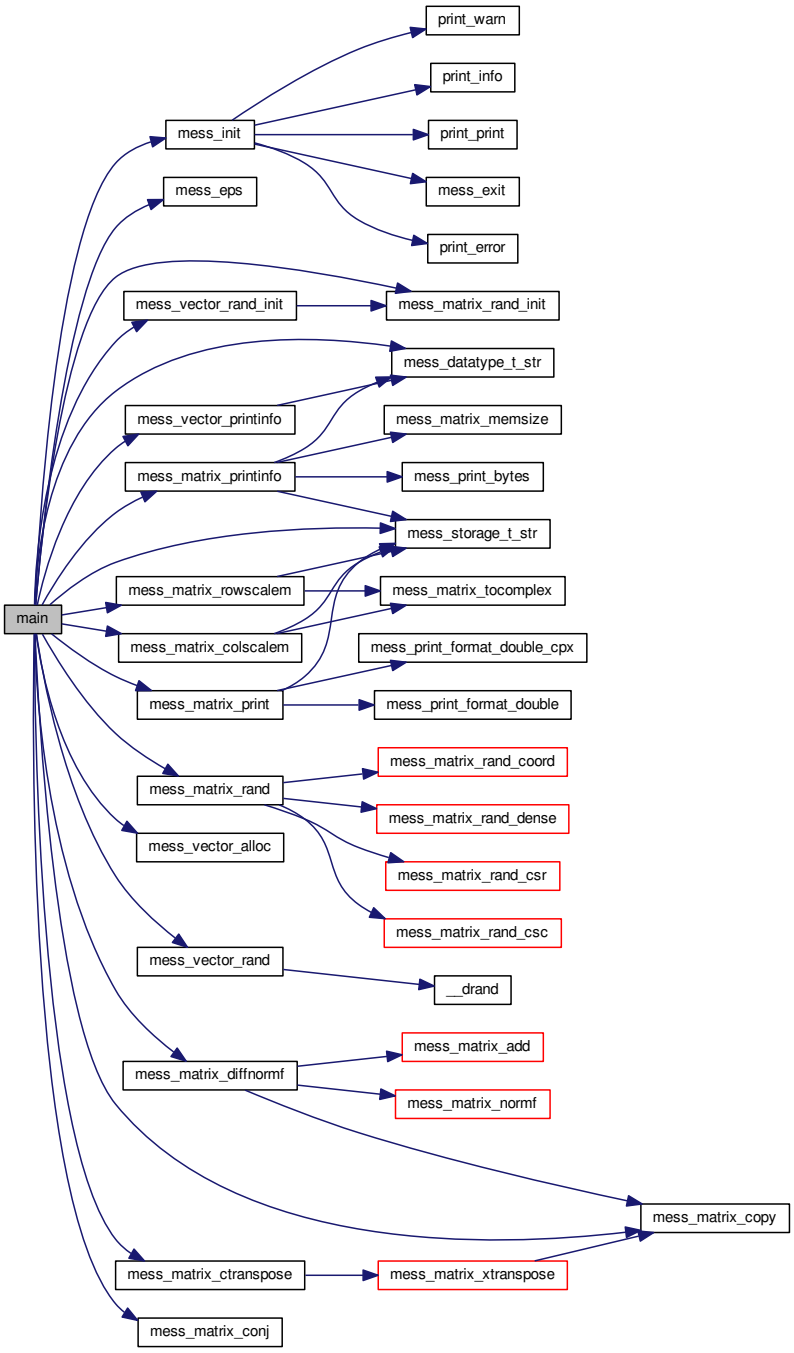
10.141.1 Detailed Description

This function checks the mess_matrix_rowscalem and mess_matrix_colscalem function defined in scal.c that means it checks the results of the function against each other.

10.141.2 Function Documentation

10.141.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.142 check_schur.c File Reference

Functions

- int main (int argc, char **argv)

10.142.1 Detailed Description

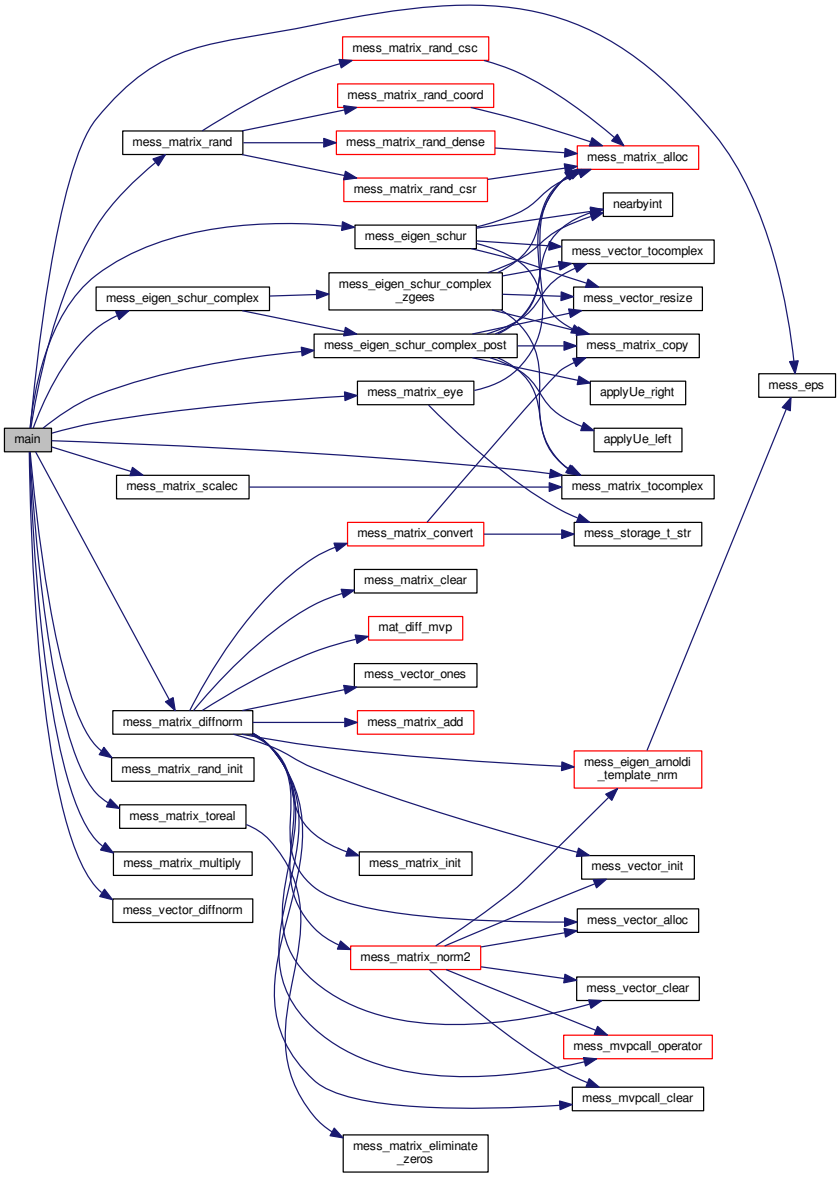
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.142.2 Function Documentation

10.142.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.143 check_set_get_col_row.c File Reference

Functions

- static double v_map_real (double val)
- static mess_double_cpx_t v_map_cpx (mess_double_cpx_t val)
- int main (int argc, char **argv)

10.143.1 Detailed Description

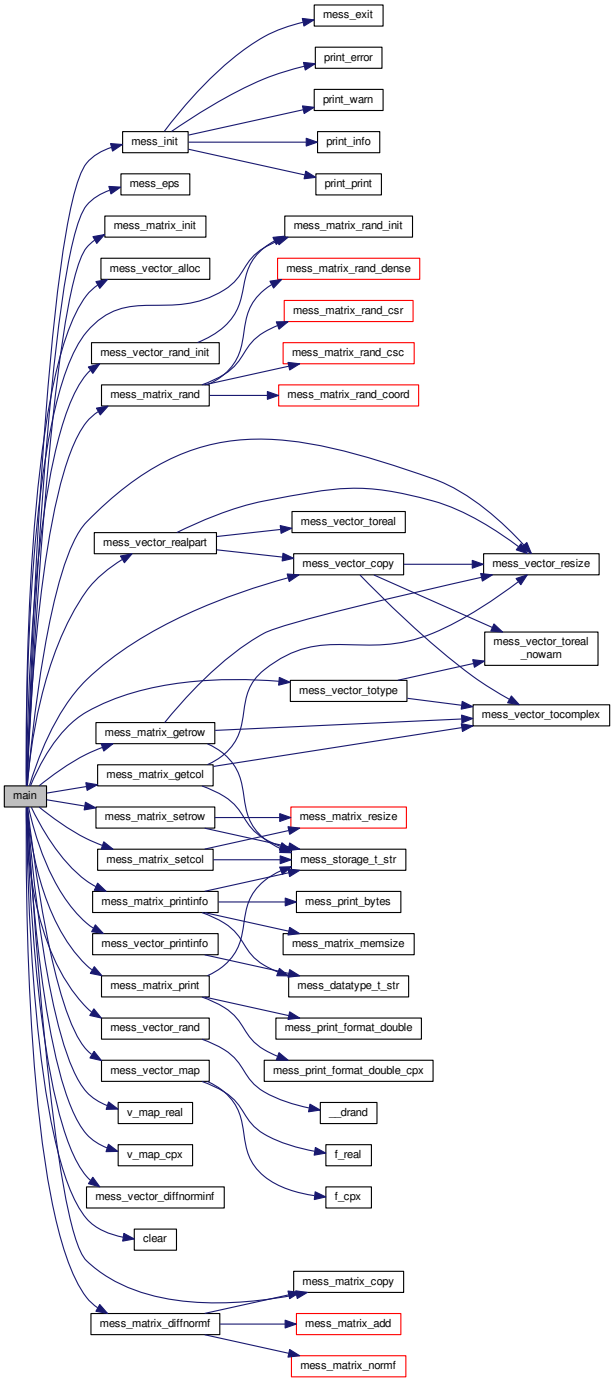
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.143.2 Function Documentation

10.143.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.143.2.2 static mess_double_cpx_t v_map_cpx (mess_double_cpx_t *val*) [static]

10.143.2.3 static double v_map_real (double *val*) [static]

10.144 check_setelement.c File Reference

Functions

- int main (int argc, char *argv[ ])

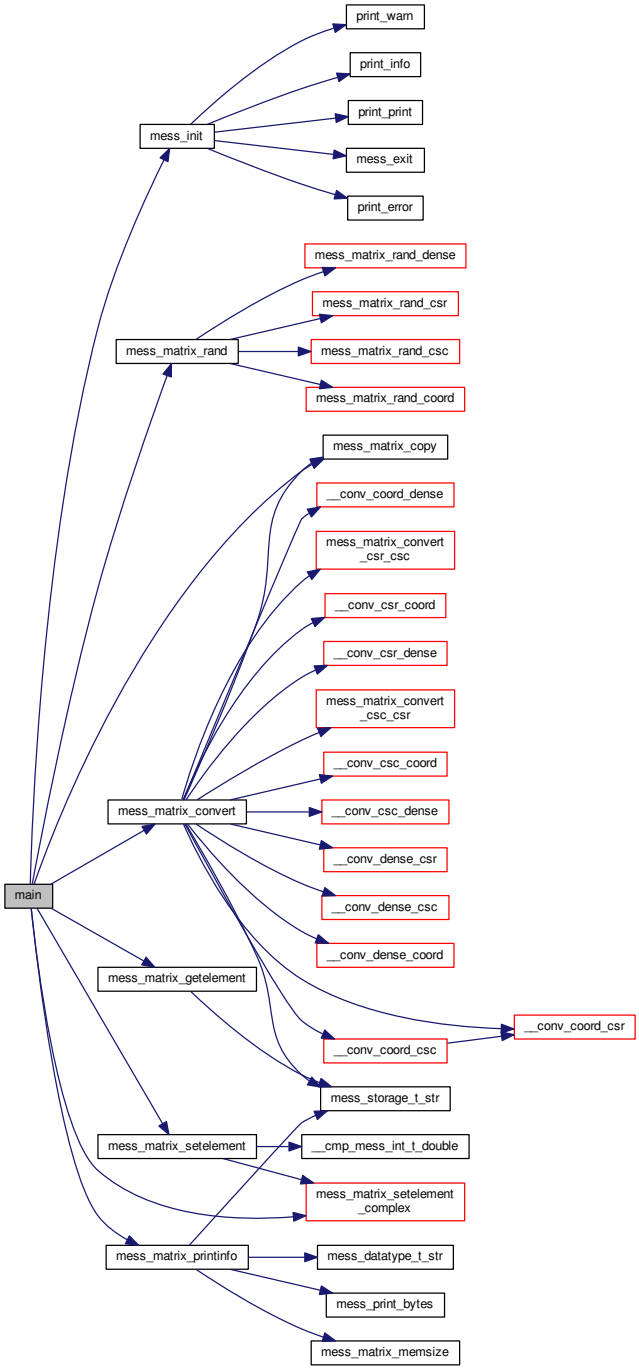
10.144.1 Detailed Description

This function checks the mess_matrix_setelement and mess_matrix_setelement_complex functions defined in setelement.c that means it checks if an element of a matrix is overwritten correctly with a given element by use of mess_matrix_getelement.

10.144.2 Function Documentation

10.144.2.1 int main (int *argc*, char ∗ *argv[ ]*)

Here is the call graph for this function:



10.145 check_sign.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.145.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_eigen_sign function defined in sign.c that means it checks if the sign of a matrix is computed correctly, by the identity

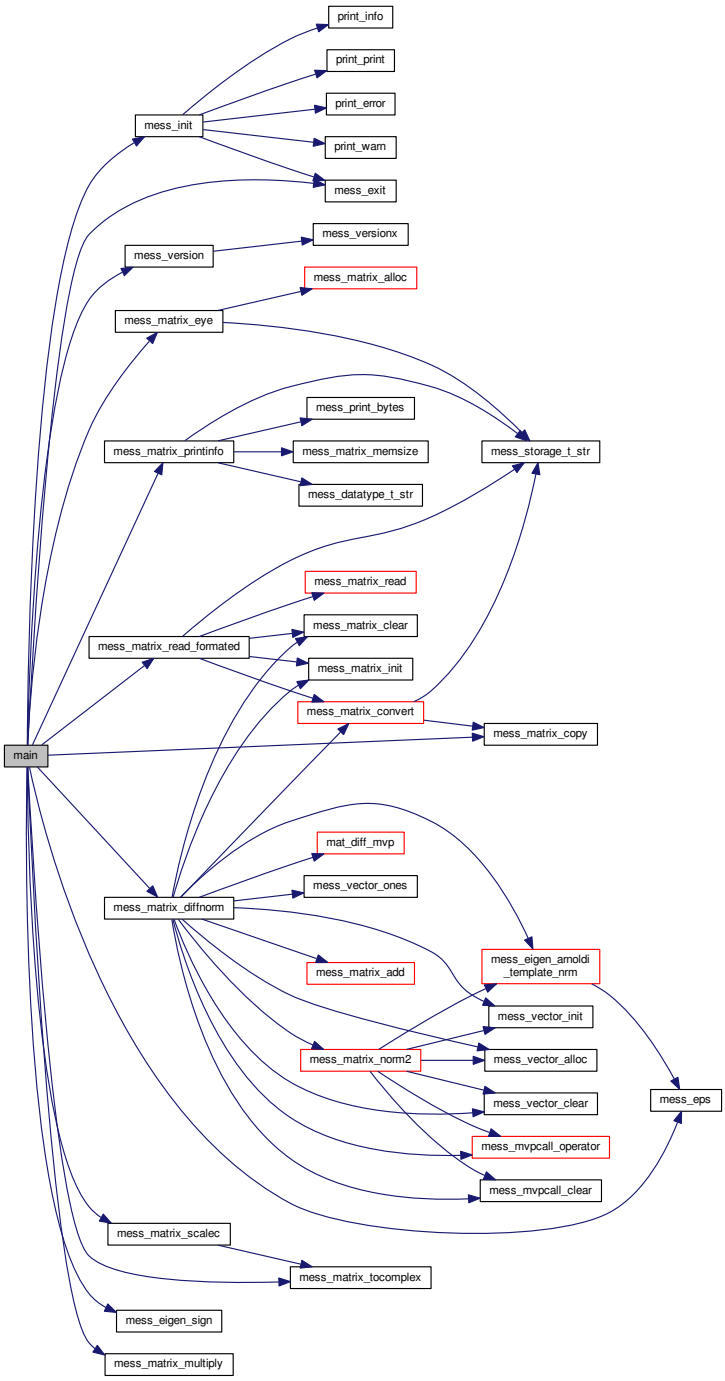
$$sign(A)^2 = I$$

. is computed correctly.

10.145.2 Function Documentation

10.145.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.146 check_so1_handles.c File Reference

Macros

- #define CHECK_FUNCTION(INFO, FNAME, DIFFFUNC, Y1, Y2, DIFF, ERR, EPS)

Functions

- int main (int argc, char **argv)

10.146.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.146.2 Macro Definition Documentation

10.146.2.1 #define CHECK_FUNCTION(INFO, FNAME, DIFFFUNC, Y1, Y2, DIFF, ERR, EPS)

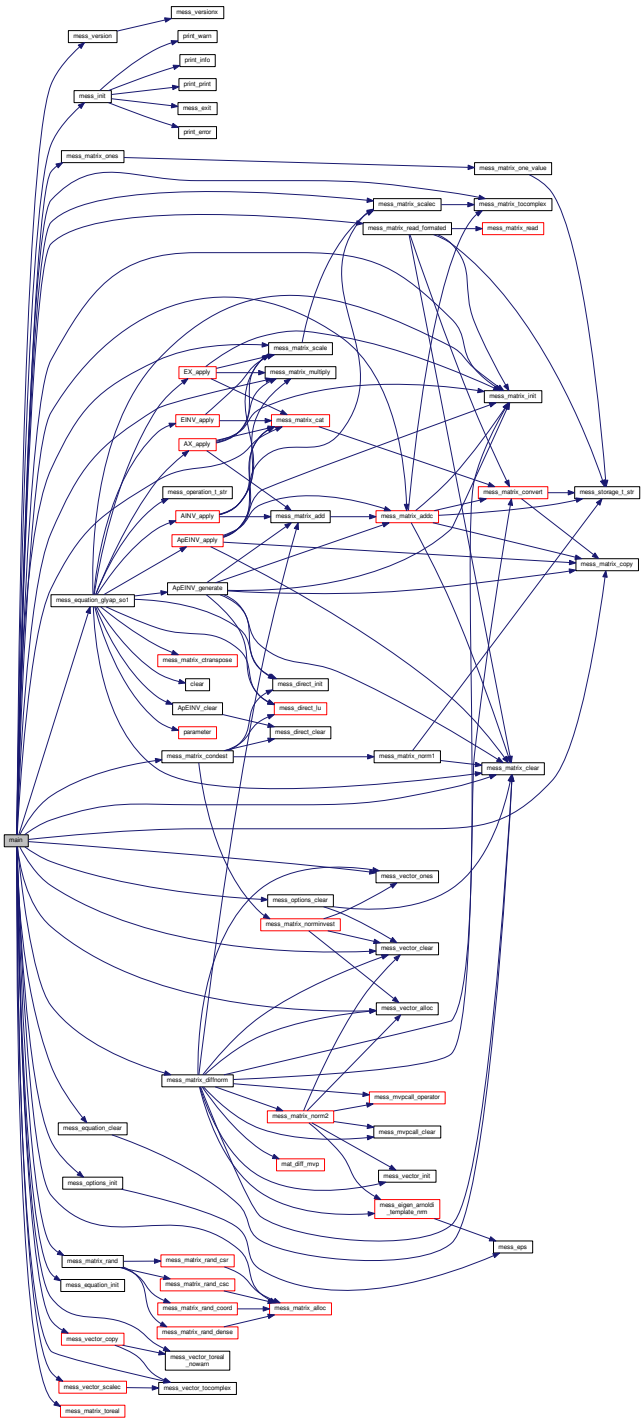
Value:

```
CALL(DIFFFUNC(Y1,Y2,&DIFF));                                    \
if(DIFF>EPS){                                                   \
    printf(INFO " " #FNAME " failed\tdiff=%e\n",diff);          \
    ERR++;                                                      \
}else{                                                          \
    printf(INFO " " #FNAME " passed with diff=%e.\n",diff);     \
}                                                               \
```

10.146.3 Function Documentation

10.146.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.147 check_so2_handles.c File Reference

Macros

- #define CHECK_FUNCTION(INFO, FNAME, DIFFFUNC, Y1, Y2, DIFF, ERR, EPS)

Functions

- int main (int argc, char **argv)

10.147.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.147.2 Macro Definition Documentation

10.147.2.1 #define CHECK_FUNCTION(INFO, FNAME, DIFFFUNC, Y1, Y2, DIFF, ERR, EPS)

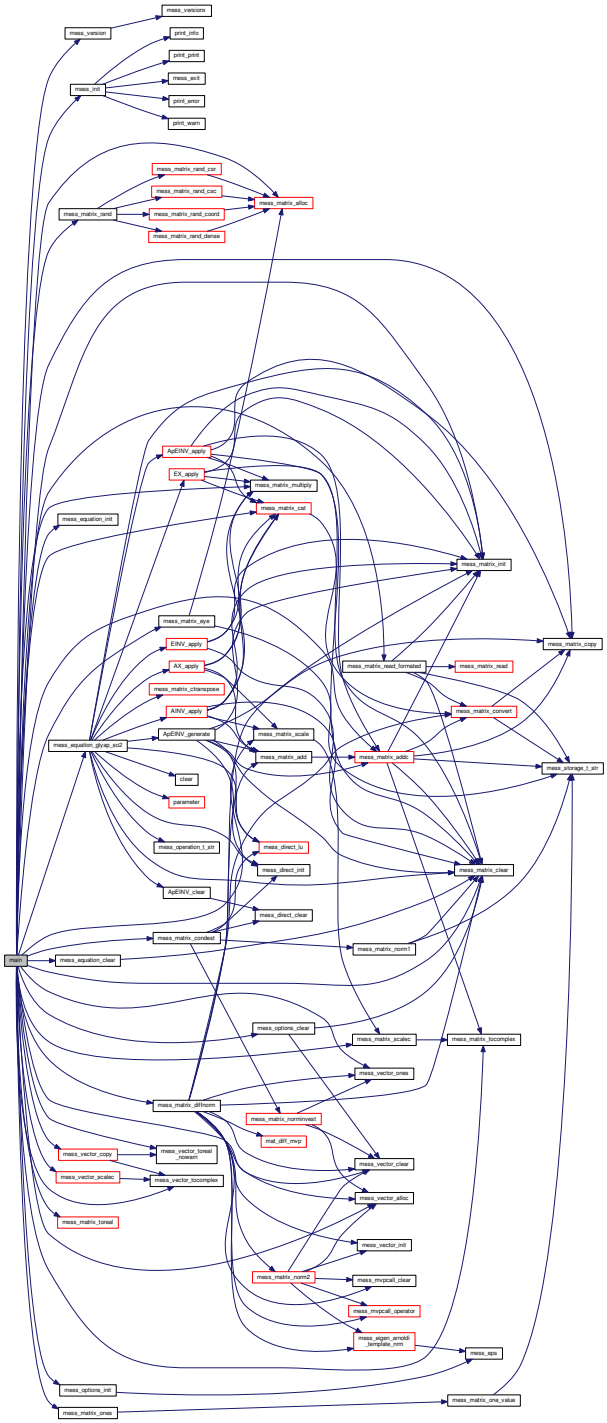
Value:

```
CALL(DIFFFUNC(Y1,Y2,&DIFF));                                    \
if(DIFF>EPS){                                                   \
    printf(INFO " " #FNAME " failed\tdiff=%e\n",diff);          \
    ERR++;                                                      \
}else{                                                          \
    printf(INFO " " #FNAME " passed with diff=%e.\n",diff);     \
}                                                               \
```

10.147.3 Function Documentation

10.147.3.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.148 check_sub.c File Reference

Macros

- #define CHECKSUB(A, TEMP, ROW1, ROW2, COL1, COL2, SOL, ERR, EPS, DIFF)

Functions

- int main ()

10.148.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.148.2 Macro Definition Documentation

10.148.2.1 #define CHECKSUB(A, *TEMP, ROW1, ROW2, COL1, COL2, SOL, ERR, EPS,* DIFF)

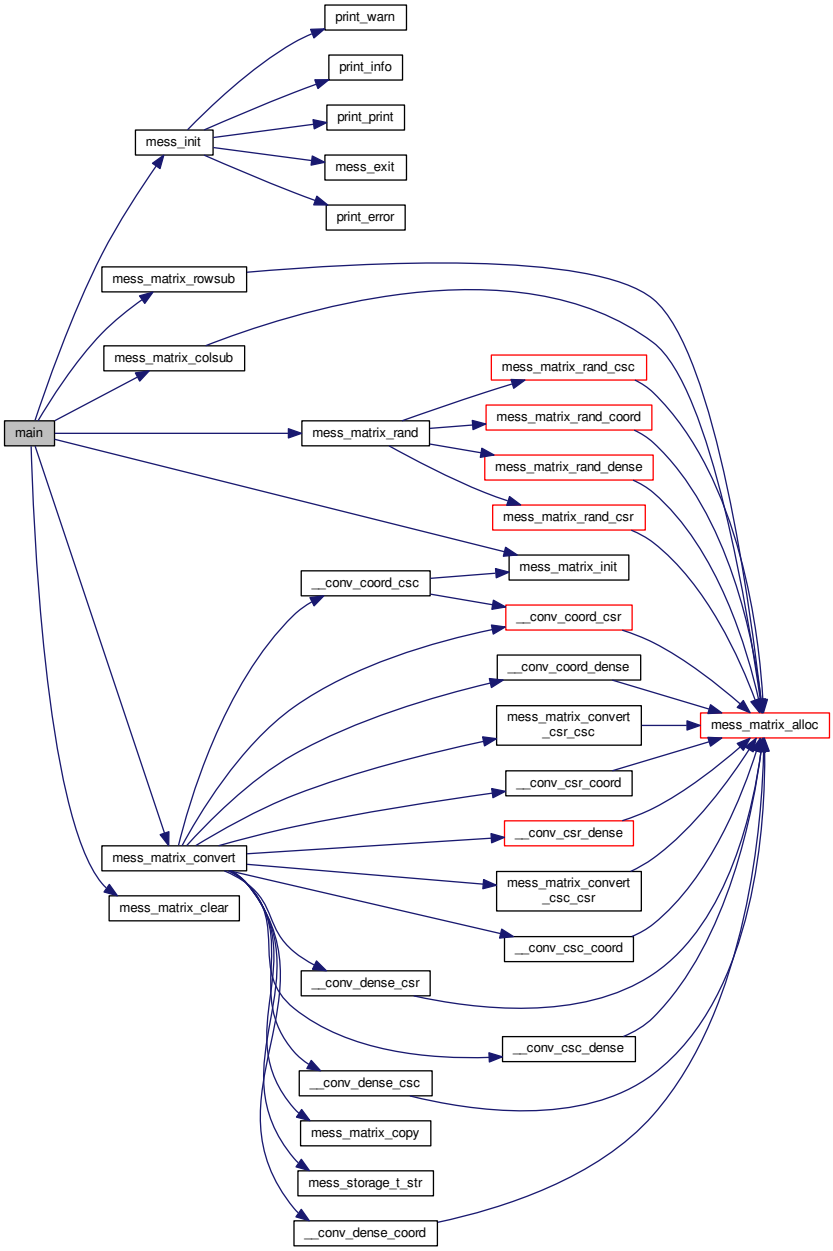
Value:

```
{                \
    mess_matrix_sub(A,ROW1,ROW2, COL1,COL2, TEMP);
      \
    mess_matrix_diffnorm(SOL,TEMP,&DIFF);                                    \
    /*mess_matrix_print(TEMP);*/                                             \
    printf("%e\n",DIFF);                                                     \
    if(DIFF>EPS){++ERR;}                                                     \
}
```

10.148.3 Function Documentation

10.148.3.1 int main (void)

Here is the call graph for this function:



10.149 check_svd.c File Reference

Functions

- int main (int argc, char **argv)

10.149.1 Detailed Description

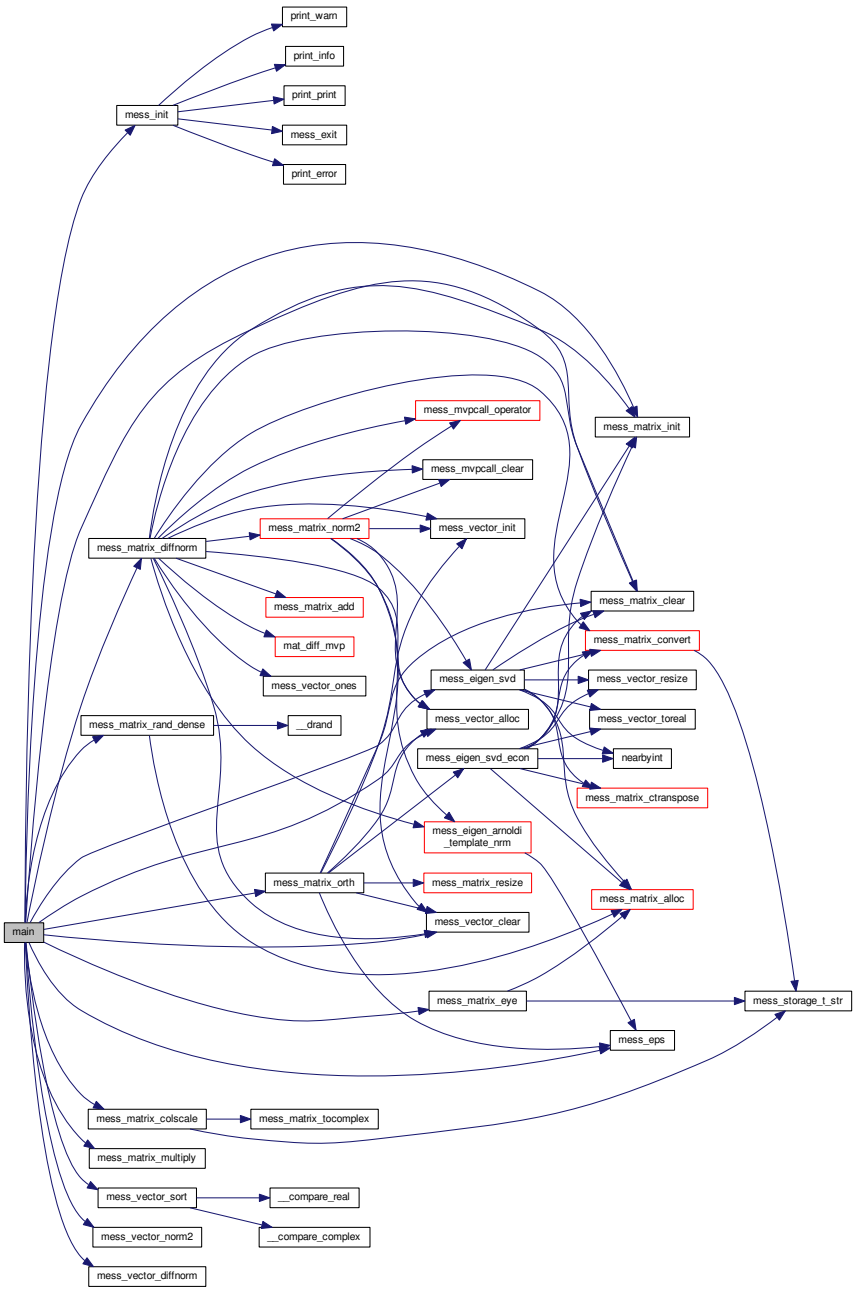
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.149.2 Function Documentation

10.149.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.150 check_svd_complex.c File Reference

Functions

- int main (int argc, char **argv)

10.150.1 Detailed Description

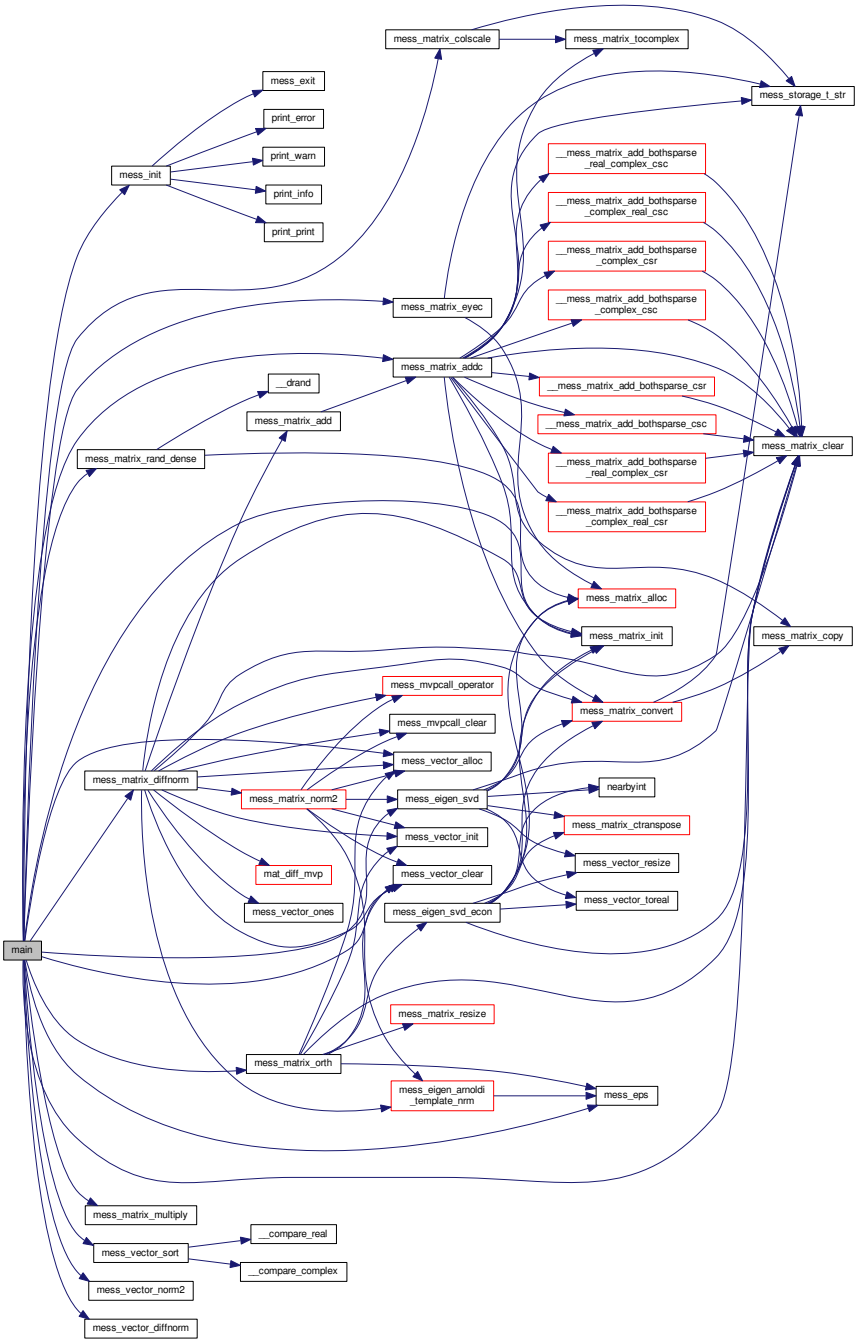
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.150.2 Function Documentation

10.150.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.151 check_svd_econ.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.151.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_eigen_svd_econ function defined in svd.c that means it checks if the economy size singular value decomposition

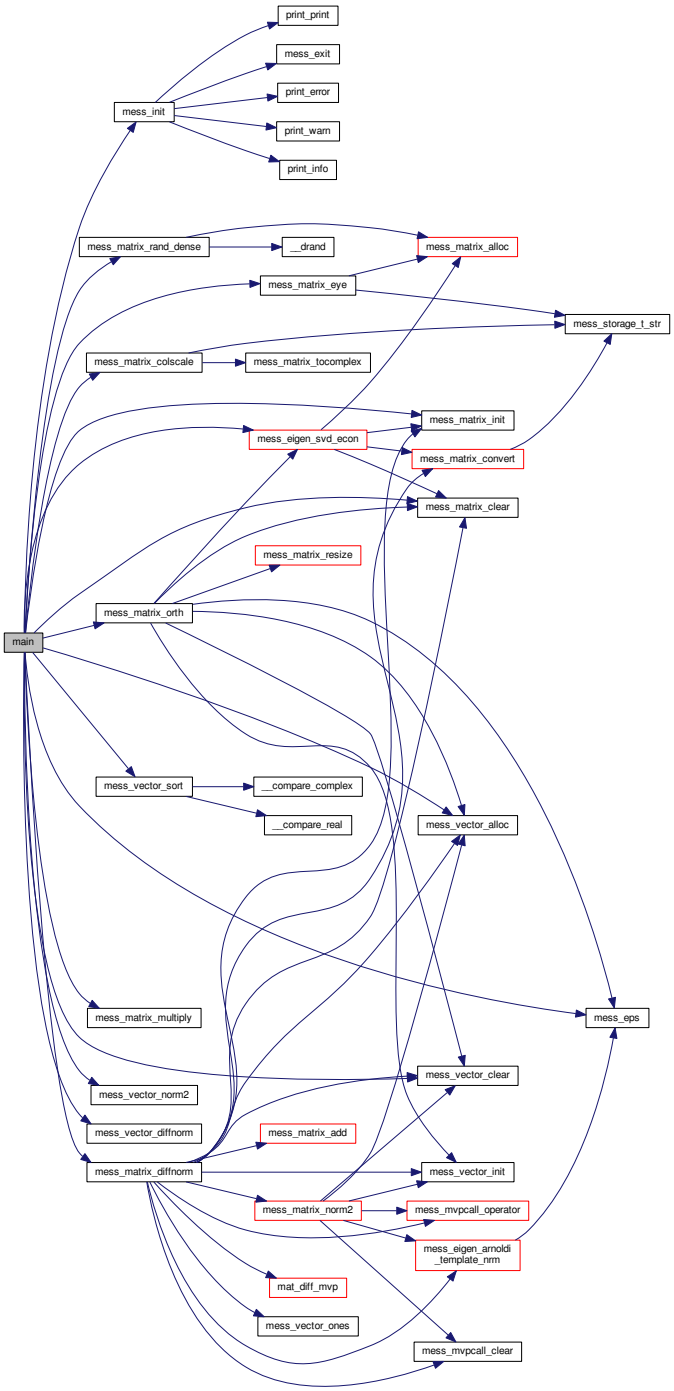
$$A = USV^T$$

is computed correctly.

10.151.2 Function Documentation

10.151.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.152 check_svd_econ_complex.c File Reference

Functions

- int main (int argc, char ∗∗argv)

10.152.1 Detailed Description

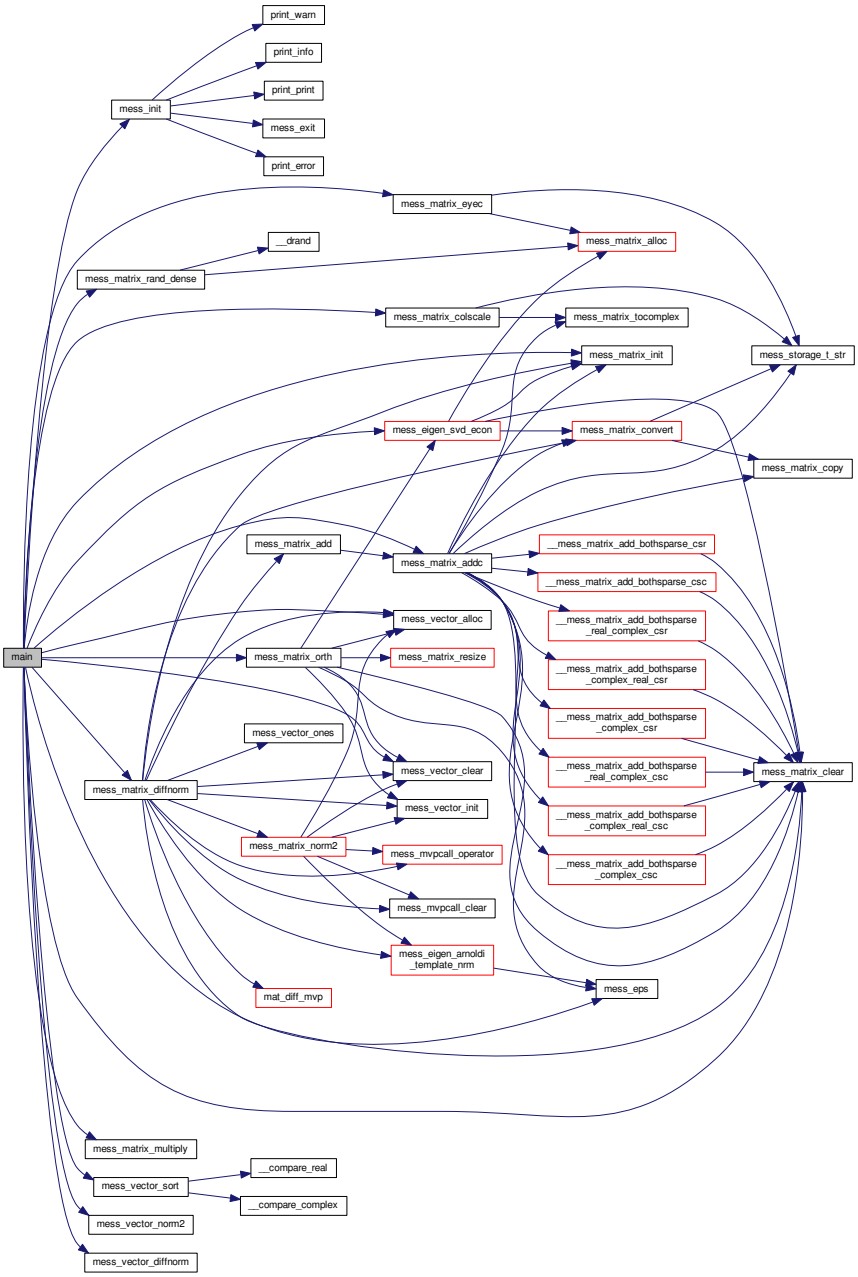
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.152.2 Function Documentation

10.152.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.153 check_trace.c File Reference

Macros

- #define CHECKTRACE(A, CHECK, EPS, ERR)

Functions

- int main (int argc, char **argv)

Variables

- double trreal =0
- mess_double_cpx_t trcpx =0

10.153.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.153.2 Macro Definition Documentation

10.153.2.1 #define CHECKTRACE(A, *CHECK*, *EPS*, *ERR*)

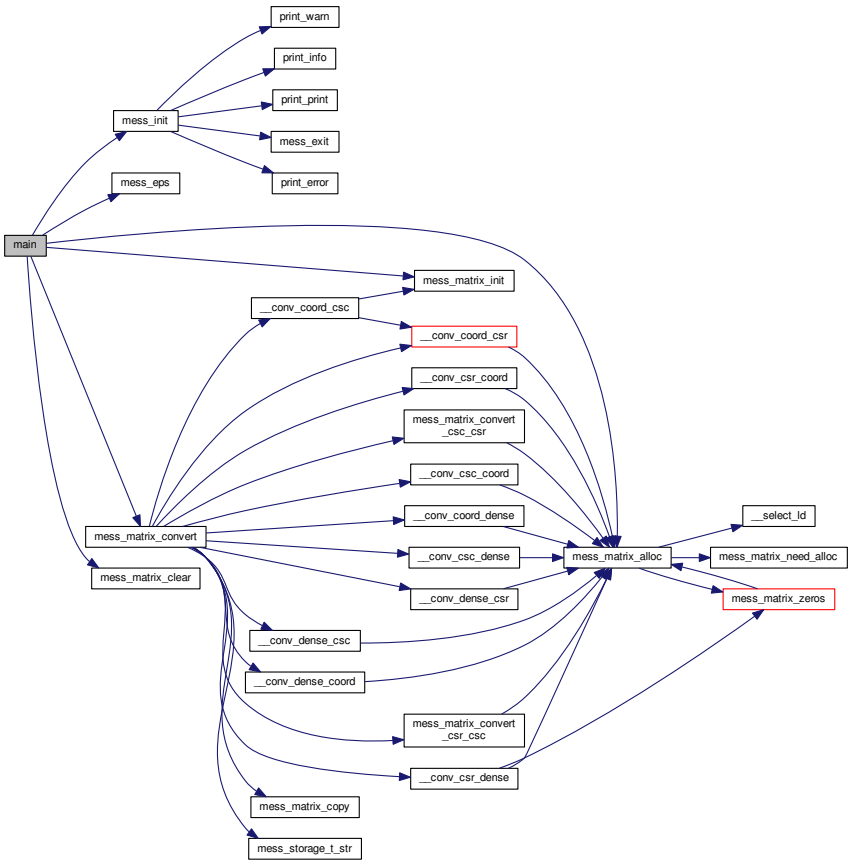
Value:

```
{                       \
    trreal=0;                                                   \
    trcpx =0;                                                   \
    if(MESS_IS_REAL((A))){                                      \
        mess_matrix_trace((A),&trreal);                         \
        if(fabs(trreal-((double)CHECK)) > 10* (EPS)){++(ERR); \
            fprintf(stderr,"err: %20.16e\n", fabs(trreal-((double)CHECK)));\
        }       \
    }else if(MESS_IS_COMPLEX((A))){                             \
        mess_matrix_tracec((A), &trcpx);                        \
        if(cabs(trcpx-((mess_double_cpx_t)CHECK)) > 10* (EPS)){++(ERR);};      \
    }                                                           \
}                                                                   \
```

10.153.3 Function Documentation

10.153.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.153.4 Variable Documentation

10.153.4.1 mess_double_cpx_t trcpx =0

10.153.4.2 double trreal =0

10.154 check_transpose.c File Reference

Macros

- #define CHECKTRANSPOSE(A, SOL, DIFF, EPS, ERR)

Functions

- int main ()

Variables

- double matrix_A [9] = { 1, 2, 3, 4, 5, 6, 7, 8, 9}
- double matrix_AT [9] = { 1, 4, 7, 2, 5, 8, 3, 6, 9}

10.154.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.154.2 Macro Definition Documentation

10.154.2.1 #define CHECKTRANSPOSE(A, SOL, DIFF, EPS, ERR)

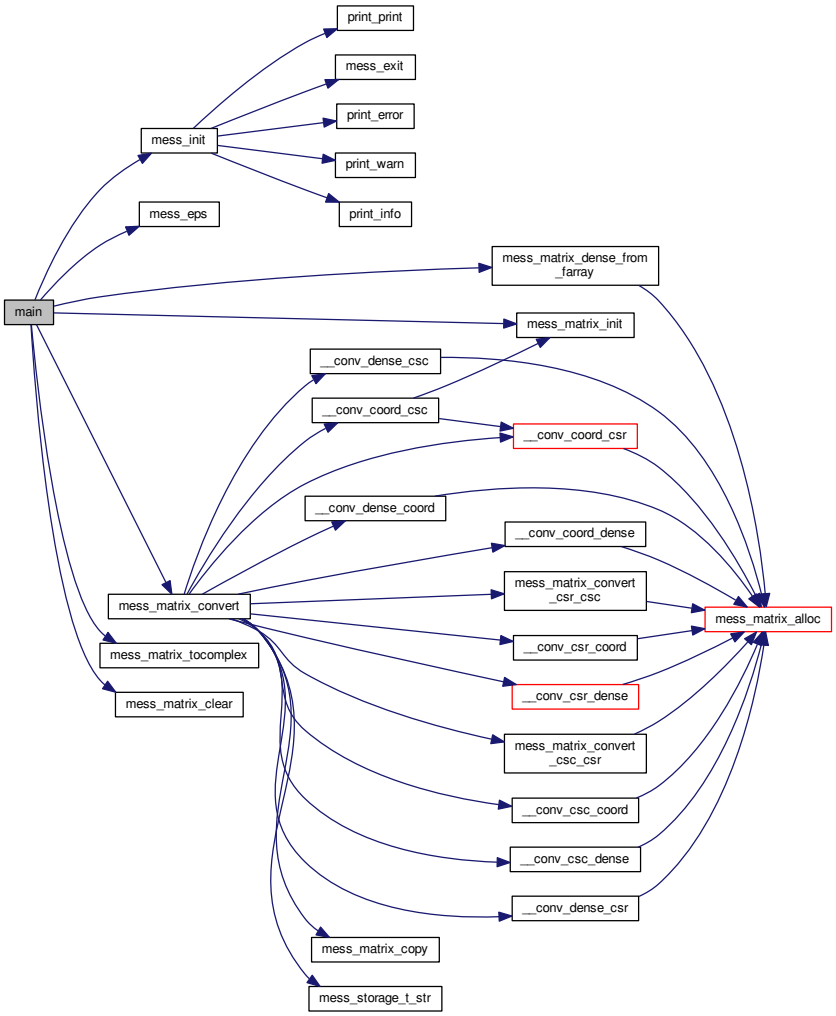
Value:

```
{                                   \
    mess_matrix_ctranspose(A,TEMP);
      \
    mess_matrix_diffnorm(TEMP,SOL,&(DIFF));                                 \
    if((DIFF)>(EPS))     {                                                  \
        ++(ERR);                                                            \
        mess_matrix_printinfo(A);                                           \
    }                                                                       \
}
```

10.154.3 Function Documentation

10.154.3.1 int main (void)

Here is the call graph for this function:



10.154.4 Variable Documentation

10.154.4.1 double matrix_A[9] = { 1, 2, 3, 4, 5, 6, 7, 8, 9}

10.154.4.2 double matrix_AT[9] = { 1, 4, 7, 2, 5, 8, 3, 6, 9}

10.155 check_tridiag.c File Reference

Functions

- int main (int argc, char ∗argv[ ])

10.155.1 Detailed Description

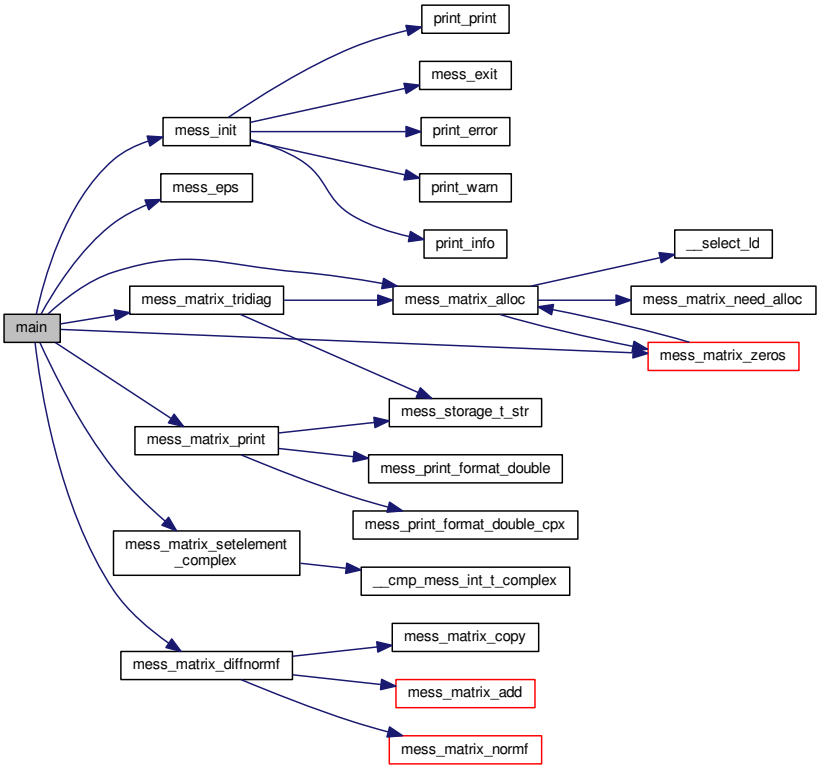
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.155.2 Function Documentation

10.155.2.1 int main (int *argc*, char ∗ *argv[ ]*)

Here is the call graph for this function:



10.156 check_trisolve.c File Reference

Macros

- #define CHECK_TRISOLVE(SOLVER, L, SOL, B, STORETYPE, L2, X, ERR, EPS, DIFF)
- #define CHECK_ALL_STORAGETYPE(MESS_DIRECT, SOLVER, OP, L, SOL, B, L2, X, ERR, EPS, DIFF)
- #define CHECK_TRISOLVEM(SOLVER, L, SOL, B, STORETYPE, L2, X, ERR, EPS, DIFF)
- #define CHECK_ALL_STORAGETYPEM(MESS_DIRECT, SOLVER, OP, L, SOL, B, L2, X, ERR, EPS, DIFF)

Functions

- int main (int argc, char ∗∗argv)

10.156.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

- MESS_OP_NONE ($op(C) = C$),
- MESS_OP_TRANSPOSE ($op(C) = C^T$),
- MESS_OP_HERMITIAN ($op(C) = C^H$).

10.156.2 Macro Definition Documentation

10.156.2.1 #define CHECK_ALL_STORAGETYPE(*MESS_DIRECT, SOLVER, OP, L, SOL,* B*, L2,* X*, ERR, EPS,* DIFF)

Value:

```
CALL(mess_direct_init(&MESS_DIRECT));
      \
        CALL(mess_direct_create_lapack_lu(L,MESS_DIRECT));
                                  \
        CALL(mess_direct_solve(OP,MESS_DIRECT,B,SOL));
         \
        CHECK_TRISOLVE(SOLVER,L,SOL,B,MESS_DENSE,L2,
      X,ERR,EPS,DIFF)                \
        CHECK_TRISOLVE(SOLVER,L,SOL,B,MESS_CSC,L2,
      X,ERR,EPS,DIFF)                  \
        CHECK_TRISOLVE(SOLVER,L,SOL,B,MESS_CSR,L2,
      X,ERR,EPS,DIFF)                  \
        CALL(mess_direct_clear(&MESS_DIRECT));
        \
```

10.156.2.2 #define CHECK_ALL_STORAGETYPEM(*MESS_DIRECT, SOLVER, OP, L, SOL,* B*, L2,* X*, ERR, EPS,* DIFF)

Value:

```
CALL(mess_direct_init(&MESS_DIRECT));
      \
        CALL(mess_direct_create_lapack_lu(L,MESS_DIRECT));
                                  \
        CALL(mess_direct_solvem(OP,MESS_DIRECT,B,SOL));
          \
        CHECK_TRISOLVEM(SOLVER,L,SOL,B,MESS_DENSE,L2,
      X,ERR,EPS,DIFF)                \
        CHECK_TRISOLVEM(SOLVER,L,SOL,B,MESS_CSC,L2,
      X,ERR,EPS,DIFF)                  \
        CHECK_TRISOLVEM(SOLVER,L,SOL,B,MESS_CSR,L2,
      X,ERR,EPS,DIFF)                  \
        CALL(mess_direct_clear(&MESS_DIRECT));
        \
```

10.156.2.3 #define CHECK_TRISOLVE(*SOLVER, L, SOL,* B, *STORETYPE, L2,* X, *ERR, EPS,* DIFF)

Value:

```
{          \
    CALL(mess_matrix_convert((L),(L2),(STORETYPE)));
      \
    CALL(mess_vector_copy((B),(X)));
      \
    CALL(SOLVER((L2),(X)));
      \
    CALL(mess_vector_diffnorm((X),(SOL),&(DIFF)));
        \
        if(DIFF>EPS){                                                    \
            printf("Failed:\n");                                            \
            printf("Solver: %s\n",#SOLVER);                                 \
            printf("Storetype: %s\n", #STORETYPE);                          \
            printf("L2: %s\n", #L2);                                        \
            CALL(mess_matrix_printinfo(L2));
            \
            printf("RHS: %s\n",#B);                                         \
            CALL(mess_vector_printinfo(B));
             \
            printf("Diff: %e\n",DIFF);                                      \
            ++ERR;                                                          \
        }                                                                   \
    /*printf("diff=%e\n",diff);*/                                           \
    /*mess_matrix_printinfo(L2);*/                                          \
    /*mess_vector_printinfo(x);*/                                           \
}
```

10.156.2.4 #define CHECK_TRISOLVEM(*SOLVER, L, SOL,* B, *STORETYPE, L2,* X, *ERR, EPS,* DIFF)

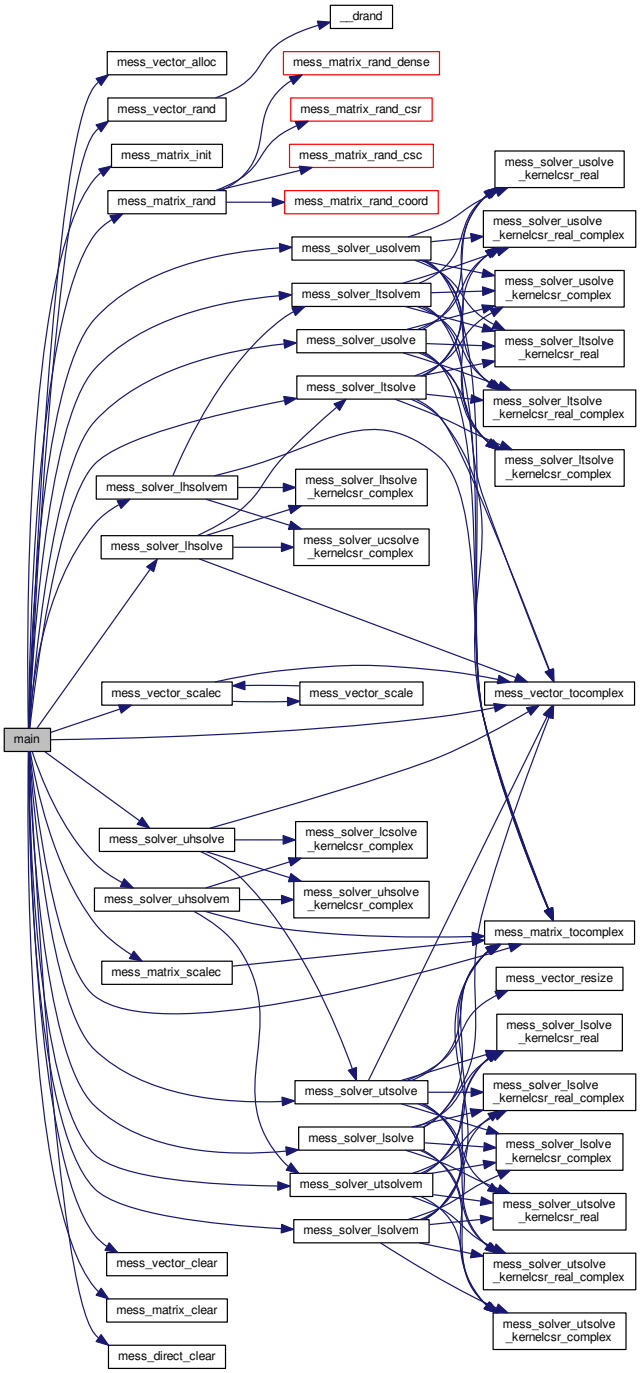
Value:

```
{          \
    CALL(mess_matrix_convert((L),(L2),(STORETYPE)));
      \
    CALL(mess_matrix_copy((B),(X)));
      \
    CALL(SOLVER((L2),(X)));
      \
    CALL(mess_matrix_diffnorm((X),(SOL),&(DIFF)));
            \
        if(DIFF>EPS){                                                    \
            printf("Failed:\n");                                            \
            printf("Solver: %s\n",#SOLVER);                                 \
            printf("Storetype: %s\n", #STORETYPE);                          \
            printf("L2: %s\n", #L2);                                        \
            CALL(mess_matrix_printinfo(L2));
            \
            printf("RHS: %s\n",#B);                                         \
            CALL(mess_matrix_printinfo(B));
             \
            printf("Diff: %e\n",DIFF);                                      \
            ++ERR;                                                          \
        }                                                                   \
    /*printf("diff=%e\n",diff);*/                                           \
    /*mess_matrix_printinfo(L2);*/                                          \
    /*mess_vector_printinfo(x);*/                                           \
}
```

10.156.3 Function Documentation

10.156.3.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.157 check_vector_perm.c File Reference

Functions

- mess_int_t * permute (mess_int_t dim)

- int check_perm (mess_vector v, double eps)

  Checks correctness of mess_vector_perm and mess_vector_iperm.
- int check_perm_inplace (mess_vector v, double eps)

  Checks correctness of mess_vector_perm_inplace and mess_vector_iperm_inplace.
- int check_perm_split (mess_vector v, double eps)

  Checks the correctness of mess_vector_perm_split and mess_vector_iperm_split.
- int check_perm_combine (mess_vector v, double eps)

  Checks the correctness of mess_vector_perm_combine and mess_vector_iperm_combine.
- int main ()

10.157.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

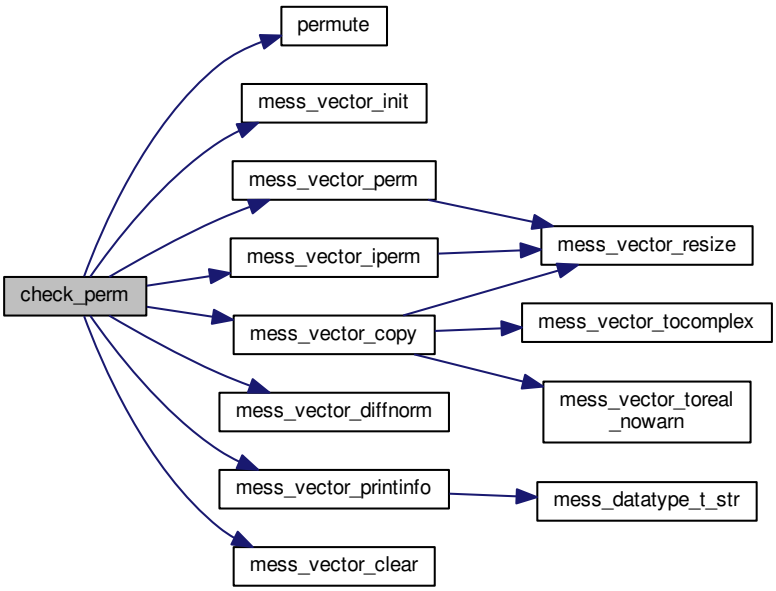
10.157.2 Function Documentation

10.157.2.1 int check_perm (mess_vector *v*, double *eps*)

Attention

Internal use only.

Here is the call graph for this function:

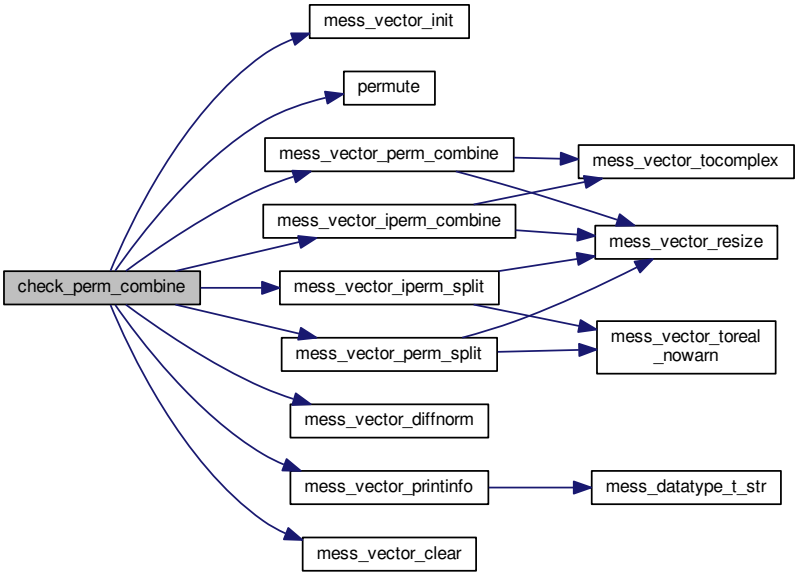


10.157.2.2 int check_perm_combine (mess_vector *v*, double *eps*)

Attention

Internal use only.

Here is the call graph for this function:

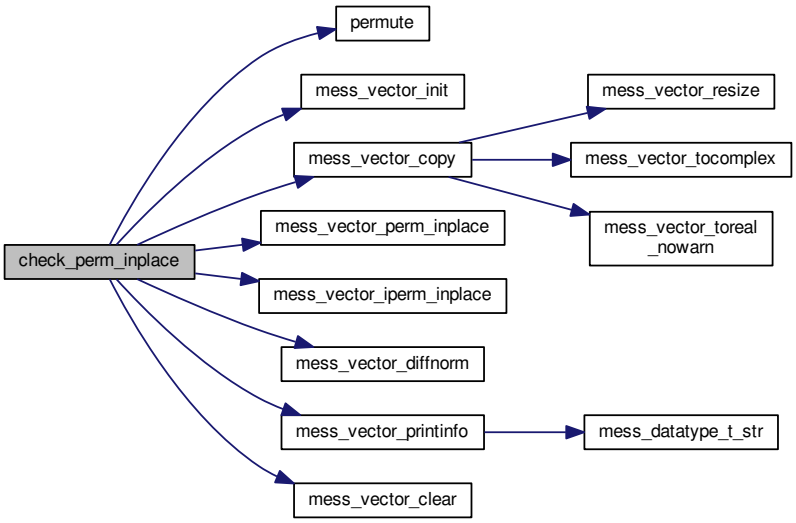


10.157.2.3 int check_perm_inplace (mess_vector *v*, double *eps*)

Attention

Internal use only.

Here is the call graph for this function:

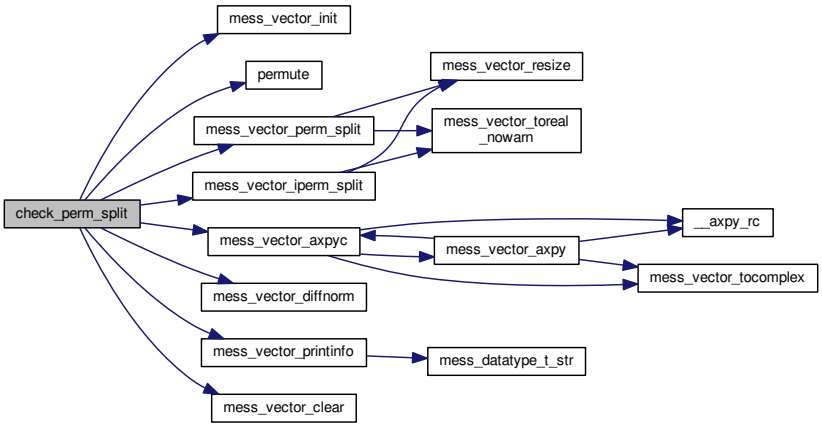


10.157.2.4 int check_perm_split (mess_vector *v*, double *eps*)

Attention

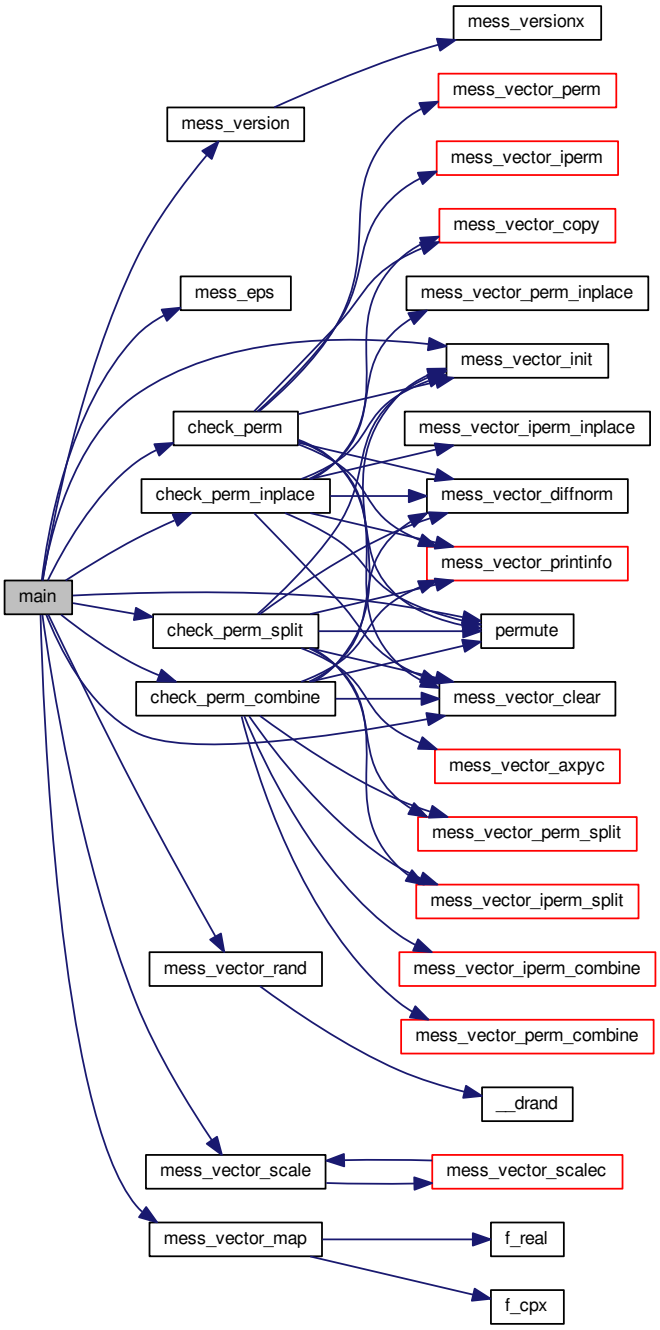
Internal use only.

Here is the call graph for this function:



10.157.2.5 int main (void)

Here is the call graph for this function:



10.157.2.6 mess_int_t* permute (mess_int_t *dim*)

10.158 check_vectorio.c File Reference

Functions

- int main (int argc, char **argv)

10.158.1 Detailed Description

Author

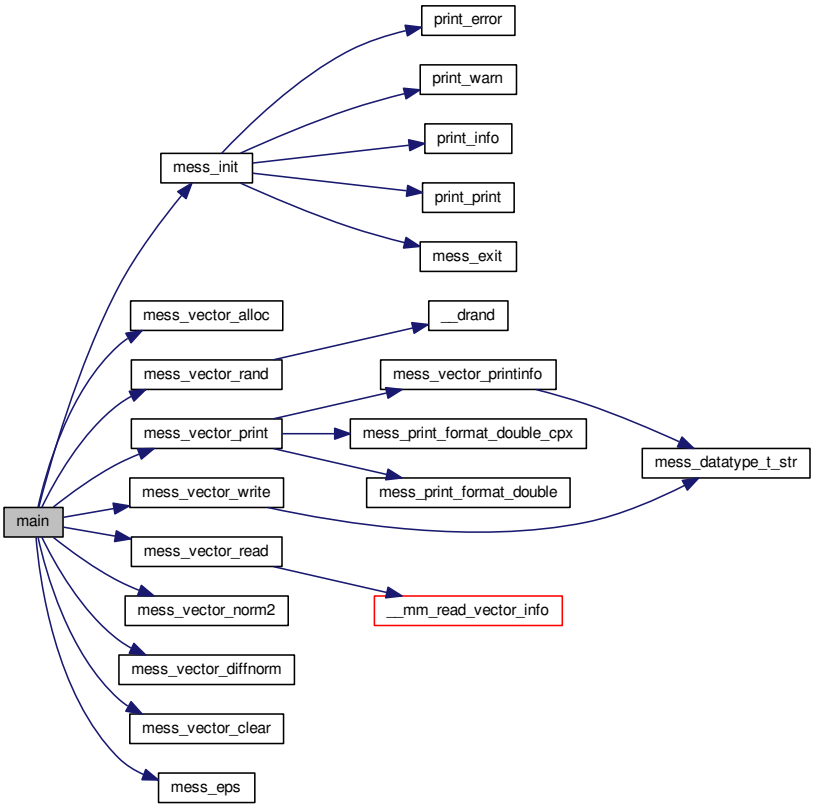
Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_vector_read and mess_vector_write functions defined in vector_read.c that means it checks if the reading and writing functions are working for vectors.

10.158.2 Function Documentation

10.158.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.159 cholesky.c File Reference

Data Structures

- struct cholesky_solver

  Internal structure for the LAPACK *based solver.*

Functions

- static int chol_clear (void ∗data)

  Clear the cholesky solver.
- static int chol_getL (void ∗data, mess_matrix L)

  Get the L factor of the cholesky solver ($A = LL^T$)
- static int chol_getU (void ∗data, mess_matrix U)

  Get the U factor (second factor) of the cholesky solver ($A = LL^T$)
- static int chol_solve (void ∗data, mess_vector b, mess_vector x)

  Solve $Ax = b$ using `LAPACK`.
- static int chol_solvem (void ∗data, mess_matrix b, mess_matrix x)

  Solve $AX = B$ using `LAPACK` *(matrix version)*
- static int chol_inverse (void ∗data, mess_matrix inv)

  Compute the inverse of a matrix using `LAPACK`.
- int mess_direct_create_cholesky (mess_matrix matrix, mess_direct solver)

  Generate a dense `LAPACK` *based Cholesky solver.*

10.159.1 Detailed Description

Author

Martin Köhler (`koehlerm@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)
Maximilian Behr (`behr@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.159.2 Function Documentation

10.159.2.1 static int chol_clear (void ∗ *data*) [static]

Parameters

in,out	*data*	pointer to the internal data

Returns

always zero

The chol_clear function clears a `LAPACK` sovler.

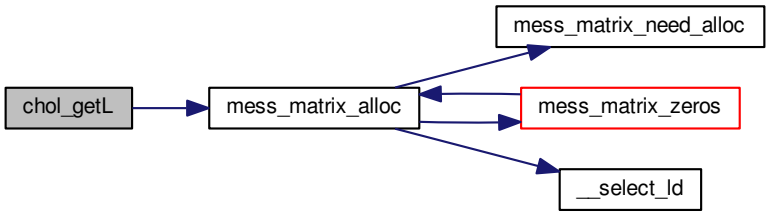
10.159.2.2 static int chol_getL (void ∗ *data*, mess_matrix *L*) [static]

Parameters

in	*data*	input pointer to the internal data
in,out	*L*	the matrix L

The chol_getL function returns the L factor such that $A = LL^T$.

Here is the call graph for this function:



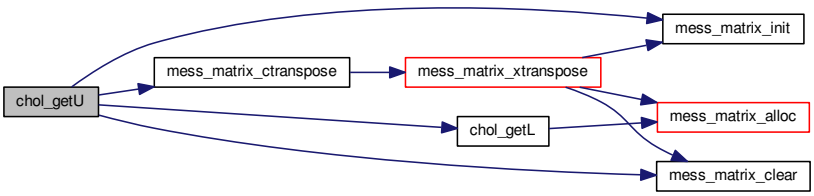
10.159.2.3 static int chol_getU (void * *data*, mess_matrix *U*) [static]

Parameters

in	*data*	input pointer to the internal data
out	*U*	output matrix contains L^T

The chol_getU function sets U to L^T.

Here is the call graph for this function:



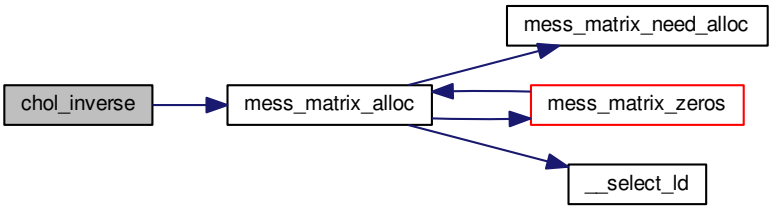
10.159.2.4 static int chol_inverse (void * *data*, mess_matrix *inv*) [static]

Parameters

in	*data*	input pointer to the internal data
in,out	*inv*	output inverse

The lapack_inverse function computes A^{-1} using LAPACK.

Here is the call graph for this function:



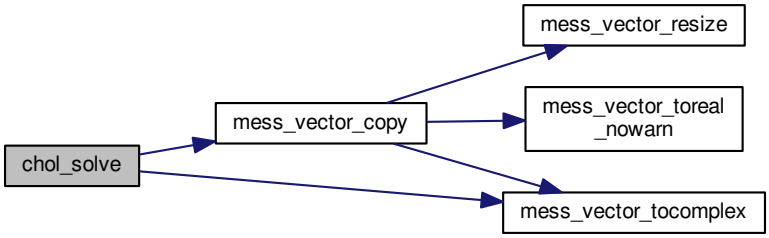
10.159.2.5 static int chol_solve (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input pointer to the internal data structure
in	*b*	input right hand side
in,out	*x*	solution

The chol_solve function solves $Ax = b$ using LAPACK.

Here is the call graph for this function:



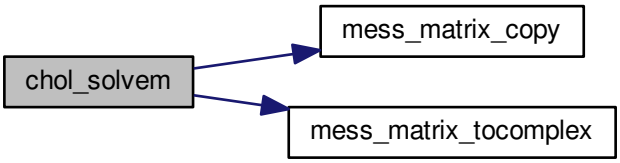
10.159.2.6 static int chol_solvem (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input pointer to the internal data structure
in	*b*	input right hand side
in,out	*x*	solution

The chol_solvem function solves $AX = B$ using LAPACK, where B and X are matrices.

Here is the call graph for this function:



10.160 cholfactor.c File Reference

Functions

- int mess_direct_cholfactor (mess_matrix ZZ, mess_matrix Z)

  Compute a Cholesky factor of a given matrix.

10.160.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.161 cholmod.c File Reference

10.161.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.162 cholmod_chol.c File Reference

10.162.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file provide an interface for CHOLMOD. You need a working SuiteSparse [2] to use it.

10.163 cimport.c File Reference

Functions

- int mess_matrix_dense_from_farray (mess_matrix mat, mess_int_t rows, mess_int_t cols, mess_int_t ld, double ∗realv, mess_double_cpx_t ∗complexv)

  Create a MESS_DENSE mess_matrix from a `Fortran` array.

- int mess_matrix_dense_from_carray (mess_matrix mat, mess_int_t rows, mess_int_t cols, double ∗∗realv, mess_double_cpx_t ∗∗complexv)

  Create a MESS_DENSE mess_matrix from a `C` 2-dimensional array.

- int mess_matrix_csr (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t ∗rowptr, mess_int↩_t ∗colptr, double ∗values, mess_double_cpx_t ∗values_cpx)

  Create a MESS_CSR mess_matrix from given arrays.

- int mess_matrix_csc (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t ∗rowptr, mess_int↩_t ∗colptr, double ∗values, mess_double_cpx_t ∗values_cpx)

  Create a MESS_CSC mess_matrix from given arrays.

- int mess_matrix_coord (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t nnz, mess_int_t onebased, mess_int_t ∗rowptr, mess_int_t ∗colptr, double ∗values, mess_double_cpx_t ∗values_cpx)

  Create a MESS_COORD mess_matrix from given arrays.

10.163.1 Detailed Description

Author

Martin Köhler (`koehlerm@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)
Maximilian Behr (`behr@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.164 CMakeCache.txt File Reference

Variables

- AMD_INCLUDE_DIR __pad0__
- usr lib x86_64 linux gnu liblzma so
- general
- m
- lpthread
- libcscutils_BINARY_DIR __pad1__
- lgfortran
- cscutils
- AMD_INCLUDE_DIR ADVANCED

10.164.1 Variable Documentation

10.164.1.1 AMD_INCLUDE_DIR __pad0__

10.164.1.2 libcscutils_BINARY_DIR __pad1__

10.164.1.3 AMD_INCLUDE_DIR ADVANCED

10.164.1.4 cscutils

10.164.1.5 general

10.164.1.6 lgfortran

10.164.1.7 lpthread

10.164.1.8 void char mess_int_t * m

10.164.1.9 usr lib x86_64 linux gnu libcholmod so

10.165 CMakeCache.txt File Reference

Variables

- CMAKE_BUILD_TYPE __pad2__
- myfort

10.165.1 Variable Documentation

10.165.1.1 CMAKE_BUILD_TYPE __pad2__

10.165.1.2 myfort

10.166 CMakeCache.txt File Reference

10.167 CMakeCache.txt File Reference

Variables

- CMAKE_BUILD_TYPE __pad5__
- myfort

10.167.1 Variable Documentation

10.167.1.1 CMAKE_BUILD_TYPE __pad5__

10.167.1.2 myfort

10.168 CMakeCCompilerId.c File Reference

Macros

- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID ""
- #define ARCHITECTURE_ID ""
- #define DEC(n)
- #define HEX(n)

Functions

- int main (int argc, char *argv[ ])

Variables

- char const * info_compiler = "INFO" ":" "compiler[" "" "]"
- char const * info_platform = "INFO" ":" "platform[" "" "]"
- char const * info_arch = "INFO" ":" "arch[" "" "]"
- const char * info_language_dialect_default

10.168.1 Macro Definition Documentation

10.168.1.1 #define ARCHITECTURE_ID ""

10.168.1.2 #define COMPILER_ID ""

10.168.1.3 #define DEC(n)

Value:

```
('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

10.168.1.4 #define HEX(n)

Value:

```
('0' + ((n)>>28 & 0xF)), \
  ('0' + ((n)>>24 & 0xF)), \
  ('0' + ((n)>>20 & 0xF)), \
  ('0' + ((n)>>16 & 0xF)), \
  ('0' + ((n)>>12 & 0xF)), \
  ('0' + ((n)>>8  & 0xF)), \
  ('0' + ((n)>>4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

10.168.1.5 #define PLATFORM_ID ""

10.168.1.6 #define STRINGIFY(X) STRINGIFY_HELPER(X)

10.168.1.7 #define STRINGIFY_HELPER(X) #X

10.168.2 Function Documentation

10.168.2.1 int main (int *argc*, char ∗ *argv[ ]*)

10.168.3 Variable Documentation

10.168.3.1 char const∗ info_arch = "INFO" ":" "arch[" "" "]"

10.168.3.2 char const∗ info_compiler = "INFO" ":" "compiler[" "" "]"

10.168.3.3 const char∗ info_language_dialect_default

Initial value:

```
= "INFO" ":" "dialect_default["

  "90"




"]"
```

10.168.3.4 char const∗ info_platform = "INFO" ":" "platform[" "" "]"

10.169 CMakeCXXCompilerId.cpp File Reference

Macros

- #define COMPILER_ID ""
- #define STRINGIFY_HELPER(X) #X
- #define STRINGIFY(X) STRINGIFY_HELPER(X)
- #define PLATFORM_ID ""
- #define ARCHITECTURE_ID ""
- #define DEC(n)
- #define HEX(n)

Functions

- int main (int argc, char *argv[ ])

Variables

- char const * info_compiler = "INFO" ":" "compiler[" "" "]"
- char const * info_platform = "INFO" ":" "platform[" "" "]"
- char const * info_arch = "INFO" ":" "arch[" "" "]"
- const char * info_language_dialect_default

10.169.1 Macro Definition Documentation

10.169.1.1 #define ARCHITECTURE_ID ""

10.169.1.2 #define COMPILER_ID ""

10.169.1.3 #define DEC(n)

Value:

```
('0' + (((n) / 10000000)%10)), \
  ('0' + (((n) / 1000000)%10)),  \
  ('0' + (((n) / 100000)%10)),   \
  ('0' + (((n) / 10000)%10)),    \
  ('0' + (((n) / 1000)%10)),     \
  ('0' + (((n) / 100)%10)),      \
  ('0' + (((n) / 10)%10)),       \
  ('0' +  ((n) % 10))
```

10.169.1.4 #define HEX(n)

Value:

```
('0' + ((n)>>28 & 0xF)), \
  ('0' + ((n)>>24 & 0xF)), \
  ('0' + ((n)>>20 & 0xF)), \
  ('0' + ((n)>>16 & 0xF)), \
  ('0' + ((n)>>12 & 0xF)), \
  ('0' + ((n)>>8  & 0xF)), \
  ('0' + ((n)>>4  & 0xF)), \
  ('0' + ((n)     & 0xF))
```

10.169.1.5 #define PLATFORM_ID ""

10.169.1.6 #define STRINGIFY(X) STRINGIFY_HELPER(X)

10.169.1.7 #define STRINGIFY_HELPER(X) #X

10.169.2 Function Documentation

10.169.2.1 int main (int *argc*, char ∗ *argv[ ]*)

10.169.3 Variable Documentation

10.169.3.1 char const∗ info_arch = "INFO" ":" "arch[" "" "]"

10.169.3.2 char const∗ info_compiler = "INFO" ":" "compiler[" "" "]"

10.169.3.3 const char∗ info_language_dialect_default

Initial value:

```
= "INFO" ":" "dialect_default["



  "98"
"]"
```

10.169.3.4 char const∗ info_platform = "INFO" ":" "platform[" "" "]"

10.170 CMakeFortranCompilerId.F File Reference

Macros

- #define DEC_8(n) (((n) / 10000000) % 10)
- #define DEC_7(n) (((n) / 1000000) % 10)
- #define DEC_6(n) (((n) / 100000) % 10)
- #define DEC_5(n) (((n) / 10000) % 10)
- #define DEC_4(n) (((n) / 1000) % 10)
- #define DEC_3(n) (((n) / 100) % 10)
- #define DEC_2(n) (((n) / 10) % 10)
- #define DEC_1(n) (((n)) % 10)
- #define HEX_8(n) ((n)>>28 & 0xF)
- #define HEX_7(n) ((n)>>24 & 0xF)
- #define HEX_6(n) ((n)>>20 & 0xF)
- #define HEX_5(n) ((n)>>16 & 0xF)
- #define HEX_4(n) ((n)>>12 & 0xF)
- #define HEX_3(n) ((n)>>8 & 0xF)
- #define HEX_2(n) ((n)>>4 & 0xF)
- #define HEX_1(n) ((n) & 0xF)

Functions/Subroutines

- program cmakefortrancompilerid

10.170.1 Macro Definition Documentation

10.170.1.1 #define DEC_1(n) (((n)) % 10)

10.170.1.2 #define DEC_2(n) (((n) / 10) % 10)

10.170.1.3 #define DEC_3(n) (((n) / 100) % 10)

10.170.1.4 #define DEC_4(n) (((n) / 1000) % 10)

10.170.1.5 #define DEC_5(n) (((n) / 10000) % 10)

10.170.1.6 #define DEC_6(n) (((n) / 100000) % 10)

10.170.1.7 #define DEC_7(n) (((n) / 1000000) % 10)

10.170.1.8 #define DEC_8(n) (((n) / 10000000) % 10)

10.170.1.9 #define HEX_1(n) ((n) & 0xF)

10.170.1.10 #define HEX_2(n) ((n)>>4 & 0xF)

10.170.1.11 #define HEX_3(n) ((n)>>8 & 0xF)

10.170.1.12 #define HEX_4(n) ((n)>>12 & 0xF)

10.170.1.13 #define HEX_5(n) ((n)>>16 & 0xF)

10.170.1.14 #define HEX_6(n) ((n)>>20 & 0xF)

10.170.1.15 #define HEX_7(n) ((n)>>24 & 0xF)

10.170.1.16 #define HEX_8(n) ((n)>>28 & 0xF)

10.170.2 Function/Subroutine Documentation

10.170.2.1 program cmakefortrancompilerid ( )

10.171 CMakeRuleHashes.txt File Reference

10.172 col_compress.c File Reference

Functions

- int mess_lrcfadi_colcompress (mess_matrix Z, double ccTol)

  Perform a column compression on a low rank factor.
- int mess_lrcfadi_ccsvd (mess_matrix Z, double ccTol)

  Perform a SVD based column compression on low rank factor.

10.172.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.173 colops.c File Reference

Functions

- int mess_matrix_colnorm (mess_matrix Q, mess_int_t col, double *norm)

  Compute the euclidean norm of a matrix column.
- int mess_matrix_colscale (mess_matrix Q, mess_int_t col, mess_double_cpx_t scale)

  Scale a column of a matrix.
- int mess_matrix_coldot (mess_matrix Q, mess_int_t col1, mess_int_t col2, double *dot)

  Compute the dot product between two columns of a real matrix.
- int mess_matrix_coldotc (mess_matrix Q, mess_int_t col1, mess_int_t col2, mess_double_cpx_t *dot)

  Compute the dot product between two columns of a complex matrix.
- int mess_matrix_colaxpy2 (mess_matrix Q, mess_double_cpx_t alpha, mess_int_t colc, mess_int_t col1)

  Update a column of a matrix by α times another column.
- int mess_matrix_colvecdot (mess_matrix Q, mess_int_t col, mess_vector v, double *dot)

  Compute the dot product between a matrix column and a vector (real).
- int mess_matrix_colvecdotc (mess_matrix Q, mess_int_t col, mess_vector v, mess_double_cpx_t *dot)

  Computes the dot product between a matrix column and a vector (complex).
- int mess_matrix_colvecaxpy (mess_double_cpx_t alpha, mess_int_t col, mess_matrix Q, mess_vector v)

  Update a vector using a column of a matrix.
- int mess_matrix_colaxpy (mess_double_cpx_t alpha, mess_vector v, mess_int_t col, mess_matrix Q)

  Perform an axpy-update on a column of a matrix.
- int mess_matrix_coldotE (mess_matrix Q, mess_matrix E, mess_int_t col1, mess_int_t col2, double *dot)

  Compute the weighted dot product of two matrix columns (real).
- int mess_matrix_coldotcE (mess_matrix Q, mess_matrix E, mess_int_t col1, mess_int_t col2, mess_double↩_cpx_t *dot)

  Compute the weighted dot product of two matrix columns (complex).
- int mess_matrix_colnormE (mess_matrix Q, mess_matrix E, mess_int_t col, double *norm)

  Compute the E norm of a matrix column.

10.173.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.174 colrowsums.c File Reference

Functions

- int mess_matrix_rowsums (mess_matrix A, mess_vector x)

  Compute the row sums of a matrix.
- int mess_matrix_colsums (mess_matrix A, mess_vector x)

  Compute the column sums of a matrix.

10.174.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.175 condest.c File Reference

Macros

- #define INFINITY HUGE_VAL

Functions

- int mess_matrix_norminvest (mess_direct decomp, double *nrm)

  Estimate 1-norm of the inverse of a real square matrix.
- int mess_matrix_condest (mess_matrix A, double *nrm)

  Compute a lower bound for the 1-norm condition number of a real square matrix.

10.175.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.175.2 Macro Definition Documentation

10.175.2.1 #define INFINITY HUGE_VAL

10.176 config.h File Reference

Macros

- #define MESS_VERSION "1.0.0"
- #define MESS_MAJOR_VERSION 1
- #define MESS_MINOR_VERSION 0
- #define MESS_PATCH_VERSION 0
- #define MESS_GIT_BRANCH "master"
- #define MESS_GIT_COMMIT_HASH "4e991918bf8e700b7cec1d0cae7fd9c604375892"
- #define MESS_CMAKE_VERSION "3.5.1"
- #define MESS_CMAKE_C_COMPILER "/usr/bin/cc (GNU) 5.4.0"
- #define MESS_CMAKE_CXX_COMPILER "/usr/bin/c++ (GNU) 5.4.0"
- #define MESS_CMAKE_Fortran_COMPILER "/usr/bin/gfortran (GNU) 5.4.0"
- #define MESS_CMAKE_HOST_SYSTEM "Linux-4.4.0-89-generic x86_64"
- #define MESS_BLA_VENDOR "Generic"
- #define MESS_BLAS_LIBRARIES "/usr/lib/libblas.so"
- #define MESS_LAPACK_LIBRARIES "/usr/lib/liblapack.so;/usr/lib/libblas.so"

- #define MESS_LIBS "/usr/lib/x86_64-linux-gnu/libm.so;-lpthread;-lgfortran;/usr/lib/liblapack.so;/usr/lib/libblas.↩ so;/usr/lib/x86_64-linux-gnu/libamd.so;/usr/lib/x86_64-linux-gnu/libcolamd.so;/usr/lib/x86_64-linux-gnu/libumfpack.↩ so;/usr/lib/x86_64-linux-gnu/libcholmod.so;/usr/lib/x86_64-linux-gnu/libcxsparse.so;cscutils"
- #define MESS_INCLUDE_DIR "/home/koehlerm/mechthild_home/work/software/cmess-releases/build/include;/home/koehlerm _home/work/software/cmess-releases/include;/usr/include/suitesparse;/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/libcscutils/include"
- #define MESS_SUITESPARSE_LIBRARIES "/usr/lib/x86_64-linux-gnu/libamd.so;/usr/lib/x86_64-linux-gnu/libcolamd.so;/usr/lib/x86_64-linux-gnu/libumfpack.so;/usr/lib/x86_64-linux-gnu/libcholmod.so;/usr/lib/x86↩ _64-linux-gnu/libcxsparse.so"
- #define MESS_SUITESPARSE_INCLUDE_DIR "/usr/include/suitesparse;/usr/include/suitesparse;/usr/include/suitesparse;/usr
- #define MESS_SUPERLU_LIBRARIES ""
- #define MESS_SUPERLU_INCLUDE_DIR ""
- #define MESS_ARPACK_LIBRARIES ""
- #define MESS_LIBS "/usr/lib/x86_64-linux-gnu/libm.so;-lpthread;-lgfortran;/usr/lib/liblapack.so;/usr/lib/libblas.↩ so;/usr/lib/x86_64-linux-gnu/libamd.so;/usr/lib/x86_64-linux-gnu/libcolamd.so;/usr/lib/x86_64-linux-gnu/libumfpack.↩ so;/usr/lib/x86_64-linux-gnu/libcholmod.so;/usr/lib/x86_64-linux-gnu/libcxsparse.so;cscutils"
- #define MESS_INCLUDE_DIR "/home/koehlerm/mechthild_home/work/software/cmess-releases/build/include;/home/koehlerm _home/work/software/cmess-releases/include;/usr/include/suitesparse;/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/libcscutils/include"
- #define MESS_CMAKE_C_FLAGS "-g3 -fPIC -Wall -D_FILE_OFFSET_BITS=64 -D_LARGEFILE64_SOU↩ RCE"
- #define MESS_CMAKE_C_FLAGS_DEBUG "-g -g3 -fPIC -Wall -D_FILE_OFFSET_BITS=64 -D_LARGEF↩ ILE64_SOURCE"
- #define MESS_CMAKE_C_FLAGS_RELEASE "-O2 -DNDEBUG"
- #define MESS_CMAKE_CXX_FLAGS "-g3 -fPIC -Wall -D_FILE_OFFSET_BITS=64 -D_LARGEFILE64_S↩ OURCE"
- #define MESS_CMAKE_CXX_FLAGS_DEBUG "-g -g3 -fPIC -Wall -D_FILE_OFFSET_BITS=64 -D_LARG↩ EFILE64_SOURCE"
- #define MESS_CMAKE_CXX_FLAGS_RELEASE "-O2 -DNDEBUG"
- #define MESS_CMAKE_Fortran_FLAGS "-g3 -fPIC -Wall -D_FILE_OFFSET_BITS=64 -D_LARGEFILE64↩ _SOURCE"
- #define MESS_CMAKE_Fortran_FLAGS_DEBUG "-g -g3 -fPIC -Wall -D_FILE_OFFSET_BITS=64 -D_LA↩ RGEFILE64_SOURCE"
- #define MESS_CMAKE_Fortran_FLAGS_RELEASE "-O2"
- #define MESS_CMAKE_BUILD_TYPE "Release"
- #define MESS_CONFIG_TIME "2018-07-13 07:36:07"
- #define MESS_HAVE_SIZE_T
- #define MESS_HAVE_PTRDIFF_T
- #define MESS_HAVE_STRNCMP 1
- #define MESS_HAVE_ZLIB
- #define MESS_HAVE_BZIP2
- #define MESS_HAVE_UMFPACK
- #define MESS_HAVE_AMD
- #define MESS_HAVE_COLAMD
- #define MESS_HAVE_CHOLMOD
- #define MESS_HAVE_CSPARSE
- #define MESS_HAVE_XPM
- #define MESS_HAVE_CLOCK_GETTIME
- #define MESS_USE_OPENBLAS
- #define MESS_HAVE_STRNDUP
- #define MESS_HAVE_NEARBYINT
- #define MESS_HAVE_ISFINITE
- #define LD_MULTIPLICITY 1
- #define LOGLEVEL 1

10.176.1 Macro Definition Documentation

10.176.1.1 #define LD_MULTIPLICITY 1

10.176.1.2 #define LOGLEVEL 1

10.176.1.3 #define MESS_ARPACK_LIBRARIES ""

10.176.1.4 #define MESS_BLA_VENDOR "Generic"

10.176.1.5 #define MESS_BLAS_LIBRARIES "/usr/lib/libblas.so"

10.176.1.6 #define MESS_CMAKE_BUILD_TYPE "Release"

10.176.1.7 #define MESS_CMAKE_C_COMPILER "/usr/bin/cc (GNU) 5.4.0"

10.176.1.8 #define MESS_CMAKE_C_FLAGS "-g3 -fPIC -Wall -D_FILE_OFFSET_BITS=64 -D_LARGEFILE64_SOURCE"

10.176.1.9 #define MESS_CMAKE_C_FLAGS_DEBUG "-g -g3 -fPIC -Wall -D_FILE_OFFSET_BITS=64 -D_LARGEFILE64_SOURCE"

10.176.1.10 #define MESS_CMAKE_C_FLAGS_RELEASE "-O2 -DNDEBUG"

10.176.1.11 #define MESS_CMAKE_CXX_COMPILER "/usr/bin/c++ (GNU) 5.4.0"

10.176.1.12 #define MESS_CMAKE_CXX_FLAGS "-g3 -fPIC -Wall -D_FILE_OFFSET_BITS=64 -D_LARGEFILE64_SOURCE"

10.176.1.13 #define MESS_CMAKE_CXX_FLAGS_DEBUG "-g -g3 -fPIC -Wall -D_FILE_OFFSET_BITS=64 -D_LARGEFILE64_SOURCE"

10.176.1.14 #define MESS_CMAKE_CXX_FLAGS_RELEASE "-O2 -DNDEBUG"

10.176.1.15 #define MESS_CMAKE_Fortran_COMPILER "/usr/bin/gfortran (GNU) 5.4.0"

10.176.1.16 #define MESS_CMAKE_Fortran_FLAGS "-g3 -fPIC -Wall -D_FILE_OFFSET_BITS=64 -D_LARGEFILE64_SOURCE"

10.176.1.17 #define MESS_CMAKE_Fortran_FLAGS_DEBUG "-g -g3 -fPIC -Wall -D_FILE_OFFSET_BITS=64 -D_LARGEFILE64_SOURCE"

10.176.1.18 #define MESS_CMAKE_Fortran_FLAGS_RELEASE "-O2"

10.176.1.19 #define MESS_CMAKE_HOST_SYSTEM "Linux-4.4.0-89-generic x86_64"

10.176.1.20 #define MESS_CMAKE_VERSION "3.5.1"

10.176.1.21 #define MESS_CONFIG_TIME "2018-07-13 07:36:07"

10.176.1.22 #define MESS_GIT_BRANCH "master"

10.176.1.23 #define MESS_GIT_COMMIT_HASH "4e991918bf8e700b7cec1d0cae7fd9c604375892"

10.176.1.24 #define MESS_HAVE_AMD

10.176.1.25 #define MESS_HAVE_BZIP2

10.176.1.26 #define MESS_HAVE_CHOLMOD

10.176.1.27 #define MESS_HAVE_CLOCK_GETTIME

10.176.1.28 #define MESS_HAVE_COLAMD

10.176.1.29 #define MESS_HAVE_CSPARSE

10.176.1.30 #define MESS_HAVE_ISFINITE

10.176.1.31 #define MESS_HAVE_NEARBYINT

10.176.1.32 #define MESS_HAVE_PTRDIFF_T

10.176.1.33 #define MESS_HAVE_SIZE_T

10.176.1.34 #define MESS_HAVE_STRNCMP 1

10.176.1.35 #define MESS_HAVE_STRNDUP

10.176.1.36 #define MESS_HAVE_UMFPACK

10.176.1.37 #define MESS_HAVE_XPM

10.176.1.38 #define MESS_HAVE_ZLIB

10.176.1.39 #define MESS_INCLUDE_DIR "/home/koehlerm/mechthild_home/work/software/cmess-releases/build/include;/home/koehlerm/mechthild_home/work/software/cmess-releases/include;/usr/include/suitesparse;/home/koehlerm/mechthild_home/work/software/cmess-releases/libcscutils/include"

10.176.1.40 #define MESS_INCLUDE_DIR "/home/koehlerm/mechthild_home/work/software/cmess-releases/build/include;/home/koehlerm/mechthild_home/work/software/cmess-releases/include;/usr/include/suitesparse;/home/koehlerm/mechthild_home/work/software/cmess-releases/libcscutils/include"

10.176.1.41 #define MESS_LAPACK_LIBRARIES "/usr/lib/liblapack.so;/usr/lib/libblas.so"

10.176.1.42 #define MESS_LIBS "/usr/lib/x86_64-linux-gnu/libm.so;-lpthread;-lgfortran;/usr/lib/liblapack.↩ so;/usr/lib/libblas.so;/usr/lib/x86_64-linux-gnu/libamd.so;/usr/lib/x86_64-linux-gnu/libcolamd.↩ so;/usr/lib/x86_64-linux-gnu/libumfpack.so;/usr/lib/x86_64-linux-gnu/libcholmod.so;/usr/lib/x86_64-linux-gnu/libcxsparse.so;cscutils"

10.176.1.43 #define MESS_LIBS "/usr/lib/x86_64-linux-gnu/libm.so;-lpthread;-lgfortran;/usr/lib/liblapack.so;/usr/lib/libblas.so;/usr/lib/x86_64-linux-gnu/libamd.so;/usr/lib/x86_64-linux-gnu/libcolamd.so;/usr/lib/x86_64-linux-gnu/libumfpack.so;/usr/lib/x86_64-linux-gnu/libcholmod.so;/usr/lib/x86_64-linux-gnu/libcxsparse.so;cscutils"

10.176.1.44 #define MESS_MAJOR_VERSION 1

10.176.1.45 #define MESS_MINOR_VERSION 0

10.176.1.46 #define MESS_PATCH_VERSION 0

10.176.1.47 #define MESS_SUITESPARSE_INCLUDE_DIR "/usr/include/suitesparse;/usr/include/suitesparse;/usr/include/suitesparse;/usr/include/suitesparse;/usr/include/suitesparse"

10.176.1.48 #define MESS_SUITESPARSE_LIBRARIES "/usr/lib/x86_64-linux-gnu/libamd.so;/usr/lib/x86_64-linux-gnu/libcolamd.so;/usr/lib/x86_64-linux-gnu/libumfpack.so;/usr/lib/x86_64-linux-gnu/libcholmod.so;/usr/lib/x86_64-linux-gnu/libcxsparse.so"

10.176.1.49 #define MESS_SUPERLU_INCLUDE_DIR ""

10.176.1.50 #define MESS_SUPERLU_LIBRARIES ""

10.176.1.51 #define MESS_USE_OPENBLAS

10.176.1.52 #define MESS_VERSION "1.0.0"

10.177 convert.c File Reference

Functions

- static int __conv_coord_csr (mess_matrix inmatrix, mess_matrix outmatrix)

  Convert a coordinate matrix to CSR.
- static int __conv_coord_csc (mess_matrix inmatrix, mess_matrix outmatrix)

  Convert a coordinate matrix to CSC.
- static int __conv_coord_dense (mess_matrix inmatrix, mess_matrix outmatrix)

  Convert a Coordinate Matrix to a Dense matrix.
- static int __conv_dense_csr (mess_matrix dense, mess_matrix csr)

  Convert a dense matrix to a Compressed Sparse Row matrix.
- static int __conv_dense_csc (mess_matrix dense, mess_matrix csc)

  Convert a Dense matrix to a Compressed Sparse Column matrix.
- static int __conv_csr_dense (mess_matrix csr, mess_matrix dense)

  Convert a Compressed Sparse Row matrix to a Dense matrix.
- static int __conv_csc_dense (mess_matrix csc, mess_matrix dense)

  Convert a Compressed Sparse Column matrix to a Dense matrix.
- static int __conv_csr_coord (mess_matrix csr, mess_matrix coord)

  Convert a Compressed Sparse Row matrix to a Coordinate matrix.
- static int __conv_csc_coord (mess_matrix csc, mess_matrix coord)

  Convert a Compressed Sparse Column matrix to a Coordinate matrix.
- static int __conv_dense_coord (mess_matrix dense, mess_matrix coord)

  Convert a Dense matrix to a Coordinate matrix.
- int mess_matrix_convert (mess_matrix input, mess_matrix output, mess_storage_t outtype)

  Convert the storage type of a matrix (universal converter interface).
- int mess_matrix_convert_csr_csc (mess_matrix inmatrix, mess_matrix outmatrix)

  Convert a Compressed Sparse Row matrix to a Compressed Sparse Column matrix.
- int mess_matrix_convert_csc_csr (mess_matrix inmatrix, mess_matrix outmatrix)

  Convert a Compressed Sparse Column matrix to a Compressed Sparse Row matrix.

10.177.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file provides the basic matrix converting routines and calls the user defined functions of a matrix if they are needed.

10.177.2 Function Documentation

10.177.2.1 static int __conv_coord_csc (mess_matrix *inmatrix*, mess_matrix *outmatrix*) [static]

Parameters

in	*inmatrix*	input coordinate matrix
out	*outmatrix*	output Compressed Sparse Column matrix

Returns

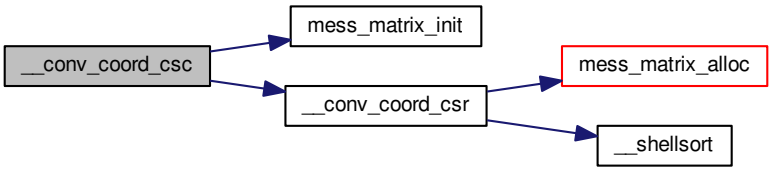
zero on success or a non zero error code otherwise

The __conv_coord_csc function converts a matrix from coordinate to Compressed Sparse Column storage. It uses the property that the CSC storage is the transpose of the CSR storage. In this way the matrix is transposed in the coordinate format and then converted to Compressed Sparse Row storage. Afterwards the data structure is modified to fit in a Compressed Sparse Column storage.

Attention

Internal use only.

Here is the call graph for this function:



10.177.2.2 static int __conv_coord_csr (mess_matrix *inmatrix*, mess_matrix *outmatrix*) [static]

Parameters

in	*inmatrix*	input coordinate matrix
out	*outmatrix*	output Compressed Sparse Row matrix

Returns

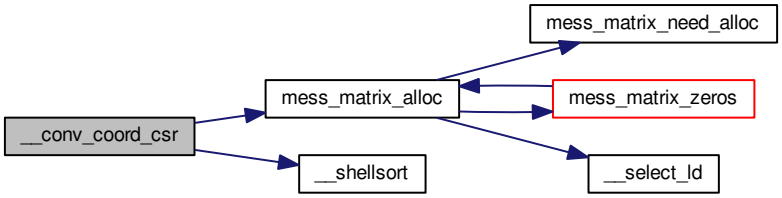
zero on success or a non zero error code otherwise

The __conv_coord_csr function converts a matrix from coordinate to Compressed Sparse Row storage.

Attention

Internal use only.

Here is the call graph for this function:



10.177.2.3 static int __conv_coord_dense (mess_matrix *inmatrix*, mess_matrix *outmatrix*) [static]

Parameters

in	*inmatrix*	input matrix
out	*outmatrix*	output matrix

Returns

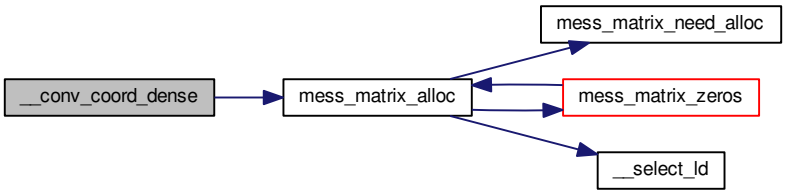
zero on success or a non zero error code otherwise

The __conv_coord_dense converts a Coordinate matrix to a Dense one.

Attention

Internal use only.

Here is the call graph for this function:



10.177.2.4 static int __conv_csc_coord (mess_matrix *csc*, mess_matrix *coord*) [static]

Parameters

in	*csc*	input matrix
out	*coord*	output matrix

Returns

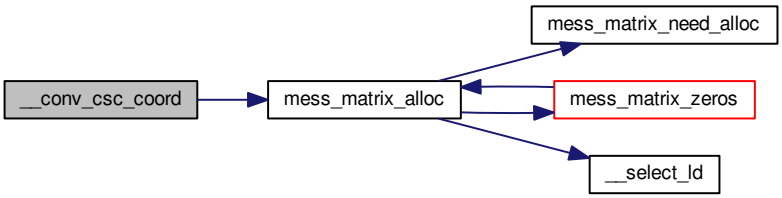
zero on success or a non zero error code otherwise

The __conv_csr_coord converts a Compressed Sparse Column matrix to a Coordinate one.

Attention

Internal use only.

Here is the call graph for this function:



10.177.2.5 static int __conv_csc_dense (mess_matrix *csc*, mess_matrix *dense*) [static]

Parameters

in	*csc*	input matrix
out	*dense*	output matrix

Returns

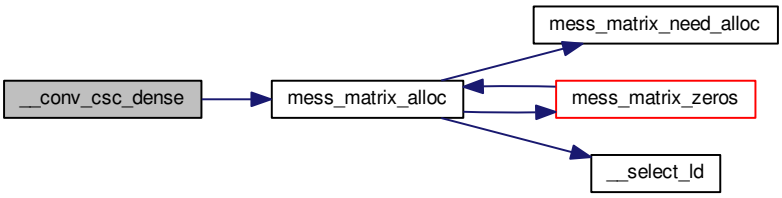
zero on success or a non zero error code otherwise

The __conv_csc_dense converts a Compressed Sparse Column matrix to a Dense one.

Attention

Internal use only.

Here is the call graph for this function:



10.177.2.6 static int __conv_csr_coord (mess_matrix *csr*, mess_matrix *coord*) [static]

Parameters

in	*csr*	input matrix
out	*coord*	output matrix

Returns

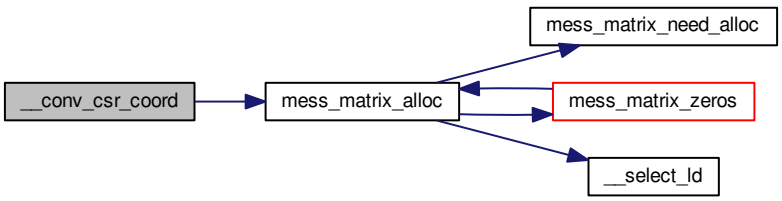
zero on success or a non zero error code otherwise

The __conv_csr_coord converts a Compressed Sparse Row matrix to a Coordinate one.

Attention

Internal use only.

Here is the call graph for this function:



10.177.2.7 static int __conv_csr_dense (mess_matrix *csr*, mess_matrix *dense*) [static]

Parameters

in	*csr*	input matrix
out	*dense*	output matrix

Returns

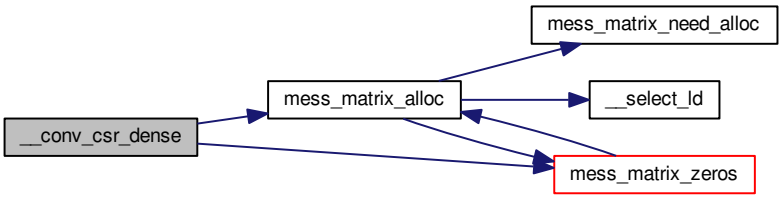
zero on success or a non zero error code otherwise

The __conv_csr_dense converts a Compressed Sparse Row matrix to a Dense one.

Attention

Internal use only.

Here is the call graph for this function:



10.177.2.8 static int __conv_dense_coord (mess_matrix *dense*, mess_matrix *coord*) [static]

Parameters

in	*dense*	input matrix
out	*coord*	output matrix

Returns

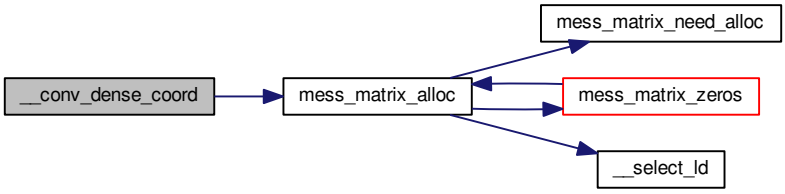
zero on success or a non zero error code otherwise

The __conv_dense_coord converts a Dense matrix to a Coordinate one.

Attention

Internal use only.

Here is the call graph for this function:



10.177.2.9 static int __conv_dense_csc (mess_matrix *dense*, mess_matrix *csc*) [static]

Parameters

in	*dense*	input matrix
out	*csc*	output matrix

Returns

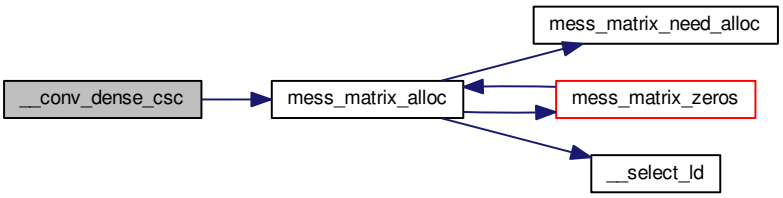
zero on success or a non zero error code otherwise

The __conv_dense_csc converts a Dense matrix matrix to a Compressed Sparse Column one.

Attention

Internal use only.

Here is the call graph for this function:



10.177.2.10 static int __conv_dense_csr (mess_matrix *dense*, mess_matrix *csr*) [static]

Parameters

in	*dense*	input matrix
out	*csr*	output matrix

Returns

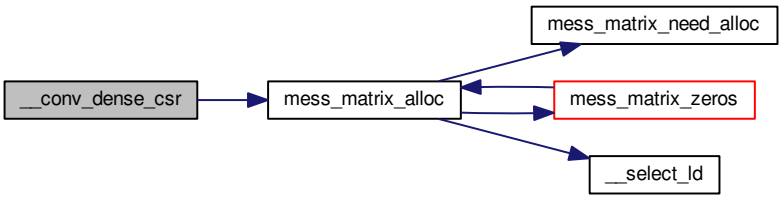
zero on success or a non zero error code otherwise

The __conv_dense_csr converts a Dense matrix matrix to a Compressed Sparse Row matrix.

Attention

Internal use only.

Here is the call graph for this function:



10.178 convert.c File Reference

Functions

- int mess_vector_from_farray (mess_vector v, mess_int_t dim, double *vals, mess_double_cpx_t *vals_cpx)

  Create a mess_vector from a `Fortran` array.
- int mess_vector_from_lapack (mess_vector v, mess_int_t dim, double *vals_re, double *vals_im)

  Create a mess_vector from two `Fortran` array.
- int mess_vector_convert_if_real (mess_vector v)

  Convert complex vector to real vector if imaginary part of every entry is small.
- int mess_vector_tomatrix (mess_vector v, mess_matrix mat)

  Convert a vector to a matrix.
- int mess_vector_frommatrix (mess_matrix mat, mess_vector v)

  Convert a single column matrix to a vector.
- int mess_vector_tocomplex (mess_vector v)

  Convert a vector to complex one.
- int mess_vector_toreal (mess_vector v)

  Convert a vector to real one.
- int mess_vector_toreal_nowarn (mess_vector v)

  Convert a vector to real values (without any warnings).
- int mess_vector_totype (mess_vector v, mess_datatype_t dt)

  Convert a vector to the given datatype.

10.178.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.179 copy.c File Reference

Functions

- int mess_vector_copy (mess_vector x, mess_vector y)

  Copy mess_vector structure x to a mess_vector y structure.
- int mess_vector_copy_tocomplex (mess_vector in, mess_vector out)

  Copy a mess_vector strucutre into a complex mess_vector structure.

Variables

- static mess_int_t __ONE = 1

10.179.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.179.2 Variable Documentation

10.179.2.1 mess_int_t __ONE = 1 [static]

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

10.180 csc.c File Reference

Functions

- static int __gaxpy_csc_ge (mess_matrix A, mess_vector x, mess_vector y)

  Sparse matrix-vector product for general matrices in MESS_CSC format.
- static int __gaxpyh_csc_ge (mess_matrix A, mess_vector x, mess_vector y)

  Hermitian transposed sparse matrix-vector product for general matrices in MESS_CSC format.
- static int __gaxpyt_csc_ge (mess_matrix A, mess_vector x, mess_vector y)

  Transposed sparse matrix-vector product for general matrices in MESS_CSC format.

10.180.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.180.2 Function Documentation

10.180.2.1 static int __gaxpy_csc_ge (mess_matrix *A*, mess_vector *x*, mess_vector *y*) [static]

Parameters

in	*A*	input matrix in MESS_CSC storage
in	*x*	input right hand side vector x
in,out	*y*	input/output vector y

Returns

zero on success or a non zero error code

The __gaxpy_csc_ge function computes

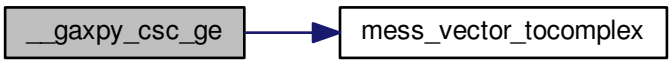
$$y \leftarrow Ax + y$$

for a general matrix in MESS_CSC format. Because this is only the kernel function it does not check any input or output arguments.

Attention

Internal use only.

Here is the call graph for this function:



10.180.2.2 static int __gaxpyh_csc_ge (mess_matrix *A*, mess_vector *x*, mess_vector *y*) [static]

Parameters

in	*A*	input matrix in Compressed Sparse Column storage
in	*x*	input right hand side vector x
in,out	*y*	input/output vector y

Returns

zero on success or a non zero error code

The __gaxpyh_csc_ge function computes

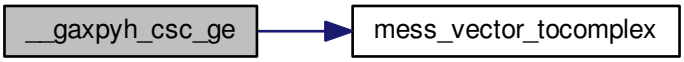
$$y \leftarrow A^H x + y$$

for a general matrix in MESS_CSC format. Because this is only the kernel function it does not check any input or output arguments.

Attention

Internal use only.

Here is the call graph for this function:



10.180.2.3 static int __gaxpyt_csc_ge (mess_matrix *A*, mess_vector *x*, mess_vector *y*) [static]

Parameters

in	*A*	input matrix in Compressed Sparse Column storage
in	*x*	input right hand side vector x
in,out	*y*	input/output vector y

Returns

zero on success or a non zero error code

The __gaxpyt_csc_ge function computes

$$y \leftarrow A^T x + y$$

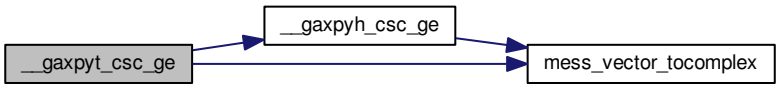
for a general matrix in MESS_CSC. If the matrix is real this function is the same as __gaxpyh_csc_ge but in the case of a complex matrix it only transposes it without using the complex conjugate values.

Because this is only the kernel function it does not check any input or output arguments.

Attention

Internal use only.

Here is the call graph for this function:



10.181 csparse.c File Reference

Data Structures

- struct csparse_solver
- struct csparse_solver_complex

Macros

- #define CSPARSE_ORDER 1
- #define CSPARSE_TOL 0.1

Functions

- static int compmess_int_t (const void ∗pl1, const void ∗pl2)
- static int cs_lsolve_analyse (long n, long ∗colptr, long ∗rowptr, mess_direct_levelset ∗levelset)

  Analyse triangular matrices.
- static mess_int_t ∗ perm_inv (mess_int_t const ∗p, mess_int_t n)
- static int cs_cl_lttsolve (const cs_cl ∗L, mess_double_cpx_t ∗x)
- static int cs_cl_uttsolve (const cs_cl ∗U, mess_double_cpx_t ∗x)
- static int cs_dlc_lsolve (const cs_dl ∗L, mess_double_cpx_t ∗x)
- static int cs_dlc_usolve (const cs_dl ∗U, mess_double_cpx_t ∗x)
- static int cs_dlc_ltsolve (const cs_dl ∗L, mess_double_cpx_t ∗x)
- static int cs_dlc_utsolve (const cs_dl ∗U, mess_double_cpx_t ∗x)
- static int csparse_getL (void ∗data, mess_matrix L)

  Get the L (the first factor) of a solver.
- static int csparse_getL_complex (void ∗data, mess_matrix L)

  Get the L (the first factor) of a solver (complex).
- static int csparse_getU (void ∗data, mess_matrix U)

  Get the U (the second factor) of a solver.
- static int csparse_getU_complex (void ∗data, mess_matrix U)

  Get the U (the second factor) of a solver (complex).
- static int csparse_getpermp (void ∗data, mess_int_t ∗p)

  Get the row permutation (real).
- static int csparse_getpermp_complex (void ∗data, mess_int_t ∗p)

  Get the row permutation (complex).
- static int csparse_getpermq (void ∗data, mess_int_t ∗q)

  Get the column permutation (real).
- static int csparse_getpermq_complex (void ∗data, mess_int_t ∗q)

  Get the column permutation (complex).
- static int csparse_solve (void ∗data, mess_vector b, mess_vector x)

  Solve $Ax = b$.
- int cs_dl_plsolve (const cs_dl ∗L, mess_direct_levelset ∗levelset, double ∗x)
- int cs_dl_ipvecsplit (const long ∗p, const mess_double_cpx_t ∗b, double ∗x1, double ∗x2, mess_int_t n)
- int cs_dl_ipveccombine (const long ∗p, double ∗x1, double ∗x2, mess_double_cpx_t ∗b, mess_int_t n)
- int cs_dl_pvecsplit (const long ∗p, const mess_double_cpx_t ∗b, double ∗x1, double ∗x2, mess_int_t n)
- int cs_dl_pveccombine (const long ∗p, double ∗x1, double ∗x2, mess_double_cpx_t ∗b, mess_int_t n)
- static int csparse_psolve (void ∗data, mess_vector b, mess_vector x)

  Solve $Ax = b$.
- static int csparse_solve_complex (void ∗data, mess_vector b, mess_vector x)

  Solve $Ax = b$ (complex)

- static int csparse_solvet (void *data, mess_vector b, mess_vector x)
  Solve $A^T x = b$.
- static int csparse_solvet_complex (void *data, mess_vector b, mess_vector x)
  Solve $A^T x = b$ (complex).
- static int csparse_solveh_complex (void *data, mess_vector b, mess_vector x)
  Solve $A^H x = b$ (complex)
- static int csparse_solvem (void *data, mess_matrix b, mess_matrix x)
  Solve $AX = B$ (matrix version)
- static int csparse_solvem_complex (void *data, mess_matrix b, mess_matrix x)
  Solve $AX = B$ (complex matrix version)
- static int csparse_solvemt (void *data, mess_matrix b, mess_matrix x)
  Solve $A^T X = B$ (matrix version)
- static int csparse_solvemt_complex (void *data, mess_matrix b, mess_matrix x)
  Solve $A^T X = B$ (complex matrix version)
- static int csparse_solvemh_complex (void *data, mess_matrix b, mess_matrix x)
  Solve $A^H X = B$ (complex matrix version)
- static int csparse_clear (void *solver)
  Clear a `CSparse` solver.
- static int csparse_clear_complex (void *solver)
  Clear a `CSparse` solver (complex).
- static int csparse_inverse (void *data, mess_matrix inv)
  Compute the inverse.
- static int csparse_inverse_complex (void *data, mess_matrix inv)
  Compute the inverse (complex).
- int mess_direct_create_csparse_lu (mess_matrix matrix, mess_direct solver)
  Generate a CXSparse LU solver.

10.181.1 Detailed Description

Author

Martin Köhler (`koehlerm@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

This file provide an interface for `CXSparse`. You need a working `SuiteSparse` to use it.

10.181.2 Macro Definition Documentation

10.181.2.1 #define CSPARSE_ORDER 1

10.181.2.2 #define CSPARSE_TOL 0.1

10.181.3 Function Documentation

10.181.3.1 static int compmess_int_t (const void * *pl1*, const void * *pl2*) [static]

10.181.3.2 static int cs_cl_lttsolve (const cs_cl * *L*, mess_double_cpx_t * *x*) [static]

10.181.3.3 static int cs_cl_uttsolve (const cs_cl * *U*, mess_double_cpx_t * *x*) [static]

10.181.3.4 int cs_dl_ipveccombine (const long * *p*, double * *x1*, double * *x2*, mess_double_cpx_t * *b*, mess_int_t *n*)

10.181.3.5 int cs_dl_ipvecsplit (const long * *p*, const mess_double_cpx_t * *b*, double * *x1*, double * *x2*, mess_int_t *n*)

10.181.3.6 int cs_dl_plsolve (const cs_dl * *L*, mess_direct_levelset * *levelset*, double * *x*)

10.181.3.7 int cs_dl_pveccombine (const long * *p*, double * *x1*, double * *x2*, mess_double_cpx_t * *b*, mess_int_t *n*)

10.181.3.8 int cs_dl_pvecsplit (const long * *p*, const mess_double_cpx_t * *b*, double * *x1*, double * *x2*, mess_int_t *n*)

10.181.3.9 static int cs_dlc_lsolve (const cs_dl * *L*, mess_double_cpx_t * *x*) [static]

10.181.3.10 static int cs_dlc_ltsolve (const cs_dl * *L*, mess_double_cpx_t * *x*) [static]

10.181.3.11 static int cs_dlc_usolve (const cs_dl * *U*, mess_double_cpx_t * *x*) [static]

10.181.3.12 static int cs_dlc_utsolve (const cs_dl * *U*, mess_double_cpx_t * *x*) [static]

10.181.3.13 static int cs_lsolve_analyse (long *n*, long * *colptr*, long * *rowptr*, mess_direct_levelset * *levelset*) [static]

Parameters

n	
colptr	
rowptr	
levelset	The cs_lsolve_analyse function analyses a triangular matrix for parallelization.

Here is the call graph for this function:

10.181.3.14 static int csparse_clear (void * *solver*) [static]

Parameters

in,out	*solver*	solver data

Returns

always zero

The csparse_clear function is the clean up function.

10.181.3.15 static int csparse_clear_complex (void * *solver*) [static]

Parameters

in,out	*solver*	solver data

Returns

always zero

The csparse_clear_complex function is the clean up function.

10.181.3.16 static int csparse_getL (void * *data*, mess_matrix *L*) [static]

Parameters

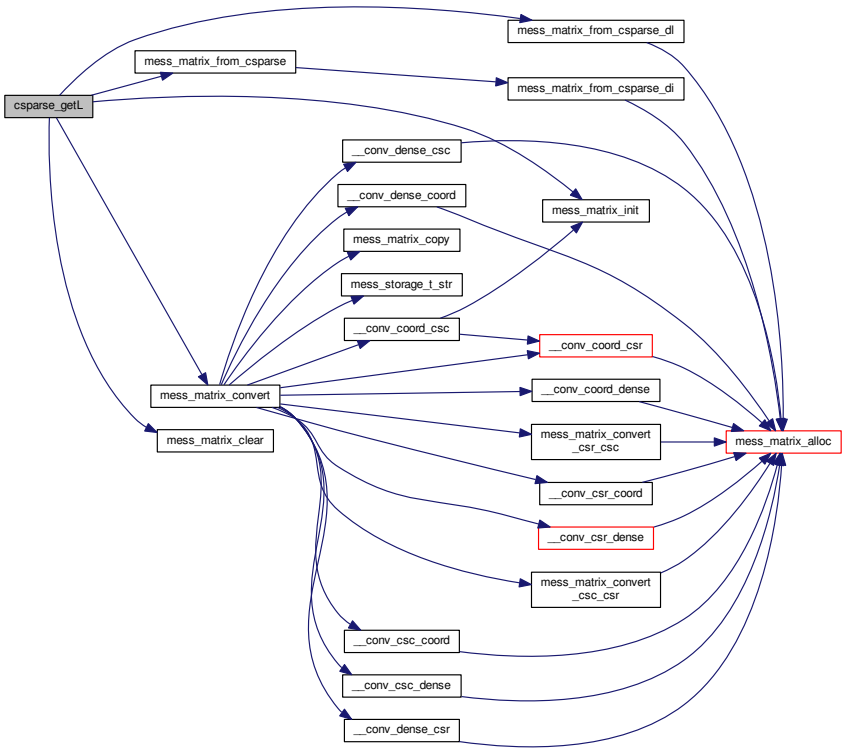
in	*data*	input solver data
in,out	*L*	output L factor

Returns

zero on success or a non-zero error value otherwise

The csparse_getL function gets the L factor from the cs_dln object. Output is in CSR format.

Here is the call graph for this function:



10.181.3.17 static int csparse_getL_complex (void * *data*, mess_matrix *L*) [static]

Parameters

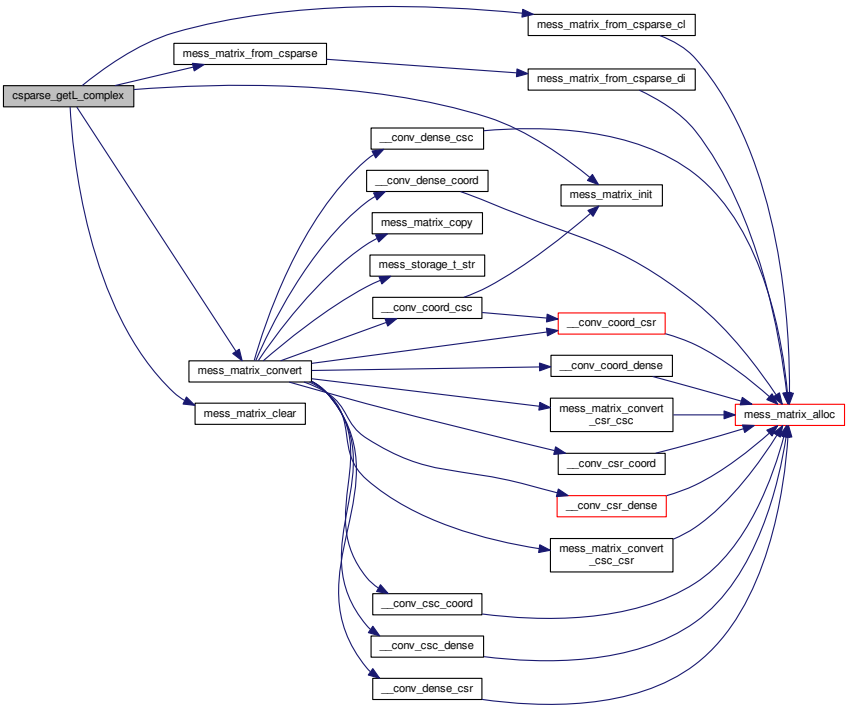
in	*data*	input solver data
in,out	*L*	output L factor

Returns

zero on success or a non-zero error value otherwise

The csparse_getL_complex function gets the L factor from the cs_cln object. Output is in CSR format.

Here is the call graph for this function:



10.181.3.18 static int csparse_getpermp (void * *data*, mess_int_t * *p*) [static]

Parameters

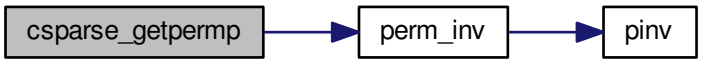
in	*data*	input solver data
in,out	*p*	output permutation

Returns

zero on success or a non-zero error value otherwise

The csparse_getpermp function gets the row permutation of the decomposition.

Here is the call graph for this function:



10.181.3.19 static int csparse_getpermp_complex (void * *data*, mess_int_t * *p*) [static]

Parameters

in	*data*	input solver data
in,out	*p*	output permutation

Returns

zero on success or a non-zero error value otherwise

The csparse_getpermp_complex function gets the row permutation of the decomposition.

Here is the call graph for this function:

10.181.3.20 static int csparse_getpermq (void * *data*, mess_int_t * *q*) [static]

Parameters

in	*data*	input solver data
in,out	*q*	output permutation

Returns

zero on success or a non-zero error value otherwise

The csparse_getpermq function gets the column permutation of the decomposition.

10.181.3.21 static int csparse_getpermq_complex (void * *data*, mess_int_t * *q*) [static]

Parameters

in	*data*	input solver data
in,out	*q*	output permutation

Returns

zero on success or a non-zero error value otherwise

The csparse_getpermq_complex function gets the column permutation of the decomposition.

10.181.3.22 static int csparse_getU (void * *data*, mess_matrix *U*) [static]

Parameters

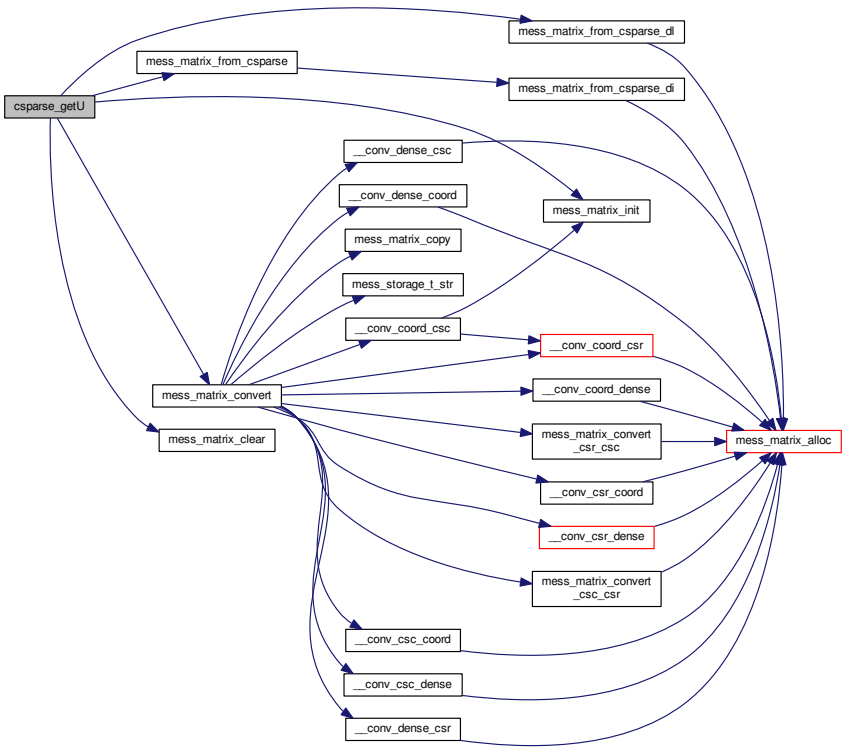
in	*data*	input solver data
in,out	*U*	output U factor

Returns

zero on success or a non-zero error value otherwise

The csparse_getU function gets the U factor from the cs_dln object. Output is in CSR format.

Here is the call graph for this function:



10.181.3.23 static int csparse_getU_complex (void * *data*, mess_matrix *U*) [static]

Parameters

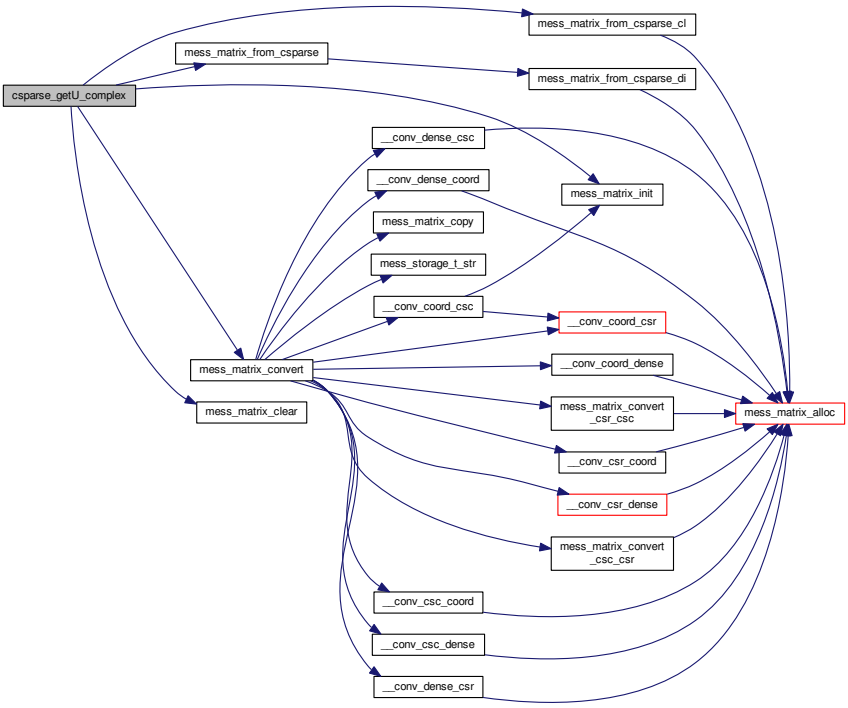
in	*data*	input solver data
in,out	*U*	output U factor

Returns

zero on success or a non-zero error value otherwise

The csparse_getU_complex function gets the U factor from the cs_cln object. Output is in CSR format.

Here is the call graph for this function:



10.181.3.24 static int csparse_inverse (void * *data*, **mess_matrix** *inv*) [static]

Parameters

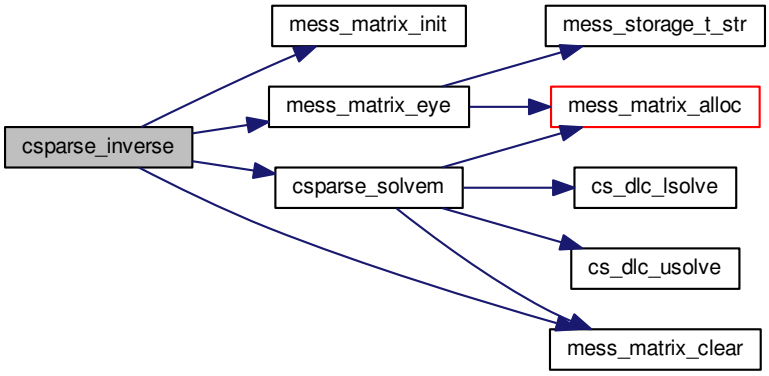
in	*data*	input pointer to the internal data structure
in,out	*inv*	output inverse

Returns

zero on success or a non-zero error value otherwise

The csparse_inverse function computes A^{-1}.

Here is the call graph for this function:



10.181.3.25 static int csparse_inverse_complex (void * *data*, mess_matrix *inv*) [static]

Parameters

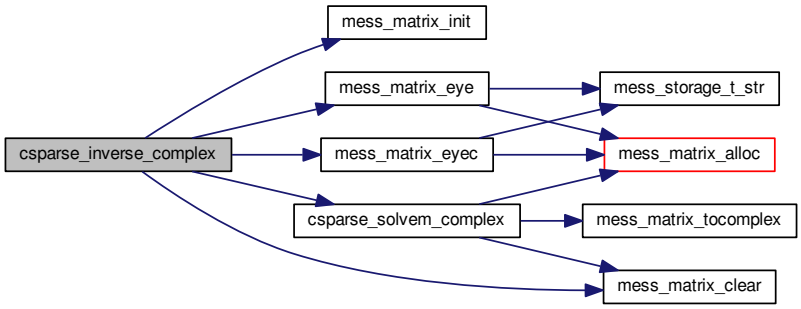
in	*data*	input pointer to the internal data structure
in,out	*inv*	output inverse

Returns

zero on success or a non-zero error value otherwise

The csparse_inverse_complex function computes A^{-1}.

Here is the call graph for this function:



10.181.3.26 static int csparse_psolve (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

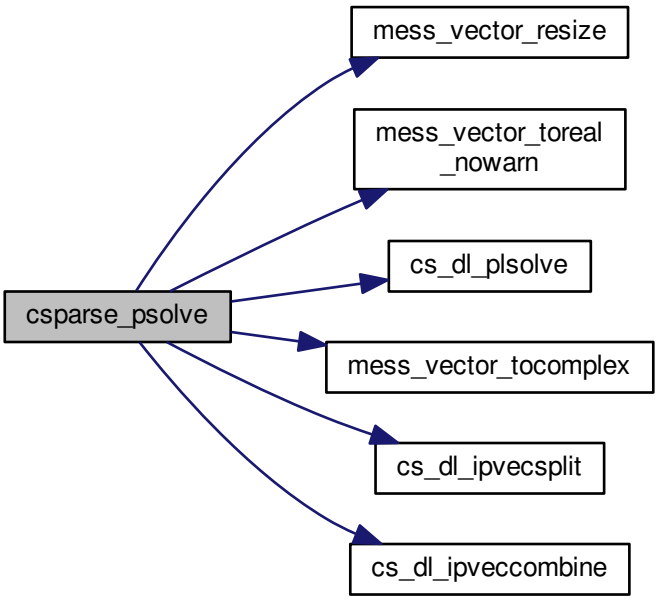
in	*data*	input solver data
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The csparse_solve function solves the system $Ax = b$ with b and x vectors.

Here is the call graph for this function:



10.181.3.27 static int csparse_solve (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input solver data
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The csparse_solve function solves the system $Ax = b$ with b and x vectors.

10.181.3.28 static int csparse_solve_complex (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

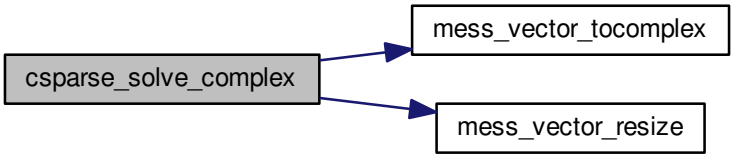
in	*data*	input solver data
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The csparse_solve_complex function solves the system $Ax = b$ with b and x vectors.

Here is the call graph for this function:



10.181.3.29 static int csparse_solveh_complex (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

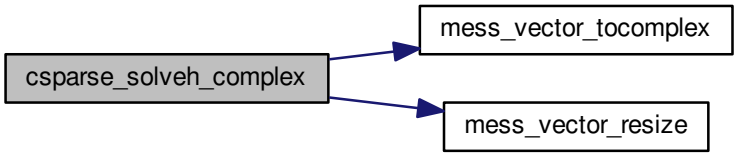
in	*data*	input solver data
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The csparse_solveh_complex function solves the system $A^H x = b$ with b and x vectors.

Here is the call graph for this function:



10.181.3.30 static int csparse_solvem (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

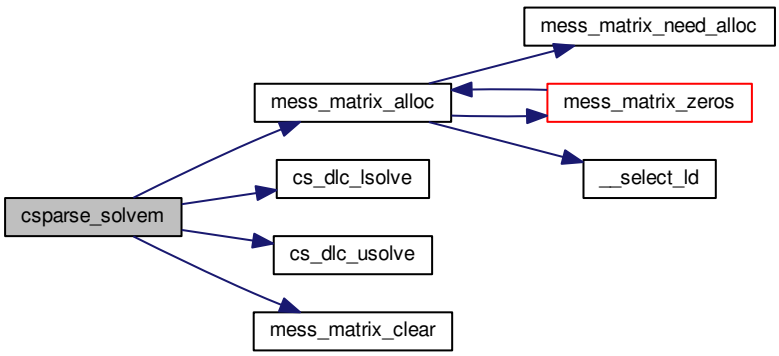
in	*data*	input solver data
in	*b*	input right hand sides
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The csparse_solvem function solves the system $AX = B$ with B and X matrices.

Here is the call graph for this function:



10.181.3.31 static int csparse_solvem_complex (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

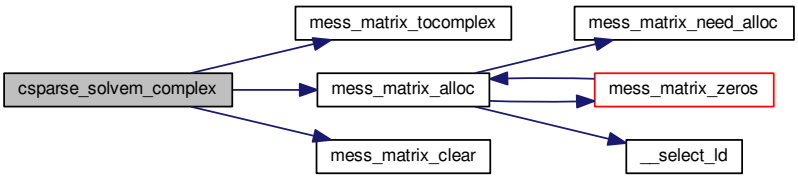
in	*data*	input solver data
in	*b*	input right hand sides
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The csparse_solvem_complex function solves the system $AX = B$ with B and X matrices.

Here is the call graph for this function:



10.181.3.32 static int csparse_solvemh_complex (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

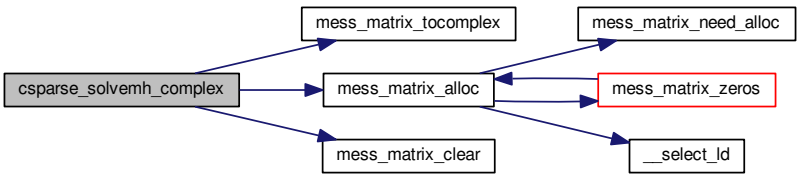
in	*data*	input solver data
in	*b*	input right hand sides
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The csparse_solvemh_complex function solves the system $A^H X = B$ with B and X matrices.

Here is the call graph for this function:



10.181.3.33 static int csparse_solvemt (void * *data,* mess_matrix *b,* mess_matrix *x*) [static]

Parameters

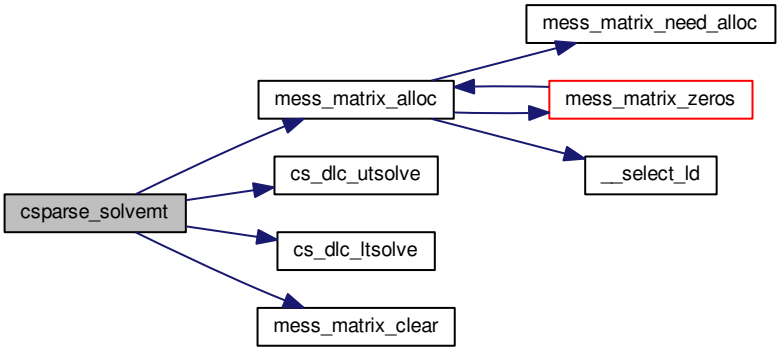
in	*data*	input solver data
in	*b*	input right hand sides
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The csparse_solvemt function solves the system $A^T X = B$ with B and X matrices.

Here is the call graph for this function:



10.181.3.34 static int csparse_solvemt_complex (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

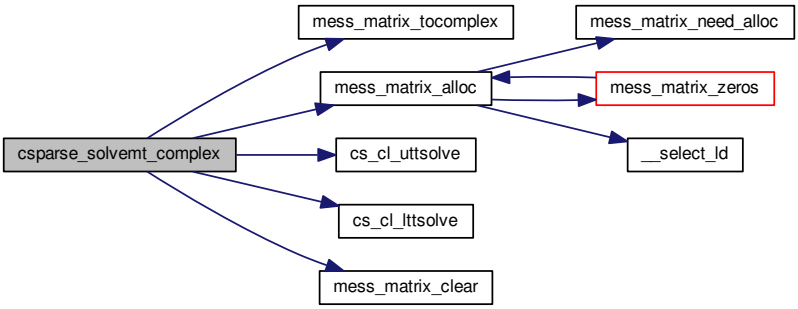
in	*data*	input solver data
in	*b*	input right hand sides
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The csparse_solvemt_complex function solves the system $A^T X = B$ with B and X matrices.

Here is the call graph for this function:



10.181.3.35 static int csparse_solvet (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

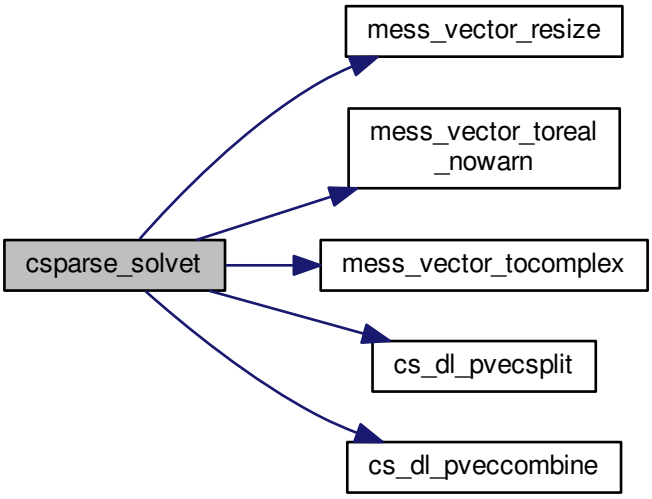
in	*data*	input solver data
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The csparse_solvet function solves the system $A^T x = b$ with b and x vectors.

Here is the call graph for this function:



10.181.3.36 static int csparse_solvet_complex (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

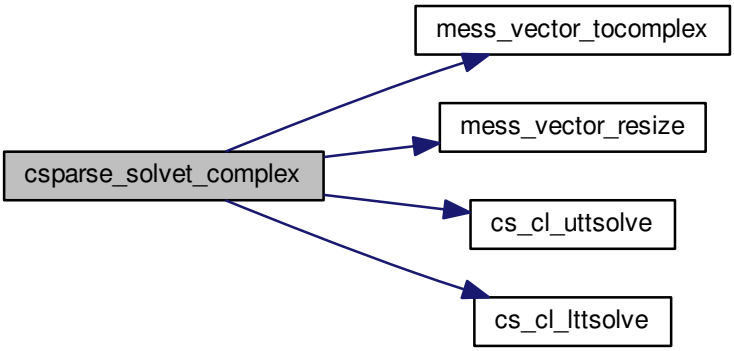
in	*data*	input solver data
in	*b*	input right hand side
in,out	*x*	input/output solution

Returns

zero on success or a non-zero error value otherwise

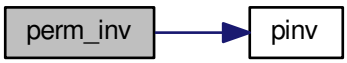
The csparse_solvet_complex function solves the system $A^T x = b$ with b and x vectors.

Here is the call graph for this function:



10.181.3.37 static mess_int_t* perm_inv (mess_int_t const * *p*, mess_int_t *n*) [static]

Here is the call graph for this function:



10.182 csparse_chol.c File Reference

Data Structures

- struct csparse_solver
- struct csparse_solver_complex

Macros

- #define CSPARSE_ORDER 2

Functions

- static mess_int_t * perm_inv (mess_int_t const *p, mess_int_t n)
- static int cs_cl_lttsolve (const cs_cl *L, mess_double_cpx_t *x)
- static int cs_dlc_lsolve (const cs_dl *L, mess_double_cpx_t *x)
- static int cs_dlc_ltsolve (const cs_dl *L, mess_double_cpx_t *x)
- static int csparse_getL (void *data, mess_matrix L)

  Get the L (the first factor) of a solver.
- static int csparse_getL_complex (void *data, mess_matrix L)

  Get the L (the first factor) of a solver (complex).
- static int csparse_getU (void *data, mess_matrix U)

  Get the U (the second factor) of a solver.
- static int csparse_getU_complex (void *data, mess_matrix U)

  Get the U (the second factor) of a solver (complex).
- static int csparse_getpermp (void *data, mess_int_t *p)

  Get the row permutation (real).
- static int csparse_getpermp_complex (void *data, mess_int_t *p)

  Get the row permutation (complex).
- static int cs_dl_ipvecsplit (const long *p, const mess_double_cpx_t *b, double *x1, double *x2, mess_int_t n)
- static int cs_dl_pveccombine (const long *p, double *x1, double *x2, mess_double_cpx_t *b, mess_int_t n)
- static int csparse_solve (void *data, mess_vector b, mess_vector x)

  Solve $Ax = b$.
- static int csparse_solve_complex (void *data, mess_vector b, mess_vector x)

  Solve $Ax = b$ *(complex)*
- static int csparse_solvet (void *data, mess_vector b, mess_vector x)

  Solve $A^T x = b$.
- static int csparse_solvet_complex (void *data, mess_vector b, mess_vector x)

  Solve $A^T x = b$ *(complex)*
- static int csparse_solveh_complex (void *data, mess_vector b, mess_vector x)

  Solve $A^H x = b$ *(complex)*
- static int csparse_solvem (void *data, mess_matrix b, mess_matrix x)

  Solve $AX = B$ *(matrix version)*
- static int csparse_solvem_complex (void *data, mess_matrix b, mess_matrix x)

  Solve AX=B (complex matrix version).
- static int csparse_solvemt (void *data, mess_matrix b, mess_matrix x)

  Solve $A^T X = B$ *(matrix version)*
- static int csparse_solvemt_complex (void *data, mess_matrix b, mess_matrix x)

  Solve $A^T X = B$ *(complex matrix version)*
- static int csparse_solvemh_complex (void *data, mess_matrix b, mess_matrix x)

  Solve $A^H X = B$ *(complex matrix version)*
- static int csparse_clear (void *solver)

  Clear a `CSparse` *solver.*
- static int csparse_clear_complex (void *solver)

  Clear a `CSparse` *solver (complex).*
- static int csparse_inverse (void *data, mess_matrix inv)

  Compute the inverse.
- static int csparse_inverse_complex (void *data, mess_matrix inv)

  Compute the inverse (complex).
- int mess_direct_create_csparse_cholesky (mess_matrix matrix, mess_direct solver)

  Generate a CXSparse Cholesky solver.

10.182.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file provide an interface for CXSparse. You need a working SuiteSparse to use it.

10.182.2 Macro Definition Documentation

10.182.2.1 #define CSPARSE_ORDER 2

10.182.3 Function Documentation

10.182.3.1 static int cs_cl_lttsolve (const cs_cl * *L*, mess_double_cpx_t * *x*) [static]

10.182.3.2 static int cs_dl_ipvecsplit (const long * *p*, const mess_double_cpx_t * *b*, double * *x1*, double * *x2*, mess_int_t *n*) [static]

10.182.3.3 static int cs_dl_pveccombine (const long * *p*, double * *x1*, double * *x2*, mess_double_cpx_t * *b*, mess_int_t *n*) [static]

10.182.3.4 static int cs_dlc_lsolve (const cs_dl * *L*, mess_double_cpx_t * *x*) [static]

10.182.3.5 static int cs_dlc_ltsolve (const cs_dl * *L*, mess_double_cpx_t * *x*) [static]

10.182.3.6 static int csparse_clear (void * *solver*) [static]

Parameters

in,out	*solver*	solver data

Returns

always zero

The csparse_clear function is the clean up function.

10.182.3.7 static int csparse_clear_complex (void * *solver*) [static]

Parameters

in,out	*solver*	solver data

Returns

always zero

The csparse_clear_complex function is the clean up function.

10.182.3.8 static int csparse_getL (void * *data*, mess_matrix *L*) [static]

Parameters

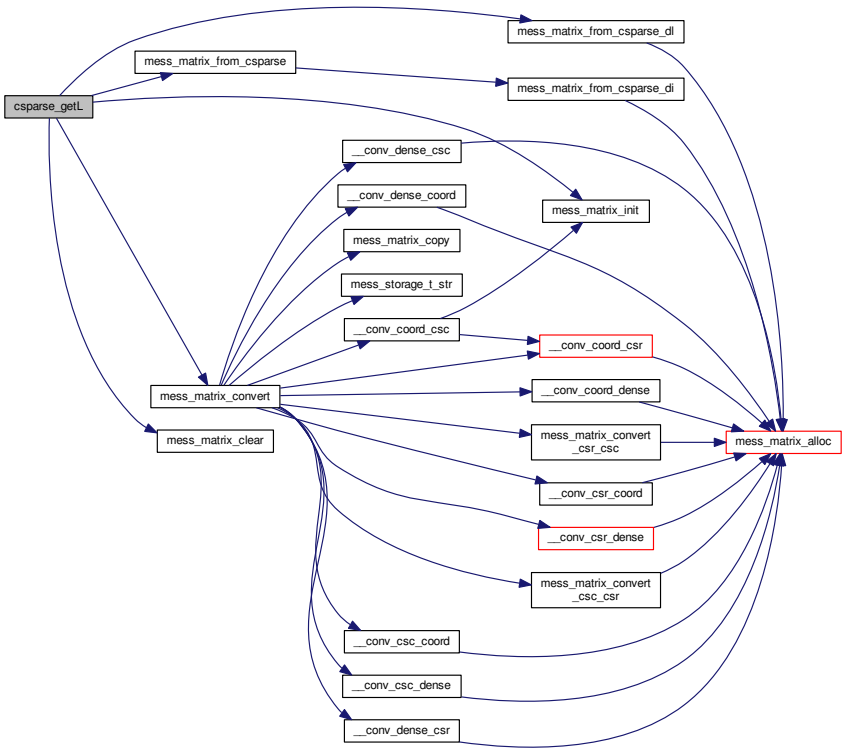
in	*data*	input solver data
in,out	*L*	output L factor

Returns

zero on success or a non zero error code otherwise

The csparse_getL function gets the L factor from the cs_dln object. Output is in CSR format.

Here is the call graph for this function:



10.182.3.9 static int csparse_getL_complex (void * *data*, mess_matrix *L*) [static]

Parameters

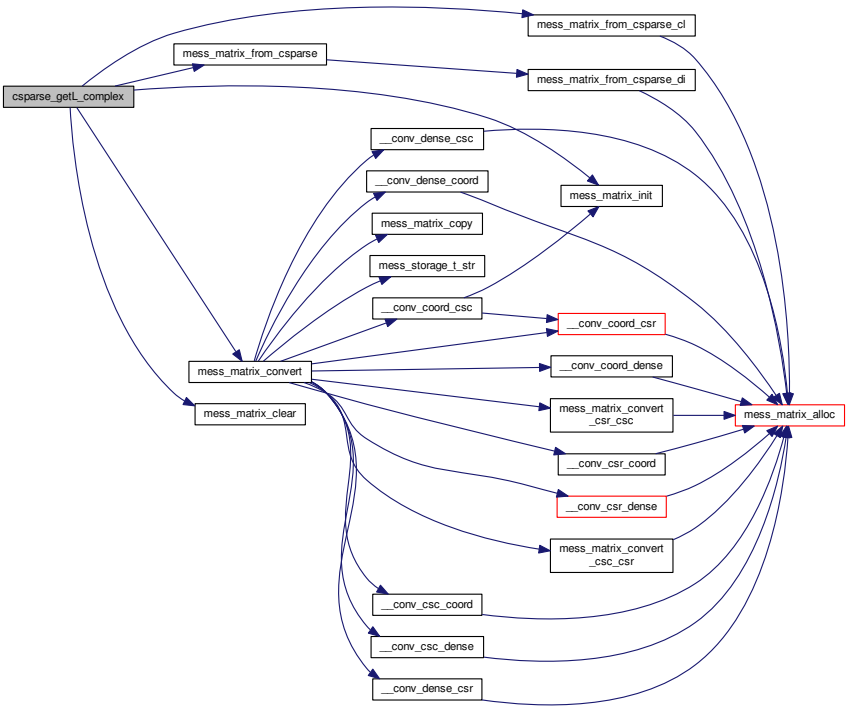
in	*data*	input solver data
in,out	*L*	output L factor

Returns

zero on success or a non zero error code otherwise

The csparse_getL_complex function gets the L factor from the cs_cln object. Output is in CSR format.

Here is the call graph for this function:



10.182.3.10 static int csparse_getpermp (void * *data*, mess_int_t * *p*) [static]

Parameters

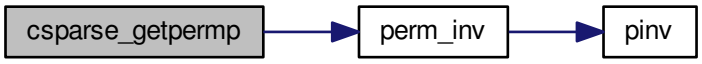
in	*data*	input solver data
in,out	*p*	output permutation

Returns

zero on success or a non zero error code otherwise

The csparse_getpermp function gets the row permutation of the decomposition.

Here is the call graph for this function:



10.182.3.11 static int csparse_getpermp_complex (void * *data*, mess_int_t * *p*) [static]

Parameters

in	*data*	input solver data
in,out	*p*	output permutation

Returns

zero on success or a non zero error code otherwise

The csparse_getpermp_complex function gets the row permutation of the decomposition.

Here is the call graph for this function:

10.182.3.12 static int csparse_getU (void * *data*, mess_matrix *U*) [static]

Parameters

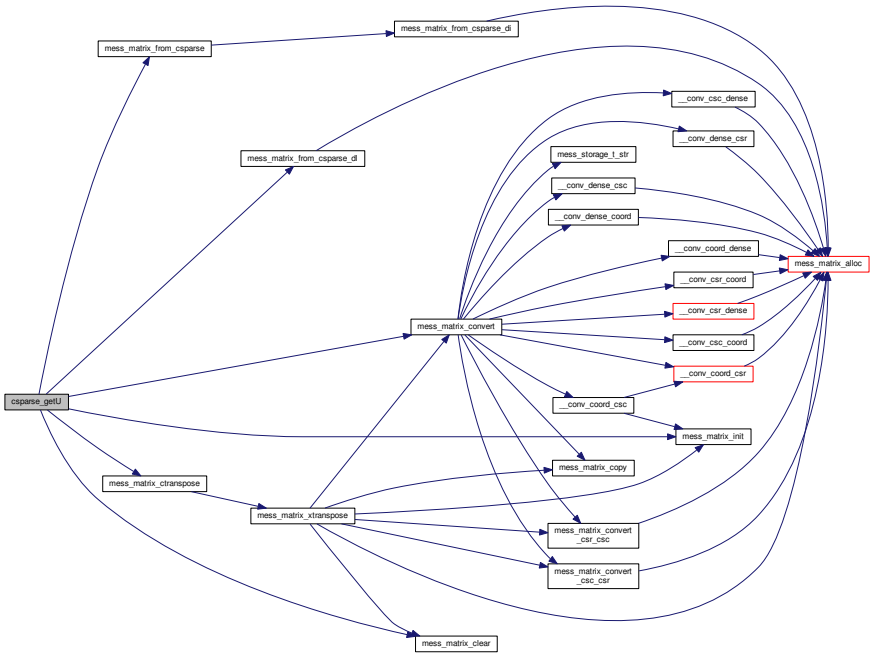
in	*data*	input solver data
in,out	*U*	output U factor

Returns

zero on success or a non zero error code otherwise

The csparse_getU function gets the U factor from the cs_dln object. Output is in CSR format.

Here is the call graph for this function:



10.182.3.13 static int csparse_getU_complex (void * *data*, mess_matrix *U*) [static]

Parameters

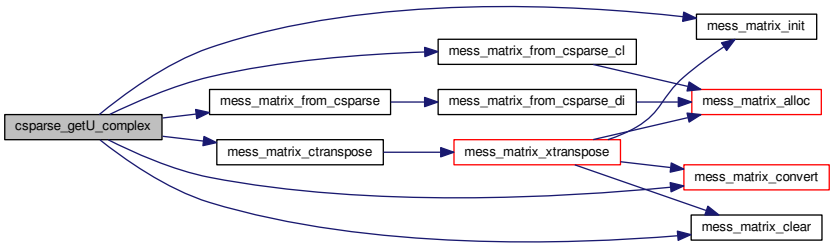
in	*data*	input solver data
in,out	*U*	output U factor

Returns

zero on success or a non zero error code otherwise

The csparse_getU_complex function gets the U factor from the cs_cln object. Output is in CSR format.

Here is the call graph for this function:



10.182.3.14 static int csparse_inverse (void * *data*, mess_matrix *inv*) [static]

Parameters

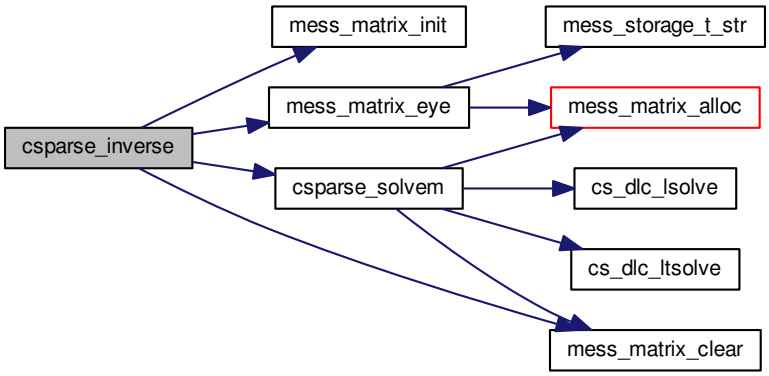
in	*data*	input pointer to the internal data structure
in,out	*inv*	output inverse

Returns

zero on success or a non zero error code otherwise

The csparse_inverse function computes A^{-1}.

Here is the call graph for this function:



10.182.3.15 static int csparse_inverse_complex (void * *data*, mess_matrix *inv*) [static]

Parameters

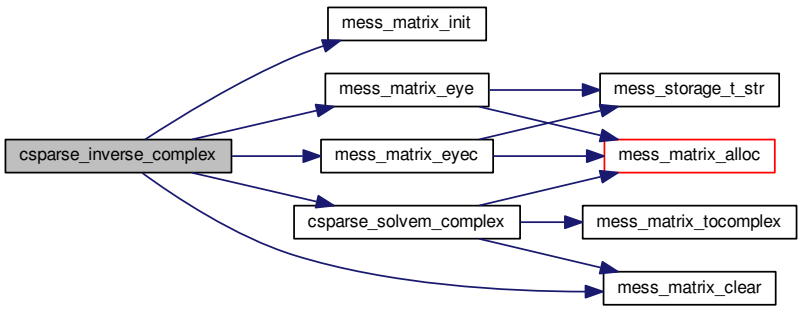
in	*data*	input pointer to the internal data structure
in,out	*inv*	output inverse

Returns

zero on success or a non zero error code otherwise

The csparse_inverse_complex function computes A^{-1}

Here is the call graph for this function:



10.182.3.16 static int csparse_solve (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

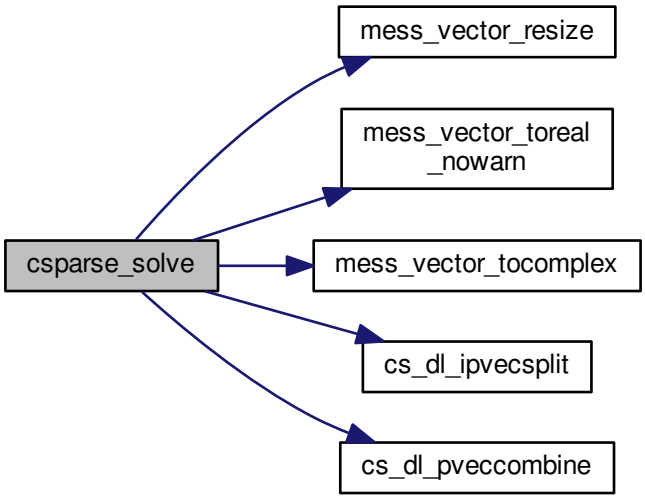
in	*data*	input solver data
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error code otherwise

The csparse_solve function solves the system $Ax = b$ with x and b vectors.

Here is the call graph for this function:



10.182.3.17 static int csparse_solve_complex (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input solver data
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error code otherwise

The csparse_solve_complex function solves the system $Ax = b$ with x and b vectors.

Here is the call graph for this function:

10.182.3.18 static int csparse_solveh_complex (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

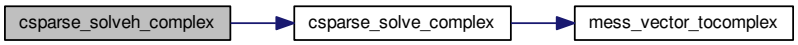
in	*data*	input solver data
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error code otherwise

The csparse_solve function solves the system $A^H x = b$ with x and b vectors.

Here is the call graph for this function:



10.182.3.19 static int csparse_solvem (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

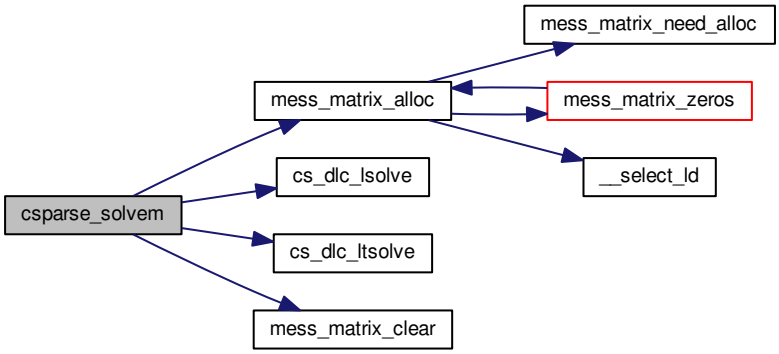
in	*data*	input solver data
in	*b*	input right hand sides
in,out	*x*	solution

Returns

zero on success or a non zero error code otherwise

The csparse_solvem function solves the system $AX = B$ with X and B matrices.

Here is the call graph for this function:



10.182.3.20 static int csparse_solvem_complex (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

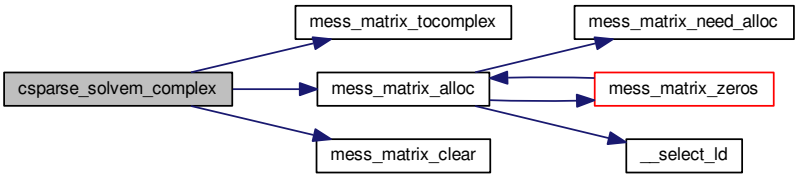
in	*data*	input solver data
in	*b*	input right hand sides
in,out	*x*	solution

Returns

zero on success or a non zero error code otherwise

The csparse_solvem_complex function solves the system $AX = B$ with X and B matrices.

Here is the call graph for this function:



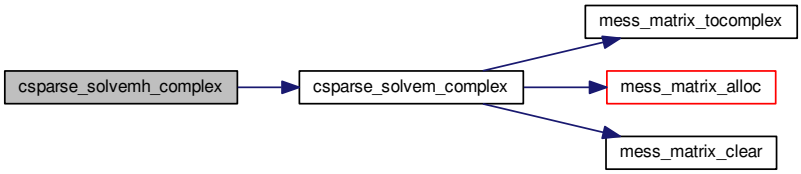
10.182.3.21 static int csparse_solvemh_complex (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input solver data
in	*b*	input right hand sides
in,out	*x*	solution

The csparse_solvemt_complex function solves the system $A^T X = B$ with X and B matrices.

Here is the call graph for this function:



10.182.3.22 static int csparse_solvemt (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

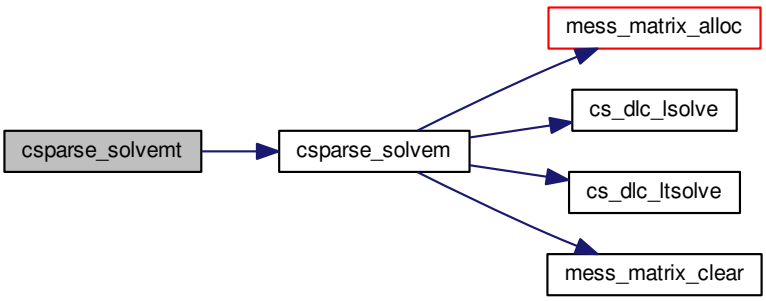
in	*data*	input solver data
in	*b*	input right hand sides
in,out	*x*	solution

Returns

zero on success or a non zero error code otherwise

The csparse_solvemt function solves the system $A^T X = B$ with X and B matrices.

Here is the call graph for this function:



10.182.3.23 static int csparse_solvemt_complex (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

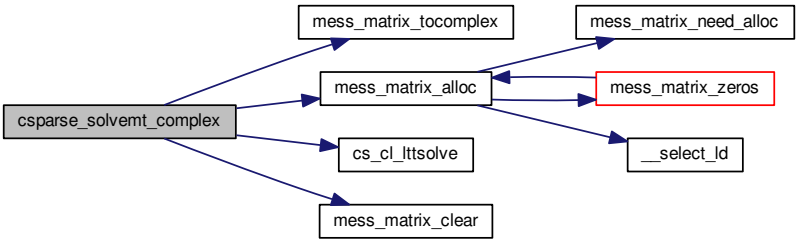
in	*data*	input solver data
in	*b*	input right hand sides
in,out	*x*	solution

Returns

zero on success or a non zero error code otherwise

The csparse_solvemt_complex function solves the system $A^T X = B$ with X and B matrices.

Here is the call graph for this function:



10.182.3.24 static int csparse_solvet (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

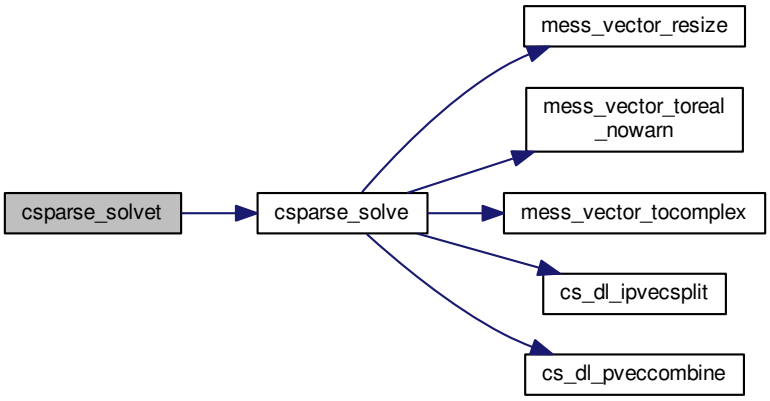
in	*data*	input solver data
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error code otherwise

The csparse_solvet function solves the system $A^T x = b$ with x and b vectors.

Here is the call graph for this function:



10.182.3.25 static int csparse_solvet_complex (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

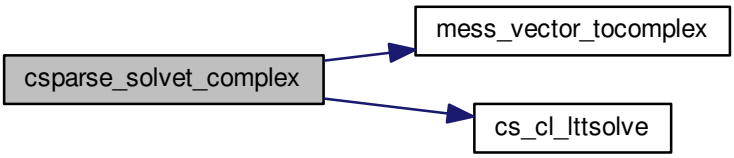
in	*data*	input solver data
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error code otherwise

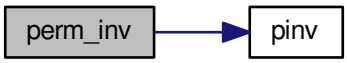
The csparse_solvet_complex function solves the system $A^T x = b$ with x and b vectors.

Here is the call graph for this function:



10.182.3.26 static mess_int_t* perm_inv (mess_int_t const * *p*, mess_int_t *n*) [static]

Here is the call graph for this function:



10.183 csr.c File Reference

Functions

- static int __gaxpy_csr_ge (mess_matrix A, mess_vector x, mess_vector y)

  Sparse matrix-vector product for general matrices in MESS_CSR format.
- static int __gaxpyh_csr_ge (mess_matrix A, mess_vector x, mess_vector y)

  Hermitian transposed sparse matrix-vector product for general matrices in MESS_CSR format.
- static int __gaxpyt_csr_ge (mess_matrix A, mess_vector x, mess_vector y)

  Hermitian transposed sparse matrix-vector product for general matrices in MESS_CSR format.
- static int __gaxpy_csr_sym (mess_matrix A, mess_vector xv, mess_vector yv)

  Sparse matrix-vector product for (skew-)symmertric matrices in MESS_CSR format.

10.183.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.183.2 Function Documentation

10.183.2.1 static int __gaxpy_csr_ge (mess_matrix *A*, mess_vector *x*, mess_vector *y*) [static]

Parameters

in	*A*	input matrix in Compressed Sparse Row storage
in	*x*	input right hand side vector x
in,out	*y*	input/output vector y

Returns

zero on success or a non zero error code

The __gaxpy_csr_ge function computes

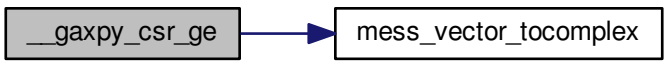
$$y \leftarrow Ax + y$$

for a general matrix in MESS_CSR format. Because this is only the kernel function it does not check any input or output arguments.

Attention

Internal use only.

Here is the call graph for this function:



10.183.2.2 static int __gaxpy_csr_sym (mess_matrix *A*, mess_vector *xv*, mess_vector *yv*) [static]

Parameters

in	*A*	input matrix in Compressed Sparse Row storage
in	*xv*	input right hand side vector
in,out	*yv*	input/output vector

Returns

zero on success or a non zero error code

The __gaxpy_csr_sym function computes

$$y \leftarrow Ax + y$$

for a (skew-)symmetric matrix in MESS_CSR format. Because this is only the kernel function it does not check any input or output arguments.

Attention

Internal use only.

10.183.2.3 static int __gaxpyh_csr_ge (mess_matrix *A*, mess_vector *x*, mess_vector *y*) [static]

Parameters

in	*A*	input matrix in Compressed Sparse Row storage
in	*x*	input right hand side vector x
in,out	*y*	input/output vector y

Returns

zero on success or a non zero error code

The __gaxpyh_csr_ge function computes

$$y \leftarrow A^H x + y$$

for a general matrix in MESS_CSR format. Because this is only the kernel function it does not check any input or output arguments.

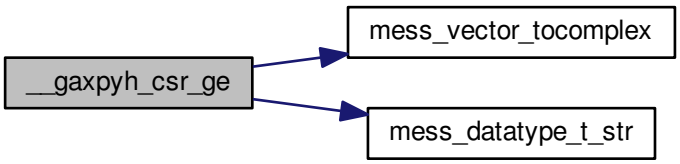
Attention

Internal use only.

Warning

untested parallel code

Here is the call graph for this function:



10.183.2.4 static int __gaxpyt_csr_ge (mess_matrix *A*, mess_vector *x*, mess_vector *y*) [static]

Parameters

in	*A*	input matrix in Compressed Sparse Row storage
in	*x*	input right hand side vector x
in,out	*y*	input/output vector y

Returns

zero on success or a non zero error code

The __gaxpyt_csr_ge function computes

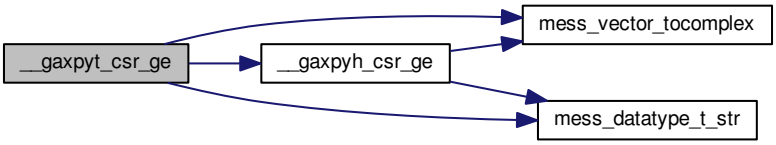
$$y \leftarrow A^T x + y$$

for a general matrix in MESS_CSR format. Because this is only the kernel function it does not check any input or output arguments. In the case of a real matrix this is the same as __gaxpyh_csr_ge.

Attention

Internal use only.

Here is the call graph for this function:



10.184 cuthill.c File Reference

Data Structures

- struct int_list

Macros

- #define SUCCESS 0
- #define FAIL 1
- #define TRY_MALLOC(__who__, __somuch__, __type__, __func__)
- #define TRY_REALLOC(__who__, __somuch__, __type__, __func__)
- #define FUNCTION_FAILURE_HANDLE(__err__, __who__, __where__)

Functions

- static int int_list_alloc (struct int_list ∗L, mess_int_t ixnmax, mess_int_t elnmax)

  Allocate memory in the int_list structure for a given size.
- static void int_list_free (struct int_list ∗L)

  Free memory in the int_list structure.
- static int comp_mess_int_t (const void ∗a1, const void ∗b1)

  Compare two integers.
- static int bandw_red_perm (mess_int_t nr, mess_int_t ∗adix, mess_int_t ∗adel, mess_int_t ∗perm, mess_int_t ∗iperm, mess_int_t ∗bandw)

  Compute bandwidth and corresponding permutation.
- static int bandw_mark_connected (mess_int_t label, mess_int_t start, mess_int_t ∗adix, mess_int_t ∗adel, mess_int_t n1, mess_int_t ∗marked)

  Mark vertices with a label.
- static int bandw_gipost (mess_int_t n1, mess_int_t n2, struct int_list ∗ad, mess_int_t ∗perm, mess_int_t ∗iperm, mess_int_t ∗bandw)

  Compute bandwidth and corresponding permutation of a connected subgraph.
- static int bandw_level_struct (mess_int_t root, mess_int_t n1, mess_int_t n2, struct int_list ∗ad, struct int_list ∗L, mess_int_t ∗hint)

  Define level structure for a given subgraph.
- static int bandw_number_adj_vert (struct int_list ∗ad, struct int_list ∗Lv, mess_int_t ∗level, mess_int_t l1, mess_int_t l2, mess_int_t n1, mess_int_t n2, mess_int_t ∗perm, mess_int_t ∗iperm, mess_int_t ∗current, mess_int_t ∗sorter)

  Number adjacent vertices.
- int mess_matrix_reorder_rcm (mess_matrix A, mess_int_t ∗perm)

  Compute the Reverse Cuthill-McKee reordering of a matrix.

10.184.1 Macro Definition Documentation

10.184.1.1 #define FAIL 1

10.184.1.2 #define FUNCTION_FAILURE_HANDLE(*__err__*, *__who__*, *__where__*)

Value:

```
{\
    if ((__err__)!=SUCCESS) \
    {\
        fprintf(stderr, \
                "line-%d in %s, file %s: call to %s returned with %d(!=SUCCESS)\n",\
                __LINE__, #__where__, __FILE__, #__who__, __err__ );\
        return FAIL; \
    }\
}
```

10.184.1.3 #define SUCCESS 0

10.184.1.4 #define TRY_MALLOC(*__who__*, *__somuch__*, *__type__*, *__func__*)

Value:

```
{       \
    mess_try_alloc2(__who__, __type__ *, sizeof(__type__) * (__somuch__)); \
    if ((__who__)==NULL)                                \
    {                                   \
        fprintf(stderr,                                 \
                "line-%d in %s, file %s: malloc for %d %s entries to %s failed\n",\
                __LINE__, #__func__, __FILE__, (int)(__somuch__), #__type__, #__who__ );     \
        return FAIL;                                    \
    }                                   \
}
```

10.184.1.5 #define TRY_REALLOC(*__who__*, *__somuch__*, *__type__*, *__func__*)

Value:

```
{       \
    mess_try_realloc2(__who__, __type__ *, (__somuch__) * sizeof(__type__)); \
    if ((__who__)==NULL)                        \
    {                                   \
        fprintf(stderr,                         \
                "line-%d in %s, file %s: realloc for %d %s entries to %s failed\n", \
                __LINE__, #__func__, __FILE__, (int)(__somuch__), #__type__, #__who__ ); \
        return FAIL;                            \
    }                                   \
}
```

10.184.2 Function Documentation

10.184.2.1 static int bandw_gipost (mess_int_t *n1*, mess_int_t *n2*, struct int_list ∗ *ad*, mess_int_t ∗ *perm*, mess_int_t ∗ *iperm*, mess_int_t ∗ *bandw*) [static]

Parameters

in	*n1*	input index of first node of connected subgraph
in	*n2*	input index of last node $+1$ of connected subgraph
in	*ad*	input adjacency list of whole graph
in	*perm*	input bandwidth reducing permuation
in	*iperm*	input reverse of $perm$
in,out	*bandw*	input/output bandwith for subgraph

Returns

zero on success or a non-zero error value otherwise

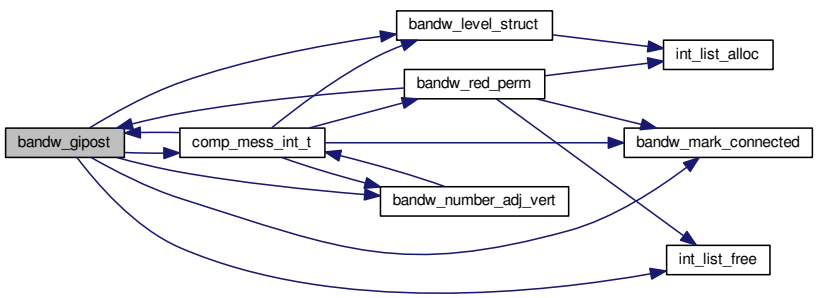
The bandw_gipost function applies the Gibbs-Poole-Stockmeyer modification of the Cuthill-McKee algorithm to produce a bandwith reducing permutaion for a given full adjency list of a connected subgraph of a sparse matrix.

In the adjacency list ad only the $n1$ to $n2-1$ parts are read and in the (inverse of the) bandwidth reducing permutation the $n1$ to $n2-1$ entries are written to an integer vector.

Attention

Internal use only.

Here is the call graph for this function:



10.184.2.2 static int bandw_level_struct (mess_int_t *root*, mess_int_t *n1*, mess_int_t *n2*, struct int_list * *ad*, struct int_list * *L*, mess_int_t * *hint*) [static]

Parameters

in	*root*	input root of level structure
in	*n1*	input index of first node of connected subgraph
in	*n2*	input index of last node $+1$ of connected subgraph
in	*ad*	input adjacency list of whole graph
out	*L*	output level structure
in,out	*hint*	input/output integer help vector

Returns

zero on success or a non-zero error value otherwise

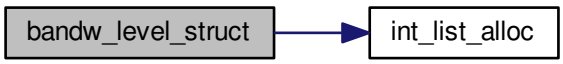
The bandw_level_struct function defines the level structure rooted at root for a given subgraph.

L has to be an empty int_list structure. L is initialized and its memory is allocated in this function.
$hint$ has to be provided by the calling routine and has a length of $n2 - n1$.

Attention

Interal use only.

Here is the call graph for this function:



10.184.2.3 static int bandw_mark_connected (mess_int_t *label*, mess_int_t *start*, mess_int_t * *adix*, mess_int_t * *adel*, mess_int_t *n1*, mess_int_t * *marked*) [static]

Parameters

in	*label*	input label of start vertex and all connected vertices
in	*start*	input vertex to start labeling
in	*adix*	input vector marking beginning and end of each node in adjacency list in $adel$
in	*adel*	input vector defining adjacency of nodes together with $adix$
in	*n1*	input offset of lowest node number in considered subgraph
in,out	*marked*	input/output vector containing marked vertices

Returns

zero on success or a non-zero error value otherwise

The bandw_mark_connected function marks the start vertex $start$ and all vertices connected to start with $label$ and stores them in $marked$.
The ouput vector $marked$ has a length of the number of vertices .

Attention

Internal use only.

10.184.2.4 static int bandw_number_adj_vert (struct int_list ∗ *ad*, struct int_list ∗ *Lv*, mess_int_t ∗ *level*, mess_int_t *l1*, mess_int_t *l2*, mess_int_t *n1*, mess_int_t *n2*, mess_int_t ∗ *perm*, mess_int_t ∗ *iperm*, mess_int_t ∗ *current*, mess_int_t ∗ *sorter*) [static]

Parameters

in	*ad*	input adjacency list of whole graph
in	*Lv*	input level structure
in	*level*	input vector specifying level for each node
in	*l1*	input start level
in	*l2*	input offset of lowest node number in considered subgraph
in	*n1*	input index of first node of connected subgraph
in	*n2*	input index of last node $+1$ of connected subgraph
in,out	*perm*	input/output integer vector of permutation
in,out	*iperm*	input/output inverse of $perm$
in,out	*current*	input/output current highest assigned number $+1$
out	*sorter*	output integer vector

Returns

zero on success or a non-zero error value otherwise

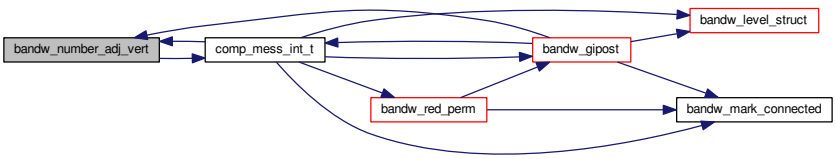
The bandw_number_adj_vert function starts with a vertex in level $l1$ of lowest assigned number, assigns all adjecent nodes in level $l2$ (which have not been assigned a number yet) a number in order of increasing degree, moves to the next until no further are left.

The permutation vector $perm$ has a length of the number of vertices, where for every i with $perm[i]! = -1$ $perm[i]$ is considered already given a number.
$current$ is updated every time a vertex is assigned a number.

Attention

Internal use only.

Here is the call graph for this function:



10.184.2.5 static int bandw_red_perm (mess_int_t *nr*, mess_int_t * *adix*, mess_int_t * *adel*, mess_int_t * *perm*, mess_int_t * *iperm*, mess_int_t * *bandw*) [static]

Parameters

in	*nr*	input number of vertices in graph
in	*adix*	input index of adjacency list
in	*adel*	input element vector of adjacency list
out	*perm*	output bandwidth reducing permuation
out	*iperm*	output inverse permutation of $perm$
out	*bandw*	output resulting bandwidth

Returns

zero on success or a non-zero error value otherwise

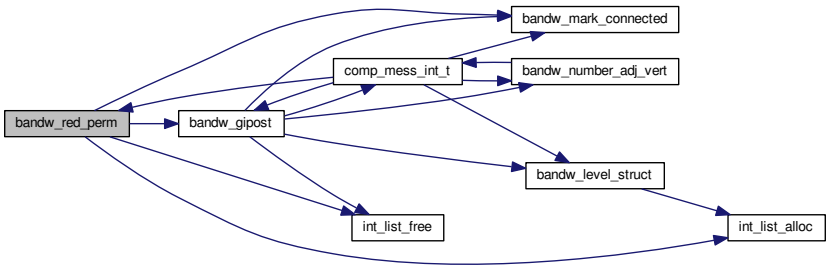
The bandw_red_perm function calls the Gibbs-Poole-Stockmeyer modification of the Cuthill-McKee algorithm to produce a bandwith reducing permutaion $perm$ for a given adjacency list $adel$ of a sparse matrix.

The element vector of the adjacency list $adel$ need to be a full adjacency list, that means it has to contain edges (i, j) and (j, i).
The permuation $perm$ and its inverse permutation $iperm$ are given as integer vectors of length nr.

Attention

Internal use only.

Here is the call graph for this function:



10.184.2.6 static int comp_mess_int_t (const void * *a1*, const void * *b1*) [static]

Parameters

in	*a1*	input pointer to first integer a_1
in	*b1*	input pointer to second integer b_1

Returns

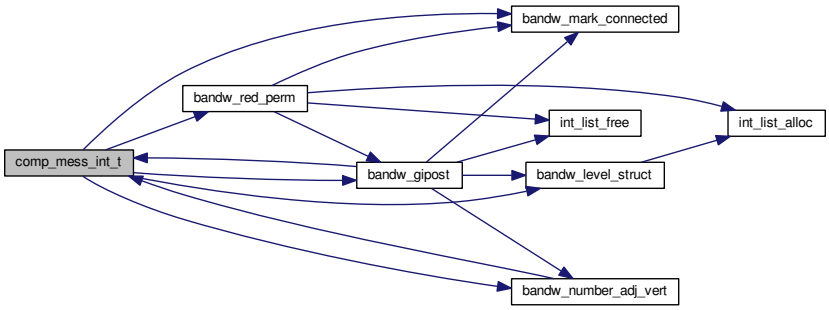
$a_1 - b_1$

The comp_mess_int_t function is a helper function for qsort and compares two integers.
This function returns $a_1 - b_1$.

Attention

Interal use only.

Here is the call graph for this function:



10.184.2.7 static int int_list_alloc (struct int_list * *L*, mess_int_t *ixnmax*, mess_int_t *elnmax*) [static]

Parameters

out	*L*	output int_list structure
in	*ixnmax*	input maximal number of parts of the list
in	*elnmax*	input maximal number of elements in the list

Returns

zero on success or a non-zero error value otherwise

The int_list_alloc function allocates memory in the int_list structure for a given size defined by the maximal number of parts $ixnmax$ and the maximal number of elements $elnmax$.

10.184.2.8 static void int_list_free (struct int_list * *L*) [static]

Parameters

in,out	*L*	input/output int_list structure

Returns

Nothing The int_list_free function frees memory in the int_list structure, so it can be freed.

10.185 dae1.c File Reference

Functions

- mess_equation eqn_conv_lyap_dae1 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric_dae1 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)

10.185.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.186 dae1.c File Reference

Functions

- mess_equation eqn_conv_lyap_dae1 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati_dae1 (PyObject *obj, mess_freelist mem)

10.186.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.187 dae2.c File Reference

Functions

- mess_equation eqn_conv_lyap_dae2 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric_dae2 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)

10.187.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.188 dae2.c File Reference

Functions

- mess_equation eqn_conv_lyap_dae2 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati_dae2 (PyObject *obj, mess_freelist mem)

10.188.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.189 DartConfiguration.tcl File Reference

10.190 DCPX_T.c File Reference

Macros

- #define SIZE (sizeof(double _Complex))

Functions

- int main (int argc, char *argv[ ])

Variables

- char info_size [ ]

10.190.1 Macro Definition Documentation

10.190.1.1 #define SIZE (sizeof(double _Complex))

10.190.2 Function Documentation

10.190.2.1 int main (int *argc*, char * *argv[ ]*)

10.190.3 Variable Documentation

10.190.3.1 char info_size[ ]

Initial value:

```
=  {'I', 'N', 'F', 'O', ':', 's','i','z','e','[',
  ('0' + (( (sizeof(double _Complex))  / 10000)%10)),
  ('0' + (( (sizeof(double _Complex))  / 1000)%10)),
  ('0' + (( (sizeof(double _Complex))  / 100)%10)),
  ('0' + (( (sizeof(double _Complex))  / 10)%10)),
  ('0' +  ( (sizeof(double _Complex))     % 10)),
  ']',


  '\0'}
```

10.191 decomp.c File Reference

Functions

- int mess_matrix_decomp (mess_matrix A, mess_matrix Asym, mess_matrix Askewsym)

  Decompose a matrix in a symmetric (hermitian) and skewsymmetric (skewhermitian) part.

10.191.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.192 default_rom_chooser.c File Reference

Functions

- int mess_bt_chooseorder_default (mess_vector SIGMA, double *tol, mess_int_t *maxr, mess_int_t FOM, void *aux)

  Choose dimension of reduced order model adaptively for balanced truncation (BT).
- int mess_bt_chooseorder_minreal (mess_vector SIGMA, double *tol, mess_int_t *maxr, mess_int_t FOM, void *aux)

  Choose dimension of reduced order model adaptively for balanced truncation (BT) with cut of of nearly zero singular values.

10.192.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.193 dense.c File Reference

Functions

- static int __gaxpy_dense (mess_matrix matrix, mess_vector x, mess_vector y)

  Dense matrix-vector product for general matrices in MESS_DENSE fomart.
- static int __gaxpyh_dense (mess_matrix matrix, mess_vector x, mess_vector y)

  Hermitian transposed dense matrix-vector product for general matrices in MESS_DENSE fomart.
- static int __gaxpyt_dense (mess_matrix matrix, mess_vector x, mess_vector y)

  Transpose dense matrix-vector product for general matrices in DENSE fomart.

10.193.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.193.2 Function Documentation

10.193.2.1 static int __gaxpy_dense (mess_matrix *matrix*, mess_vector *x*, mess_vector *y*) [static]

Parameters

in	*matrix*	input matrix in DENSE storage
in	*x*	input right hand side vector
in,out	*y*	input/output vector

Returns

zero on success or a non zero error code

The __gaxpy_dense function computes

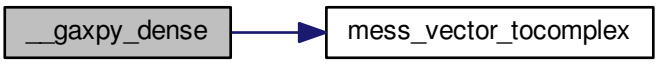
$$y \leftarrow Ax + y$$

for a general matrix in MESS_DENSE format. Because this is only the kernel function it does not check any input or output arguments.

Attention

Internal use only.

Here is the call graph for this function:



10.193.2.2 static int __gaxpyh_dense (mess_matrix *matrix*, mess_vector *x*, mess_vector *y*) [static]

Parameters

in	*matrix*	input matrix in DENSE storage
in	*x*	input right hand side vector
in,out	*y*	input/output vector

Returns

zero on success or a non zero error code

The __gaxpyh_dense function computes

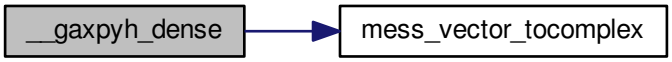
$$y \leftarrow A^H x + y$$

for a general matrix in MESS_DENSE format. Because this is only the kernel function it does not check any input or output arguments.

Attention

Internal use only.

Here is the call graph for this function:



10.193.2.3 static int __gaxpyt_dense (mess_matrix *matrix*, mess_vector *x*, mess_vector *y*) [static]

Parameters

in	*matrix*	input matrix in DENSE storage
in	*x*	input right hand side vector
in,out	*y*	output vector

Returns

zero on success or a non zero error code

The __gaxpyt_dense function computes

$$y \leftarrow A^T x + y$$

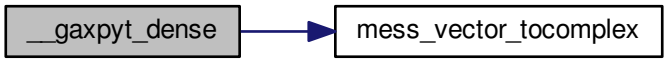
for a general matrix in MESS_DENSE format. If the matrix is real this function is the same as __gaxpyh_dense but in the case of a complex matrix it only transposes it without using the complex conjugate values.

Because this is only the kernel function it does not check any input or output arguments.

Attention

Internal use only.

Here is the call graph for this function:



10.194 dense_nm.c File Reference

Macros

- #define MIN_LINESEARCH_PARA 1e-8
- #define LINESEARCH_BREAKDOWN 1e-8
- #define MAXIT 50
- #define RELRES_TOL sqrt(mess_eps())
- #define ABSRES_TOL 1e-10

Functions

- int mess_dense_nm_gmpare (mess_matrix X0, mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_int_t plus, mess_int_t linesearch, mess_operation_t trans, mess_int_t maxit, mess_norm_t nrm_t, double absres_tol, double relres_tol, double ∗absres, double ∗relres, mess_matrix RES, mess_matrix X)

  Implementation of the dense Newton method to solve the Riccati Equation $A^TXE + EXA + / - XGX + Q = 0$.
- int mess_dense_nm_gmare (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X)

  Implementation of the dense Newton-method to solve the generalized Riccati Equation $A^TXE + E^TXA - E^TXGXE + Q = 0$.
- int mess_dense_nm_gpare (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X)

  Implementation of the dense Newton-method to solve the generalized Riccati Equation $A^TXE + E^TXA + E^TXGXE + Q = 0$.
- int mess_dense_res_gmpare (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X, mess_int_t plus, mess_operation_t trans, mess_norm_t nrm_t, double ∗res, double ∗rel)

  Compute the residual of the algebraic (generalized) Riccati (or Lyapunov) equation.

10.194.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Robert Dykstra (dykstra@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.194.2 Macro Definition Documentation

10.194.2.1 #define ABSRES_TOL 1e-10

10.194.2.2 #define LINESEARCH_BREAKDOWN 1e-8

10.194.2.3 #define MAXIT 50

10.194.2.4 #define MIN_LINESEARCH_PARA 1e-8

10.194.2.5 #define RELRES_TOL sqrt(mess_eps())

10.195 dense_nm_gmpare.c File Reference

Functions

- PyObject ∗ pymess_dense_nm_gmpare (PyObject ∗self, PyObject ∗args)

  Wrapper around the interface mess_dense_nm_gmpare for use with Python.

10.195.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Robert Dykstra (dykstra@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.196 dfs.c File Reference

Macros

- #define FLIP(x) (-(x)-2)
- #define UNFLIP(x) (((x) < 0)? FLIP(x):(x))
- #define MARKED(w, x) (w[(x)] < 0)
- #define MARK(w, x) { w[(x)] = FLIP(w[(x)]); }

Functions

- int mess_graph_dfs (mess_matrix G, mess_int_t j, mess_int_t *top, mess_int_t *xi, mess_int_t *pstack, const mess_int_t *pinv)

  Perform a depth-first-search on a directed graph defined by a matrix.
- int mess_graph_reach (mess_matrix G, mess_matrix B, mess_int_t k, mess_int_t *top, mess_int_t *xi, const mess_int_t *pinv)

  Compute the reachability set from a node in a graph.

10.196.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC) The functions are based on [13].

10.196.2 Macro Definition Documentation

10.196.2.1 #define FLIP(x) (-(x)-2)

10.196.2.2 #define MARK(w, x) { w[(x)] = FLIP(w[(x)]); }

10.196.2.3 #define MARKED(w, x) (w[(x)] < 0)

10.196.2.4 #define UNFLIP(x) (((x) < 0)? FLIP(x):(x))

10.197 dgeqp3.c File Reference

Functions

- int mess_direct_dgeqp3 (mess_matrix A, mess_matrix Q, mess_matrix R, mess_int_t *perm)

  Interface for LAPACK *DGEQP3.*

10.197.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.198 diag.c File Reference

Functions

- int mess_matrix_diag (mess_matrix matrix, mess_vector d)

  Get the main diagonal elements of from matrix.
- int mess_matrix_diagpos (mess_matrix matrix, mess_int_t *pos)

  Find the position of the main diagonal elements.
- int mess_matrix_diag_from_vector (mess_vector v, mess_matrix diag)

  Create a diagonal mess_matrix from mess_vector.

10.198.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.199 diag.c File Reference

Functions

- static int __mess_precond_diag_solve (mess_precond myself, mess_solver_options opt, mess_vector in, mess_vector out)

  Solves a linear system arises from a diagonal preconditioner (real case).
- static int __mess_precond_diag_solvecomplex (mess_precond myself, mess_solver_options opt, mess_vector in, mess_vector out)

  Solves a linear system arises from a diagonal preconditioner (complex case).
- static int __mess_precond_diag_clear (mess_precond myself)

  Clear a mess_precond structure (real case).
- static int __mess_precond_diag_complexclear (mess_precond myself)

  Clear a mess_precond structure (complex case).
- int mess_precond_diag (mess_precond pre, mess_matrix mat)

  Setup a diagonal preconditioner.

10.199.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.199.2 Function Documentation

10.199.2.1 static int __mess_precond_diag_clear (mess_precond *myself*) [static]

Parameters

in,out	*myself*	input/output mess_precond structure to clear

Returns

zero always

The __mess_precond_diag_clear function clears a mess_precond structure (real case).

Attention

Internal use only.

10.199.2.2 static int __mess_precond_diag_complexclear (mess_precond *myself*) [static]

Parameters

in,out	*myself*	input/output mess_precond structure to clear

Returns

zero always

The __mess_precond_diag_complexclear function clears a mess_precond structure (complex case).

Attention

Internal use only.

10.199.2.3 static int __mess_precond_diag_solve (mess_precond *myself*, mess_solver_options *opt*, mess_vector *in*, mess_vector *out*) [static]

Parameters

in	*myself*	input mess_precond diagonal preconditioner
in	*opt*	input mess_solver_options structure
in	*in*	input the right hand side?
out	*out*	output mess_vector structure contains the solution

Returns

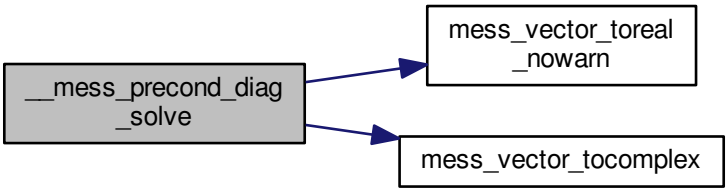
zero on success or a non-zero error value otherwise

The __mess_precond_diag_solve function solves a linear System arises from a diagonal preconditioner (real case).

Attention

Internal use only.

Here is the call graph for this function:



10.199.2.4 static int __mess_precond_diag_solvecomplex (mess_precond *myself*, mess_solver_options *opt*, mess_vector *in*, mess_vector *out*) [static]

Parameters

in	*myself*	input mess_precond diagonal preconditioner
in	*opt*	input mess_solver_options structure
in	*in*	input the right hand side?
out	*out*	output mess_vector structure contains the solution

Returns

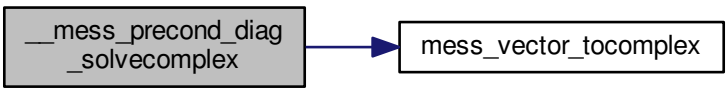
zero on success or a non-zero error value otherwise

The __mess_precond_diag_solvecomplex function solves a linear System arises from a diagonal preconditioner (complex case).

Attention

Internal use only.

Here is the call graph for this function:



10.200 direct.c File Reference

Functions

- int mess_direct_init (mess_direct ∗direct)

  Initialize a mess_direct object.
- int mess_direct_clear (mess_direct ∗direct)

  Clear a mess_direct object.
- int mess_direct_solvem (mess_operation_t op, mess_direct solver, mess_matrix B, mess_matrix X)

  Solve $op(A)X = B$, where B and X are matrices.
- int mess_direct_solve (mess_operation_t op, mess_direct solver, mess_vector b, mess_vector x)

  Solve $op(A)x = b$, where b and x are vectors.
- int mess_direct_determinant (mess_direct solver, double ∗m, double ∗e)

  Calculate the determinant of a matrix.
- int mess_direct_determinantc (mess_direct solver, double ∗mr, double ∗mi, double ∗e)

  Calculate the determinant of a complex matrix.
- int mess_direct_getL (mess_direct solver, mess_matrix L)

  Retrieve the left matrix of a decomposition.
- int mess_direct_getU (mess_direct solver, mess_matrix U)

  Retrieve the right matrix of a decomposition.
- int mess_direct_getpermp (mess_direct solver, mess_int_t ∗p)

  Get the row permutation of a decomposition.
- int mess_direct_getpermq (mess_direct solver, mess_int_t ∗q)

  Get the column permutation of a decomposition.
- int mess_direct_getscalerow (mess_direct solver, mess_vector r)

  Get the row scaling factors of a decomposition.
- int mess_direct_getscalecol (mess_direct solver, mess_vector c)

  Get the col scaling factors of a decomposition.
- int mess_direct_inverse (mess_direct solver, mess_matrix inv)

  Compute the inverse of a matrix.
- int mess_direct_lu (mess_matrix matrix, mess_direct solver)

  Compute a LU decomposition of a matrix.
- int mess_direct_lu_select (mess_direct_lupackage_t lu)

  Change the default behaviour of mess_direct_lu.
- int mess_direct_chol (mess_matrix matrix, mess_direct solver)

  Compute an Chol decomposition of a matrix.
- int mess_direct_chol_select (mess_direct_cholpackage_t chol)

  Change the default behaviour of mess_direct_chol.

Variables

- static mess_direct_lupackage_t lu_type = MESS_DIRECT_DEFAULT_LU
- static pthread_mutex_t lu_type_mutex = PTHREAD_MUTEX_INITIALIZER
- static mess_direct_cholpackage_t chol_type = MESS_DIRECT_DEFAULT_CHOLESKY
- static pthread_mutex_t chol_type_mutex = PTHREAD_MUTEX_INITIALIZER

10.200.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.200.2 Variable Documentation

10.200.2.1 mess_direct_cholpackage_t chol_type = MESS_DIRECT_DEFAULT_CHOLESKY [static]

10.200.2.2 pthread_mutex_t chol_type_mutex = PTHREAD_MUTEX_INITIALIZER [static]

10.200.2.3 mess_direct_lupackage_t lu_type = MESS_DIRECT_DEFAULT_LU [static]

10.200.2.4 pthread_mutex_t lu_type_mutex = PTHREAD_MUTEX_INITIALIZER [static]

10.201 direct.h File Reference

Data Structures

- struct mess_direct_st

  Representation of a generic direct solver.
- struct mess_multidirect_st

  Representation of a generic direct solver.
- struct mess_direct_levelset_st

Macros

- #define MESS_HAVE_SOLVE(solver) ((solver)->solve != NULL)
- #define MESS_HAVE_SOLVET(solver) ((solver)->solvet != NULL)
- #define MESS_HAVE_SOLVEH(solver) ((solver)->solveh != NULL)
- #define MESS_HAVE_SOLVEM(solver) ((solver)->solvem != NULL)
- #define MESS_HAVE_SOLVEMT(solver) ((solver)->solvemt != NULL)
- #define MESS_HAVE_SOLVEMH(solver) ((solver)->solvemh != NULL)
- #define MESS_HAVE_GETL(solver) ((solver)->getL != NULL)
- #define MESS_HAVE_GETU(solver) ((solver)->getU != NULL)
- #define MESS_HAVE_GETPERMP(solver) ((solver)->getpermp != NULL)
- #define MESS_HAVE_GETPERMQ(solver) ((solver)->getpermq != NULL)
- #define MESS_HAVE_GETSCALEROW(solver) ((solver)->getscalerow != NULL)
- #define MESS_HAVE_GETSCALECOL(solver) ((solver)->getscalecol != NULL)
- #define MESS_HAVE_DET(solver) ((solver)->det != NULL)
- #define MESS_HAVE_DETC(solver) ((solver)->det != NULL)
- #define MESS_HAVE_INVERSE(solver) ((solver)->inverse != NULL)

Typedefs

- typedef struct mess_direct_st * mess_direct

  Type definition to hide the pointer on the mess_direct_st structure.
- typedef struct mess_multidirect_st * mess_multidirect

  Type definition to hide one pointer of the mess_multidirect_st structure.
- typedef struct mess_direct_levelset_st mess_direct_levelset

Enumerations

Functions

- int mess_solver_usolve (mess_matrix U, mess_vector y)

  Solve $Ux = b$ (user interface).
- int mess_solver_usolvem (mess_matrix U, mess_matrix Y)

  Solve $UX = B$ (user interface).
- int mess_solver_usolve_kernelcsr_real (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, double *y)

  Solve $Ux = b$ (CSR, kernel only, real).
- int mess_solver_usolve_kernelcsr_real_complex (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $Ux = b$ (CSR, kernel only, real U, complex y).
- int mess_solver_usolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $Ux = b$ (CSR, kernel only, complex).
- int mess_solver_lsolve (mess_matrix L, mess_vector y)

  Solve $Lx = b$ (user interface).
- int mess_solver_lsolvem (mess_matrix L, mess_matrix Y)

  Solve $LX = B$ (user interface).
- int mess_solver_lsolve_kernelcsr_real (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, double *y)

  Solve $Lx = b$ (CSR, kernel only, real).
- int mess_solver_lsolve_kernelcsr_real_complex (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $Lx = b$ (CSR, kernel only, real L, complex b).
- int mess_solver_lsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $Lx = b$ (CSR, kernel only, complex).
- int mess_solver_lusolvem_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *lval, mess_int_t *lcolptr, mess_int_t *lrowptr, mess_double_cpx_t *uval, mess_int_t *ucolptr, mess_int_t *urowptr, mess_int_t *p, mess_int_t *q, mess_matrix b, mess_matrix x)

  LU solver for complex CSR stored L and U.
- int mess_solver_ltsolve (mess_matrix L, mess_vector y)

  Solve $L^T x = b$ (user interface).
- int mess_solver_ltsolvem (mess_matrix L, mess_matrix Y)

  Solve $L^T X = B$ (user interface).
- int mess_solver_lhsolve (mess_matrix L, mess_vector y)

  Solve $L^H x = b$ (user interface).
- int mess_solver_lhsolvem (mess_matrix L, mess_matrix Y)

  Solve $L^H X = B$ (user interface).
- int mess_solver_ltsolve_kernelcsr_real (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, double *y)

  Solve $L^T x = b$ (CSR, kernel only, real).
- int mess_solver_ltsolve_kernelcsr_real_complex (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $L^T x = b$ (CSR, kernel only, real L, complex y).
- int mess_solver_utsolve_kernelcsr_real (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, double *x)

  Solve $U^T x = b$ (CSR, kernel only, real).

- int mess_solver_utsolve_kernelcsr_real_complex (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *x)

  Solve $U^T x = b$ (CSR, kernel only, real U, complex y).
- int mess_solver_ltsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $L^T x = b$ (CSR, kernel only, complex).
- int mess_solver_lhsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $L^H x = b$ (CSR, kernel only, complex).
- int mess_solver_utsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *x)

  Solve $U^T x = b$ (CSR, kernel only, complex).
- int mess_solver_uhsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *x)

  Solve $U^H x = b$ (CSR, kernel only, complex).
- int mess_solver_utsolve (mess_matrix U, mess_vector y)

  Solve $U^T x = b$ (user interface).
- int mess_solver_uhsolve (mess_matrix U, mess_vector y)

  Solve $U^H x = b$ (user interface).
- int mess_solver_utsolvem (mess_matrix U, mess_matrix y)

  Solve $U^T X = B$ (user interface).
- int mess_solver_uhsolvem (mess_matrix U, mess_matrix y)

  Solve $U^H X = B$ (user interface).
- int mess_direct_init (mess_direct *direct)

  Initialize a mess_direct object.
- int mess_direct_clear (mess_direct *direct)

  Clear a mess_direct object.
- int mess_direct_solve (mess_operation_t op, mess_direct solver, mess_vector b, mess_vector x)

  Solve $op(A)x = b$, where b and x are vectors.
- int mess_direct_solvem (mess_operation_t op, mess_direct solver, mess_matrix b, mess_matrix x)

  Solve $op(A)X = B$, where B and X are matrices.
- int mess_direct_determinant (mess_direct solver, double *m, double *e)

  Calculate the determinant of a matrix.
- int mess_direct_determinantc (mess_direct solver, double *mr, double *mi, double *e)

  Calculate the determinant of a complex matrix.
- int mess_direct_getL (mess_direct solver, mess_matrix L)

  Retrieve the left matrix of a decomposition.
- int mess_direct_getU (mess_direct solver, mess_matrix U)

  Retrieve the right matrix of a decomposition.
- int mess_direct_getpermp (mess_direct solver, mess_int_t *p)

  Get the row permutation of a decomposition.
- int mess_direct_getpermq (mess_direct solver, mess_int_t *q)

  Get the column permutation of a decomposition.
- int mess_direct_getscalerow (mess_direct solver, mess_vector r)

  Get the row scaling factors of a decomposition.
- int mess_direct_getscalecol (mess_direct solver, mess_vector c)

  Get the col scaling factors of a decomposition.
- int mess_direct_inverse (mess_direct solver, mess_matrix inv)

  Compute the inverse of a matrix.
- int mess_matrix_backslash (mess_operation_t op, mess_matrix A, mess_vector b, mess_vector x)

  Solve $op(A)x = b$ with b and x vectors.

- int mess_matrix_backslashm (mess_operation_t op, mess_matrix A, mess_matrix b, mess_matrix x)

  Solve $op(A)x = b$ with b and x matrices.
- int mess_matrix_pinv (mess_matrix A, mess_matrix Pinv)

  Compute the pseudoinverse of a dense matrix A.
- int mess_direct_lu (mess_matrix matrix, mess_direct direct)

  Compute a LU decomposition of a matrix.
- int mess_direct_lu_select (mess_direct_lupackage_t lupackage)

  Change the default behaviour of mess_direct_lu.
- int mess_direct_chol (mess_matrix matrix, mess_direct direct)

  Compute an Chol decomposition of a matrix.
- int mess_direct_chol_select (mess_direct_cholpackage_t cholpackage)

  Change the default behaviour of mess_direct_chol.
- int mess_direct_create_umfpack (mess_matrix matrix, mess_direct direct)

  Generate a direct linear solver for standard linear systems $Ax = b$ with `UMFPACK`
- int mess_direct_create_umfpack_control (mess_matrix matrix, mess_direct direct, double *Control)

  Generate a direct linear solver for standard linear systems $Ax = b$ with `UMFPACK`.
- int mess_direct_create_csparse_lu (mess_matrix matrix, mess_direct direct)

  Generate a CXSparse LU solver.
- int mess_direct_create_csparse_cholesky (mess_matrix matrix, mess_direct direct)

  Generate a CXSparse Cholesky solver.
- int mess_direct_create_banded (mess_matrix matrix, mess_direct direct)

  Generate a direct linear banded solver for standard linear systems $Ax = b$ with LAPACK.
- int mess_direct_create_bicgstab (mess_matrix matrix, mess_direct solver)

  Generate a bicgstab solver in order to use it as direct solver.
- int mess_direct_create_lapack_lu (mess_matrix matrix, mess_direct solver)

  Generate a dense `LAPACK` *based LU-solver.*
- int mess_direct_create_cholesky (mess_matrix matrix, mess_direct solver)

  Generate a dense `LAPACK` *based Cholesky solver.*
- int mess_direct_create_lapack_qr (mess_matrix matrix, mess_direct solver)

  Construct a solver based on QR or LQ decomposition from `LAPACK`.
- int mess_direct_create_sparse_lu (mess_matrix A, mess_direct sol)

  Compute a sparse left-looking LU decomposition.
- int mess_direct_create_hessenberg_lu (mess_matrix matrix, mess_direct solver)

  Generate a dense `LAPACK` *based LU-solver for upper hessenberg matrices.*
- int mess_direct_splsolve (mess_matrix G, mess_matrix B, mess_int_t k, mess_int_t *top, mess_int_t *xi, double *x, const mess_int_t *pinv)

  Solve $x = G^{-1}B(:,k)$ with all sparse.
- int mess_direct_splsolvec (mess_matrix G, mess_matrix B, mess_int_t k, mess_int_t *top, mess_int_t *xi, mess_double_cpx_t *x, const mess_int_t *pinv)

  Solve $x = G^{-1}B(:,k)$ with all sparse with complex data.
- int mess_decomp_lureuse_kernelcsr (mess_int_t rows, mess_int_t cols, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_int_t *lcolptr, mess_int_t *lrowptr, mess_int_t *ucolptr, mess_int_t *urowptr, mess_int_t *p, mess_int_t *q, double *lvalues, double *uvalues)
- int mess_decomp_lureuse_kernelcsr_complex (mess_int_t rows, mess_int_t cols, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_int_t *lcolptr, mess_int_t *lrowptr, mess_int_t *ucolptr, mess_int_t *urowptr, mess_int_t *p, mess_int_t *q, mess_double_cpx_t *lvalues, mess_double_cpx_t *uvalues)
- int mess_decomp_lureuse_kernelcsr_flopcount (mess_int_t rows, mess_int_t cols, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_int_t *lcolptr, mess_int_t *lrowptr, mess_int_t *ucolptr, mess_int_t *urowptr, mess_int_t *p, mess_int_t *q, double *lvalues, double *uvalues, mess_int_t *flops)
- int mess_direct_create_generalized_lyapunovchol (mess_matrix A, mess_matrix E, mess_direct solver)

  Generate a dense standard / generalized Lyapunov equation solver for the cholesky factor.

- int mess_direct_create_generalized_lyapunov (mess_matrix A, mess_matrix E, mess_direct solver)

  Generate a dense standard / generalized Lyapunov equation solver.
- int mess_direct_create_generalized_stein (mess_matrix A, mess_matrix E, mess_direct solver)

  Generate a dense standard / generalized Stein equation solver.
- int mess_direct_care (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X)

  Compute the solution of a (generalized) Riccati Equation using a dense newton method.
- int mess_direct_care_sign (mess_matrix A, mess_matrix Q, mess_matrix G, mess_matrix X)
- int mess_direct_create_sylvester_sparsedense (mess_matrix A, mess_matrix F, mess_matrix E, mess_matrix H, mess_direct solver)

  Generate a solver for a sparse-dense Sylvester equation with a small second matrix.
- int mess_direct_create_sylvester_dense (mess_matrix A, mess_matrix F, mess_matrix E, mess_matrix H, mess_direct solver)

  Generate a solver for a dense Sylvester equation using `LAPACK`.
- int mess_direct_dgeqp3 (mess_matrix A, mess_matrix Q, mess_matrix R, mess_int_t *perm)

  Interface for `LAPACK` *DGEQP3.*
- int mess_direct_cholfactor (mess_matrix ZZ, mess_matrix Z)

  Compute a Cholesky factor of a given matrix.
- int mess_direct_res2 (mess_operation_t op, mess_matrix A, mess_vector x, mess_vector b, double *resid, double *relresid)

  Compute the (relative) residual of $Ax = b$.
- int mess_direct_res2m (mess_operation_t op, mess_matrix A, mess_matrix x, mess_matrix b, double *resid, double *relresid)

  Compute the (relative) residual of $Ax = b$.
- int mess_direct_sylvester_res2 (mess_operation_t op, mess_matrix A, mess_matrix H, mess_matrix M, mess_matrix X, double *res)

  Compute the 2*-norm residual of a standard Sylvester equation.*
- int mess_direct_sylvestersg_res2 (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix H, mess_matrix M, mess_matrix X, double *res)

  Compute the 2*-norm residual of a semi-generalized Sylvester equation.*
- int mess_direct_generalized_sylvester_res2 (mess_operation_t op, mess_matrix A, mess_matrix F, mess_matrix E, mess_matrix H, mess_matrix M, mess_matrix X, double *res)

  Compute the 2*-norm residual of a generalized Sylvester equation.*
- int mess_direct_lyapunov_res2 (mess_operation_t op, mess_matrix A, mess_matrix M, mess_matrix X, double *res)

  Compute the 2*-norm residual of a Lyapunov Equation (DENSE).*
- int mess_direct_generalized_lyapunov_res2 (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix M, mess_matrix X, double *res2)

  Compute the 2*-norm residual of a generalized Lyapunov Equation.*
- int mess_direct_stein_res2 (mess_operation_t op, mess_matrix A, mess_matrix M, mess_matrix X, double *res2)

  Compute the 2*-norm residual of a Stein Equation (dense).*
- int mess_direct_generalized_stein_res2 (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix M, mess_matrix X, double *res2)

  Compute the 2*-norm residual of a generalized Stein Equation.*
- int mess_direct_care_res2 (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X, double *res2)

  Compute the 2*-norm residual of the algebraic (generalized) Riccati equation.*
- int mess_multidirect_init (mess_multidirect *direct)

  Initialize a mess_multidirect object.
- int mess_multidirect_clear (mess_multidirect *direct)

  Clear a mess_multidirect object.
- int mess_multidirect_solve (mess_operation_t op, mess_multidirect solver, mess_int_t ind, mess_vector b, mess_vector x)

Solve $op(A(i))x = b$.

- int mess_multidirect_solvem (mess_operation_t op, mess_multidirect solver, mess_int_t ind, mess_matrix b, mess_matrix x)

  Solve $op(A(i))X = B$.

- int mess_multidirect_create_umfpack (mess_matrix matrix, mess_vector shiftsl, mess_vector shiftsr, mess_multidirect mlu, mess_direct base, mess_matrix shiftmatrix)

  Create a multisolver on base of UMFPACK.

- int mess_multidirect_create_sparse_lu (mess_matrix matrix, mess_vector shiftsl, mess_vector shiftsr, mess_multidirect mlu, mess_direct base, mess_matrix shiftmatrix)

  Create a single-pattern-multi-value LU decomposition of $(sA + pI)$ *or* $(sA + pE)$.

- int mess_multidirect_create (mess_matrix matrix, mess_vector shiftsl, mess_vector shiftsr, mess_multidirect mlu, mess_direct base, mess_matrix shiftmatrix)

  Create a multisolver for shifted linear systems $(sA + pI)$ *or* $(sA + pE)$.

- int mess_multidirect_select (mess_multidirect_t solver)

  Select the multisolver which is used by mess_multidirect_create.

- int mess_matrix_norminvest (mess_direct decomp, double *nrm)

  Estimate 1*-norm of the inverse of a real square matrix.*

- int mess_matrix_condest (mess_matrix A, double *nrm)

  Compute a lower bound for the 1*-norm condition number of a real square matrix.*

10.201.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.202 direct_chol_select.c File Reference

Functions

- PyObject * pymess_direct_chol_select (PyObject *self, PyObject *args)

  Wrapper around mess_direct_chol_select for use with Python.

10.202.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.203 direct_select.c File Reference

Functions

- PyObject * pymess_direct_select (PyObject *self, PyObject *args)

  Wrapper around mess_direct_lu_select for use with Python.

10.203.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.204 doxy_main.c File Reference

10.205 dupl.c File Reference

Functions

- int mess_matrix_dupl (mess_matrix mat)

  Detect and sum up duplicate entries in a sparse matrix.
- int mess_matrix_eliminate_zeros (mess_matrix mat)

  Remove zero entries in a sparse matrix.

10.205.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.206 dynorm.c File Reference

Functions

- int mess_matrix_dynorm2 (mess_matrix A, double *nrm)

  Compute the 2*-norm of* $A^H A$ *or* AA^H.
- int mess_matrix_dynormf2 (mess_matrix A, double *nrm)

  Compute the squared Frobenius-norm of $A^H A$ *or* AA^H.
- int mess_matrix_dynormf (mess_matrix A, double *nrm)

  Compute the Frobenius-norm of $A^H A$ *or* AA^H.
- int mess_matrix_indefinite_dynorm2 (mess_matrix W, mess_matrix K, double *nrm)

  Compute the 2*-norm of* $WW^T - KK^T$, $WW^H - KK^T$, $WW^T - KK^H$ *or* $WW^H - KK^H$.
- int mess_matrix_indefinite_dynormf2 (mess_matrix W, mess_matrix K, double *nrm)

  Compute the square of the Frobenius-norm of $WW^T - KK^T$, $WW^H - KK^T$, $WW^T - KK^H$ *or* $WW^H - KK^H$.
- int mess_matrix_indefinite_dynormf (mess_matrix W, mess_matrix K, double *nrm)

  Compute the Frobenius-norm of $WW^T - KK^T$, $WW^H - KK^T$, $WW^T - KK^H$ *or* $WW^H - KK^H$.

10.206.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.207 dynsys.h File Reference

Data Structures

- struct mess_dynsys_st

  Representation of a dynamical system.

Macros

- #define MESS_DYNSYS_LTI 0x01
- #define MESS_DYNSYS_GLTI 0x02
- #define MESS_DYNSYS_2ND 0x03
- #define MESS_IS_DYNSYS_LTI(x) ((∗x).type == MESS_DYNSYS_LTI)
- #define MESS_IS_DYNSYS_GLTI(x) ((∗x).type == MESS_DYNSYS_GLTI)
- #define MESS_IS_DYNSYS_2ND(x) ((∗x).type == MESS_DYNSYS_2ND)
- #define MESS_IS_DYNSYS_SISO(x) (((∗x).inputs==1) && ((∗x).outputs==1))

Typedefs

- typedef struct mess_dynsys_st ∗ mess_dynsys

  Type definition of easy usage of mess_dynsys_st.

Functions

- char ∗ mess_dynsysstr (unsigned short type)

  Return name of a dynamical system type.
- int mess_dynsys_init (mess_dynsys ∗sys)

  Initialize a mess_dynsys object.
- int mess_dynsys_clear (mess_dynsys ∗sys)

  Clean up a mess_dynsys object.
- int mess_dynsys_lti (mess_dynsys lti, mess_matrix A, mess_matrix B, mess_matrix C)

  Generate a LTI system from matrices A, B and C.
- int mess_dynsys_lti_copy (mess_dynsys lti, mess_matrix A, mess_matrix B, mess_matrix C)

  Generate a LTI system from A, B and C (copy version).
- int mess_dynsys_glti (mess_dynsys lti, mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix E)

  Generate a generalized LTI system from matrices A, B, C and E.
- int mess_dynsys_glti_copy (mess_dynsys lti, mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix E)

  Generate a LTI system from A, B, C and E (copy version).
- int mess_dynsys_2nd (mess_dynsys sys, mess_matrix M, mess_matrix G, mess_matrix K, mess_matrix B, mess_matrix Cp, mess_matrix Cv)

  Generate a second order system from matrices M, G, K, B, Cp and Cv.
- int mess_dynsys_2nd_copy (mess_dynsys sys, mess_matrix M, mess_matrix G, mess_matrix K, mess_↩matrix B, mess_matrix Cp, mess_matrix Cv)

  Generate a second order system from matrices M, G, K, B, Cp and Cv (copy version).
- int mess_dynsys_2nd_to_1st (mess_dynsys sys2nd, mess_dynsys sys1st)

  Convert a second order system to a first order system.
- int mess_dynsys_printinfo (mess_dynsys sys)

  Print information about a dynamical system.
- int mess_dynsys_isstable (mess_dynsys sys, int ∗stable)

Check if a dynamical system is stable.

- int mess_dynsys_evaltransfer (mess_dynsys sys, double aa, double bb, mess_int_t nsample, mess_vector omega, mess_vector G, mess_matrix *Gout)

  Evaluate a transfer function.
- int mess_dynsys_transfer_diff (mess_int_t nsample, mess_matrix *Go, mess_matrix *Gr, double *err2, double *rel2, double *errinf, double *relinf, mess_vector diffvec)

  Compute difference and relative difference between two transfer functions.
- int mess_dynsys_project_to_lti (mess_dynsys sys, mess_matrix V, mess_matrix W, mess_dynsys red)

  Project a dynamical system (LTI or GLTI) to a LTI system.
- int mess_dynsys_project_to_glti (mess_dynsys sys, mess_matrix V, mess_matrix W, mess_dynsys red)

  Project a GLTI system to a GLTI system.
- int mess_dynsys_project_2nd_to_1st (mess_dynsys sys2nd, mess_matrix V, mess_matrix W, mess_dynsys sys1st)

  Project a second order system to a first order system.
- int mess_dynsys_project_2nd_to_2nd (mess_dynsys sys, mess_matrix V, mess_matrix W, mess_dynsys red)

  Project a second order system to a second order system.
- int mess_matrix_isstableg (mess_matrix A, mess_matrix E, int *stable)

  Check if a matrix pair is stable.
- int mess_matrix_isstable (mess_matrix A, int *stable)

  Check if a matrix is stable.

10.207.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.208 dynsysinit.c File Reference

Functions

- char * mess_dynsysstr (unsigned short type)

  Return name of a dynamical system type.
- int mess_dynsys_init (mess_dynsys *sys)

  Initialize a mess_dynsys object.
- int mess_dynsys_clear (mess_dynsys *sys)

  Clean up a mess_dynsys object.
- int mess_dynsys_lti (mess_dynsys lti, mess_matrix A, mess_matrix B, mess_matrix C)

  Generate a LTI system from matrices A, B and C.
- int mess_dynsys_lti_copy (mess_dynsys lti, mess_matrix A, mess_matrix B, mess_matrix C)

  Generate a LTI system from A, B and C (copy version).
- int mess_dynsys_glti (mess_dynsys lti, mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix E)

  Generate a generalized LTI system from matrices A, B, C and E.
- int mess_dynsys_glti_copy (mess_dynsys lti, mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix E)

  Generate a LTI system from A, B, C and E (copy version).
- int mess_dynsys_2nd (mess_dynsys sys, mess_matrix M, mess_matrix G, mess_matrix K, mess_matrix B, mess_matrix Cp, mess_matrix Cv)

  Generate a second order system from matrices M, G, K, B, Cp and Cv.

- int mess_dynsys_2nd_copy (mess_dynsys sys, mess_matrix M, mess_matrix G, mess_matrix K, mess_matrix B, mess_matrix Cp, mess_matrix Cv)

  Generate a second order system from matrices M, G, K, B, Cp and Cv (copy version).
- int mess_dynsys_2nd_to_1st (mess_dynsys sys2nd, mess_dynsys sys1st)

  Convert a second order system to a first order system.
- int mess_dynsys_printinfo (mess_dynsys sys)

  Print information about a dynamical system.

10.208.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.209 easyfrontend.h File Reference

Functions

- int mess_lyap (mess_matrix A, mess_matrix E, mess_matrix B, mess_matrix Z)

  Frontend to compute a factorized solution of a (generalized) Lyapunov Equation.
- int mess_care (mess_matrix A, mess_matrix E, mess_matrix B, mess_matrix C, mess_matrix Z)

  Frontend to compute a factorized solution of a (generalized) Riccati Equation.
- int mess_glyap (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix Y, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix X)

  Solve a standard / generalized Lyapunov equation.
- int mess_gstein (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix Y, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix X)

  Solve a standard / generalized Stein equation.
- int mess_tglyap (mess_operation_t op, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix Y, mess_matrix X)

  Solve a standard / generalized Lyapunov equation from given Schur decomposition.
- int mess_tgstein (mess_operation_t op, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix Y, mess_matrix X)

  Solve a standard / generalized Stein equation from given Schur decomposition.

10.209.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.210 eig.c File Reference

Functions

- int mess_eigen_eig (mess_matrix A, mess_vector ew, mess_matrix ev)

  Compute all eigenvalues and eigenvectors.
- int mess_eigen_eigg (mess_matrix A, mess_matrix B, mess_vector ew, mess_vector alpha, mess_vector beta, mess_matrix ev)

  Compute all eigenvalues and right eigenvectors of the generalized eigenvalue problem.

10.210.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.211 eig_hessenberg.c File Reference

Functions

- int mess_eigen_hessenberg_schur (mess_matrix hess, mess_vector ev, mess_matrix Q, mess_matrix T)

  Compute Schur decomposition of an upoer Hessenberg matrix.
- int mess_eigen_hessenberg_abs_largest (mess_matrix hess, mess_vector evector, mess_double_cpx_t *largest_ev)

  Compute the largest eigenvalue in magnitude and corresponding eigenvector of a upper Hessenberg matrix,.

10.211.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.212 eigenvalue.h File Reference

Data Structures

- struct mess_eigen_arpack_options_t

  Options structure for ARPACK *.*

Macros

- #define MESS_EIGEN_ARPACK_DEFAULT { .tol = 0.0, .b0 = NULL, .ncv = 100, .maxit = 300, .which = MESS_EIGEN_ARPACK_LARGE_MAGNITUDE }

  Default initializer for the mess_eigen_arpack_options_t structure.

Enumerations

Functions

- int mess_eigen_arnoldi (mess_matrix A, mess_matrix B, mess_matrix K, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method with respect to A or $(A-UW)$.
- int mess_eigen_arnoldig (mess_matrix A, mess_matrix U, mess_matrix W, mess_direct E, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method with respect to $E^{-1}A$ or $E^{-1}(A-UW)$.
- int mess_eigen_arnoldig_inv (mess_direct A, mess_matrix U, mess_matrix W, mess_matrix E, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method on $A^{-1}E$ or $(A-UW)^{-1}E$.

- int mess_eigen_arnoldi_inv (mess_direct A, mess_matrix B, mess_matrix K, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method on A^{-1} or $(A-UW)^{-1}$.
- int mess_eigen_arnoldi_template (mess_mvpcall A, mess_int_t k, mess_vector sv, mess_matrix H, mess_matrix V)

  Compute matrices H and V with Arnoldis method with respect to A.
- int mess_eigen_arnoldi_template_nrm (mess_mvpcall A, mess_int_t k, mess_vector sv, double *abseig)

  Compute absolut value of the largest eigenvalue using Arnoldi process.
- int mess_eigen_lanczos_template_nrm (mess_mvpcall A, mess_int_t k, mess_vector sv, mess_vector diag, mess_vector subdiag, mess_matrix V, double *abseig)

  Lanczos method for 2-norm computations (real symmetric case).
- int mess_eigen_eigs (mess_matrix A, mess_int_t kp, mess_int_t km, mess_vector r, mess_vector ev)

  Compute some eigenvalues with the Arnoldi process.
- int mess_eigen_eiggs (mess_matrix A, mess_matrix E, mess_int_t kp, mess_int_t km, mess_vector r, mess_vector ev)

  Compute some generalized eigenvalues with the Arnoldi process.
- int mess_eigen_arpack (mess_matrix A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)

  An interface to the Arnoldi process from `ARPACK`.
- int mess_eigen_arpack_template (mess_mvpcall A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)

  An interface to the Arnoldi process from `ARPACK` *(template version).*
- int mess_eigen_arpack_lanczos (mess_matrix A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)

  An interface to the Lanczos process from `ARPACK`.
- int mess_eigen_arpack_lanczos_template (mess_mvpcall A, mess_int_t nev, mess_eigen_arpack_options_t opt, mess_vector ev, mess_matrix V)

  An interface to the Lanczos process from `ARPACK` *(template version).*
- int mess_eigen_eig (mess_matrix A, mess_vector ew, mess_matrix ev)

  Compute all eigenvalues and eigenvectors.
- int mess_eigen_eigg (mess_matrix A, mess_matrix B, mess_vector ew, mess_vector alpha, mess_vector beta, mess_matrix ev)

  Compute all eigenvalues and right eigenvectors of the generalized eigenvalue problem.
- int mess_eigen_schur_complex (mess_matrix A, mess_matrix T, mess_matrix U, mess_vector EV)

  Wrapper around both complex Schur decomposition routines.
- int mess_eigen_schur_complex_zgees (mess_matrix A, mess_matrix T, mess_matrix U, mess_vector EV)

  Compute the complex Schur decomposition of a matrix.
- int mess_eigen_schur_complex_post (mess_matrix A, mess_matrix T, mess_matrix U, mess_vector EV)

  Compute the complex Schur decomposition of a matrix using post processing.
- int mess_eigen_schur (mess_matrix A, mess_matrix T, mess_matrix U, mess_vector EV)

  Compute the real Schur decomposition of a matrix.
- int mess_eigen_gschur (mess_matrix A, mess_matrix B, mess_matrix S, mess_matrix T, mess_matrix U, mess_matrix V, mess_vector EVA, mess_vector EVB, mess_vector EV)

  Compute the real generalized Schur decomposition of a matrix pair (A,B).
- int mess_eigen_gschur_complex (mess_matrix A, mess_matrix B, mess_matrix S, mess_matrix T, mess_matrix U, mess_matrix V, mess_vector EVA, mess_vector EVB, mess_vector EV)

  Compute the generalized Schur decomposition of a matrix pair (A,B).
- int mess_eigen_svd (mess_matrix A, mess_vector S, mess_matrix U, mess_matrix V)

  Compute the singular value decompositon (SVD) of a matrix.
- int mess_eigen_svd_econ (mess_matrix A, mess_vector S, mess_matrix U, mess_matrix V)

  Compute the singular value decompositon (SVD) of a matrix.
- int mess_eigen_schur_to_evd (mess_matrix T, mess_matrix Q, mess_matrix V)

Compute eigenvectors from a given Schur decomposition-.

- int mess_eigen_hessenberg_schur (mess_matrix hess, mess_vector ew, mess_matrix Q, mess_matrix T)

  Compute Schur decomposition of an upoer Hessenberg matrix.

- int mess_eigen_hessenberg_abs_largest (mess_matrix hess, mess_vector evector, mess_double_cpx_t *abs_largest_ev)

  Compute the largest eigenvalue in magnitude and corresponding eigenvector of a upper Hessenberg matrix,.

- int mess_eigen_sign (mess_matrix A, mess_matrix Z, mess_sign_scale_t scale)
- int mess_eigen_gsign (mess_matrix A, mess_matrix B, mess_matrix Z, mess_sign_scale_t scale)

  Compute the sign of a matrix pencil.

10.212.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.213 enum_conversions.c File Reference

Macros

- #define MEXMESS_GENERATOR(CMESS_ENUM)

Functions

- void mex_test_mess_direct_cholpackage_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_direct_lupackage_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_equation_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_memusage_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_multidirect_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_operation_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_parameter_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_residual_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_norm_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])

10.213.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.213.2 Macro Definition Documentation

10.213.2.1 #define MEXMESS_GENERATOR(*CMESS_ENUM*)

Value:

```
void mex_test_##CMESS_ENUM( int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]){
        \


        \
         init_mexmess();
                                \

        \
          /* check input/output arguments */
        \
         if (nrhs != 1 ) {
        \
             csc_error_message("The function needs one input argument but %d are given.\n", (int) nrhs);
        \
         }
        \

        \
          /* check if class is correct */
        \
         if (!mxIsClass(prhs[0], #CMESS_ENUM )){
        \
             csc_error_message("Expected "#CMESS_ENUM" class.\n");
        \
             return ;
        \
         }
        \

        \
         /* call conversion function from MEX-M.E.S.S. to C-M.E.S.S. */
        \
         CMESS_ENUM c_enum = CMESS_ENUM##_from_mexmess(prhs[0]);
        \

        \
         /* call conversion function from C-M.E.S.S. to MEX-M.E.S.S. */
        \
         plhs[0] = CMESS_ENUM##_to_mexmess(c_enum);
        \

        \
         return;
        \
    }
```

10.214 eps.c File Reference

Functions

- double F77_GLOBAL (dlamch, DLAMCH)

  Calcute the machine epsilon.

10.214.1 Function Documentation

10.214.1.1 double F77_GLOBAL (dlamch , DLAMCH)

Returns

machine epsilon

The mess_eps function calculates the machine precision.

10.215 eps.c File Reference

Functions

- PyObject ∗ pymess_eps (PyObject ∗self, PyObject ∗args)

  Return machine epsilon. The pymess_eps is callable as `eps` *and calls internally the mess_eps function and returns the machine epsilon.*

10.215.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Nitin Prasad (nitin.prasad93@gmail.com, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.216 equal.c File Reference

Data Structures

- struct mat_diff

  Helper structure for mat_diff_mvp.

Functions

- int mess_matrix_equal (mess_matrix mat1, mess_matrix mat2)

  Check if two matrices are equal on data structure level.
- int mess_matrix_equal_verbose (mess_matrix mat1, mess_matrix mat2)

  Check if two matrices are equal on data structure level (with verbose output).
- int mess_matrix_equalpattern (mess_matrix A, mess_matrix B)

  Check if two matrices have the same pattern.
- int mess_matrix_joinpatterns (mess_matrix A, mess_matrix B)

  Join the pattern of two sparse matrices.
- static int mat_diff_mvp (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)

  Implement the $y = (A - B)x$ *matrix-vector product for the Arnoldi process.*
- int mess_matrix_diffnorm (mess_matrix A, mess_matrix B, double ∗nrm)

  Compute the 2*-norm of the difference of two matrices.*
- int mess_matrix_diffnormf (mess_matrix A, mess_matrix B, double ∗nrm)

  Compute the Frobenius norm of the difference of two matrices.

10.216.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.216.2 Function Documentation

10.216.2.1 static int mat_diff_mvp (void ∗ *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Parameters

in	*data*	input pointer to the matrix data
in	*op*	input (not used)
in	*x*	input right hand side vector x
out	*y*	output vector y

Returns

zero on success or a non-zero error value otherwise

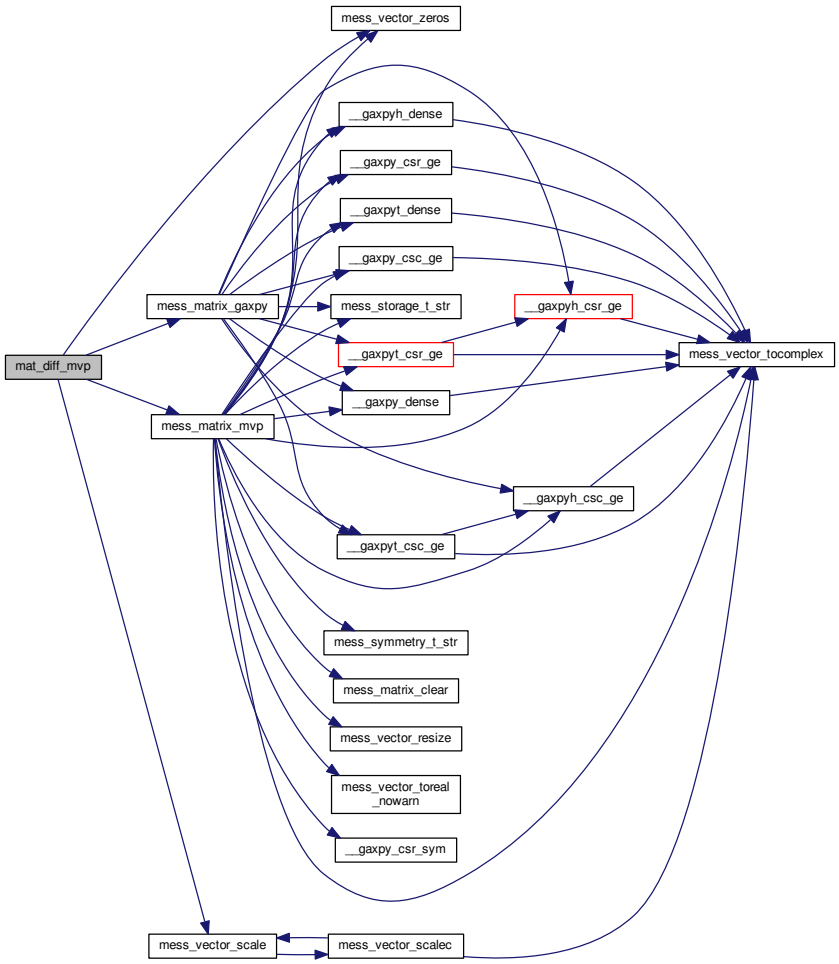
The mat_diff_mvp function computes

$$y = (A - B)x.$$

Attention

Internal use only.

Here is the call graph for this function:



10.217 equation.c File Reference

Functions

- int mess_equation_init (mess_equation *eqn)

  Initialize an empty mess_equation object.
- int mess_equation_clear (mess_equation *eqn)

  Clean a mess_equation structure.
- int mess_equation_clean (mess_equation eqn)

  Call cleanup functions for all internally generated data.
- int mess_equation_set_rhs (mess_equation eqn, mess_options opt, mess_matrix rhs)

  Set a new right hand side in a mess_equation object.
- int mess_equation_set_rhs2 (mess_equation eqn, mess_matrix rhs)

  Set a new right hand side in a mess_equation object without additional checks.
- mess_int_t mess_equation_dim (mess_equation eqn)

  Return the dimension of the equation of object.
- mess_int_t mess_equation_has_A (mess_equation eqn)

  Checks if $x = \mathcal{A}y$ is implemented.
- mess_int_t mess_equation_has_As (mess_equation eqn)

  Checks if $\mathcal{A}x = y$ is implemented.
- mess_int_t mess_equation_has_E (mess_equation eqn)

  Checks if $x = \mathcal{E}y$ is implemented.
- mess_int_t mess_equation_has_Es (mess_equation eqn)

  Checks if $\mathcal{E}x = y$ is implemented.
- mess_int_t mess_equation_has_ApE (mess_equation eqn)

  Checks if $x = (\mathcal{A} + \mathcal{E})y$ is implemented.
- mess_int_t mess_equation_has_ApEs (mess_equation eqn)

  Checks if $(\mathcal{A} + p\mathcal{E})x = y$ is implemented.

10.217.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.218 equation_apply.c File Reference

Functions

- int mess_equation_A_pre (mess_equation eqn)

  Initialization function of the linear mapping $x = \mathcal{A}y$.
- int mess_equation_A_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $x = \mathcal{A}y$.
- int mess_equation_A_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute $x = \mathcal{A}y$.
- int mess_equation_A_apply_vector (mess_equation eqn, mess_operation_t op, mess_vector in, mess_vector out)

  Compute $x = \mathcal{A}y$ vector valued.
- int mess_equation_E_pre (mess_equation eqn)

  Initialization function of the linear mapping $x = \mathcal{E}y$.

- int mess_equation_E_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $x = \mathcal{E}y$.
- int mess_equation_E_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute $x = \mathcal{E}y$.
- int mess_equation_E_apply_vector (mess_equation eqn, mess_operation_t op, mess_vector in, mess_vector out)

  Compute $x = \mathcal{E}y$ vector valued.
- int mess_equation_As_pre (mess_equation eqn)

  Initialization function of the linear mapping $\mathcal{A}x = y$.
- int mess_equation_As_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $\mathcal{A}x = y$.
- int mess_equation_As_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute $\mathcal{A}x = y$.
- int mess_equation_As_apply_vector (mess_equation eqn, mess_operation_t op, mess_vector in, mess_vector out)

  Compute $\mathcal{A}x = y$ vector valued.
- int mess_equation_Es_pre (mess_equation eqn)

  Initialization function of the linear mapping $\mathcal{E}x = y$.
- int mess_equation_Es_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $\mathcal{E}x = y$.
- int mess_equation_Es_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute $\mathcal{E}x = y$.
- int mess_equation_Es_apply_vector (mess_equation eqn, mess_operation_t op, mess_vector in, mess_vector out)

  Compute $\mathcal{E}\S = y$ vector valued.
- int mess_equation_ApE_pre (mess_equation eqn, mess_vector parameters)

  Initialization function of the linear mapping $x = (\mathcal{A} + p\mathcal{E})y$.
- int mess_equation_ApE_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $x = (\mathcal{A} + \mathcal{E})y$.
- int mess_equation_ApE_apply (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t p_idx, mess_matrix in, mess_matrix out)

  Compute $x = (\mathcal{A} + p\mathcal{E})y$.
- int mess_equation_ApE_apply_vector (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t p_idx, mess_vector in, mess_vector out)

  Compute $x = (\mathcal{A} + p\mathcal{E})y$ with vectors..
- int mess_equation_ApEs_pre (mess_equation eqn, mess_vector parameters)

  Initialization function of the linear mapping $(\mathcal{A} + p\mathcal{E})x = y$.
- int mess_equation_ApEs_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $(\mathcal{A} + \mathcal{E})x = y$.
- int mess_equation_ApEs_apply (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t p_idx, mess_matrix in, mess_matrix out)

  Compute $(\mathcal{A} + p\mathcal{E})x = y$.
- int mess_equation_ApEs_apply_vector (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t p_idx, mess_vector in, mess_vector out)

  Compute $(\mathcal{A} + p\mathcal{E})x = y$ with vectors..

10.218.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.219 equation_glyap.c File Reference

Data Structures

- struct __lyap

  Contains matrices, solvers and additional data for a generalized Lyapunov Equation.

Functions

- static int AX_apply (mess_equation e, mess_operation_t op, mess_matrix x, mess_matrix y)

  Compute matrix-matrix product with system matrix A.
- static int EX_apply (mess_equation e, mess_operation_t op, mess_matrix x, mess_matrix y)

  Compute matrix-matrix product with system matrix E.
- static int AINV_generate (mess_equation e)
- static int AINV_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int AINV_clear (mess_equation e)
- static int EINV_generate (mess_equation e)
- static int EINV_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int EINV_clear (mess_equation e)
- static int ApEINV_generate (mess_equation e, mess_vector parameters)
- static int ApEINV_apply (mess_equation e, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)
- static int ApEINV_clear (mess_equation e)
- static int ApEX_generate (mess_equation e, mess_vector parameters)
- static int ApEX_apply (mess_equation e, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)
- static int ApEX_clear (mess_equation e)
- static int clear (mess_equation e)
- static int parameter (mess_equation e, mess_options opt, mess_status stat)
- int mess_equation_glyap (mess_equation e, mess_options opt, mess_matrix A, mess_matrix E, mess_matrix B)

  Setup a mess_Equation object to solve a generalized Lyapunov Equation.
- int mess_equation_griccati (mess_equation e, mess_options opt, mess_matrix A, mess_matrix E, mess_matrix B, mess_matrix C)

  Setup a mess_equation object to solve a generalized Riccati Equation.

10.219.1 Detailed Description

Author

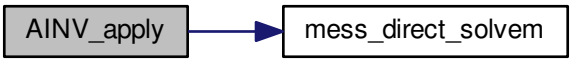
Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Generate the mess_equation object for the generalized Lyapunov and Riccati equatuion.

10.219.2 Function Documentation

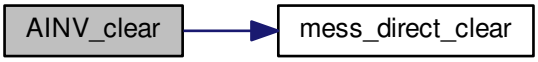
10.219.2.1 static int AINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Here is the call graph for this function:



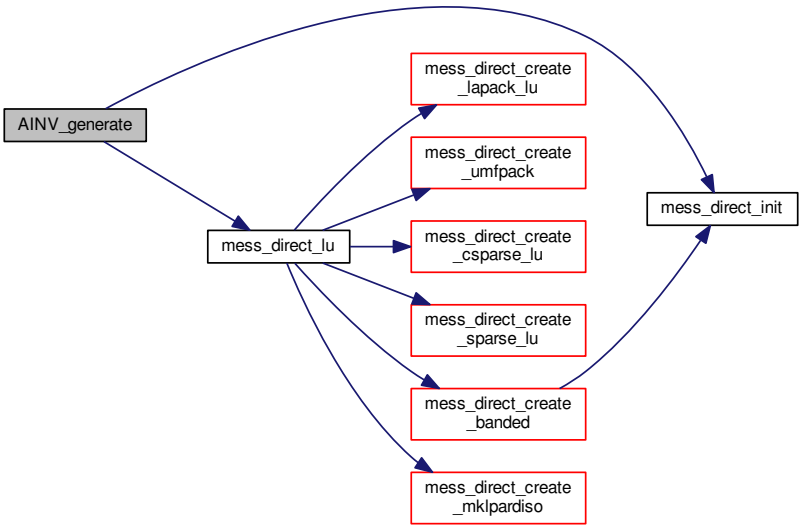
10.219.2.2 static int AINV_clear (mess_equation *e*) [static]

Here is the call graph for this function:



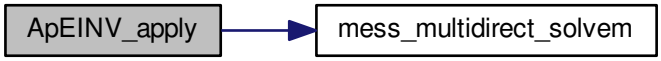
10.219.2.3 static int AINV_generate (mess_equation *e*) [static]

Here is the call graph for this function:



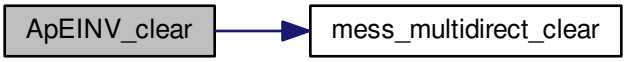
10.219.2.4 static int ApEINV_apply (mess_equation *e*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *idx_p*, mess_matrix *in*, mess_matrix *out*) [static]

Here is the call graph for this function:



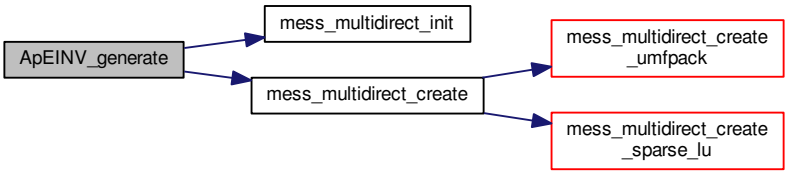
10.219.2.5 static int ApEINV_clear (mess_equation *e*) [static]

Here is the call graph for this function:



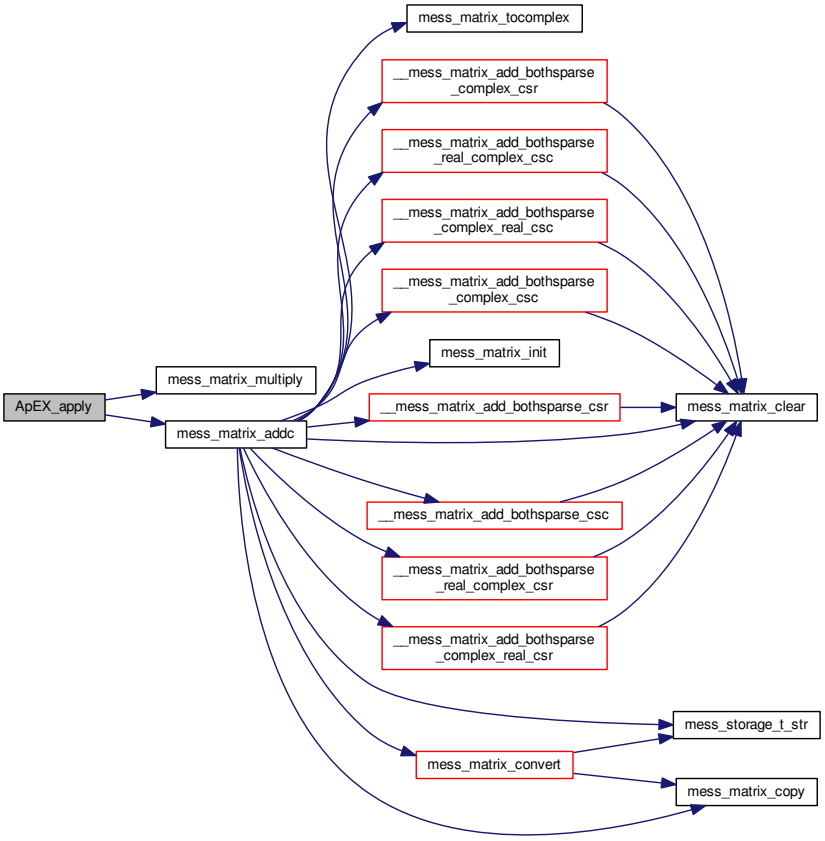
10.219.2.6 static int ApEINV_generate (mess_equation *e*, mess_vector *parameters*) [static]

Here is the call graph for this function:



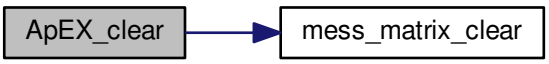
10.219.2.7 static int ApEX_apply (mess_equation *e*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *idx_p*, mess_matrix *in*, mess_matrix *out*) [static]

Here is the call graph for this function:



10.219.2.8 static int ApEX_clear (mess_equation *e*) [static]

Here is the call graph for this function:



10.219.2.9 static int ApEX_generate (mess_equation *e*, mess_vector *parameters*) [static]

Here is the call graph for this function:

10.219.2.10 static int AX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *x*, mess_matrix *y*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to A
in	*x*	input matrix x
out	*y*	output matrix y

Returns

zero on succes or a non-zero error value otherwise

The AX_apply function computes the matrix-matrix product, i.e.

$$y = op(A)x.$$

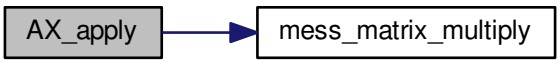
See also

mess_matrix_multiply

Attention

Internal use only.

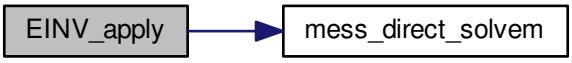
Here is the call graph for this function:



10.219.2.11 static int clear (mess_equation *e*) [static]

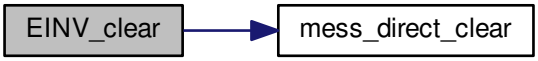
10.219.2.12 static int EINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Here is the call graph for this function:



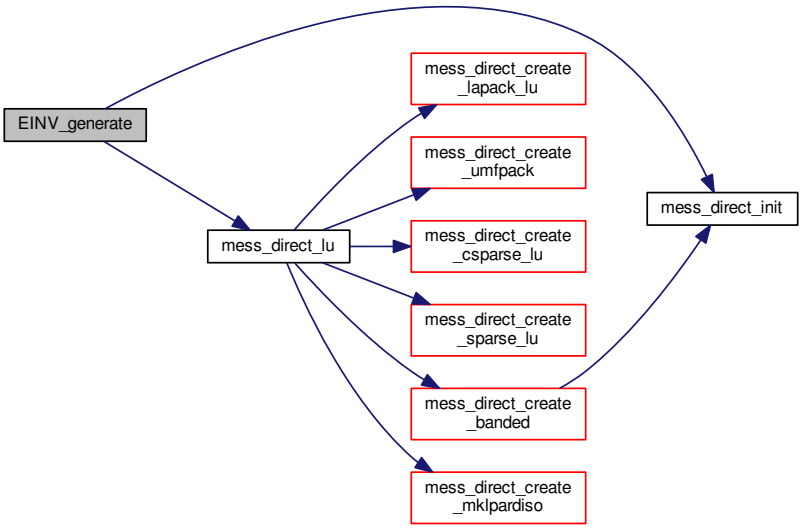
10.219.2.13 static int EINV_clear (mess_equation *e*) [static]

Here is the call graph for this function:



10.219.2.14 static int EINV_generate (mess_equation *e*) [static]

Here is the call graph for this function:



10.219.2.15 static int EX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *x*, mess_matrix *y*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to E
in	*x*	input matrix x
out	*y*	output matrix y

Returns

zero on succes or a non-zero error value otherwise

The EX_apply function computes the matrix-matrix product, i.e.

$$y = op(E)x.$$

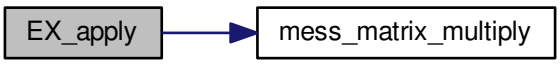
See also

mess_matrix_multiply

Attention

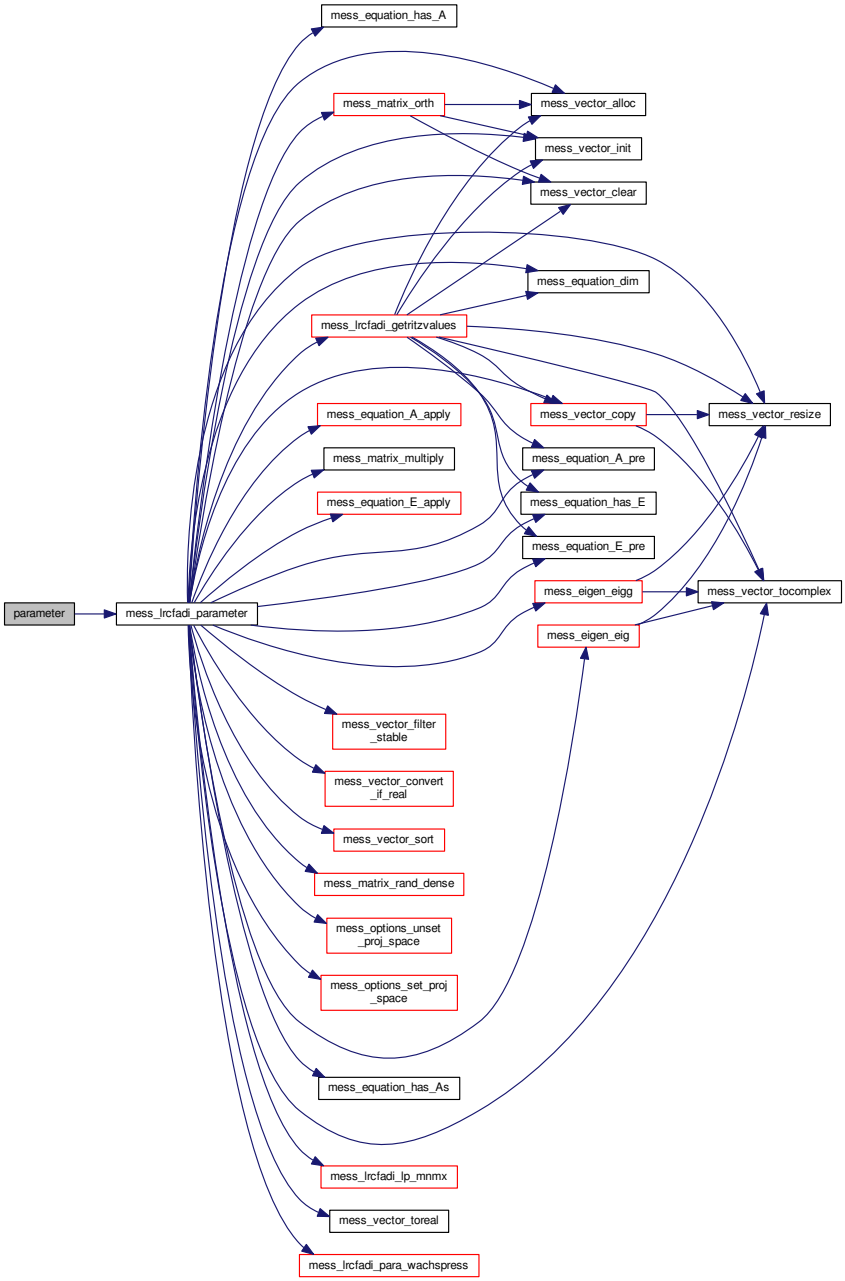
Internal use only.

Here is the call graph for this function:



10.219.2.16 static int parameter (mess_equation *e*, mess_options *opt*, mess_status *stat*) [static]

Here is the call graph for this function:



10.220 equation_glyap_dae1.c File Reference

Data Structures

- struct __glyap_dae1

  Contains matrices, solvers and additional data for a DAE 1 function handles.

Functions

- static int AX_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute matrix-matrix multiplication with $A_{11} - A_{12}A_{22}^{-1}A_{22}$.
- static int AINV_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Solve linear system with block matrix $op(A_{11} - A_{12}A_{22}^{-1}A_{22})$.
- static int EX_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute matrix-matrix multiplication with E_{11}.
- static int EINV_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Solve linear system with block matrix $op(E_{11})$.
- static int ApEINV2_generate (mess_equation e, mess_vector parameters)

  Generate a mess_multidirect solve for the system matrix $\begin{bmatrix} A_{11} + pE_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$.
- static int ApEINV2_apply (mess_equation e, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)

  Solve shifted linear block system with matrix $\begin{bmatrix} A_{11} + pE_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$.
- static int ApEINV2_clear (mess_equation e)

  Clear a mess_multidirect solve for the system matrix $\begin{bmatrix} A_{11} + pE_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$.
- static int parameter (mess_equation e, mess_options opt, mess_status stat)

  Call the mess_lrcfadi_parameter function.
- static int clear (mess_equation e)

  Clear additional allocated field of __glyap_dae1 structure.
- int mess_equation_glyap_dae1 (mess_equation e, mess_options opt, mess_matrix E11, mess_matrix A11, mess_matrix A12, mess_matrix A21, mess_matrix A22, mess_matrix rhs)

  Generate an Equation object to solve a generalized Lyapunov Equation from Index 1 DAE.
- int mess_equation_griccati_dae1 (mess_equation e, mess_options opt, mess_matrix E11, mess_matrix A11, mess_matrix A12, mess_matrix A21, mess_matrix A22, mess_matrix B, mess_matrix C)

  Generate an Equation object to solve a generalized Lyapunov Equation from Index 1 DAE.

10.220.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.220.2 Function Documentation

10.220.2.1 static int AINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

The AINV_apply function solve a linear system

$$op(A_{11} - A_{12}A_{22}^{-1}A_{22})out = in$$

by using the equivalent block formulation

$$op\left(\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\right)\begin{bmatrix} out \\ * \end{bmatrix} = \begin{bmatrix} in \\ 0 \end{bmatrix}.$$

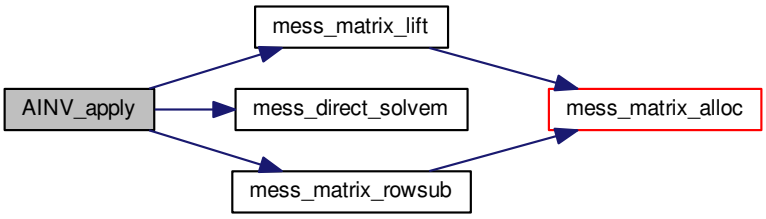
See also

mess_matrix_rowsub
mess_direct_solvem
mess_matrix_lift

Attention

Internal use only.

Here is the call graph for this function:



10.220.2.2 static int ApEINV2_apply (mess_equation *e*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *idx_p*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in,out	*e*	input mess_equation structure
in	*op*	input operation type
in	*p*	input shift parameter
in	*idx_p*	input index of parameter for shift
in	*in*	input matrix
out	*out*	output matrix

Returns

zero on succes or a non-zero error value otherwise

The ApEINV2_apply function solves the system

$$op((A_{11} - A_{12}A_{22}^{-1}A_{21}) + pE_{11})out = in$$

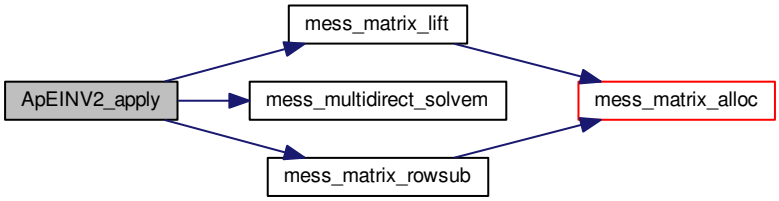
via the equivalent block formulation

$$op\left(\begin{bmatrix} A_{11} + pE_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\right)\begin{bmatrix} out \\ * \end{bmatrix} = \begin{bmatrix} in \\ 0 \end{bmatrix}.$$

Attention

Internal use only.

Here is the call graph for this function:



10.220.2.3 static int ApEINV2_clear (mess_equation *e*) [static]

Parameters

in	*e*	input mess_equation structure

Returns

zero on succes or a non-zero error value otherwise

The ApEINV2_clear builds a mess_multidirect solver for the matrices

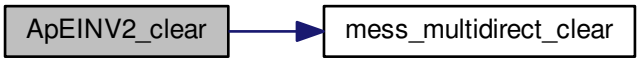
$$\begin{bmatrix} A_{11} + pE_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}.$$

by using mess_multidirect_clear the field __glyap_dae1::Amsolver is cleared.

Attention

Internal use only.

Here is the call graph for this function:



10.220.2.4 static int ApEINV2_generate (mess_equation *e*, mess_vector *parameters*) [static]

Parameters

in	*e*	input mess_equation structure
in	*parameters*	input mess_vector with shift parameters

Returns

zero on succes or a non-zero error value otherwise

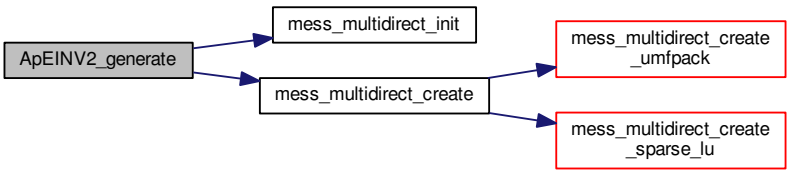
The ApEINV2_generate builds a mess_multidirect solver for the matrices

$$\begin{bmatrix} A_{11} + pE_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}.$$

Attention

Internal use only.

Here is the call graph for this function:



10.220.2.5 static int AX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

The AX_apply function computes the matrix-matrix product, i.e.

$$in = op(A_{11} - A_{12}A_{22}^{-1}A_{22})out.$$

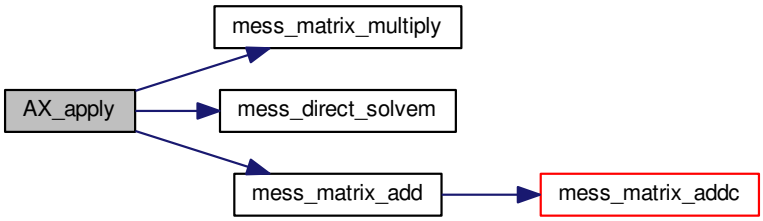
See also

mess_matrix_multiply
mess_matrix_add
mess_direct_solvem

Attention

Internal use only.

Here is the call graph for this function:



10.220.2.6 static int clear (mess_equation *e*) [static]

Parameters

in,out	*e*	input mess_equation structure

Returns

zero on succes or a non-zero error value otherwise

The clear function, clears the following additional allocated data

field	mathematical description
__glyap_dae1::fullA	$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$

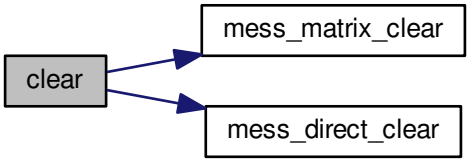
field	mathematical description
__glyap_dae1::fullE	$\begin{bmatrix} E_{11} & 0 \\ 0 & 0 \end{bmatrix}$
__glyap_dae1::A22sol	A_{22}^{-1}
__glyap_dae1::fullAsol	$\left(\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}\right)^{-1}$
__glyap_dae1::E11sol	E_{11}^{-1}

.

Attention

Internal use only.

Here is the call graph for this function:



10.220.2.7 static int EINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

The EINV_apply function solve a linear system

$$op(E_{11} - A_{12}A_{22}^{-1}A_{22})out = in.$$

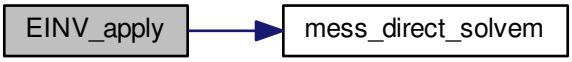
See also

mess_direct_solvem

Attention

Internal use only.

Here is the call graph for this function:



10.220.2.8 static int EX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

The EX_apply function computes the matrix-matrix product, i.e.

$$out = op(E_{11})in.$$

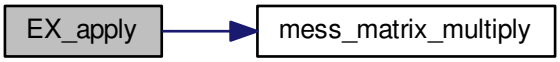
See also

mess_matrix_multiply

Attention

Internal use only.

Here is the call graph for this function:



10.220.2.9 static int parameter (mess_equation *e*, mess_options *opt*, mess_status *stat*) [static]

Parameters

in	*e*	input mess_equation structure
in	*opt*	input options
in	*stat*	input status

Returns

zero on succes or a non-zero error value otherwise

The parameter function calls directly mess_lrcfadi_parameter. If mess_equation::parent of `in` is not `NULL` the equation was stabilized by a feedback and mess_lrcfadi_parameter is called with the stabilized equation object mess_equation::parent otherwise `e` is simply used.

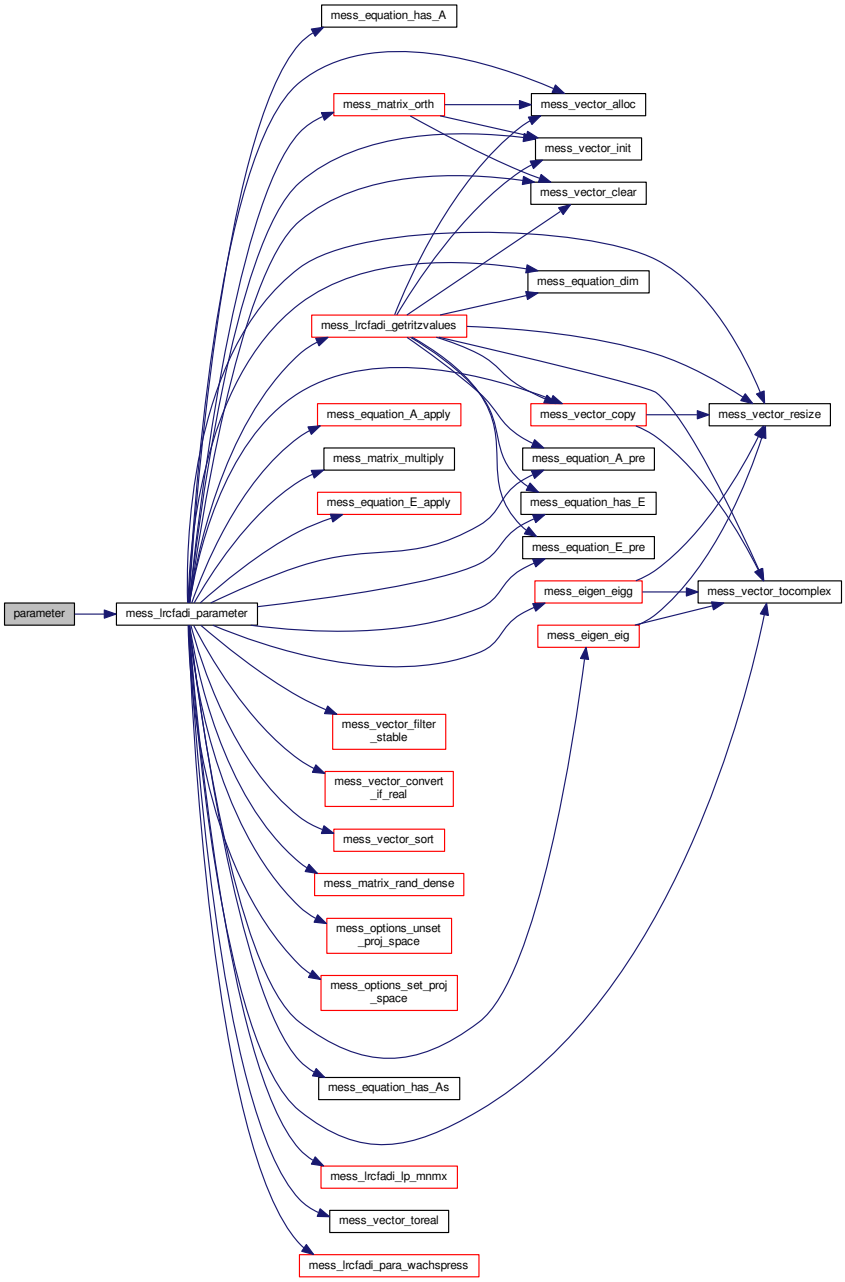
See also

mess_lrcfadi_parameter

Attention

Internal use only.

Here is the call graph for this function:



10.221 equation_glyap_dae2.c File Reference

Data Structures

- struct __glyap_dae2

  Contains matrices, solvers and additional data for a DAE 2 function handles.

Functions

- static int mess_matrix_applyPI_dae2 (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Apply the projector $\Pi := I - G(G^T M^{-1} G)^{-1} G^T M^{-1}$ *to a mess_matrix.*
- static int AX_apply_T (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Apply the projector $op(A\Pi^T)$ *to a mess_matrix.*
- static int AX_apply_N (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Apply the projector $op(\Pi A)$ *to a mess_matrix.*
- static int AX_apply_shifts (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Apply the projector $op(\Pi^T A \Pi)$ *or* $\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}$ *to a mess_matrix.*
- static int EX_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Multiplication with M *and a mess_matrix.*
- static int EX_apply_shifts (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Multiplication with M *or* $\begin{bmatrix} M & \delta G \\ \delta G^T & 0 \end{bmatrix}$ *and a mess_matrix.*
- static int AINV_apply_shifts (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Apply $\left(\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}\right)^{-1}$ *to a mess_matrix.*
- static int EINV_apply_shifts (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Apply $\left(\begin{bmatrix} M & \delta G \\ \delta G^T & 0 \end{bmatrix}\right)^{-1}$ *to a mess_matrix.*
- static int ApEINV_generate_memory1 (mess_equation e, mess_vector parameters)

  Generate a shifted solver for the system matrix $\begin{bmatrix} A + pM & G \\ G^T & 0 \end{bmatrix}$ *(high memory variant).*
- static int ApEINV_apply_memory1 (mess_equation e, mess_operation_t op, mess_double_cpx_t p, mess↩_int_t idx_p, mess_matrix in, mess_matrix out)

  Solve shifted linear block system with matrix $\begin{bmatrix} A + pM & G \\ G^T & 0 \end{bmatrix}$ *(high memory variant).*
- static int ApEINV_clear_memory1 (mess_equation e)

  Clear a array of direct solvers for the system matrix $\begin{bmatrix} A + pM & G \\ G^T & 0 \end{bmatrix}$ *(high memory variant).*
- static int ApEINV_generate_memory0 (mess_equation e, mess_vector parameters)

  Dummy for generating direct solvers (low memory variant).
- static int ApEINV_apply_memory0 (mess_equation e, mess_operation_t op, mess_double_cpx_t p, mess↩_int_t idx_p, mess_matrix in, mess_matrix out)

  Solve shifted linear block system with matrix $\begin{bmatrix} A + pM & G \\ G^T & 0 \end{bmatrix}$ *(low memory variant).*
- static int ApEINV_clear_memory0 (mess_equation e)

  Dummy function (low memory variant).
- static int parameter (mess_equation e, mess_options opt, mess_status stat)

  Call the mess_lrcfadi_parameter and exchange function pointers.
- static int init_rhs (mess_equation e, mess_options opt)

  Apply Π *to the right hand side of an equation.*
- static int clear (mess_equation e)

  Clear additional allocated field of __glyap_dae2 structure.
- int mess_equation_glyap_dae2 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix A, mess_matrix G, mess_matrix rhs, double delta)

  Generate an Equation object to solve a generalized Lyapunov Equation from Hessenberg Index 2 DAE.
- int mess_equation_griccati_dae2 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix A, mess_matrix G, mess_matrix B, mess_matrix C, double delta)

  Generate an equation object to a Riccati Equation arising from a Hessenberg Index 2 DAE.

10.221.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.221.2 Function Documentation

10.221.2.1 static int AINV_apply_shifts (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

The AINV_apply_shifts function computes $out = op\left(\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}\right)^{-1} in$. The number of rows of in must be $n_v + n_p$. This function should only be used for MESS_LRCFADI_PARA_MINMAX or MESS_LRCFADI_PARA_↩MINMAX_REAL shift parameter computations.

Attention

Internal use only.

Here is the call graph for this function:

10.221.2.2 static int ApEINV_apply_memory0 (mess_equation *e*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *idx_p*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in,out	*e*	input mess_equation structure
in	*op*	input operation type
in	*p*	input shift parameter
in	*idx*↩ *_p*	input index of parameter for shift
in	*in*	input matrix
out	*out*	output matrix

Returns

zero on succes or a non-zero error value otherwise

The ApEINV_apply_memory0 function solves the system

$$op(\Pi A + pM)out = in$$

via the equivalent block reformulation

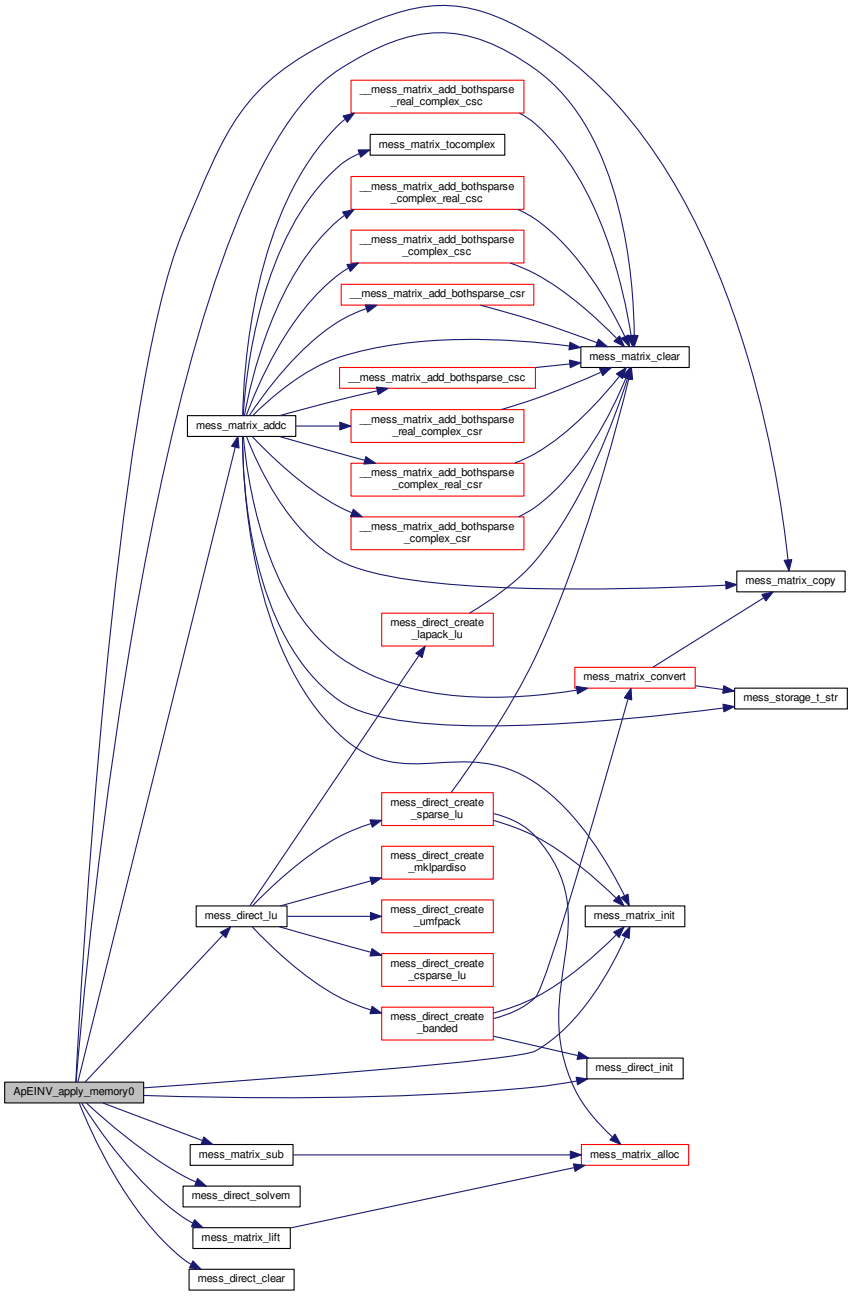
$$op\left(\begin{bmatrix} A + pM & G \\ G^T & 0 \end{bmatrix}\right)\begin{bmatrix} out \\ * \end{bmatrix} = \begin{bmatrix} in \\ 0 \end{bmatrix}.$$

The formulation is mathematical equivalent if $in \in image(\Pi)$. Needed mess_direct instances to solve the linear system are generated and cleared in that function.

Attention

Internal use only.

Here is the call graph for this function:



10.221.2.3 static int ApEINV_apply_memory1 (mess_equation *e*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *idx_p*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in,out	*e*	input mess_equation structure
in	*op*	input operation type
in	*p*	input shift parameter
in	*idx_p*	input index of parameter for shift

Parameters

in	*in*	input matrix
out	*out*	output matrix

Returns

zero on succes or a non-zero error value otherwise

The ApEINV_apply_memory1 function solves the system

$$op(\Pi A + pM)out = in$$

via the equivalent block reformulation

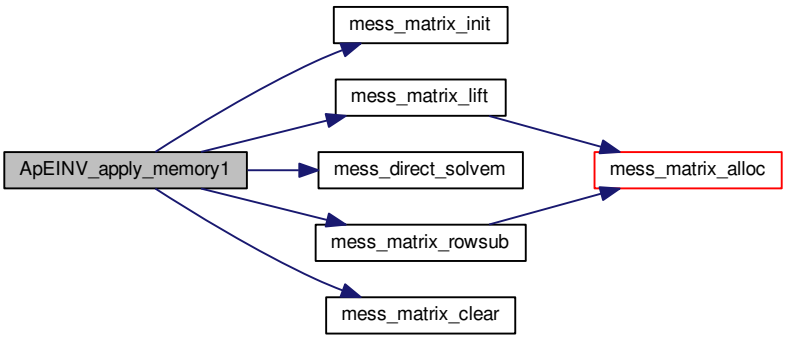
$$op\left(\begin{bmatrix} A + pM & G \\ G^T & 0 \end{bmatrix}\right)\begin{bmatrix} out \\ * \end{bmatrix} = \begin{bmatrix} in \\ 0 \end{bmatrix}.$$

The formulation is mathematical equivalent if $in \in image(\Pi)$.

Attention

Internal use only.

Here is the call graph for this function:



10.221.2.4 static int ApEINV_clear_memory0 (mess_equation *e*) [static]

Parameters

in	*e*	input mess_equation structure

Returns

zero on succes or a non-zero error value otherwise

The ApEINV_clear_memory0 is only a dummy function because all data are cleared directly in ApEINV_apply_memory0.

Attention

Internal use only.

10.221.2.5 static int ApEINV_clear_memory1 (mess_equation *e*) [static]

Parameters

in	*e*	input mess_equation structure

Returns

zero on succes or a non-zero error value otherwise

The ApEINV_clear_memory1 clear generated solvers for the matrices

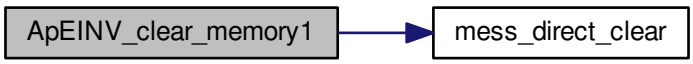
$$\begin{bmatrix} A + pM & G \\ G^T & 0 \end{bmatrix}.$$

by using mess_direct_clear the fields of __glyap_dae2::shiftsolvers are cleared.

Attention

Internal use only.

Here is the call graph for this function:



10.221.2.6 static int ApEINV_generate_memory0 (mess_equation *e*, mess_vector *parameters*) [static]

Parameters

in	*e*	input mess_equation structure
in	*parameters*	input shift parameters

Returns

zero on succes or a non-zero error value otherwise

The ApEINV_generate_memory0 is only a dummy function which sets the field. The decomposition for solving the shifted system is directly computed in ApEINV_apply_memory0.

Attention

Internal use only.

10.221.2.7 static int ApEINV_generate_memory1 (mess_equation *e*, mess_vector *parameters*) [static]

Parameters

in	*e*	input mess_equation structure
in	*parameters*	input mess_vector of shift parameters

Returns

zero on succes or a non-zero error value otherwise

The ApEINV_generate_memory1 builds a mess_multidirect solver for the matrices

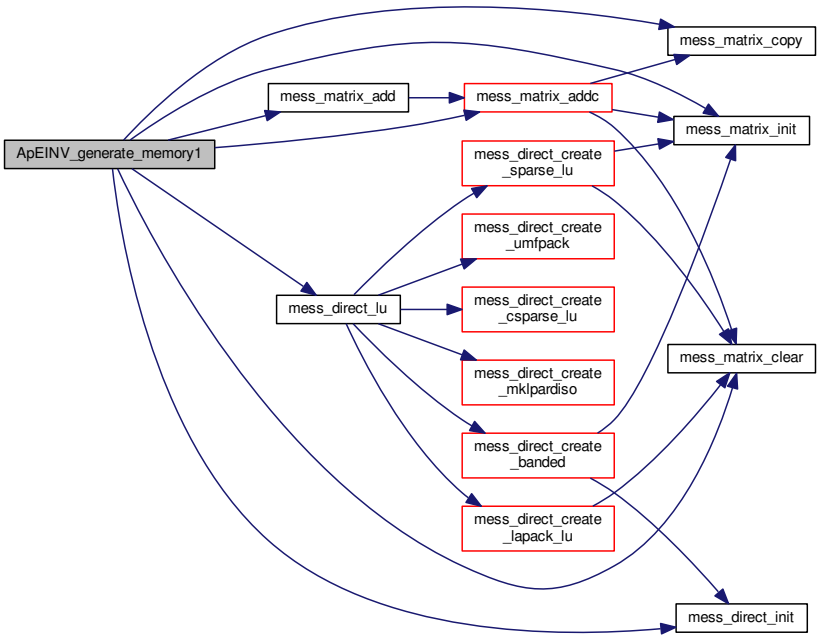
$$\begin{bmatrix} A + pM & G \\ G^T & 0 \end{bmatrix}.$$

It uses for each shift parameter a mess_direct instance.

Attention

Internal use only.

Here is the call graph for this function:



10.221.2.8 static int AX_apply_N (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

The AX_apply_N function computes $out = op(\Pi A)in$. This function is only used in residual computation, therefore it is assumed that in has n_v rows.

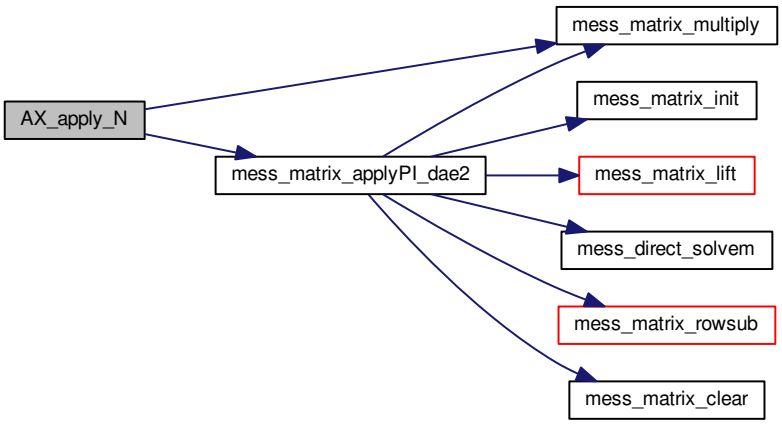
See also

mess_matrix_applyPI_dae2
AX_apply_N

Attention

Internal use only.

Here is the call graph for this function:



10.221.2.9 static int AX_apply_shifts (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

The AX_apply_shifts function computes $op(\Pi^T A \Pi)$ or $\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}$. If the number of rows of in is n_v then we perform

$$out = op(\Pi^T A \Pi) in$$

, because it is assumed that the context of usage is in a shift parameter computation in the case of MESS_LRCFADI_PARA_ADAPTIVE_V or MESS_LRCFADI_PARA_ADAPTIVE_Z. If the number of rows of in is $n_v + n_p$ we perform $out = \begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix} in$. In that case it is assumed that the function is used for MESS_LRCFADI_PARA_MINMAX or MESS_LRCFADI_PARA_MINMAX_REAL.

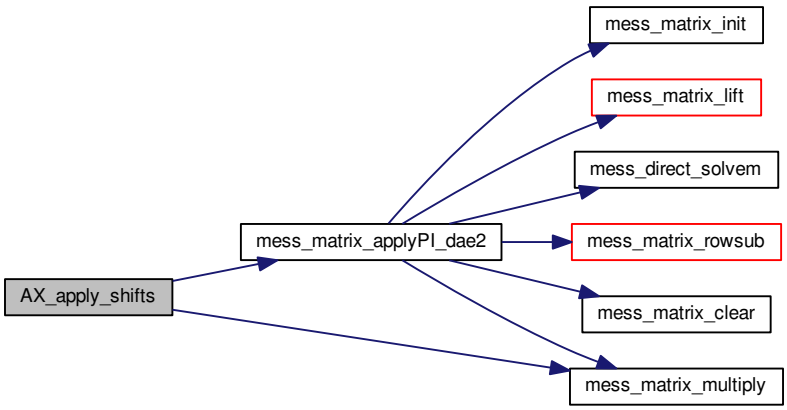
See also

mess_matrix_applyPI_dae2

Attention

Internal use only.

Here is the call graph for this function:



10.221.2.10 static int AX_apply_T (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

The AX_apply_T function computes $out = op(A\Pi^T)in$. This function is only used in residual computation, therefore it is assumed that in n_v rows.

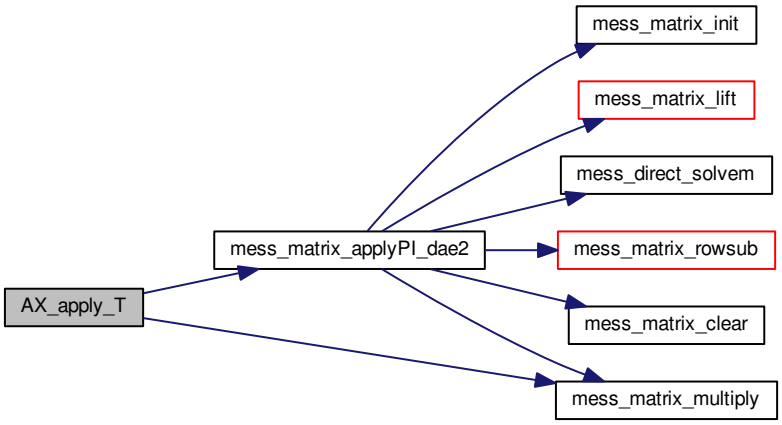
See also

mess_matrix_applyPI_dae2
AX_apply_N

Attention

Internal use only.

Here is the call graph for this function:



10.221.2.11 static int clear (mess_equation *e*) [static]

Parameters

in,out	*e*	input mess_equation structure

Returns

zero on succes or a non-zero error value otherwise

The clear function, clears the following additional allocated data

field	mathematical description
__glyap_dae2::fullA	$\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}$
__glyap_dae2::fullM	$\begin{bmatrix} M & \delta G \\ \delta G^T & 0 \end{bmatrix}$
__glyap_dae2::MM	$\begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}$
__glyap_dae2::fullAsolver	$\left(\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}\right)^{-1}$
__glyap_dae2::fullMsolver	$\begin{bmatrix} M & \delta G \\ \delta G^T & 0 \end{bmatrix}$
__glyap_dae2::applyPIsolver	$\left(\begin{bmatrix} M & \delta G \\ \delta G^T & 0 \end{bmatrix}\right)^{-1}$

.

Attention

Internal use only.

10.221.2.12 static int EINV_apply_shifts (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

The EINV_apply_shifts function computes $out = op\left(\begin{bmatrix} M & \delta G \\ \delta G^T & 0 \end{bmatrix}\right)^{-1} in$. The number of rows of in must be $n_v + n_p$. This function should only be used for MESS_LRCFADI_PARA_MINMAX or MESS_LRCFADI_PARA_↩MINMAX_REAL shift parameter computations.

Attention

Internal use only.

Here is the call graph for this function:

10.221.2.13 static int EX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

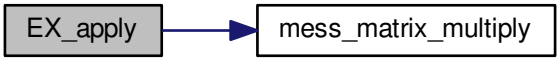
zero on succes or a non-zero error value otherwise

The EX_apply function computes $out = op(M)in$. It is assumed that in has n_v rows.

Attention

Internal use only.

Here is the call graph for this function:



10.221.2.14 static int EX_apply_shifts (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

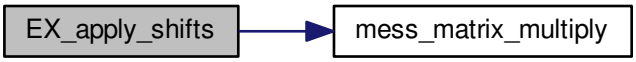
zero on succes or a non-zero error value otherwise

The EX_apply_shifts function computes $out = op(M)in$ or $out = op\left(\begin{bmatrix} M & \delta G \\ \delta G^T & 0 \end{bmatrix}\right) in$. If the number of rows of in is n_v we perform the first operation. If the number of rows of in is $n_v + n_p$ we perform the second operation. Therfore it is assumed that in has n_v or $n_v + n_p$ rows. This function is only used for shift parameter computations.

Attention

Internal use only.

Here is the call graph for this function:



10.221.2.15 static int init_rhs (mess_equation *e*, mess_options *opt*) [static]

Parameters

in,out	*e*	input mess_equation structure
in	*opt*	input mess_options structure

Returns

zero on succes or a non-zero error value otherwise

The init_rhs function applies Π to the mess_matrix mess_equation::RHS. If the equation is stabilized, i.e. mess_equation::parent is not NULL then Π is applied to the parent equation, otherwise to the original one.

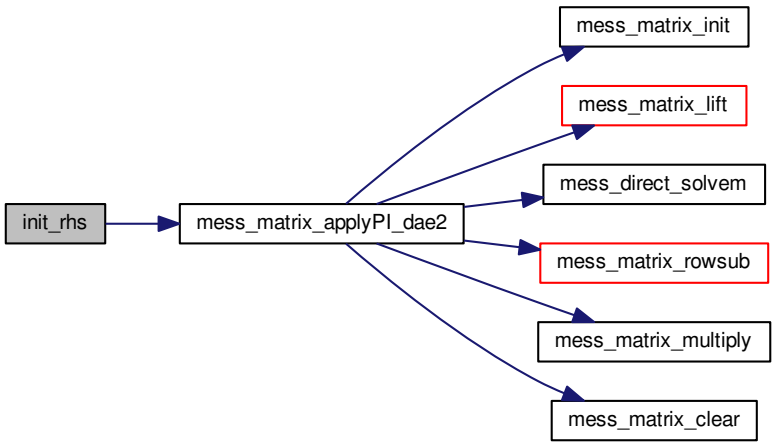
See also

mess_matrix_applyPI_dae2

Attention

Internal use only.

Here is the call graph for this function:



10.221.2.16 static int mess_matrix_applyPI_dae2 (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

The mess_matrix_applyPI_dae2 function applies the projector Π to a matrix, i.e. it computes

$$out = op(\Pi)in = op(I - G(G^T M^{-1} G)^{-1} G^T M^{-1})in$$

by solving

$$\begin{bmatrix} M & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} M^{-1} out \\ * \end{bmatrix} = \begin{bmatrix} in \\ 0 \end{bmatrix}$$

for

$$\Pi in = out$$

or $\begin{bmatrix} M & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} out \\ * \end{bmatrix} = \begin{bmatrix} Min \\ 0 \end{bmatrix}$ for$\Pi^T in = out.$

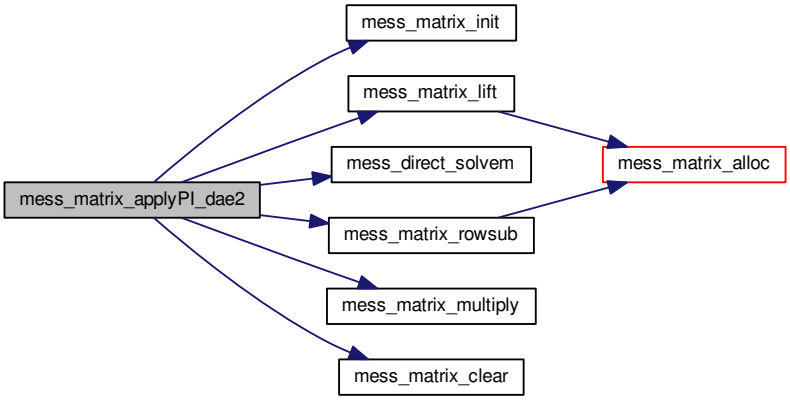
See also

mess_matrix_multiply
mess_matrix_add
mess_direct_solvem

Attention

Internal use only.

Here is the call graph for this function:



10.221.2.17 static int parameter (mess_equation *e*, mess_options *opt*, mess_status *stat*) [static]

Parameters

in	*e*	input mess_equation structure
in	*opt*	input options
in	*stat*	input status

Returns

zero on succes or a non-zero error value otherwise

The parameter function performs the shift parameter computation for the dae2 function handles. In the case of MESS_LRCFADI_PARA_ADAPTIVE_V or MESS_LRCFADI_PARA_ADAPTIVE_Z function pointers are exchanged and the dimension of the equation is changed. After that the mess_lrcfadi_parameter function is called with the stabilized or original equation.

In the case of MESS_LRCFADI_PARA_MINMAX or MESS_LRCFADI_PARA_MINMAX_REAL function pointers are exchanged the matrix B and K are lifted, that the $(1,1)$ block of the matrix if stabilized.

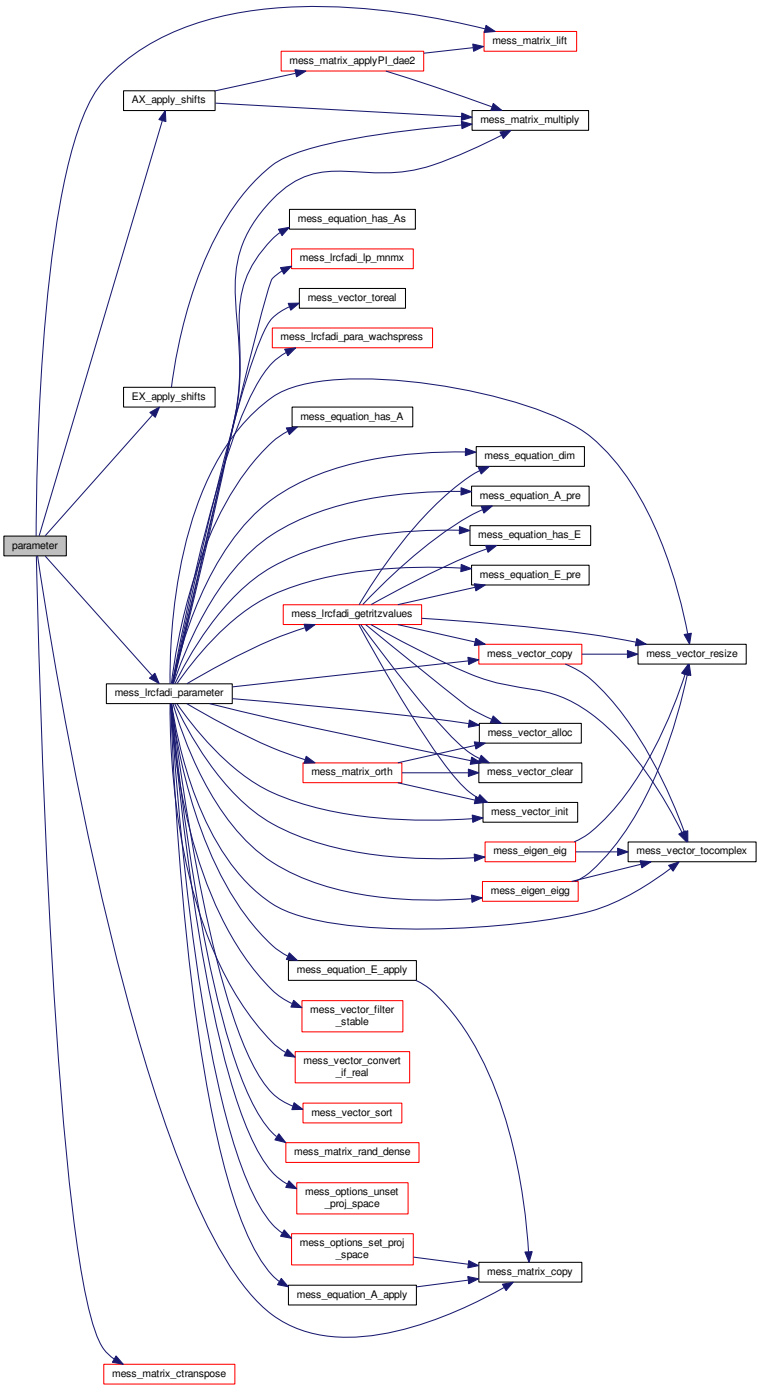
See also

AX_apply_shifts
EX_apply_shifts
mess_lrcfadi_parameter

Attention

Internal use only.

Here is the call graph for this function:



10.222 equation_glyap_so1.c File Reference

Data Structures

- struct __glyap_so1

  Contains matrices, solvers and additional data for Second Order System function handles.

Functions

- static int AX_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Multiply with operator $\begin{bmatrix} 0 & -K \\ -K & -D \end{bmatrix}$ *to a mess_matrix.*
- static int EX_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Multiply with operator $\begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix}$ *to a mess_matrix.*
- static int AINV_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Solve with operator $\begin{bmatrix} 0 & -K \\ -K & -D \end{bmatrix}$ *to a mess_matrix.*
- static int EINV_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Solve with $\begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix}$.
- static int ApEINV_generate (mess_equation e, mess_vector parameters)

  Generate a shifted solver for the system matrix $\begin{bmatrix} -pK & -K \\ -K & -D+pM \end{bmatrix}$.
- static int ApEINV_apply (mess_equation e, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)

  Solves shifted linear block system matrix $\begin{bmatrix} -pK & -K \\ -K & -D+pM \end{bmatrix}$.
- static int ApEINV_clear (mess_equation e)

  Clear a array of direct solvers.
- static int parameter (mess_equation e, mess_options opt, mess_status stat)

  Call the mess_lrcfadi_parameter and filter shift parameters.
- static int clear (mess_equation e)

  Clear additional allocated field of __glyap_so1 structure.
- int mess_equation_griccati_so1 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix D, mess_matrix K, mess_matrix B, mess_matrix C, double lowerbound, double upperbound)

  Generate an equation object to a Riccati Equation arising from a second order system.
- int mess_equation_glyap_so1 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix D, mess_matrix K, mess_matrix B, double lowerbound, double upperbound)

  Generate an Equation object to a Lyapunov Equation arising from a second order system.

10.222.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.222.2 Function Documentation

10.222.2.1 static int AINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

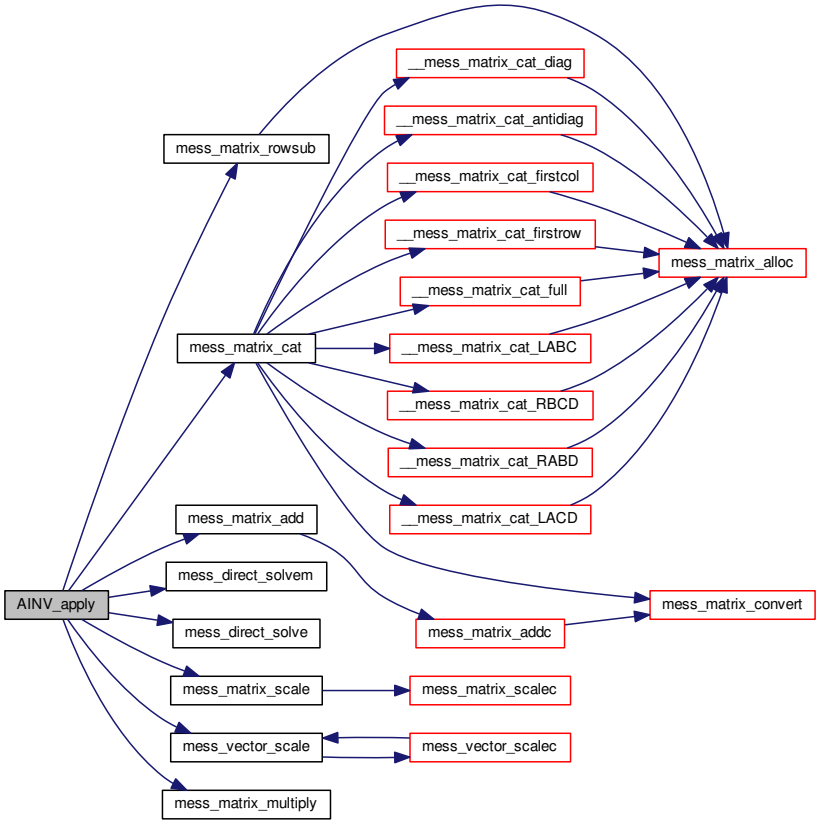
The AINV_apply function solves the system $\begin{bmatrix} 0 & -K \\ -K & -D \end{bmatrix} out = in$ via the equivalent formulation

$$\begin{aligned} out(n+1:end,:) &= -K^{-1} in(1:n,:) \\ out(1:n,:) &= -K^{-1}(in(n+1:end,:) + Dout(n+1:end,:)) \end{aligned}.$$

Attention

Internal use only.

Here is the call graph for this function:



10.222.2.2 static int ApEINV_apply (mess_equation *e*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *idx_p*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in,out	*e*	input mess_equation structure
in	*op*	input operation type
in	*p*	input shift parameter
in	*idx↩ _p*	input index of parameter for shift
in	*in*	input matrix
out	*out*	output matrix

Returns

zero on succes or a non-zero error value otherwise

The ApEINV_apply_memory1 function solves the system

$$op\left(\begin{bmatrix} -pK & -K \\ -K & -D+pM \end{bmatrix}\right) out = in$$

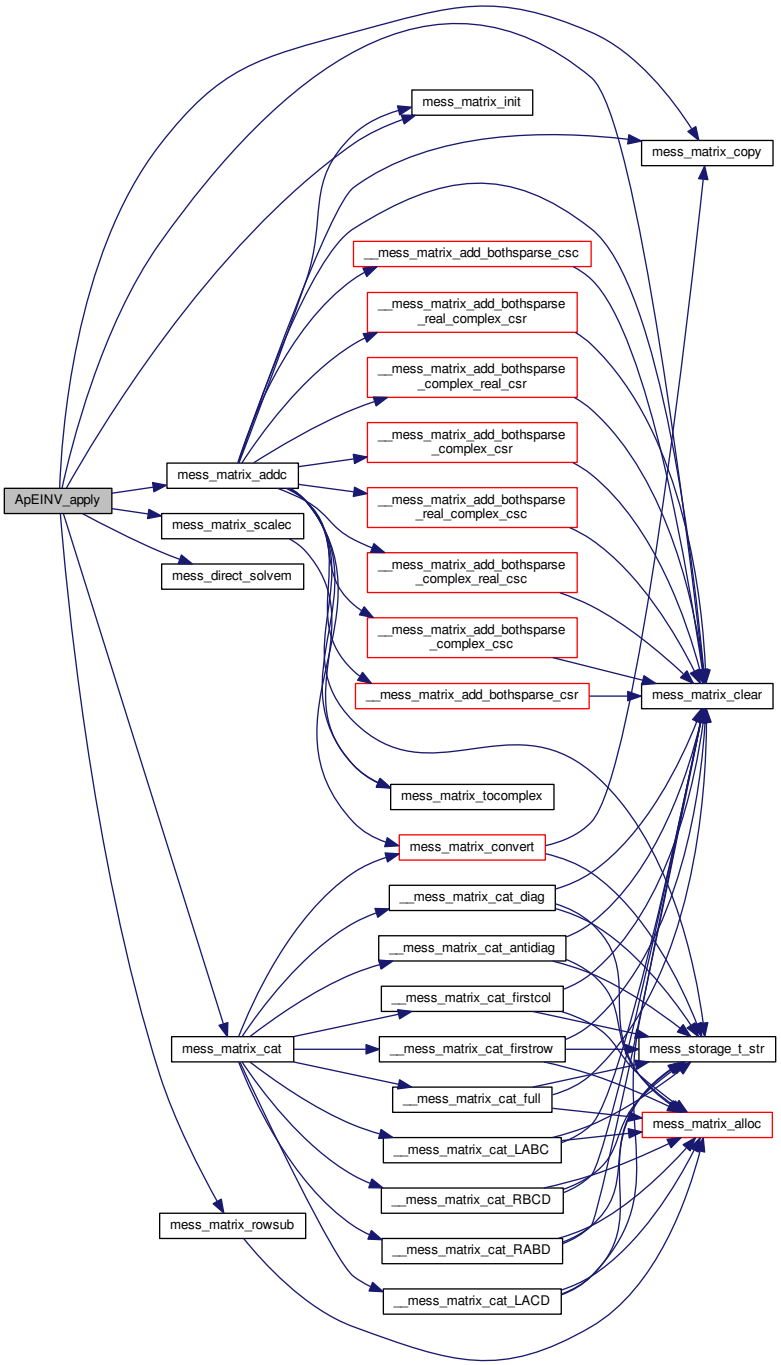
via

$$\begin{aligned} -pKout(1:n,:) &= in(1:n,:) + Kout(n+1:end,:) \\ (K - pD + p^2M)out(n+1:end,:) &= pin(n+1:end,:) - in(1:n,:) \end{aligned}$$

Attention

Internal use only.

Here is the call graph for this function:



10.222.2.3 static int ApEINV_clear (mess_equation *e*) [static]

Parameters

in	*e*	input mess_equation structure

Returns

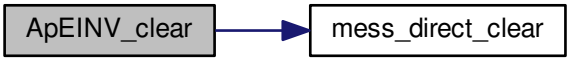
zero on succes or a non-zero error value otherwise

The ApEINV_clear clear generated solvers by using mess_direct_clear the fields of __glyap_so1::shiftsolver are cleared.

Attention

Internal use only.

Here is the call graph for this function:



10.222.2.4 static int ApEINV_generate (mess_equation *e*, mess_vector *parameters*) [static]

Parameters

in	*e*	input mess_equation structure
in	*parameters*	input mess_vector of shift parameters

Returns

zero on succes or a non-zero error value otherwise

The ApEINV_generate builds a several direct solvers for the matrices

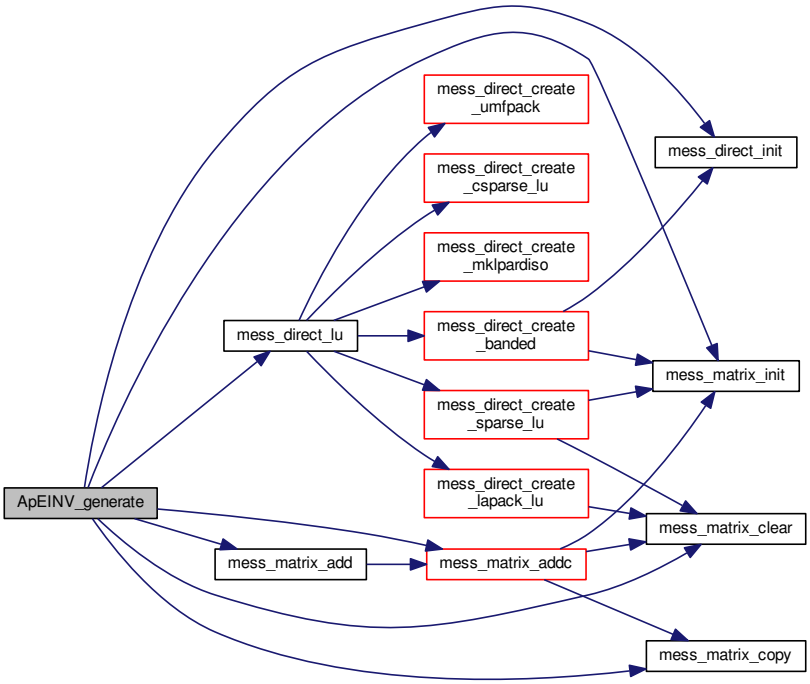
$$\begin{bmatrix} -pK & -K \\ -K & -D+pM \end{bmatrix}.$$

It uses for each shift parameter a mess_direct instance.

Attention

Internal use only.

Here is the call graph for this function:



10.222.2.5 static int AX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

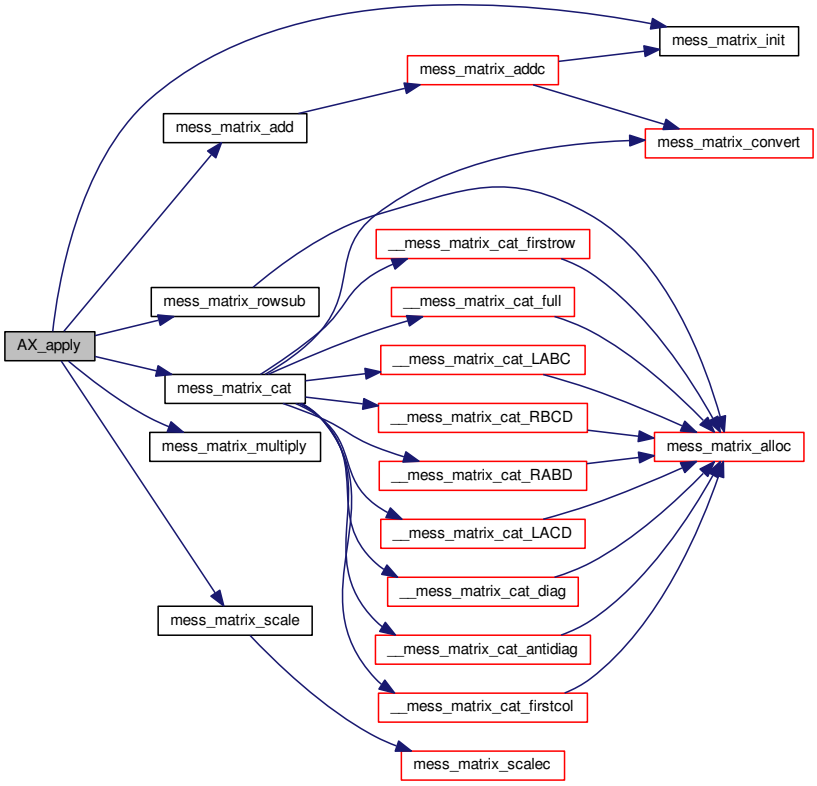
zero on succes or a non-zero error value otherwise

The AX_apply function computes $out = \begin{bmatrix} 0 & -K \\ -K & -D \end{bmatrix} in.$

Attention

Internal use only.

Here is the call graph for this function:



10.222.2.6 static int clear (mess_equation *e*) [static]

Parameters

in,out	*e*	input mess_equation structure

Returns

zero on succes or a non-zero error value otherwise

The clear function, clears the following additional allocated data

field	mathematical description
__glyap_so1::Ksolver	K^{-1}
__glyap_so1::Msolver	M^{-1}

.

Attention

Internal use only.

10.222.2.7 static int EINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

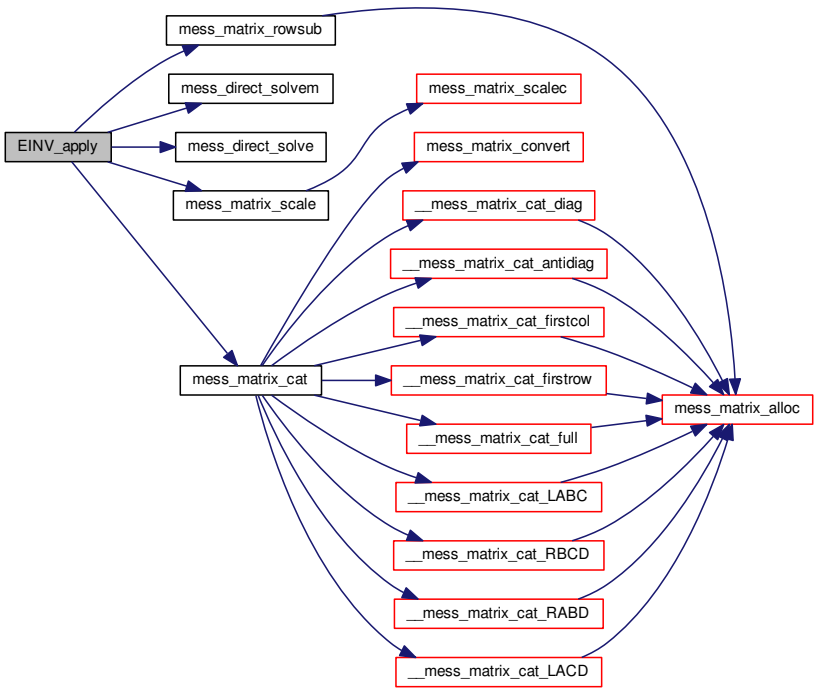
The EINV_apply function solves the system $\begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix} out = in$ via

$$out(1:n,:) = -K^{-1}(in(1:n,:))$$
$$out(n+1:end,: = M^{-1}(in(n+1:end,:))^*$$

Attention

Internal use only.

Here is the call graph for this function:



10.222.2.8 static int EX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

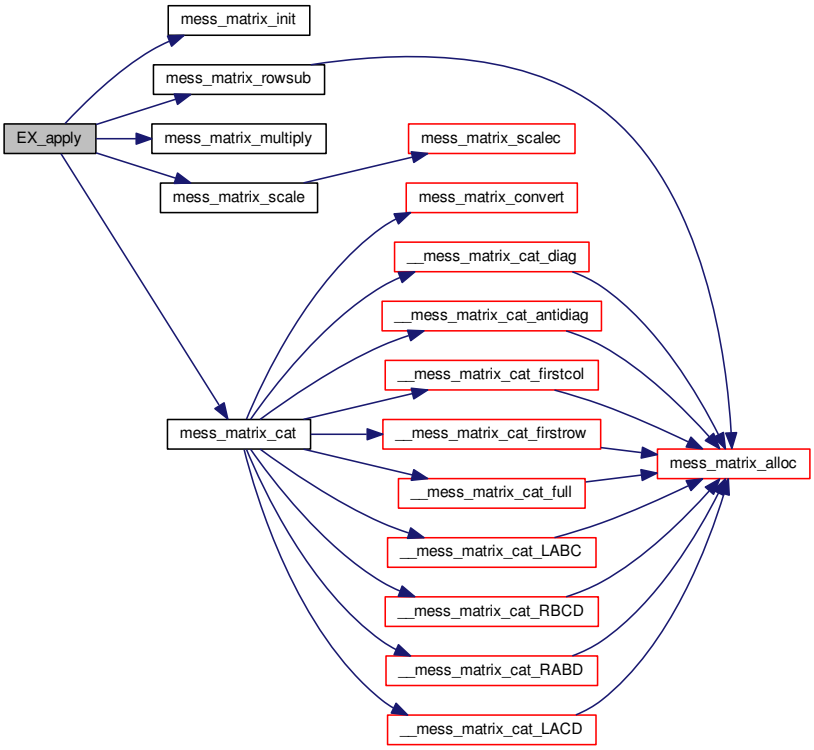
zero on succes or a non-zero error value otherwise

The EX_apply function computes $out = \begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix} in$.

Attention

Internal use only.

Here is the call graph for this function:



10.222.2.9 static int parameter (mess_equation *e*, mess_options *opt*, mess_status *stat*) [static]

Parameters

in	*e*	input mess_equation structure
in	*opt*	input options
in	*stat*	input status

Returns

zero on succes or a non-zero error value otherwise

The parameter function performs the shift parameter computation for the so1 function handles. After calling mess_lrcfadi_parameter shift parameters are removed which are not fullfill the boundness criteria $lowerbound < |p| < upperbound$.

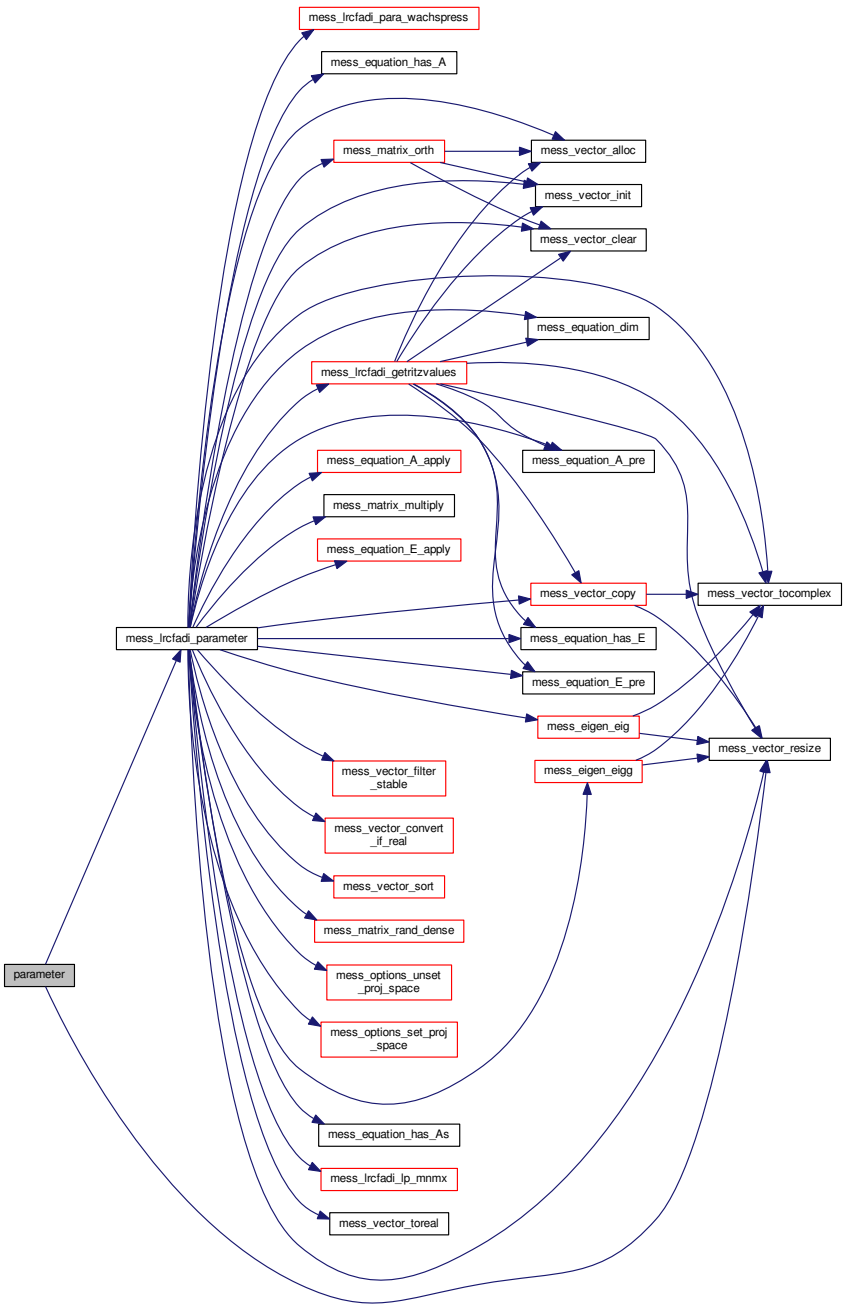
See also

mess_lrcfadi_parameter

Attention

Internal use only.

Here is the call graph for this function:



10.223 equation_glyap_so2.c File Reference

Data Structures

- struct __glyap_so2

  Contains matrices, solvers and additional data for Second Order System function handles.

Functions

- static int AX_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Multiply with operator $\begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix}$ *to a mess_matrix.*
- static int EX_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Multiply with operator $\begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix}$ *to a mess_matrix.*
- static int AINV_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Solve with operator $\begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix}$ *to a mess_matrix.*
- static int EINV_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)

  Solve with $\begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix}$.
- static int ApEINV_generate (mess_equation e, mess_vector parameters)

  Generate a shifted solver for the system matrix $\begin{bmatrix} -pI & -I \\ -K & -D+pM \end{bmatrix}$.
- static int ApEINV_apply (mess_equation e, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)

  Solves shifted linear block system matrix $\begin{bmatrix} -pI & I \\ -K & -D+pM \end{bmatrix}$.
- static int ApEINV_clear (mess_equation e)

  Clear a array of direct solvers.
- static int parameter (mess_equation e, mess_options opt, mess_status stat)

  Call the mess_lrcfadi_parameter and filter shift parameters.
- static int clear (mess_equation e)

  Clear additional allocated field of __glyap_so2 structure.
- int mess_equation_griccati_so2 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix D, mess_matrix K, mess_matrix B, mess_matrix C, double lowerbound, double upperbound)

  Generate an equation object to a generalized Riccati Equation arising from a second order system.
- int mess_equation_glyap_so2 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix D, mess_matrix K, mess_matrix B, double lowerbound, double upperbound)

  Generate an Equation object to a generalized Lyapunov Equation arising from a second order system.

10.223.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.223.2 Function Documentation

10.223.2.1 static int AINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

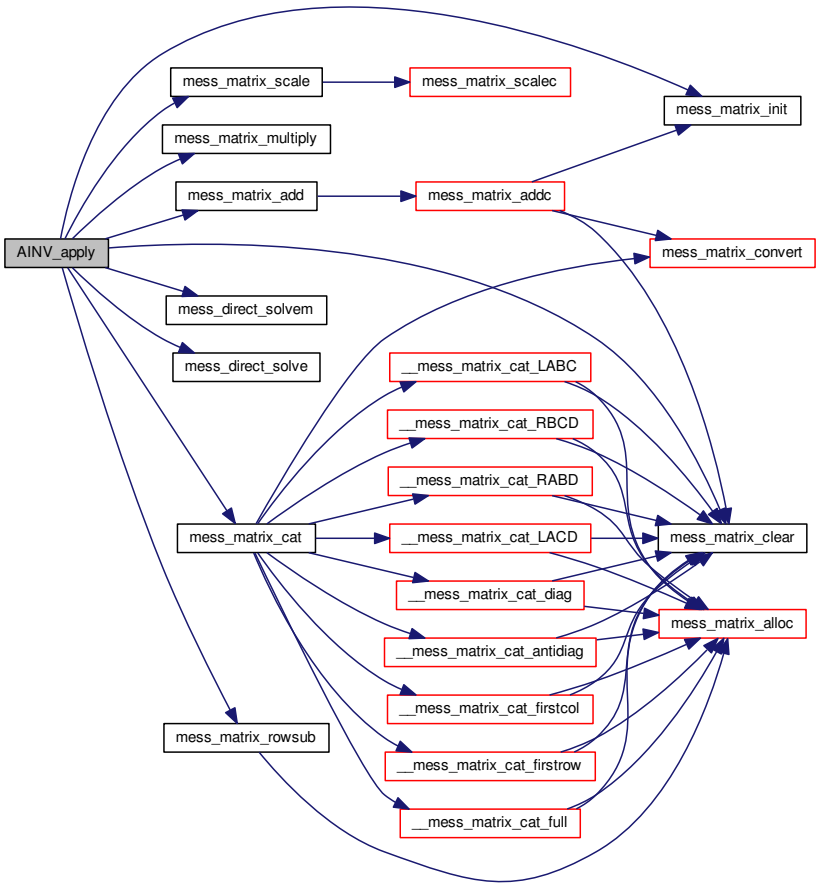
The AINV_apply function solves the system $\begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix} out = in$ via the equivalent formulation

$$\begin{aligned} out(n+1:end,:) &= in(1:n,:) \\ out(1:n,:) &= -K^{-1}(in(n+1:end,:) + Dout(n+1:end,:)) \end{aligned}.$$

Attention

Internal use only.

Here is the call graph for this function:



10.223.2.2 static int ApEINV_apply (mess_equation *e*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *idx_p*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in,out	*e*	input mess_equation structure

Parameters

in	*op*	input operation type
in	*p*	input shift parameter
in	*idx↩ _p*	input index of parameter for shift
in	*in*	input matrix
out	*out*	output matrix

Returns

zero on succes or a non-zero error value otherwise

The ApEINV_apply_memory1 function solves the system

$$op\left(\begin{bmatrix} -pI & I \\ -K & -D+pM \end{bmatrix}\right) out = in$$

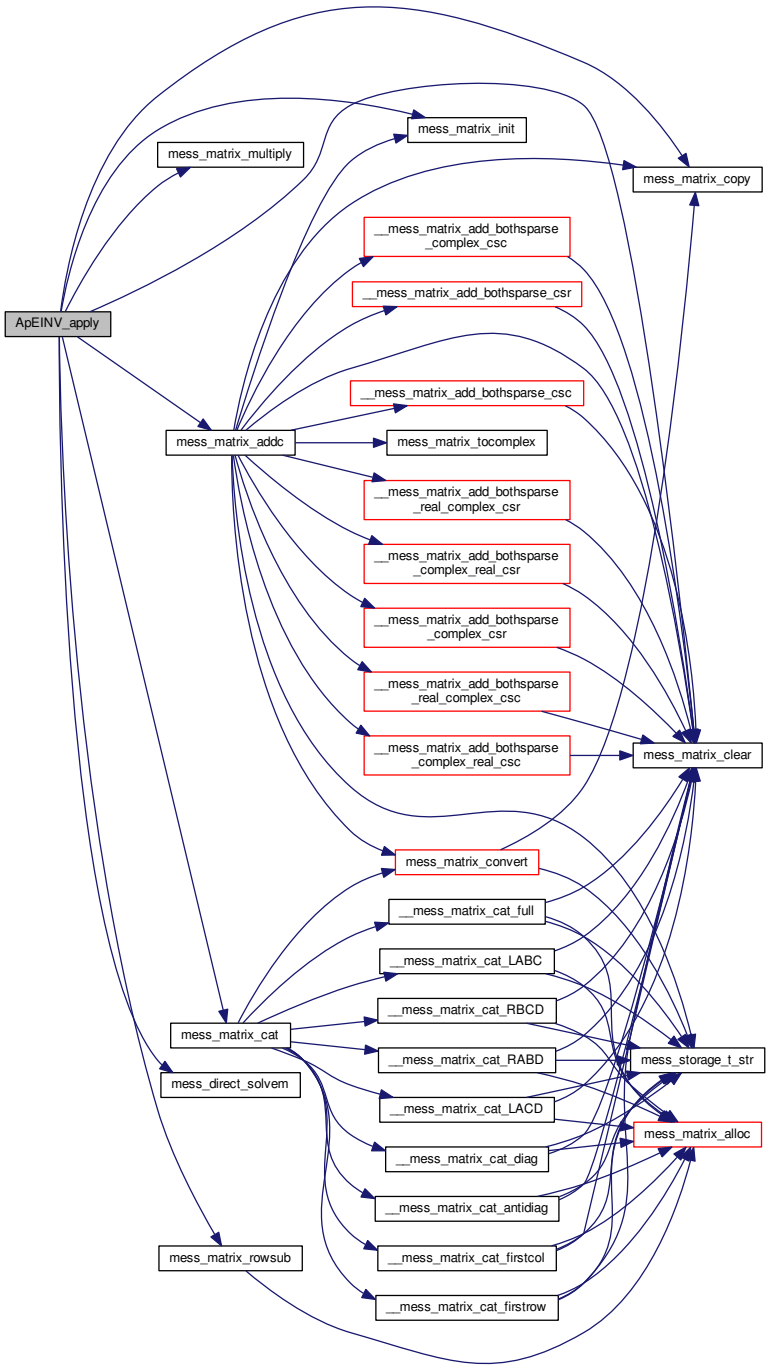
via

$$\begin{aligned} out(n+1:end,:) &= in(1:n,:) - pout(1:n,:) \\ (-K+pD-p^2M)out(1:n,:) &= in(n+1:end,:) + Din(1:n,:) - pMin(1:n,:) \end{aligned}$$

Attention

Internal use only.

Here is the call graph for this function:



10.223.2.3 static int ApEINV_clear (mess_equation *e*) [static]

Parameters

in	*e*	input mess_equation structure

Returns

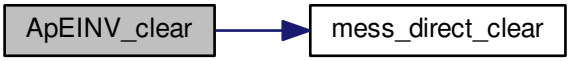
zero on succes or a non-zero error value otherwise

The ApEINV_clear clear generated solvers by using mess_direct_clear the fields of __glyap_so2::shiftsolver are cleared.

Attention

Internal use only.

Here is the call graph for this function:



10.223.2.4 static int ApEINV_generate (mess_equation *e*, mess_vector *parameters*) [static]

Parameters

in	*e*	input mess_equation structure
in	*parameters*	input mess_vector of shift parameters

Returns

zero on succes or a non-zero error value otherwise

The ApEINV_generate builds a several direct solvers for the matrices

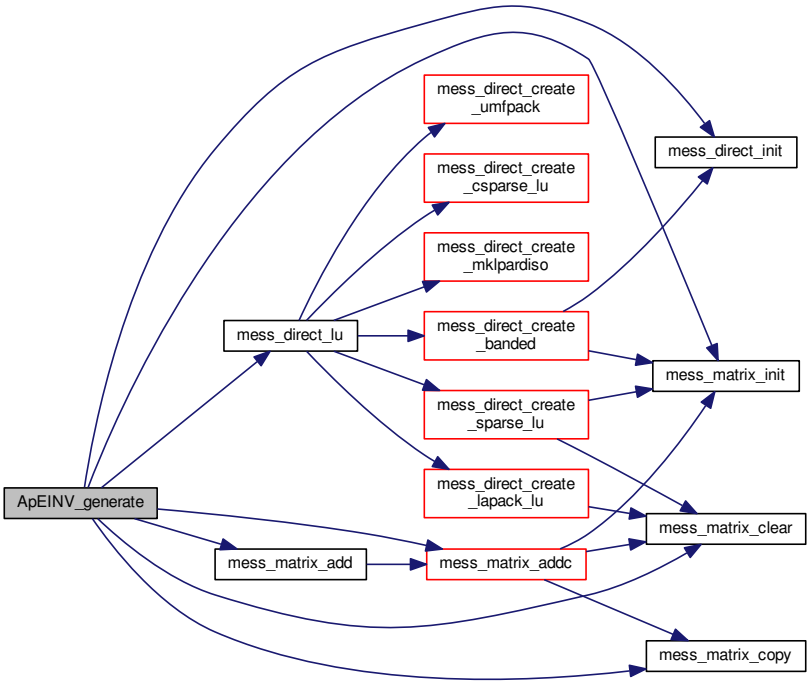
$$\begin{bmatrix} -pI & -I \\ -K & -D+pM \end{bmatrix}.$$

It uses for each shift parameter a mess_direct instance.

Attention

Internal use only.

Here is the call graph for this function:



10.223.2.5 static int AX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

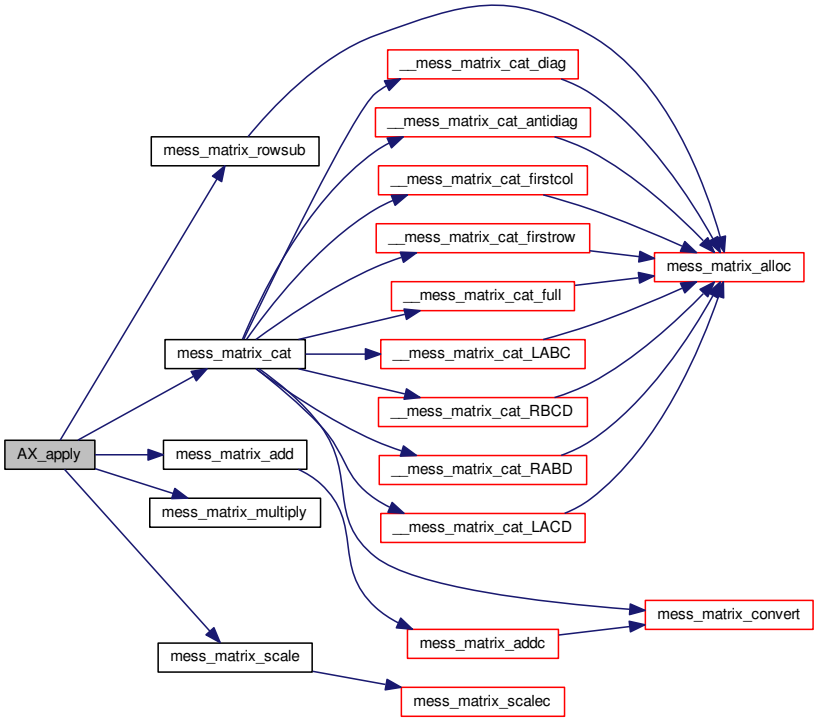
zero on succes or a non-zero error value otherwise

The AX_apply function computes $out = \begin{bmatrix} 0 & I \\ -K & -D \end{bmatrix} in.$

Attention

Internal use only.

Here is the call graph for this function:



10.223.2.6 static int clear (mess_equation *e*) [static]

Parameters

in,out	*e*	input mess_equation structure

Returns

zero on succes or a non-zero error value otherwise

The clear function, clears the following additional allocated data

field	mathematical description
__glyap_so2::Ksolver	K^{-1}
__glyap_so2::Msolver	M^{-1}

.

Attention

Internal use only.

10.223.2.7 static int EINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

zero on succes or a non-zero error value otherwise

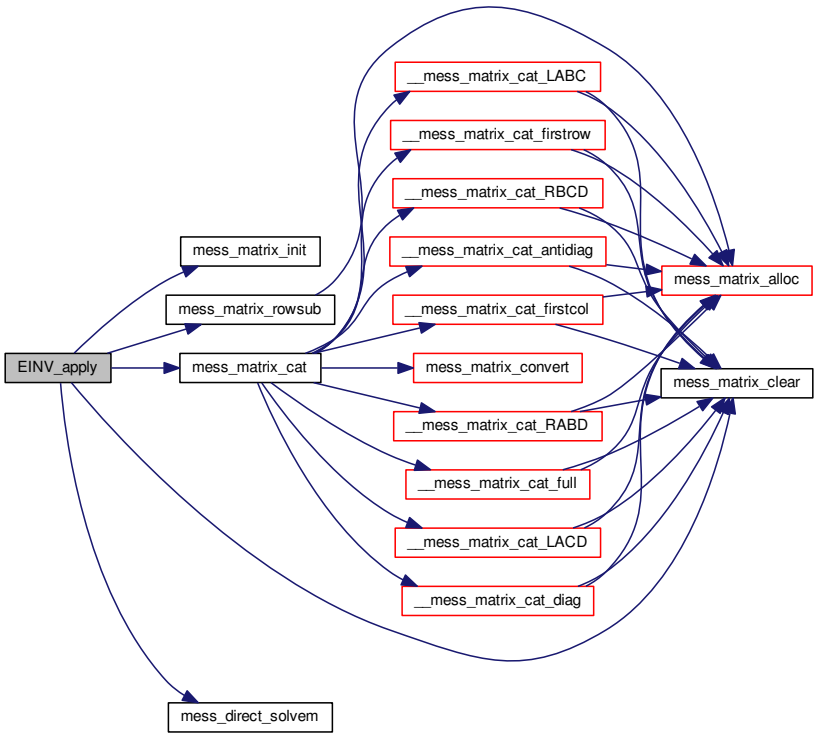
The EINV_apply function solves the system $\begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix} out = in$ via

$$out(1:n,:) = in(1:n,:)$$
$$out(n+1:end,:) = M^{-1}(in(n+1:end,:))^*$$

Attention

Internal use only.

Here is the call graph for this function:



10.223.2.8 static int EX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to block matrix
in	*in*	input matrix in
out	*out*	output matrix out

Returns

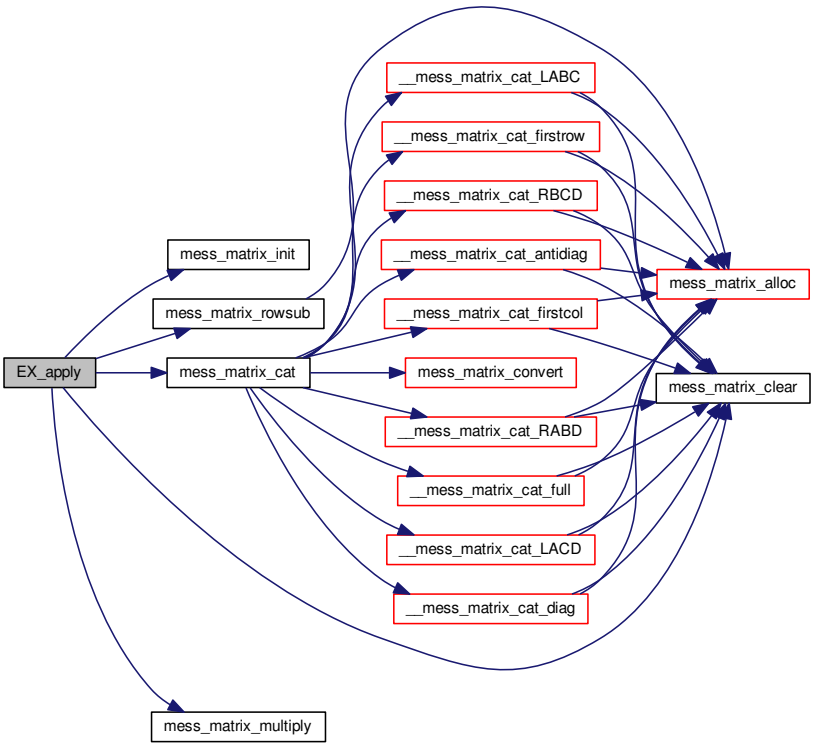
zero on succes or a non-zero error value otherwise

The EX_apply function computes $out = \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix} in$.

Attention

Internal use only.

Here is the call graph for this function:



10.223.2.9 static int parameter (mess_equation *e*, mess_options *opt*, mess_status *stat*) [static]

Parameters

in	*e*	input mess_equation structure
in	*opt*	input options
in	*stat*	input status

Returns

zero on succes or a non-zero error value otherwise

The parameter function performs the shift parameter computation for the so1 function handles. After calling mess_lrcfadi_parameter shift parameters are removed which are not fullfill the boundness criteria $lowerbound < |p| < upperbound$.

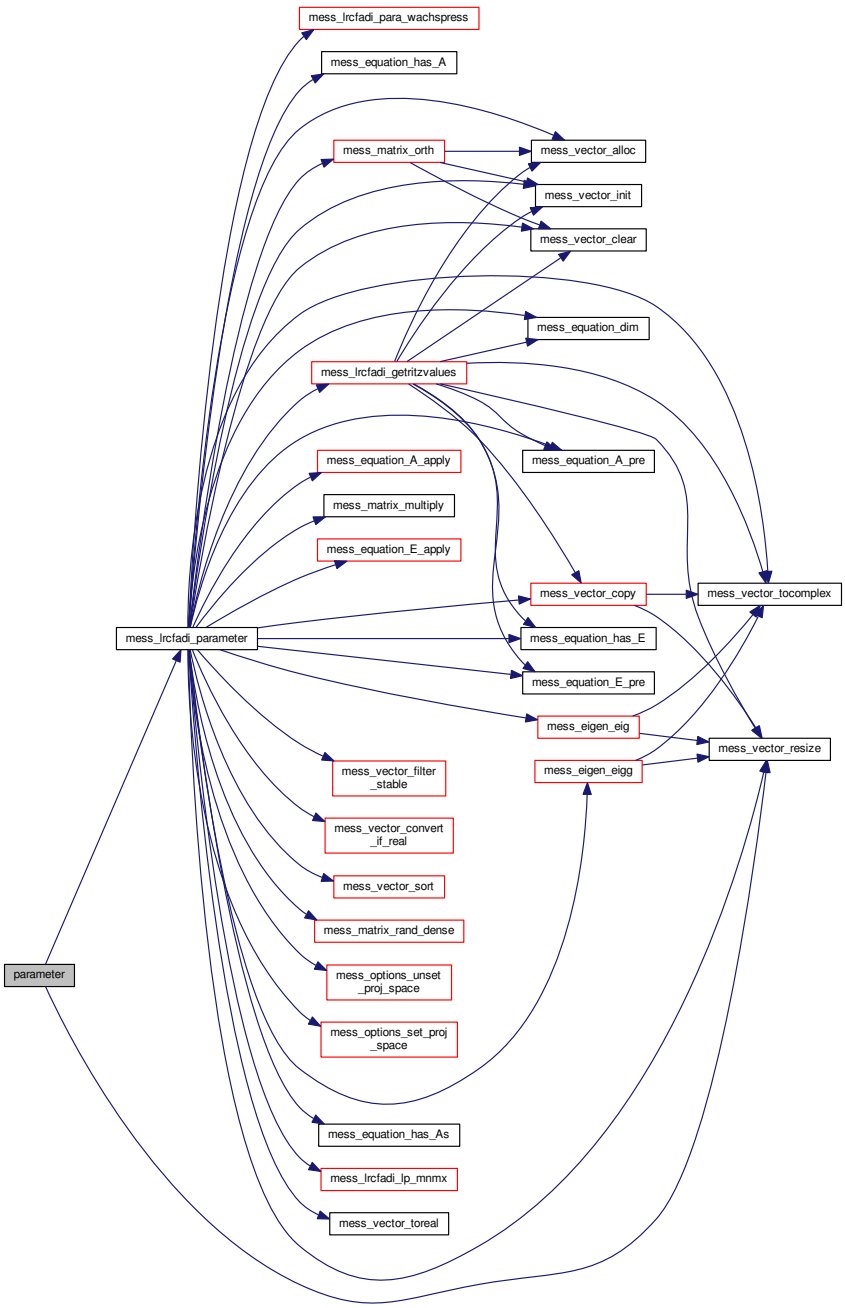
See also

mess_lrcfadi_parameter

Attention

Internal use only.

Here is the call graph for this function:



10.224 equation_lyap.c File Reference

Data Structures

- struct __lyap

  Contains matrices, solvers and additional data for a generalized Lyapunov Equation.

Functions

- static int __lyap_A_multiplym (mess_equation e, mess_operation_t op, mess_matrix X, mess_matrix Y)

  Compute matrix-matrix multitplication with system matrix A.
- static int __lyap_AINV_generate (mess_equation e)

  Generate a direct solver for system matrix A.
- static int __lyap_AINV_apply (mess_equation e, mess_operation_t op, mess_matrix Y, mess_matrix X)

  Solve system $op(A)X = Y$.
- static int __lyap_AINV_clear (mess_equation e)

  Clear a direct solver for system matrix A.
- static int __lyap_ApE_generate (mess_equation e, mess_vector parameters)
- static int __lyap_ApE_solve (mess_equation e, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)
- static int __lyap_ApE_clear (mess_equation e)
- static int __lyap_ApEX_generate (mess_equation e, mess_vector parameters)
- static int __lyap_ApEX_apply (mess_equation e, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)
- static int __lyap_ApEX_clear (mess_equation e)
- static int __lyap_clear (mess_equation e)
- static int parameter (mess_equation e, mess_options opt, mess_status stat)

  Call the mess_lrcfadi_parameter function.
- int mess_equation_std_lyap (mess_equation e, mess_options opt, mess_matrix A, mess_matrix B)

  Setup a mess_Equation object to solve a standard Lyapunov Equation.
- int mess_equation_std_riccati (mess_equation e, mess_options opt, mess_matrix A, mess_matrix B, mess_matrix C)

  Setup a mess_equation object to solve a standard Riccati Equation.
- int mess_equation_lyap (mess_equation e, mess_options opt, mess_matrix A, mess_matrix E, mess_matrix B)

  Setup a mess_Equation object to solve a matrix-valued Lyapunov Equation.
- int mess_equation_riccati (mess_equation e, mess_options opt, mess_matrix A, mess_matrix E, mess_matrix B, mess_matrix C)

  Setup a mess_equation object to solve a matrix-valued Riccati Equation.

10.224.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Generate the mess_equation object for the standard Lyapunov and Riccati Equations.

10.224.2 Function Documentation

10.224.2.1 static int __lyap_A_multiplym (mess_equation *e*, mess_operation_t *op*, mess_matrix *X*, mess_matrix *Y*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to A
in	*X*	input matrix X
out	*Y*	output matrix Y

Returns

zero on succes or a non-zero error value otherwise

The __lyap_A_multiplym function computes the matrix-matrix product, i.e.

$$Y = op(A)X.$$

See also

mess_matrix_multiply

Attention

Internal use only.

Here is the call graph for this function:

10.224.2.2 static int __lyap_AINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *Y*, mess_matrix *X*) [static]

Parameters

in	*e*	input mess_equation structure
in	*op*	input operation applied to A
in	*Y*	input matrix right-hand side Y
out	*X*	output matrix solution X

Returns

zero on succes or a non-zero error value otherwise

The __lyap_AINV_apply function solves $op(A)X = Y$.

See also

mess_direct_solvem

Attention

Internal use only.

Here is the call graph for this function:

10.224.2.3 static int __lyap_AINV_clear (mess_equation *e*) [static]

Parameters

in,out	*e*	input mess_equation structure

Returns

zero on succes or a non-zero error value otherwise

The __lyap_AINV_clear function clears a generated solver in __lyap::Asolver.

See also

mess_direct_clear

Attention

Internal use only.

Here is the call graph for this function:

10.224.2.4 static int __lyap_AINV_generate (mess_equation *e*) [static]

Parameters

in,out	*e*	input mess_equation structure

Returns

zero on succes or a non-zero error value otherwise

The __lyap_AINV_generate function generates a mess_direct_lu solver for system matrix A.
After function call the __lyap::Asolver field holds a generated mess_direct_lu solver. If __lyap::Asolver is already generated 0 is return immediately.

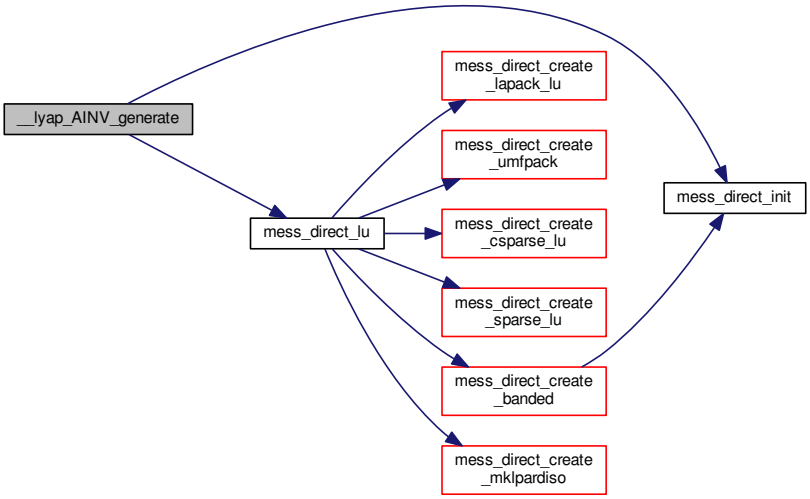
See also

mess_direct_init
mess_direct_lu

Attention

Internal use only.

Here is the call graph for this function:



10.224.2.5 static int __lyap_ApE_clear (mess_equation *e*) [static]

Parameters

in,out	*e*	input mess_equation structure

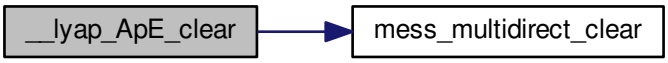
Returns

zero on succes or a non-zero error value otherwise

Attention

Internal use only.

Here is the call graph for this function:



10.224.2.6 static int __lyap_ApE_generate (mess_equation *e*, mess_vector *parameters*) [static]

Parameters

in,out	*e*	input mess_equation structure
in	*parameters*	input shift parameter vector

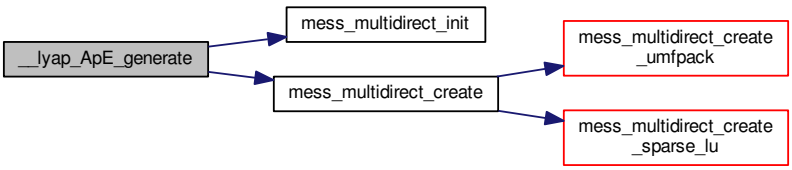
Returns

zero on succes or a non-zero error value otherwise

Attention

Internal use only.

Here is the call graph for this function:



10.224.2.7 static int __lyap_ApE_solve (mess_equation *e*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *idx_p*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in,out	*e*	input mess_equation structure
in	*op*	input operation type
in	*p*	input shift parameter
in	*idx↩ _p*	input index of parameter for shift
in	*in*	input matrix
out	*out*	output matrix

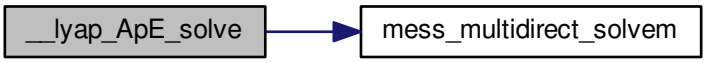
Returns

zero on succes or a non-zero error value otherwise

Attention

Internal use only.

Here is the call graph for this function:



10.224.2.8 static int __lyap_ApEX_apply (mess_equation *e*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *idx_p*, mess_matrix *in*, mess_matrix *out*) [static]

Parameters

in,out	*e*	input mess_equation structure
in	*op*	input operation type
in	*p*	input shift parameter
in	*idx↩ _p*	input index of parameter for shift
in	*in*	input matrix
out	*out*	output matrix

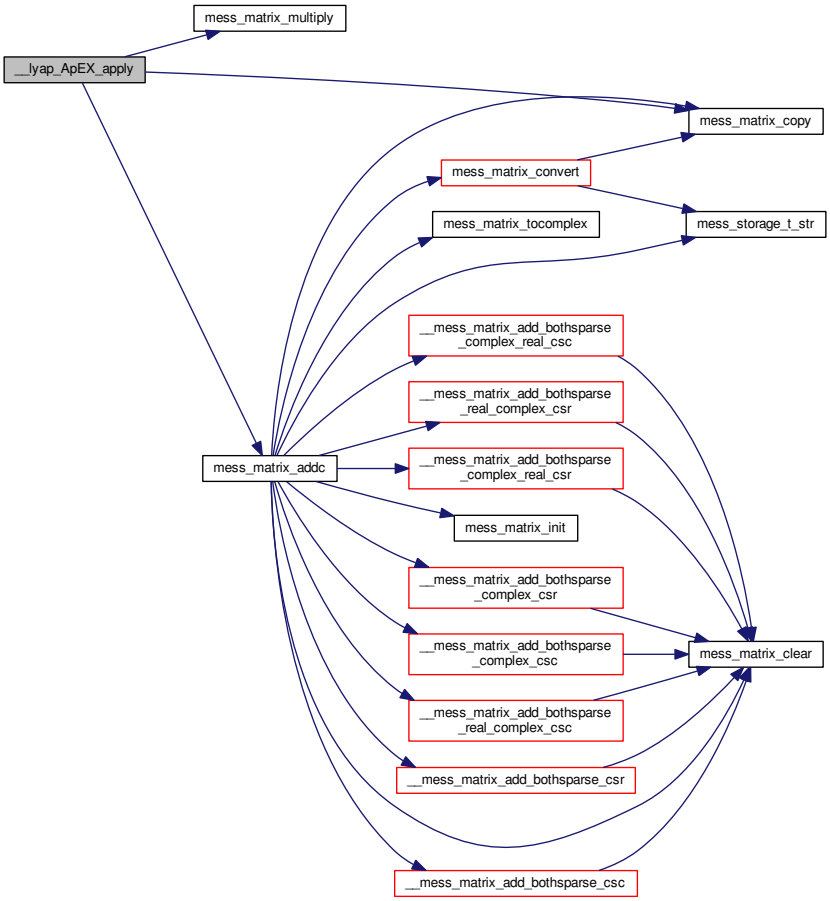
Returns

zero on succes or a non-zero error value otherwise

Attention

Internal use only.

Here is the call graph for this function:



10.224.2.9 static int __lyap_ApEX_clear (mess_equation *e*) [static]

Parameters

in,out	*e*	input mess_equation structure

Returns

zero on succes or a non-zero error value otherwise

Attention

Internal use only.

Here is the call graph for this function:

10.224.2.10 static int __lyap_ApEX_generate (mess_equation *e*, mess_vector *parameters*) [static]

Parameters

in,out	*e*	input mess_equation structure
in	*parameters*	input vector of shift parameters

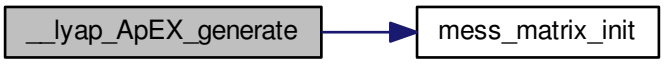
Returns

zero on succes or a non-zero error value otherwise

Attention

Internal use only.

Here is the call graph for this function:



10.224.2.11 static int __lyap_clear (mess_equation *e*) [static]

Parameters

in,out	*e*	input mess_equation structure

Returns

zero on succes or a non-zero error value otherwise

Attention

Internal use only.

10.224.2.12 static int parameter (mess_equation *e*, mess_options *opt*, mess_status *stat*) [static]

Parameters

in	*e*	input mess_equation structure
in	*opt*	input options
in	*stat*	input status

Returns

zero on succes or a non-zero error value otherwise

The parameter function calls directly mess_lrcfadi_parameter. If mess_equation::parent of `in` is not `NULL` the equation was stabilized by a feedback and mess_lrcfadi_parameter is called with the stabilized equation object mess_equation::parent otherwise `e` is simply used.

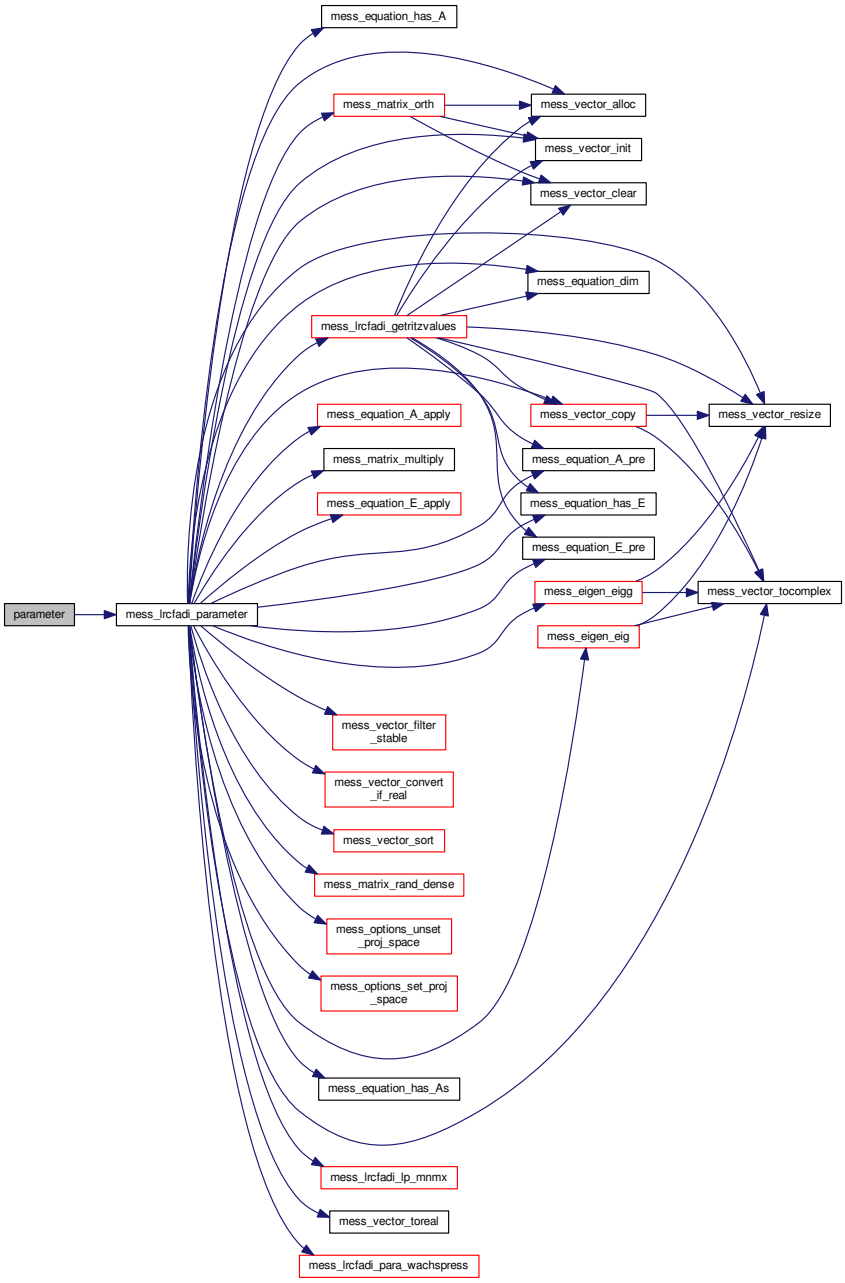
See also

mess_lrcfadi_parameter

Attention

Internal use only.

Here is the call graph for this function:



10.225 equation_stable.c File Reference

Data Structures

- struct __stable_lyap

Macros

- #define IS_REAL(X) (cimag((X))==0.0)

Functions

- static int AX_generate (mess_equation e)
- static int AX_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int AX_clear (mess_equation e)
- static int EX_generate (mess_equation e)
- static int EX_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int EX_clear (mess_equation e)
- static int EINV_generate (mess_equation e)
- static int EINV_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int EINV_clear (mess_equation e)
- static int ApEINV_generate (mess_equation e, mess_vector parameters)
- static int ApEINV_apply (mess_equation e, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)
- static int ApEINV_clear (mess_equation e)
- static int AINV_generate (mess_equation e)
- static int AINV_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int AINV_clear (mess_equation e)
- static int init_rhs (mess_equation e, mess_options opt)
- static int parameter (mess_equation e, mess_options opt, mess_status stat)
- static int clear (mess_equation e)
- int mess_equation_stable (mess_equation e, mess_options opt, mess_equation child, mess_matrix U, mess_matrix V)

  Generate a stabilized Lyapunov Equation object.

10.225.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

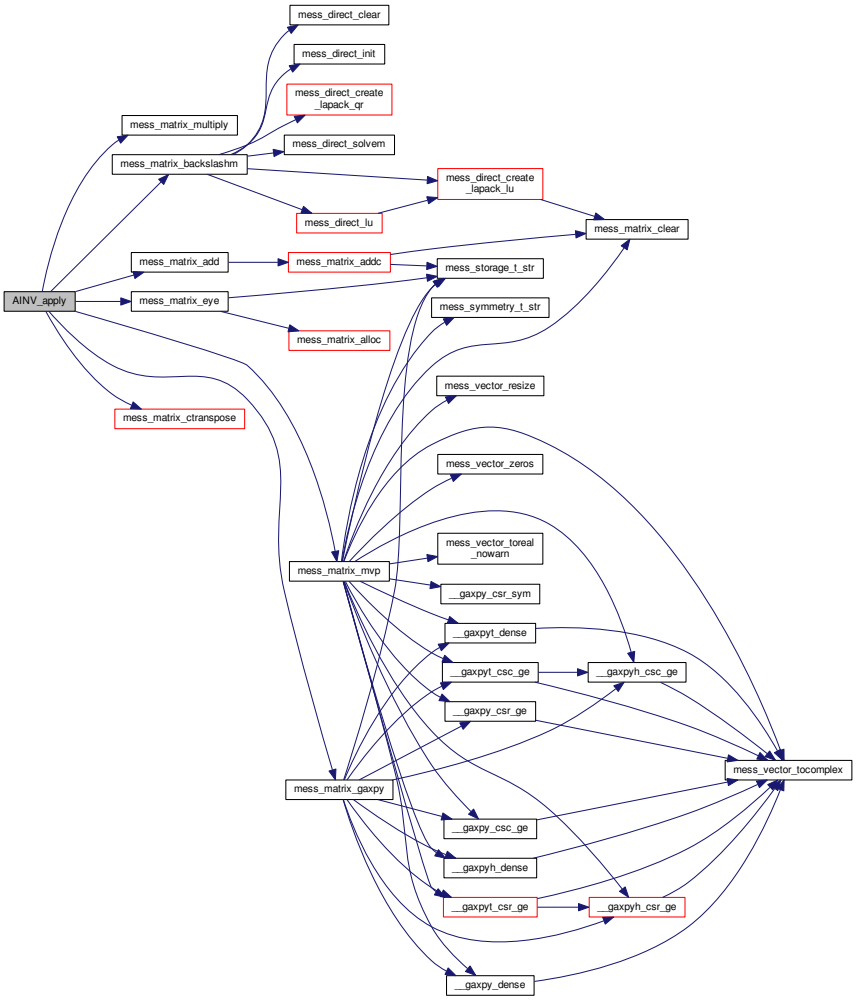
10.225.2 Macro Definition Documentation

10.225.2.1 #define IS_REAL(X) (cimag((X))==0.0)

10.225.3 Function Documentation

10.225.3.1 static int AINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Here is the call graph for this function:

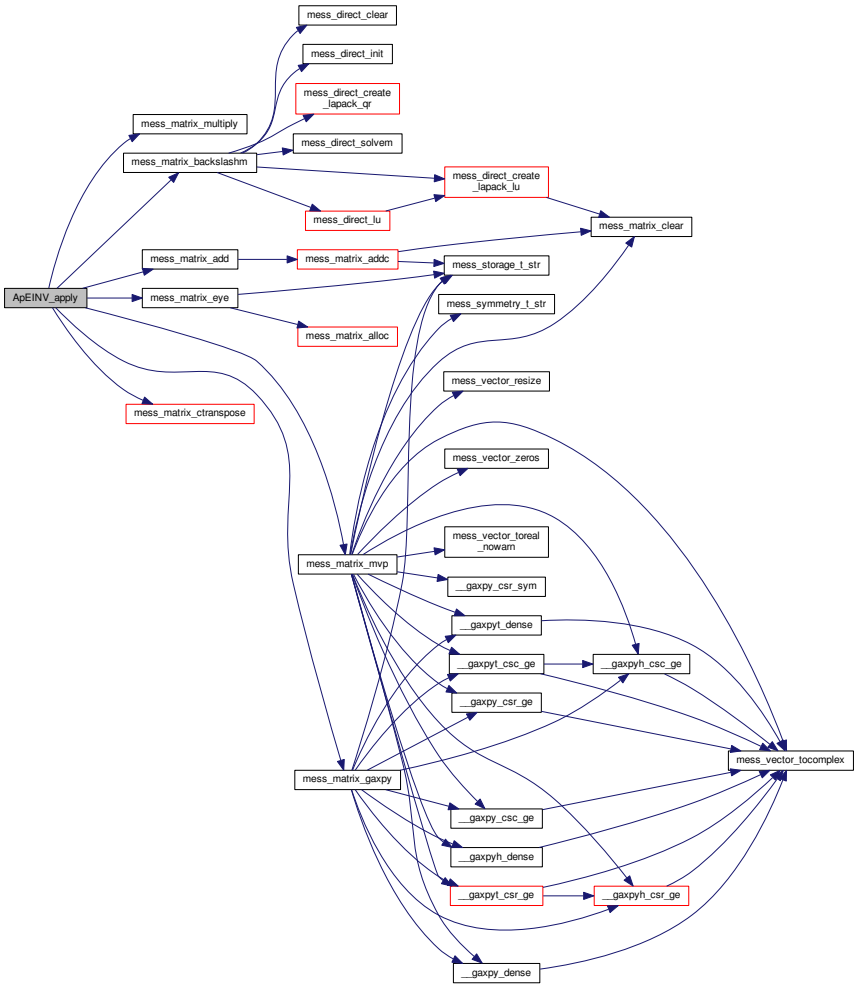


10.225.3.2 static int AINV_clear (mess_equation *e*) [static]

10.225.3.3 static int AINV_generate (mess_equation *e*) [static]

10.225.3.4 static int ApEINV_apply (mess_equation *e*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *idx_p*, mess_matrix *in*, mess_matrix *out*) [static]

Here is the call graph for this function:

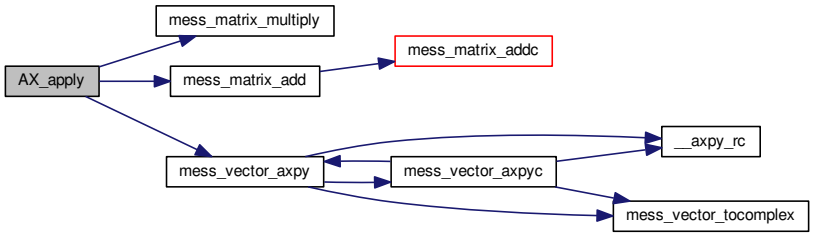


10.225.3.5 static int ApEINV_clear (mess_equation *e*) [static]

10.225.3.6 static int ApEINV_generate (mess_equation *e*, mess_vector *parameters*) [static]

10.225.3.7 static int AX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

Here is the call graph for this function:



10.225.3.8 static int AX_clear (mess_equation *e*) [static]

10.225.3.9 static int AX_generate (mess_equation *e*) [static]

10.225.3.10 static int clear (mess_equation *e*) [static]

10.225.3.11 static int EINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

10.225.3.12 static int EINV_clear (mess_equation *e*) [static]

10.225.3.13 static int EINV_generate (mess_equation *e*) [static]

10.225.3.14 static int EX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

10.225.3.15 static int EX_clear (mess_equation *e*) [static]

10.225.3.16 static int EX_generate (mess_equation *e*) [static]

10.225.3.17 static int init_rhs (mess_equation *e*, mess_options *opt*) [static]

10.225.3.18 static int parameter (mess_equation *e*, mess_options *opt*, mess_status *stat*) [static]

10.226 equations.c File Reference

Data Structures

- struct _equations_t

  Structure for all **Py-M.E.S.S.** *to* **M.E.S.S.** *converters. Structure for all* **Py-M.E.S.S.** *to* **M.E.S.S.** *converters.*

Macros

- #define PRE_POST_FUNCTION(cname, pyname)
- #define PRE_POST_FUNCTION2(cname, pyname)
- #define APPLY_FUNCTION(cname, pyname)
- #define APPLY_FUNCTION2(cname, pyname)
- #define SET_IF_IMPLEMENTED(EQN_NAME, C_NAME, PY_NAME)

Typedefs

- typedef mess_equation(∗ eqn_converter) (PyObject ∗, mess_freelist)
- typedef struct _equations_t equations_t

  Structure for all **`Py-M.E.S.S.`** *to* **`M.E.S.S.`** *converters. Structure for all* **`Py-M.E.S.S.`** *to* **`M.E.S.S.`** *converters.*

Functions

- static int op_to_int (mess_operation_t op)
- static int callback_AX_generate (mess_equation e)
- static int callback_AX_clear (mess_equation e)
- static int callback_AX_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int callback_EX_generate (mess_equation e)
- static int callback_EX_clear (mess_equation e)
- static int callback_EX_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int callback_AINV_generate (mess_equation e)
- static int callback_AINV_clear (mess_equation e)
- static int callback_AINV_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int callback_EINV_generate (mess_equation e)
- static int callback_EINV_clear (mess_equation e)
- static int callback_EINV_apply (mess_equation e, mess_operation_t op, mess_matrix in, mess_matrix out)
- static int callback_ApEX_generate (mess_equation e, mess_vector p)
- static int callback_ApEX_clear (mess_equation e)
- static int callback_ApEX_apply (mess_equation e, mess_operation_t op, mess_double_cpx_t p, mess_int_t idx_p, mess_matrix in, mess_matrix out)
- static int callback_ApEINV_generate (mess_equation e, mess_vector p)
- static int callback_ApEINV_clear (mess_equation e)
- static int callback_ApEINV_apply (mess_equation e, mess_operation_t op, mess_double_cpx_t p, mess_↩ int_t idx_p, mess_matrix in, mess_matrix out)
- static int parameter (mess_equation eqn, mess_options opt, mess_status stat)
- static int clear (mess_equation e)
- mess_equation mess_equation_from_python (PyObject ∗obj, mess_freelist mem, mess_equation_t eqn_↩ type, int ∗need_callback)

Variables

- static equations_t eqns [ ]

10.226.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Nitin Prasad (nitin.prasad93@gmail.com, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.226.2 Macro Definition Documentation

10.226.2.1 #define APPLY_FUNCTION(*cname, pyname*)

10.226.2.2 #define APPLY_FUNCTION2(*cname, pyname*)

10.226.2.3 #define PRE_POST_FUNCTION(*cname, pyname*)

Value:

```
static int cname (mess_equation e) {\
    MSG_FNAME(__func__); \
    PyObject *pyOper;\
    PyObject *pyResult;\
    PyObject *pyError = NULL;\
    /* Check input  */\
    mess_check_nullpointer(e); \
    pyOper = (PyObject*) e->aux;\
    mess_check_nullpointer(pyOper);\
    /*  Call Function  */\
    Py_INCREF(pyOper);\
    PyErr_Clear(); \
    pyResult =  PyObject_CallMethod(pyOper, #pyname, NULL);\
    Py_XDECREF(pyResult);\
    pyError = PyErr_Occurred(); \
    if ( pyError != NULL ) { \
        PyErr_Print(); \
        MSG_ERROR("The Python callback for %s failed.\n", #pyname);\
        Py_DECREF(pyOper);\
        return MESS_ERROR_PYTHON; \
    }\
    Py_DECREF(pyOper);\
    return 0;\
}
```

10.226.2.4 #define PRE_POST_FUNCTION2(*cname, pyname*)

Value:

```
static int cname (mess_equation e, mess_vector p) {\
    MSG_FNAME(__func__); \
    PyObject *pyOper;\
    PyObject *pyResult;\
    PyObject *pyP; \
    PyObject *pyError = NULL;\
    /* Check input  */\
    mess_check_nullpointer(e); \
    pyOper = (PyObject*) e->aux;\
    mess_check_nullpointer(pyOper);\
    /*  Call Function  */\
    pyP = vector_to_python(p,1);\
    if ( pyP == NULL) { \
        MSG_ERROR("Cannot convert the parameter vector to Python.\n");\
    }\
    Py_INCREF(pyOper);\
    PyErr_Clear(); \
    pyResult =  PyObject_CallMethod(pyOper, #pyname, "O", pyP);\
    Py_XDECREF(pyResult);\
    Py_XDECREF(pyP); \
    pyError = PyErr_Occurred(); \
    if ( pyError != NULL ) { \
        PyErr_Print(); \
        Py_DECREF(pyOper);\
        return MESS_ERROR_PYTHON; \
    }\
    Py_DECREF(pyOper);\
    return 0;\
}
```

10.226.2.5 #define SET_IF_IMPLEMENTED(*EQN_NAME, C_NAME, PY_NAME*)

Value:

```
do {\
    if ( pymess_is_reimplemented(pymess_eqn_base, obj, #PY_NAME) == 1) { \
        /*  MSG_PRINT("Set %s to %s\n", #C_NAME, #PY_NAME);*/\
        eqn->EQN_NAME = C_NAME;\
    } else { \
        eqn->EQN_NAME = NULL; \
    }\
} while (0)
```

10.226.3 Typedef Documentation

10.226.3.1 typedef mess_equation(∗ eqn_converter) (PyObject ∗, mess_freelist)

10.226.3.2 typedef struct _equations_t equations_t

Attention

Internal use only.

10.226.4 Function Documentation

10.226.4.1 static int callback_AINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

10.226.4.2 static int callback_AINV_clear (mess_equation *e*) [static]

10.226.4.3 static int callback_AINV_generate (mess_equation *e*) [static]

10.226.4.4 static int callback_ApEINV_apply (mess_equation *e*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *idx_p*, mess_matrix *in*, mess_matrix *out*) [static]

10.226.4.5 static int callback_ApEINV_clear (mess_equation *e*) [static]

10.226.4.6 static int callback_ApEINV_generate (mess_equation *e*, mess_vector *p*) [static]

10.226.4.7 static int callback_ApEX_apply (mess_equation *e*, mess_operation_t *op*, mess_double_cpx_t *p*, mess_int_t *idx_p*, mess_matrix *in*, mess_matrix *out*) [static]

10.226.4.8 static int callback_ApEX_clear (mess_equation *e*) [static]

10.226.4.9 static int callback_ApEX_generate (mess_equation *e*, mess_vector *p*) [static]

10.226.4.10 static int callback_AX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

10.226.4.11 static int callback_AX_clear (mess_equation *e*) [static]

10.226.4.12 static int callback_AX_generate (mess_equation *e*) [static]

10.226.4.13 static int callback_EINV_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

10.226.4.14 static int callback_EINV_clear (mess_equation *e*) [static]

10.226.4.15 static int callback_EINV_generate (mess_equation *e*) [static]

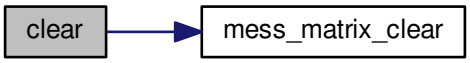
10.226.4.16 static int callback_EX_apply (mess_equation *e*, mess_operation_t *op*, mess_matrix *in*, mess_matrix *out*) [static]

10.226.4.17 static int callback_EX_clear (mess_equation *e*) [static]

10.226.4.18 static int callback_EX_generate (mess_equation *e*) [static]

10.226.4.19 static int clear (mess_equation *e*) [static]

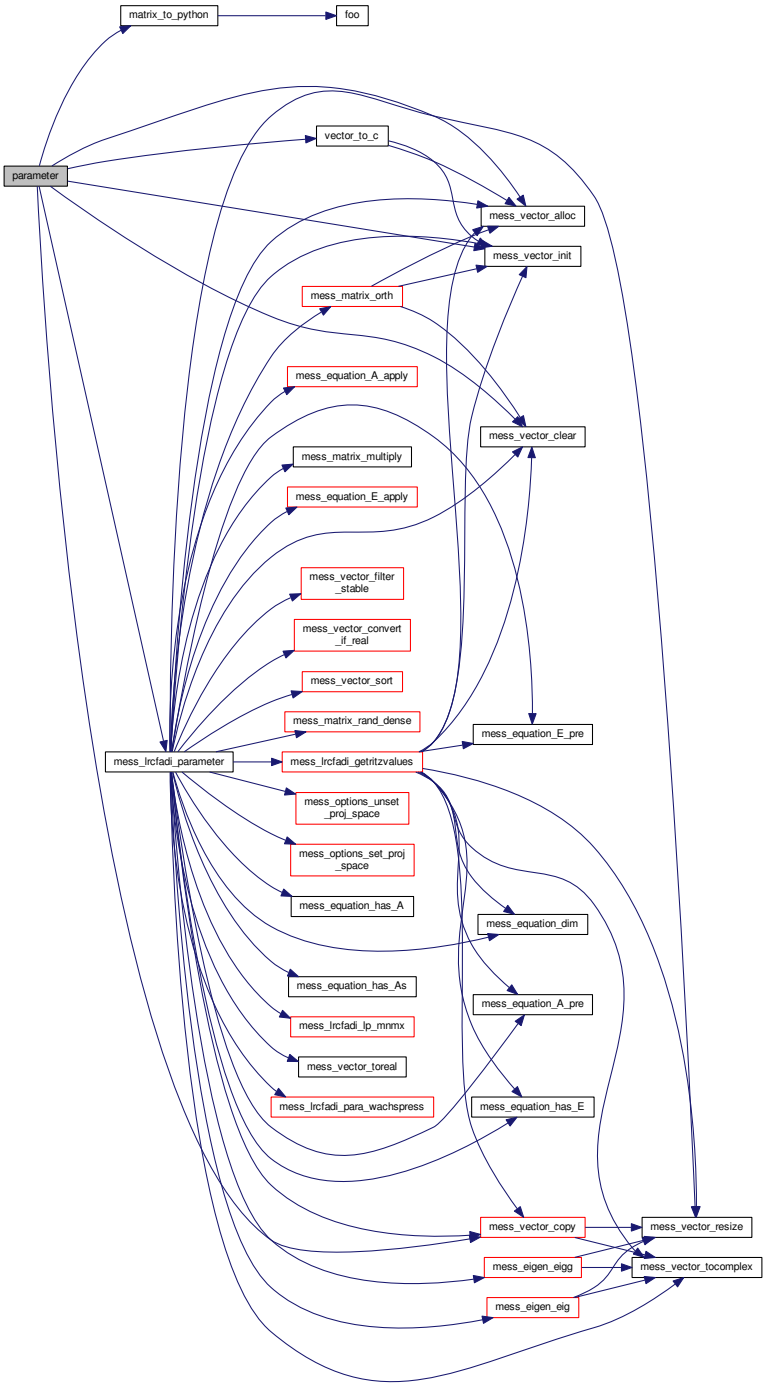
Here is the call graph for this function:



10.226.4.20 static int op_to_int (mess_operation_t *op*) [static]

10.226.4.21 static int parameter (mess_equation *eqn*, mess_options *opt*, mess_status *stat*) [static]

Here is the call graph for this function:



10.226.5 Variable Documentation

10.226.5.1 equations_t eqns[ ] [static]

Initial value:

```
= {
    {"EquationGLyap",                eqn_conv_lyap},
    {"EquationGRiccati",             eqn_conv_riccati},
    {"EquationGLyapSO1",             eqn_conv_lyap_so1},
    {"EquationGRiccatiSO1",          eqn_conv_riccati_so1},
    {"EquationGLyapSO2",             eqn_conv_lyap_so2},
    {"EquationGRiccatiSO2",          eqn_conv_riccati_so2},
    {"EquationGLyapDAE1",            eqn_conv_lyap_dae1},
    {"EquationGRiccatiDAE1",         eqn_conv_riccati_dae1},
    {"EquationGLyapDAE2",            eqn_conv_lyap_dae2},
    {"EquationGRiccatiDAE2",         eqn_conv_riccati_dae2},
    {NULL, NULL}
}
```

10.227 error.c File Reference

Functions

- int mess_set_errorlevel (int level)

  Set error level.
- char ∗ mess_get_error (int code)

  Return an error message.

10.227.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.228 error_macro.h File Reference

Macros

- #define BTRSIZE 50
- #define MSG_ERROR(text, args...)
- #define MSG_WARN(text, args...)
- #define MSG_INFO(text, args...)
- #define MSG_WARN_NOBT(text, args...)
- #define MSG_PRINT(text, args...)
- #define MSG_FNAME(nn) static const char∗ __funcname = nn
- #define FUNCTION_FAILURE_HANDLE(err, cond, fun)
- #define FUNCTION_FAILURE_HANDLE_OMP(err, cond, fun)

10.228.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.228.2 Macro Definition Documentation

10.228.2.1 #define BTRSIZE 50

10.229 errors.h File Reference

Macros

- #define MESS_SUCCESS 0
- #define MESS_ERROR_PYTHON 225
- #define MESS_ERROR_DYNSYS 229
- #define MESS_ERROR_CONVERGE 230
- #define MESS_ERROR_GENERAL 231
- #define MESS_ERROR_MEMLOC 233
- #define MESS_ERROR_LAPACK 234
- #define MESS_ERROR_EIGENVALUES 235
- #define MESS_ERROR_NOSUPPORT 236
- #define MESS_ERROR_MISSING 237
- #define MESS_ERROR_UMFPACK 238
- #define MESS_ERROR_MEMORY 239
- #define MESS_ERROR_NOTLOAD 240
- #define MESS_ERROR_NOPATTERN 242
- #define MESS_ERROR_NOCOMPLEX 243
- #define MESS_ERROR_SINGULAR 244
- #define MESS_ERROR_DIMENSION 245
- #define MESS_ERROR_CONVERT 246
- #define MESS_ERROR_SYMMETRIC 247
- #define MESS_ERROR_ARGUMENTS 248
- #define MESS_ERROR_STORAGETYPE 249
- #define MESS_ERROR_DATATYPE 250
- #define MESS_ERROR_DATA 251
- #define MESS_ERROR_WRONG_HEADER 252
- #define MESS_ERROR_FILEIO 253
- #define MESS_ERROR_NULLPOINTER 254
- #define MESS_ERROR_NOTNULLPOINTER 255
- #define MESS_ERROR_MISC 256
- #define MESS_ERROR_CHOLMOD 257
- #define MESS_ERROR_MKLPARDISO 258
- #define MESS_ERROR_PYTHON_SIG 259

Functions

- char * mess_get_error (int code)

  Return an error message.
- int mess_set_errorlevel (int level)

  Set error level.

Variables

- int mess_error_level

10.229.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.230 eye.c File Reference

Macros

- #define MIN(A, B) ((A)<(B))?(A):(B)

Functions

- int mess_matrix_eye (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_storage_t store)

  Generate a real identity matrix.
- int mess_matrix_eyec (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_storage_t store)

  Generate a complex identity matrix.

10.230.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.230.2 Macro Definition Documentation

10.230.2.1 #define MIN(A, B) ((A)<(B))?(A):(B)

10.231 fdm_matrix.c File Reference

Functions

- static double zero_func (double x, double y)
- int mess_matgen_fdmmatrix (mess_matrix A, mess_int_t n0, mess_matgen_fdm_function fnfx, mess_matgen_fdm_function fnfy, mess_matgen_fdm_function fng)

  Generate a stiffness matrix A for the finite difference discretization (equidistant grid) of the PDE.
- int mess_matgen_fdmvector (mess_vector v, mess_int_t n0, mess_matgen_fdm_function func)

  Generate a vector v containing values of a function $f(x,y)$ on an equidistant grid in the interior of the unit square.
- int mess_matgen_fdmcolumn (mess_matrix B, mess_int_t n0, mess_matgen_fdm_function func)

  Generate a one-column matrix containing values of a function $f(x,y)$ on an equidistant grid in the interior of the unit square.
- int mess_matgen_fdmrow (mess_matrix C, mess_int_t n0, mess_matgen_fdm_function func)

  Generates a one-row matrix containing values of a function $f(x,y)$ on an equidistant grid in the interior of the unit square.

10.231.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.231.2 Function Documentation

10.231.2.1 static double zero_func (double *x*, double *y*) [static]

10.232 feature_tests.c File Reference

Functions

- int main (int argc, char **argv)

Variables

- const char features [ ]

10.232.1 Function Documentation

10.232.1.1 int main (int *argc*, char ** *argv*)

10.232.2 Variable Documentation

10.232.2.1 const char features[ ]

10.233 feature_tests.cxx File Reference

Functions

- int main (int argc, char **argv)

Variables

- const char features [ ]

10.233.1 Function Documentation

10.233.1.1 int main (int *argc*, char ** *argv*)

10.233.2 Variable Documentation

10.233.2.1 const char features[ ]

10.234 fill.c File Reference

Functions

- int mess_matrix_symfillup (mess_matrix mat)

  Fill up the symmetry in a coordinate matrix.

10.234.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.235 filter.c File Reference

Functions

- int mess_vector_filter (mess_vector in, int(∗filter_real)(const double ∗), int(∗filter_complex)(const mess_double_cpx_t ∗))

  Filter values from vector.
- static int __filter_stable_real (const double ∗val)
- static int __filter_stable_complex (const mess_double_cpx_t ∗val)
- int mess_vector_filter_stable (mess_vector in)

  Filter values with positive realpart from vector.

10.235.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.235.2 Function Documentation

10.235.2.1 static int __filter_stable_complex (const mess_double_cpx_t ∗ *val*) [static]

10.235.2.2 static int __filter_stable_real (const double ∗ *val*) [static]

10.236 filter2D.C File Reference

10.237 fortran_translate.h File Reference

Macros

- #define F77_GLOBAL(name, NAME) name##_
- #define F77_GLOBAL_(name, NAME) name##_
- #define F77_MODULE(mod_name, name, mod_NAME, NAME) __##mod_name##_MOD_##name
- #define F77_MODULE_(mod_name, name, mod_NAME, NAME) __##mod_name##_MOD_##name

10.237.1 Macro Definition Documentation

10.237.1.1 #define F77_GLOBAL(*name, NAME*) name##_

10.237.1.2 #define F77_GLOBAL_(*name, NAME*) name##_

10.237.1.3 #define F77_MODULE(*mod_name, name, mod_NAME, NAME*) __##mod_name##_MOD_##name

10.237.1.4 #define F77_MODULE_(*mod_name, name, mod_NAME, NAME*) __##mod_name##_MOD_##name

10.238 freelist.c File Reference

Macros

- #define MESS_FREELIST_ADD_GENERIC(FNAME, ADDTYPE, ADDVAR, LISTVARCOUNTER, LISTVARPOINTER)

Functions

- int mess_freelist_init (mess_freelist *list)

  Initialized a mess_freelist instance.
- int mess_freelist_add_ptr (mess_freelist list, void *ptr)

  Insert an arbitrary pointer in the memory manager.
- int mess_freelist_add_mess_vector (mess_freelist list, mess_vector vec)

  Insert a mess_vector in the memory manager.
- int mess_freelist_add_mess_matrix (mess_freelist list, mess_matrix mat)

  Insert a mess_matrix in the memory manager.
- int mess_freelist_add_mess_equation (mess_freelist list, mess_equation eqn)

  Insert a mess_equation in the memory manager.
- int mess_freelist_add_mess_options (mess_freelist list, mess_options opt)

  Insert a mess_options structure in the memory manager.
- int mess_freelist_add_mess_status (mess_freelist list, mess_status stat)

  Insert a mess_status structure in the memory manager.
- int mess_freelist_print (mess_freelist list)

  Print a mess_freelist instance.
- int mess_freelist_clear (mess_freelist *list)

  Free all objects stored in the memory manager.

10.238.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.238.2 Macro Definition Documentation

10.238.2.1 #define MESS_FREELIST_ADD_GENERIC(*FNAME, ADDTYPE, ADDVAR, LISTVARCOUNTER, LISTVARPOINTER*)

Value:

```
int FNAME(mess_freelist list, ADDTYPE ADDVAR){
        \
        MSG_FNAME(__func__);
                \
        int ret=0;                                                                    \
        size_t newlen = list->LISTVARCOUNTER+1;
      \
        mess_try_realloc(list->LISTVARPOINTER , ADDTYPE*, sizeof(ADDTYPE)* newlen);
                        \
        list->LISTVARPOINTER[list-> LISTVARCOUNTER] = ADDVAR;                         \
        list->LISTVARCOUNTER = newlen;                                                \
        return ret;                                                                   \
    }
```

10.239 freelist.h File Reference

Data Structures

- struct mess_freelist_st

  Structure containing a minimal memory manage in order to avoid memory leaks in the Python *and* ***MEX-M.E.S.S.*** *interface.*

Typedefs

- typedef struct mess_freelist_st * mess_freelist

  Structure containing a minimal memory manage in order to avoid memory leaks in the `Python` *and* ***MEX-M.E.S.S.*** *interface.*

Functions

- int mess_freelist_init (mess_freelist *list)

  Initialized a mess_freelist instance.
- int mess_freelist_print (mess_freelist list)

  Print a mess_freelist instance.
- int mess_freelist_add_ptr (mess_freelist list, void *ptr)

  Insert an arbitrary pointer in the memory manager.
- int mess_freelist_add_mess_vector (mess_freelist list, mess_vector vec)

  Insert a mess_vector in the memory manager.
- int mess_freelist_add_mess_matrix (mess_freelist list, mess_matrix mat)

  Insert a mess_matrix in the memory manager.
- int mess_freelist_add_mess_equation (mess_freelist list, mess_equation eqn)

  Insert a mess_equation in the memory manager.
- int mess_freelist_add_mess_options (mess_freelist list, mess_options opt)

  Insert a mess_options structure in the memory manager.
- int mess_freelist_add_mess_status (mess_freelist list, mess_status stat)

  Insert a mess_status structure in the memory manager.
- int mess_freelist_clear (mess_freelist *list)

  Free all objects stored in the memory manager.

10.239.1 Detailed Description

Author

Maximilian Behr (`behr@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.240 galerkin.c File Reference

Macros

- #define IS_REAL(X) (cimag((X))==0.0)

Functions

- int mess_lrcfadi_galerkin (mess_equation eqn, mess_options opt, mess_equation_t eqn_type, mess_matrix Z)

  Refine a solution of a matrix equation using Galerkin projection.

10.240.1 Detailed Description

Author

Martin Köhler (`koehlerm@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.240.2 Macro Definition Documentation

10.240.2.1 #define IS_REAL(X) (cimag((X))==0.0)

10.241 gaxpy.c File Reference

Functions

- int mess_matrix_gaxpy (mess_operation_t op, mess_matrix A, mess_vector x, mess_vector y)

  Compute a matrix-vector product with update of the output.

10.241.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.242 getcol.c File Reference

Functions

- int mess_matrix_getcol (mess_matrix matrix, mess_int_t col, mess_vector c)

  Get a column from a matrix.
- int mess_matrix_setcol (mess_matrix matrix, mess_int_t col, mess_vector colv)

  Set a column in a matrix.

10.242.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.243 getelement.c File Reference

Functions

- int mess_matrix_getelement (mess_matrix matrix, mess_int_t row, mess_int_t col, double *val, mess_double_cpx_t *valc)

  Get an element from a matrix.

10.243.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.244 getrow.c File Reference

Functions

- int mess_matrix_getrow (mess_matrix matrix, mess_int_t row, mess_vector r)

  Get a row from a matrix.
- int mess_matrix_setrow (mess_matrix matrix, mess_int_t row, mess_vector rowv)

  Set a row in a matrix.

10.244.1 Detailed Description

Author

Martin Köhler (`koehlerm@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.245 getset.c File Reference

Functions

- int mess_vector_get (mess_vector v, mess_int_t i, mess_double_cpx_t *val)

  Get entry `i` of a mess_vector instance.

10.245.1 Detailed Description

Author

Maximilian Behr (`behr@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.246 gettypes.c File Reference

Macros

- #define MESS_ENUM_STR_1(RETALT, ARG, ENUM) if(ARG == ENUM){ return #ENUM; } return RETALT;
- #define MESS_ENUM_STR_2(RETALT, ARG, ENUM, ...) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_1(RETALT,ARG,__VA_ARGS__) }
- #define MESS_ENUM_STR_3(RETALT, ARG, ENUM, ...) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_2(RETALT,ARG,__VA_ARGS__) }
- #define MESS_ENUM_STR_4(RETALT, ARG, ENUM, ...) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_3(RETALT,ARG,__VA_ARGS__) }
- #define MESS_ENUM_STR_5(RETALT, ARG, ENUM, ...) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_4(RETALT,ARG,__VA_ARGS__) }
- #define MESS_ENUM_STR_6(RETALT, ARG, ENUM, ...) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_5(RETALT,ARG,__VA_ARGS__) }
- #define MESS_ENUM_STR_7(RETALT, ARG, ENUM, ...) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_6(RETALT,ARG,__VA_ARGS__) }
- #define MESS_ENUM_STR_8(RETALT, ARG, ENUM, ...) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_7(RETALT,ARG,__VA_ARGS__) }
- #define MESS_ENUM_STR_9(RETALT, ARG, ENUM, ...) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_8(RETALT,ARG,__VA_ARGS__) }

Functions

- const char * mess_datatype_t_str (const mess_datatype_t data_type)

  Return a human readable name of a data type value.
- const char * mess_storage_t_str (const mess_storage_t store_type)

  Return a human readable name of a storage type value.
- const char * mess_symmetry_t_str (const mess_symmetry_t sym)

  Return a human readable name of a symmetry type value.
- const char * mess_operation_t_str (const mess_operation_t op)

  Return the human readable name of an operation type value.
- const char * mess_equation_t_str (const mess_equation_t eqn_type)

  Convert the mess_equation_t to a human readable string.
- const char * mess_parameter_t_str (const mess_parameter_t para_type)

  Convert the mess_parameter_t to a human readable string.
- const char * mess_memusage_t_str (const mess_memusage_t mem_type)

  Convert the mess_memusage_t to a human readable string.
- const char * mess_norm_t_str (const mess_norm_t nrm_type)

  Convert the mess_norm_t to a human readable string.
- const char * mess_sign_scale_t_str (const mess_sign_scale_t sc_type)

  Convert the mess_sign_scale_t to a human readable string.
- const char * mess_direct_lupackage_t_str (const mess_direct_lupackage_t lu_type)

  Convert the mess_direct_lupackage_t to a human readable string.
- const char * mess_direct_cholpackage_t_str (const mess_direct_cholpackage_t chol_type)

  Convert the mess_direct_cholpackage_t to a human readable string.
- const char * mess_multidirect_t_str (const mess_multidirect_t ml_type)

  Convert the mess_multidirect_t to a human readable string.
- const char * mess_residual_t_str (const mess_residual_t res_type)

  Convert the mess_residual_t to a human readable string.
- const char * mess_print_format_t_str (const mess_print_format_t print_type)

  Convert the mess_print_format_t to a human readable string.

10.246.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file contain function to retrieve human readable strings from type definitions and type enumerations.

10.246.2 Macro Definition Documentation

10.246.2.1 #define MESS_ENUM_STR_1(*RETALT, ARG, ENUM*) if(ARG == ENUM){ return #ENUM; } return RETALT;

10.246.2.2 #define MESS_ENUM_STR_2(*RETALT, ARG, ENUM, ...*) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_1(RETALT,ARG,__VA_ARGS__) }

10.246.2.3 #define MESS_ENUM_STR_3(*RETALT, ARG, ENUM, ...*) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_2(RETALT,ARG,__VA_ARGS__) }

10.246.2.4 #define MESS_ENUM_STR_4(*RETALT, ARG, ENUM, ...*) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_3(RETALT,ARG,__VA_ARGS__) }

10.246.2.5 #define MESS_ENUM_STR_5(*RETALT, ARG, ENUM, ...*) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_4(RETALT,ARG,__VA_ARGS__) }

10.246.2.6 #define MESS_ENUM_STR_6(*RETALT, ARG, ENUM, ...*) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_5(RETALT,ARG,__VA_ARGS__) }

10.246.2.7 #define MESS_ENUM_STR_7(*RETALT, ARG, ENUM, ...*) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_6(RETALT,ARG,__VA_ARGS__) }

10.246.2.8 #define MESS_ENUM_STR_8(*RETALT, ARG, ENUM, ...*) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_7(RETALT,ARG,__VA_ARGS__) }

10.246.2.9 #define MESS_ENUM_STR_9(*RETALT, ARG, ENUM, ...*) if(ARG == ENUM){ return #ENUM; } else { MESS_ENUM_STR_8(RETALT,ARG,__VA_ARGS__) }

10.246.3 Function Documentation

10.246.3.1 const char* mess_print_format_t_str (const mess_print_format_t *print_type*)

Parameters

in	*print_type*	type of print methods

Returns

human readable string of `print_type`

The mess_print_format_t_str function converts a mess_print_format_t to a human readable string.

10.247 gglrsrm.c File Reference

Functions

- int mess_bt_gglrsrm (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Generalized low-rank square root balanced truncation for first order GLTI to GLTI systems.

10.247.1 Detailed Description

Author

Martin Köhler (`koehlerm@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.248 glyap.c File Reference

Functions

- PyObject * pymess_glyap (PyObject *self, PyObject *args)

  Interface to mess_glyap for use with `Python`.

10.248.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.249 glyap3.c File Reference

Data Structures

- struct _glyap3_st

Macros

- #define GLYAP3_NB 64
- #define GLYAP3_ISOLVE 1

Typedefs

- typedef struct _glyap3_st _glyap3

Enumerations

Functions

- const char * glyap3_eqn_t_str (const _glyap3_eqn_t eqn_t)

  Return a human readable name of a data type value.
- static int glyap3_clear (void *solver)

  Clean up handler for _glyap3.
- static int glyap3_solvemx (mess_operation_t op, void *data, mess_matrix Y, mess_matrix X)

  Solve the Lyapunov or Stein Equation using _glyap3 and GLYAP3
- static int glyap3_solvem (void *data, mess_matrix Y, mess_matrix X)

  Solve function the Lyapunov or Stein Equation using _glyap3 and GLYAP3
- static int glyap3_solvemt (void *data, mess_matrix Y, mess_matrix X)

  Solve function the Lyapunov or Stein Equation using _glyap3 and GLYAP3
- int mess_direct_create_glyap3 (_glyap3_eqn_t eqn_t, mess_matrix A, mess_matrix E, mess_direct solver)

  Create a solver the Lyapunov or Stein Equation using _glyap3 and GLYAP3
- int mess_direct_create_generalized_lyapunov (mess_matrix A, mess_matrix E, mess_direct solver)

  Generate a dense standard / generalized Lyapunov equation solver.
- int mess_direct_create_generalized_stein (mess_matrix A, mess_matrix E, mess_direct solver)

  Generate a dense standard / generalized Stein equation solver.
- static int glyapchol_solvemx (mess_operation_t op, void *data, mess_matrix B, mess_matrix Z)

  Solve the generalized Lyapunov Equation for Cholesky Factor.
- static int glyapchol_solvem (void *data, mess_matrix B, mess_matrix Z)

  Solve the generalized Lyapunov Equation for Cholesky Factor.
- static int glyapchol_solvemt (void *data, mess_matrix B, mess_matrix Z)

  Solve the generalized Lyapunov Equation for Cholesky Factor.
- int mess_direct_create_generalized_lyapunovchol (mess_matrix A, mess_matrix E, mess_direct solver)

  Generate a dense standard / generalized Lyapunov equation solver for the cholesky factor.

- int mess_glyap3x (_glyap3_eqn_t eqn_t, mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix Y, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix X)

  Solve a standard / generalized Lyapunov / Stein equation.
- int mess_tglyap3x (_glyap3_eqn_t eqn_t, mess_operation_t op, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix Y, mess_matrix X)

  Solve a standard / generalized Lyapunov / Stein equation from given Schur decomposition.
- int mess_glyap (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix Y, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix X)

  Solve a standard / generalized Lyapunov equation.
- int mess_gstein (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix Y, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix X)

  Solve a standard / generalized Stein equation.
- int mess_tglyap (mess_operation_t op, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix Y, mess_matrix X)

  Solve a standard / generalized Lyapunov equation from given Schur decomposition.
- int mess_tgstein (mess_operation_t op, mess_matrix Ahat, mess_matrix QA, mess_matrix Ehat, mess_matrix QE, mess_matrix Y, mess_matrix X)

  Solve a standard / generalized Stein equation from given Schur decomposition.

10.249.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This code implements a dense Lyapunov / Stein equation solver.

Lyapunov Equations:

- Standard: $AX + XA^T + Y = 0$, $A^TX + XA + Y = 0$
- Generalized: $AXE^T + EXA^T + Y = 0$, $A^TXE + E^TXA + Y = 0$

Stein Equations:

- Standard: $AXA^T - X + Y = 0$, $A^TXA - X + Y = 0$
- Generalized: $AXA^T - EXE^T + Y = 0$, $A^TXA - E^TXE + Y = 0$

with A, E, Y dense real matrices.

If (A, E) the eigenvalues lie in the left open halfplane the solution of the Lyapunov equation is symmetric positive semidefinite provided that the right hand side is symmetric negative semidefinite.

Solvers for the following Lyapunov equations are implemented too:

- Standard: $AX + XA^T + BB^T = 0$, $A^TX + XA + B^TB = 0$
- Generalized: $AXE^T + EXA^T + BB^T = 0$, $A^TXE + E^TXA + B^TB = 0$

Attention

Standard Stein Equations are not supported by GLYAP3, we use the solver for generalized Stein equations of GLYAP3.

For references see [24] and [20].

10.249.2 Macro Definition Documentation

10.249.2.1 #define GLYAP3_ISOLVE 1

10.249.2.2 #define GLYAP3_NB 64

10.249.3 Typedef Documentation

10.249.3.1 typedef struct _glyap3_st _glyap3

Attention

Internal use only.

10.249.4 Enumeration Type Documentation

10.249.4.1 enum _glyap3_eqn_t

The _glyap3_eqn_t enumeration used to represent the type of supported Lyapunov / Stein equations of `GLYAP3`.

Attention

Internal use only.

Enumerator

GLYAP3_STD_LYAP Represents a standard Lyapunov Equation $AX+XA^T+C=0$ or $A^TX+XA+C=0$

GLYAP3_GEN_LYAP Represents a generalized Lyapunov Equation $AXE^T+EXA^T+C=0$ or $A^TXE+E^TXA+C=0$

GLYAP3_STD_STEIN Represents a standard Stein Equation $AXA^T-X+C=0$ or $A^TXA-X+C=0$

GLYAP3_GEN_STEIN Represents a generalized Stein Equation $AXA^T-EXE^T+C=0$ or $A^TXA-E^TXE+C=0$

10.249.5 Function Documentation

10.249.5.1 static int glyap3_clear (void * *solver*) [static]

Parameters

in	*solver*	input pointer to the solver data

Returns

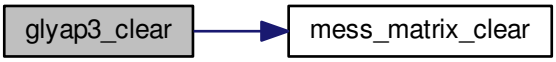
always zero

The glyap3_clear function is the clean up function.

Attention

Internal use only.

Here is the call graph for this function:



10.249.5.2 const char* glyap3_eqn_t_str (const _glyap3_eqn_t *eqn_t*)

Parameters

in	*eqn_t*	input eqn_type value

Returns

constant pointer to a human readable string containing the name of the data type

The glyap3_eqn_t_str function translates a data type constant to the corresponding string and returns a constant pointer to this string.

Attention

Internal use only.

10.249.5.3 static int glyap3_solvem (void * *data*, mess_matrix *Y*, mess_matrix *X*) [static]

Parameters

in	*data*	input pointer to the internal data structure
in	*Y*	input real symmetric right hand side
in,out	*X*	solution X

Returns

zero on success or a non-zero error value otherwise

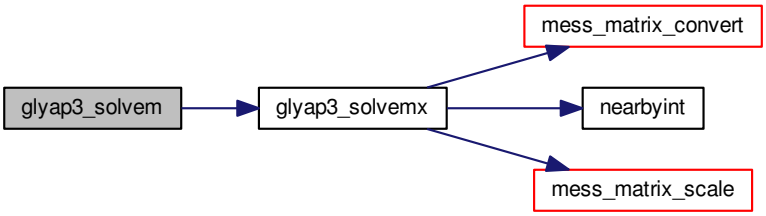
See also

glyap3_solvemx

Attention

Internal use only.

Here is the call graph for this function:



10.249.5.4 static int glyap3_solvemt (void * *data*, mess_matrix *Y*, mess_matrix *X*) [static]

Parameters

in	*data*	input pointer to the internal data structure
in	*Y*	input real symmetric right hand side
in,out	*X*	solution X

Returns

zero on success or a non-zero error value otherwise

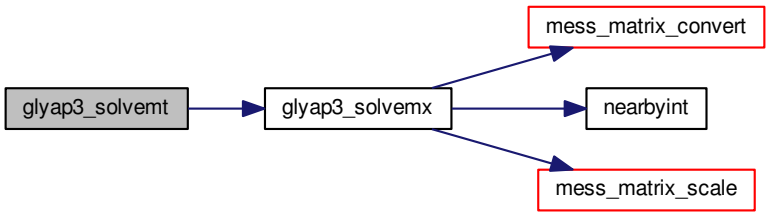
See also

glyap3_solvemx

Attention

Internal use only.

Here is the call graph for this function:



10.249.5.5 static int glyap3_solvemx (mess_operation_t *op*, void * *data*, mess_matrix *Y*, mess_matrix *X*) [static]

Parameters

in	*op*	input operation type
in	*data*	input pointer to the internal data structure
in	*Y*	input real symmetric right hand side
in,out	*X*	solution X

Returns

zero on success or a non-zero error value otherwise

The glyap3_solvemx function solves a dense Lyapunov or Stein Equation and returns the solution X. Depending on the value of _glyap3::eqn_type the following equations and op

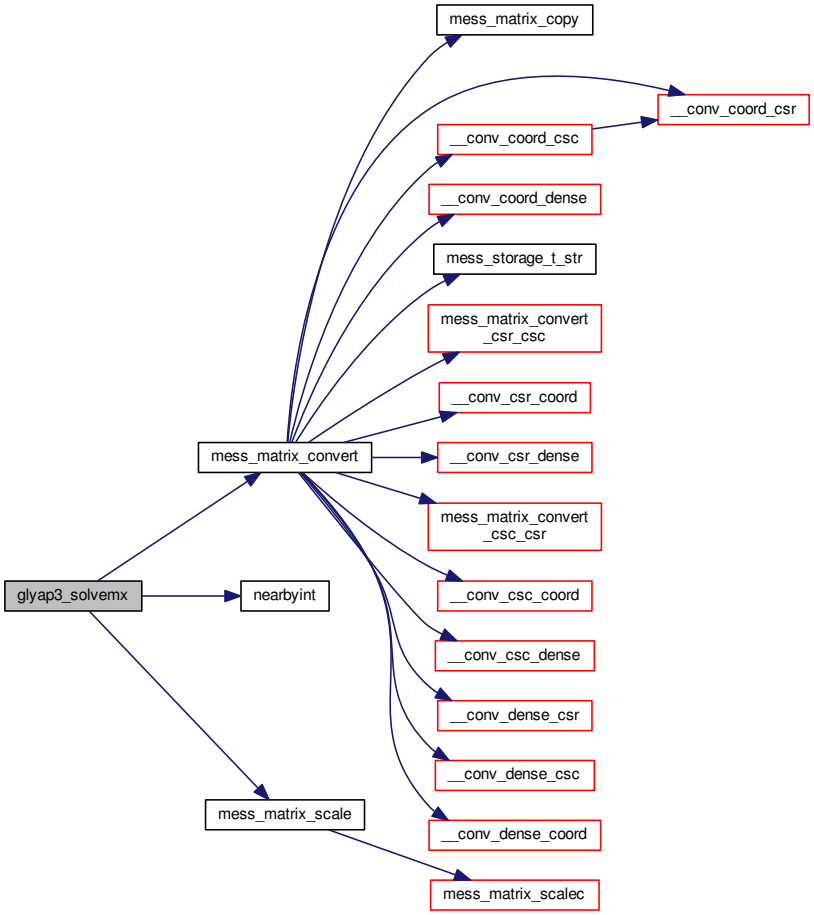
- GLYAP3_STD_LYAP and MESS_OP_NONE $AX + XA^T + Y = 0$.
- GLYAP3_STD_LYAP and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^T X + XA + Y = 0$.
- GLYAP3_GEN_LYAP and MESS_OP_NONE $AXE^T + EXA^T + Y = 0$.
- GLYAP3_GEN_LYAP and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^T XE + E^T XA + Y = 0$.

- GLYAP3_STD_STEIN and MESS_OP_NONE $AXA^T - X + Y = 0$.
- GLYAP3_STD_STEIN and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $AXA^T - X + Y = 0$.
- GLYAP3_GEN_STEIN and MESS_OP_NONE $AXA^T - EXE^T + Y = 0$.
- GLYAP3_GEN_STEIN and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^T XA - E^T XE + Y = 0$.

Attention

`dgelyp` does not support standard stein equation therefore the case GLYAP3_STD_STEIN is not supported. Internal use only.

Here is the call graph for this function:



10.249.5.6 static int glyapchol_solvem (void * *data*, mess_matrix *B*, mess_matrix *Z*) [static]

Parameters

in	*data*	input solver data
in	*B*	input real low rank factor of the right hand side
in,out	*Z*	input/output low rank factor of the solution X.

Returns

zero on success or a non-zero error value otherwise

The glyapchol_solvem function solves a dense Lyapunov Equation and returns the Cholesky factor of the solution X. Depending on `op` the following Lyapunov equations are solved:

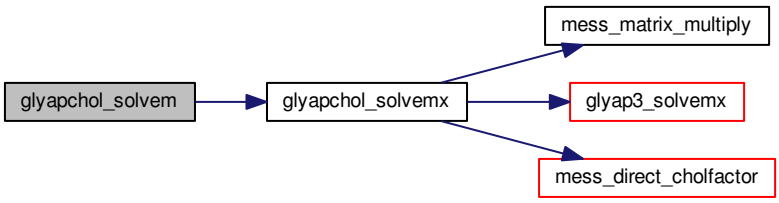
Depending on the value of _glyap3::eqn_type the following equations are solved

- GLYAP3_STD_LYAP $AX + XA^T + BB^T = 0$.
- GLYAP3_GEN_LYAP $AXE^T + EXA^T + BB^T = 0$.

Attention

Internal use only.

Here is the call graph for this function:



10.249.5.7 static int glyapchol_solvemt (void * *data*, mess_matrix *B*, mess_matrix *Z*) [static]

Parameters

in	*data*	input solver data
in	*B*	input real low rank factor of the right hand side
in,out	*Z*	input/output low rank factor of the solution X.

Returns

zero on success or a non-zero error value otherwise

The glyapchol_solvemt function solves a dense Lyapunov Equation and returns the Cholesky factor of the solution X. Depending on op the following Lyapunov equations are solved:

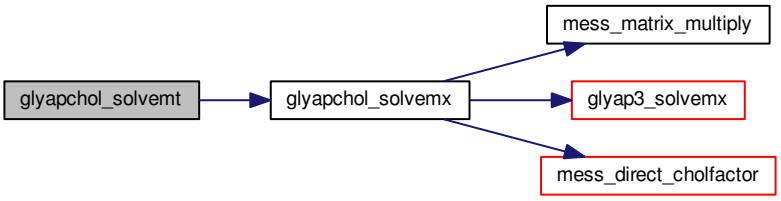
Depending on the value of _glyap3::eqn_type the following equations are solved:

- GLYAP3_STD_LYAP $A^T X + XA + B^T B = 0$.
- GLYAP3_GEN_LYAP $A^T XE + E^T XA + B^T B = 0$.

Attention

Internal use only.

Here is the call graph for this function:



10.249.5.8 static int glyapchol_solvemx (mess_operation_t *op*, void * *data*, mess_matrix *B*, mess_matrix *Z*) [static]

Parameters

in	*op*	input determine type of equation
in	*data*	input solver data
in	*B*	input real low rank factor of the right hand side
in,out	*Z*	input/output low rank factor of the solution X.

Returns

zero on success or a non-zero error value otherwise

The glyapchol_solvemx function solves a dense Lyapunov Equation and returns the Cholesky factor of the solution X. Depending on op the following Lyapunov equations are solved:

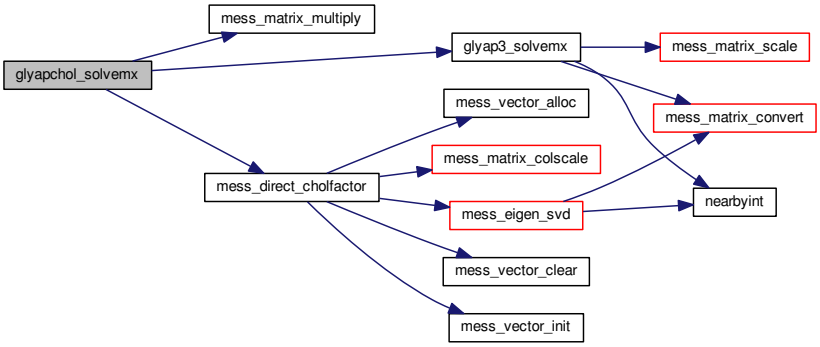
Depending on the value of _glyap3::eqn_type the and op following equations are solved

- GLYAP3_STD_LYAP and MESS_OP_NONE $AX + XA^T + BB^T = 0$.
- GLYAP3_STD_LYAP and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^TX + XA + B^TB = 0$.
- GLYAP3_GEN_LYAP and MESS_OP_NONE $AXE^T + EXA^T + BB^T = 0$.
- GLYAP3_GEN_LYAP and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^TXE + E^TXA + B^TB = 0$.

Attention

Internal use only.

Here is the call graph for this function:



10.249.5.9 int mess_direct_create_glyap3 (_glyap3_eqn_t *eqn_t*, mess_matrix *A*, mess_matrix *E*, mess_direct *solver*)

Parameters

in	*eqn↩ _t*	input _glyap3_eqn_t type of equationyyyyyy
in	*A*	input real matrix A
in	*E*	input real matrix E, (NULL if not wanted)
out	*solver*	output solver

Returns

zero on success or a non-zero error value otherwise

The mess_direct_create_glyap3 function generates a solver based on GLYAP3 for standard / generalized Lyapunov and Stein equations. The solver only wokrs correctly with symmetric right hand sides. The type of equation is determined by eqn_t. If eqn_t is GLYAP3_GEN_STEIN and GLYAP3_GEN_LYAP the matrix E must not points to NULL. All matrices have to be real and dense.

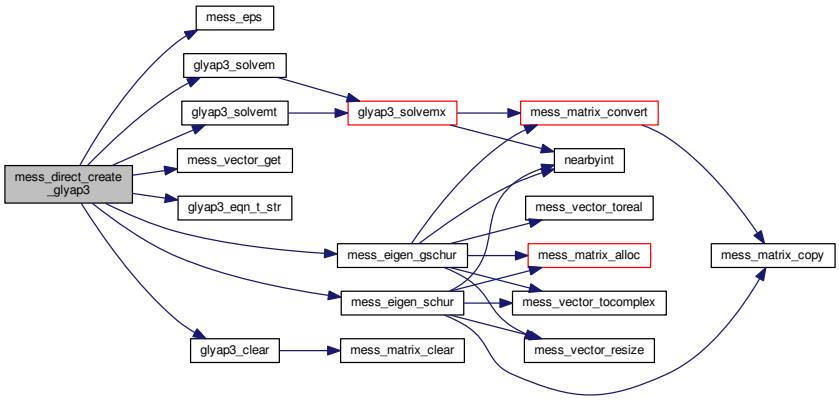
See also

mess_direct_create_generalized_lyapunov
mess_direct_create_generalized_stein

Attention

dgelyp does not support standard stein equation therefore the case GLYAP3_STD_STEIN is not supported. Internal use only.

Here is the call graph for this function:



10.249.5.10 int mess_glyap3x (_glyap3_eqn_t *eqn_t*, mess_operation_t *op*, mess_matrix *A*, mess_matrix *E*, mess_matrix *Y*, mess_matrix *Ahat*, mess_matrix *QA*, mess_matrix *Ehat*, mess_matrix *QE*, mess_matrix *X*)

Parameters

in	*eqn↩ _t*	input _glyap3_eqn_t type of equation
in	*op*	input determine which equation has to be solved
in	*A*	input real dense matrix A
in	*E*	input real dense matrix E
in	*Y*	input real dense matrix symmetric Y
out	*Ahat*	output real dense matrix Schur Form of A, (optional `NULL` if not wanted)
out	*QA*	output real dense matrix Schur transformation matrix of A, (optional `NULL` if not wanted)
out	*Ehat*	output real dense matrix Schur Form of E, (optional `NULL` if not wanted)
out	*QE*	output real dense matrix Schur transformation matrix of E, (optional `NULL` if not wanted)
out	*X*	output real dense symmetrix matrix, solution of equation X

Returns

zero on success or a non-zero error value otherwise

The mess_glyap3x function solves the standard / generalized Lyapunov or Stein equation.

Depending on `eqn_t` and `op` the following equation is solved:

- GLYAP3_STD_LYAP and MESS_OP_NONE $AX + XA^T + Y = 0$.
- GLYAP3_STD_LYAP and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^T X + XA + Y = 0$.
- GLYAP3_GEN_LYAP and MESS_OP_NONE $AXE^T + EXA^T + Y = 0$.
- GLYAP3_GEN_LYAP and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^T XE + E^T XA + Y = 0$.

- GLYAP3_STD_STEIN and MESS_OP_NONE $AXA^T - X + Y = 0$.
- GLYAP3_STD_STEIN and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $AXA^T - X + Y = 0$.
- GLYAP3_GEN_STEIN and MESS_OP_NONE $AXA^T - EXE^T + Y = 0$.
- GLYAP3_GEN_STEIN and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^TXA - E^TXE + Y = 0$.

The right hand side matrix `Y` must be real and symmetric.

In order to solve the equations a (generalized) Schur decomposition of A / (A, E) is computed internally.

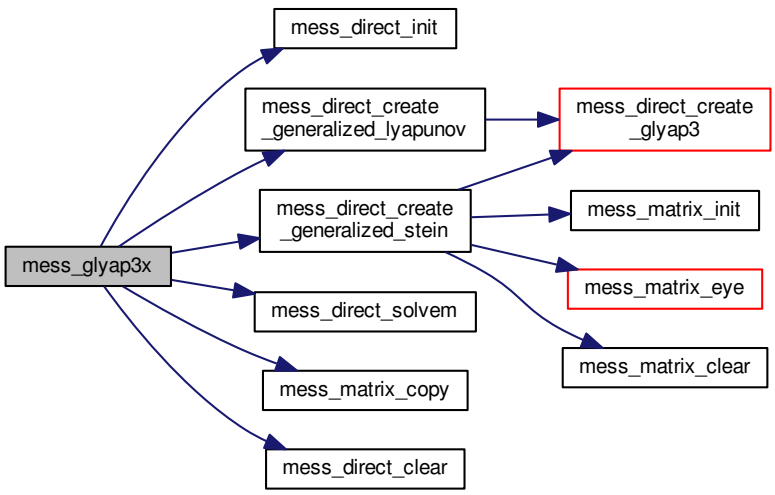
If a standard Lyapunov or Stein equation is solved then the real Schur decomposition fullfills: $A = Q_A \hat{A} {Q_A}^H$. If a generalized Lyapunov or Stein equation is solved then the real Schur decomposition fullfills: $A = Q_A \hat{A} {Q_E}^H$ and $E = Q_A \hat{E} {Q_E}^H$.

If `Ahat`, `QA`, `Ehat` or `QE` is not `NULL` then the corresponding matrix of the Schur decomposition is written to that argument.

Attention

Internal use only.

Here is the call graph for this function:



10.249.5.11 int mess_tglyap3x (_glyap3_eqn_t *eqn_t*, mess_operation_t *op*, mess_matrix *Ahat*, mess_matrix *QA*, mess_matrix *Ehat*, mess_matrix *QE*, mess_matrix *Y*, mess_matrix *X*)

Parameters

in	*eqn_t*	input _glyap3_eqn_t type of equation
in	*op*	input determine which equation has to be solved

Parameters

in	*Ahat*	input real dense matrix Schur Form of A
in	*QA*	input real dense matrix Schur transformation matrix of A
in	*Ehat*	input real dense matrix Schur Form of E, (only necessary for generalized equation)
in	*QE*	input real dense matrix Schur transformation matrix of E, (only necessary for generalized equation)
in	*Y*	input real dense matrix symmetric Y
out	*X*	output real dense symmetrix matrix, solution of equation X

Returns

zero on success or a non-zero error value otherwise

The mess_tglyap3x function solves the standard / generalized Lyapunov or Stein equation.

Depending on `eqn_t` and `op` the following equation is solved:

- GLYAP3_STD_LYAP and MESS_OP_NONE $AX + XA^T + Y = 0$.
- GLYAP3_STD_LYAP and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^T X + XA + Y = 0$.
- GLYAP3_GEN_LYAP and MESS_OP_NONE $AXE^T + EXA^T + Y = 0$.
- GLYAP3_GEN_LYAP and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^T XE + E^T XA + Y = 0$.

- GLYAP3_STD_STEIN and MESS_OP_NONE $AXA^T - X + Y = 0$.
- GLYAP3_STD_STEIN and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $AXA^T - X + Y = 0$.
- GLYAP3_GEN_STEIN and MESS_OP_NONE $AXA^T - EXE^T + Y = 0$.
- GLYAP3_GEN_STEIN and MESS_OP_TRANSPOSE or MESS_OP_HERMITIAN $A^T XA - E^T XE + Y = 0$.

The right hand side matrix `Y` must be real and symmetric.

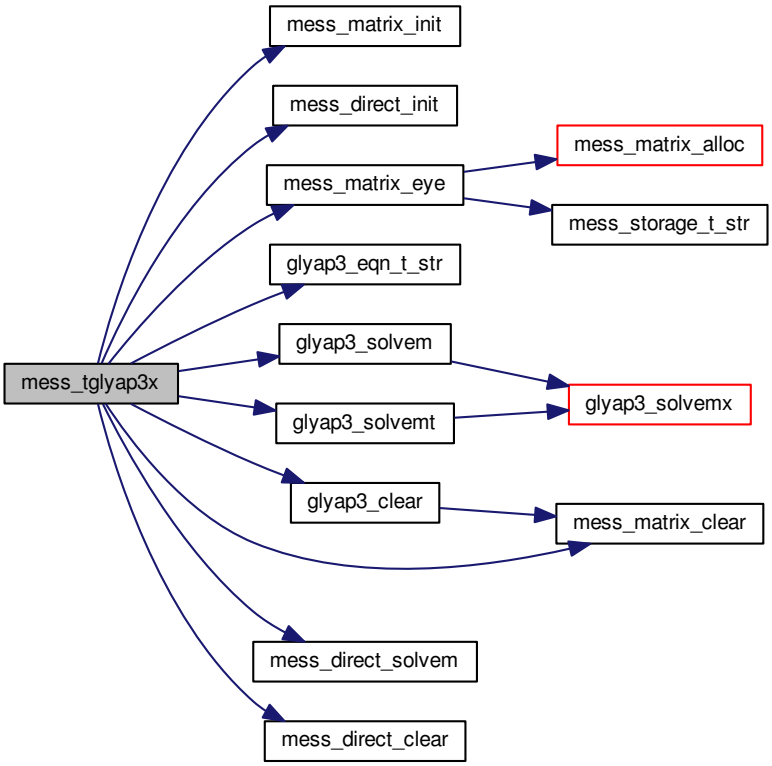
In order to solve the equations a (generalized) Schur decomposition of A / (A, E) is requiered.

If a standard Lyapunov or Stein equation is solved then the real Schur decomposition must fullfill: $A = Q_A \hat{A} {Q_A}^H$. If a generalized Lyapunov or Stein equation is solved then the real Schur decomposition must fullfill: $A = Q_A \hat{A} {Q_E}^H$ and $E = Q_A \hat{E} {Q_E}^H$.

Attention

Internal use only.

Here is the call graph for this function:



10.250 glyap_test.c File Reference

Functions

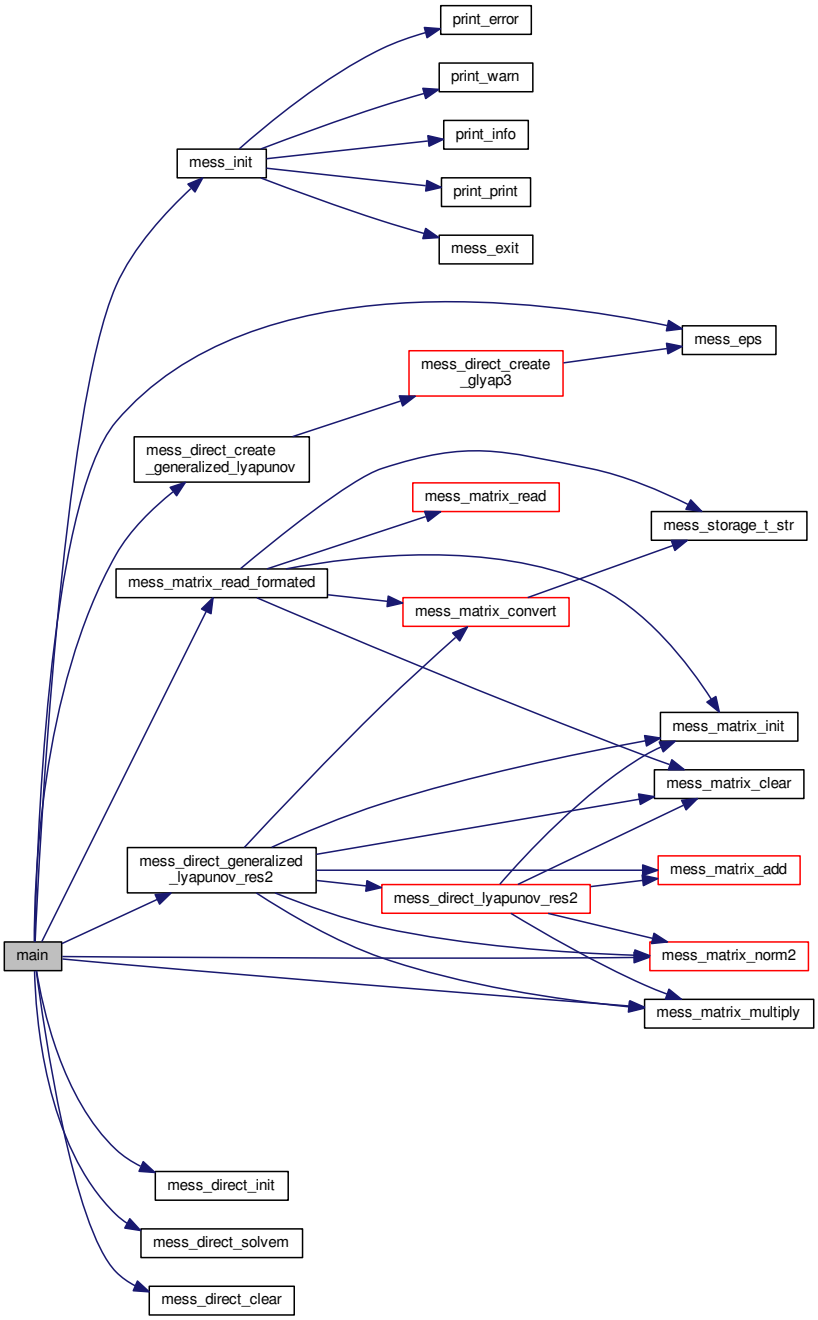
- int main (int argc, char **argv)

10.250.1 Detailed Description

10.250.2 Function Documentation

10.250.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.251 gmres.c File Reference

Functions

- int mess_solver_gmres (mess_mvpcall matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

Solve a system of linear equations with (preconditioned) generalized minimal residual algorithm.

- int mess_solver_gmres_restart (mess_mvpcall matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with restarted GMRES.

10.251.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.252 gslrsrm.c File Reference

Functions

- int mess_bt_gslrsrm (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Generalized low-rank square root balanced truncation for first order GLTI to LTI systems.

10.252.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.253 gstein.c File Reference

Functions

- PyObject * pymess_gstein (PyObject *self, PyObject *args)

  Interface to mess_gstein for use with Python.

10.253.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.254 gstein_test.c File Reference

Functions

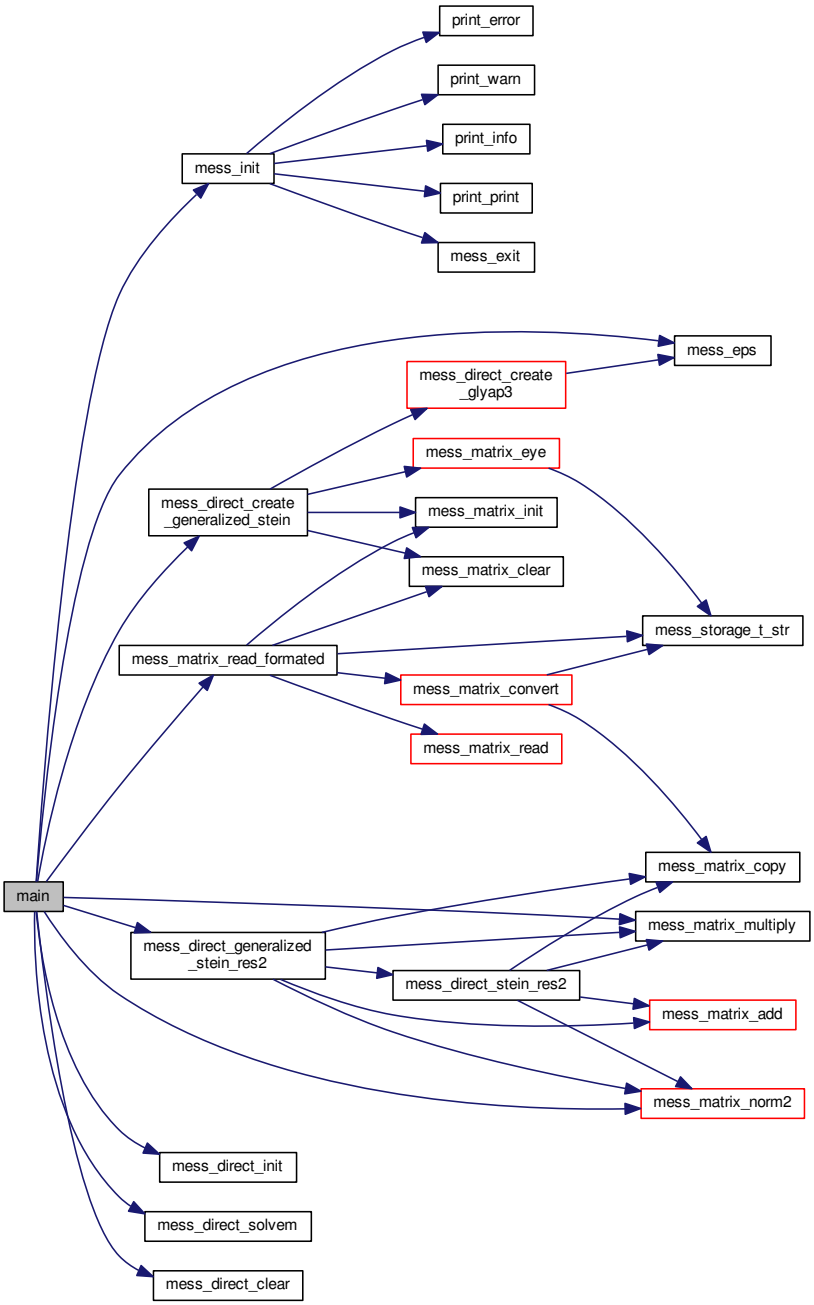
- int main (int argc, char **argv)

10.254.1 Detailed Description

10.254.2 Function Documentation

10.254.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:



10.255 h2.h File Reference

Data Structures

- struct mess_h2_options_st

  Structure to store $\mathcal{H}_2$ options.
- struct mess_h2_status_st

  Structure to store $\mathcal{H}_2$ status.
- struct mess_bt_options_st

  Options structure for balanced truncation .
- struct mess_bt_status_st

  Status structure for balanced truncation.

Macros

- #define MESS_H2NORM_NO 0
- #define MESS_H2NORM_FULL 1
- #define MESS_H2NORM_UPDATE 2
- #define MESS_BT_POSITION 0x01
- #define MESS_BT_VELOCITY 0x02
- #define MESS_BT_POSITION_VELOCITY 0x03
- #define MESS_BT_VELOCITY_POSITION 0x04

Typedefs

- typedef struct mess_h2_options_st ∗ mess_h2_options

  Type definition for the $\mathcal{H}_2$ options.
- typedef struct mess_h2_status_st ∗ mess_h2_status

  Type definition for the $\mathcal{H}_2$ status.
- typedef int(∗ mess_bt_chooseorder_func) (mess_vector, double ∗, mess_int_t ∗, mess_int_t, void ∗)

  Type definition for the choose order function.
- typedef struct mess_bt_options_st ∗ mess_bt_options

  Type definition for the balanced truncation options.
- typedef struct mess_bt_status_st ∗ mess_bt_status

  Type definition for the balanced truncation status.

Functions

- int mess_h2_norm_internal (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix E, double ∗norm)

  Compute the $\mathcal{H}_2$-norm of a LTI System given as matrices.
- int mess_h2_norm (mess_dynsys LTI, double ∗norm)

  Compute the $\mathcal{H}_2$-norm of a LTI system.
- int mess_h2_error_internal (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix E, mess_matrix Ar, mess_matrix Br, mess_matrix Cr, mess_matrix Er, double ∗norm)

  Compute the $\mathcal{H}_2$ error of two LTI systems.
- int mess_h2_error (mess_dynsys ltiA, mess_dynsys ltiB, double ∗err)

  Compute the $\mathcal{H}_2$-error between two LTI systems.
- int mess_h2_irka_init (mess_matrix A, mess_direct Asolver, mess_matrix E, mess_int_t ∗r0, mess_int_t kp, mess_int_t km, mess_vector sigma)

  Initialize parameters for IRKA.

- int mess_h2_irka_biorth (mess_dynsys orig, mess_vector sigma, mess_h2_options opt, mess_dynsys reduced, mess_matrix V, mess_matrix W, mess_h2_status status)

  Compute a $\mathcal{H}_2$ reduced model via the IRKA algorithm (biorthonormal version).
- int mess_h2_tsia (mess_dynsys orig, mess_vector sigmaEX, mess_h2_options opt, mess_dynsys redu, mess_matrix V, mess_matrix W, mess_h2_status status)

  Compute a $\mathcal{H}_2$ reduced model via the TSIA algorithm.
- int mess_h2_tsiag (mess_dynsys orig, mess_vector sigmaEX, mess_h2_options opt, mess_dynsys redu, mess_matrix V, mess_matrix W, mess_h2_status status)

  Compute a $\mathcal{H}_2$ reduced model via the TSIA algorithm.
- int mess_h2_options_init (mess_h2_options ∗opt)

  Initialize a mess_h2_options object.
- int mess_h2_options_clear (mess_h2_options ∗opt)

  Clean up a mess_h2_options object.
- int mess_h2_options_print (mess_h2_options opt)

  Print a mess_h2_options object to screen.
- int mess_h2_status_init (mess_h2_status ∗status)

  Initialize a mess_h2_status object.
- int mess_h2_status_clear (mess_h2_status ∗status)

  Clean up a mess_h2_status object.
- int mess_h2_status_print (mess_h2_status status)

  Print a mess_h2_status object.
- int mess_h2_status_printfull (mess_h2_status status)

  Print a mess_h2_status object (full output).
- int mess_h2_status_write_mfile (mess_h2_status status, const char ∗filename)

  Write a mfile to plot in MATLAB .
- int mess_bt_options_init (mess_bt_options ∗opt)

  Initialize a mess_bt_options object.
- int mess_bt_options_clear (mess_bt_options ∗opt)

  Clear a mess_bt_options object.
- int mess_bt_options_print (mess_bt_options opt)

  Print a mess_bt_options object to stdout.
- const char ∗ mess_bt_getsotypestr (unsigned short so_type)

  Get the string of the so_type value.
- int mess_bt_status_init (mess_bt_status ∗status)

  Initialize a mess_bt_status object.
- int mess_bt_status_clear (mess_bt_status ∗status)

  Clear a mess_bt_status object.
- int mess_bt_status_print (mess_bt_status status)

  Print a mess_bt_status object to stdout.
- int mess_bt_chooseorder_default (mess_vector SIGMA, double ∗tol, mess_int_t ∗maxr, mess_int_t FOM, void ∗aux)

  Choose dimension of reduced order model adaptively for balanced truncation (BT).
- int mess_bt_chooseorder_minreal (mess_vector SIGMA, double ∗tol, mess_int_t ∗maxr, mess_int_t FOM, void ∗aux)

  Choose dimension of reduced order model adaptively for balanced truncation (BT) with cut of of nearly zero singular values.
- int mess_bt_lrsrm (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Low-rank square root balanced truncation for first order LTI systems.
- int mess_bt_gslrsrm (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Generalized low-rank square root balanced truncation for first order GLTI to LTI systems.

- int mess_bt_gglrsrm (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Generalized low-rank square root balanced truncation for first order GLTI to GLTI systems.
- int mess_bt_lrsrm_somor (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Low-rank square root balanced truncation for second order LTI systems to first order LTI systems.
- int mess_bt_lrsrm_somor_so (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Low-rank square root balanced truncation for second order LTI systems to second order LTI systems.
- int mess_bt_lrsrm_somor_so_gram (mess_dynsys sys, mess_matrix ZB, mess_matrix ZC, mess_bt_options btopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Low-rank square root balanced truncation for second order LTI systems to second order LTI systems (Gramian version).

10.255.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.256 h2_norm.c File Reference

Macros

- #define INFINITY 1.0/0.0

Functions

- int mess_h2_norm_internal (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix E, double *norm)

  Compute the $\mathcal{H}_2$-norm of a LTI System given as matrices.
- int mess_h2_norm (mess_dynsys lti, double *norm)

  Compute the $\mathcal{H}_2$-norm of a LTI system.

10.256.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.256.2 Macro Definition Documentation

10.256.2.1 #define INFINITY 1.0/0.0

10.257 h2error.c File Reference

Macros

- #define INFINITY 1.0/0.0

Functions

- int mess_h2_error_internal (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix E, mess_matrix Ar, mess_matrix Br, mess_matrix Cr, mess_matrix Er, double *norm)

  Compute the $\mathcal{H}_2$ error of two LTI systems.
- int mess_h2_error (mess_dynsys ltiA, mess_dynsys ltiB, double *err)

  Compute the $\mathcal{H}_2$-error between two LTI systems.

10.257.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.257.2 Macro Definition Documentation

10.257.2.1 #define INFINITY 1.0/0.0

10.258 hessenberg_lu.c File Reference

Data Structures

- struct hessenberg_solver

  Internal structure for the LAPACK based solver.

Typedefs

- typedef struct hessenberg_solver hessenberg_solver

  Internal structure for the LAPACK based solver.

Functions

- static int hessenberg_clear (void *data)

  Clear the internal structure hessenberg_solver.
- static int hessenberg_solvex (void *data, mess_operation_t op, mess_int_t nrhs, double *rhs, mess_double_cpx_t *rhs_cpx, mess_int_t ld_rhs)

  Solve a linear system using the hessenberg_solver.
- static int hessenberg_solve_all (void *data, mess_operation_t op, mess_vector b, mess_vector x)

  Solve a linear system using the hessenberg_solver.
- static int hessenberg_solvem_all (void *data, mess_operation_t op, mess_matrix b, mess_matrix x)

  Solve a linear system using the hessenberg_solver.
- static int hessenberg_solve (void *data, mess_vector b, mess_vector x)
- static int hessenberg_solvet (void *data, mess_vector b, mess_vector x)
- static int hessenberg_solveh (void *data, mess_vector b, mess_vector x)
- static int hessenberg_solvem (void *data, mess_matrix b, mess_matrix x)
- static int hessenberg_solvemt (void *data, mess_matrix b, mess_matrix x)
- static int hessenberg_solvemh (void *data, mess_matrix b, mess_matrix x)
- int mess_direct_create_hessenberg_lu (mess_matrix matrix, mess_direct solver)

  Generate a dense LAPACK based LU-solver for upper hessenberg matrices.

10.258.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.258.2 Typedef Documentation

10.258.2.1 typedef struct hessenberg_solver hessenberg_solver

Attention

Internal use only.

10.258.3 Function Documentation

10.258.3.1 static int hessenberg_clear (void * *data*) [static]

Parameters

in,out	*data*	pointer to the internal data

Returns

always zero

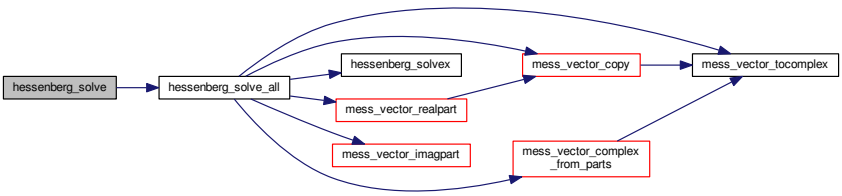
The hessenberg_clear function clears a LAPACK solver.

Attention

Internal use only.

10.258.3.2 static int hessenberg_solve (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Here is the call graph for this function:



10.258.3.3 static int hessenberg_solve_all (void * *data*, mess_operation_t *op*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input pointer to the internal data
in	*op*	input determine which equation has to be solved
in	*b*	input right hand side
out	*x*	output the solution

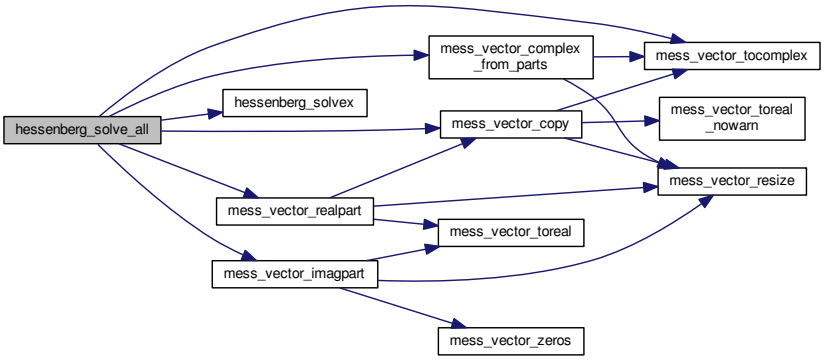
Returns

zero on success or a non-zero error value otherwise

Attention

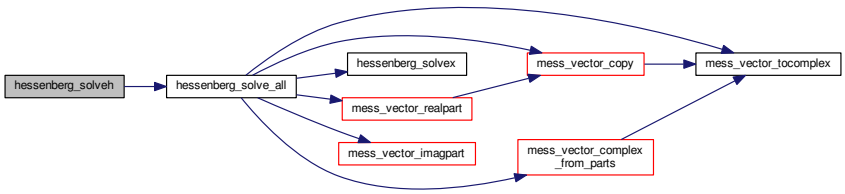
Internal use only.

Here is the call graph for this function:



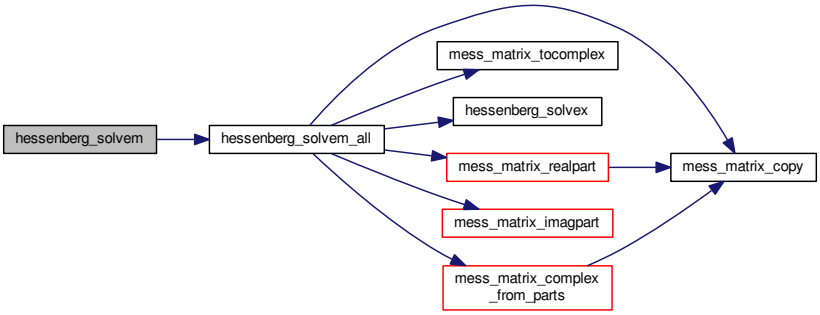
10.258.3.4 static int hessenberg_solveh (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Here is the call graph for this function:



10.258.3.5 static int hessenberg_solvem (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Here is the call graph for this function:



10.258.3.6 static int hessenberg_solvem_all (void * *data*, mess_operation_t *op*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input pointer to the internal data
in	*op*	input determine which equation has to be solved
in	*b*	input right hand side
out	*x*	output the solution

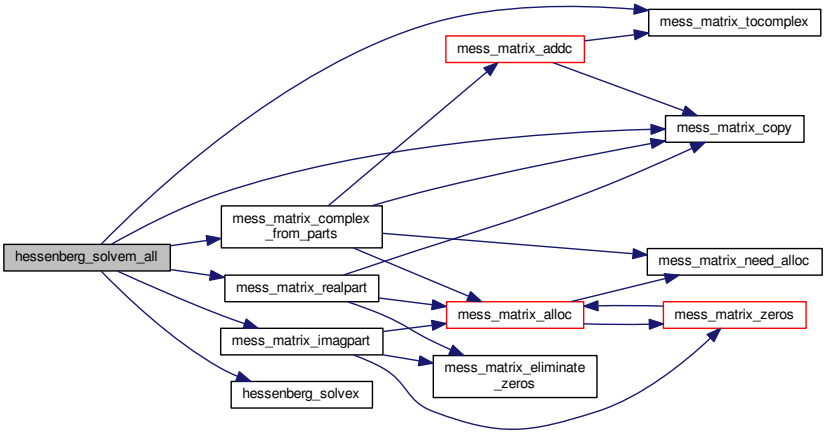
Returns

zero on success or a non-zero error value otherwise

Attention

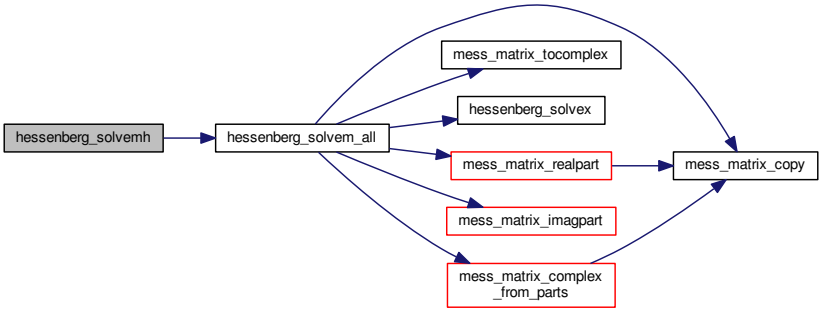
Internal use only.

Here is the call graph for this function:



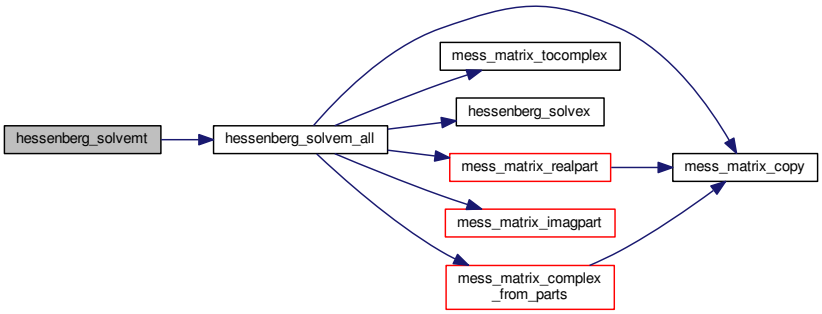
10.258.3.7 static int hessenberg_solvemh (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Here is the call graph for this function:



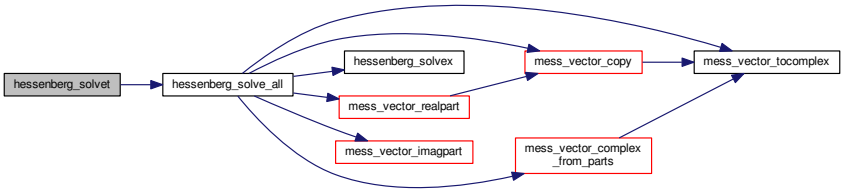
10.258.3.8 static int hessenberg_solvemt (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Here is the call graph for this function:



10.258.3.9 static int hessenberg_solvet (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Here is the call graph for this function:



10.258.3.10 static int hessenberg_solvex (void * *data*, mess_operation_t *op*, mess_int_t *nrhs*, double * *rhs*, mess_double_cpx_t * *rhs_cpx*, mess_int_t *ld_rhs*) [static]

Parameters

in	*data*	input pointer to the internal data
in	*op*	input determine which equation has to be solved
in	*nrhs*	input number of right hand sides
in,out	*rhs*	input/output points to the real right hand side
in,out	*rhs_cpx*	input/output points to the complex right hand side
in	*ld_rhs*	input leading dimension of the right hand side array

Returns

zero on success or a non-zero error value otherwise

The hessenberg_solvex function solves a linear system $Ax = b$ using the hessenberg_solver. The right hand side data array is `rhs` or `rhs_cpx`. If the hessenberg_solver is MESS_REAL `rhs` must be given and `rhs_cpx` must points to `NULL` or vice versa. The solution is at exit written to `rhs` / `rhs_cpx` respectively.

Attention

Internal use only.

10.259 ilu0.c File Reference

Data Structures

- struct __ilu0_data

  Internal structure for $ILU(0)$ preconditioner. Internal structure for $ILU(0)$ preconditioner.

Typedefs

- typedef struct __ilu0_data ilu0_data

  Internal structure for $ILU(0)$ preconditioner. Internal structure for $ILU(0)$ preconditioner.

Functions

- static int __mess_precond_ilu0_solve (mess_precond ilu, mess_solver_options opt, mess_vector in, mess_vector out)

  Solve with $ILU(0)$ preconditioner.
- static int __mess_precond_ilu0_clear (mess_precond ilu)

  Clear function for ILU(0).
- int mess_precond_ilu0 (mess_precond pre, mess_matrix matrix)

  Compute an $ILU(0)$ preconditioner.

10.259.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.259.2 Typedef Documentation

10.259.2.1 typedef struct __ilu0_data ilu0_data

Attention

Interal use only.

10.259.3 Function Documentation

10.259.3.1 static int __mess_precond_ilu0_clear (mess_precond *ilu*) [static]

Parameters

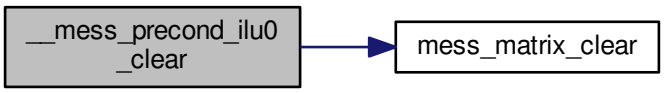
in,out	*ilu*	mess_precond object

Returns

always zero

The __mess_precond_ilu0_clear function clears the $ILU(0)$ preconditioner.

Here is the call graph for this function:



10.259.3.2 static int __mess_precond_ilu0_solve (mess_precond *ilu*, mess_solver_options *opt*, mess_vector *in*, mess_vector *out*) [static]

Parameters

in	*ilu*	input ilu factorized matrix
in	*opt*	input option from the iterative solver (unused)
out	*out*	output solution of the equation
in	*in*	input right hand side

Returns

zero on success or a non-zero error value otherwise

The __mess_precond_ilu0_solve function solves

$$LUx = b$$

with forward and backward substitution.

Attention

Interal use only.

10.260 iluk.c File Reference

Data Structures

- struct CCSRFmt
- struct __iluk

Macros

- #define min(a, b) (((a)>(b))?(b):(a))
- #define max(a, b) (((a)>(b))?(a):(b))

Typedefs

- typedef struct CCSRFmt CCSRMat
- typedef struct CCSRFmt * ccsrptr
- typedef struct __iluk ILUK
- typedef struct __iluk * ilukptr

Functions

- static int symbolic_iluk (mess_int_t lofM, mess_matrix mat, ilukptr ilu)
- static int __iluk_solve_complex (mess_precond myself, mess_solver_options opt, mess_vector in, mess_vector out)
- static int __iluk_solve (mess_precond myself, mess_solver_options opt, mess_vector in, mess_vector out)
- int clean_ccsrmat (ccsrptr mat)

  Clear a matrix.
- static int __iluk_clear (mess_precond myself)
- int mess_precond_iluk (mess_precond pre, mess_matrix mat, mess_int_t lof)

  Setup an $ILU(k)$ preconditioner from a matrix.

10.260.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.260.2 Macro Definition Documentation

10.260.2.1 #define max(a, b) (((a)>(b))?(a):(b))

10.260.2.2 #define min(a, b) (((a)>(b))?(b):(a))

10.260.3 Typedef Documentation

10.260.3.1 typedef struct CCSRFmt CCSRMat

10.260.3.2 typedef struct CCSRFmt * ccsrptr

10.260.3.3 typedef struct __iluk ILUK

10.260.3.4 typedef struct __iluk * ilukptr

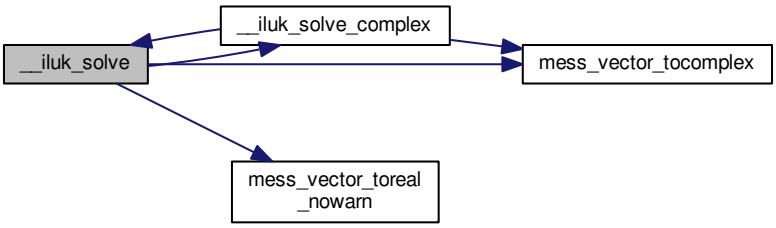
10.260.4 Function Documentation

10.260.4.1 static int __iluk_clear (mess_precond *myself*) [static]

Here is the call graph for this function:

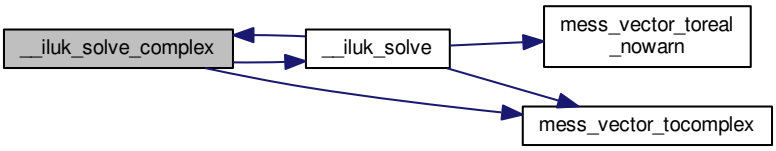
10.260.4.2 static int __iluk_solve (mess_precond *myself*, mess_solver_options *opt*, mess_vector *in*, mess_vector *out*) [static]

Here is the call graph for this function:



10.260.4.3 static int __iluk_solve_complex (mess_precond *myself,* mess_solver_options *opt,* mess_vector *in,* mess_vector *out*) [static]

Here is the call graph for this function:



10.260.4.4 int clean_ccsrmat (ccsrptr *mat*)

Parameters

in,out	*mat*	matrix to clear

Returns

always zero

The clean_ccsrmat functions cleans up a matrix mat.

10.260.4.5 static int symbolic_iluk (mess_int_t *lofM,* mess_matrix *mat,* ilukptr *ilu*) [static]

10.261 init.c File Reference

Functions

- int mess_matrix_clear (mess_matrix *matrix)

  Remove a mess_matrix object from memory.
- int mess_matrix_init (mess_matrix *matrix)

  Initialize the mess_matrix structure.
- int mess_matrix_copy (mess_matrix in, mess_matrix out)

  Copy a matrix to another one.
- mess_int_t mess_matrix_memsize (mess_matrix matrix)

  Return the size of a matrix in bytes.
- mess_int_t mess_matrix_memsize_nnz (mess_int_t rows, mess_int_t cols, mess_int_t nnz, mess_int_t type, mess_int_t dtype)

  Determine memory size of a matrix from parameters.
- int mess_matrix_tocomplex (mess_matrix m)

  Change the data type of a matrix to complex.
- int mess_matrix_toreal (mess_matrix m)

  Change the data type of a matrix to real.
- int mess_matrix_totype (mess_matrix mat, mess_datatype_t dt)

  Convert a matrix to a specified data type.

10.261.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.262 init.c File Reference

Functions

- static void print_error (const char *str)
- static void print_warn (const char *str)
- static void print_info (const char *str)
- static void print_print (const char *str)
- int mess_init ()

  *Initialize **M.E.S.S.**.*
- int mess_exit ()

  *Clean up some initial **M.E.S.S.** structures.*

Variables

- static int init_done = 0
- static int exit_done = 0
- int mess_error_level = LOGLEVEL

10.262.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.262.2 Function Documentation

10.262.2.1 static void print_error (const char * *str*) [static]

10.262.2.2 static void print_info (const char * *str*) [static]

10.262.2.3 static void print_print (const char * *str*) [static]

10.262.2.4 static void print_warn (const char * *str*) [static]

10.262.3 Variable Documentation

10.262.3.1 int exit_done = 0 [static]

10.262.3.2 int init_done = 0 [static]

10.263 init.c File Reference

Functions

- int mess_vector_init (mess_vector *v)

  Initialize the mess_vector structure.

10.263.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.264 INSTALL.md File Reference

10.265 interface_cholmod.h File Reference

Functions

- int mess_matrix_convert_dense_to_cholmod (mess_matrix A, cholmod_dense **A_cholmod, cholmod_common *c)
- int mess_matrix_convert_cholmod_to_dense (cholmod_dense *A_cholmod, mess_matrix A, cholmod_common *c)
- int mess_matrix_convert_csc_to_cholmod (mess_matrix A, cholmod_sparse **A_cholmod, cholmod_common *c)
- int mess_matrix_convert_cholmod_to_csc (cholmod_sparse *A_cholmod, mess_matrix A, cholmod_common *c)
- int mess_vector_convert_dense_to_cholmod (mess_vector v, cholmod_dense **v_cholmod, cholmod_common *c)
- int mess_vector_convert_cholmod_to_dense (cholmod_dense *v_cholmod, mess_vector v, cholmod_common *c)

10.265.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.266 interface_csparse.h File Reference

Functions

- static int mess_matrix_from_csparse_di (const cs_di *input, mess_matrix output)

  Copy a real matrix from CSparse *to* ***M.E.S.S.*** *(double-integer variant).*
- static int mess_matrix_from_csparse (const cs *input, mess_matrix output)

  Default wrapper for mess_matrix_from_csparse_di.
- static int mess_matrix_from_csparse_dl (const cs_dl *input, mess_matrix output)

  Copy a real matrix from CSparse *to* ***M.E.S.S.*** *(double-long variant).*
- static int mess_matrix_from_csparse_ci (const cs_ci *input, mess_matrix output)

  Copy a complex matrix from CSparse *to* ***M.E.S.S.*** *(complex-int variant).*
- static int mess_matrix_from_csparse_cl (const cs_cl *input, mess_matrix output)

  Copy a complex matrix from CSparse *to* ***M.E.S.S.*** *(complex-long variant).*
- static int mess_matrix_to_csparse_di (mess_matrix input, cs_di **o)

  Copy a real matrix from ***M.E.S.S.*** *to* CSparse *(double-int variant).*
- static int mess_matrix_to_csparse (mess_matrix input, cs **o)

  Wrapper around mess_matrix_to_csparse_di.
- static int mess_matrix_to_csparse_dl (mess_matrix input, cs_dl **o)

  Copy a real matrix from ***M.E.S.S.*** *to* CSparse *(double-long variant).*
- static int mess_matrix_to_csparse_ci (mess_matrix input, cs_ci **o)

  Copy a complex matrix from ***M.E.S.S.*** *to* CSparse *(complex-int variant).*
- static int mess_matrix_to_csparse_cl (mess_matrix input, cs_cl **o)

  Copy a complex matrix from ***M.E.S.S.*** *to* CSparse *(complex-long variant).*

10.267 interface_matlab.c File Reference

Functions

- void my_mexPrintf (const char ∗str)
- void my_mexError (const char ∗str)
- void my_mexWarn (const char ∗str)
- static void ∗ matlab_malloc (size_t len)
- static void ∗ matlab_realloc (void ∗ptr, size_t len)
- static void ∗ matlab_calloc (size_t nmeb, size_t smeb)
- static void matlab_free (void ∗ptr)
- void init_mexmess ()

  Initialize the ***M.E.S.S.*** *to be used in a MeX function.*
- double mess_double_from_mexmess (const mxArray ∗Amatlab)

  Get a real valued scalar from MATLAB.
- mess_double_cpx_t mess_complex_from_mexmess (const mxArray ∗Amatlab)

  Get a complex valued scalar from MATLAB.
- int mess_int_array_to_matlab (mess_int_t ∗p, mess_int_t len, mxArray ∗∗Amat)
- mxArray ∗ mxCreateComplexScalar (mess_double_cpx_t s)

Variables

- csc_error_print_t error_handle
- csc_error_print_t warn_handle
- csc_error_print_t info_handle
- csc_error_print_t print_handle

10.267.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.267.2 Function Documentation

10.267.2.1 static void∗ matlab_calloc (size_t *nmeb*, size_t *smeb*) [static]

10.267.2.2 static void matlab_free (void ∗ *ptr*) [static]

10.267.2.3 static void∗ matlab_malloc (size_t *len*) [static]

10.267.2.4 static void∗ matlab_realloc (void ∗ *ptr*, size_t *len*) [static]

10.267.2.5 int mess_int_array_to_matlab (mess_int_t ∗ *p*, mess_int_t *len*, mxArray ∗∗ *Amat*)

10.267.2.6 void my_mexError (const char ∗ *str*)

10.267.2.7 void my_mexPrintf (const char ∗ *str*)

10.267.2.8 void my_mexWarn (const char * *str*)

10.267.3 Variable Documentation

10.267.3.1 csc_error_print_t error_handle

10.267.3.2 csc_error_print_t info_handle

10.267.3.3 csc_error_print_t print_handle

10.267.3.4 csc_error_print_t warn_handle

10.268 interface_matlab.h File Reference

Macros

- #define MESS64
- #define MESS_MATLAB
- #define FUNCTION_FAILURE_HANDLE_MEX(err, cond, fun) if ((cond)) { char text[1024]; snprintf(text,1023," %s returned with %d - %s\n", #fun, err, mess_get_error(err)); mexErrMsgTxt(text); }

  Function failure handler for `MATLAB`.
- #define mess_check_struct(X, str) if (!mxIsStruct(X)) { csc_error_message(str); return MESS_ERROR_A↩RGUMENTS; }

  Check if the given mxArray is a structure.

Functions

- char * mess_classname_from_mexmess (const mxArray *instance)

  Get a class name from `MATLAB` *object.*
- double mess_double_from_mexmess (const mxArray *Amatlab)

  Get a real valued scalar from `MATLAB`.
- int mxIsSubClass (const mxArray *instance, char *class)
- mxArray * mxCreateComplexScalar (mess_double_cpx_t s)
- mess_double_cpx_t mess_complex_from_mexmess (const mxArray *Amatlab)

  Get a complex valued scalar from `MATLAB`.
- void init_mexmess ()

  Initialize the ***`M.E.S.S.`*** *to be used in a MeX function.*
- mess_direct_cholpackage_t mess_direct_cholpackage_t_from_mexmess (const mxArray *instance)
- mess_direct_lupackage_t mess_direct_lupackage_t_from_mexmess (const mxArray *instance)
- mess_equation_t mess_equation_t_from_mexmess (const mxArray *instance)
- mess_memusage_t mess_memusage_t_from_mexmess (const mxArray *instance)
- mess_multidirect_t mess_multidirect_t_from_mexmess (const mxArray *instance)
- mess_operation_t mess_operation_t_from_mexmess (const mxArray *instance)
- mess_parameter_t mess_parameter_t_from_mexmess (const mxArray *instance)
- mess_residual_t mess_residual_t_from_mexmess (const mxArray *instance)
- mess_norm_t mess_norm_t_from_mexmess (const mxArray *instance)
- mxArray * mess_direct_cholpackage_t_to_mexmess (mess_direct_cholpackage_t chol_t)
- mxArray * mess_direct_lupackage_t_to_mexmess (mess_direct_lupackage_t chol_t)
- mxArray * mess_equation_t_to_mexmess (mess_equation_t eqn_t)
- mxArray * mess_memusage_t_to_mexmess (mess_memusage_t mem)
- mxArray * mess_multidirect_t_to_mexmess (mess_multidirect_t mdirect_t)

- mxArray * mess_operation_t_to_mexmess (mess_operation_t op)
- mxArray * mess_parameter_t_to_mexmess (mess_parameter_t para)
- mxArray * mess_residual_t_to_mexmess (mess_residual_t res_t)
- mxArray * mess_norm_t_to_mexmess (mess_norm_t nrm_t)
- mess_options mess_options_from_mexmess (const mxArray *instance)
- mxArray * mess_options_to_mexmess (mess_options opt)
- mxArray * mess_status_to_mexmess (mess_status stat)
- int mess_callback_equation_mexmess (mess_equation eqn, mess_options opt, const mxArray *meqn)
- mess_equation eqn_conv_lyap (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_lyap_dae1 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_lyap_dae2 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_lyap_so1 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_lyap_so2 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric_dae1 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric_dae2 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric_so1 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric_so2 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation mess_equation_from_mexmess (mxArray *m_eqn, mxArray *m_opt, mess_freelist mem, mess_equation_t eqn_type)
- mess_matrix mess_matrix_from_mexmess (const mxArray *Amatlab)

  Copy a matrix from MATLAB *to a mess_matrix.*
- mxArray * mess_matrix_to_mexmess (mess_matrix A)

  Copy a matrix from ***M.E.S.S.*** *to* MATLAB.
- mess_vector mess_vector_from_mexmess (const mxArray *Amatlab)

  Copy a vector/matrix from MATLAB *to a column vector in* ***M.E.S.S.****.*
- mxArray * mess_vector_to_mexmess (mess_vector A)

  Copy a vector from ***M.E.S.S.*** *to* MATLAB.
- void mex_mess_direct_lu_select (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_direct_chol_select (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_multidirect_select (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_amd (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_arpack (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_bzip2 (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_cholmod (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_colamd (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_csparse (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_matio (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_mess64 (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_mklpardiso (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_openmp (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_superlu (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_umfpack (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_have_zlib (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_is_debug (int nlhs, mxArray *plhs[], int nrsh, const mxArray *prhs[])
- void mex_mess_version (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version_major (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version_minor (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version_patch (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version_verbose (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_git_id (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_git_branch (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_lyap (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_care (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])

- void mex_mess_dense_nm_gmpare (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_lrnm (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_lradi (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_sylvester_sparsedense (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_sylvester_dense (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_glyap (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_gstein (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_direct_cholpackage_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_direct_lupackage_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_equation_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_memusage_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_multidirect_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_operation_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_parameter_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_residual_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_mess_norm_t (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_vector (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_matrix (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_status (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_test_options (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])

10.268.1 Detailed Description

This header file provides an interface to MATLAB to convert matrices easily from MATLAB to **M.E.S.S.** and the way back. Include this file after you included the mex.h header.

10.268.2 Macro Definition Documentation

10.268.2.1 #define MESS64

10.268.2.2 #define MESS_MATLAB

10.269 inverse.c File Reference

Functions

- int select_solver (int i, mess_matrix A, mess_direct sol)

  Select a solver.
- int testmat (int i, mess_matrix A)

  Test inverting a matrix.
- int main (int argc, char **argv)

10.269.1 Detailed Description

Author

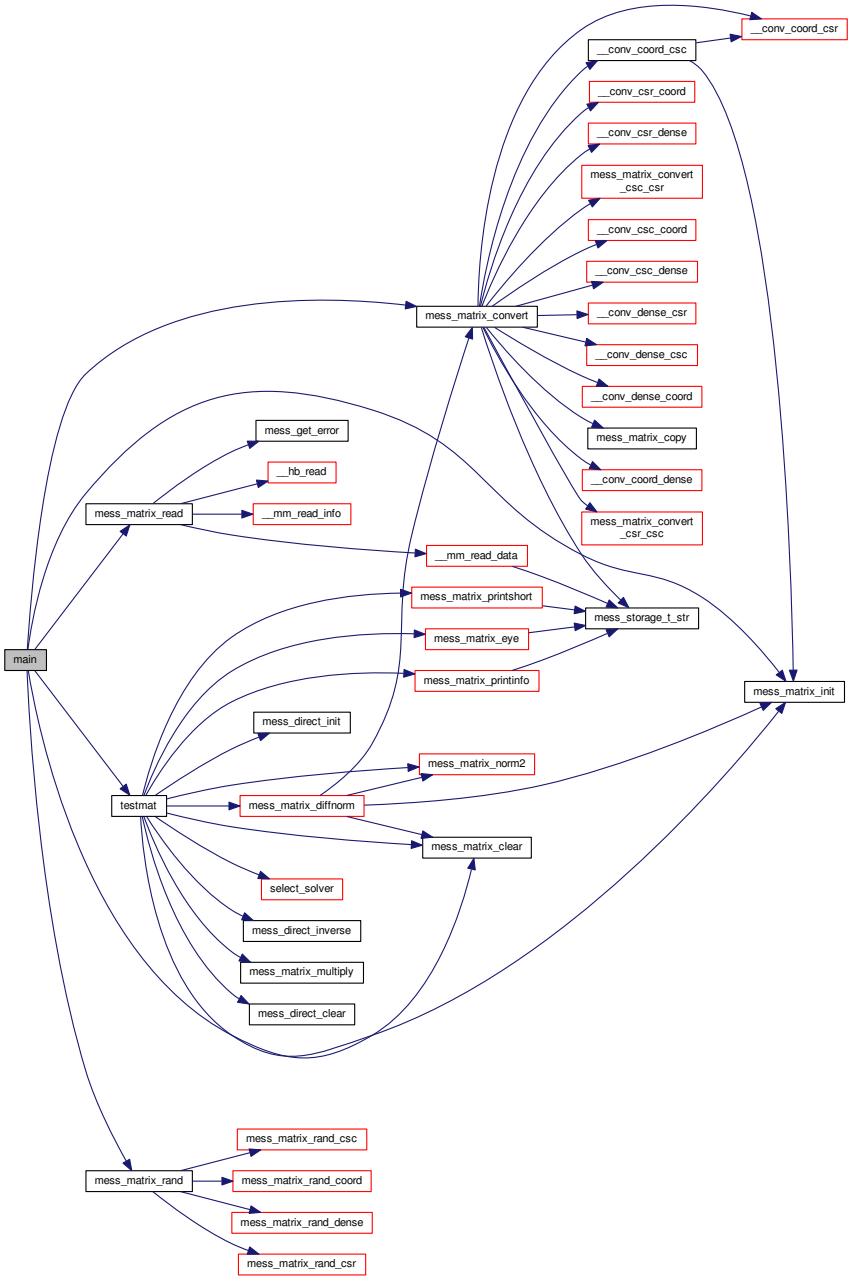
Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

- 0 - mess_direct_create_lapack_lu,
- 1 - mess_direct_create_csparse_lu,
- 2 - mess_direct_create_umfpack,
- 3 - mess_direct_create_sparse_lu,
- 4 - mess_direct_create_cholesky,
- 5 - mess_direct_create_csparse_cholesky.

10.269.2 Function Documentation

10.269.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.269.2.2 int select_solver (int *i*, mess_matrix *A*, mess_direct *sol*)

Parameters

in	*i*	input solver to choose
in	*A*	input matrix
out	*sol*	generated solver

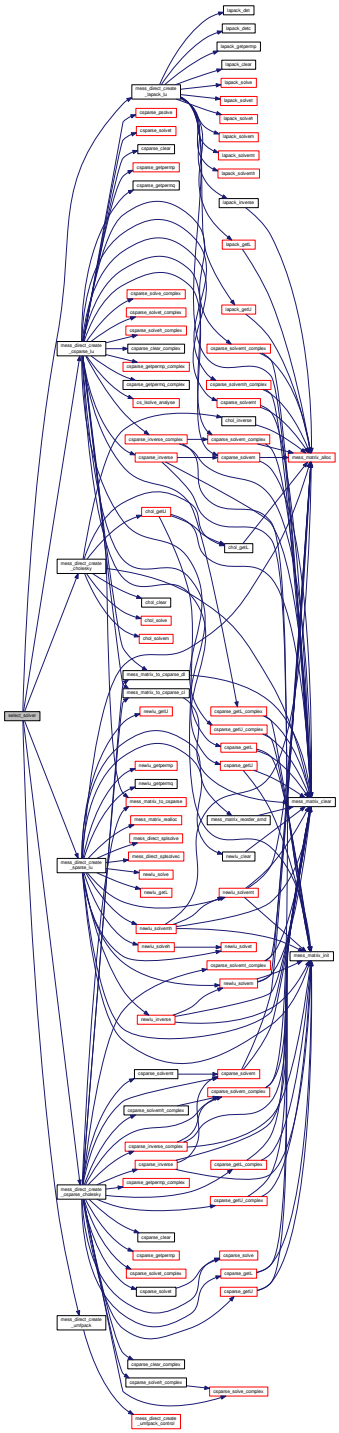
Returns

zero on success or a non-zero error value otherwise

The select_solver function generates a solver for a matrix A.
Solvers are choosen with i. Possible values are:

- 0 - mess_direct_create_lapack_lu,
- 1 - mess_direct_create_csparse_lu,
- 2 - mess_direct_create_umfpack,
- 3 - mess_direct_create_sparse_lu,
- 4 - mess_direct_create_cholesky,
- 5 - mess_direct_create_csparse_cholesky.

Here is the call graph for this function:



10.269.2.3 int testmat (int *i*, mess_matrix *A*)

Parameters

in	*i*	input solver to choose
in	*A*	input matrix

Returns

zero on success or a non-zero error value otherwise

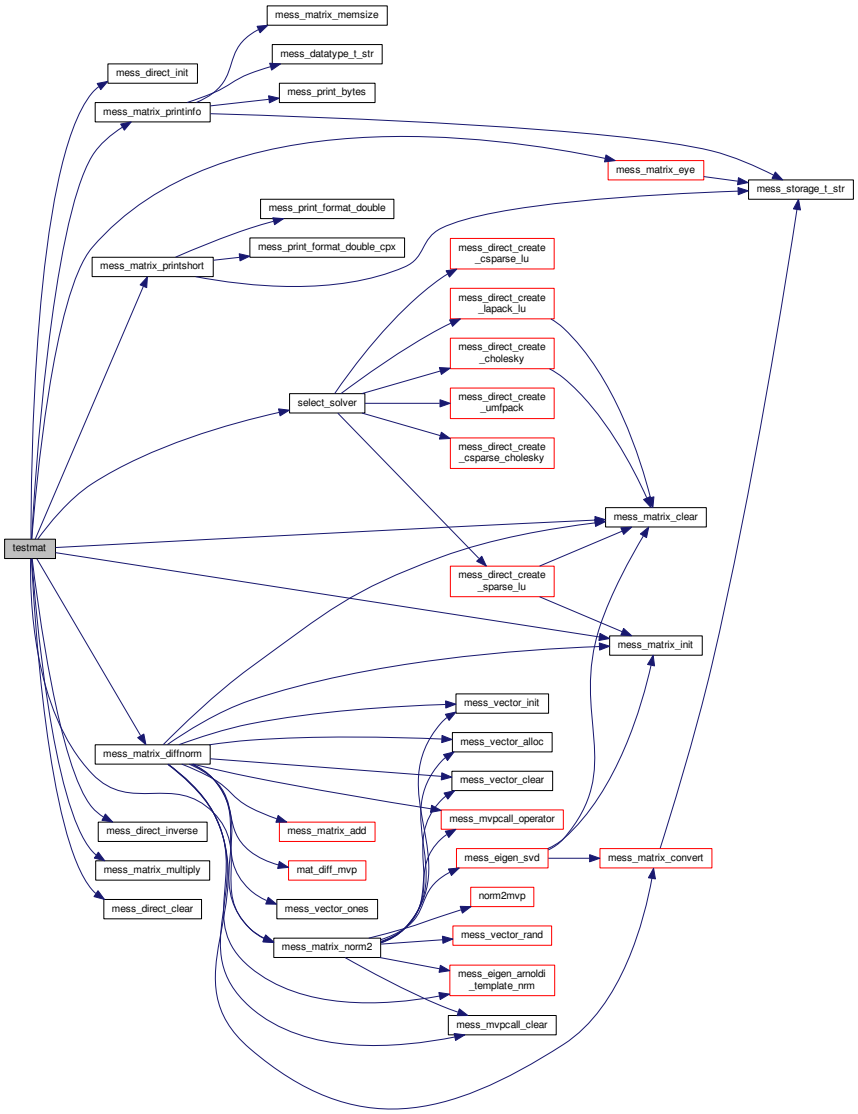
The testmat function generates an inverse for a matrix A and checks if

$$AA^{-1} = I,$$

where I denotes the identity matrix. Parameter i is only used to choose the solver. Possible values are:

- 0 - mess_direct_create_lapack_lu,
- 1 - mess_direct_create_csparse_lu,
- 2 - mess_direct_create_umfpack,
- 3 - mess_direct_create_sparse_lu,
- 4 - mess_direct_create_cholesky,
- 5 - mess_direct_create_csparse_cholesky.

Here is the call graph for this function:



10.270 irka_biorth.c File Reference

Macros

- #define STARTTIME(X) {MSG_WARN(X); ts=mess_wtime();}
- #define ENDTIME {te=mess_wtime(); MSG_WARN("end: %lg s\n", te-ts);}

Functions

- int mess_h2_irka_biorth (mess_dynsys orig, mess_vector sigma, mess_h2_options opt, mess_dynsys reduced, mess_matrix V, mess_matrix W, mess_h2_status status)

  Compute a $\mathcal{H}_2$ reduced model via the IRKA algorithm (biorthonormal version).

10.270.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.270.2 Macro Definition Documentation

10.270.2.1 #define ENDTIME {te=mess_wtime(); MSG_WARN("end: %lg s\n", te-ts);}

10.270.2.2 #define STARTTIME(X) {MSG_WARN(X); ts=mess_wtime();}

10.271 irka_common.c File Reference

Data Structures

- struct VWjob

Macros

- #define AR_W_VA 1

Functions

- static int __project_A (mess_matrix W, mess_matrix A, mess_matrix V, mess_matrix Ar)

  Project A with W and V.
- static int __project_B (mess_matrix W, mess_vector b, mess_vector br)

  Project b.
- static int __project_Bmat (mess_matrix W, mess_matrix B, mess_matrix Br)

  Project B (matrix version).
- static int __project_C (mess_matrix V, mess_vector c, mess_vector cr)

  Project c.
- static int __project_Cmat (mess_matrix V, mess_matrix C, mess_matrix Cr)

  Project C (matrix version).
- static int __constructVW_seq (mess_matrix A, mess_direct Abase, mess_matrix E, mess_vector b, mess_vector c, mess_vector sigma, mess_matrix V, mess_matrix W)

  Construct V and W for IRKA (sequential).
- static int __constructVW_sec (mess_matrix A, mess_direct Abase, mess_matrix E, mess_vector b, mess_vector c, mess_vector sigma, mess_matrix V, mess_matrix W)

  Construct V and W for IRKA (OpenMP version).
- static void VW_worker (void *data)
- static int __constructVW_pre (mess_matrix A, mess_direct Abase, mess_matrix E, mess_vector b, mess_vector c, mess_vector sigma, mess_matrix V, mess_matrix W)

  Construct V and W for IRKA (OpenMP Preprocessing).
- static int __constructVW_pthread (mess_matrix A, mess_direct Abase, mess_matrix E, mess_vector b, mess_vector c, mess_vector sigma, mess_matrix V, mess_matrix W)

  Construct V and W for IRKA (OpenMP Preprocessing).
- static char * get_char (char *c1, char *c2)
- static int __constructVW (mess_matrix A, mess_direct Abase, mess_matrix E, mess_vector b, mess_vector c, mess_vector sigma, mess_matrix V, mess_matrix W)

  Construct V and W for IRKA (wrapper).
- static int __constructVWmat (mess_matrix A, mess_direct Abase, mess_matrix E, mess_matrix B, mess_matrix C, mess_vector sigma, mess_matrix V, mess_matrix W)

  Construct V and W for IRKA (wrapper, matrix inputs).

10.271.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.272 irka_para_init.c File Reference

Functions

- static int compare_complex (const void *pc1, const void *pc2)

  Compare two complex numbers for qsort.
- int mess_h2_irka_init (mess_matrix A, mess_direct Asolver, mess_matrix E, mess_int_t *r0, mess_int_t kp, mess_int_t km, mess_vector sigma)

  Initialize parameters for IRKA.

10.272.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.272.2 Function Documentation

10.272.2.1 static int compare_complex (const void * *pc1*, const void * *pc2*) [static]

Parameters

in	*pc1*	input pointer to the first number
in	*pc2*	input pointer to the second number

Returns

-1, 0 , 1 The compare_complex function compares two complex numbers by the absolute value (cabs). If the absolute value is equal, the comparison is done by the complex argument (carg) The function is used for stdlibc qsort function.

10.273 jacobi.c File Reference

Functions

- int mess_solver_jacobi (mess_matrix matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with the Jacobi method.
- int mess_solver_jacobi_convergence (mess_matrix matrix, int *convergence)

  Check if the Jacobi iteration is guaranteed to converge.

10.273.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.274 kron.c File Reference

Functions

- int mess_vector_kron (mess_vector in1, mess_vector in2, mess_vector out)

  Compute Kronecker product for two vectors.

10.274.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.275 lanczos_template_nrm.c File Reference

Functions

- int mess_eigen_lanczos_template_nrm (mess_mvpcall A, mess_int_t k, mess_vector sv, mess_vector diag, mess_vector subdiag, mess_matrix V, double *abseig)

  Lanczos method for 2-norm computations (real symmetric case).

10.275.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file contains the template for the lanczos iteration (hermitian case).

10.276 lapack.c File Reference

Data Structures

- struct lapack_solver

  Internal structure for the LAPACK based solver.

Functions

- static int lapack_clear (void *data)

  Clear the LAPACK solver.
- static int lapack_solve (void *data, mess_vector b, mess_vector x)

  Solve $Ax = b$ using LAPACK.
- static int lapack_solvet (void *data, mess_vector b, mess_vector x)

  Solve $A^T x = b$ using LAPACK.
- static int lapack_solveh (void *data, mess_vector b, mess_vector x)

  Solve $A^H x = b$ using LAPACK.
- static int lapack_solvem (void *data, mess_matrix b, mess_matrix x)

  Solve $AX = B$ using LAPACK (matrix version).
- static int lapack_solvemt (void *data, mess_matrix b, mess_matrix x)

  Solve $A^T X = B$ using LAPACK.
- static int lapack_solvemh (void *data, mess_matrix b, mess_matrix x)

  Solve $A^H X = B$ using LAPACK.
- static int lapack_inverse (void *data, mess_matrix inv)

  Compute the inverse of a matrix using LAPACK.
- static int lapack_det (void *data, double *m, double *e)

  Compute the determinant of a matrix using LAPACK.
- static int lapack_detc (void *data, double *mr, double *mi, double *e)

  Compute the determinant of a matrix using LAPACK.
- static int lapack_getL (void *data, mess_matrix L)

  Get the L (the first factor) of a solver.
- static int lapack_getU (void *data, mess_matrix U)

  Get the U (the second factor) of a solver.
- static int lapack_getpermp (void *data, mess_int_t *p)

  Get the row permutation.
- int mess_direct_create_lapack_lu (mess_matrix matrix, mess_direct solver)

  Generate a dense LAPACK based LU-solver.

10.276.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.276.2 Function Documentation

10.276.2.1 static int lapack_clear (void * *data*) [static]

Parameters

in,out	*data*	pointer to the internal data

Returns

always zero

The lapack_clear function clears a LAPACK solver.

10.276.2.2 static int lapack_det (void * *data*, double * *m*, double * *e*) [static]

Parameters

in	*data*	input pointer to the internal data
out	*m*	output mantissa of determinant
out	*e*	output exponent of determinant

Returns

zero on success or a non-zero error value otherwise

The lapack_det function computes $\det(A)$ using LAPACK.

10.276.2.3 static int lapack_detc (void * *data*, double * *mr*, double * *mi*, double * *e*) [static]

Parameters

in	*data*	input pointer to the internal data
out	*mr*	output mantissa for real part of determinant
out	*mi*	output mantissa for imaginary part of determinant
out	*e*	output exponent of determinant

Returns

zero on success or a non-zero error value otherwise

The lapack_detc function computes $\det(A)$ using LAPACK.

10.276.2.4 static int lapack_getL (void * *data*, mess_matrix *L*) [static]

Parameters

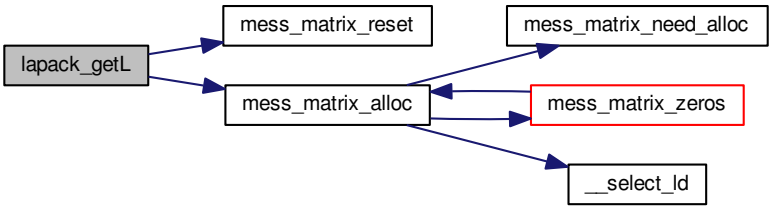
in	*data*	input solver data
in,out	*L*	output L factor

Returns

zero on success or a non-zero error value otherwise

The lapack_getL function gets the L factor of $A = PLU$. Output is in MESS_DENSE format.

Here is the call graph for this function:



10.276.2.5 static int lapack_getpermp (void * *data*, mess_int_t * *p*) [static]

Parameters

in	*data*	input solver data
in,out	*p*	output permutation

Returns

zero on success or a non-zero error value otherwise

The lapack_getpermp function gets the row permutation of the decomposition $PA = LU$.

10.276.2.6 static int lapack_getU (void * *data*, mess_matrix *U*) [static]

Parameters

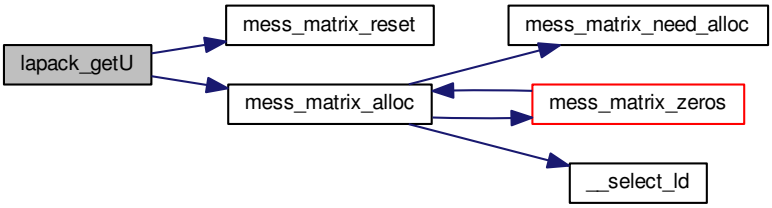
in	*data*	input solver data
in,out	*U*	output U factor

Returns

zero on success or a non-zero error value otherwise

The lapack_getU function gets the U factor of $A = PLU$. Output is in MESS_DENSE format.

Here is the call graph for this function:



10.276.2.7 static int lapack_inverse (void * *data*, **mess_matrix** *inv*) [static]

Parameters

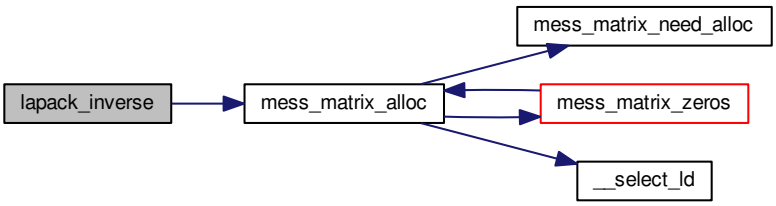
in	*data*	input pointer to the internal data
in,out	*inv*	output inverse

Returns

zero on success or a non-zero error value otherwise

The lapack_inverse function computes A^{-1} using LAPACK.

Here is the call graph for this function:



10.276.2.8 static int lapack_solve (void * *data*, **mess_vector** *b*, **mess_vector** *x*) [static]

Parameters

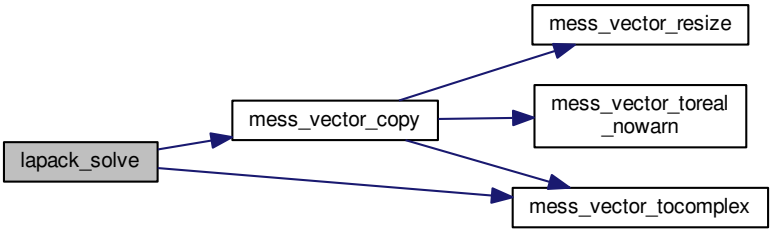
in	*data*	input pointer to the internal data structure
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The lapack_solve function solves $Ax = b$ using LAPACK.

Here is the call graph for this function:



10.276.2.9 static int lapack_solveh (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

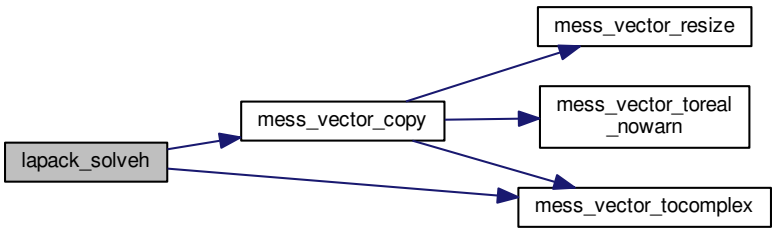
in	*data*	input pointer to the internal data structure
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The lapack_solveh function solves $A^H x = b$ using LAPACK.

Here is the call graph for this function:



10.276.2.10 static int lapack_solvem (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

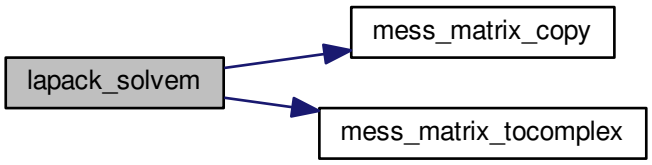
in	*data*	input pointer to the internal data structure
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The lapack_solvem function solves $AX = B$ using LAPACK where B and X are matrices.

Here is the call graph for this function:



10.276.2.11 static int lapack_solvemh (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

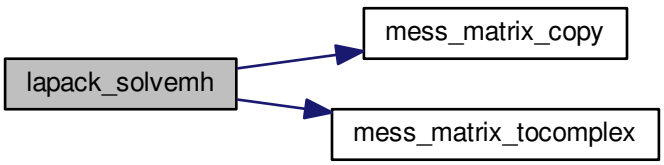
in	*data*	input pointer to the internal data structure
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The lapack_solvemt function solves $A^H X = B$ using LAPACK where B and X are matrices.

Here is the call graph for this function:



10.276.2.12 static int lapack_solvemt (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

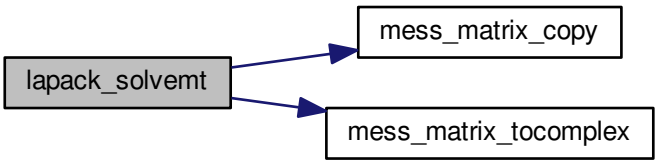
in	*data*	input pointer to the internal data structure
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The lapack_solvemt function solves $A^T X = B$ using LAPACK where B and X are matrices .

Here is the call graph for this function:



10.276.2.13 static int lapack_solvet (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

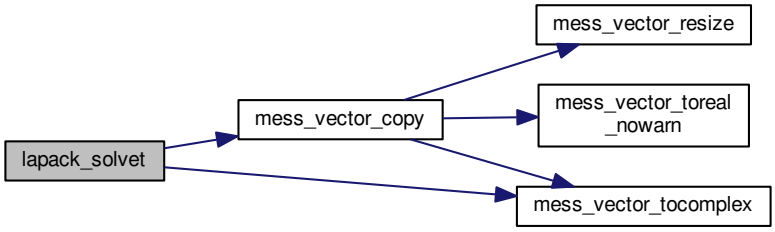
in	*data*	input pointer to the internal data structure
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non-zero error value otherwise

The lapack_solvet function solves $A^T x = b$ using LAPACK.

Here is the call graph for this function:



10.277 lapack_qr.c File Reference

Data Structures

- struct lapackqr_solver

  Internal data structure. Internal data structure for QR/LQ decompostitions.

Functions

- static int lapackqr_clear (void ∗data)

  Clear a LAPACK QR solver.
- static int lapackqr_solve_over (void ∗data, mess_vector b, mess_vector x)

  Solve $Ax = (QR)x = b$.
- static int lapackqr_solvet_over (void ∗data, mess_vector b, mess_vector x)

  Solve $A^T x = (QR)^T x = b$.
- static int lapackqr_solveh_over (void ∗data, mess_vector b, mess_vector x)

  Solve $A^H x = (QR)^H x = b$.
- static int lapackqr_solvem_over (void ∗data, mess_matrix b, mess_matrix x)

  Solve $AX = (QR)X = B$ *(matrix version).*
- static int lapackqr_solvemt_over (void ∗data, mess_matrix b, mess_matrix x)

  Solve $A^T x = (QR)^T x = b(matrixversion)$.
- static int lapackqr_solvemh_over (void ∗data, mess_matrix b, mess_matrix x)

  Solve $A^H x = (QR)^H x = b(matrixversion)$.
- static int lapackqr_getL_over (void ∗data, mess_matrix L)

  Get the L (the first factor) of a solver.
- static int lapackqr_getU_over (void ∗data, mess_matrix U)

  Get the U (the second factor) of a solver.
- static int mess_direct_create_lapack_qr_over (mess_matrix matrix, mess_direct solver)

  Construct a solver based on QR-decomposition from LAPACK for overdetermined systems.
- static int lapackqr_solve_under (void ∗data, mess_vector b, mess_vector x)

Solve $Ax = (LQ)x = b$.

- static int lapackqr_solvet_under (void *data, mess_vector b, mess_vector x)

  Solve $Ax = (LQ)^T x = b$.
- static int lapackqr_solveh_under (void *data, mess_vector b, mess_vector x)

  Solve $Ax = (LQ)^H x = b$.
- static int lapackqr_solvem_under (void *data, mess_matrix b, mess_matrix x)

  Solve $Ax = (LQ)x = b$ *(matrix version).*
- static int lapackqr_solvemt_under (void *data, mess_matrix b, mess_matrix x)

  Solve $Ax = (LQ)^T x = b(matrixversion)$.
- static int lapackqr_solvemh_under (void *data, mess_matrix b, mess_matrix x)

  Solve $Ax = (LQ)^H x = b$.
- static int lapackqr_getL_under (void *data, mess_matrix L)

  Get the L (the first factor) of a solver.
- static int lapackqr_getU_under (void *data, mess_matrix U)

  Get the U (the second factor) of a solver.
- static int mess_direct_create_lapack_qr_under (mess_matrix matrix, mess_direct solver)

  Construct a solver based on LQ-decomposition from `LAPACK` *for underdetermined Systems.*
- int mess_direct_create_lapack_qr (mess_matrix matrix, mess_direct solver)

  Construct a solver based on QR or LQ decomposition from `LAPACK`.

10.277.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.277.2 Function Documentation

10.277.2.1 static int lapackqr_clear (void * *data*) [static]

Parameters

in,out	*data*	pointer to the data structure

Returns

always zero

The lapackqr_clear function clear a lapack-qr solver.

Attention

Internal use only.

10.277.2.2 static int lapackqr_getL_over (void * *data,* mess_matrix *L*) [static]

Parameters

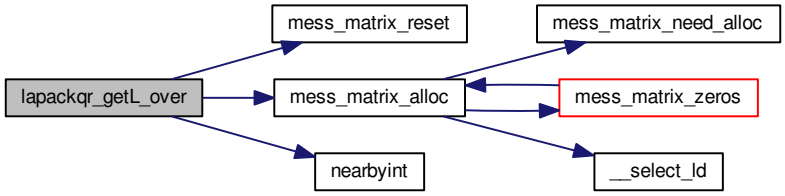
in	*data*	input solver data
in,out	*L*	output L factor

Returns

zero on success or a non-zero error value otherwise

The lapackqr_getL_over function gets the Q factor of $A = QR$. Output is in MESS_DENSE format.

Here is the call graph for this function:



10.277.2.3 static int lapackqr_getL_under (void * *data*, mess_matrix *L*) [static]

Parameters

in	*data*	input solver data
in,out	*L*	output L factor

Returns

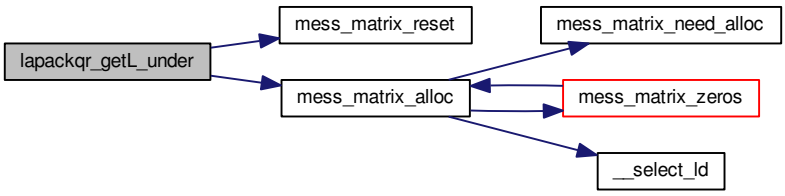
zero on success or a non-zero error value otherwise

The lapackqr_getL_under function gets the L factor of $A = LQ$. Output is in MESS_DENSE format.

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.4 static int lapackqr_getU_over (void * *data*, mess_matrix *U*) [static]

Parameters

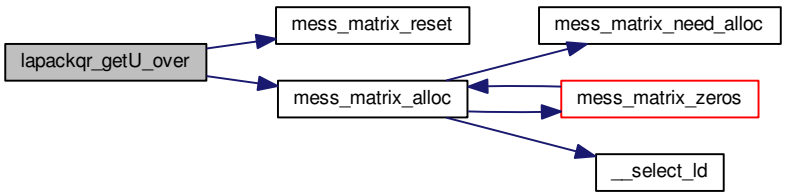
in	*data*	input solver data
in,out	*U*	output U factor

Returns

zero on success or a non-zero error value otherwise

The lapackqr_getU_over function gets the R factor of $A = QR$. Output is in MESS_DENSE format.

Here is the call graph for this function:



10.277.2.5 static int lapackqr_getU_under (void * *data*, mess_matrix *U*) [static]

Parameters

in	*data*	input solver data
in,out	*U*	output U factor

Returns

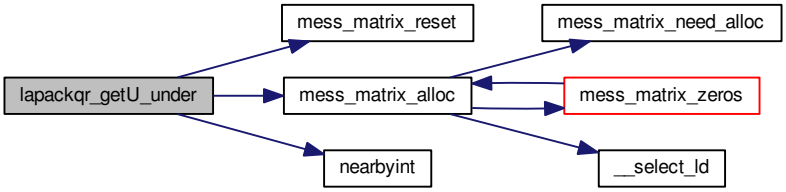
zero on success or a non-zero error value otherwise

The lapackqr_getU_under function gets the Q factor of $A = LQ$. Output is in MESS_DENSE format.

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.6 static int lapackqr_solve_over (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input data of the solver
in	*b*	input right hand side
in,out	*x*	input/output solution

Returns

zero on success or a non zero error code otherwise

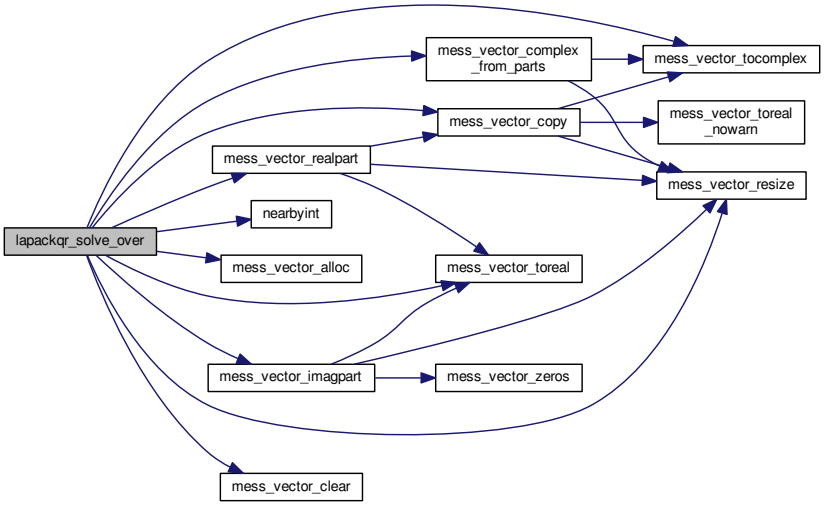
The lapackqr_solve_over function solves the overdetermined system

$$Ax = b \Leftrightarrow QRx = b.$$

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.7 static int lapackqr_solve_under (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input data of the solver
in	*b*	input right hand side
in,out	*x*	input/output solution

Returns

zero on success or a non zero error code otherwise

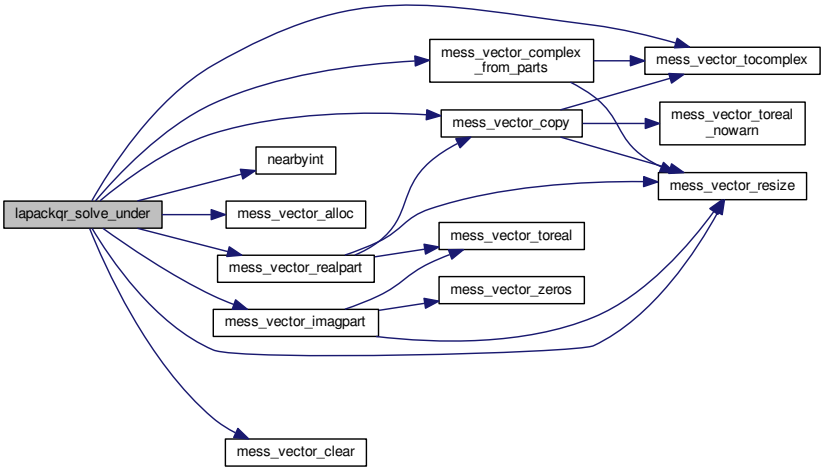
The lapackqr_solve_under function solves the underdetermined system

$$Ax = b \Leftrightarrow LQx = b.$$

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.8 static int lapackqr_solveh_over (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input data of the solver
in	*b*	input right hand side
in,out	*x*	input/output solution

Returns

zero on success or a non zero error code otherwise

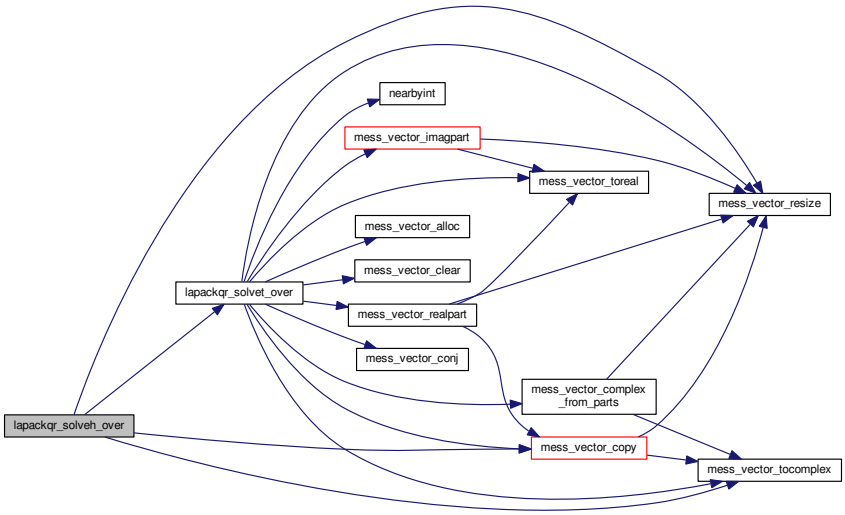
The lapackqr_solvet_over function solves the overdetermined system

$$A^T x = b \Leftrightarrow (QR)^H x = b.$$

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.9 static int lapackqr_solveh_under (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input data of the solver
in	*b*	input right hand side
in,out	*x*	input/output solution

Returns

zero on success or a non zero error code otherwise

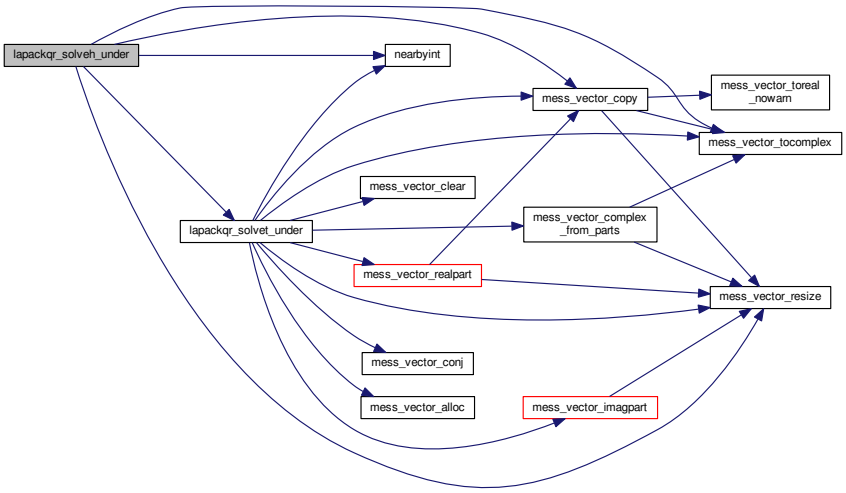
The lapackqr_solve_under function solves the underdetermined system

$$Ax = b \Leftrightarrow (LQ)^H x = b.$$

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.10 static int lapackqr_solvem_over (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input data of the solver
in	*b*	input right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error code otherwise

The lapackqr_solvem_over function solves the overdetermined system

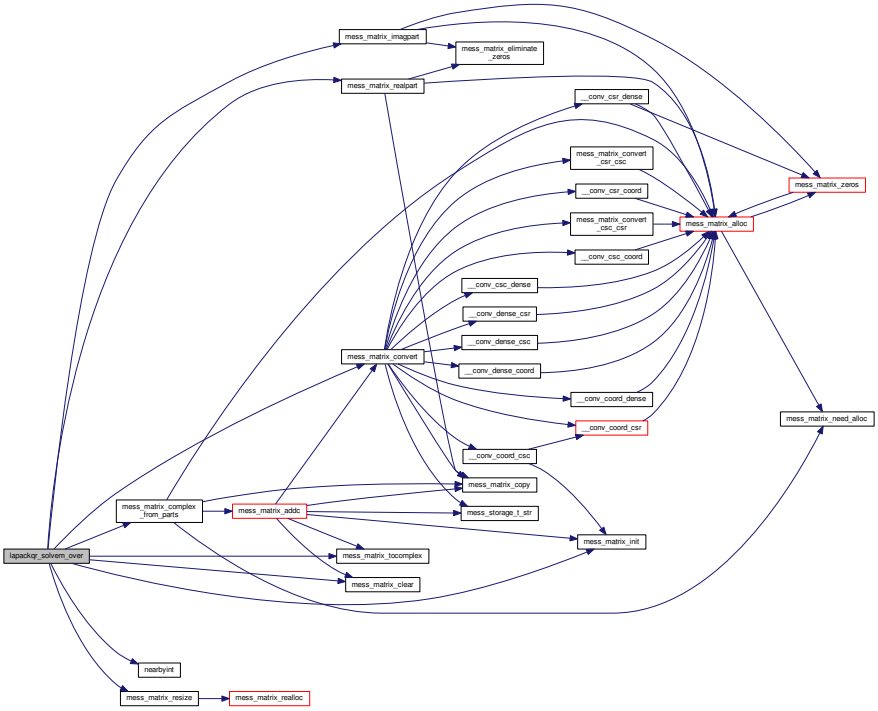
$$Ax = b \Leftrightarrow QRx = b,$$

where b and x are matrices.

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.11 static int lapackqr_solvem_under (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input data of the solver
in	*b*	input right hand side
in,out	*x*	input/output solution

Returns

zero on success or a non zero error code otherwise

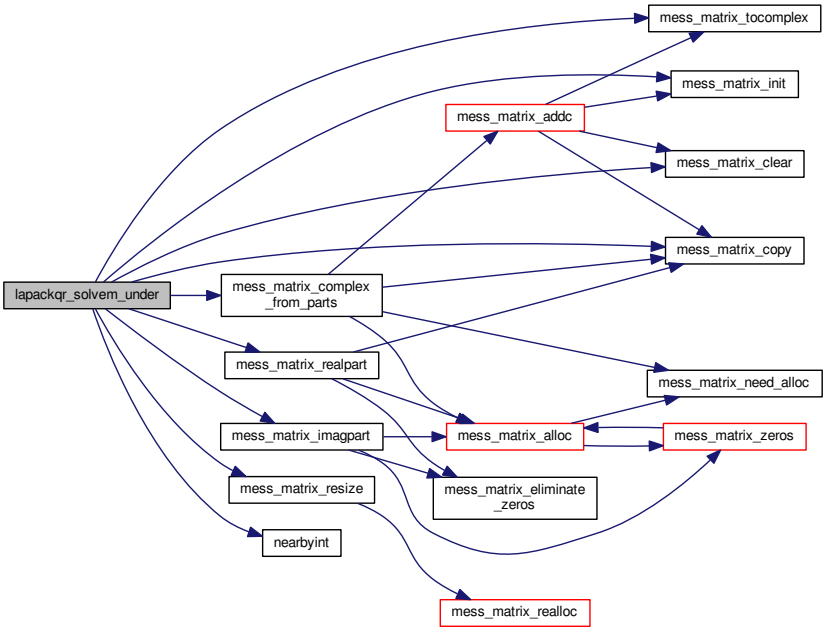
The lapackqr_solvem_under function solves the underdetermined system

$$Ax = b \Leftrightarrow LQx = b.$$

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.12 static int lapackqr_solvemh_over (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input data of the solver
in	*b*	input right hand side
in,out	*x*	input/output solution

Returns

zero on success or a non zero error code otherwise

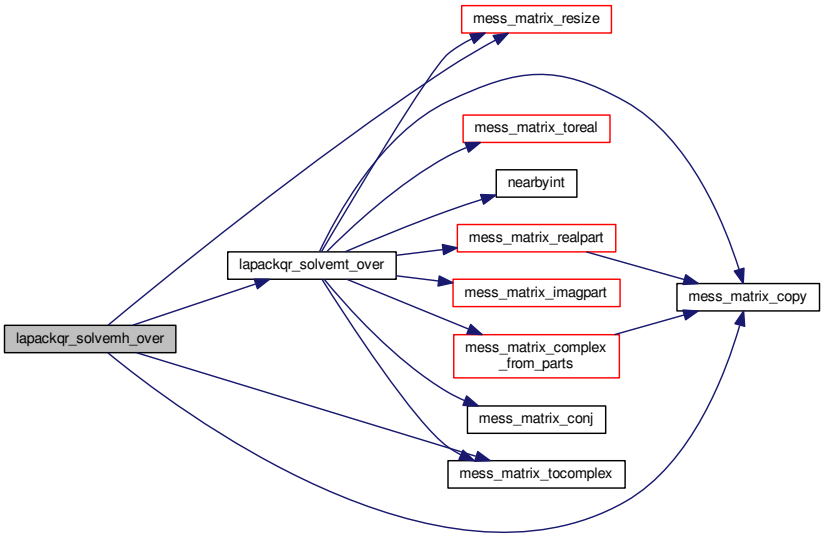
The lapackqr_solvemh_over function solves the overdetermined system

$$A^T x = b \Leftrightarrow (QR)^H x = b.$$

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.13 static int lapackqr_solvemh_under (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input data of the solver
in	*b*	input right hand side
in,out	*x*	input/output solution

Returns

zero on success or a non zero error code otherwise

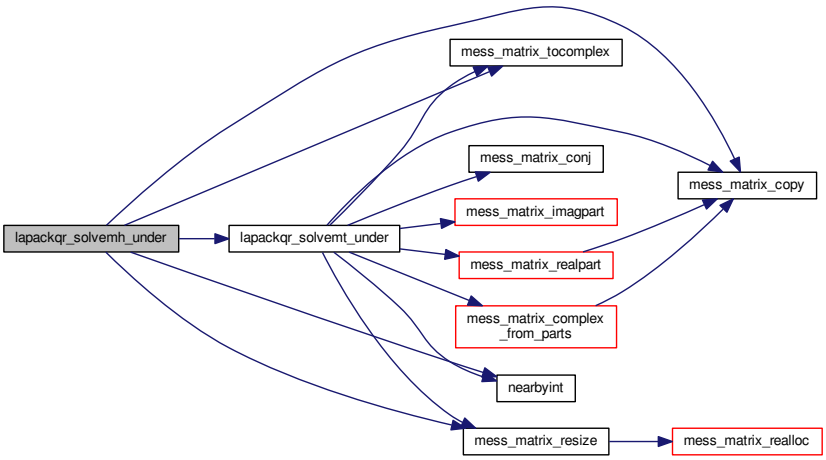
The lapackqr_solve_under function solves the underdetermined system

$$Ax = b \Leftrightarrow (LQ)^H x = b.$$

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.14 static int lapackqr_solvemt_over (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input data of the solver
in	*b*	input right hand side
in,out	*x*	input/output solution

Returns

zero on success or a non zero error code otherwise

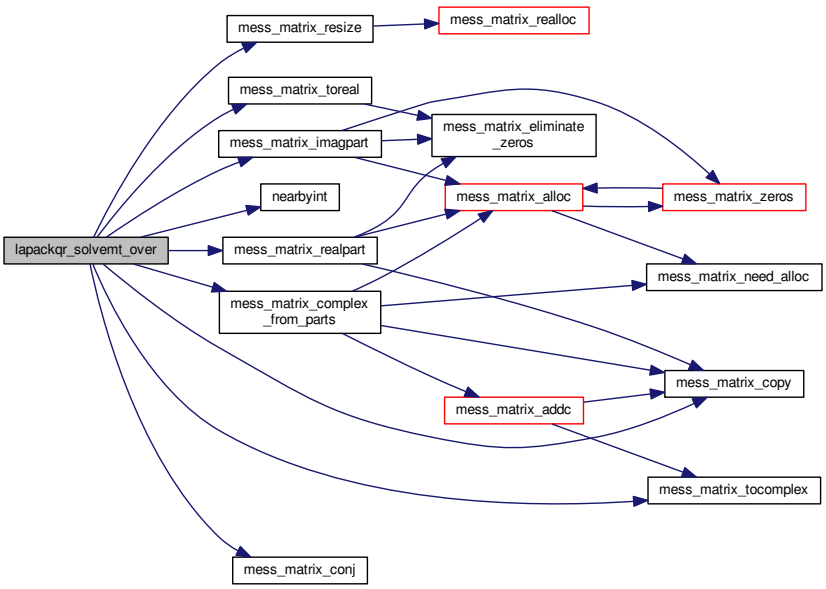
The lapackqr_solvemt_over function solves the overdetermined system

$$A^T x = b \Leftrightarrow (QR)^T x = b.$$

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.15 static int lapackqr_solvemt_under (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input data of the solver
in	*b*	input right hand side
in,out	*x*	input/output solution

Returns

zero on success or a non zero error code otherwise

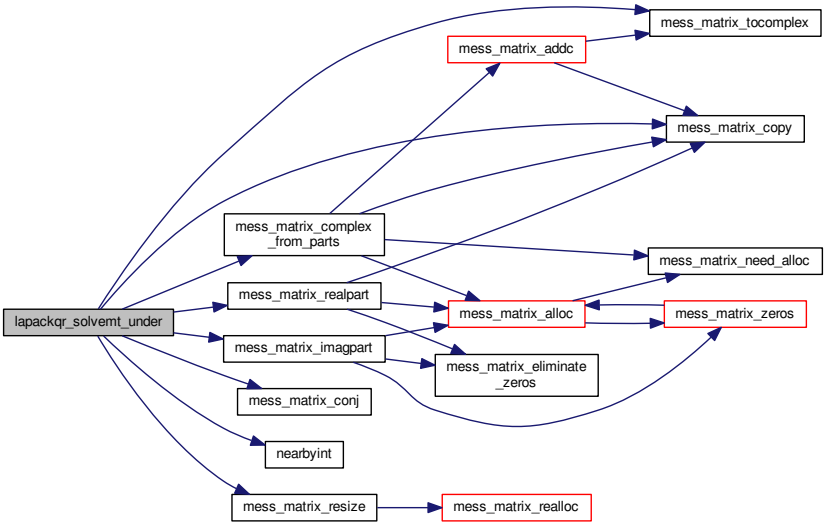
The lapackqr_solve_under function solves the underdetermined system

$$Ax = b \Leftrightarrow (LQ)^T x = b.$$

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.16 static int lapackqr_solvet_over (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input data of the solver
in	*b*	input right hand side
in,out	*x*	input/output solution

Returns

zero on success or a non zero error code otherwise

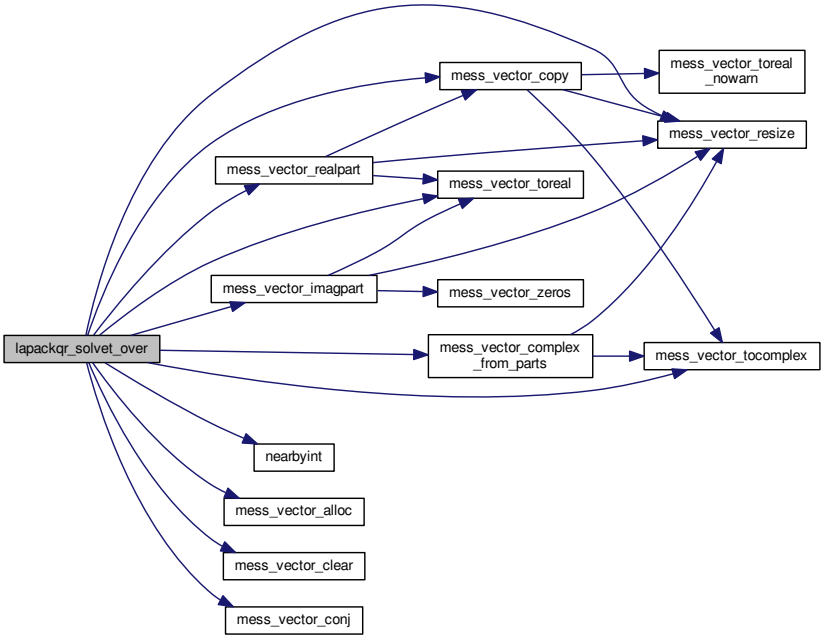
The lapackqr_solvet_over function solves the overdetermined system

$$A^T x = b \Leftrightarrow (QR)^T x = b.$$

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.17 static int lapackqr_solvet_under (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input data of the solver
in	*b*	input right hand side
in,out	*x*	input/output solution

Returns

zero on success or a non zero error code otherwise

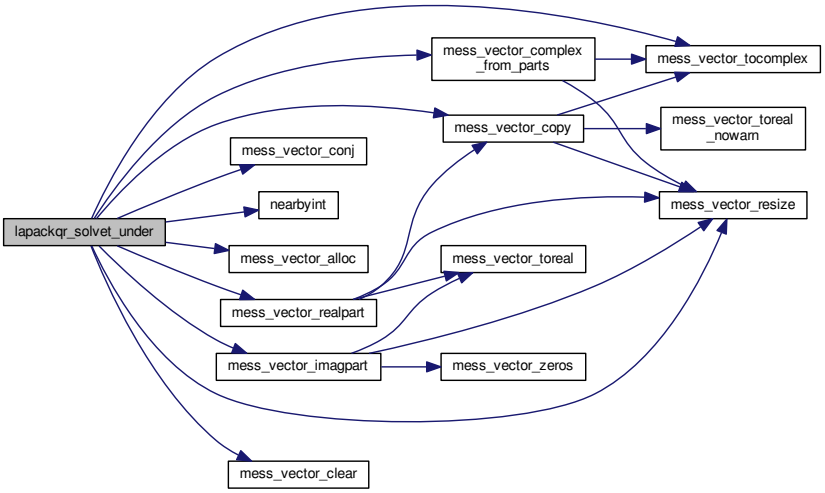
The lapackqr_solve_under function solves the underdetermined system

$$Ax = b \Leftrightarrow (LQ)^T x = b.$$

Attention

Internal use only.

Here is the call graph for this function:



10.277.2.18 static int mess_direct_create_lapack_qr_over (mess_matrix *matrix*, mess_direct *solver*) [static]

Parameters

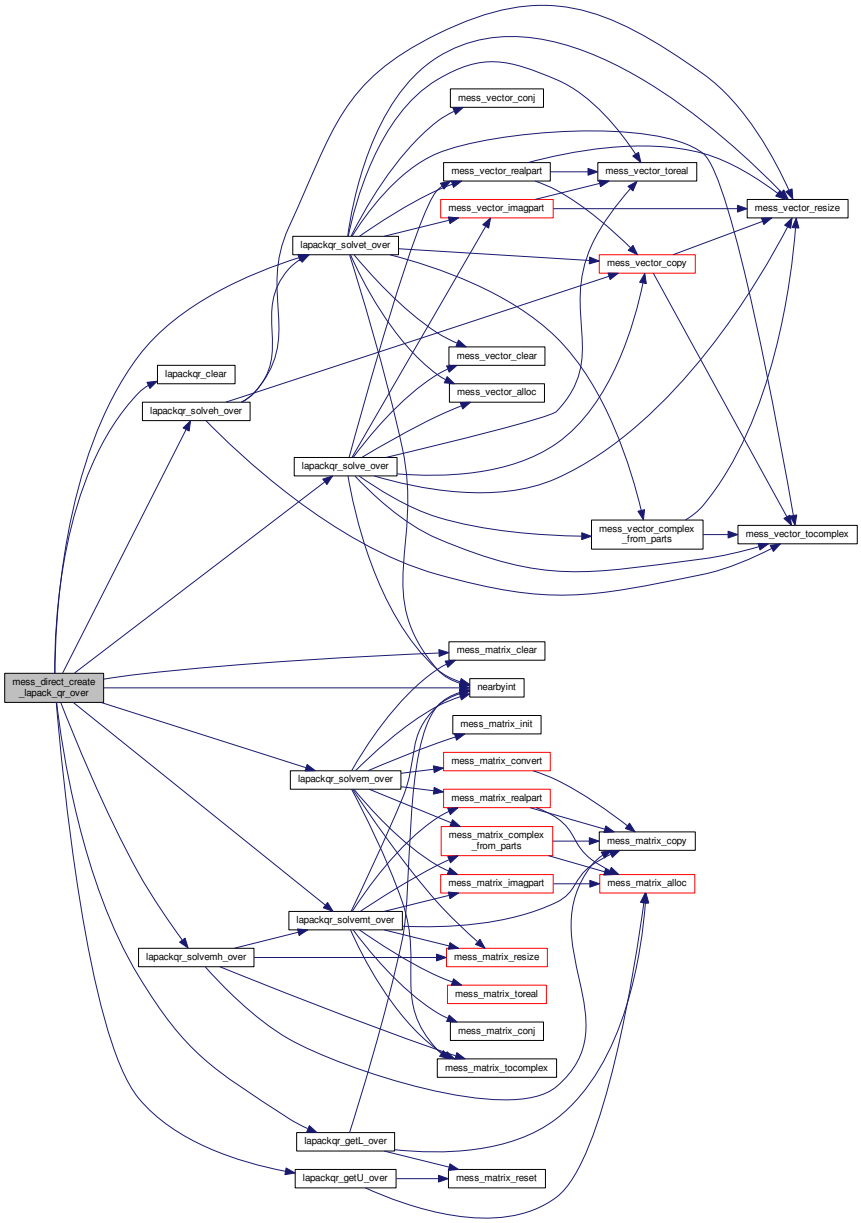
in	*matrix*	input system matrix
out	*solver*	output

Returns

zero on success or a non zero error code otherwise

The mess_direct_create_lapack_qr_over function generates a solver based on the QR-decomposition from LAP↩ ACK for overdetermined systems, i.e. the number of rows is greater equals than the number of cols.

Here is the call graph for this function:



10.277.2.19 static int mess_direct_create_lapack_qr_under (mess_matrix *matrix*, mess_direct *solver*) [static]

Parameters

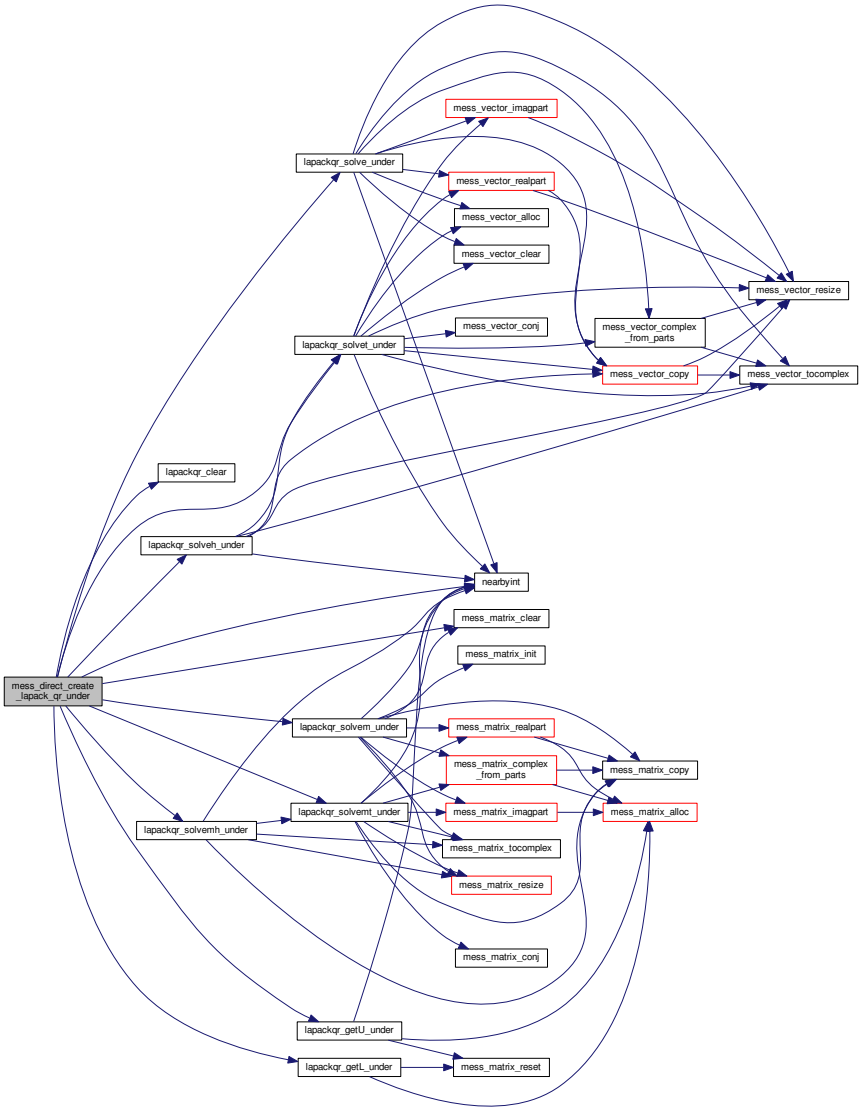
in	*matrix*	input system matrix
out	*solver*	output

Returns

zero on success or a non zero error code otherwise

The mess_direct_create_lapack_qr_under function generates a solver based on the LQ-decomposition from LAPACK for underdetermined systems, i.e. the number of cols is greater than the number of rows. Internal use only.

Here is the call graph for this function:



10.278 lift.c File Reference

Functions

- int mess_matrix_lift (mess_matrix in, mess_int_t n, mess_matrix out)

  Add lower trailing block of n rows to matrix.

10.278.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.279 link.txt File Reference

10.280 link.txt File Reference

10.281 link.txt File Reference

10.282 link.txt File Reference

10.283 link.txt File Reference

10.284 link.txt File Reference

10.285 link.txt File Reference

10.286 link.txt File Reference

10.287 link.txt File Reference

Variables

- usr bin gfortran fPIC g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -shared -Wl
- usr bin gfortran fPIC g3 fPIC Wall soname

10.287.1 Variable Documentation

10.287.1.1 usr bin gfortran fPIC g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -shared -Wl

10.287.1.2 usr bin gfortran fPIC g3 fPIC Wall soname

10.288 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩MakeFiles/check_backslash.dir/check_backslash.c.o -o check_backslash -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.288.1 Variable Documentation

10.288.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_backslash.dir/check_backslash.c.o -o check_backslash -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.288.1.2 usr bin cc g3 fPIC Wall rpath

10.289 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩Files/check_det.dir/check_det.c.o -o check_det -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.289.1 Variable Documentation

10.289.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_↩det.dir/check_det.c.o -o check_det -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.289.1.2 usr bin cc g3 fPIC Wall rpath

10.290 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/check_dgeqp3.dir/check_dgeqp3.c.o -o check_dgeqp3 -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.290.1 Variable Documentation

10.290.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dgeqp3.dir/check_dgeqp3.c.o -o check_dgeqp3 -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.290.1.2 usr bin cc g3 fPIC Wall rpath

10.291 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩MakeFiles/check_direct_care.dir/check_direct_care.c.o -o check_direct_care -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.291.1 Variable Documentation

10.291.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_direct_care.dir/check_direct_care.c.o -o check_direct_care -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.291.1.2 usr bin cc g3 fPIC Wall rpath

10.292 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDE↩BUG CMakeFiles/check_getLU.dir/check_getLU.c.o -o check_getLU -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.292.1 Variable Documentation

10.292.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_getLU.dir/check_getLU.c.o -o check_getLU -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.292.1.2 usr bin cc g3 fPIC Wall rpath

10.293 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/check_hessenberg.dir/check_hessenberg.c.o -o check_hessenberg -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.293.1 Variable Documentation

10.293.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_hessenberg.dir/check_hessenberg.c.o -o check_hessenberg -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.293.1.2 usr bin cc g3 fPIC Wall rpath

10.294 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lapackqr.dir/check_lapackqr.c.o -o check_lapackqr -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.294.1 Variable Documentation

10.294.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lapackqr.dir/check_lapackqr.c.o -o check_lapackqr -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.294.1.2 usr bin cc g3 fPIC Wall rpath

10.295 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/check_pinv.dir/check_pinv.c.o -o check_pinv -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.295.1 Variable Documentation

10.295.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_↩ pinv.dir/check_pinv.c.o -o check_pinv -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.295.1.2 usr bin cc g3 fPIC Wall rpath

10.296 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩ UG CMakeFiles/check_trisolve.dir/check_trisolve.c.o -o check_trisolve -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.296.1 Variable Documentation

10.296.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_trisolve.dir/check_trisolve.c.o -o check_trisolve -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.296.1.2 usr bin cc g3 fPIC Wall rpath

10.297 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/glyap_test.dir/glyap_test.c.o -o glyap_test -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.297.1 Variable Documentation

10.297.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/glyap_↩ test.dir/glyap_test.c.o -o glyap_test -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.297.1.2 usr bin cc g3 fPIC Wall rpath

10.298 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/gstein_test.dir/gstein_test.c.o -o gstein_test -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.298.1 Variable Documentation

10.298.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/gstein_↩ test.dir/gstein_test.c.o -o gstein_test -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.298.1.2 usr bin cc g3 fPIC Wall rpath

10.299 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/inverse.dir/inverse.c.o -o inverse -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.299.1 Variable Documentation

10.299.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/inverse.dir/inverse.c.o -o inverse -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.299.1.2 usr bin cc g3 fPIC Wall rpath

10.300 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/linsolve.dir/linsolve.c.o -o linsolve -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.300.1 Variable Documentation

10.300.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/linsolve.dir/linsolve.c.o -o linsolve -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.300.1.2 usr bin cc g3 fPIC Wall rpath

10.301 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/linsolve_multi.dir/linsolve_multi.c.o -o linsolve_multi -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.301.1 Variable Documentation

10.301.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/linsolve_multi.dir/linsolve_multi.c.o -o linsolve_multi -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.301.1.2 usr bin cc g3 fPIC Wall rpath

10.302 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/linsolvem.dir/linsolvem.c.o -o linsolvem -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.302.1 Variable Documentation

10.302.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/linsolvem.dir/linsolvem.c.o -o linsolvem -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.302.1.2 usr bin cc g3 fPIC Wall rpath

10.303 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/linsolvem_multi.dir/linsolvem_multi.c.o -o linsolvem_multi -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.303.1 Variable Documentation

10.303.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/linsolvem_multi.dir/linsolvem_multi.c.o -o linsolvem_multi -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.303.1.2 usr bin cc g3 fPIC Wall rpath

10.304 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩MakeFiles/sylvester_dense.dir/sylvester_dense.c.o -o sylvester_dense -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.304.1 Variable Documentation

10.304.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/sylvester_dense.dir/sylvester_dense.c.o -o sylvester_dense -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.304.1.2 usr bin cc g3 fPIC Wall rpath

10.305 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/sylvester_sparsedense.dir/sylvester_sparsedense.c.o -o sylvester_sparsedense -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.305.1 Variable Documentation

10.305.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/sylvester_sparsedense.dir/sylvester_sparsedense.c.o -o sylvester_sparsedense -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.305.1.2 usr bin cc g3 fPIC Wall rpath

10.306 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_arnoldi_template.dir/check_arnoldi_template.c.o -o check_arnoldi_template -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.306.1 Variable Documentation

10.306.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_arnoldi_template.dir/check_arnoldi_template.c.o -o check_arnoldi_template -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.306.1.2 usr bin cc g3 fPIC Wall rpath

10.307 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/check_arnoldi_template_nrm.dir/check_arnoldi_template_nrm.c.o -o check_arnoldi_template_↩ nrm -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.↩ so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.307.1 Variable Documentation

10.307.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_arnoldi_template_nrm.dir/check_arnoldi_template_nrm.c.o -o check_arnoldi_template_nrm -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.307.1.2 usr bin cc g3 fPIC Wall rpath

10.308 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/check_eig.dir/check_eig.c.o -o check_eig -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.308.1 Variable Documentation

10.308.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_eig.dir/check_eig.c.o -o check_eig -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.308.1.2 usr bin cc g3 fPIC Wall rpath

10.309 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_eig2.dir/check_eig2.c.o -o check_eig2 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.309.1 Variable Documentation

10.309.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_eig2.dir/check_eig2.c.o -o check_eig2 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.309.1.2 usr bin cc g3 fPIC Wall rpath

10.310 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_eig_complex.dir/check_eig_complex.c.o -o check_eig_complex -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.310.1 Variable Documentation

10.310.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_eig_complex.dir/check_eig_complex.c.o -o check_eig_complex -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.310.1.2 usr bin cc g3 fPIC Wall rpath

10.311 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_eig_hessenberg_schur.dir/check_eig_hessenberg_schur.c.o -o check_eig_hessenberg_schur -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.311.1 Variable Documentation

10.311.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_eig_hessenberg_schur.dir/check_eig_hessenberg_schur.c.o -o check_eig_hessenberg_schur -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.311.1.2 usr bin cc g3 fPIC Wall rpath

10.312 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_eig_hessenberg_schur_largest_ev.dir/check_eig_hessenberg_schur_largest_ev.c.↩ o -o check_eig_hessenberg_schur_largest_ev -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.312.1 Variable Documentation

10.312.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_eig_hessenberg_schur_largest_ev.dir/check_eig_hessenberg_schur_largest_ev.c.o -o check_↩ eig_hessenberg_schur_largest_ev -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.312.1.2 usr bin cc g3 fPIC Wall rpath

10.313 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DND↩ EBUG CMakeFiles/check_gsign.dir/check_gsign.c.o -o check_gsign -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.313.1 Variable Documentation

10.313.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_gsign.dir/check_gsign.c.o -o check_gsign -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.313.1.2 usr bin cc g3 fPIC Wall rpath

10.314 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DND↩EBUG CMakeFiles/check_schur.dir/check_schur.c.o -o check_schur -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.314.1 Variable Documentation

10.314.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_schur.dir/check_schur.c.o -o check_schur -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.314.1.2 usr bin cc g3 fPIC Wall rpath

10.315 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩Files/check_sign.dir/check_sign.c.o -o check_sign -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.315.1 Variable Documentation

10.315.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_sign.dir/check_sign.c.o -o check_sign -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.315.1.2 usr bin cc g3 fPIC Wall rpath

10.316 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_svd.dir/check_svd.c.o -o check_svd -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.316.1 Variable Documentation

10.316.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_svd.dir/check_svd.c.o -o check_svd -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.316.1.2 usr bin cc g3 fPIC Wall rpath

10.317 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_svd_complex.dir/check_svd_complex.c.o -o check_svd_complex -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.317.1 Variable Documentation

10.317.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_svd_complex.dir/check_svd_complex.c.o -o check_svd_complex -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.317.1.2 usr bin cc g3 fPIC Wall rpath

10.318 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_svd_econ.dir/check_svd_econ.c.o -o check_svd_econ -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.318.1 Variable Documentation

10.318.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_svd_econ.dir/check_svd_econ.c.o -o check_svd_econ -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.318.1.2 usr bin cc g3 fPIC Wall rpath

10.319 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_svd_econ_complex.dir/check_svd_econ_complex.c.o -o check_svd_econ_complex -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.↩ 0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.319.1 Variable Documentation

10.319.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_svd_econ_complex.dir/check_svd_econ_complex.c.o -o check_svd_econ_complex -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.319.1.2 usr bin cc g3 fPIC Wall rpath

10.320 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_cholmod.dir/check_cholmod.c.o -o check_cholmod -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.320.1 Variable Documentation

10.320.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_cholmod.dir/check_cholmod.c.o -o check_cholmod -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.320.1.2 usr bin cc g3 fPIC Wall rpath

10.321 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/check_cimport.dir/check_cimport.c.o -o check_cimport -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.321.1 Variable Documentation

10.321.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_cimport.dir/check_cimport.c.o -o check_cimport -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.321.1.2 usr bin cc g3 fPIC Wall rpath

10.322 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩Files/check_cimport_c2d.dir/check_cimport_c2d.c.o -o check_cimport_c2d -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.322.1 Variable Documentation

10.322.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_cimport_c2d.dir/check_cimport_c2d.c.o -o check_cimport_c2d -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.322.1.2 usr bin cc g3 fPIC Wall rpath

10.323 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩ UG CMakeFiles/check_convert.dir/check_convert.c.o -o check_convert -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.323.1 Variable Documentation

10.323.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_convert.dir/check_convert.c.o -o check_convert -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.323.1.2 usr bin cc g3 fPIC Wall rpath

10.324 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DND↩ EBUG CMakeFiles/check_mvpcall_matrix.dir/check_mvpcall_matrix.c.o -o check_mvpcall_matrix -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.↩ 1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.324.1 Variable Documentation

10.324.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mvpcall_matrix.dir/check_mvpcall_matrix.c.o -o check_mvpcall_matrix -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.324.1.2 usr bin cc g3 fPIC Wall rpath

10.325 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DND↩EBUG CMakeFiles/check_perm.dir/check_perm.c.o -o check_perm -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.325.1 Variable Documentation

10.325.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_perm.dir/check_perm.c.o -o check_perm -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.325.1.2 usr bin cc g3 fPIC Wall rpath

10.326 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DND↩EBUG CMakeFiles/check_real_imag_part.dir/check_real_imag_part.c.o -o check_real_imag_part -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.↩1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.326.1 Variable Documentation

10.326.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_real_imag_part.dir/check_real_imag_part.c.o -o check_real_imag_part -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.326.1.2 usr bin cc g3 fPIC Wall rpath

10.327 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/check_fputs_block.dir/check_fputs_block.c.o -o check_fputs_block -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.327.1 Variable Documentation

10.327.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_fputs_block.dir/check_fputs_block.c.o -o check_fputs_block -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.327.1.2 usr bin cc g3 fPIC Wall rpath

10.328 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩ UG CMakeFiles/check_matrixio.dir/check_matrixio.c.o -o check_matrixio -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.328.1 Variable Documentation

10.328.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_matrixio.dir/check_matrixio.c.o -o check_matrixio -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.328.1.2 usr bin cc g3 fPIC Wall rpath

10.329 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_vectorio.dir/check_vectorio.c.o -o check_vectorio -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.329.1 Variable Documentation

10.329.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_vectorio.dir/check_vectorio.c.o -o check_vectorio -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.329.1.2 usr bin cc g3 fPIC Wall rpath

10.330 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_itsolver.dir/check_itsolver.c.o -o check_itsolver -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.330.1 Variable Documentation

10.330.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_itsolver.dir/check_itsolver.c.o -o check_itsolver -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.330.1.2 usr bin cc g3 fPIC Wall rpath

10.331 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/check_precond.dir/check_precond.c.o -o check_precond -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.331.1 Variable Documentation

10.331.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_precond.dir/check_precond.c.o -o check_precond -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.331.1.2 usr bin cc g3 fPIC Wall rpath

10.332 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩Files/check_ber_sign_fac.dir/check_ber_sign_fac.c.o -o check_ber_sign_fac -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.332.1 Variable Documentation

10.332.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_ber_sign_fac.dir/check_ber_sign_fac.c.o -o check_ber_sign_fac -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.332.1.2 usr bin cc g3 fPIC Wall rpath

10.333 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/check_dae2_handles_adi.dir/check_dae2_handles_adi.c.o -o check_dae2_handles_adi -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.↩0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.333.1 Variable Documentation

10.333.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dae2_handles_adi.dir/check_dae2_handles_adi.c.o -o check_dae2_handles_adi -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.333.1.2 usr bin cc g3 fPIC Wall rpath

10.334 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩MakeFiles/check_dae2_handles_para.dir/check_dae2_handles_para.c.o -o check_dae2_handles_para -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.↩0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.334.1 Variable Documentation

10.334.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dae2_handles_para.dir/check_dae2_handles_para.c.o -o check_dae2_handles_para -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.334.1.2 usr bin cc g3 fPIC Wall rpath

10.335 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/check_dense_nm.dir/check_dense_nm.c.o -o check_dense_nm -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.335.1 Variable Documentation

10.335.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dense_nm.dir/check_dense_nm.c.o -o check_dense_nm -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.335.1.2 usr bin cc g3 fPIC Wall rpath

10.336 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩ UG CMakeFiles/check_dense_nm_tridiag.dir/check_dense_nm_tridiag.c.o -o check_dense_nm_tridiag -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.↩ 0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.336.1 Variable Documentation

10.336.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dense_nm_tridiag.dir/check_dense_nm_tridiag.c.o -o check_dense_nm_tridiag -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.336.1.2 usr bin cc g3 fPIC Wall rpath

10.337 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/check_linesearch.dir/check_linesearch.c.o -o check_linesearch -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.337.1 Variable Documentation

10.337.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_linesearch.dir/check_linesearch.c.o -o check_linesearch -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.337.1.2 usr bin cc g3 fPIC Wall rpath

10.338 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/check_lradi_dae1.dir/check_lradi_dae1.c.o -o check_lradi_dae1 -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.338.1 Variable Documentation

10.338.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lradi_dae1.dir/check_lradi_dae1.c.o -o check_lradi_dae1 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.338.1.2 usr bin cc g3 fPIC Wall rpath

10.339 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lradi_dae2.dir/check_lradi_dae2.c.o -o check_lradi_dae2 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.339.1 Variable Documentation

10.339.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lradi_dae2.dir/check_lradi_dae2.c.o -o check_lradi_dae2 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.339.1.2 usr bin cc g3 fPIC Wall rpath

10.340 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lradi_glyap.dir/check_lradi_glyap.c.o -o check_lradi_glyap -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.340.1 Variable Documentation

10.340.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lradi_glyap.dir/check_lradi_glyap.c.o -o check_lradi_glyap -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.340.1.2 usr bin cc g3 fPIC Wall rpath

10.341 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/check_lradi_so.dir/check_lradi_so.c.o -o check_lradi_so -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.341.1 Variable Documentation

10.341.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lradi_so.dir/check_lradi_so.c.o -o check_lradi_so -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.341.1.2 usr bin cc g3 fPIC Wall rpath

10.342 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩Files/check_lradi_unsym.dir/check_lradi_unsym.c.o -o check_lradi_unsym -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.342.1 Variable Documentation

10.342.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lradi_unsym.dir/check_lradi_unsym.c.o -o check_lradi_unsym -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.342.1.2 usr bin cc g3 fPIC Wall rpath

10.343 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lrnm_dae1.dir/check_lrnm_dae1.c.o -o check_lrnm_dae1 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.343.1 Variable Documentation

10.343.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lrnm_dae1.dir/check_lrnm_dae1.c.o -o check_lrnm_dae1 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.343.1.2 usr bin cc g3 fPIC Wall rpath

10.344 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lrnm_dae2.dir/check_lrnm_dae2.c.o -o check_lrnm_dae2 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.344.1 Variable Documentation

10.344.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lrnm_dae2.dir/check_lrnm_dae2.c.o -o check_lrnm_dae2 -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.344.1.2 usr bin cc g3 fPIC Wall rpath

10.345 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/check_lrnm_g.dir/check_lrnm_g.c.o -o check_lrnm_g -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.345.1 Variable Documentation

10.345.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lrnm_g.dir/check_lrnm_g.c.o -o check_lrnm_g -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.345.1.2 usr bin cc g3 fPIC Wall rpath

10.346 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/check_lrnm_so.dir/check_lrnm_so.c.o -o check_lrnm_so -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.346.1 Variable Documentation

10.346.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lrnm_so.dir/check_lrnm_so.c.o -o check_lrnm_so -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.346.1.2 usr bin cc g3 fPIC Wall rpath

10.347 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lrnm_unsym.dir/check_lrnm_unsym.c.o -o check_lrnm_unsym -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.347.1 Variable Documentation

10.347.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lrnm_unsym.dir/check_lrnm_unsym.c.o -o check_lrnm_unsym -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.347.1.2 usr bin cc g3 fPIC Wall rpath

10.348 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lyap_sign_fac.dir/check_lyap_sign_fac.c.o -o check_lyap_sign_fac -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.348.1 Variable Documentation

10.348.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_lyap_sign_fac.dir/check_lyap_sign_fac.c.o -o check_lyap_sign_fac -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.348.1.2 usr bin cc g3 fPIC Wall rpath

10.349 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩Files/check_so1_handles.dir/check_so1_handles.c.o -o check_so1_handles -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.349.1 Variable Documentation

10.349.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_so1_handles.dir/check_so1_handles.c.o -o check_so1_handles -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.349.1.2 usr bin cc g3 fPIC Wall rpath

10.350 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩Files/check_so2_handles.dir/check_so2_handles.c.o -o check_so2_handles -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.350.1 Variable Documentation

10.350.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_so2_handles.dir/check_so2_handles.c.o -o check_so2_handles -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.350.1.2 usr bin cc g3 fPIC Wall rpath

10.351 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩Files/check_add.dir/check_add.c.o -o check_add -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.351.1 Variable Documentation

10.351.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_↩add.dir/check_add.c.o -o check_add -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.351.1.2 usr bin cc g3 fPIC Wall rpath

10.352 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩MakeFiles/check_addcol2p.dir/check_addcol2p.c.o -o check_addcol2p -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.352.1 Variable Documentation

10.352.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_addcol2p.dir/check_addcol2p.c.o -o check_addcol2p -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.352.1.2 usr bin cc g3 fPIC Wall rpath

10.353 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩Files/check_any.dir/check_any.c.o -o check_any -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.353.1 Variable Documentation

10.353.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_↩any.dir/check_any.c.o -o check_any -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.353.1.2 usr bin cc g3 fPIC Wall rpath

10.354 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩MakeFiles/check_bandwidth.dir/check_bandwidth.c.o -o check_bandwidth -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.354.1 Variable Documentation

10.354.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_bandwidth.dir/check_bandwidth.c.o -o check_bandwidth -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.354.1.2 usr bin cc g3 fPIC Wall rpath

10.355 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDE↩ BUG CMakeFiles/check_biorth.dir/check_biorth.c.o -o check_biorth -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.355.1 Variable Documentation

10.355.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_biorth.dir/check_biorth.c.o -o check_biorth -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.355.1.2 usr bin cc g3 fPIC Wall rpath

10.356 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/check_cat.dir/check_cat.c.o -o check_cat -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.356.1 Variable Documentation

10.356.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_cat.dir/check_cat.c.o -o check_cat -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.356.1.2 usr bin cc g3 fPIC Wall rpath

10.357 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_colaxpy2.dir/check_colaxpy2.c.o -o check_colaxpy2 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.357.1 Variable Documentation

10.357.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_colaxpy2.dir/check_colaxpy2.c.o -o check_colaxpy2 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.357.1.2 usr bin cc g3 fPIC Wall rpath

10.358 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_coldot.dir/check_coldot.c.o -o check_coldot -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.358.1 Variable Documentation

10.358.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_coldot.dir/check_coldot.c.o -o check_coldot -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.358.1.2 usr bin cc g3 fPIC Wall rpath

10.359 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/check_coldotc.dir/check_coldotc.c.o -o check_coldotc -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.359.1 Variable Documentation

10.359.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_coldotc.dir/check_coldotc.c.o -o check_coldotc -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.359.1.2 usr bin cc g3 fPIC Wall rpath

10.360 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_coldotcE.dir/check_coldotcE.c.o -o check_coldotcE -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.360.1 Variable Documentation

10.360.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_coldotcE.dir/check_coldotcE.c.o -o check_coldotcE -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.360.1.2 usr bin cc g3 fPIC Wall rpath

10.361 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩ UG CMakeFiles/check_coldotE.dir/check_coldotE.c.o -o check_coldotE -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.361.1 Variable Documentation

10.361.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_coldotE.dir/check_coldotE.c.o -o check_coldotE -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.361.1.2 usr bin cc g3 fPIC Wall rpath

10.362 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩ UG CMakeFiles/check_colnorm.dir/check_colnorm.c.o -o check_colnorm -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.362.1 Variable Documentation

10.362.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_colnorm.dir/check_colnorm.c.o -o check_colnorm -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.362.1.2 usr bin cc g3 fPIC Wall rpath

10.363 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/check_colscale.dir/check_colscale.c.o -o check_colscale -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.363.1 Variable Documentation

10.363.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_colscale.dir/check_colscale.c.o -o check_colscale -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.363.1.2 usr bin cc g3 fPIC Wall rpath

10.364 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/check_colsub.dir/check_colsub.c.o -o check_colsub -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.364.1 Variable Documentation

10.364.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_colsub.dir/check_colsub.c.o -o check_colsub -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.364.1.2 usr bin cc g3 fPIC Wall rpath

10.365 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_colvecaxpy.dir/check_colvecaxpy.c.o -o check_colvecaxpy -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.365.1 Variable Documentation

10.365.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_colvecaxpy.dir/check_colvecaxpy.c.o -o check_colvecaxpy -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.365.1.2 usr bin cc g3 fPIC Wall rpath

10.366 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_colvecdot.dir/check_colvecdot.c.o -o check_colvecdot -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.366.1 Variable Documentation

10.366.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_colvecdot.dir/check_colvecdot.c.o -o check_colvecdot -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.366.1.2 usr bin cc g3 fPIC Wall rpath

10.367 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩MakeFiles/check_colvecdotc.dir/check_colvecdotc.c.o -o check_colvecdotc -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.367.1 Variable Documentation

10.367.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_colvecdotc.dir/check_colvecdotc.c.o -o check_colvecdotc -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.367.1.2 usr bin cc g3 fPIC Wall rpath

10.368 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/check_condest.dir/check_condest.c.o -o check_condest -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.368.1 Variable Documentation

10.368.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_condest.dir/check_condest.c.o -o check_condest -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.368.1.2 usr bin cc g3 fPIC Wall rpath

10.369 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩ UG CMakeFiles/check_decomp.dir/check_decomp.c.o -o check_decomp -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.369.1 Variable Documentation

10.369.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_decomp.dir/check_decomp.c.o -o check_decomp -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.369.1.2 usr bin cc g3 fPIC Wall rpath

10.370 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DN↩ DEBUG CMakeFiles/check_diag.dir/check_diag.c.o -o check_diag -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.370.1 Variable Documentation

10.370.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_diag.dir/check_diag.c.o -o check_diag -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.370.1.2 usr bin cc g3 fPIC Wall rpath

10.371 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dynorm2.dir/check_dynorm2.c.o -o check_dynorm2 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.371.1 Variable Documentation

10.371.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dynorm2.dir/check_dynorm2.c.o -o check_dynorm2 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.371.1.2 usr bin cc g3 fPIC Wall rpath

10.372 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dynorm2t.dir/check_dynorm2t.c.o -o check_dynorm2t -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.372.1 Variable Documentation

10.372.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dynorm2t.dir/check_dynorm2t.c.o -o check_dynorm2t -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.372.1.2 usr bin cc g3 fPIC Wall rpath

10.373 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dynorm_indefinite2.dir/check_dynorm_indefinite2.c.o -o check_dynorm_indefinite2 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.↩ 0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.373.1 Variable Documentation

10.373.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dynorm_indefinite2.dir/check_dynorm_indefinite2.c.o -o check_dynorm_indefinite2 -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.373.1.2 usr bin cc g3 fPIC Wall rpath

10.374 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩ UG CMakeFiles/check_dynorm_indefinitef.dir/check_dynorm_indefinitef.c.o -o check_dynorm_indefinitef -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.↩ 0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.374.1 Variable Documentation

10.374.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dynorm_indefinitef.dir/check_dynorm_indefinitef.c.o -o check_dynorm_indefinitef -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.374.1.2 usr bin cc g3 fPIC Wall rpath

10.375 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩ UG CMakeFiles/check_dynormf.dir/check_dynormf.c.o -o check_dynormf -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.375.1 Variable Documentation

10.375.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dynormf.dir/check_dynormf.c.o -o check_dynormf -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.375.1.2 usr bin cc g3 fPIC Wall rpath

10.376 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dynormft.dir/check_dynormft.c.o -o check_dynormft -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.376.1 Variable Documentation

10.376.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_dynormft.dir/check_dynormft.c.o -o check_dynormft -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.376.1.2 usr bin cc g3 fPIC Wall rpath

10.377 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DND↩ EBUG CMakeFiles/check_eliminate_zeros.dir/check_eliminate_zeros.c.o -o check_eliminate_zeros -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.↩ 1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.377.1 Variable Documentation

10.377.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_eliminate_zeros.dir/check_eliminate_zeros.c.o -o check_eliminate_zeros -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.377.1.2 usr bin cc g3 fPIC Wall rpath

10.378 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/check_eps.dir/check_eps.c.o -o check_eps -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.378.1 Variable Documentation

10.378.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_eps.dir/check_eps.c.o -o check_eps -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.378.1.2 usr bin cc g3 fPIC Wall rpath

10.379 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_fro_inn.dir/check_fro_inn.c.o -o check_fro_inn -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.379.1 Variable Documentation

10.379.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_fro_inn.dir/check_fro_inn.c.o -o check_fro_inn -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.379.1.2 usr bin cc g3 fPIC Wall rpath

10.380 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_gaxpy.dir/check_gaxpy.c.o -o check_gaxpy -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.380.1 Variable Documentation

10.380.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_gaxpy.dir/check_gaxpy.c.o -o check_gaxpy -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.380.1.2 usr bin cc g3 fPIC Wall rpath

10.381 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/check_getelement.dir/check_getelement.c.o -o check_getelement -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.381.1 Variable Documentation

10.381.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_getelement.dir/check_getelement.c.o -o check_getelement -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.381.1.2 usr bin cc g3 fPIC Wall rpath

10.382 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/check_matrix_biorth.dir/check_matrix_biorth.c.o -o check_matrix_biorth -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.382.1 Variable Documentation

10.382.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_matrix_biorth.dir/check_matrix_biorth.c.o -o check_matrix_biorth -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.382.1.2 usr bin cc g3 fPIC Wall rpath

10.383 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DN↩DEBUG CMakeFiles/check_matrix_colaxpy.dir/check_matrix_colaxpy.c.o -o check_matrix_colaxpy -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.↩1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.383.1 Variable Documentation

10.383.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_matrix_colaxpy.dir/check_matrix_colaxpy.c.o -o check_matrix_colaxpy -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.383.1.2 usr bin cc g3 fPIC Wall rpath

10.384 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DN↩DEBUG CMakeFiles/check_matrix_gbiorth.dir/check_matrix_gbiorth.c.o -o check_matrix_gbiorth -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.↩1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.384.1 Variable Documentation

10.384.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_matrix_gbiorth.dir/check_matrix_gbiorth.c.o -o check_matrix_gbiorth -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.384.1.2 usr bin cc g3 fPIC Wall rpath

10.385 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/check_matrix_null.dir/check_matrix_null.c.o -o check_matrix_null -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.385.1 Variable Documentation

10.385.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_matrix_null.dir/check_matrix_null.c.o -o check_matrix_null -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.385.1.2 usr bin cc g3 fPIC Wall rpath

10.386 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/check_matrix_sum.dir/check_matrix_sum.c.o -o check_matrix_sum -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.386.1 Variable Documentation

10.386.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_matrix_sum.dir/check_matrix_sum.c.o -o check_matrix_sum -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.386.1.2 usr bin cc g3 fPIC Wall rpath

10.387 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mgs.dir/check_mgs.c.o -o check_mgs -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.387.1 Variable Documentation

10.387.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mgs.dir/check_mgs.c.o -o check_mgs -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.387.1.2 usr bin cc g3 fPIC Wall rpath

10.388 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mgs_inplace.dir/check_mgs_inplace.c.o -o check_mgs_inplace -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.388.1 Variable Documentation

10.388.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mgs_inplace.dir/check_mgs_inplace.c.o -o check_mgs_inplace -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.388.1.2 usr bin cc g3 fPIC Wall rpath

10.389 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/check_mgsadd.dir/check_mgsadd.c.o -o check_mgsadd -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.389.1 Variable Documentation

10.389.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mgsadd.dir/check_mgsadd.c.o -o check_mgsadd -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.389.1.2 usr bin cc g3 fPIC Wall rpath

10.390 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩Files/check_mul.dir/check_mul.c.o -o check_mul -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.390.1 Variable Documentation

10.390.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mul.dir/check_mul.c.o -o check_mul -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.390.1.2 usr bin cc g3 fPIC Wall rpath

10.391 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mul_small.dir/check_mul_small.c.o -o check_mul_small -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.391.1 Variable Documentation

10.391.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mul_small.dir/check_mul_small.c.o -o check_mul_small -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.391.1.2 usr bin cc g3 fPIC Wall rpath

10.392 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mul_tall.dir/check_mul_tall.c.o -o check_mul_tall -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.392.1 Variable Documentation

10.392.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mul_tall.dir/check_mul_tall.c.o -o check_mul_tall -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.392.1.2 usr bin cc g3 fPIC Wall rpath

10.393 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/check_mulnormf2.dir/check_mulnormf2.c.o -o check_mulnormf2 -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.393.1 Variable Documentation

10.393.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mulnormf2.dir/check_mulnormf2.c.o -o check_mulnormf2 -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.393.1.2 usr bin cc g3 fPIC Wall rpath

10.394 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DN↩ DEBUG CMakeFiles/check_mvp.dir/check_mvp.c.o -o check_mvp -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.394.1 Variable Documentation

10.394.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_mvp.dir/check_mvp.c.o -o check_mvp -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.394.1.2 usr bin cc g3 fPIC Wall rpath

10.395 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_norm.dir/check_norm.c.o -o check_norm -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.395.1 Variable Documentation

10.395.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_norm.dir/check_norm.c.o -o check_norm -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.395.1.2 usr bin cc g3 fPIC Wall rpath

10.396 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_one_value.dir/check_one_value.c.o -o check_one_value -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.396.1 Variable Documentation

10.396.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_one_value.dir/check_one_value.c.o -o check_one_value -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.396.1.2 usr bin cc g3 fPIC Wall rpath

10.397 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_orth.dir/check_orth.c.o -o check_orth -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.397.1 Variable Documentation

10.397.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_orth.dir/check_orth.c.o -o check_orth -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.397.1.2 usr bin cc g3 fPIC Wall rpath

10.398 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_proj_sym.dir/check_proj_sym.c.o -o check_proj_sym -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.398.1 Variable Documentation

10.398.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_proj_sym.dir/check_proj_sym.c.o -o check_proj_sym -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.398.1.2 usr bin cc g3 fPIC Wall rpath

10.399 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_rank.dir/check_rank.c.o -o check_rank -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.399.1 Variable Documentation

10.399.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_rank.dir/check_rank.c.o -o check_rank -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.399.1.2 usr bin cc g3 fPIC Wall rpath

10.400 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_rowsub.dir/check_rowsub.c.o -o check_rowsub -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.400.1 Variable Documentation

10.400.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_rowsub.dir/check_rowsub.c.o -o check_rowsub -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.400.1.2 usr bin cc g3 fPIC Wall rpath

10.401 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩ UG CMakeFiles/check_scalem.dir/check_scalem.c.o -o check_scalem -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.401.1 Variable Documentation

10.401.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_scalem.dir/check_scalem.c.o -o check_scalem -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.401.1.2 usr bin cc g3 fPIC Wall rpath

10.402 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDE↩ BUG CMakeFiles/check_set_get_col_row.dir/check_set_get_col_row.c.o -o check_set_get_col_row -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.↩ 1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.402.1 Variable Documentation

10.402.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_set_get_col_row.dir/check_set_get_col_row.c.o -o check_set_get_col_row -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.402.1.2 usr bin cc g3 fPIC Wall rpath

10.403 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/check_setelement.dir/check_setelement.c.o -o check_setelement -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.403.1 Variable Documentation

10.403.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_setelement.dir/check_setelement.c.o -o check_setelement -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.403.1.2 usr bin cc g3 fPIC Wall rpath

10.404 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/check_sub.dir/check_sub.c.o -o check_sub -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.404.1 Variable Documentation

10.404.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_sub.dir/check_sub.c.o -o check_sub -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.404.1.2 usr bin cc g3 fPIC Wall rpath

10.405 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_trace.dir/check_trace.c.o -o check_trace -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.405.1 Variable Documentation

10.405.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_trace.dir/check_trace.c.o -o check_trace -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.405.1.2 usr bin cc g3 fPIC Wall rpath

10.406 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_transpose.dir/check_transpose.c.o -o check_transpose -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.406.1 Variable Documentation

10.406.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_transpose.dir/check_transpose.c.o -o check_transpose -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.406.1.2 usr bin cc g3 fPIC Wall rpath

10.407 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_tridiag.dir/check_tridiag.c.o -o check_tridiag -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.407.1 Variable Documentation

10.407.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_tridiag.dir/check_tridiag.c.o -o check_tridiag -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.407.1.2 usr bin cc g3 fPIC Wall rpath

10.408 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩ UG CMakeFiles/check_freelist.dir/check_freelist.c.o -o check_freelist -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.408.1 Variable Documentation

10.408.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/check_freelist.dir/check_freelist.c.o -o check_freelist -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.408.1.2 usr bin cc g3 fPIC Wall rpath

10.409 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/tutorial_bt.dir/tutorial_bt.c.o -o tutorial_bt -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.409.1 Variable Documentation

10.409.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial↩ _bt.dir/tutorial_bt.c.o -o tutorial_bt -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.409.1.2 usr bin cc g3 fPIC Wall rpath

10.410 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/tutorial_bt_gglrsrm.dir/tutorial_bt_gglrsrm.c.o -o tutorial_bt_gglrsrm -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.410.1 Variable Documentation

10.410.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_bt_gglrsrm.dir/tutorial_bt_gglrsrm.c.o -o tutorial_bt_gglrsrm -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.410.1.2 usr bin cc g3 fPIC Wall rpath

10.411 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/tutorial_bt_gslrsrm.dir/tutorial_bt_gslrsrm.c.o -o tutorial_bt_gslrsrm -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.411.1 Variable Documentation

10.411.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_bt_gslrsrm.dir/tutorial_bt_gslrsrm.c.o -o tutorial_bt_gslrsrm -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.411.1.2 usr bin cc g3 fPIC Wall rpath

10.412 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/tutorial_bt_lrsrm.dir/tutorial_bt_lrsrm.c.o -o tutorial_bt_lrsrm -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.412.1 Variable Documentation

10.412.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_bt_lrsrm.dir/tutorial_bt_lrsrm.c.o -o tutorial_bt_lrsrm -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.412.1.2 usr bin cc g3 fPIC Wall rpath

10.413 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDE↩ BUG CMakeFiles/tutorial_bt_lrsrm_somor.dir/tutorial_bt_lrsrm_somor.c.o -o tutorial_bt_lrsrm_somor -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.↩ 1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.413.1 Variable Documentation

10.413.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_bt_lrsrm_somor.dir/tutorial_bt_lrsrm_somor.c.o -o tutorial_bt_lrsrm_somor -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.413.1.2 usr bin cc g3 fPIC Wall rpath

10.414 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/tutorial_bt_lrsrm_somor_so.dir/tutorial_bt_lrsrm_somor_so.c.o -o tutorial_bt_lrsrm_somor_so -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.↩ 0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.414.1 Variable Documentation

10.414.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_bt_lrsrm_somor_so.dir/tutorial_bt_lrsrm_somor_so.c.o -o tutorial_bt_lrsrm_somor_so -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.414.1.2 usr bin cc g3 fPIC Wall rpath

10.415 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/tutorial_backslash.dir/tutorial_backslash.c.o -o tutorial_backslash -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.415.1 Variable Documentation

10.415.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_backslash.dir/tutorial_backslash.c.o -o tutorial_backslash -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.415.1.2 usr bin cc g3 fPIC Wall rpath

10.416 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_direct_solvers.dir/tutorial_direct_solvers.c.o -o tutorial_direct_solvers -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.416.1 Variable Documentation

10.416.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_direct_solvers.dir/tutorial_direct_solvers.c.o -o tutorial_direct_solvers -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.416.1.2 usr bin cc g3 fPIC Wall rpath

10.417 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_qr.dir/tutorial_qr.c.o -o tutorial_qr -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.417.1 Variable Documentation

10.417.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_qr.dir/tutorial_qr.c.o -o tutorial_qr -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.417.1.2 usr bin cc g3 fPIC Wall rpath

10.418 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_sylvester.dir/tutorial_sylvester.c.o -o tutorial_sylvester -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.418.1 Variable Documentation

10.418.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_sylvester.dir/tutorial_sylvester.c.o -o tutorial_sylvester -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.418.1.2 usr bin cc g3 fPIC Wall rpath

10.419 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_eig.dir/tutorial_eig.c.o -o tutorial_eig -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.419.1 Variable Documentation

10.419.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_eig.dir/tutorial_eig.c.o -o tutorial_eig -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.419.1.2 usr bin cc g3 fPIC Wall rpath

10.420 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_eigs.dir/tutorial_eigs.c.o -o tutorial_eigs -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.420.1 Variable Documentation

10.420.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_eigs.dir/tutorial_eigs.c.o -o tutorial_eigs -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.420.1.2 usr bin cc g3 fPIC Wall rpath

10.421 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_gschur.dir/tutorial_gschur.c.o -o tutorial_gschur -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.421.1 Variable Documentation

10.421.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_gschur.dir/tutorial_gschur.c.o -o tutorial_gschur -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.421.1.2 usr bin cc g3 fPIC Wall rpath

10.422 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_schur.dir/tutorial_schur.c.o -o tutorial_schur -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.422.1 Variable Documentation

10.422.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_schur.dir/tutorial_schur.c.o -o tutorial_schur -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.422.1.2 usr bin cc g3 fPIC Wall rpath

10.423 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DND↩ EBUG CMakeFiles/tutorial_sign.dir/tutorial_sign.c.o -o tutorial_sign -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.423.1 Variable Documentation

10.423.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_↩ sign.dir/tutorial_sign.c.o -o tutorial_sign -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.423.1.2 usr bin cc g3 fPIC Wall rpath

10.424 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_h2norm.dir/tutorial_h2norm.c.o -o tutorial_h2norm -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.424.1 Variable Documentation

10.424.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_h2norm.dir/tutorial_h2norm.c.o -o tutorial_h2norm -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.424.1.2 usr bin cc g3 fPIC Wall rpath

10.425 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_irka_biorth.dir/tutorial_irka_biorth.c.o -o tutorial_irka_biorth -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.425.1 Variable Documentation

10.425.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_irka_biorth.dir/tutorial_irka_biorth.c.o -o tutorial_irka_biorth -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.425.1.2 usr bin cc g3 fPIC Wall rpath

10.426 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_tsia.dir/tutorial_tsia.c.o -o tutorial_tsia -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.426.1 Variable Documentation

10.426.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial↩ _tsia.dir/tutorial_tsia.c.o -o tutorial_tsia -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.426.1.2 usr bin cc g3 fPIC Wall rpath

10.427 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩ Files/tutorial_vector_read.dir/tutorial_vector_read.c.o -o tutorial_vector_read -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.427.1 Variable Documentation

10.427.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_vector_read.dir/tutorial_vector_read.c.o -o tutorial_vector_read -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.427.1.2 usr bin cc g3 fPIC Wall rpath

10.428 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG C↩ MakeFiles/tutorial_it_unsym.dir/tutorial_it_unsym.c.o -o tutorial_it_unsym -L/home/koehlerm/mechthild↩ _home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.428.1 Variable Documentation

10.428.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_it_unsym.dir/tutorial_it_unsym.c.o -o tutorial_it_unsym -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.428.1.2 usr bin cc g3 fPIC Wall rpath

10.429 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DND↩EBUG CMakeFiles/tutorial_care.dir/tutorial_care.c.o -o tutorial_care -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.429.1 Variable Documentation

10.429.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_↩care.dir/tutorial_care.c.o -o tutorial_care -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.429.1.2 usr bin cc g3 fPIC Wall rpath

10.430 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DND↩EBUG CMakeFiles/tutorial_lradi.dir/tutorial_lradi.c.o -o tutorial_lradi -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.430.1 Variable Documentation

10.430.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_↩lradi.dir/tutorial_lradi.c.o -o tutorial_lradi -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.430.1.2 usr bin cc g3 fPIC Wall rpath

10.431 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/tutorial_lrnm-g.dir/tutorial_lrnm-g.c.o -o tutorial_lrnm-g -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.431.1 Variable Documentation

10.431.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_lrnm-g.dir/tutorial_lrnm-g.c.o -o tutorial_lrnm-g -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.431.1.2 usr bin cc g3 fPIC Wall rpath

10.432 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_lrnm.dir/tutorial_lrnm.c.o -o tutorial_lrnm -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.432.1 Variable Documentation

10.432.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_lrnm.dir/tutorial_lrnm.c.o -o tutorial_lrnm -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.432.1.2 usr bin cc g3 fPIC Wall rpath

10.433 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_lyap.dir/tutorial_lyap.c.o -o tutorial_lyap -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.433.1 Variable Documentation

10.433.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_lyap.dir/tutorial_lyap.c.o -o tutorial_lyap -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.433.1.2 usr bin cc g3 fPIC Wall rpath

10.434 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_gen_fdm_cd.dir/tutorial_gen_fdm_cd.c.o -o tutorial_gen_fdm_cd -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.434.1 Variable Documentation

10.434.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_gen_fdm_cd.dir/tutorial_gen_fdm_cd.c.o -o tutorial_gen_fdm_cd -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.434.1.2 usr bin cc g3 fPIC Wall rpath

10.435 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMake↩Files/tutorial_gen_laplace.dir/tutorial_gen_laplace.c.o -o tutorial_gen_laplace -L/home/koehlerm/mechthild↩_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.435.1 Variable Documentation

10.435.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_gen_laplace.dir/tutorial_gen_laplace.c.o -o tutorial_gen_laplace -L/home/koehlerm/mechthild_↩home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.435.1.2 usr bin cc g3 fPIC Wall rpath

10.436 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩UG CMakeFiles/tutorial_matrix_operations.dir/tutorial_matrix_operations.c.o -o tutorial_matrix_operations -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.↩0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.436.1 Variable Documentation

10.436.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_matrix_operations.dir/tutorial_matrix_operations.c.o -o tutorial_matrix_operations -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.436.1.2 usr bin cc g3 fPIC Wall rpath

10.437 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEB↩ UG CMakeFiles/tutorial_norms.dir/tutorial_norms.c.o -o tutorial_norms -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.437.1 Variable Documentation

10.437.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial_norms.dir/tutorial_norms.c.o -o tutorial_norms -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.437.1.2 usr bin cc g3 fPIC Wall rpath

10.438 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DN↩ DEBUG CMakeFiles/tutorial_plot.dir/tutorial_plot.c.o -o tutorial_plot -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.438.1 Variable Documentation

10.438.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial↩ _plot.dir/tutorial_plot.c.o -o tutorial_plot -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.438.1.2 usr bin cc g3 fPIC Wall rpath

10.439 link.txt File Reference

Variables

- usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DN↩ DEBUG CMakeFiles/tutorial_spy.dir/tutorial_spy.c.o -o tutorial_spy -L/home/koehlerm/mechthild_↩ home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl
- usr bin cc g3 fPIC Wall rpath

10.439.1 Variable Documentation

10.439.1.1 usr bin cc g3 fPIC Wall D_FILE_OFFSET_BITS =64 -D_LARGEFILE64_SOURCE -O2 -DNDEBUG CMakeFiles/tutorial↩ _spy.dir/tutorial_spy.c.o -o tutorial_spy -L/home/koehlerm/mechthild_home/work/software/cmess-releases/build/lib -rdynamic ../../lib/libmess.so.1.0.0 -lm -lpthread -lgfortran -llapack -lblas -lblas -lamd -lcolamd -lumfpack -lcholmod -lcxsparse ../../libcscutils/lib/libcscutils.a -llzma -lbz2 -lz -lm -lpthread -lgfortran -lm -lquadmath -lm -Wl

10.439.1.2 usr bin cc g3 fPIC Wall rpath

10.440 linsolve.c File Reference

Functions

- int select_solver (int i, mess_matrix A, mess_direct sol)

  Select a solver.
- int testmat (int i, mess_matrix A)

  Test solving a linear system of equations.
- int main (int argc, char $**$argv)

10.440.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This function checks the mess_direct_solve function defined in direct.c for a linear system of equations that means it checks if the solution of

$$op(A)x = b,$$

is computed correctly for a given matrix A and a given vector b.
Operations $op(.)$ on matrix A can be

- MESS_OP_NONE ($op(A) = A$),
- MESS_OP_TRANSPOSE ($op(A) = A^T$),
- MESS_OP_HERMITIAN ($op(A) = A^H$).

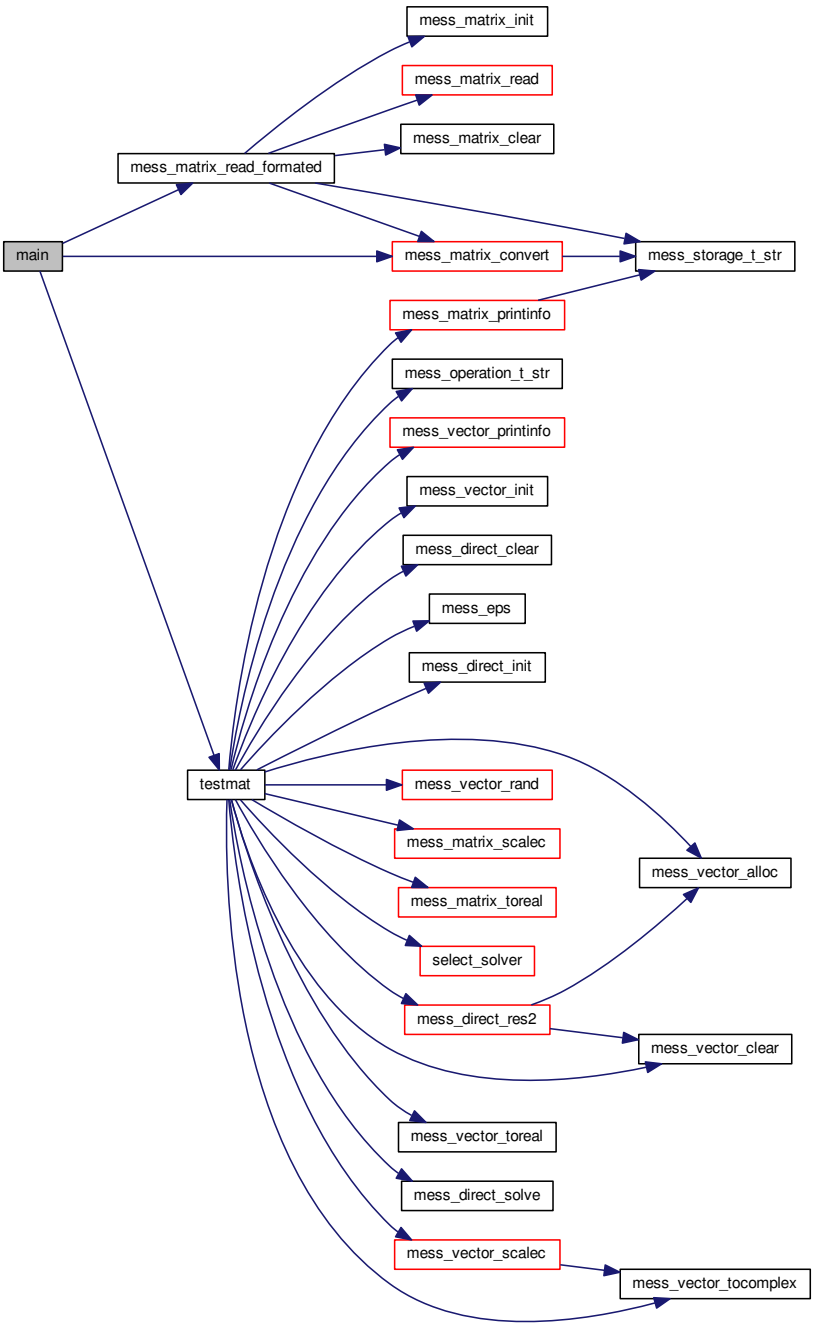
The solver can be selected by the third argument for the second input:

Input	Description
0	mess_direct_create_sparse_lu
1	mess_direct_create_csparse_lu
2	mess_direct_create_umfpack
3	mess_direct_create_bicgstab
4	mess_direct_create_lapack_lu
5	mess_direct_create_lapack_qr
6	mess_direct_create_cholesky
7	mess_direct_create_csparse_cholesky
8	mess_direct_create_cholmod_cholesky
9	mess_direct_create_banded
10	mess_direct_create_superlu
11	mess_direct_create_mklpardiso

10.440.2 Function Documentation

10.440.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.440.2.2 int select_solver (int *i*, mess_matrix *A*, mess_direct *sol*)

Parameters

in	*i*	input solver to choose
in	*A*	input matrix
out	*sol*	generated solver

Returns

zero on success or a non-zero error value otherwise

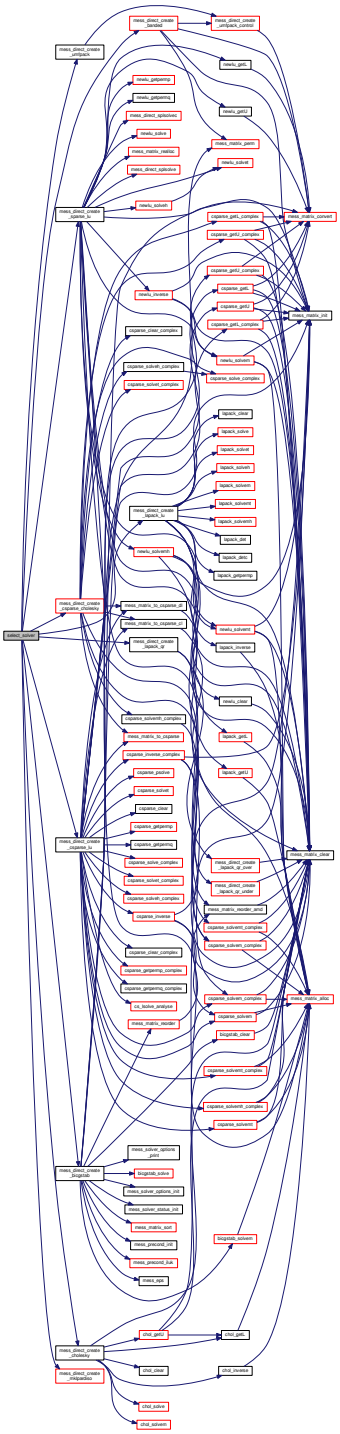
The select_solver function generates a solver for a linear system of equations

$$Ax = b$$

Solvers are choosen with i. Possible values are:

Input	Description
0	mess_direct_create_sparse_lu
1	mess_direct_create_csparse_lu
2	mess_direct_create_umfpack
3	mess_direct_create_bicgstab
4	mess_direct_create_lapack_lu
5	mess_direct_create_lapack_qr
6	mess_direct_create_cholesky
7	mess_direct_create_csparse_cholesky
8	mess_direct_create_cholmod_cholesky
9	mess_direct_create_banded
10	mess_direct_create_superlu
11	mess_direct_create_mklpardiso

Here is the call graph for this function:



10.440.2.3 int testmat (int *i*, mess_matrix *A*)

Parameters

in	*i*	input solver to choose
in	*A*	input matrix

Returns

zero on success or a non-zero error value otherwise

The testmat function solves a linear system of equations

$$op(A)x = b$$

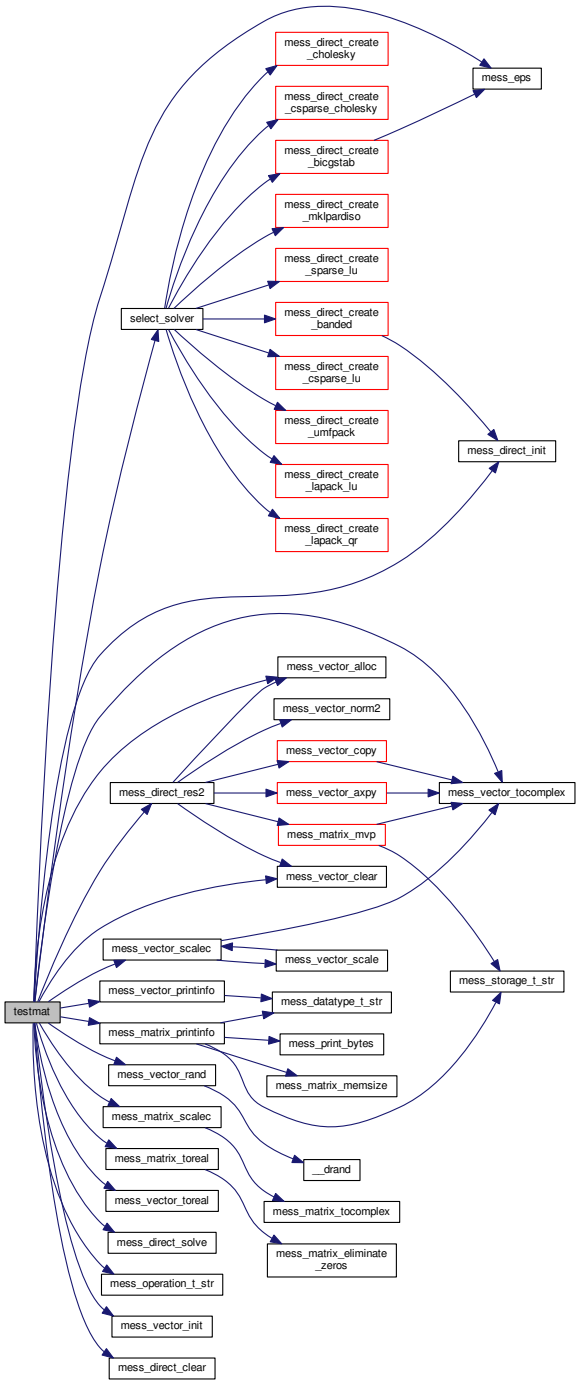
and checks if the solution is computed correctly.
Operations $op(.)$ on matrix A can be

- MESS_OP_NONE ($op(A) = A$),
- MESS_OP_TRANSPOSE ($op(A) = A^T$),
- MESS_OP_HERMITIAN ($op(A) = A^H$).

Parameter i is only used to choose the solver. Possible values are:

Input	Description
0	mess_direct_create_sparse_lu
1	mess_direct_create_csparse_lu
2	mess_direct_create_umfpack
3	mess_direct_create_bicgstab
4	mess_direct_create_lapack_lu
5	mess_direct_create_lapack_qr
6	mess_direct_create_cholesky
7	mess_direct_create_csparse_cholesky
8	mess_direct_create_cholmod_cholesky
9	mess_direct_create_banded
10	mess_direct_create_superlu
11	mess_direct_pardiso

Here is the call graph for this function:



10.441 linsolve_multi.c File Reference

Functions

- int select_solver (int i, mess_matrix A, mess_vector shifts, mess_multidirect sol)

  Select a solver.

- int testmat (int i, int cpx, mess_matrix A)

  Test solving a shifted linear system of equations.

- int main (int argc, char **argv)

10.441.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

- MESS_OP_NONE ($op(A+pI) = A+pI$),
- MESS_OP_TRANSPOSE ($op(A+pI) = A^T+pI$),
- MESS_OP_HERMITIAN ($op(A) = A^H+pI$).

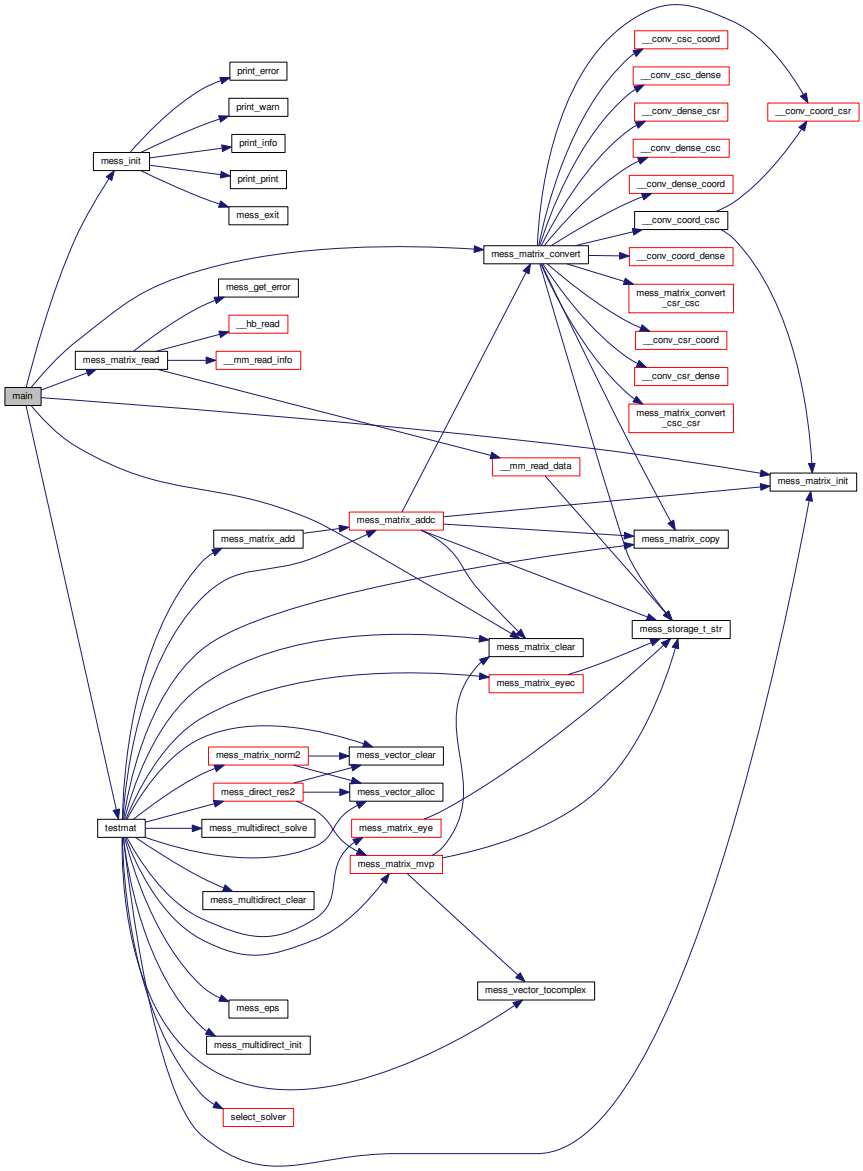
The solver can be selected by the third argument for the second input:

- 0 - mess_multidirect_create_sparse_lu solver defined in multilu_advanced.c,
- 1 - mess_multidirect_create_umfpack solver defined in multilu_umfpack.c,

10.441.2 Function Documentation

10.441.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.441.2.2 int select_solver (int *i*, mess_matrix *A*, mess_vector *shifts*, mess_multidirect *sol*)

Parameters

in	*i*	input solver to choose
in	*A*	input matrix
in	*shifts*	input shifts
out	*sol*	generated solver

Returns

zero on success or a non-zero error value otherwise

The select_solver function generates a solver for a shifted linear system of equations

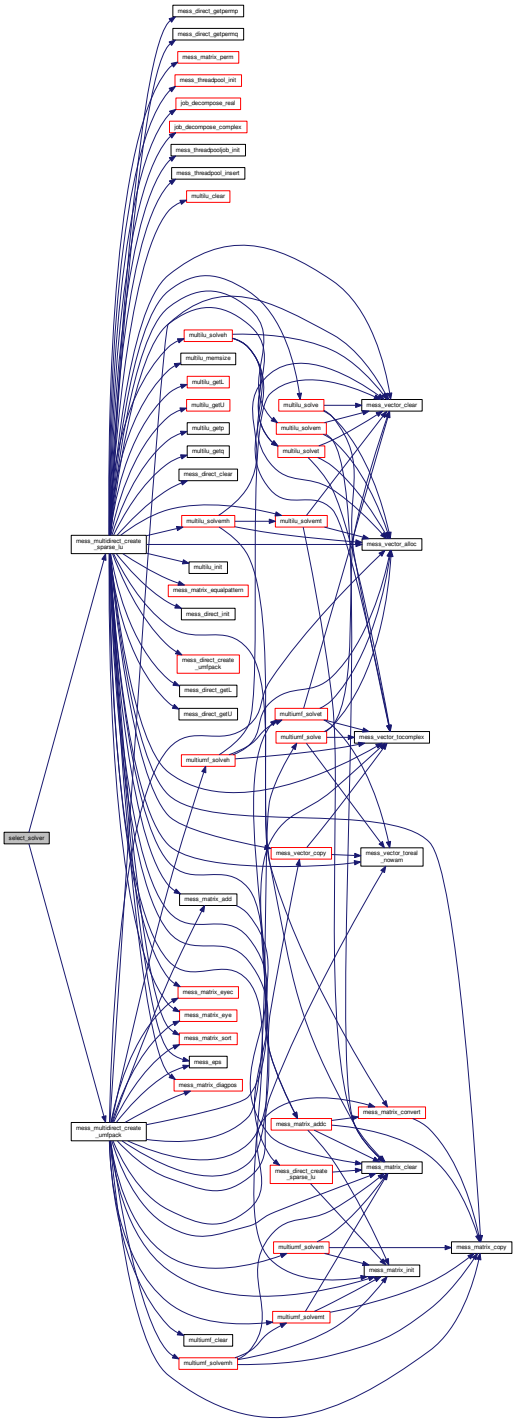
$$(A + shiftsI)x = b$$

Solvers are choosen with i. Possible values are:

- 0 - mess_multidirect_create_sparse_lu,

- 1 - mess_multidirect_create_umfpack,

Here is the call graph for this function:



10.441.2.3 int testmat (int *i*, int *cpx*, mess_matrix *A*)

Parameters

in	*i*	input solver to choose
in	*cpx*	input true if complex values are used
in	*A*	input matrix

Returns

zero on success or a non-zero error value otherwise

The testmat function solves a shifted linear system of equations

$$op(A + pI)x = b$$

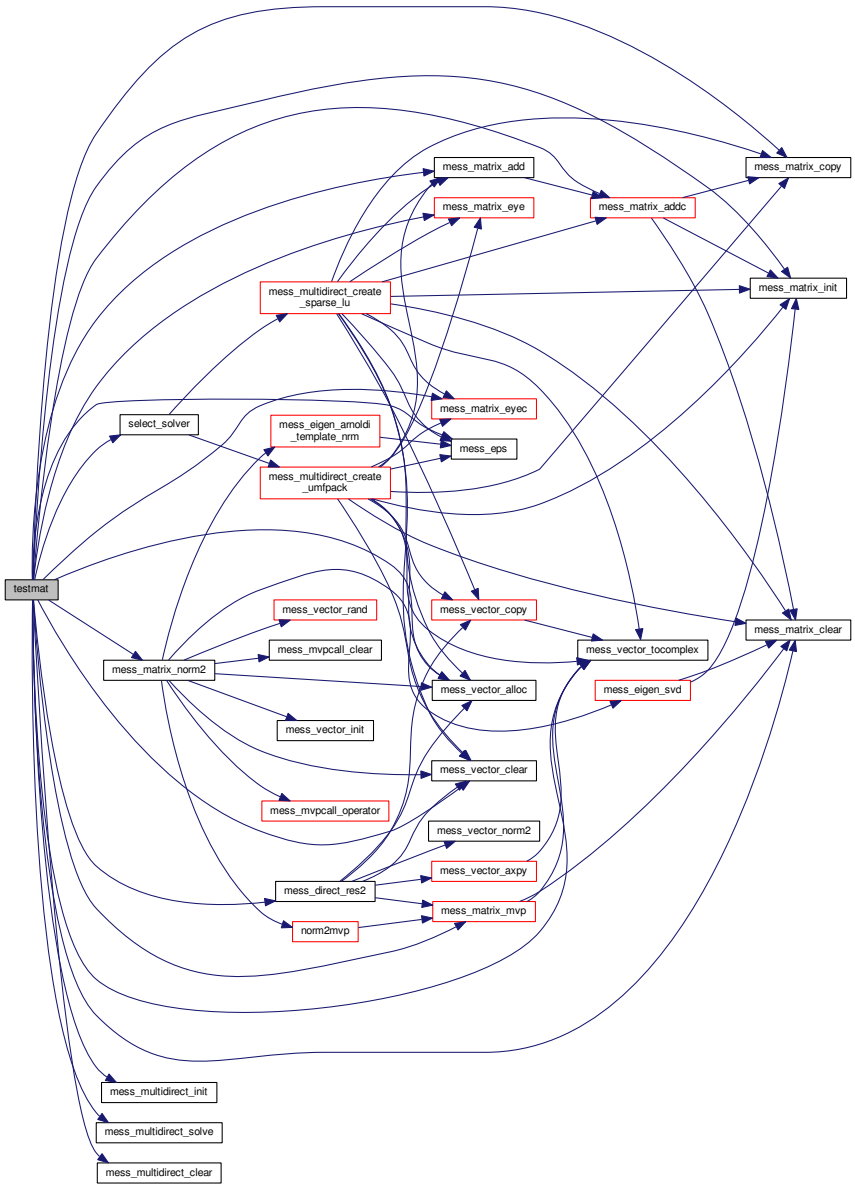
and checks if the solution is computed correctly.
Operations $op(.)$ on matrix $A + pI$ can be

- MESS_OP_NONE ($op(A + pI) = A + pI$),
- MESS_OP_TRANSPOSE ($op(A + pI) = A^T + pI$),
- MESS_OP_HERMITIAN ($op(A) = A^H + pI$).

Parameter i is only used to choose the solver. Possible values are:

- 0 - mess_multidirect_create_sparse_lu,
- 1 - mess_multidirect_create_umfpack,

Here is the call graph for this function:



10.442 linsolvem.c File Reference

Functions

- int select_solver (int i, mess_matrix A, mess_direct sol)

  Select a solver.
- int testmat (int i, mess_matrix A)

  Test solving a linear system of equations (matrix version).
- int main (int argc, char ∗∗argv)

10.442.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

- MESS_OP_NONE ($op(A) = A$),
- MESS_OP_TRANSPOSE ($op(A) = A^T$),
- MESS_OP_HERMITIAN ($op(A) = A^H$).

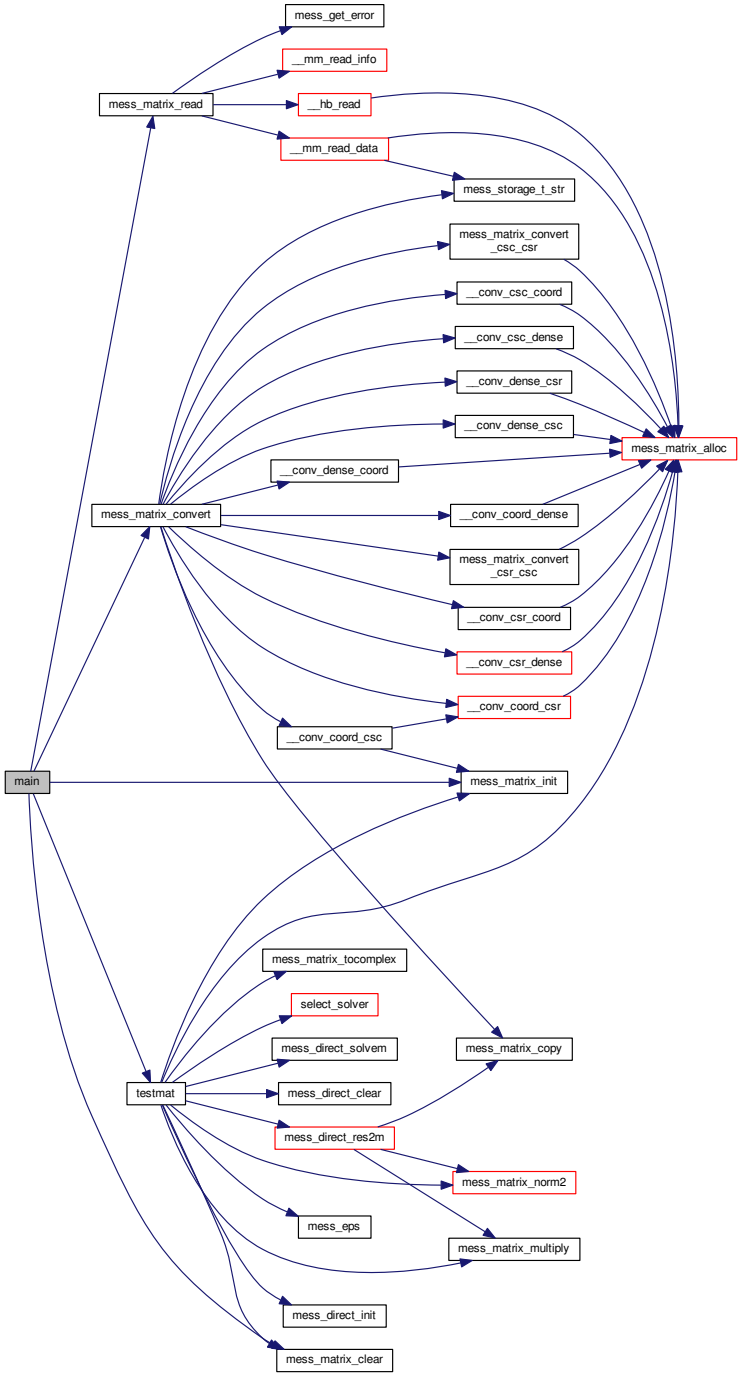
The solver can be selected by the third argument for the second input:

Input	Description
0	mess_direct_create_sparse_lu defined in newlu.c
1	mess_direct_create_csparse_lu defined in csparse.c
2	mess_direct_create_umfpack defined in umfpack.c
3	mess_direct_create_bicgstab defined in bicgstab.c
4	mess_direct_create_lapack_lu defined in lapack.c
5	mess_direct_create_lapack_qr defined in lapackqr.c
6	mess_direct_create_cholesky defined in cholesky.c
7	mess_direct_create_csparse_cholesky defined in csparse_chol.c
8	mess_direct_create_cholmod_cholesky defined in cholmod_chol.c
9	mess_direct_create_banded defined in banded.c
10	mess_direct_create_superlu defined in superlu.c
11	mess_direct_create_mklpardiso defined in mklpardiso.c

10.442.2 Function Documentation

10.442.2.1 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.442.2.2 int select_solver (int *i*, mess_matrix *A*, mess_direct *sol*)

Parameters

in	*i*	input solver to choose

Parameters

in	*A*	input matrix
out	*sol*	generated solver

Returns

zero on success or a non-zero error value otherwise

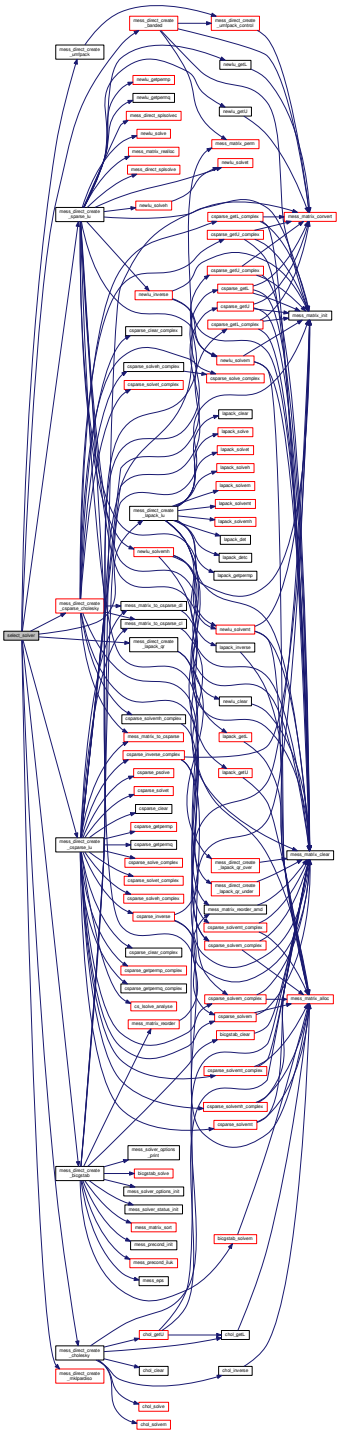
The select_solver function generates a solver for a linear system of equations

$$AX = B$$

Solvers are choosen with i. Possible values are:

Input	Description
0	mess_direct_create_sparse_lu defined in newlu.c
1	mess_direct_create_csparse_lu defined in csparse.c
2	mess_direct_create_umfpack defined in umfpack.c
3	mess_direct_create_bicgstab defined in bicgstab.c
4	mess_direct_create_lapack_lu defined in lapack.c
5	mess_direct_create_lapack_qr defined in lapackqr.c
6	mess_direct_create_cholesky defined in cholesky.c
7	mess_direct_create_csparse_cholesky defined in csparse_chol.c
8	mess_direct_create_cholmod_cholesky defined in cholmod_chol.c
9	mess_direct_create_banded defined in banded.c
10	mess_direct_create_superlu defined in superlu.c
11	mess_direct_create_mklpardiso defined in mklpardiso.c

Here is the call graph for this function:



10.442.2.3 int testmat (int *i*, mess_matrix *A*)

Parameters

in	*i*	input solver to choose
in	*A*	input matrix

Returns

zero on success or a non-zero error value otherwise

The testmat function solves a linear system of equations

$$op(A)X = B$$

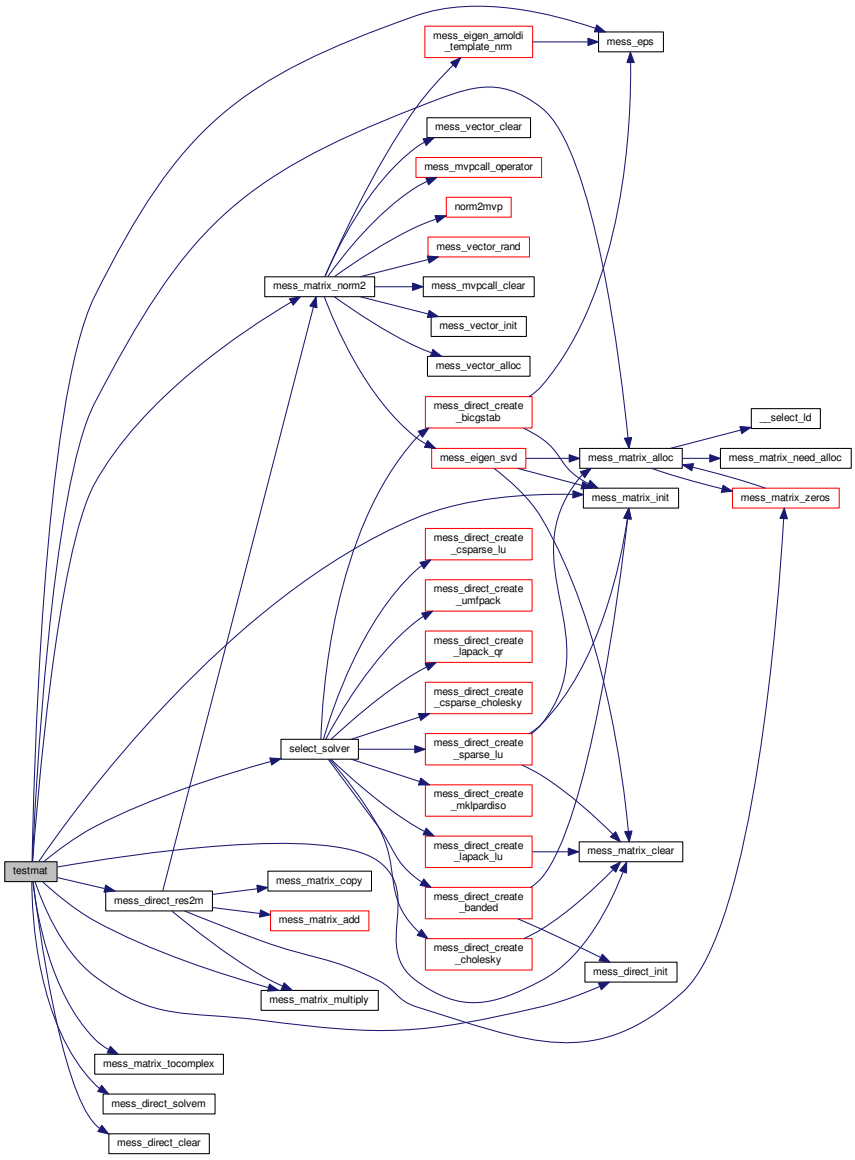
and checks if the solution is computed correctly.
Operations $op(.)$ on matrix A can be

- MESS_OP_NONE ($op(A) = A$),
- MESS_OP_TRANSPOSE ($op(A) = A^T$),
- MESS_OP_HERMITIAN ($op(A) = A^H$).

Parameter i is only used to choose the solver. Possible values are:

Input	Description
0	mess_direct_create_sparse_lu
1	mess_direct_create_csparse_lu
2	mess_direct_create_umfpack
3	mess_direct_create_bicgstab
4	mess_direct_create_lapack_lu
5	mess_direct_create_lapack_qr
6	mess_direct_create_cholesky
7	mess_direct_create_csparse_cholesky
8	mess_direct_create_cholmod_cholesky
9	mess_direct_create_banded
10	mess_direct_create_superlu
11	mess_direct_create_mklpardiso

Here is the call graph for this function:



10.443 linsolvem_multi.c File Reference

Functions

- int select_solver (int i, mess_matrix A, mess_vector shifts, mess_multidirect sol)

  Select a solver.

- int linsolvem_multi_testmat (int i, int cpx, mess_matrix A)

  Test solving a shifted linear system of equations (matrix version).

- int main (int argc, char ∗∗argv)

10.443.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.443.2 Function Documentation

10.443.2.1 int linsolvem_multi_testmat (int *i*, int *cpx*, mess_matrix *A*)

Parameters

in	*i*	input solver to choose
in	*cpx*	input true if complex values are used
in	*A*	input matrix

Returns

zero on success or a non-zero error value otherwise

The linsolvem_multi_testmat function solves a shifted linear system of equations

$$op(A + pI)X = B$$

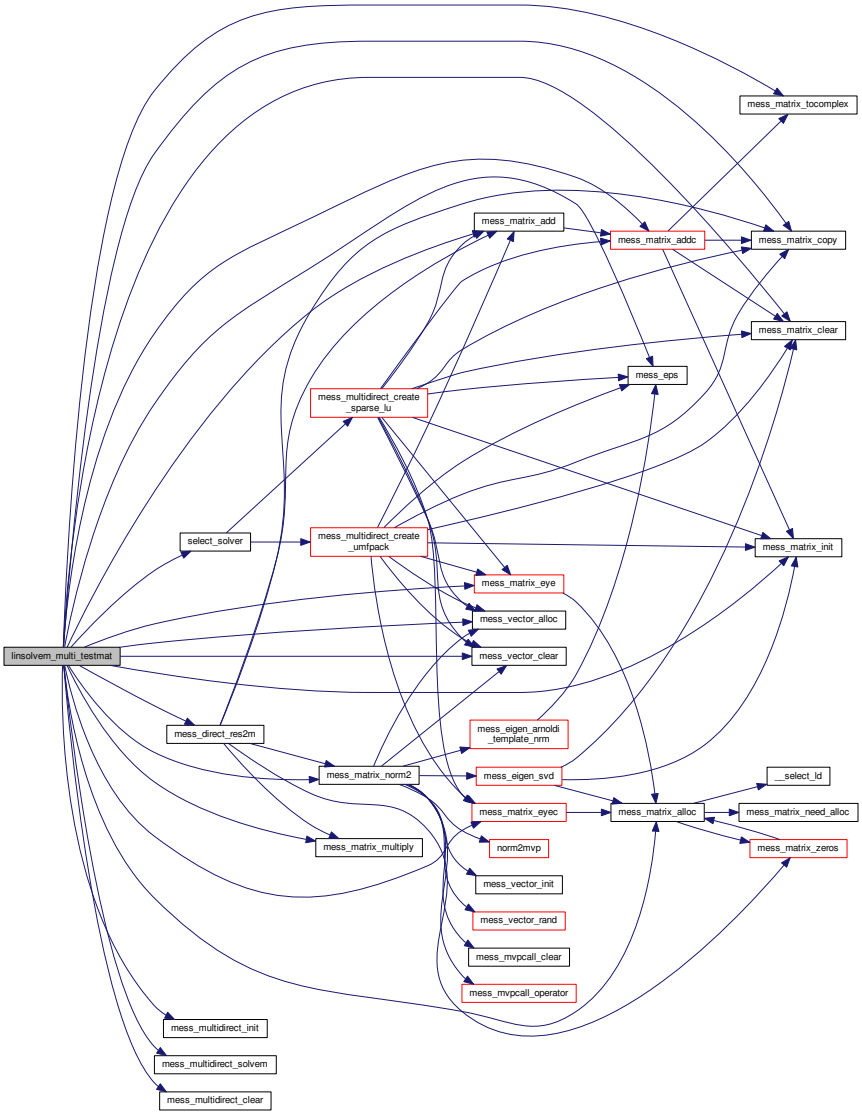
and checks if the solution is computed correctly.
Operations $op(.)$ on matrix $A + pI$ can be

- MESS_OP_NONE ($op(A + pI) = A + pI$),
- MESS_OP_TRANSPOSE ($op(A + pI) = A^T + pI$),
- MESS_OP_HERMITIAN ($op(A) = A^H + pI$).

Parameter i is only used to choose the solver. Possible values are:

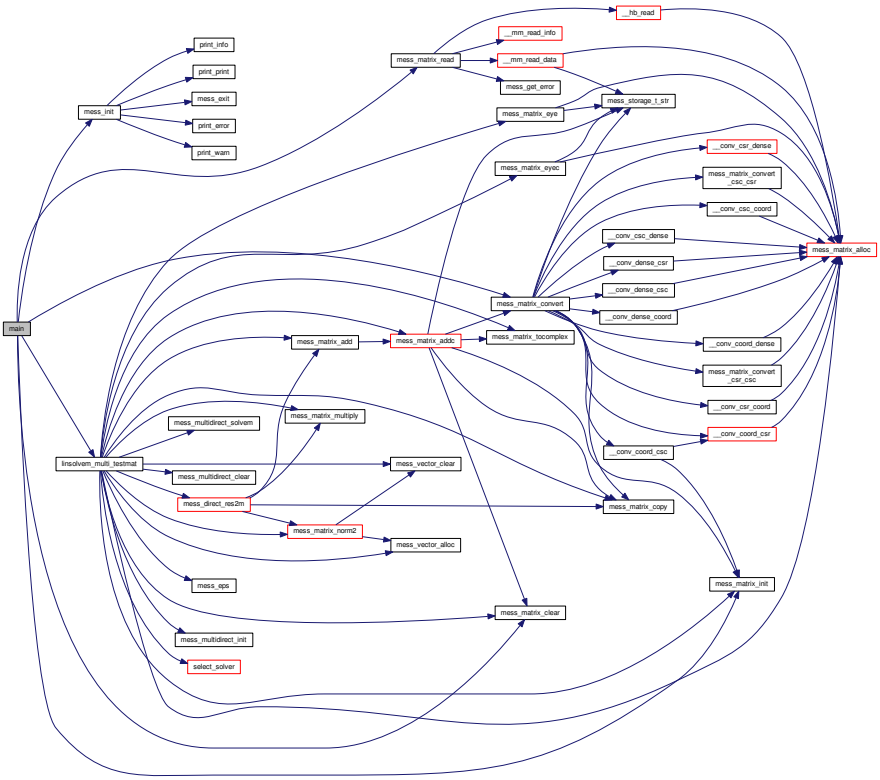
- 0 - mess_multidirect_create_sparse_lu,
- 1 - mess_multidirect_create_umfpack,

Here is the call graph for this function:



10.443.2.2 int main (int *argc*, char ** *argv*)

Here is the call graph for this function:



10.443.2.3 int select_solver (int *i*, mess_matrix *A*, mess_vector *shifts*, mess_multidirect *sol*)

Parameters

in	*i*	input solver to choose
in	*A*	input matrix
in	*shifts*	input shifts
out	*sol*	generated solver

Returns

zero on success or a non-zero error value otherwise

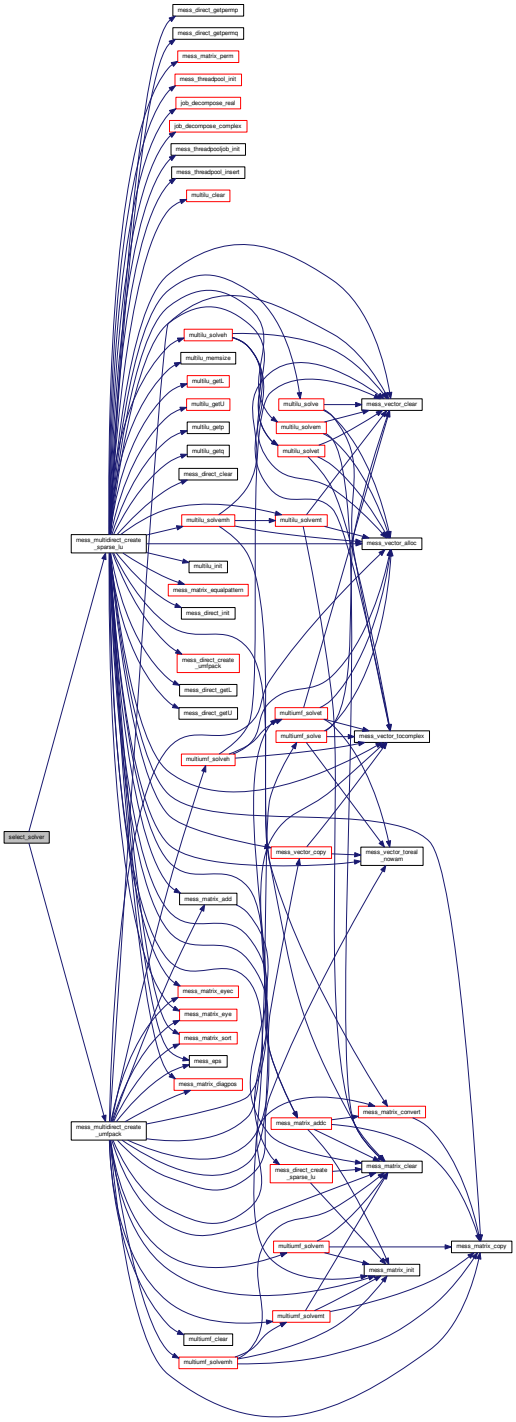
The select_solver function generates a solver for a shifted linear system of equations

$$(A + shiftsI)X = B$$

Solvers are choosen with i. Possible values are:

- 0 - mess_multidirect_create_sparse_lu,
- 1 - mess_multidirect_create_umfpack,

Here is the call graph for this function:



10.444 linspace.c File Reference

Functions

- int mess_vector_linspace (mess_vector vect, double a, double b, mess_int_t nsample)

  Create a linspace vector.

- int mess_vector_logspace10 (mess_vector vect, double a, double b, mess_int_t nsample)

  Create a $\log_{10}$ space vector.
- int mess_vector_logspacee (mess_vector vect, double a, double b, mess_int_t nsample)

  Create a $\log_{e}$ space.
- int mess_vector_logspace2 (mess_vector vect, double a, double b, mess_int_t nsample)

  Create a $\log_{2}$ space.

10.444.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.445 lradi.c File Reference

Macros

- #define RES2_COUNT 3
- #define RES_ABORT 1e+12
- #define IS_REAL(X) (cimag((X))==0.0)

Functions

- int mess_lradi (mess_equation eqn, mess_options opt, mess_status stat, mess_matrix Z)

  Wrapper/Alternative name for mess_lrcfadi_adi.
- int mess_lrcfadi_adi (mess_equation eqn, mess_options opt, mess_status stat, mess_matrix Z)

  Solve a (generalized) Lyapunov Equation using the low-rank-Cholesky-factor ADI.

10.445.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.445.2 Macro Definition Documentation

10.445.2.1 #define IS_REAL(X) (cimag((X))==0.0)

10.445.2.2 #define RES2_COUNT 3

10.445.2.3 #define RES_ABORT 1e+12

10.446 lradi.c File Reference

Functions

- PyObject ∗ pymess_lradi (PyObject ∗self, PyObject ∗args)

10.446.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Nitin Prasad (nitin.prasad93@gmail.com, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.447 lrcf_adi.h File Reference

Data Structures

- struct __mess_lrcfadi_step

  Structure to pass information to the step function inside the LRCF iterations.
- struct mess_options_st

  Options for the LRCF-ADI algorithm family.
- struct mess_status_st

  Structure/object to save the final status of a LRCF algorithm.
- struct mess_equation_st

  Structure defining an equation as a set of function pointers.
- struct mess_equation_st::AX_st

  Structure defining $Y = op(A)X$.
- struct mess_equation_st::EX_st

  Structure defining $Y = op(E)X$.
- struct mess_equation_st::AINV_st

  Structure defining $op(A)Y = X$.
- struct mess_equation_st::EINV_st

  Structure defining $op(E)Y = X$.
- struct mess_equation_st::ApEX_st

  Structure defining $Y = op(A + \alpha pE)X$.
- struct mess_equation_st::ApEINV_st

  Structure defining $op(A + pE)Y = X$.

Typedefs

- typedef struct __mess_lrcfadi_step mess_lrcfadi_step

  Structure to pass information to the step function inside the LRCF iterations.
- typedef struct mess_options_st mess_options_st

  Options for the LRCF-ADI algorithm family.
- typedef mess_options_st $*$ mess_options
- typedef struct mess_status_st mess_status_st

  Structure/object to save the final status of a LRCF algorithm.
- typedef mess_status_st $*$ mess_status
- typedef struct mess_equation_st $*$ mess_equation

  Type definition for the definition of an equation.

Enumerations

Functions

- int mess_equation_init (mess_equation *eqn)

  Initialize an empty mess_equation object.
- int mess_equation_clean (mess_equation eqn)

  Call cleanup functions for all internally generated data.
- int mess_equation_clear (mess_equation *eqn)

  Clean a mess_equation structure.
- int mess_equation_set_rhs (mess_equation eqn, mess_options opt, mess_matrix rhs)

  Set a new right hand side in a mess_equation object.
- int mess_equation_set_rhs2 (mess_equation eqn, mess_matrix rhs)

  Set a new right hand side in a mess_equation object without additional checks.
- int mess_status_init (mess_status *status)

  Initialize a mess_status object.
- int mess_status_clear (mess_status *status)

  Clear a mess_status object.
- int mess_status_print (mess_status stat)

  Print a mess_status object to the standard output.
- int mess_status_printfull (mess_status stat)

  Print a mess_status object to the standard output including embbed objects.
- int mess_status_copy (mess_status src, mess_status dest)

  Copy a mess_status object to another one.
- mess_int_t mess_status_memsize (mess_status status)

  Return the size of a mess_status in bytes.
- int mess_status_write (mess_status stat, const char *filename)

  Save the information of a mess_status object to a text file.
- int mess_options_init (mess_options *opt)

  Initialize LRCF-ADI options with default values.
- int mess_options_clear (mess_options *opt)

  Clear a mess_options object.
- int mess_options_print (mess_options opt)

  Display a mess_options object on standard output.
- int mess_options_set_proj_space (mess_options opt, mess_matrix proj_space)

  Copy the mess_matrix `proj_space`.
- int mess_options_unset_proj_space (mess_options opt)

  Delete the field `proj_space` from mess_options instance.
- int mess_options_set_K0 (mess_options opt, mess_matrix K0)

  Copy the mess_matrix `K0`.
- int mess_options_unset_K0 (mess_options opt)

  Delete the field `K0` from mess_options instance.
- int mess_options_set_W (mess_options opt, mess_matrix W)

  Copy the mess_matrix `W`.
- int mess_options_unset_W (mess_options opt)

  Delete the field `W` from mess_options instance.
- mess_int_t mess_options_memsize (mess_options opt)

  Return the size of a mess_options in bytes.
- int mess_options_copy (mess_options src, mess_options dest)

  Copy a mess_options object to another one.
- int mess_parameter (mess_equation eqn, mess_options opt, mess_status stat)

Wrapper around mess_lrcfadi_parameter except that mess_equation.parameter is preferred (if available).

- int mess_lrcfadi_parameter (mess_equation eqn, mess_options opt, mess_status stat)

  Compute parameters for the low-rank Cholesky factor ADI iteration.
- int mess_lrcfadi_getritzvalues (mess_equation eqn, mess_options opt, mess_vector ritz)

  Compute a set of Ritz values from a given equation.
- int mess_lrcfadi_residual (mess_equation eqn, mess_options opt, mess_matrix Z, double ∗nrm)

  Compute 2-norm residual of an equation.
- int mess_lradi (mess_equation eqn, mess_options opt, mess_status stat, mess_matrix Z)

  Wrapper/Alternative name for mess_lrcfadi_adi.
- int mess_lrcfadi_adi (mess_equation eqn, mess_options opt, mess_status stat, mess_matrix Z)

  Solve a (generalized) Lyapunov Equation using the low-rank-Cholesky-factor ADI.
- int mess_lrnm (mess_equation eqn, mess_options opt, mess_status stat, mess_matrix Z)

  Wrapper/Alternative name for mess_lrcfadi_nm.
- int mess_lrcfadi_nm (mess_equation eqn, mess_options opt, mess_status stat, mess_matrix Z)

  Solve an algebraic Riccati Equation using the low-rank-Cholesky factor Newton method.
- int mess_lrcfadi_galerkin (mess_equation eqn, mess_options opt, mess_equation_t eqn_type, mess_matrix Z)

  Refine a solution of a matrix equation using Galerkin projection.
- int mess_lrcfadi_nm_linesearch_poly (mess_matrix W, mess_matrix Wnew, mess_matrix deltaK, mess_matrix deltaKnew, double ∗alpha, double ∗beta, double ∗delta, double ∗gamma, double ∗epsilon, double ∗zeta)

  Compute the coefficients of the polynomial for linesearch.
- int mess_lrcfadi_nm_linesearch_lambda (double ∗alpha, double ∗beta, double ∗delta, double ∗gamma, double ∗epsilon, double ∗zeta, double ∗lambda)

  Compute the optimal step size for linesearch.
- int mess_lrcfadi_nm_linesearch (mess_matrix W, mess_matrix Wnew, mess_matrix deltaK, mess_matrix deltaKnew, double ∗lambda)

  Compute the optimal step size for linesearch.
- int mess_lrcfadi_lp_s (mess_double_cpx_t ∗p, mess_int_t lp, mess_double_cpx_t ∗set, mess_int_t lset, double ∗max_r, mess_int_t ∗ind)

  Compute the maximum magnitude of the rational ADI function.
- int mess_lrcfadi_lp_mnmx (mess_double_cpx_t ∗rw, mess_int_t lrw, mess_int_t l0, mess_double_cpx_t ∗p, mess_int_t ∗lp)

  Compute ADI parameters solving the ADI minimax problem suboptimal.
- int mess_lrcfadi_fac2diff (mess_matrix Zold, mess_matrix Z, double ∗chg)

  Compute 2-norm difference between two factored solutions of a matrix equation.
- int mess_lrcfadi_facFdiff (mess_matrix Zold, mess_matrix Z, double ∗chg)

  Compute the relative Frobenius-norm difference between two factored solutions of a matrix equation.
- int mess_lrcfadi_para_wachspress (double a, double b, double alpha, double tol, double ∗p, mess_int_t ∗lp)

  Compute Wachspress parameters.
- int mess_lrcfadi_ccsvd (mess_matrix Z, double ccTol)

  Perform a SVD based column compression on low rank factor.
- int mess_lrcfadi_colcompress (mess_matrix Z, double ccTol)

  Perform a column compression on a low rank factor.
- int mess_lrcfadi_res2 (mess_matrix F, mess_matrix G, mess_matrix Z, double ∗nrm)

  Compute 2-norm of the Lyapunov residual via an Arnoldi process.
- int mess_lrcfadi_res2BK (mess_matrix F, mess_matrix G, mess_matrix Z, mess_matrix B, mess_matrix K, double ∗nrm)

  Compute the 2-norm of the Lyapunov residual (BK version).
- int mess_lrcfadi_res2g (mess_matrix F, mess_matrix M, mess_matrix G, mess_matrix Z, double ∗nrm)

  Compute the 2-norm of the generalized Lyapunov residual.

- int mess_lrcfadi_res2gBK (mess_matrix F, mess_matrix M, mess_matrix G, mess_matrix Z, mess_matrix B, mess_matrix K, double *nrm)

  Compute the 2-norm of the generalized Lyapunov residual (BK-Version).
- int mess_lrcfadi_res2t (mess_matrix F, mess_matrix G, mess_matrix Z, double *nrm)

  Compute 2 norm of the Lyapunov residual (transposed version).
- int mess_lrcfadi_res2gt (mess_matrix F, mess_matrix M, mess_matrix G, mess_matrix Z, double *nrm)

  Compute the 2-norm of the generalized Lyapunov residual (transposed version).
- int mess_lrcfadi_res2g_so (mess_matrix M, mess_matrix G, mess_matrix K, mess_matrix RHS, mess_matrix Z, double *nrm)
- int mess_lrcfadi_res2gt_so (mess_matrix M, mess_matrix G, mess_matrix K, mess_matrix RHS, mess_matrix Z, double *nrm)
- int mess_lrcfadi_res2nm (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix Z, double *nrm)

  Compute the 2-norm of the Riccati residual.
- int mess_lrcfadi_res2nmg (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix M, mess_matrix Z, double *nrm)

  Compute the 2-norm of the generalized Riccati residual.
- int mess_dense_nm_gmare (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X)

  Implementation of the dense Newton-method to solve the generalized Riccati Equation $A^TXE + E^TXA - E^TXGXE + Q = 0$.
- int mess_dense_nm_gpare (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X)

  Implementation of the dense Newton-method to solve the generalized Riccati Equation $A^TXE + E^TXA + E^TXGXE + Q = 0$.
- int mess_dense_nm_gmpare (mess_matrix X0, mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_int_t plus, mess_int_t linesearch, mess_operation_t trans, mess_int_t maxit, mess_norm_t nrm, double absres_tol, double relres_tol, double *absres, double *relres, mess_matrix RES, mess_matrix X)

  Implementation of the dense Newton method to solve the Riccati Equation $A^TXE + EXA + / - XGX + Q = 0$.
- int mess_dense_res_gmpare (mess_matrix A, mess_matrix E, mess_matrix Q, mess_matrix G, mess_matrix X, mess_int_t plus, mess_operation_t trans, mess_norm_t nrm_t, double *res, double *rel)

  Compute the residual of the algebraic (generalized) Riccati (or Lyapunov) equation.
- int mess_lrcf_gsignfac (mess_matrix A, mess_matrix E, mess_matrix B, mess_int_t *maxit, double *tol, mess_matrix Z, mess_sign_scale_t scale)

  Solve a generalized Lyapunov Equation $A^TXE + E^TXA + B^TB = 0$ via sign function iteration.
- int mess_lrcf_signfac (mess_matrix A, mess_matrix B, mess_int_t *maxit, double *tol, mess_matrix Z, mess_sign_scale_t scale)

  Solve a standard Lyapunov Equation $A^TX + XA + B^TB = 0$ via sign function iteration.
- int mess_lrcf_gsignber (mess_matrix A, mess_matrix E, mess_matrix B, mess_int_t *maxit, double *tol, mess_matrix Z, mess_sign_scale_t scale)

  Solve a generalized Bernoulli equation $A^TXE + E^TXA - E^TXB^TBXE = 0$ via sign function iteration.
- int mess_lrcf_signber (mess_matrix A, mess_matrix B, mess_int_t *maxit, double *tol, mess_matrix Z, mess_sign_scale_t scale)

  Solve a standard Bernoulli equation $A^TX + XA - XB^TBX = 0$ via sign function iteration.
- int mess_equation_lyap (mess_equation eqn, mess_options opt, mess_matrix A, mess_matrix E, mess_matrix B)

  Setup a mess_Equation object to solve a matrix-valued Lyapunov Equation.
- int mess_equation_riccati (mess_equation eqn, mess_options opt, mess_matrix A, mess_matrix E, mess_matrix B, mess_matrix C)

  Setup a mess_equation object to solve a matrix-valued Riccati Equation.
- int mess_equation_std_lyap (mess_equation eqn, mess_options opt, mess_matrix A, mess_matrix B)

  Setup a mess_Equation object to solve a standard Lyapunov Equation.
- int mess_equation_std_riccati (mess_equation eqn, mess_options opt, mess_matrix A, mess_matrix B, mess_matrix C)

Setup a mess_equation object to solve a standard Riccati Equation.

- int mess_equation_glyap (mess_equation eqn, mess_options opt, mess_matrix A, mess_matrix E, mess_matrix B)

  Setup a mess_Equation object to solve a generalized Lyapunov Equation.
- int mess_equation_griccati (mess_equation eqn, mess_options opt, mess_matrix A, mess_matrix E, mess_matrix B, mess_matrix C)

  Setup a mess_equation object to solve a generalized Riccati Equation.
- int mess_equation_glyap_so1 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix D, mess_matrix K, mess_matrix B, double lowerbound, double upperbound)

  Generate an Equation object to a Lyapunov Equation arising from a second order system.
- int mess_equation_griccati_so1 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix D, mess_matrix K, mess_matrix B, mess_matrix C, double lowerbound, double upperbound)

  Generate an equation object to a Riccati Equation arising from a second order system.
- int mess_equation_glyap_so2 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix D, mess_matrix K, mess_matrix B, double lowerbound, double upperbound)

  Generate an Equation object to a generalized Lyapunov Equation arising from a second order system.
- int mess_equation_griccati_so2 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix D, mess_matrix K, mess_matrix B, mess_matrix C, double lowerbound, double upperbound)

  Generate an equation object to a generalized Riccati Equation arising from a second order system.
- int mess_equation_glyap_dae2 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix A, mess_matrix G, mess_matrix B, double delta)

  Generate an Equation object to solve a generalized Lyapunov Equation from Hessenberg Index 2 DAE.
- int mess_equation_griccati_dae2 (mess_equation e, mess_options opt, mess_matrix M, mess_matrix A, mess_matrix G, mess_matrix B, mess_matrix C, double delta)

  Generate an equation object to a Riccati Equation arising from a Hessenberg Index 2 DAE.
- int mess_equation_glyap_dae1 (mess_equation e, mess_options opt, mess_matrix E11, mess_matrix A11, mess_matrix A12, mess_matrix A21, mess_matrix A22, mess_matrix rhs)

  Generate an Equation object to solve a generalized Lyapunov Equation from Index 1 DAE.
- int mess_equation_griccati_dae1 (mess_equation e, mess_options opt, mess_matrix E11, mess_matrix A11, mess_matrix A12, mess_matrix A21, mess_matrix A22, mess_matrix B, mess_matrix C)

  Generate an Equation object to solve a generalized Lyapunov Equation from Index 1 DAE.
- int mess_equation_stable (mess_equation e, mess_options opt, mess_equation base, mess_matrix B, mess_matrix K)

  Generate a stabilized Lyapunov Equation object.
- mess_int_t mess_equation_dim (mess_equation eqn)

  Return the dimension of the equation of object.
- mess_int_t mess_equation_has_A (mess_equation eqn)

  Checks if $x = \mathcal{A}y$ is implemented.
- mess_int_t mess_equation_has_As (mess_equation eqn)

  Checks if $\mathcal{A}x = y$ is implemented.
- mess_int_t mess_equation_has_E (mess_equation eqn)

  Checks if $x = \mathcal{E}y$ is implemented.
- mess_int_t mess_equation_has_Es (mess_equation eqn)

  Checks if $\mathcal{E}x = y$ is implemented.
- mess_int_t mess_equation_has_ApE (mess_equation eqn)

  Checks if $x = (\mathcal{A} + \mathcal{E})y$ is implemented.
- mess_int_t mess_equation_has_ApEs (mess_equation eqn)

  Checks if $(\mathcal{A} + p\mathcal{E})x = y$ is implemented.
- int mess_equation_A_pre (mess_equation eqn)

  Initialization function of the linear mapping $x = \mathcal{A}y$.
- int mess_equation_A_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $x = \mathcal{A}y$.

- int mess_equation_A_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute $x = \mathcal{A}y$.
- int mess_equation_A_apply_vector (mess_equation eqn, mess_operation_t op, mess_vector in, mess_vector out)

  Compute $x = \mathcal{A}y$ vector valued.
- int mess_equation_E_pre (mess_equation eqn)

  Initialization function of the linear mapping $x = \mathcal{E}y$.
- int mess_equation_E_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $x = \mathcal{E}y$.
- int mess_equation_E_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute $x = \mathcal{E}y$.
- int mess_equation_E_apply_vector (mess_equation eqn, mess_operation_t op, mess_vector in, mess_vector out)

  Compute $x = \mathcal{E}y$ vector valued.
- int mess_equation_As_pre (mess_equation eqn)

  Initialization function of the linear mapping $\mathcal{A}x = y$.
- int mess_equation_As_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $\mathcal{A}x = y$.
- int mess_equation_As_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute $\mathcal{A}x = y$.
- int mess_equation_As_apply_vector (mess_equation eqn, mess_operation_t op, mess_vector in, mess_↩vector out)

  Compute $\mathcal{A}x = y$ vector valued.
- int mess_equation_Es_pre (mess_equation eqn)

  Initialization function of the linear mapping $\mathcal{E}x = y$.
- int mess_equation_Es_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $\mathcal{E}x = y$.
- int mess_equation_Es_apply (mess_equation eqn, mess_operation_t op, mess_matrix in, mess_matrix out)

  Compute $\mathcal{E}x = y$.
- int mess_equation_Es_apply_vector (mess_equation eqn, mess_operation_t op, mess_vector in, mess_↩vector out)

  Compute $\mathcal{E}\S = y$ vector valued.
- int mess_equation_ApE_pre (mess_equation eqn, mess_vector parameters)

  Initialization function of the linear mapping $x = (\mathcal{A} + p\mathcal{E})y$.
- int mess_equation_ApE_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $x = (\mathcal{A} + \mathcal{E})y$.
- int mess_equation_ApE_apply (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_↩int_t p_idx, mess_matrix in, mess_matrix out)

  Compute $x = (\mathcal{A} + p\mathcal{E})y$.
- int mess_equation_ApE_apply_vector (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t p_idx, mess_vector in, mess_vector out)

  Compute $x = (\mathcal{A} + p\mathcal{E})y$ with vectors..
- int mess_equation_ApEs_pre (mess_equation eqn, mess_vector parameters)

  Initialization function of the linear mapping $(\mathcal{A} + p\mathcal{E})x = y$.
- int mess_equation_ApEs_post (mess_equation eqn)

  Finalizes/Clears the linear mapping $(\mathcal{A} + \mathcal{E})x = y$.
- int mess_equation_ApEs_apply (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess↩_int_t p_idx, mess_matrix in, mess_matrix out)

  Compute $(\mathcal{A} + p\mathcal{E})x = y$.
- int mess_equation_ApEs_apply_vector (mess_equation eqn, mess_operation_t op, mess_double_cpx_t p, mess_int_t p_idx, mess_vector in, mess_vector out)

  Compute $(\mathcal{A} + p\mathcal{E})x = y$ with vectors..

10.447.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.448 lrnm.c File Reference

Macros

- #define IS_REAL(X) (cimag((X))==0.0)
- #define SET_COLOR MSG_PRINT("\033[35m");
- #define UNSET_COLOR MSG_PRINT("\n\033[0m");

Functions

- static int _scalecat (mess_operation_t op, double alpha, mess_matrix A, double beta, mess_matrix B)

  Perform $A = [\alpha A, \beta B]$ *or* $A = [\alpha A; \beta B]$.
- int mess_lrnm (mess_equation eqn, mess_options opt, mess_status status, mess_matrix Zout)

  Wrapper/Alternative name for mess_lrcfadi_nm.
- int mess_lrcfadi_nm (mess_equation eqn, mess_options opt, mess_status status, mess_matrix Zout)

  Solve an algebraic Riccati Equation using the low-rank-Cholesky factor Newton method.

10.448.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.448.2 Macro Definition Documentation

10.448.2.1 #define IS_REAL(X) (cimag((X))==0.0)

10.448.2.2 #define SET_COLOR MSG_PRINT("\033[35m");

10.448.2.3 #define UNSET_COLOR MSG_PRINT("\n\033[0m");

10.448.3 Function Documentation

10.448.3.1 static int _scalecat (mess_operation_t *op*, double *alpha*, mess_matrix *A*, double *beta*, mess_matrix *B*) [static]

Parameters

in	*op*	input operation type
in	*alpha*	input double scalar α
in	*A*	input mess_matrix A
in	*beta*	input double scalar β
in,out	*B*	input mess_matrix B

Returns

zero on succes or a non-zero error value otherwise

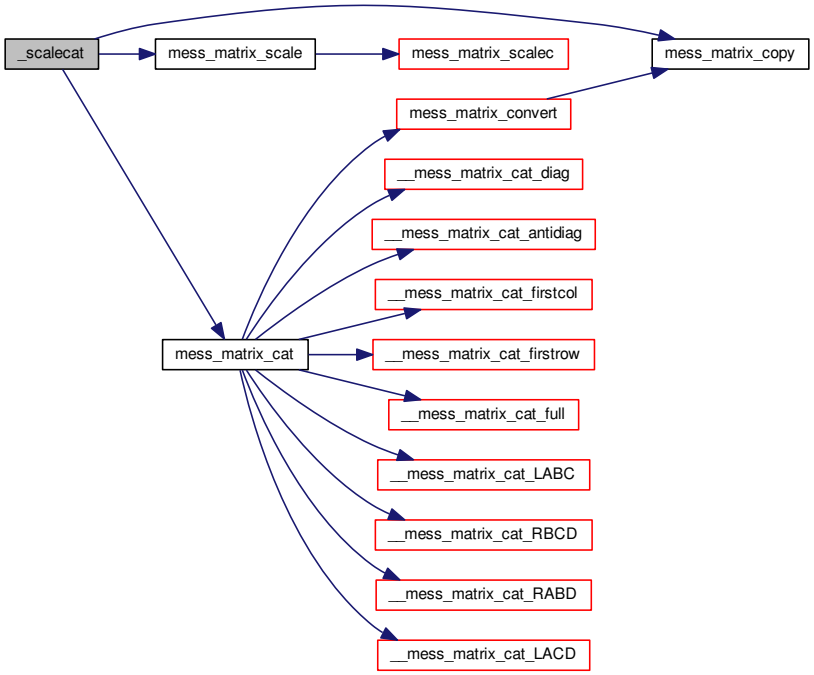
The _scalecat function performs $A = [\alpha A, \beta B]$ or $A = [\alpha A; \beta B]$.
If `op` is equal to MESS_OP_NONE the first operation will be performed, otherwise the second one.

The output `B` will be MESS_DENSE.

Attention

Internal use only.

Here is the call graph for this function:



10.449 lrnm.c File Reference

Functions

- PyObject ∗ pymess_lrnm (PyObject ∗self, PyObject ∗args)

10.449.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Nitin Prasad (nitin.prasad93@gmail.com, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.450 lrnm_linesearch.c File Reference

Functions

- int mess_lrcfadi_nm_linesearch_poly (mess_matrix W, mess_matrix Wnew, mess_matrix deltaK, mess_matrix deltaKnew, double *alpha, double *beta, double *delta, double *gamma, double *epsilon, double *zeta)

  Compute the coefficients of the polynomial for linesearch.
- static int __filter_linesearch_real (const double *val)
- static int __filter_linesearch_cpx (const mess_double_cpx_t *val)
- static double linesearch_poly_eval (double t, double *alpha, double *beta, double *delta, double *gamma, double *epsilon, double *zeta)
- int mess_lrcfadi_nm_linesearch_lambda (double *alpha, double *beta, double *delta, double *gamma, double *epsilon, double *zeta, double *lambda)

  Compute the optimal step size for linesearch.
- int mess_lrcfadi_nm_linesearch (mess_matrix W, mess_matrix Wnew, mess_matrix deltaK, mess_matrix deltaKnew, double *lambda)

  Compute the optimal step size for linesearch.

10.450.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

[7]

10.450.2 Function Documentation

10.450.2.1 static int __filter_linesearch_cpx (const mess_double_cpx_t * *val*) [static]

10.450.2.2 static int __filter_linesearch_real (const double * *val*) [static]

10.450.2.3 static double linesearch_poly_eval (double *t*, double * *alpha*, double * *beta*, double * *delta*, double * *gamma*, double * *epsilon*, double * *zeta*) [static]

10.451 lrsrm.c File Reference

Functions

- int mess_bt_lrsrm (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Low-rank square root balanced truncation for first order LTI systems.

10.451.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.452 lrsrm_somor.c File Reference

Functions

- int mess_bt_lrsrm_somor (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Low-rank square root balanced truncation for second order LTI systems to first order LTI systems.

10.452.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.453 lrsrm_somor_so.c File Reference

Functions

- int mess_bt_lrsrm_somor_so (mess_dynsys sys, mess_bt_options btopt, mess_options adiopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Low-rank square root balanced truncation for second order LTI systems to second order LTI systems.
- int mess_bt_lrsrm_somor_so_gram (mess_dynsys sys, mess_matrix ZB, mess_matrix ZC, mess_bt_options btopt, mess_matrix V, mess_matrix W, mess_bt_status status)

  Low-rank square root balanced truncation for second order LTI systems to second order LTI systems (Gramian version).

10.453.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.454 lu.c File Reference

Macros

- #define USE_FORTRAN_KERNELS

Functions

- void F77_GLOBAL (lurcsrr, LURCSRR)
- int mess_decomp_lureuse_kernelcsr_complex (mess_int_t rows, mess_int_t cols, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_int_t *lcolptr, mess_int_t *lrowptr, mess_int_t *ucolptr, mess_int_t *urowptr, mess_int_t *p, mess_int_t *q, mess_double_cpx_t *lvalues, mess_double_cpx_t *uvalues)

10.454.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.454.2 Macro Definition Documentation

10.454.2.1 #define USE_FORTRAN_KERNELS

10.454.3 Function Documentation

10.454.3.1 void F77_GLOBAL (lurcsrr , LURCSRR)

10.455 lyap.c File Reference

Functions

- int mess_lyap (mess_matrix A, mess_matrix E, mess_matrix B, mess_matrix Z)

  Frontend to compute a factorized solution of a (generalized) Lyapunov Equation.

10.455.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.456 lyap.c File Reference

Functions

- PyObject ∗ pymess_lyap (PyObject ∗self, PyObject ∗args)

  Wrapper around the easy to use interface mess_lyap for use with Python.

10.456.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Nitin Prasad (nitin.prasad93@gmail.com, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.457 lyap_res2.c File Reference

Functions

- int mess_direct_lyapunov_res2 (mess_operation_t op, mess_matrix A, mess_matrix M, mess_matrix X, double ∗res)

  Compute the 2-norm residual of a Lyapunov Equation (DENSE).
- int mess_direct_generalized_lyapunov_res2 (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix M, mess_matrix X, double ∗res2)

  Compute the 2-norm residual of a generalized Lyapunov Equation.

10.457.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.458 lyap_sgn_fac.c File Reference

Functions

- int mess_lrcf_gsignfac (mess_matrix A, mess_matrix E, mess_matrix B, mess_int_t *maxit, double *tol, mess_matrix Z, mess_sign_scale_t scale)

  Solve a generalized Lyapunov Equation $A^T XE + E^T XA + B^T B = 0$ via sign function iteration.
- int mess_lrcf_signfac (mess_matrix A, mess_matrix B, mess_int_t *maxit, double *tol, mess_matrix Z, mess_sign_scale_t scale)

  Solve a standard Lyapunov Equation $A^T X + XA + B^T B = 0$ via sign function iteration.

10.458.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.459 malloc.c File Reference

Typedefs

- typedef void *(* malloc_call_t) (size_t size)
- typedef void *(* realloc_call_t) (void *ptr, size_t size)
- typedef void *(* calloc_call_t) (size_t nmeb, size_t selem)
- typedef void(* free_call_t) (void *ptr)

Functions

- void mess_set_malloc (void *malloc_ptr, void *realloc_ptr, void *calloc_ptr, void *free_ptr)

  Set the malloc implementation to be used.
- void * __mess_malloc (size_t size)

  Wrapper around malloc.
- void * __mess_realloc (void *ptr, size_t newsize)

  Wrapper around realloc.
- void * __mess_calloc (size_t nmemb, size_t size)

  Wrapper around calloc.
- void __mess_free (void *ptr)

  Wrapper around free.

Variables

- static malloc_call_t fn_malloc = (void *) &malloc
- static realloc_call_t fn_realloc = (void *) &realloc
- static calloc_call_t fn_calloc = (void *) &calloc
- static free_call_t fn_free = (void *) &free

10.459.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.459.2 Typedef Documentation

10.459.2.1 typedef void∗(∗ calloc_call_t) (size_t nmeb, size_t selem)

10.459.2.2 typedef void(∗ free_call_t) (void ∗ptr)

10.459.2.3 typedef void∗(∗ malloc_call_t) (size_t size)

10.459.2.4 typedef void∗(∗ realloc_call_t) (void ∗ptr, size_t size)

10.459.3 Variable Documentation

10.459.3.1 calloc_call_t fn_calloc = (void ∗) &calloc [static]

10.459.3.2 free_call_t fn_free = (void ∗) &free [static]

10.459.3.3 malloc_call_t fn_malloc = (void ∗) &malloc [static]

10.459.3.4 realloc_call_t fn_realloc = (void ∗) &realloc [static]

10.460 map.c File Reference

Macros

- #define MESS_MAP_STATIC_FUNC(BASENAME, REALFUNC, CPXFUNC)
- #define MESS_MAP_STATIC_FUNC2(BASENAME, REALFUNC)
- #define MESS_MAP_STATIC_FUNC3(BASENAME, REALFUNC)

Functions

- int mess_matrix_map (mess_matrix mat, double(∗f_real)(double), mess_double_cpx_t(∗f_cpx)(mess_double_cpx_t), int full)

  Map a scalar function to a matrix.
- static double map_abs_real (double d)
- static mess_double_cpx_t map_abs_cpx (mess_double_cpx_t z)
- int mess_matrix_map_abs (mess_matrix mat, int full)
- static double map_acos_real (double d)
- static mess_double_cpx_t map_acos_cpx (mess_double_cpx_t z)
- int mess_matrix_map_acos (mess_matrix mat, int full)
- static double map_acosh_real (double d)
- static mess_double_cpx_t map_acosh_cpx (mess_double_cpx_t z)
- int mess_matrix_map_acosh (mess_matrix mat, int full)
- static double map_arg_real (double d)
- static mess_double_cpx_t map_arg_cpx (mess_double_cpx_t z)

- int mess_matrix_map_arg (mess_matrix mat, int full)
- static double map_asin_real (double d)
- static mess_double_cpx_t map_asin_cpx (mess_double_cpx_t z)
- int mess_matrix_map_asin (mess_matrix mat, int full)
- static double map_asinh_real (double d)
- static mess_double_cpx_t map_asinh_cpx (mess_double_cpx_t z)
- int mess_matrix_map_asinh (mess_matrix mat, int full)
- static double map_atan_real (double d)
- static mess_double_cpx_t map_atan_cpx (mess_double_cpx_t z)
- int mess_matrix_map_atan (mess_matrix mat, int full)
- static double map_atanh_real (double d)
- static mess_double_cpx_t map_atanh_cpx (mess_double_cpx_t z)
- int mess_matrix_map_atanh (mess_matrix mat, int full)
- static double map_ceil_real (double d)
- static mess_double_cpx_t map_ceil_cpx (mess_double_cpx_t z)
- int mess_matrix_map_ceil (mess_matrix mat, int full)
- static double map_conj_real (double d)
- static mess_double_cpx_t map_conj_cpx (mess_double_cpx_t z)
- int mess_matrix_map_conj (mess_matrix mat, int full)
- static double map_cos_real (double d)
- static mess_double_cpx_t map_cos_cpx (mess_double_cpx_t z)
- int mess_matrix_map_cos (mess_matrix mat, int full)
- static double map_cosh_real (double d)
- static mess_double_cpx_t map_cosh_cpx (mess_double_cpx_t z)
- int mess_matrix_map_cosh (mess_matrix mat, int full)
- static double map_exp_real (double d)
- static mess_double_cpx_t map_exp_cpx (mess_double_cpx_t z)
- int mess_matrix_map_exp (mess_matrix mat, int full)
- static double map_expm1_real (double d)
- static mess_double_cpx_t map_expm1_cpx (mess_double_cpx_t z)
- int mess_matrix_map_expm1 (mess_matrix mat, int full)
- static double map_floor_real (double d)
- static mess_double_cpx_t map_floor_cpx (mess_double_cpx_t z)
- int mess_matrix_map_floor (mess_matrix mat, int full)
- static double map_log_real (double d)
- static mess_double_cpx_t map_log_cpx (mess_double_cpx_t z)
- int mess_matrix_map_log (mess_matrix mat, int full)
- static double map_round_real (double d)
- static mess_double_cpx_t map_round_cpx (mess_double_cpx_t z)
- int mess_matrix_map_round (mess_matrix mat, int full)
- static double map_sin_real (double d)
- static mess_double_cpx_t map_sin_cpx (mess_double_cpx_t z)
- int mess_matrix_map_sin (mess_matrix mat, int full)
- static double map_sinh_real (double d)
- static mess_double_cpx_t map_sinh_cpx (mess_double_cpx_t z)
- int mess_matrix_map_sinh (mess_matrix mat, int full)
- static double map_sqrt_real (double d)
- static mess_double_cpx_t map_sqrt_cpx (mess_double_cpx_t z)
- int mess_matrix_map_sqrt (mess_matrix mat, int full)
- static double map_tan_real (double d)
- static mess_double_cpx_t map_tan_cpx (mess_double_cpx_t z)
- int mess_matrix_map_tan (mess_matrix mat, int full)
- static double map_tanh_real (double d)
- static mess_double_cpx_t map_tanh_cpx (mess_double_cpx_t z)
- int mess_matrix_map_tanh (mess_matrix mat, int full)

- static double map_isfinite_real (double d)
- static mess_double_cpx_t map_isfinite_cpx (mess_double_cpx_t z)
- int mess_matrix_map_isfinite (mess_matrix mat, int full)
- static double map_isinf_real (double d)
- static mess_double_cpx_t map_isinf_cpx (mess_double_cpx_t z)
- int mess_matrix_map_isinf (mess_matrix mat, int full)
- static double map_isnan_real (double d)
- static mess_double_cpx_t map_isnan_cpx (mess_double_cpx_t z)
- int mess_matrix_map_isnan (mess_matrix mat, int full)
- static double map_not_real (double d)
- static mess_double_cpx_t map_not_cpx (mess_double_cpx_t z)
- int mess_matrix_map_not (mess_matrix mat, int full)

10.460.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.460.2 Macro Definition Documentation

10.460.2.1 #define MESS_MAP_STATIC_FUNC(*BASENAME, REALFUNC, CPXFUNC*)

Value:

```
static double                map_##BASENAME##_real(double d){return REALFUNC(d);}             \
    static mess_double_cpx_t     map_##BASENAME##_cpx(
      mess_double_cpx_t z){return CPXFUNC(z);}    \
    int mess_matrix_map_##BASENAME (mess_matrix mat, int full){
      \
        return mess_matrix_map(mat,map_##BASENAME##_real,map_##BASENAME##_cpx, full);
          \
    }                                                                                          \
```

10.460.2.2 #define MESS_MAP_STATIC_FUNC2(*BASENAME, REALFUNC*)

Value:

```
static double                map_##BASENAME##_real(double d){return REALFUNC(d);}
                  \
    static mess_double_cpx_t     map_##BASENAME##_cpx(
      mess_double_cpx_t z){return REALFUNC(creal(z))+I*REALFUNC(cimag(z));}  \
    int mess_matrix_map_##BASENAME (mess_matrix mat, int full){
                                    \
        return mess_matrix_map(mat,map_##BASENAME##_real,map_##BASENAME##_cpx,full);
                                        \
    }
                      \
```

10.460.2.3 #define MESS_MAP_STATIC_FUNC3(*BASENAME, REALFUNC*)

Value:

```
static double                map_##BASENAME##_real(double d){return REALFUNC(d);}
                                \
    static mess_double_cpx_t     map_##BASENAME##_cpx(
      mess_double_cpx_t z){return abs(REALFUNC(creal(z))) && abs(REALFUNC(cimag(z)));}   \
    int mess_matrix_map_##BASENAME (mess_matrix mat, int full){
                                                \
        return mess_matrix_map(mat,map_##BASENAME##_real,map_##BASENAME##_cpx,full);
                                                    \
    }
                                  \
```

10.460.3 Function Documentation

10.460.3.1 static mess_double_cpx_t map_abs_cpx (mess_double_cpx_t z) [static]

10.460.3.2 static double map_abs_real (double d) [static]

10.460.3.3 static mess_double_cpx_t map_acos_cpx (mess_double_cpx_t z) [static]

10.460.3.4 static double map_acos_real (double d) [static]

10.460.3.5 static mess_double_cpx_t map_acosh_cpx (mess_double_cpx_t z) [static]

10.460.3.6 static double map_acosh_real (double d) [static]

10.460.3.7 static mess_double_cpx_t map_arg_cpx (mess_double_cpx_t z) [static]

10.460.3.8 static double map_arg_real (double d) [static]

10.460.3.9 static mess_double_cpx_t map_asin_cpx (mess_double_cpx_t z) [static]

10.460.3.10 static double map_asin_real (double d) [static]

10.460.3.11 static mess_double_cpx_t map_asinh_cpx (mess_double_cpx_t z) [static]

10.460.3.12 static double map_asinh_real (double d) [static]

10.460.3.13 static mess_double_cpx_t map_atan_cpx (mess_double_cpx_t z) [static]

10.460.3.14 static double map_atan_real (double d) [static]

10.460.3.15 static mess_double_cpx_t map_atanh_cpx (mess_double_cpx_t z) [static]

10.460.3.16 static double map_atanh_real (double d) [static]

10.460.3.17 static mess_double_cpx_t map_ceil_cpx (mess_double_cpx_t z) [static]

10.460.3.18 static double map_ceil_real (double d) [static]

10.460.3.19 static mess_double_cpx_t map_conj_cpx (mess_double_cpx_t z) [static]

10.460.3.20 static double map_conj_real (double d) [static]

10.460.3.21 static mess_double_cpx_t map_cos_cpx (mess_double_cpx_t z) [static]

10.460.3.22 static double map_cos_real (double d) [static]

10.460.3.23 static mess_double_cpx_t map_cosh_cpx (mess_double_cpx_t z) [static]

10.460.3.24 static double map_cosh_real (double d) [static]

10.460.3.25 static mess_double_cpx_t map_exp_cpx (mess_double_cpx_t z) [static]

10.460.3.26 static double map_exp_real (double d) [static]

10.460.3.27 static mess_double_cpx_t map_expm1_cpx (mess_double_cpx_t z) [static]

10.460.3.28 static double map_expm1_real (double d) [static]

10.460.3.29 static mess_double_cpx_t map_floor_cpx (mess_double_cpx_t z) [static]

10.460.3.30 static double map_floor_real (double d) [static]

10.460.3.31 static mess_double_cpx_t map_isfinite_cpx (mess_double_cpx_t z) [static]

10.460.3.32 static double map_isfinite_real (double d) [static]

10.460.3.33 static mess_double_cpx_t map_isinf_cpx (mess_double_cpx_t z) [static]

10.460.3.34 static double map_isinf_real (double d) [static]

10.460.3.35 static mess_double_cpx_t map_isnan_cpx (mess_double_cpx_t z) [static]

10.460.3.36 static double map_isnan_real (double d) [static]

10.460.3.37 static mess_double_cpx_t map_log_cpx (mess_double_cpx_t z) [static]

10.460.3.38 static double map_log_real (double d) [static]

10.460.3.39 static mess_double_cpx_t map_not_cpx (mess_double_cpx_t z) [static]

10.460.3.40 static double map_not_real (double d) [static]

10.460.3.41 static mess_double_cpx_t map_round_cpx (mess_double_cpx_t z) [static]

10.460.3.42 static double map_round_real (double d) [static]

10.460.3.43 static mess_double_cpx_t map_sin_cpx (mess_double_cpx_t z) [static]

10.460.3.44 static double map_sin_real (double d) [static]

10.460.3.45 static mess_double_cpx_t map_sinh_cpx (mess_double_cpx_t z) [static]

10.460.3.46 static double map_sinh_real (double d) [static]

10.460.3.47 static mess_double_cpx_t map_sqrt_cpx (mess_double_cpx_t z) [static]

10.460.3.48 static double map_sqrt_real (double d) [static]

10.460.3.49 static mess_double_cpx_t map_tan_cpx (mess_double_cpx_t z) [static]

10.460.3.50 static double map_tan_real (double d) [static]

10.460.3.51 static mess_double_cpx_t map_tanh_cpx (mess_double_cpx_t z) [static]

10.460.3.52 static double map_tanh_real (double d) [static]

10.461 map.c File Reference

Macros

- #define MESS_MAP_STATIC_FUNC(BASENAME, REALFUNC, CPXFUNC)
- #define MESS_MAP_STATIC_FUNC2(BASENAME, REALFUNC)
- #define MESS_MAP_STATIC_FUNC3(BASENAME, REALFUNC)

Functions

- int mess_vector_map (mess_vector v, double(∗f_real)(double), mess_double_cpx_t(∗f_cpx)(mess_double↩ _cpx_t))

  Map a scalar function to a vector.
- static double map_abs_real (double d)
- static mess_double_cpx_t map_abs_cpx (mess_double_cpx_t z)
- int mess_vector_map_abs (mess_vector v)
- static double map_acos_real (double d)
- static mess_double_cpx_t map_acos_cpx (mess_double_cpx_t z)
- int mess_vector_map_acos (mess_vector v)
- static double map_acosh_real (double d)
- static mess_double_cpx_t map_acosh_cpx (mess_double_cpx_t z)
- int mess_vector_map_acosh (mess_vector v)
- static double map_arg_real (double d)
- static mess_double_cpx_t map_arg_cpx (mess_double_cpx_t z)
- int mess_vector_map_arg (mess_vector v)
- static double map_asin_real (double d)
- static mess_double_cpx_t map_asin_cpx (mess_double_cpx_t z)
- int mess_vector_map_asin (mess_vector v)
- static double map_asinh_real (double d)
- static mess_double_cpx_t map_asinh_cpx (mess_double_cpx_t z)
- int mess_vector_map_asinh (mess_vector v)
- static double map_atan_real (double d)
- static mess_double_cpx_t map_atan_cpx (mess_double_cpx_t z)
- int mess_vector_map_atan (mess_vector v)
- static double map_atanh_real (double d)
- static mess_double_cpx_t map_atanh_cpx (mess_double_cpx_t z)
- int mess_vector_map_atanh (mess_vector v)
- static double map_ceil_real (double d)
- static mess_double_cpx_t map_ceil_cpx (mess_double_cpx_t z)
- int mess_vector_map_ceil (mess_vector v)
- static double map_conj_real (double d)
- static mess_double_cpx_t map_conj_cpx (mess_double_cpx_t z)
- int mess_vector_map_conj (mess_vector v)
- static double map_cos_real (double d)
- static mess_double_cpx_t map_cos_cpx (mess_double_cpx_t z)
- int mess_vector_map_cos (mess_vector v)
- static double map_cosh_real (double d)
- static mess_double_cpx_t map_cosh_cpx (mess_double_cpx_t z)
- int mess_vector_map_cosh (mess_vector v)
- static double map_exp_real (double d)
- static mess_double_cpx_t map_exp_cpx (mess_double_cpx_t z)
- int mess_vector_map_exp (mess_vector v)
- static double map_expm1_real (double d)
- static mess_double_cpx_t map_expm1_cpx (mess_double_cpx_t z)
- int mess_vector_map_expm1 (mess_vector v)
- static double map_floor_real (double d)
- static mess_double_cpx_t map_floor_cpx (mess_double_cpx_t z)
- int mess_vector_map_floor (mess_vector v)
- static double map_log_real (double d)
- static mess_double_cpx_t map_log_cpx (mess_double_cpx_t z)
- int mess_vector_map_log (mess_vector v)
- static double map_round_real (double d)
- static mess_double_cpx_t map_round_cpx (mess_double_cpx_t z)

- int mess_vector_map_round (mess_vector v)
- static double map_sin_real (double d)
- static mess_double_cpx_t map_sin_cpx (mess_double_cpx_t z)
- int mess_vector_map_sin (mess_vector v)
- static double map_sinh_real (double d)
- static mess_double_cpx_t map_sinh_cpx (mess_double_cpx_t z)
- int mess_vector_map_sinh (mess_vector v)
- static double map_sqrt_real (double d)
- static mess_double_cpx_t map_sqrt_cpx (mess_double_cpx_t z)
- int mess_vector_map_sqrt (mess_vector v)
- static double map_tan_real (double d)
- static mess_double_cpx_t map_tan_cpx (mess_double_cpx_t z)
- int mess_vector_map_tan (mess_vector v)
- static double map_tanh_real (double d)
- static mess_double_cpx_t map_tanh_cpx (mess_double_cpx_t z)
- int mess_vector_map_tanh (mess_vector v)
- static double map_isfinite_real (double d)
- static mess_double_cpx_t map_isfinite_cpx (mess_double_cpx_t z)
- int mess_vector_map_isfinite (mess_vector v)
- static double map_isinf_real (double d)
- static mess_double_cpx_t map_isinf_cpx (mess_double_cpx_t z)
- int mess_vector_map_isinf (mess_vector v)
- static double map_isnan_real (double d)
- static mess_double_cpx_t map_isnan_cpx (mess_double_cpx_t z)
- int mess_vector_map_isnan (mess_vector v)
- static double map_not_real (double d)
- static mess_double_cpx_t map_not_cpx (mess_double_cpx_t z)
- int mess_vector_map_not (mess_vector v)

10.461.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.461.2 Macro Definition Documentation

10.461.2.1 #define MESS_MAP_STATIC_FUNC(*BASENAME, REALFUNC, CPXFUNC*)

Value:

```
\
    static double                map_##BASENAME##_real(double d){return REALFUNC(d);}            \
    static mess_double_cpx_t     map_##BASENAME##_cpx(
      mess_double_cpx_t z){return CPXFUNC(z);}    \
    int mess_vector_map_##BASENAME (mess_vector v){
       \
        return mess_vector_map(v,map_##BASENAME##_real,map_##BASENAME##_cpx);
          \
    }                                                                                            \
```

10.461.2.2 #define MESS_MAP_STATIC_FUNC2(*BASENAME*, *REALFUNC*)

Value:

```
static double                map_##BASENAME##_real(double d){return REALFUNC(d);}
                    \
    static mess_double_cpx_t     map_##BASENAME##_cpx(
      mess_double_cpx_t z){return REALFUNC(creal(z))+I*REALFUNC(cimag(z));}  \
    int mess_vector_map_##BASENAME (mess_vector v){
                                                \
        return mess_vector_map(v,map_##BASENAME##_real,map_##BASENAME##_cpx);
                                                  \
    }
                        \
```

10.461.2.3 #define MESS_MAP_STATIC_FUNC3(*BASENAME*, *REALFUNC*)

Value:

```
static double                map_##BASENAME##_real(double d){return REALFUNC(d);}
                                \
    static mess_double_cpx_t     map_##BASENAME##_cpx(
      mess_double_cpx_t z){return abs(REALFUNC(creal(z))) && abs(REALFUNC(cimag(z)));}   \
    int mess_vector_map_##BASENAME (mess_vector v){
                                                    \
        return mess_vector_map(v,map_##BASENAME##_real,map_##BASENAME##_cpx);
                                                      \
    }
                                    \
```

10.461.3 Function Documentation

10.461.3.1 static mess_double_cpx_t map_abs_cpx (mess_double_cpx_t *z*) [static]

10.461.3.2 static double map_abs_real (double *d*) [static]

10.461.3.3 static mess_double_cpx_t map_acos_cpx (mess_double_cpx_t *z*) [static]

10.461.3.4 static double map_acos_real (double *d*) [static]

10.461.3.5 static mess_double_cpx_t map_acosh_cpx (mess_double_cpx_t *z*) [static]

10.461.3.6 static double map_acosh_real (double *d*) [static]

10.461.3.7 static mess_double_cpx_t map_arg_cpx (mess_double_cpx_t *z*) [static]

10.461.3.8 static double map_arg_real (double *d*) [static]

10.461.3.9 static mess_double_cpx_t map_asin_cpx (mess_double_cpx_t *z*) [static]

10.461.3.10 static double map_asin_real (double *d*) [static]

10.461.3.11 static mess_double_cpx_t map_asinh_cpx (mess_double_cpx_t *z*) [static]

10.461.3.12 static double map_asinh_real (double *d*) [static]

10.461.3.13 static **mess_double_cpx_t** map_atan_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.14 static double map_atan_real (double *d*) [static]

10.461.3.15 static **mess_double_cpx_t** map_atanh_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.16 static double map_atanh_real (double *d*) [static]

10.461.3.17 static **mess_double_cpx_t** map_ceil_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.18 static double map_ceil_real (double *d*) [static]

10.461.3.19 static **mess_double_cpx_t** map_conj_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.20 static double map_conj_real (double *d*) [static]

10.461.3.21 static **mess_double_cpx_t** map_cos_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.22 static double map_cos_real (double *d*) [static]

10.461.3.23 static **mess_double_cpx_t** map_cosh_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.24 static double map_cosh_real (double *d*) [static]

10.461.3.25 static **mess_double_cpx_t** map_exp_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.26 static double map_exp_real (double *d*) [static]

10.461.3.27 static **mess_double_cpx_t** map_expm1_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.28 static double map_expm1_real (double *d*) [static]

10.461.3.29 static **mess_double_cpx_t** map_floor_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.30 static double map_floor_real (double *d*) [static]

10.461.3.31 static **mess_double_cpx_t** map_isfinite_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.32 static double map_isfinite_real (double *d*) [static]

10.461.3.33 static **mess_double_cpx_t** map_isinf_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.34 static double map_isinf_real (double *d*) [static]

10.461.3.35 static **mess_double_cpx_t** map_isnan_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.36 static double map_isnan_real (double *d*) [static]

10.461.3.37 static **mess_double_cpx_t** map_log_cpx (**mess_double_cpx_t** *z*) [static]

10.461.3.38 static double map_log_real (double *d*) [static]

10.461.3.39 static mess_double_cpx_t map_not_cpx (mess_double_cpx_t *z*) [static]

10.461.3.40 static double map_not_real (double *d*) [static]

10.461.3.41 static mess_double_cpx_t map_round_cpx (mess_double_cpx_t *z*) [static]

10.461.3.42 static double map_round_real (double *d*) [static]

10.461.3.43 static mess_double_cpx_t map_sin_cpx (mess_double_cpx_t *z*) [static]

10.461.3.44 static double map_sin_real (double *d*) [static]

10.461.3.45 static mess_double_cpx_t map_sinh_cpx (mess_double_cpx_t *z*) [static]

10.461.3.46 static double map_sinh_real (double *d*) [static]

10.461.3.47 static mess_double_cpx_t map_sqrt_cpx (mess_double_cpx_t *z*) [static]

10.461.3.48 static double map_sqrt_real (double *d*) [static]

10.461.3.49 static mess_double_cpx_t map_tan_cpx (mess_double_cpx_t *z*) [static]

10.461.3.50 static double map_tan_real (double *d*) [static]

10.461.3.51 static mess_double_cpx_t map_tanh_cpx (mess_double_cpx_t *z*) [static]

10.461.3.52 static double map_tanh_real (double *d*) [static]

10.462 matrix.c File Reference

Macros

- #define NPY_NO_DEPRECATED_API NPY_1_7_API_VERSION
- #define PY_ARRAY_UNIQUE_SYMBOL MESS_VECTOR_MATRIX_PYTHON_C_API
- #define NO_IMPORT_ARRAY
- #define ConvStringtoC(X) PyBytes_AsString(X)
- #define NPY_ARRAY_OWNDATA NPY_OWNDATA
- #define PyArray_ENABLEFLAGS(arr, flag) ((PyArrayObject∗)(arr))->flags |= (flag)
- #define PyArray_CALL_ISSIGNED(x) PyArray_ISSIGNED((PyArrayObject∗)(x))
- #define PyArray_CALL_ISUNSIGNED(x) PyArray_ISUNSIGNED((PyArrayObject∗)(x))
- #define PyArray_CALL_DATA(x) PyArray_DATA((PyArrayObject∗)(x))

Functions

- mess_matrix matrix_to_c (PyObject ∗data)

  Converts `Python` *matrix to mess_matrix.*
- PyObject ∗ matrix_to_python (mess_matrix c_matrix)

  Converts mess_matrix to `Python` *matrix.*

10.462.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Nitin Prasad (nitin.prasad93@gmail.com, MPI Magdeburg, CSC)

10.462.2 Macro Definition Documentation

10.462.2.1 #define ConvStringtoC(X) PyBytes_AsString(X)

10.462.2.2 #define NO_IMPORT_ARRAY

10.462.2.3 #define NPY_ARRAY_OWNDATA NPY_OWNDATA

10.462.2.4 #define NPY_NO_DEPRECATED_API NPY_1_7_API_VERSION

10.462.2.5 #define PY_ARRAY_UNIQUE_SYMBOL MESS_VECTOR_MATRIX_PYTHON_C_API

10.462.2.6 #define PyArray_CALL_DATA(x) PyArray_DATA((PyArrayObject∗)(x))

10.462.2.7 #define PyArray_CALL_ISSIGNED(x) PyArray_ISSIGNED((PyArrayObject∗)(x))

10.462.2.8 #define PyArray_CALL_ISUNSIGNED(x) PyArray_ISUNSIGNED((PyArrayObject∗)(x))

10.462.2.9 #define PyArray_ENABLEFLAGS(*arr, flag*) ((PyArrayObject∗)(arr))->flags |= (flag)

10.463 matrix.h File Reference

Data Structures

- struct mess_matrix_st

  The mess_matrix structure to handle dense and sparse matrices in a common context.
- struct mess_matrix_file_st

  Type definition and structure for mess_matrix files.
- struct mess_mvpcall_st

  Represent a generic matrix-vector-product.

Macros

- #define MESS_IS_CSR(c) ((∗c).store_type == MESS_CSR)

  Macro to check if a matrix is stored in MESS_CSR storage.
- #define MESS_IS_CSC(c) ((∗c).store_type == MESS_CSC)

  Macro to check if a matrix is stored in MESS_CSC storage.
- #define MESS_IS_DENSE(c) ((∗c).store_type == MESS_DENSE)

  Macro to check if a matrix is stored in MESS_DENSE storage.
- #define MESS_IS_COORD(c) ((∗c).store_type == MESS_COORD)

  Macro to check if a matrix is stored in MESS_COORD storage.
- #define MESS_IS_SPARSE(c) (MESS_IS_CSR(c) || MESS_IS_CSC(c) || MESS_IS_COORD(c))

  Macro to check if a matrix is stored in some sparse storage.
- #define MESS_IS_GENERAL(m) ((∗m).symmetry == MESS_GENERAL)

  Check if a matrix is a general one, that means it has no special symmetry.
- #define MESS_IS_UNSYMMETRIC(m) ((∗m).symmetry == MESS_GENERAL)

  Check if a matrix is a general one.
- #define MESS_IS_SYMMETRIC(m) ((∗m).symmetry == MESS_SYMMETRIC)

  Check if a matrix is stored symmetric.
- #define MESS_IS_SKEWSYMMETRIC(m) ((∗m).symmetry == MESS_SKEWSYMMETRIC)

  Check if a matrix is stored skew-symmetric.
- #define MESS_IS_HERMITIAN(m) ((∗m).symmetry == MESS_HERMITIAN)

  Check if a matrix is stored symmetric.
- #define MESS_MATRIX_FILE_MAGIC {0x63,0x6D,0x65,0x73,0x73,0x4D}
- #define MESS_MATRIX_FILE_MAGIC_SET(m) {m[0]=0x63; m[1]=0x6D; m[2]=0x65; m[3]=0x73 ; m[4]=0x73 ; m[5]=0x4D; }
- #define MESS_MATRIX_FILE_MAGIC_CHECK(m) ((m[0] == 0x63)&&(m[1]==0x6D) &&(m[2]==0x65)&&(m[3]==0x73)&&(m[4] D))
- #define MESS_MATRIX_FILE_VERSION 0x01
- #define MESS_MATRIX_CHECKFORMAT(input, output, status, fmt)

  Check if a matrix is stored in the correct storage format and converts it if necessary.
- #define MESS_MATRIX_RESET(matrix) do { mess_matrix_reset((matrix)); } while(0)

  Resets a mess_matrix object.

Typedefs

- typedef struct mess_matrix_st mess_matrix_st

  The mess_matrix structure to handle dense and sparse matrices in a common context.
- typedef struct mess_matrix_st ∗ mess_matrix

  Type definition to make mess_matrix_st easier to use.
- typedef struct mess_matrix_file_st mess_matrix_file

  Type definition and structure for mess_matrix files.
- typedef struct mess_mvpcall_st ∗ mess_mvpcall

  Represent a generic matrix-vector-product.
- typedef double(∗ mess_matgen_fdm_function) (double x, double y)

  Type definition for the f_x, f_y and g function in the FDM operator.

Enumerations

Functions

- int mess_matrix_clear (mess_matrix *matrix)

  Remove a mess_matrix object from memory.
- int mess_matrix_init (mess_matrix *matrix)

  Initialize the mess_matrix structure.
- int mess_matrix_read (const char *filename, mess_matrix matrix)

  Read a matrix from a file into a mess_matrix structure.
- int mess_matrix_write (const char *filename, mess_matrix matrix)

  Write a mess_matrix to a `MatrixMarket` *file.*
- int mess_matrix_read_formated (const char *filename, mess_matrix matrix, mess_storage_t format)

  Read a matrix from a file and convert it to the desired format.
- int mess_matrix_display (mess_matrix matrix)

  Display a mess_matrix to stdout.
- int mess_matrix_print (mess_matrix matrix)

  Print a mess_matrix to stdout.
- int mess_matrix_printshort (mess_matrix matrix)

  Print the first values of a matrix to stdout.
- int mess_matrix_printdata (mess_matrix mat)

  Print underlying data structures of a matrix to stdout.
- int mess_matrix_printinfo (mess_matrix mat)

  Print basic information of a matrix to the stdout.
- int mess_matrix_alloc (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t nnz, mess_storage_t storetype, mess_datatype_t datatype)

  Allocate a sparse or dense $(m \times n)$ *matrix.*
- void mess_matrix_alloc_setld (mess_int_t ld_m)
- int mess_matrix_realloc (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t nnz)

  Reallocate or resize a $(m \times n)$ *matrix.*
- int mess_matrix_resize (mess_matrix matrix, mess_int_t rows, mess_int_t cols)

  Wrapper for mess_matrix_realloc.
- int mess_matrix_reset (mess_matrix matrix)

  Resets a mess_matrix object.
- int mess_matrix_convert (mess_matrix input, mess_matrix output, mess_storage_t outtype)

  Convert the storage type of a matrix (universal converter interface).
- int mess_matrix_sort (mess_matrix mat)

  Sort the internal structure of a sparse matrix.
- int mess_matrix_cat (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix D, mess_storage_t otype, mess_matrix O)

  Concatenate up to four matrices.
- int mess_matrix_rowsub (mess_matrix input, mess_int_t srow, mess_int_t erow, mess_matrix out)

  Copy a row-block out of a matrix.
- int mess_matrix_colsub (mess_matrix input, mess_int_t scol, mess_int_t ecol, mess_matrix out)

  Copy a column block out of a matrix.
- int mess_matrix_sub (mess_matrix in, mess_int_t rowS, mess_int_t rowE, mess_int_t colS, mess_int_t colE, mess_matrix out)

  Copy a sub matrix out of a matrix.
- int mess_matrix_lift (mess_matrix in, mess_int_t n, mess_matrix out)

  Add lower trailing block of n rows to matrix.
- int mess_matrix_realpart (mess_matrix matrix, mess_matrix realpart)

Get the real part of a matrix.

- int mess_matrix_imagpart (mess_matrix matrix, mess_matrix imagpart)

  Get the imaginary part of a matrix.
- int mess_matrix_complex_from_parts (mess_matrix xr, mess_matrix xc, mess_matrix x)

  Create a complex matrix from real and imaginary parts.
- int mess_matrix_getelement (mess_matrix matrix, mess_int_t row, mess_int_t col, double *val, mess_double_cpx_t *valc)

  Get an element from a matrix.
- int mess_matrix_setelement_complex (mess_matrix matrix, mess_int_t row, mess_int_t col, mess_double_cpx_t value)

  Set a complex entry in a matrix.
- int mess_matrix_setelement (mess_matrix matrix, mess_int_t row, mess_int_t col, double value)

  Set a real entry in a matrix.
- int mess_matrix_convert_csr_csc (mess_matrix inmatrix, mess_matrix outmatrix)

  Convert a Compressed Sparse Row matrix to a Compressed Sparse Column matrix.
- int mess_matrix_convert_csc_csr (mess_matrix inmatrix, mess_matrix outmatrix)

  Convert a Compressed Sparse Column matrix to a Compressed Sparse Row matrix.
- int mess_matrix_symfillup (mess_matrix mat)

  Fill up the symmetry in a coordinate matrix.
- int mess_matrix_dense_from_farray (mess_matrix mat, mess_int_t rows, mess_int_t cols, mess_int_t ld, double *realv, mess_double_cpx_t *complexv)

  Create a MESS_DENSE mess_matrix from a `Fortran` array.
- int mess_matrix_dense_from_carray (mess_matrix mat, mess_int_t rows, mess_int_t cols, double **realv, mess_double_cpx_t **complexv)

  Create a MESS_DENSE mess_matrix from a `C` 2-dimensional array.
- int mess_matrix_csr (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t *rowptr, mess_int_t *colptr, double *values, mess_double_cpx_t *values_cpx)

  Create a MESS_CSR mess_matrix from given arrays.
- int mess_matrix_csc (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t *rowptr, mess_int_t *colptr, double *values, mess_double_cpx_t *values_cpx)

  Create a MESS_CSC mess_matrix from given arrays.
- int mess_matrix_coord (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t nnz, mess_int_t onebased, mess_int_t *rowptr, mess_int_t *colptr, double *values, mess_double_cpx_t *values_cpx)

  Create a MESS_COORD mess_matrix from given arrays.
- int mess_matrix_eye (mess_matrix A, mess_int_t rows, mess_int_t cols, mess_storage_t storetype)

  Generate a real identity matrix.
- int mess_matrix_eyec (mess_matrix mat, mess_int_t rows, mess_int_t cols, mess_storage_t store)

  Generate a complex identity matrix.
- int mess_matrix_tridiag (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_storage_t store_t, mess_datatype_t data_t, mess_double_cpx_t lower, mess_double_cpx_t diag, mess_double_cpx_t upper)

  Create a tridiagonal matrix.
- int mess_matrix_rand_init (mess_int_t *seed)

  Init random number generator with given seed.
- int mess_matrix_rand (mess_matrix mat, mess_int_t rows, mess_int_t cols, mess_storage_t storetype, mess_datatype_t dt, double p)

  Wrapper around mess_matrix_rand_csr, mess_matrix_rand_csc, mess_matrix_rand_dense and mess_matrix_rand_coord.
- int mess_matrix_rand_dense (mess_matrix mat, mess_int_t rows, mess_int_t cols, mess_datatype_t dt)

  Generate a MESS_REAL MESS_COMPLEX random dense matrix.
- int mess_matrix_rand_dense_uniform (mess_matrix mat, mess_int_t *seed, mess_int_t rows, mess_int_t cols)

  Generate a random MESS_REAL dense matrix with a uniform value distribution $(-1, 1)$.

- int mess_matrix_rand_csr (mess_matrix mat, mess_int_t rows, mess_int_t cols, double p, mess_datatype_t dt)

  Generate a MESS_REAL/MESS_COMPLEX random MESS_CSR matrix.
- int mess_matrix_rand_csc (mess_matrix mat, mess_int_t rows, mess_int_t cols, double p, mess_datatype_t dt)

  Generate a MESS_REAL/MESS_COMPLEX random MESS_CSC matrix.
- int mess_matrix_rand_coord (mess_matrix mat, mess_int_t rows, mess_int_t cols, double p, mess_datatype_t dt)

  Generate a MESS_REAL/MESS_COMPLEX random COORD matrix.
- int mess_matrix_dupl (mess_matrix mat)

  Detect and sum up duplicate entries in a sparse matrix.
- int mess_matrix_eliminate_zeros (mess_matrix mat)

  Remove zero entries in a sparse matrix.
- int mess_matrix_zeros (mess_matrix matrix)

  Set all entries in a matrix to zero.
- int mess_matrix_ones (mess_matrix matrix)

  Set all entries in a matrix to one.
- int mess_matrix_one_value (mess_matrix matrix, double value)

  Set all entries of a matrix to one value.
- int mess_matrix_one_valuec (mess_matrix matrix, mess_double_cpx_t value)

  Set all entries of a matrix to one value (complex version).
- int mess_matrix_need_alloc (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_int_t nnz, mess_storage_t storetype, mess_datatype_t datatype)

  Check if a new allocation of a matrix is necessary.
- int mess_matrix_copy (mess_matrix in, mess_matrix out)

  Copy a matrix to another one.
- int mess_matrix_getcol (mess_matrix matrix, mess_int_t col, mess_vector c)

  Get a column from a matrix.
- int mess_matrix_getrow (mess_matrix matrix, mess_int_t row, mess_vector r)

  Get a row from a matrix.
- int mess_matrix_setcol (mess_matrix matrix, mess_int_t col, mess_vector colv)

  Set a column in a matrix.
- int mess_matrix_triu (mess_matrix mat, mess_int_t k)

  Upper triangular part of mess_matrix.
- int mess_matrix_tril (mess_matrix mat, mess_int_t k)

  Lower triangular part of mess_matrix.
- int mess_matrix_rowscalem (mess_vector r, mess_matrix mat)

  Scale rows of a matrix.
- int mess_matrix_colscalem (mess_vector c, mess_matrix mat)

  Scale cols of a matrix.
- int mess_matrix_setrow (mess_matrix matrix, mess_int_t row, mess_vector rowv)

  Set a row in a matrix.
- int mess_matrix_diag (mess_matrix matrix, mess_vector d)

  Get the main diagonal elements of from matrix.
- int mess_matrix_diag_from_vector (mess_vector v, mess_matrix diag)

  Create a diagonal mess_matrix from mess_vector.
- int mess_matrix_diagpos (mess_matrix matrix, mess_int_t *pos)

  Find the position of the main diagonal elements.
- int mess_matrix_addcols (mess_matrix matrix, mess_matrix toadd)

  Add columns to an existing matrix.
- int mess_matrix_addcols2p (mess_matrix Z, double s1, mess_matrix V1, double s2, mess_matrix V2)

Add the sum of two column matrices to a real matrix.

- int mess_matrix_addcols1 (mess_matrix Z, double s1, mess_matrix V1)

  Add a scaled column to a real matrix.
- int mess_matrix_toreal (mess_matrix m)

  Change the data type of a matrix to real.
- int mess_matrix_tocomplex (mess_matrix m)

  Change the data type of a matrix to complex.
- int mess_matrix_totype (mess_matrix mat, mess_datatype_t dt)

  Convert a matrix to a specified data type.
- int mess_matrix_view (mess_matrix original, mess_int_t srow, mess_int_t scol, mess_int_t rows, mess_int_t cols, mess_matrix view)
- int mess_matrix_bandwidth (mess_matrix matrix, mess_int_t *kl, mess_int_t *ku)

  Determine the bandwidth of a matrix.
- int mess_matrix_norm1 (mess_matrix A, double *nrm)

  Compute the 1-norm of a matrix.
- int mess_matrix_norm2 (mess_matrix A, double *nrm)

  Compute the 2-norm of a matrix.
- int mess_matrix_norm2inv (mess_matrix A, double *nrm)

  Compute the 2-norm of the inverse of a matrix.
- int mess_matrix_normf (mess_matrix A, double *nrm)

  Compute the Frobenius-norm of a matrix.
- int mess_matrix_normf2 (mess_matrix A, double *nrm)

  Compute the square of the Frobenius-norm of a matrix.
- int mess_matrix_norminf (mess_matrix A, double *nrm)

  Compute the ∞-norm of a matrix.
- int mess_matrix_norm (mess_matrix A, mess_norm_t nrm_t, double *nrm)

  Compute the norm of a matrix.
- int mess_matrix_diffnorm (mess_matrix A, mess_matrix B, double *nrm)

  Compute the 2-norm of the difference of two matrices.
- int mess_matrix_diffnormf (mess_matrix A, mess_matrix B, double *nrm)

  Compute the Frobenius norm of the difference of two matrices.
- int mess_matrix_dynorm2 (mess_matrix A, double *nrm)

  Compute the 2-norm of $A^H A$ or AA^H.
- int mess_matrix_dynormf (mess_matrix A, double *nrm)

  Compute the Frobenius-norm of $A^H A$ or AA^H.
- int mess_matrix_dynormf2 (mess_matrix A, double *nrm)

  Compute the squared Frobenius-norm of $A^H A$ or AA^H.
- int mess_matrix_indefinite_dynorm2 (mess_matrix W, mess_matrix K, double *nrm)

  Compute the 2-norm of $WW^T - KK^T$, $WW^H - KK^T$, $WW^T - KK^H$ or $WW^H - KK^H$.
- int mess_matrix_indefinite_dynormf (mess_matrix W, mess_matrix K, double *nrm)

  Compute the Frobenius-norm of $WW^T - KK^T$, $WW^H - KK^T$, $WW^T - KK^H$ or $WW^H - KK^H$.
- int mess_matrix_indefinite_dynormf2 (mess_matrix W, mess_matrix K, double *nrm)

  Compute the square of the Frobenius-norm of $WW^T - KK^T$, $WW^H - KK^T$, $WW^T - KK^H$ or $WW^H - KK^H$.
- int mess_matrix_mulnormf2 (mess_operation_t op1, mess_matrix A, mess_operation_t op2, mess_matrix B, double *nrm)

  Compute the square of the Frobenius-norm of a $op1(A) * op2(B)$.
- int mess_matrix_ctranspose (mess_matrix A, mess_matrix AH)

  Transpose a matrix.
- int mess_matrix_transpose (mess_matrix A, mess_matrix AT)

  Transpose a matrix.
- int mess_matrix_xtranspose (mess_matrix A, mess_matrix AT, int hermitian)

Transpose a matrix.

- int mess_matrix_conj (mess_matrix A)

  Conjugate a matrix.
- int mess_matrix_proj_sym (mess_matrix matrix)

  Get projection of a matrix on symmetric matrix space.
- int mess_matrix_rank (mess_matrix A, mess_int_t ∗rank)

  Estimate the rank of a matrix.
- int mess_matrix_orth (mess_matrix A, mess_matrix Q)

  Compute an orthogonal basis for the range of a matrix.
- int mess_matrix_null (mess_matrix A, mess_matrix Z)

  Compute an orthogonal basis for the nullspace of a matrix.
- int mess_matrix_mgs (mess_matrix A, mess_matrix Q, mess_matrix R)

  Perform a modified Gram-Schmidt orthogonalization.
- int mess_matrix_mgs_inplace (mess_matrix Q)

  Perform a modified Gram-Schmidt orthogonalization inplace on a matrix.
- int mess_matrix_mgs_add (mess_matrix Z, mess_matrix V, mess_matrix E)

  Use the modified Gram-Schmidt orthogonalization to add new columns to an orthogonal matrix.
- int mess_matrix_mvp (mess_operation_t op, mess_matrix A, mess_vector x, mess_vector y)

  Compute a matrix-vector product.
- int mess_matrix_gaxpy (mess_operation_t op, mess_matrix A, mess_vector x, mess_vector y)

  Compute a matrix-vector product with update of the output.
- int mess_matrix_colsums (mess_matrix A, mess_vector x)

  Compute the column sums of a matrix.
- int mess_matrix_rowsums (mess_matrix A, mess_vector x)

  Compute the row sums of a matrix.
- int mess_matrix_add (double alpha, mess_matrix A, double beta, mess_matrix B)

  Add two real matrices.
- int mess_matrix_addc (mess_double_cpx_t alpha, mess_matrix A, mess_double_cpx_t beta, mess_matrix B)

  Add two matrices.
- int mess_matrix_multiply (mess_operation_t opA, mess_matrix A, mess_operation_t opB, mess_matrix B, mess_matrix C)
- int mess_matrix_scale (double alpha, mess_matrix mat)

  Scale a real matrix by a scalar.
- int mess_matrix_scalec (mess_double_cpx_t alpha, mess_matrix mat)

  Scale a complex matrix by a scalar.
- int mess_matrix_fro_inn (mess_operation_t op, mess_matrix A, mess_matrix B, void ∗fro)

  Compute frobenius inner product of two dense matrices.
- int mess_matrix_trace (mess_matrix A, double ∗tr)

  Compute the trace of a real matrix.
- int mess_matrix_tracec (mess_matrix A, mess_double_cpx_t ∗tr)

  Compute the trace of a complex matrix.
- int mess_matrix_biorth (mess_matrix Vin, mess_matrix Win, mess_matrix Vout, mess_matrix Wout)

  Construct two biorthonormal matrices from two given matrices.
- int mess_matrix_gbiorth (mess_matrix E, mess_matrix Vin, mess_matrix Win, mess_matrix Vout, mess_matrix Wout)

  Construct two biorthonormal matrices from two given matrices with respect to a weighted scalar product.
- int mess_matrix_proj_mgs (mess_vector in, mess_matrix Q, mess_matrix QT, mess_int_t colcount, mess_vector out, double ∗nrmout)

  Skew projection using Q and QT.

- int mess_matrix_proj_gmgs (mess_vector in, mess_matrix Q, mess_matrix QT, mess_matrix E, char *Eflag, mess_int_t colcount, mess_vector out, double *nrmout)

  Generalized skew projection using Q and QT.
- int mess_matrix_map (mess_matrix mat, double(*f_real)(double), mess_double_cpx_t(*f_cpx)(mess_double_cpx_t), int full)

  Map a scalar function to a matrix.
- int mess_matrix_map_abs (mess_matrix mat, int full)
- int mess_matrix_map_acos (mess_matrix mat, int full)
- int mess_matrix_map_acosh (mess_matrix mat, int full)
- int mess_matrix_map_arg (mess_matrix mat, int full)
- int mess_matrix_map_asin (mess_matrix mat, int full)
- int mess_matrix_map_asinh (mess_matrix mat, int full)
- int mess_matrix_map_atan (mess_matrix mat, int full)
- int mess_matrix_map_atanh (mess_matrix mat, int full)
- int mess_matrix_map_ceil (mess_matrix mat, int full)
- int mess_matrix_map_conj (mess_matrix mat, int full)
- int mess_matrix_map_cos (mess_matrix mat, int full)
- int mess_matrix_map_cosh (mess_matrix mat, int full)
- int mess_matrix_map_exp (mess_matrix mat, int full)
- int mess_matrix_map_expm1 (mess_matrix mat, int full)
- int mess_matrix_map_floor (mess_matrix mat, int full)
- int mess_matrix_map_log (mess_matrix mat, int full)
- int mess_matrix_map_not (mess_matrix mat, int full)
- int mess_matrix_map_round (mess_matrix mat, int full)
- int mess_matrix_map_sin (mess_matrix mat, int full)
- int mess_matrix_map_sinh (mess_matrix mat, int full)
- int mess_matrix_map_sqrt (mess_matrix mat, int full)
- int mess_matrix_map_tan (mess_matrix mat, int full)
- int mess_matrix_map_tanh (mess_matrix mat, int full)
- int mess_matrix_map_isfinite (mess_matrix mat, int full)
- int mess_matrix_map_isinf (mess_matrix mat, int full)
- int mess_matrix_map_isnan (mess_matrix mat, int full)
- int mess_matrix_any (mess_matrix mat, mess_int_t(*f_real)(double), mess_int_t(*f_cpx)(mess_double_cpx_t), mess_int_t dim, mess_vector anyvec)

  Checks a matrix rowwise/columnwise if a given predicate is true.
- mess_int_t mess_matrix_memsize (mess_matrix matrix)

  Return the size of a matrix in bytes.
- mess_int_t mess_matrix_memsize_nnz (mess_int_t rows, mess_int_t cols, mess_int_t nnz, mess_int_t type, mess_int_t dtype)

  Determine memory size of a matrix from parameters.
- int mess_matrix_spy (mess_matrix matrix, const char *filename, int width, int height)

  Generate a bitmap file from the non zero structure of a matrix.
- int mess_matrix_reorder (mess_reorder_t reorder, mess_matrix matrix, mess_int_t *p, mess_int_t *q)

  Select and compute a matrix reordering.
- int mess_matrix_reorder_amd (mess_matrix matrix, mess_int_t *p)

  Compute `AMD` *reordering of a matrix.*
- int mess_matrix_reorder_colamd (mess_matrix matrix, mess_int_t *p)

  Compute the `COLAMD` *reordering of a matrix.*
- int mess_matrix_reorder_rcm (mess_matrix A, mess_int_t *perm)

  Compute the Reverse Cuthill-McKee reordering of a matrix.
- int mess_matrix_perm (mess_matrix matrix, mess_int_t *p, mess_int_t *q)

  Permute a general matrix.
- int mess_matrix_permcopy (mess_matrix matrix, mess_int_t *p, mess_int_t *q, mess_matrix out)

Copy and permute a matrix.

- int mess_matrix_colperm (mess_matrix A, mess_int_t *perm)

  Permute the columns of a matrix inplace.
- int mess_matrix_colpermcopy (mess_matrix A, mess_int_t *perm, mess_matrix B)

  Permute the columns of a matrix while copying.
- int mess_matrix_joinpatterns (mess_matrix A, mess_matrix B)

  Join the pattern of two sparse matrices.
- int mess_matrix_equalpattern (mess_matrix A, mess_matrix B)

  Check if two matrices have the same pattern.
- int mess_vector_tomatrix (mess_vector v, mess_matrix mat)

  Convert a vector to a matrix.
- int mess_vector_frommatrix (mess_matrix mat, mess_vector v)

  Convert a single column matrix to a vector.
- int mess_matrix_equal (mess_matrix mat1, mess_matrix mat2)

  Check if two matrices are equal on data structure level.
- int mess_matrix_equal_verbose (mess_matrix mat1, mess_matrix mat2)

  Check if two matrices are equal on data structure level (with verbose output).
- int mess_matrix_decomp (mess_matrix A, mess_matrix Asym, mess_matrix Askewsym)

  Decompose a matrix in a symmetric (hermitian) and skewsymmetric (skewhermitian) part.
- int mess_matrix_colnorm (mess_matrix Q, mess_int_t col, double *norm)

  Compute the euclidean norm of a matrix column.
- int mess_matrix_colscale (mess_matrix Q, mess_int_t col, mess_double_cpx_t scale)

  Scale a column of a matrix.
- int mess_matrix_coldot (mess_matrix Q, mess_int_t col1, mess_int_t col2, double *dot)

  Compute the dot product between two columns of a real matrix.
- int mess_matrix_coldotc (mess_matrix Q, mess_int_t col1, mess_int_t col2, mess_double_cpx_t *dot)

  Compute the dot product between two columns of a complex matrix.
- int mess_matrix_colaxpy (mess_double_cpx_t alpha, mess_vector vect, mess_int_t col, mess_matrix Q)

  Perform an axpy-update on a column of a matrix.
- int mess_matrix_colaxpy2 (mess_matrix Q, mess_double_cpx_t alpha, mess_int_t colc, mess_int_t col1)

  Update a column of a matrix by α times another column.
- int mess_matrix_colvecdot (mess_matrix Q, mess_int_t col, mess_vector v, double *dot)

  Compute the dot product between a matrix column and a vector (real).
- int mess_matrix_colvecdotc (mess_matrix Q, mess_int_t col, mess_vector v, mess_double_cpx_t *dot)

  Computes the dot product between a matrix column and a vector (complex).
- int mess_matrix_colvecaxpy (mess_double_cpx_t alpha, mess_int_t col, mess_matrix Q, mess_vector v)

  Update a vector using a column of a matrix.
- int mess_matrix_coldotE (mess_matrix Q, mess_matrix E, mess_int_t col1, mess_int_t col2, double *dot)

  Compute the weighted dot product of two matrix columns (real).
- int mess_matrix_coldotcE (mess_matrix Q, mess_matrix E, mess_int_t col1, mess_int_t col2, mess_double↩_cpx_t *dot)

  Compute the weighted dot product of two matrix columns (complex).
- int mess_matrix_colnormE (mess_matrix Q, mess_matrix E, mess_int_t col, double *dot)

  Compute the E norm of a matrix column.
- int mess_graph_reach (mess_matrix G, mess_matrix B, mess_int_t k, mess_int_t *top, mess_int_t *xi, const mess_int_t *pinv)

  Compute the reachability set from a node in a graph.
- int mess_graph_dfs (mess_matrix G, mess_int_t j, mess_int_t *top, mess_int_t *xi, mess_int_t *pstack, const mess_int_t *pinv)

  Perform a depth-first-search on a directed graph defined by a matrix.
- int mess_graph_lblockpart (mess_matrix G, mess_int_t *perm, mess_int_t *nblocks, mess_int_t *blocks)

Compute a lower triangular block partition of a matrix.

- int mess_mvpcall_matrix (mess_mvpcall *mvpcall, mess_operation_t op, mess_matrix A)

  Generate a mess_mvpcall object for a simple matrix.
- int mess_mvpcall_operator (mess_mvpcall *mvpcall, mess_int_t dim, mess_datatype_t data_type, int(*mvp)(void *data, mess_operation_t op, mess_vector x, mess_vector y), void *data)

  Generate a mess_mvpcall object for a complex operator.
- int mess_mvpcall_apply (mess_mvpcall mvpcall, mess_operation_t op, mess_vector x, mess_vector y)

  Apply a generic matrix-vector product.
- int mess_mvpcall_clear (mess_mvpcall *mvpcall)

  Clean a mess_mvpcall object .
- int mess_matgen_fdmmatrix (mess_matrix A, mess_int_t n0, mess_matgen_fdm_function fnfx, mess_matgen_fdm_function fnfy, mess_matgen_fdm_function fng)

  Generate a stiffness matrix A for the finite difference discretization (equidistant grid) of the PDE.
- int mess_matgen_fdmvector (mess_vector v, mess_int_t n0, mess_matgen_fdm_function func)

  Generate a vector v containing values of a function $f(x,y)$ on an equidistant grid in the interior of the unit square.
- int mess_matgen_fdmcolumn (mess_matrix B, mess_int_t n0, mess_matgen_fdm_function func)

  Generate a one-column matrix containing values of a function $f(x,y)$ on an equidistant grid in the interior of the unit square.
- int mess_matgen_fdmrow (mess_matrix C, mess_int_t n0, mess_matgen_fdm_function func)

  Generates a one-row matrix containing values of a function $f(x,y)$ on an equidistant grid in the interior of the unit square.

10.463.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.464 matrix_print.c File Reference

Functions

- int mess_matrix_display (mess_matrix matrix)

  Display a mess_matrix to stdout.
- int mess_matrix_print (mess_matrix matrix)

  Print a mess_matrix to stdout.
- int mess_matrix_printshort (mess_matrix matrix)

  Print the first values of a matrix to stdout.
- int mess_matrix_printdata (mess_matrix mat)

  Print underlying data structures of a matrix to stdout.
- int mess_matrix_printinfo (mess_matrix matrix)

  Print basic information of a matrix to the stdout.

10.464.1 Detailed Description

Print formatting.

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.465 memsize.c File Reference

Functions

- mess_int_t mess_vector_memsize (mess_vector v)

  Return the size of a mess_vector in bytes.

10.465.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.466 mess.h File Reference

Macros

- #define DEPRECATED __attribute__ ((deprecated))
- #define MESS_PRINTF_INT "%d"

  String to print a mess_int_t inside printf.
- #define MESS_PRINTF_INT2D "%2d"

  String to print a mess_int_t with 2 digits inside printf.
- #define MESS_INTEGER MESS_REAL

  Define for compatibility with the MatrixMarket file format.
- #define MESS_IS_REAL(m) ((*m).data_type == MESS_REAL)

  Check if a mess_matrix, mess_vector or mess_direct object is real.
- #define MESS_IS_INTEGER(m) ((*m).data_type == MESS_REAL)

  Check if a mess_matrix, mess_vector or mess_direct object is real.
- #define MESS_IS_COMPLEX(m) ((*m).data_type == MESS_COMPLEX)

  Check if a mess_matrix, mess_vector or mess_direct object is complex.

Typedefs

- typedef double _Complex mess_double_cpx_t

  *Default double complex of **M.E.S.S.**.*
- typedef float _Complex mess_float_cpx_t

  *Default float complex of **M.E.S.S.**.*
- typedef int mess_int_t

  *Default integer of **M.E.S.S.**.*
- typedef unsigned short MESS_STORETYPE

  *Type definition for the data type in **M.E.S.S.**.*
- typedef unsigned short MESS_SYMMETRY

  *Type definition for the data type in **M.E.S.S.**.*

Enumerations

10.466.1 Detailed Description

This file is the main header file for **M.E.S.S.**.
Please include this file in every programm you use **M.E.S.S.**.

10.467 mess_call.c File Reference

Data Structures

- struct _mess_call

Typedefs

- typedef void(∗ mess_call_func) (int nlhs, mxArray ∗plhs[ ], int nrhs, const mxArray ∗prhs[ ])
- typedef struct _mess_call mess_call_t

Functions

- void mexFunction (int nlhs, mxArray ∗plhs[ ], int nrhs, const mxArray ∗prhs[ ])

Variables

- static mess_call_t mess_calls [ ]

10.467.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.467.2 Typedef Documentation

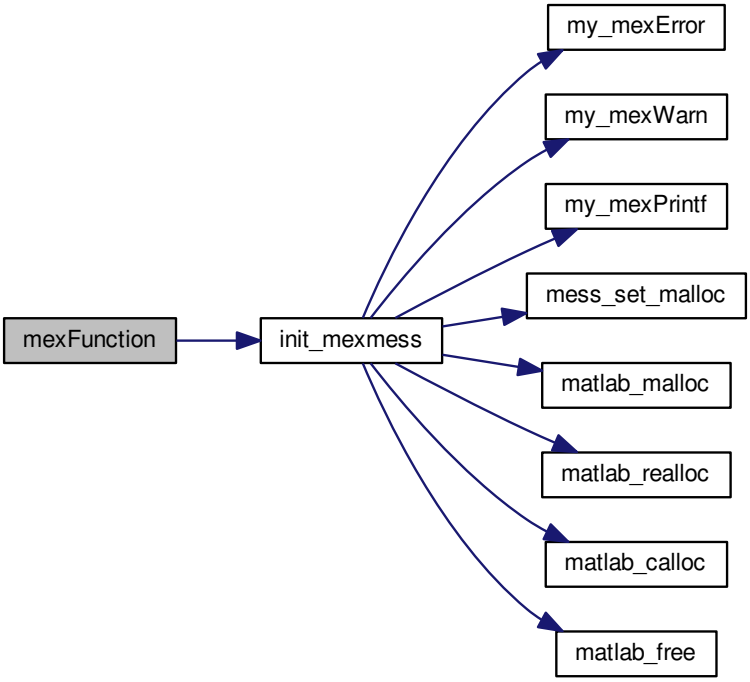
10.467.2.1 typedef void(∗ mess_call_func) (int nlhs, mxArray ∗plhs[ ], int nrhs, const mxArray ∗prhs[ ])

10.467.2.2 typedef struct _mess_call mess_call_t

10.467.3 Function Documentation

10.467.3.1 void mexFunction (int *nlhs*, mxArray ∗ *plhs[ ]*, int *nrhs*, const mxArray ∗ *prhs[ ]*)

Here is the call graph for this function:



10.467.4 Variable Documentation

10.467.4.1 mess_call_t mess_calls[ ] [static]

10.468 mess_classname_from_mexmess.c File Reference

Functions

- char ∗ mess_classname_from_mexmess (const mxArray ∗instance)

  Get a class name from MATLAB *object.*

10.468.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.469 MESS_direct.cc File Reference

Functions

- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_direct,"mess_direct","mess_direct")

10.469.1 Function Documentation

10.469.1.1 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_direct , "mess_direct" , "mess_direct")

10.470 MESS_direct.h File Reference

Data Structures

- class MESS_direct

10.471 MESS_direct_install.cc File Reference

10.472 MESS_enum.cc File Reference

Macros

- #define OCTMESS_GENERATOR(ENUMT, ENUMVAL)

Functions

- DEFUN_DLD (MESS_UNKNOWN, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_storage_t")
- DEFUN_DLD (MESS_CSR, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_storage_t")
- DEFUN_DLD (MESS_CSC, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_storage_t")
- DEFUN_DLD (MESS_DENSE, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_storage_t")
- DEFUN_DLD (MESS_COORD, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_storage_t")
- DEFUN_DLD (MESS_EQN_NONE, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_equation↩_t")
- DEFUN_DLD (MESS_EQN_LYAP, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_equation_t")
- DEFUN_DLD (MESS_EQN_GLYAP, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_equation↩_t")
- DEFUN_DLD (MESS_EQN_RICCATI, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_↩equation_t")
- DEFUN_DLD (MESS_EQN_GRICCATI, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_↩equation_t")
- DEFUN_DLD (MESS_MEMORY_LOW, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_↩memusage_t")
- DEFUN_DLD (MESS_MEMORY_MID, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_↩memusage_t")
- DEFUN_DLD (MESS_MEMORY_HIGH, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_↩memusage_t")
- DEFUN_DLD (MESS_OP_NONE, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_operation_t")
- DEFUN_DLD (MESS_OP_TRANSPOSE, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_↩operation_t")
- DEFUN_DLD (MESS_OP_HERMITIAN, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_↩operation_t")
- DEFUN_DLD (MESS_LRCFADI_PARA_MINMAX, args, nargout,"Representation of C-M.E.S.S. enum ""M↩ESS_parameter_t")
- DEFUN_DLD (MESS_LRCFADI_PARA_MINMAX_REAL, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_parameter_t")
- DEFUN_DLD (MESS_LRCFADI_PARA_WACHSPRESS, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_parameter_t")

- DEFUN_DLD (MESS_LRCFADI_PARA_ADAPTIVE_V, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_parameter_t")
- DEFUN_DLD (MESS_LRCFADI_PARA_ADAPTIVE_Z, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_parameter_t")
- DEFUN_DLD (MESS_DIRECT_DEFAULT_LU, args, nargout,"Representation of C-M.E.S.S. enum ""MES↩S_direct_lupackage_t")
- DEFUN_DLD (MESS_DIRECT_SPARSE_LU, args, nargout,"Representation of C-M.E.S.S. enum ""MESS↩_direct_lupackage_t")
- DEFUN_DLD (MESS_DIRECT_LAPACK_LU, args, nargout,"Representation of C-M.E.S.S. enum ""MESS↩_direct_lupackage_t")
- DEFUN_DLD (MESS_DIRECT_UMFPACK_LU, args, nargout,"Representation of C-M.E.S.S. enum ""ME↩SS_direct_lupackage_t")
- DEFUN_DLD (MESS_DIRECT_SUPERLU_LU, args, nargout,"Representation of C-M.E.S.S. enum ""MES↩S_direct_lupackage_t")
- DEFUN_DLD (MESS_DIRECT_CSPARSE_LU, args, nargout,"Representation of C-M.E.S.S. enum ""ME↩SS_direct_lupackage_t")
- DEFUN_DLD (MESS_DIRECT_BANDED_LU, args, nargout,"Representation of C-M.E.S.S. enum ""MES↩S_direct_lupackage_t")
- DEFUN_DLD (MESS_DIRECT_MKLPARDISO_LU, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_direct_lupackage_t")
- DEFUN_DLD (MESS_DIRECT_DEFAULT_CHOLESKY, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_direct_cholpackage_t")
- DEFUN_DLD (MESS_DIRECT_LAPACK_CHOLESKY, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_direct_cholpackage_t")
- DEFUN_DLD (MESS_DIRECT_CSPARSE_CHOLESKY, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_direct_cholpackage_t")
- DEFUN_DLD (MESS_DIRECT_CHOLMOD_CHOLESKY, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_direct_cholpackage_t")
- DEFUN_DLD (MESS_MULTIDIRECT_SPARSE_LU, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_multidirect_t")
- DEFUN_DLD (MESS_MULTIDIRECT_UMFPACK_LU, args, nargout,"Representation of C-M.E.S.S. enum ""MESS_multidirect_t")
- DEFUN_DLD (MESS_RESIDUAL_INDEFINITE, args, nargout,"Representation of C-M.E.S.S. enum ""ME↩SS_residual_t")
- DEFUN_DLD (MESS_RESIDUAL_SPECTRAL, args, nargout,"Representation of C-M.E.S.S. enum ""MES↩S_residual_t")

10.472.1 Macro Definition Documentation

10.472.1.1 #define OCTMESS_GENERATOR(*ENUMT, ENUMVAL*)

Value:

```
DEFUN_DLD(ENUMVAL, args, nargout, "Representation of C-M.E.S.S. enum "#ENUMT){  \
        if(nargout > 1 || args.length()){                                       \
            print_usage();                                                      \
            return octave_value_list();                                         \
        }                                                                       \
        return octave_value(new ENUMT(ENUMVAL));                                \
    }
```

10.472.2 Function Documentation

10.472.2.1 DEFUN_DLD (MESS_UNKNOWN , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_storage_t")

10.472.2.2 DEFUN_DLD (MESS_CSR , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_storage_t")

10.472.2.3 DEFUN_DLD (MESS_CSC , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_storage_t")

10.472.2.4 DEFUN_DLD (MESS_DENSE , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_storage_t")

10.472.2.5 DEFUN_DLD (MESS_COORD , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_storage_t")

10.472.2.6 DEFUN_DLD (MESS_EQN_NONE , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_equation_t")

10.472.2.7 DEFUN_DLD (MESS_EQN_LYAP , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_equation_t")

10.472.2.8 DEFUN_DLD (MESS_EQN_GLYAP , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_equation_t")

10.472.2.9 DEFUN_DLD (MESS_EQN_RICCATI , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_equation_t")

10.472.2.10 DEFUN_DLD (MESS_EQN_GRICCATI , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_equation_t")

10.472.2.11 DEFUN_DLD (MESS_MEMORY_LOW , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_memusage_t")

10.472.2.12 DEFUN_DLD (MESS_MEMORY_MID , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_memusage_t")

10.472.2.13 DEFUN_DLD (MESS_MEMORY_HIGH , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_memusage_t")

10.472.2.14 DEFUN_DLD (MESS_OP_NONE , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_operation_t")

10.472.2.15 DEFUN_DLD (MESS_OP_TRANSPOSE , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_operation_t")

10.472.2.16 DEFUN_DLD (MESS_OP_HERMITIAN , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_operation_t")

10.472.2.17 DEFUN_DLD (MESS_LRCFADI_PARA_MINMAX , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_parameter_t")

10.472.2.18 DEFUN_DLD (MESS_LRCFADI_PARA_MINMAX_REAL , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_parameter_t")

10.472.2.19 DEFUN_DLD (MESS_LRCFADI_PARA_WACHSPRESS , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_parameter_t")

10.472.2.20 DEFUN_DLD (MESS_LRCFADI_PARA_ADAPTIVE_V , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_parameter_t")

10.472.2.21 DEFUN_DLD (MESS_LRCFADI_PARA_ADAPTIVE_Z , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_parameter_t")

10.472.2.22 DEFUN_DLD (MESS_DIRECT_DEFAULT_LU , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_direct_lupackage_t")

10.472.2.23 DEFUN_DLD (MESS_DIRECT_SPARSE_LU , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_direct_lupackage_t")

10.472.2.24 DEFUN_DLD (MESS_DIRECT_LAPACK_LU , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_direct_lupackage_t")

10.472.2.25 DEFUN_DLD (MESS_DIRECT_UMFPACK_LU , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_direct_lupackage_t")

10.472.2.26 DEFUN_DLD (MESS_DIRECT_SUPERLU_LU , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_direct_lupackage_t")

10.472.2.27 DEFUN_DLD (MESS_DIRECT_CSPARSE_LU , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_direct_lupackage_t")

10.472.2.28 DEFUN_DLD (MESS_DIRECT_BANDED_LU , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_direct_lupackage_t")

10.472.2.29 DEFUN_DLD (MESS_DIRECT_MKLPARDISO_LU , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_direct_lupackage_t")

10.472.2.30 DEFUN_DLD (MESS_DIRECT_DEFAULT_CHOLESKY , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_direct_cholpackage_t")

10.472.2.31 DEFUN_DLD (MESS_DIRECT_LAPACK_CHOLESKY , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_direct_cholpackage_t")

10.472.2.32 DEFUN_DLD (MESS_DIRECT_CSPARSE_CHOLESKY , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_direct_cholpackage_t")

10.472.2.33 DEFUN_DLD (MESS_DIRECT_CHOLMOD_CHOLESKY , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_direct_cholpackage_t")

10.472.2.34 DEFUN_DLD (MESS_MULTIDIRECT_SPARSE_LU , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_multidirect_t")

10.472.2.35 DEFUN_DLD (MESS_MULTIDIRECT_UMFPACK_LU , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_multidirect_t")

10.472.2.36 DEFUN_DLD (MESS_RESIDUAL_INDEFINITE , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_residual_t")

10.472.2.37 DEFUN_DLD (MESS_RESIDUAL_SPECTRAL , args , nargout , "Representation of C-M.E.S.S. enum ""MESS_residual_t")

10.473 mess_enum_converter.c File Reference

Macros

- #define MESS_ENUM_CONVERTER(CMESS_ENUM_NAME, DEFAULT_ENUM)

Functions

- mess_direct_cholpackage_t mess_direct_cholpackage_t_from_mexmess (const mxArray *instance)
- mxArray * mess_direct_cholpackage_t_to_mexmess (mess_direct_cholpackage_t c_enum)
- mess_direct_lupackage_t mess_direct_lupackage_t_from_mexmess (const mxArray *instance)
- mxArray * mess_direct_lupackage_t_to_mexmess (mess_direct_lupackage_t c_enum)
- mess_equation_t mess_equation_t_from_mexmess (const mxArray *instance)
- mxArray * mess_equation_t_to_mexmess (mess_equation_t c_enum)
- mess_memusage_t mess_memusage_t_from_mexmess (const mxArray *instance)
- mxArray * mess_memusage_t_to_mexmess (mess_memusage_t c_enum)
- mess_multidirect_t mess_multidirect_t_from_mexmess (const mxArray *instance)
- mxArray * mess_multidirect_t_to_mexmess (mess_multidirect_t c_enum)
- mess_operation_t mess_operation_t_from_mexmess (const mxArray *instance)
- mxArray * mess_operation_t_to_mexmess (mess_operation_t c_enum)
- mess_parameter_t mess_parameter_t_from_mexmess (const mxArray *instance)
- mxArray * mess_parameter_t_to_mexmess (mess_parameter_t c_enum)
- mess_residual_t mess_residual_t_from_mexmess (const mxArray *instance)
- mxArray * mess_residual_t_to_mexmess (mess_residual_t c_enum)
- mess_norm_t mess_norm_t_from_mexmess (const mxArray *instance)
- mxArray * mess_norm_t_to_mexmess (mess_norm_t c_enum)

10.473.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.473.2 Macro Definition Documentation

10.473.2.1 #define MESS_ENUM_CONVERTER(*CMESS_ENUM_NAME*, *DEFAULT_ENUM*)

10.474 MESS_enums.cc File Reference

Macros

- #define OCTMESS_GENERATOR(SHORTNAME, OCTMESSENUM, CMESSENUM)

Functions

- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_storage_t,"mess_storage_t","mess_storage_t")
- octave_value op_eq (const MESS_storage_t &o1, const MESS_storage_t &o2)
- octave_value op_ne (const MESS_storage_t &o1, const MESS_storage_t &o2)
- DEFBINOP (mst_mst_eq, MESS_storage_t, MESS_storage_t)
- DEFBINOP (mst_mst_ne, MESS_storage_t, MESS_storage_t)
- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_datatype_t,"mess_datatype_t","mess_datatype_t")
- octave_value op_eq (const MESS_datatype_t &o1, const MESS_datatype_t &o2)
- octave_value op_ne (const MESS_datatype_t &o1, const MESS_datatype_t &o2)
- DEFBINOP (mdt_mdt_eq, MESS_datatype_t, MESS_datatype_t)
- DEFBINOP (mdt_mdt_ne, MESS_datatype_t, MESS_datatype_t)

- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_equation_t,"mess_equation_t","mess_equation_t")
- octave_value op_eq (const MESS_equation_t &o1, const MESS_equation_t &o2)
- octave_value op_ne (const MESS_equation_t &o1, const MESS_equation_t &o2)
- DEFBINOP (met_met_eq, MESS_equation_t, MESS_equation_t)
- DEFBINOP (met_met_ne, MESS_equation_t, MESS_equation_t)
- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_memusage_t,"mess_memusage_t","mess_memusage_t")
- octave_value op_eq (const MESS_memusage_t &o1, const MESS_memusage_t &o2)
- octave_value op_ne (const MESS_memusage_t &o1, const MESS_memusage_t &o2)
- DEFBINOP (mmt_mmt_eq, MESS_memusage_t, MESS_memusage_t)
- DEFBINOP (mmt_mmt_ne, MESS_memusage_t, MESS_memusage_t)
- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_operation_t,"mess_operation_t","mess_operation_t")
- octave_value op_eq (const MESS_operation_t &o1, const MESS_operation_t &o2)
- octave_value op_ne (const MESS_operation_t &o1, const MESS_operation_t &o2)
- DEFBINOP (mot_mot_eq, MESS_operation_t, MESS_operation_t)
- DEFBINOP (mot_mot_ne, MESS_operation_t, MESS_operation_t)
- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_parameter_t,"mess_parameter_t","mess_parameter_t")
- octave_value op_eq (const MESS_parameter_t &o1, const MESS_parameter_t &o2)
- octave_value op_ne (const MESS_parameter_t &o1, const MESS_parameter_t &o2)
- DEFBINOP (mpt_mpt_eq, MESS_parameter_t, MESS_parameter_t)
- DEFBINOP (mpt_mpt_ne, MESS_parameter_t, MESS_parameter_t)
- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_direct_lupackage_t,"mess_direct_lupackage_t","mess_direct_lupackage_t")
- octave_value op_eq (const MESS_direct_lupackage_t &o1, const MESS_direct_lupackage_t &o2)
- octave_value op_ne (const MESS_direct_lupackage_t &o1, const MESS_direct_lupackage_t &o2)
- DEFBINOP (mdlt_mdlt_eq, MESS_direct_lupackage_t, MESS_direct_lupackage_t)
- DEFBINOP (mdlt_mdlt_ne, MESS_direct_lupackage_t, MESS_direct_lupackage_t)
- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_direct_cholpackage_t,"mess_direct_cholpackage_t","mess_direct_cholpackage_t")
- octave_value op_eq (const MESS_direct_cholpackage_t &o1, const MESS_direct_cholpackage_t &o2)
- octave_value op_ne (const MESS_direct_cholpackage_t &o1, const MESS_direct_cholpackage_t &o2)
- DEFBINOP (mdct_mdct_eq, MESS_direct_cholpackage_t, MESS_direct_cholpackage_t)
- DEFBINOP (mdct_mdct_ne, MESS_direct_cholpackage_t, MESS_direct_cholpackage_t)
- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_multidirect_t,"mess_multidirect_t","mess_multidirect_t")
- octave_value op_eq (const MESS_multidirect_t &o1, const MESS_multidirect_t &o2)
- octave_value op_ne (const MESS_multidirect_t &o1, const MESS_multidirect_t &o2)
- DEFBINOP (mmdt_mmdt_eq, MESS_multidirect_t, MESS_multidirect_t)
- DEFBINOP (mmdt_mmdt_ne, MESS_multidirect_t, MESS_multidirect_t)
- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_residual_t,"mess_residual_t","mess_residual_t")
- octave_value op_eq (const MESS_residual_t &o1, const MESS_residual_t &o2)
- octave_value op_ne (const MESS_residual_t &o1, const MESS_residual_t &o2)
- DEFBINOP (mrt_mrt_eq, MESS_residual_t, MESS_residual_t)
- DEFBINOP (mrt_mrt_ne, MESS_residual_t, MESS_residual_t)

10.474.1 Macro Definition Documentation

10.474.1.1 #define OCTMESS_GENERATOR(*SHORTNAME*, *OCTMESSENUM*, *CMESSENUM*)

Value:

```
DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (OCTMESSENUM, #CMESSENUM, #CMESSENUM
      );                                                                                                  \

                                                                    \
    /*-----------------------------------------------------------------------------
                                                                    \
     * implement binary operations
                                                                    \
     *-----------------------------------------------------------------------------*/
                                                                    \
    octave_value op_eq(const OCTMESSENUM & o1, const OCTMESSENUM & o2){return octave_value(o1==o2);}
                                                                        \
    octave_value op_ne(const OCTMESSENUM & o1, const OCTMESSENUM & o2){return octave_value(o1!=o2);}
                                                                        \
    DEFBINOP(SHORTNAME##_##SHORTNAME##_eq, OCTMESSENUM, OCTMESSENUM){return
      op_eq(dynamic_cast<const OCTMESSENUM & > (a1), dynamic_cast<const OCTMESSENUM & > (a2));}
      \
    DEFBINOP(SHORTNAME##_##SHORTNAME##_ne, OCTMESSENUM, OCTMESSENUM){return
      op_ne(dynamic_cast<const OCTMESSENUM & > (a1), dynamic_cast<const OCTMESSENUM & > (a2));}   \

                                                                    \
    bool OCTMESSENUM::type_installed = false;
                                                                    \
    void OCTMESSENUM::install(void){
                                                                    \
        if(!OCTMESSENUM::type_installed){
                                                                    \
            OCTMESSENUM::register_type();
                                                                    \
            octave_stdout << "Install " << OCTMESSENUM::static_type_name()
                                                                    \
                          << " at type-id = " <<  OCTMESSENUM::static_type_id() << "\n";
                                                                    \
            INSTALL_BINOP(op_eq, OCTMESSENUM, OCTMESSENUM,  SHORTNAME## _ ##SHORTNAME## _eq);
                                                                        \
            INSTALL_BINOP(op_ne, OCTMESSENUM, OCTMESSENUM,  SHORTNAME## _ ##SHORTNAME## _ne);
                                                                        \

                                                                    \
        }
                                                                    \
        OCTMESSENUM::type_installed = true;
                                                                    \
    }
                                                                    \
```

10.474.2 Function Documentation

10.474.2.1 DEFBINOP (mst_mst_eq , MESS_storage_t , MESS_storage_t)

10.474.2.2 DEFBINOP (mst_mst_ne , MESS_storage_t , MESS_storage_t)

10.474.2.3 DEFBINOP (mdt_mdt_eq , MESS_datatype_t , MESS_datatype_t)

10.474.2.4 DEFBINOP (mdt_mdt_ne , MESS_datatype_t , MESS_datatype_t)

10.474.2.5 DEFBINOP (met_met_eq , MESS_equation_t , MESS_equation_t)

10.474.2.6 DEFBINOP (met_met_ne , MESS_equation_t , MESS_equation_t)

10.474.2.7 DEFBINOP (mmt_mmt_eq , MESS_memusage_t , MESS_memusage_t)

10.474.2.8 DEFBINOP (mmt_mmt_ne , MESS_memusage_t , MESS_memusage_t)

10.474.2.9 DEFBINOP (mot_mot_eq , MESS_operation_t , MESS_operation_t)

10.474.2.10 DEFBINOP (mot_mot_ne , MESS_operation_t , MESS_operation_t)

10.474.2.11 DEFBINOP (mpt_mpt_eq , MESS_parameter_t , MESS_parameter_t)

10.474.2.12 DEFBINOP (mpt_mpt_ne , MESS_parameter_t , MESS_parameter_t)

10.474.2.13 DEFBINOP (mdlt_mdlt_eq , MESS_direct_lupackage_t , MESS_direct_lupackage_t)

10.474.2.14 DEFBINOP (mdlt_mdlt_ne , MESS_direct_lupackage_t , MESS_direct_lupackage_t)

10.474.2.15 DEFBINOP (mdct_mdct_eq , MESS_direct_cholpackage_t , MESS_direct_cholpackage_t)

10.474.2.16 DEFBINOP (mdct_mdct_ne , MESS_direct_cholpackage_t , MESS_direct_cholpackage_t)

10.474.2.17 DEFBINOP (mmdt_mmdt_eq , MESS_multidirect_t , MESS_multidirect_t)

10.474.2.18 DEFBINOP (mmdt_mmdt_ne , MESS_multidirect_t , MESS_multidirect_t)

10.474.2.19 DEFBINOP (mrt_mrt_eq , MESS_residual_t , MESS_residual_t)

10.474.2.20 DEFBINOP (mrt_mrt_ne , MESS_residual_t , MESS_residual_t)

10.474.2.21 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_storage_t , "mess_storage_t" , "mess_storage_t")

10.474.2.22 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_datatype_t , "mess_datatype_t" , "mess_datatype_t")

10.474.2.23 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_equation_t , "mess_equation_t" , "mess_equation_t")

10.474.2.24 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_memusage_t , "mess_memusage_t" , "mess_memusage_t")

10.474.2.25 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_operation_t , "mess_operation_t" , "mess_operation_t")

10.474.2.26 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_parameter_t , "mess_parameter_t" , "mess_parameter_t")

10.474.2.27 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_direct_lupackage_t , "mess_direct_lupackage_t" , "mess_direct_lupackage_t")

10.474.2.28 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_direct_cholpackage_t , "mess_direct_cholpackage_t" , "mess_direct_cholpackage_t")

10.474.2.29 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_multidirect_t , "mess_multidirect_t" , "mess_multidirect_t")

10.474.2.30 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_residual_t , "mess_residual_t" , "mess_residual_t")

10.474.2.31 octave_value op_eq (const MESS_storage_t & *o1*, const MESS_storage_t & *o2*)

10.474.2.32 octave_value op_eq (const MESS_datatype_t & *o1*, const MESS_datatype_t & *o2*)

10.474.2.33 octave_value op_eq (const MESS_equation_t & *o1*, const MESS_equation_t & *o2*)

10.474.2.34 octave_value op_eq (const MESS_memusage_t & *o1*, const MESS_memusage_t & *o2*)

10.474.2.35 octave_value op_eq (const MESS_operation_t & *o1*, const MESS_operation_t & *o2*)

10.474.2.36 octave_value op_eq (const MESS_parameter_t & *o1*, const MESS_parameter_t & *o2*)

10.474.2.37 octave_value op_eq (const MESS_direct_lupackage_t & *o1*, const MESS_direct_lupackage_t & *o2*)

10.474.2.38 octave_value op_eq (const MESS_direct_cholpackage_t & *o1*, const MESS_direct_cholpackage_t & *o2*)

10.474.2.39 octave_value op_eq (const MESS_multidirect_t & *o1*, const MESS_multidirect_t & *o2*)

10.474.2.40 octave_value op_eq (const MESS_residual_t & *o1*, const MESS_residual_t & *o2*)

10.474.2.41 octave_value op_ne (const MESS_storage_t & *o1*, const MESS_storage_t & *o2*)

10.474.2.42 octave_value op_ne (const MESS_datatype_t & *o1*, const MESS_datatype_t & *o2*)

10.474.2.43 octave_value op_ne (const MESS_equation_t & *o1*, const MESS_equation_t & *o2*)

10.474.2.44 octave_value op_ne (const MESS_memusage_t & *o1*, const MESS_memusage_t & *o2*)

10.474.2.45 octave_value op_ne (const MESS_operation_t & *o1*, const MESS_operation_t & *o2*)

10.474.2.46 octave_value op_ne (const MESS_parameter_t & *o1*, const MESS_parameter_t & *o2*)

10.474.2.47 octave_value op_ne (const MESS_direct_lupackage_t & *o1*, const MESS_direct_lupackage_t & *o2*)

10.474.2.48 octave_value op_ne (const MESS_direct_cholpackage_t & *o1*, const MESS_direct_cholpackage_t & *o2*)

10.474.2.49 octave_value op_ne (const MESS_multidirect_t & *o1*, const MESS_multidirect_t & *o2*)

10.474.2.50 octave_value op_ne (const MESS_residual_t & *o1*, const MESS_residual_t & *o2*)

10.475 MESS_enums.h File Reference

Data Structures

- class MESS_storage_t
- class MESS_datatype_t
- class MESS_equation_t
- class MESS_memusage_t
- class MESS_operation_t
- class MESS_parameter_t
- class MESS_direct_lupackage_t
- class MESS_direct_cholpackage_t
- class MESS_multidirect_t
- class MESS_residual_t

Macros

- #define OCTMESS_GENERATOR(OCTMESSENUM, MESSENUM, MESSENUMINIT, PRINTFCT)

Functions

- octave_value op_eq (const MESS_storage_t &o1, const MESS_storage_t &o2)
- octave_value op_ne (const MESS_storage_t &o1, const MESS_storage_t &o2)
- octave_value op_eq (const MESS_datatype_t &o1, const MESS_datatype_t &o2)
- octave_value op_ne (const MESS_datatype_t &o1, const MESS_datatype_t &o2)
- octave_value op_eq (const MESS_equation_t &o1, const MESS_equation_t &o2)
- octave_value op_ne (const MESS_equation_t &o1, const MESS_equation_t &o2)
- octave_value op_eq (const MESS_memusage_t &o1, const MESS_memusage_t &o2)
- octave_value op_ne (const MESS_memusage_t &o1, const MESS_memusage_t &o2)
- octave_value op_eq (const MESS_operation_t &o1, const MESS_operation_t &o2)
- octave_value op_ne (const MESS_operation_t &o1, const MESS_operation_t &o2)
- octave_value op_eq (const MESS_parameter_t &o1, const MESS_parameter_t &o2)
- octave_value op_ne (const MESS_parameter_t &o1, const MESS_parameter_t &o2)
- octave_value op_eq (const MESS_direct_lupackage_t &o1, const MESS_direct_lupackage_t &o2)
- octave_value op_ne (const MESS_direct_lupackage_t &o1, const MESS_direct_lupackage_t &o2)
- octave_value op_eq (const MESS_direct_cholpackage_t &o1, const MESS_direct_cholpackage_t &o2)
- octave_value op_ne (const MESS_direct_cholpackage_t &o1, const MESS_direct_cholpackage_t &o2)
- octave_value op_eq (const MESS_multidirect_t &o1, const MESS_multidirect_t &o2)
- octave_value op_ne (const MESS_multidirect_t &o1, const MESS_multidirect_t &o2)
- octave_value op_eq (const MESS_residual_t &o1, const MESS_residual_t &o2)
- octave_value op_ne (const MESS_residual_t &o1, const MESS_residual_t &o2)

10.475.1 Macro Definition Documentation

10.475.1.1 #define OCTMESS_GENERATOR(*OCTMESSENUM, MESSENUM, MESSENUMINIT, PRINTFCT*)

10.475.2 Function Documentation

10.475.2.1 octave_value op_eq (const MESS_storage_t & *o1*, const MESS_storage_t & *o2*)

10.475.2.2 octave_value op_eq (const MESS_datatype_t & *o1*, const MESS_datatype_t & *o2*)

10.475.2.3 octave_value op_eq (const MESS_equation_t & *o1*, const MESS_equation_t & *o2*)

10.475.2.4 octave_value op_eq (const MESS_memusage_t & *o1*, const MESS_memusage_t & *o2*)

10.475.2.5 octave_value op_eq (const MESS_operation_t & *o1*, const MESS_operation_t & *o2*)

10.475.2.6 octave_value op_eq (const MESS_parameter_t & *o1*, const MESS_parameter_t & *o2*)

10.475.2.7 octave_value op_eq (const MESS_direct_lupackage_t & *o1*, const MESS_direct_lupackage_t & *o2*)

10.475.2.8 octave_value op_eq (const MESS_direct_cholpackage_t & *o1*, const MESS_direct_cholpackage_t & *o2*)

10.475.2.9 octave_value op_eq (const MESS_multidirect_t & *o1*, const MESS_multidirect_t & *o2*)

10.475.2.10 octave_value op_eq (const MESS_residual_t & *o1*, const MESS_residual_t & *o2*)

10.475.2.11 octave_value op_ne (const MESS_storage_t & *o1*, const MESS_storage_t & *o2*)

10.475.2.12 octave_value op_ne (const MESS_datatype_t & *o1*, const MESS_datatype_t & *o2*)

10.475.2.13 octave_value op_ne (const MESS_equation_t & *o1*, const MESS_equation_t & *o2*)

10.475.2.14 octave_value op_ne (const MESS_memusage_t & *o1*, const MESS_memusage_t & *o2*)

10.475.2.15 octave_value op_ne (const MESS_operation_t & *o1*, const MESS_operation_t & *o2*)

10.475.2.16 octave_value op_ne (const MESS_parameter_t & *o1*, const MESS_parameter_t & *o2*)

10.475.2.17 octave_value op_ne (const MESS_direct_lupackage_t & *o1*, const MESS_direct_lupackage_t & *o2*)

10.475.2.18 octave_value op_ne (const MESS_direct_cholpackage_t & *o1*, const MESS_direct_cholpackage_t & *o2*)

10.475.2.19 octave_value op_ne (const MESS_multidirect_t & *o1*, const MESS_multidirect_t & *o2*)

10.475.2.20 octave_value op_ne (const MESS_residual_t & *o1*, const MESS_residual_t & *o2*)

10.476 MESS_equation.cc File Reference

Functions

- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_equation,"mess_equation","mess_equation")

10.476.1 Function Documentation

10.476.1.1 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_equation , "mess_equation" , "mess_equation")

10.477 MESS_equation.h File Reference

Data Structures

- class MESS_equation

10.478 mess_equation_from_mexmess.c File Reference

Data Structures

- struct _equations_t

  Structure for all **`Py-M.E.S.S.`** *to* **`M.E.S.S.`** *converters. Structure for all* **`Py-M.E.S.S.`** *to* **`M.E.S.S.`** *converters.*

Typedefs

- typedef mess_equation(∗ eqn_converter) (const mxArray ∗, const mxArray ∗m_opt, mess_freelist)
- typedef struct _equations_t equations_t

  *Structure for all **Py-M.E.S.S.** to **M.E.S.S.** converters. Structure for all **Py-M.E.S.S.** to **M.E.S.S.** converters.*

Functions

- mess_equation mess_equation_from_mexmess (mxArray ∗m_eqn, mxArray ∗m_opt, mess_freelist mem, mess_equation_t eqn_type)

Variables

- static equations_t eqns [ ]

10.478.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.478.2 Typedef Documentation

10.478.2.1 typedef mess_equation(∗ eqn_converter) (const mxArray ∗, const mxArray ∗m_opt, mess_freelist)

10.478.2.2 typedef struct _equations_t equations_t

Attention

Internal use only.

10.478.3 Variable Documentation

10.478.3.1 equations_t eqns[ ] [static]

Initial value:

```
= {
    {"mess_equation_glyap",                 eqn_conv_lyap},
    {"mess_equation_griccati",              eqn_conv_ric},
    {"mess_equation_glyap_so1",             eqn_conv_lyap_so1},
    {"mess_equation_griccati_so1",          eqn_conv_ric_so1},
    {"mess_equation_glyap_so2",             eqn_conv_lyap_so2},
    {"mess_equation_griccati_so2",          eqn_conv_ric_so2},
    {"mess_equation_glyap_dae2",            eqn_conv_lyap_dae2},
    {"mess_equation_griccati_dae2",         eqn_conv_ric_dae2},
    {"mess_equation_glyap_dae1",            eqn_conv_lyap_dae1},
    {"mess_equation_griccati_dae1",         eqn_conv_ric_dae1},
    {NULL, NULL}
}
```

10.479 MESS_equation_install.cc File Reference

10.480 MESS_error.h File Reference

Macros

- #define OCTMESS_ERROR(RET, COND, FUN)

10.480.1 Macro Definition Documentation

10.480.1.1 #define OCTMESS_ERROR(*RET, COND, FUN*)

Value:

```
{                                                                                    \
    if((COND)){
                        \
        error("OCT-M.E.S.S.: " __FILE__ ":%d\n\t %s returned Error %d - %s",__LINE__,#FUN,(RET),
      mess_get_error((RET)));    \
    }
                        \
}
                        \
```

10.481 MESS_HAVE_PTRDIFF_T.c File Reference

Macros

- #define SIZE (sizeof(ptrdiff_t))

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char info_size [ ]

10.481.1 Macro Definition Documentation

10.481.1.1 #define SIZE (sizeof(ptrdiff_t))

10.481.2 Function Documentation

10.481.2.1 int main (int *argc,* char ∗ *argv[ ]*)

10.481.3 Variable Documentation

10.481.3.1 char info_size[ ]

Initial value:

```
=  {'I', 'N', 'F', 'O', ':', 's','i','z','e','[',
  ('0' + (( (sizeof(ptrdiff_t))  / 10000)%10)),
  ('0' + (( (sizeof(ptrdiff_t))  / 1000)%10)),
  ('0' + (( (sizeof(ptrdiff_t))  / 100)%10)),
  ('0' + (( (sizeof(ptrdiff_t))  / 10)%10)),
  ('0' +  ( (sizeof(ptrdiff_t))     % 10)),
  ']',


  '\0'}
```

10.482 MESS_HAVE_SIZE_T.c File Reference

Macros

- #define SIZE (sizeof(size_t))

Functions

- int main (int argc, char ∗argv[ ])

Variables

- char info_size [ ]

10.482.1 Macro Definition Documentation

10.482.1.1 #define SIZE (sizeof(size_t))

10.482.2 Function Documentation

10.482.2.1 int main (int *argc*, char ∗ *argv[ ]*)

10.482.3 Variable Documentation

10.482.3.1 char info_size[ ]

Initial value:

```
=  {'I', 'N', 'F', 'O', ':', 's','i','z','e','[',
  ('0' + (( (sizeof(size_t))  / 10000)%10)),
  ('0' + (( (sizeof(size_t))  / 1000)%10)),
  ('0' + (( (sizeof(size_t))  / 100)%10)),
  ('0' + (( (sizeof(size_t))  / 10)%10)),
  ('0' +  ( (sizeof(size_t))     % 10)),
  ']',


  '\0'}
```

10.483 mess_matrix.c File Reference

Functions

- mess_matrix mess_matrix_from_mexmess (const mxArray ∗Amatlab)

  Copy a matrix from MATLAB *to a mess_matrix.*
- mxArray ∗ mess_matrix_to_mexmess (mess_matrix A)

  Copy a matrix from ***M.E.S.S.*** *to* MATLAB.

10.483.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.484 MESS_matrix.cc File Reference

Macros

- #define OCTMESS_GENERATOR(MAPNAME)

Functions

- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_matrix,"mess_matrix","mess_matrix")

10.484.1 Macro Definition Documentation

10.484.1.1 #define OCTMESS_GENERATOR(*MAPNAME*)

Value:

```
case umap_##MAPNAME:
        \
          ret = mess_matrix_copy(this->ptr, result);
        OCTMESS_ERROR(ret,(ret!=0),mess_matrix_copy);                \
          ret = mess_matrix_map_##MAPNAME(result, 1); OCTMESS_ERROR(ret,(ret!=0),
        mess_matrix_map_##MAPNAME);       \
          return new MESS_matrix(result);
                        \
```

10.484.2 Function Documentation

10.484.2.1 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_matrix , "mess_matrix" , "mess_matrix")

10.485 MESS_matrix.cc File Reference

Macros

- #define OCTMESS_MESS_MATRIX_PRINT(PRINTFCT)

Functions

- DEFUN_DLD (mess_matrix, args, nargout,"blabla")
- DEFUN_DLD (mess_matrix_print, args, nargout,"Interface to ""mess_matrix_print")
- DEFUN_DLD (mess_matrix_printinfo, args, nargout,"Interface to ""mess_matrix_printinfo")
- DEFUN_DLD (mess_matrix_printshort, args, nargout,"Interface to ""mess_matrix_printshort")
- DEFUN_DLD (mess_matrix_printdata, args, nargout,"Interface to ""mess_matrix_printdata")

10.485.1 Macro Definition Documentation

10.485.1.1 #define OCTMESS_MESS_MATRIX_PRINT(*PRINTFCT*)

Value:

```
DEFUN_DLD(PRINTFCT, args, nargout, "Interface to " #PRINTFCT){
                                    \
    int ret = 0;
                        \
    if(nargout || args.length()!=1 || args(0).type_id() != MESS_matrix::static_type_id()){
                        \
        print_usage();
                        \
    }else{
                        \
        ret = PRINTFCT((mess_matrix)const_cast<MESS_matrix &> (dynamic_cast<const MESS_matrix &>
       (args(0).get_rep())));      \
        OCTMESS_ERROR(ret,(ret!=0),PRINTFCT);
                                            \
    }
                        \
    return octave_value_list();
                        \
}
```

10.485.2 Function Documentation

10.485.2.1 DEFUN_DLD (mess_matrix , args , nargout , "blabla")

10.485.2.2 DEFUN_DLD (mess_matrix_print , args , nargout , "Interface to ""mess_matrix_print")

10.485.2.3 DEFUN_DLD (mess_matrix_printinfo , args , nargout , "Interface to ""mess_matrix_printinfo")

10.485.2.4 DEFUN_DLD (mess_matrix_printshort , args , nargout , "Interface to ""mess_matrix_printshort")

10.485.2.5 DEFUN_DLD (mess_matrix_printdata , args , nargout , "Interface to ""mess_matrix_printdata")

10.486 MESS_matrix.h File Reference

Data Structures

- class MESS_matrix

Macros

- #define OCTMESS_GENERATOR(TYPEA, TYPEB)

Functions

- octave_value op_mul (const octave_scalar &s, const MESS_matrix &m)
- octave_value op_mul (const MESS_matrix &m, const octave_scalar &s)
- octave_value op_mul (const octave_complex_scalar &s, const MESS_matrix &m)
- octave_value op_mul (const MESS_matrix &m, const octave_complex_scalar &s)
- octave_value op_not (const MESS_matrix &m)
- octave_value op_uplus (const MESS_matrix &m)
- octave_value op_uminus (const MESS_matrix &m)
- octave_value op_transpose (const MESS_matrix &m)
- octave_value op_hermitian (const MESS_matrix &m)
- octave_value op_add (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_sub (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_xmul (const mess_operation_t opA, MESS_matrix &m1, const mess_operation_t opB, const MESS_matrix &m2)
- octave_value op_mul (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_trans_mul (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_herm_mul (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_mul_trans (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_mul_herm (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_xldiv (const MESS_matrix &m1, const mess_operation_t opA, const MESS_matrix &m2)
- octave_value op_ldiv (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_trans_ldiv (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_herm_ldiv (const MESS_matrix &m1, const MESS_matrix &m2)
- mess_matrix_st octmess_matrix_conv (const Matrix &m)
- octave_value op_add (const MESS_matrix &m1, const octave_matrix &m2)
- octave_value op_sub (const MESS_matrix &m1, const octave_matrix &m2)
- octave_value op_xmul (const mess_operation_t opA, MESS_matrix &m1, const mess_operation_t opB, const octave_matrix &m2)
- octave_value op_mul (const MESS_matrix &m1, const octave_matrix &m2)
- octave_value op_trans_mul (const MESS_matrix &m1, const octave_matrix &m2)
- octave_value op_herm_mul (const MESS_matrix &m1, const octave_matrix &m2)
- octave_value op_mul_trans (const MESS_matrix &m1, const octave_matrix &m2)
- octave_value op_mul_herm (const MESS_matrix &m1, const octave_matrix &m2)
- octave_value op_xldiv (const MESS_matrix &m1, const mess_operation_t opA, const octave_matrix &m2)
- octave_value op_ldiv (const MESS_matrix &m1, const octave_matrix &m2)
- octave_value op_trans_ldiv (const MESS_matrix &m1, const octave_matrix &m2)
- octave_value op_herm_ldiv (const MESS_matrix &m1, const octave_matrix &m2)
- octave_value op_add (const octave_matrix &m1, const MESS_matrix &m2)
- octave_value op_sub (const octave_matrix &m1, const MESS_matrix &m2)
- octave_value op_xmul (const mess_operation_t opA, octave_matrix &m1, const mess_operation_t opB, const MESS_matrix &m2)
- octave_value op_mul (const octave_matrix &m1, const MESS_matrix &m2)
- octave_value op_trans_mul (const octave_matrix &m1, const MESS_matrix &m2)
- octave_value op_herm_mul (const octave_matrix &m1, const MESS_matrix &m2)
- octave_value op_mul_trans (const octave_matrix &m1, const MESS_matrix &m2)
- octave_value op_mul_herm (const octave_matrix &m1, const MESS_matrix &m2)
- octave_value op_xldiv (const octave_matrix &m1, const mess_operation_t opA, const MESS_matrix &m2)
- octave_value op_ldiv (const octave_matrix &m1, const MESS_matrix &m2)
- octave_value op_trans_ldiv (const octave_matrix &m1, const MESS_matrix &m2)
- octave_value op_herm_ldiv (const octave_matrix &m1, const MESS_matrix &m2)
- mess_matrix_st octmess_matrix_conv (const ComplexMatrix &m)
- octave_value op_add (const MESS_matrix &m1, const octave_complex_matrix &m2)
- octave_value op_sub (const MESS_matrix &m1, const octave_complex_matrix &m2)

- octave_value op_xmul (const mess_operation_t opA, MESS_matrix &m1, const mess_operation_t opB, const octave_complex_matrix &m2)
- octave_value op_mul (const MESS_matrix &m1, const octave_complex_matrix &m2)
- octave_value op_trans_mul (const MESS_matrix &m1, const octave_complex_matrix &m2)
- octave_value op_herm_mul (const MESS_matrix &m1, const octave_complex_matrix &m2)
- octave_value op_mul_trans (const MESS_matrix &m1, const octave_complex_matrix &m2)
- octave_value op_mul_herm (const MESS_matrix &m1, const octave_complex_matrix &m2)
- octave_value op_xldiv (const MESS_matrix &m1, const mess_operation_t opA, const octave_complex_matrix &m2)
- octave_value op_ldiv (const MESS_matrix &m1, const octave_complex_matrix &m2)
- octave_value op_trans_ldiv (const MESS_matrix &m1, const octave_complex_matrix &m2)
- octave_value op_herm_ldiv (const MESS_matrix &m1, const octave_complex_matrix &m2)
- octave_value op_add (const octave_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_sub (const octave_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_xmul (const mess_operation_t opA, octave_complex_matrix &m1, const mess_operation_t opB, const MESS_matrix &m2)
- octave_value op_mul (const octave_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_trans_mul (const octave_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_herm_mul (const octave_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_mul_trans (const octave_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_mul_herm (const octave_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_xldiv (const octave_complex_matrix &m1, const mess_operation_t opA, const MESS_matrix &m2)
- octave_value op_ldiv (const octave_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_trans_ldiv (const octave_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_herm_ldiv (const octave_complex_matrix &m1, const MESS_matrix &m2)
- mess_matrix_st octmess_matrix_conv (const SparseMatrix &m)
- octave_value op_add (const MESS_matrix &m1, const octave_sparse_matrix &m2)
- octave_value op_sub (const MESS_matrix &m1, const octave_sparse_matrix &m2)
- octave_value op_xmul (const mess_operation_t opA, MESS_matrix &m1, const mess_operation_t opB, const octave_sparse_matrix &m2)
- octave_value op_mul (const MESS_matrix &m1, const octave_sparse_matrix &m2)
- octave_value op_trans_mul (const MESS_matrix &m1, const octave_sparse_matrix &m2)
- octave_value op_herm_mul (const MESS_matrix &m1, const octave_sparse_matrix &m2)
- octave_value op_mul_trans (const MESS_matrix &m1, const octave_sparse_matrix &m2)
- octave_value op_mul_herm (const MESS_matrix &m1, const octave_sparse_matrix &m2)
- octave_value op_xldiv (const MESS_matrix &m1, const mess_operation_t opA, const octave_sparse_matrix &m2)
- octave_value op_ldiv (const MESS_matrix &m1, const octave_sparse_matrix &m2)
- octave_value op_trans_ldiv (const MESS_matrix &m1, const octave_sparse_matrix &m2)
- octave_value op_herm_ldiv (const MESS_matrix &m1, const octave_sparse_matrix &m2)
- octave_value op_add (const octave_sparse_matrix &m1, const MESS_matrix &m2)
- octave_value op_sub (const octave_sparse_matrix &m1, const MESS_matrix &m2)
- octave_value op_xmul (const mess_operation_t opA, octave_sparse_matrix &m1, const mess_operation_t opB, const MESS_matrix &m2)
- octave_value op_mul (const octave_sparse_matrix &m1, const MESS_matrix &m2)
- octave_value op_trans_mul (const octave_sparse_matrix &m1, const MESS_matrix &m2)
- octave_value op_herm_mul (const octave_sparse_matrix &m1, const MESS_matrix &m2)
- octave_value op_mul_trans (const octave_sparse_matrix &m1, const MESS_matrix &m2)
- octave_value op_mul_herm (const octave_sparse_matrix &m1, const MESS_matrix &m2)
- octave_value op_xldiv (const octave_sparse_matrix &m1, const mess_operation_t opA, const MESS_matrix &m2)
- octave_value op_ldiv (const octave_sparse_matrix &m1, const MESS_matrix &m2)
- octave_value op_trans_ldiv (const octave_sparse_matrix &m1, const MESS_matrix &m2)
- octave_value op_herm_ldiv (const octave_sparse_matrix &m1, const MESS_matrix &m2)

- mess_matrix_st octmess_matrix_conv (const SparseComplexMatrix &m)
- octave_value op_add (const MESS_matrix &m1, const octave_sparse_complex_matrix &m2)
- octave_value op_sub (const MESS_matrix &m1, const octave_sparse_complex_matrix &m2)
- octave_value op_xmul (const mess_operation_t opA, MESS_matrix &m1, const mess_operation_t opB, const octave_sparse_complex_matrix &m2)
- octave_value op_mul (const MESS_matrix &m1, const octave_sparse_complex_matrix &m2)
- octave_value op_trans_mul (const MESS_matrix &m1, const octave_sparse_complex_matrix &m2)
- octave_value op_herm_mul (const MESS_matrix &m1, const octave_sparse_complex_matrix &m2)
- octave_value op_mul_trans (const MESS_matrix &m1, const octave_sparse_complex_matrix &m2)
- octave_value op_mul_herm (const MESS_matrix &m1, const octave_sparse_complex_matrix &m2)
- octave_value op_xldiv (const MESS_matrix &m1, const mess_operation_t opA, const octave_sparse_complex_matrix &m2)
- octave_value op_ldiv (const MESS_matrix &m1, const octave_sparse_complex_matrix &m2)
- octave_value op_trans_ldiv (const MESS_matrix &m1, const octave_sparse_complex_matrix &m2)
- octave_value op_herm_ldiv (const MESS_matrix &m1, const octave_sparse_complex_matrix &m2)
- octave_value op_add (const octave_sparse_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_sub (const octave_sparse_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_xmul (const mess_operation_t opA, octave_sparse_complex_matrix &m1, const mess_operation_t opB, const MESS_matrix &m2)
- octave_value op_mul (const octave_sparse_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_trans_mul (const octave_sparse_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_herm_mul (const octave_sparse_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_mul_trans (const octave_sparse_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_mul_herm (const octave_sparse_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_xldiv (const octave_sparse_complex_matrix &m1, const mess_operation_t opA, const MESS_matrix &m2)
- octave_value op_ldiv (const octave_sparse_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_trans_ldiv (const octave_sparse_complex_matrix &m1, const MESS_matrix &m2)
- octave_value op_herm_ldiv (const octave_sparse_complex_matrix &m1, const MESS_matrix &m2)

10.486.1 Macro Definition Documentation

10.486.1.1 #define OCTMESS_GENERATOR(*TYPEA, TYPEB*)

Value:

```
octave_value op_add(const TYPEA & m1, const TYPEB &m2);
                \
    octave_value op_sub(const TYPEA & m1, const TYPEB &m2);
                    \
    octave_value op_xmul(const mess_operation_t opA, TYPEA & m1, const
      mess_operation_t opB, const TYPEB &m2);       \
    octave_value op_mul(const TYPEA & m1, const TYPEB & m2);
                    \
    octave_value op_trans_mul(const TYPEA & m1, const TYPEB & m2);
                        \
    octave_value op_herm_mul(const TYPEA & m1, const TYPEB & m2);
                        \
    octave_value op_mul_trans(const TYPEA & m1, const TYPEB & m2);
                        \
    octave_value op_mul_herm(const TYPEA & m1, const TYPEB & m2);
                        \
    octave_value op_xldiv(const TYPEA & m1, const mess_operation_t opA, const TYPEB
       & m2);                           \
    octave_value op_ldiv(const TYPEA & m1, const TYPEB & m2);
                    \
    octave_value op_trans_ldiv(const TYPEA & m1, const TYPEB &m2);
                        \
    octave_value op_herm_ldiv(const TYPEA & m1, const TYPEB &m2);
                        \
```

10.486.2 Function Documentation

10.486.2.1 mess_matrix_st octmess_matrix_conv (const Matrix & *m*) [inline]

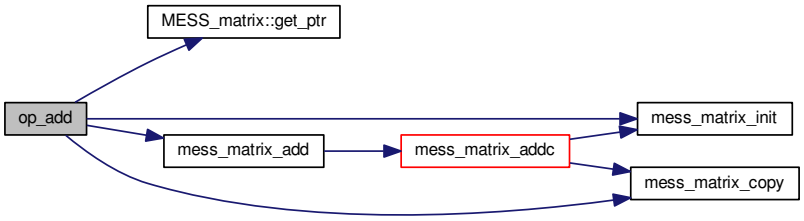
10.486.2.2 mess_matrix_st octmess_matrix_conv (const ComplexMatrix & *m*) [inline]

10.486.2.3 mess_matrix_st octmess_matrix_conv (const SparseMatrix & *m*) [inline]

10.486.2.4 mess_matrix_st octmess_matrix_conv (const SparseComplexMatrix & *m*) [inline]

10.486.2.5 octave_value op_add (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:



10.486.2.6 octave_value op_add (const MESS_matrix & *m1*, const octave_matrix & *m2*)

10.486.2.7 octave_value op_add (const octave_matrix & *m1*, const MESS_matrix & *m2*)

10.486.2.8 octave_value op_add (const MESS_matrix & *m1*, const octave_complex_matrix & *m2*)

10.486.2.9 octave_value op_add (const octave_complex_matrix & *m1*, const MESS_matrix & *m2*)

10.486.2.10 octave_value op_add (const MESS_matrix & *m1*, const octave_sparse_matrix & *m2*)

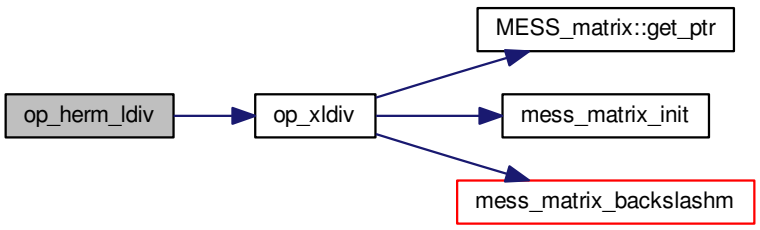
10.486.2.11 octave_value op_add (const octave_sparse_matrix & *m1*, const MESS_matrix & *m2*)

10.486.2.12 octave_value op_add (const MESS_matrix & *m1*, const octave_sparse_complex_matrix & *m2*)

10.486.2.13 octave_value op_add (const octave_sparse_complex_matrix & *m1*, const MESS_matrix & *m2*)

10.486.2.14 octave_value op_herm_ldiv (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:



10.486.2.15 octave_value op_herm_ldiv (const **MESS_matrix** & *m1*, const octave_matrix & *m2*)

10.486.2.16 octave_value op_herm_ldiv (const octave_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.17 octave_value op_herm_ldiv (const **MESS_matrix** & *m1*, const octave_complex_matrix & *m2*)

10.486.2.18 octave_value op_herm_ldiv (const octave_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.19 octave_value op_herm_ldiv (const **MESS_matrix** & *m1*, const octave_sparse_matrix & *m2*)

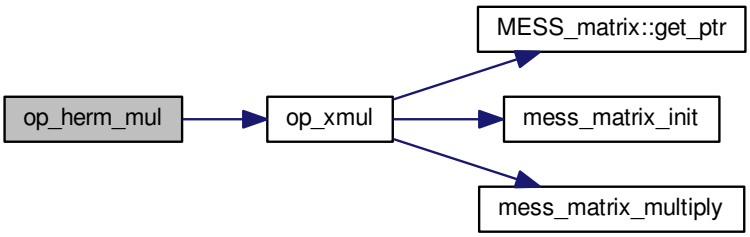
10.486.2.20 octave_value op_herm_ldiv (const octave_sparse_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.21 octave_value op_herm_ldiv (const **MESS_matrix** & *m1*, const octave_sparse_complex_matrix & *m2*)

10.486.2.22 octave_value op_herm_ldiv (const octave_sparse_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.23 octave_value op_herm_mul (const **MESS_matrix** & *m1*, const **MESS_matrix** & *m2*)

Here is the call graph for this function:



10.486.2.24 octave_value op_herm_mul (const **MESS_matrix** & *m1*, const octave_matrix & *m2*)

10.486.2.25 octave_value op_herm_mul (const octave_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.26 octave_value op_herm_mul (const **MESS_matrix** & *m1*, const octave_complex_matrix & *m2*)

10.486.2.27 octave_value op_herm_mul (const octave_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.28 octave_value op_herm_mul (const **MESS_matrix** & *m1*, const octave_sparse_matrix & *m2*)

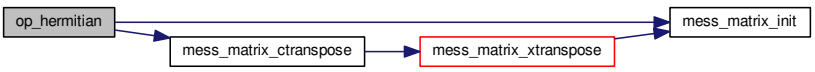
10.486.2.29 octave_value op_herm_mul (const octave_sparse_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.30 octave_value op_herm_mul (const **MESS_matrix** & *m1*, const octave_sparse_complex_matrix & *m2*)

10.486.2.31 octave_value op_herm_mul (const octave_sparse_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

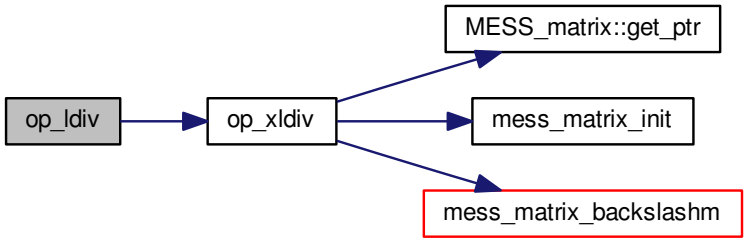
10.486.2.32 octave_value op_hermitian (const **MESS_matrix** & *m*)

Here is the call graph for this function:



10.486.2.33 octave_value op_ldiv (const **MESS_matrix** & *m1*, const **MESS_matrix** & *m2*)

Here is the call graph for this function:



10.486.2.34 octave_value op_ldiv (const **MESS_matrix** & *m1*, const octave_matrix & *m2*)

10.486.2.35 octave_value op_ldiv (const octave_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.36 octave_value op_ldiv (const **MESS_matrix** & *m1*, const octave_complex_matrix & *m2*)

10.486.2.37 octave_value op_ldiv (const octave_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.38 octave_value op_ldiv (const **MESS_matrix** & *m1*, const octave_sparse_matrix & *m2*)

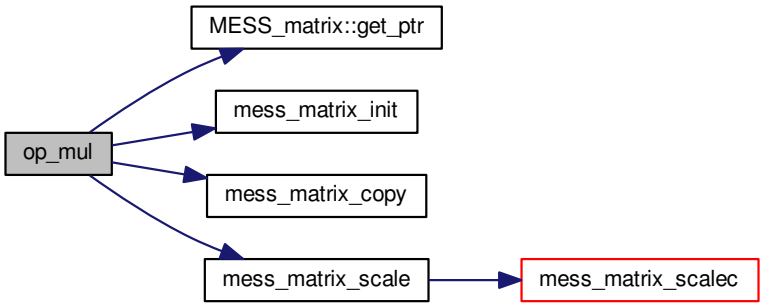
10.486.2.39 octave_value op_ldiv (const octave_sparse_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.40 octave_value op_ldiv (const **MESS_matrix** & *m1*, const octave_sparse_complex_matrix & *m2*)

10.486.2.41 octave_value op_ldiv (const octave_sparse_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

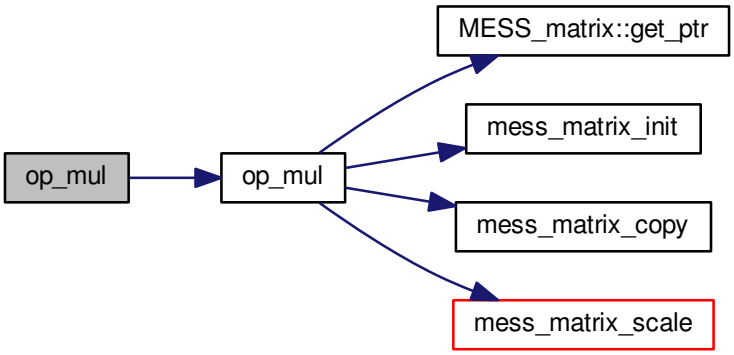
10.486.2.42 octave_value op_mul (const octave_scalar & *s*, const **MESS_matrix** & *m*)

Here is the call graph for this function:



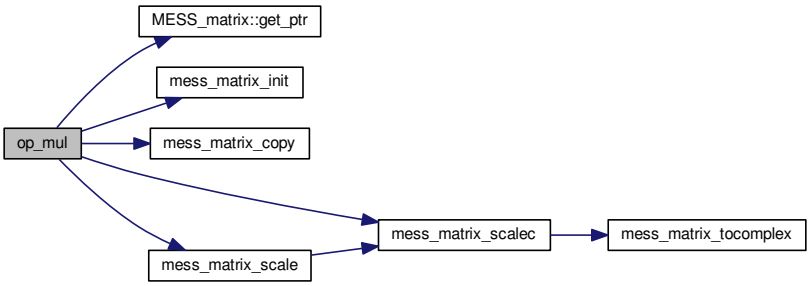
10.486.2.43 octave_value op_mul (const MESS_matrix & *m*, const octave_scalar & *s*)

Here is the call graph for this function:



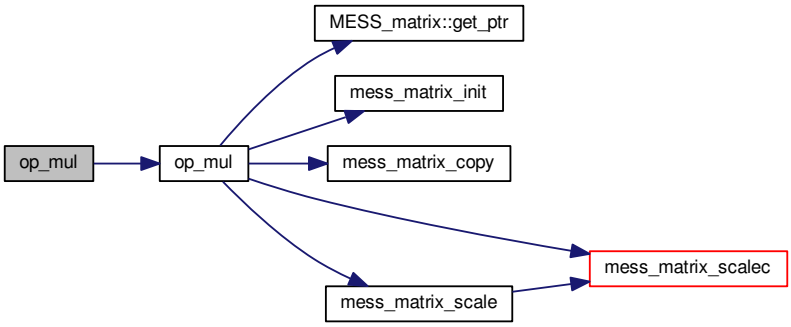
10.486.2.44 octave_value op_mul (const octave_complex_scalar & *s*, const MESS_matrix & *m*)

Here is the call graph for this function:



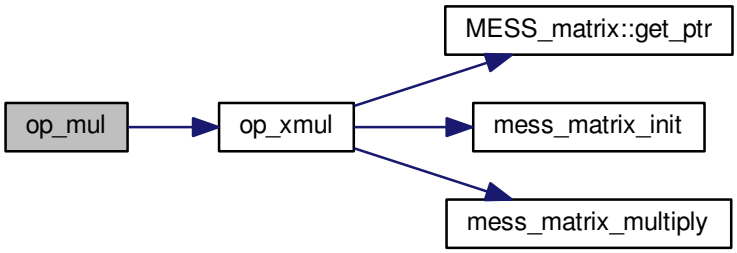
10.486.2.45 octave_value op_mul (const MESS_matrix & *m*, const octave_complex_scalar & *s*)

Here is the call graph for this function:



10.486.2.46 octave_value op_mul (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:



10.486.2.47 octave_value op_mul (const MESS_matrix & *m1*, const octave_matrix & *m2*)

10.486.2.48 octave_value op_mul (const octave_matrix & *m1*, const MESS_matrix & *m2*)

10.486.2.49 octave_value op_mul (const MESS_matrix & *m1*, const octave_complex_matrix & *m2*)

10.486.2.50 octave_value op_mul (const octave_complex_matrix & *m1*, const MESS_matrix & *m2*)

10.486.2.51 octave_value op_mul (const MESS_matrix & *m1*, const octave_sparse_matrix & *m2*)

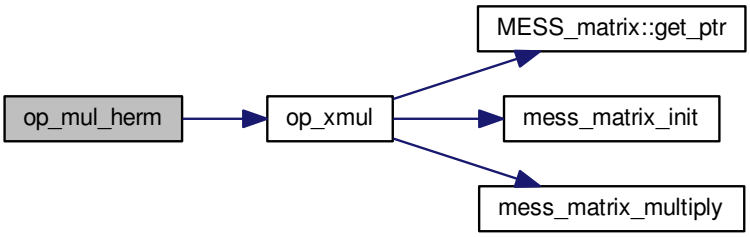
10.486.2.52 octave_value op_mul (const octave_sparse_matrix & *m1*, const MESS_matrix & *m2*)

10.486.2.53 octave_value op_mul (const **MESS_matrix** & *m1*, const octave_sparse_complex_matrix & *m2*)

10.486.2.54 octave_value op_mul (const octave_sparse_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.55 octave_value op_mul_herm (const **MESS_matrix** & *m1*, const **MESS_matrix** & *m2*)

Here is the call graph for this function:



10.486.2.56 octave_value op_mul_herm (const **MESS_matrix** & *m1*, const octave_matrix & *m2*)

10.486.2.57 octave_value op_mul_herm (const octave_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.58 octave_value op_mul_herm (const **MESS_matrix** & *m1*, const octave_complex_matrix & *m2*)

10.486.2.59 octave_value op_mul_herm (const octave_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.60 octave_value op_mul_herm (const **MESS_matrix** & *m1*, const octave_sparse_matrix & *m2*)

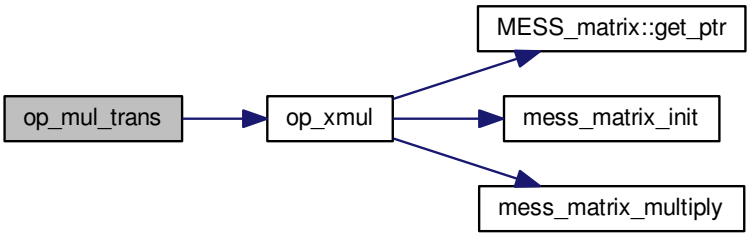
10.486.2.61 octave_value op_mul_herm (const octave_sparse_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.62 octave_value op_mul_herm (const **MESS_matrix** & *m1*, const octave_sparse_complex_matrix & *m2*)

10.486.2.63 octave_value op_mul_herm (const octave_sparse_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.64 octave_value op_mul_trans (const **MESS_matrix** & *m1*, const **MESS_matrix** & *m2*)

Here is the call graph for this function:



10.486.2.65 octave_value op_mul_trans (const **MESS_matrix** & *m1*, const octave_matrix & *m2*)

10.486.2.66 octave_value op_mul_trans (const octave_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.67 octave_value op_mul_trans (const **MESS_matrix** & *m1*, const octave_complex_matrix & *m2*)

10.486.2.68 octave_value op_mul_trans (const octave_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.69 octave_value op_mul_trans (const **MESS_matrix** & *m1*, const octave_sparse_matrix & *m2*)

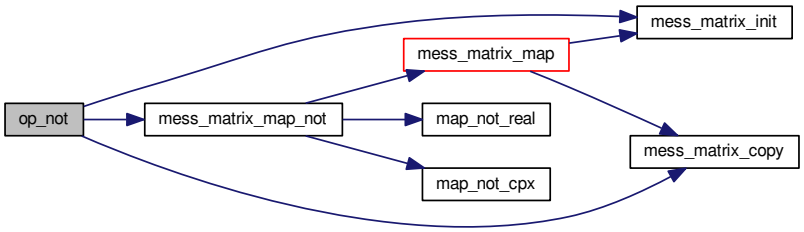
10.486.2.70 octave_value op_mul_trans (const octave_sparse_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.71 octave_value op_mul_trans (const **MESS_matrix** & *m1*, const octave_sparse_complex_matrix & *m2*)

10.486.2.72 octave_value op_mul_trans (const octave_sparse_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

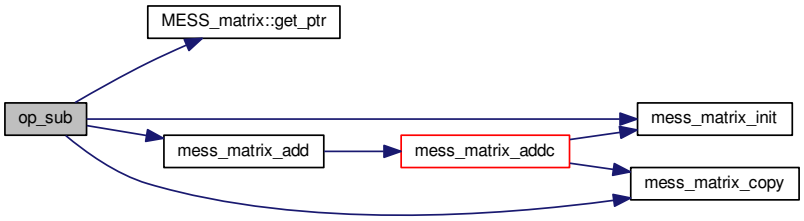
10.486.2.73 octave_value op_not (const **MESS_matrix** & *m*)

Here is the call graph for this function:



10.486.2.74 octave_value op_sub (const **MESS_matrix** & *m1*, const **MESS_matrix** & *m2*)

Here is the call graph for this function:



10.486.2.75 octave_value op_sub (const **MESS_matrix** & *m1*, const octave_matrix & *m2*)

10.486.2.76 octave_value op_sub (const octave_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.77 octave_value op_sub (const **MESS_matrix** & *m1*, const octave_complex_matrix & *m2*)

10.486.2.78 octave_value op_sub (const octave_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.79 octave_value op_sub (const **MESS_matrix** & *m1*, const octave_sparse_matrix & *m2*)

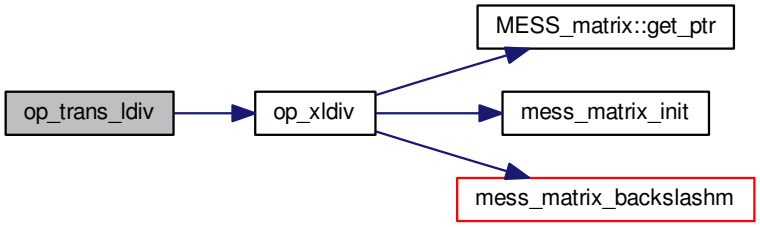
10.486.2.80 octave_value op_sub (const octave_sparse_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.81 octave_value op_sub (const **MESS_matrix** & *m1*, const octave_sparse_complex_matrix & *m2*)

10.486.2.82 octave_value op_sub (const octave_sparse_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.83 octave_value op_trans_ldiv (const **MESS_matrix** & *m1*, const **MESS_matrix** & *m2*)

Here is the call graph for this function:



10.486.2.84 octave_value op_trans_ldiv (const **MESS_matrix** & *m1*, const octave_matrix & *m2*)

10.486.2.85 octave_value op_trans_ldiv (const octave_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.86 octave_value op_trans_ldiv (const **MESS_matrix** & *m1*, const octave_complex_matrix & *m2*)

10.486.2.87 octave_value op_trans_ldiv (const octave_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.88 octave_value op_trans_ldiv (const **MESS_matrix** & *m1*, const octave_sparse_matrix & *m2*)

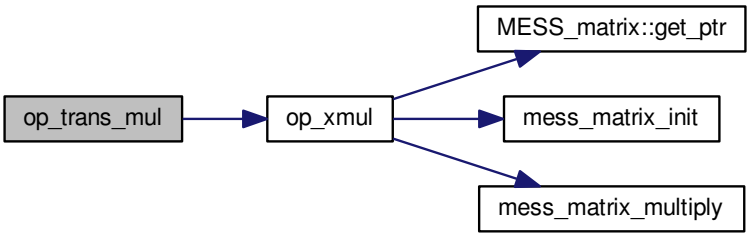
10.486.2.89 octave_value op_trans_ldiv (const octave_sparse_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.90 octave_value op_trans_ldiv (const **MESS_matrix** & *m1*, const octave_sparse_complex_matrix & *m2*)

10.486.2.91 octave_value op_trans_ldiv (const octave_sparse_complex_matrix & *m1*, const **MESS_matrix** & *m2*)

10.486.2.92 octave_value op_trans_mul (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:



10.486.2.93 octave_value op_trans_mul (const MESS_matrix & *m1*, const octave_matrix & *m2*)

10.486.2.94 octave_value op_trans_mul (const octave_matrix & *m1*, const MESS_matrix & *m2*)

10.486.2.95 octave_value op_trans_mul (const MESS_matrix & *m1*, const octave_complex_matrix & *m2*)

10.486.2.96 octave_value op_trans_mul (const octave_complex_matrix & *m1*, const MESS_matrix & *m2*)

10.486.2.97 octave_value op_trans_mul (const MESS_matrix & *m1*, const octave_sparse_matrix & *m2*)

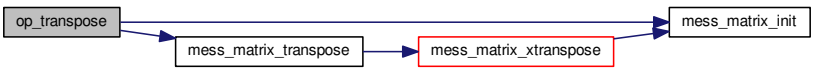
10.486.2.98 octave_value op_trans_mul (const octave_sparse_matrix & *m1*, const MESS_matrix & *m2*)

10.486.2.99 octave_value op_trans_mul (const MESS_matrix & *m1*, const octave_sparse_complex_matrix & *m2*)

10.486.2.100 octave_value op_trans_mul (const octave_sparse_complex_matrix & *m1*, const MESS_matrix & *m2*)

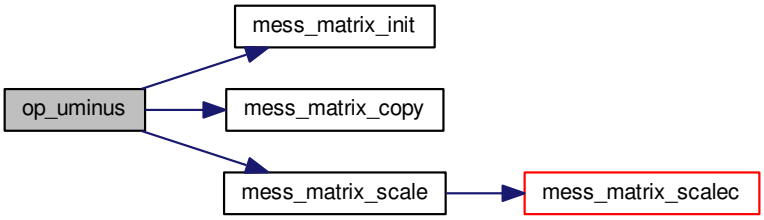
10.486.2.101 octave_value op_transpose (const MESS_matrix & *m*)

Here is the call graph for this function:



10.486.2.102 octave_value op_uminus (const MESS_matrix & *m*)

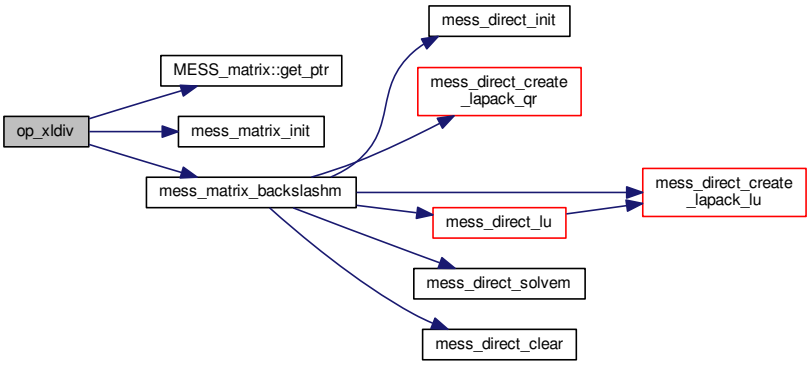
Here is the call graph for this function:



10.486.2.103 octave_value op_uplus (const MESS_matrix & *m*)

10.486.2.104 octave_value op_xldiv (const MESS_matrix & *m1*, const mess_operation_t *opA*, const MESS_matrix & *m2*)

Here is the call graph for this function:



10.486.2.105 octave_value op_xldiv (const MESS_matrix & *m1*, const mess_operation_t *opA*, const octave_matrix & *m2*)

10.486.2.106 octave_value op_xldiv (const octave_matrix & *m1*, const mess_operation_t *opA*, const MESS_matrix & *m2*)

10.486.2.107 octave_value op_xldiv (const MESS_matrix & *m1*, const mess_operation_t *opA*, const octave_complex_matrix & *m2*)

10.486.2.108 octave_value op_xldiv (const octave_complex_matrix & *m1*, const mess_operation_t *opA*, const MESS_matrix & *m2*)

10.486.2.109 octave_value op_xldiv (const **MESS_matrix** & *m1*, const **mess_operation_t** *opA*, const octave_sparse_matrix & *m2*)

10.486.2.110 octave_value op_xldiv (const octave_sparse_matrix & *m1*, const **mess_operation_t** *opA*, const **MESS_matrix** & *m2*)

10.486.2.111 octave_value op_xldiv (const **MESS_matrix** & *m1*, const **mess_operation_t** *opA*, const octave_sparse_complex_matrix & *m2*)

10.486.2.112 octave_value op_xldiv (const octave_sparse_complex_matrix & *m1*, const **mess_operation_t** *opA*, const **MESS_matrix** & *m2*)

10.486.2.113 octave_value op_xmul (const **mess_operation_t** *opA*, **MESS_matrix** & *m1*, const **mess_operation_t** *opB*, const **MESS_matrix** & *m2*)

10.486.2.114 octave_value op_xmul (const **mess_operation_t** *opA*, **MESS_matrix** & *m1*, const **mess_operation_t** *opB*, const octave_matrix & *m2*)

10.486.2.115 octave_value op_xmul (const **mess_operation_t** *opA*, octave_matrix & *m1*, const **mess_operation_t** *opB*, const **MESS_matrix** & *m2*)

10.486.2.116 octave_value op_xmul (const **mess_operation_t** *opA*, **MESS_matrix** & *m1*, const **mess_operation_t** *opB*, const octave_complex_matrix & *m2*)

10.486.2.117 octave_value op_xmul (const **mess_operation_t** *opA*, octave_complex_matrix & *m1*, const **mess_operation_t** *opB*, const **MESS_matrix** & *m2*)

10.486.2.118 octave_value op_xmul (const **mess_operation_t** *opA*, **MESS_matrix** & *m1*, const **mess_operation_t** *opB*, const octave_sparse_matrix & *m2*)

10.486.2.119 octave_value op_xmul (const **mess_operation_t** *opA*, octave_sparse_matrix & *m1*, const **mess_operation_t** *opB*, const **MESS_matrix** & *m2*)

10.486.2.120 octave_value op_xmul (const **mess_operation_t** *opA*, **MESS_matrix** & *m1*, const **mess_operation_t** *opB*, const octave_sparse_complex_matrix & *m2*)

10.486.2.121 octave_value op_xmul (const **mess_operation_t** *opA*, octave_sparse_complex_matrix & *m1*, const **mess_operation_t** *opB*, const **MESS_matrix** & *m2*)

10.487 MESS_matrix_install.cc File Reference

Macros

- #define OCTMESS_GENERATOR(SHORTTYPEA, TYPEA, SHORTTYPEB, TYPEB)
- #define OCTMESS_GENERATOR(SHORTTYPEA, TYPEA, SHORTTYPEB, TYPEB)

Functions

- DEFUNOP (mm_not, MESS_matrix)
- DEFUNOP (mm_uplus, MESS_matrix)
- DEFUNOP (mm_uminus, MESS_matrix)
- DEFUNOP (mm_transpose, MESS_matrix)
- DEFUNOP (mm_hermitian, MESS_matrix)
- DEFBINOP (mm_mm_add, MESS_matrix, MESS_matrix)
- DEFBINOP (mm_mm_sub, MESS_matrix, MESS_matrix)
- DEFBINOP (mm_mm_mul, MESS_matrix, MESS_matrix)
- DEFBINOP (mm_mm_mul_trans, MESS_matrix, MESS_matrix)
- DEFBINOP (mm_mm_mul_herm, MESS_matrix, MESS_matrix)
- DEFBINOP (mm_mm_trans_mul, MESS_matrix, MESS_matrix)
- DEFBINOP (mm_mm_herm_mul, MESS_matrix, MESS_matrix)
- DEFBINOP (mm_mm_ldiv, MESS_matrix, MESS_matrix)
- DEFBINOP (mm_mm_trans_ldiv, MESS_matrix, MESS_matrix)
- DEFBINOP (mm_mm_herm_ldiv, MESS_matrix, MESS_matrix)
- DEFBINOP (mm_om_add, MESS_matrix, octave_matrix)
- DEFBINOP (mm_om_sub, MESS_matrix, octave_matrix)
- DEFBINOP (mm_om_mul, MESS_matrix, octave_matrix)
- DEFBINOP (mm_om_mul_trans, MESS_matrix, octave_matrix)
- DEFBINOP (mm_om_mul_herm, MESS_matrix, octave_matrix)
- DEFBINOP (mm_om_trans_mul, MESS_matrix, octave_matrix)
- DEFBINOP (mm_om_herm_mul, MESS_matrix, octave_matrix)
- DEFBINOP (mm_om_ldiv, MESS_matrix, octave_matrix)
- DEFBINOP (mm_om_trans_ldiv, MESS_matrix, octave_matrix)
- DEFBINOP (mm_om_herm_ldiv, MESS_matrix, octave_matrix)
- DEFBINOP (om_mm_add, octave_matrix, MESS_matrix)
- DEFBINOP (om_mm_sub, octave_matrix, MESS_matrix)
- DEFBINOP (om_mm_mul, octave_matrix, MESS_matrix)
- DEFBINOP (om_mm_mul_trans, octave_matrix, MESS_matrix)
- DEFBINOP (om_mm_mul_herm, octave_matrix, MESS_matrix)
- DEFBINOP (om_mm_trans_mul, octave_matrix, MESS_matrix)
- DEFBINOP (om_mm_herm_mul, octave_matrix, MESS_matrix)
- DEFBINOP (om_mm_ldiv, octave_matrix, MESS_matrix)
- DEFBINOP (om_mm_trans_ldiv, octave_matrix, MESS_matrix)
- DEFBINOP (om_mm_herm_ldiv, octave_matrix, MESS_matrix)
- DEFBINOP (mm_ocm_add, MESS_matrix, octave_complex_matrix)
- DEFBINOP (mm_ocm_sub, MESS_matrix, octave_complex_matrix)
- DEFBINOP (mm_ocm_mul, MESS_matrix, octave_complex_matrix)
- DEFBINOP (mm_ocm_mul_trans, MESS_matrix, octave_complex_matrix)
- DEFBINOP (mm_ocm_mul_herm, MESS_matrix, octave_complex_matrix)
- DEFBINOP (mm_ocm_trans_mul, MESS_matrix, octave_complex_matrix)
- DEFBINOP (mm_ocm_herm_mul, MESS_matrix, octave_complex_matrix)
- DEFBINOP (mm_ocm_ldiv, MESS_matrix, octave_complex_matrix)
- DEFBINOP (mm_ocm_trans_ldiv, MESS_matrix, octave_complex_matrix)
- DEFBINOP (mm_ocm_herm_ldiv, MESS_matrix, octave_complex_matrix)
- DEFBINOP (ocm_mm_add, octave_complex_matrix, MESS_matrix)
- DEFBINOP (ocm_mm_sub, octave_complex_matrix, MESS_matrix)
- DEFBINOP (ocm_mm_mul, octave_complex_matrix, MESS_matrix)
- DEFBINOP (ocm_mm_mul_trans, octave_complex_matrix, MESS_matrix)
- DEFBINOP (ocm_mm_mul_herm, octave_complex_matrix, MESS_matrix)
- DEFBINOP (ocm_mm_trans_mul, octave_complex_matrix, MESS_matrix)
- DEFBINOP (ocm_mm_herm_mul, octave_complex_matrix, MESS_matrix)
- DEFBINOP (ocm_mm_ldiv, octave_complex_matrix, MESS_matrix)

- DEFBINOP (ocm_mm_trans_ldiv, octave_complex_matrix, MESS_matrix)
- DEFBINOP (ocm_mm_herm_ldiv, octave_complex_matrix, MESS_matrix)
- DEFBINOP (mm_osm_add, MESS_matrix, octave_sparse_matrix)
- DEFBINOP (mm_osm_sub, MESS_matrix, octave_sparse_matrix)
- DEFBINOP (mm_osm_mul, MESS_matrix, octave_sparse_matrix)
- DEFBINOP (mm_osm_mul_trans, MESS_matrix, octave_sparse_matrix)
- DEFBINOP (mm_osm_mul_herm, MESS_matrix, octave_sparse_matrix)
- DEFBINOP (mm_osm_trans_mul, MESS_matrix, octave_sparse_matrix)
- DEFBINOP (mm_osm_herm_mul, MESS_matrix, octave_sparse_matrix)
- DEFBINOP (mm_osm_ldiv, MESS_matrix, octave_sparse_matrix)
- DEFBINOP (mm_osm_trans_ldiv, MESS_matrix, octave_sparse_matrix)
- DEFBINOP (mm_osm_herm_ldiv, MESS_matrix, octave_sparse_matrix)
- DEFBINOP (osm_mm_add, octave_sparse_matrix, MESS_matrix)
- DEFBINOP (osm_mm_sub, octave_sparse_matrix, MESS_matrix)
- DEFBINOP (osm_mm_mul, octave_sparse_matrix, MESS_matrix)
- DEFBINOP (osm_mm_mul_trans, octave_sparse_matrix, MESS_matrix)
- DEFBINOP (osm_mm_mul_herm, octave_sparse_matrix, MESS_matrix)
- DEFBINOP (osm_mm_trans_mul, octave_sparse_matrix, MESS_matrix)
- DEFBINOP (osm_mm_herm_mul, octave_sparse_matrix, MESS_matrix)
- DEFBINOP (osm_mm_ldiv, octave_sparse_matrix, MESS_matrix)
- DEFBINOP (osm_mm_trans_ldiv, octave_sparse_matrix, MESS_matrix)
- DEFBINOP (osm_mm_herm_ldiv, octave_sparse_matrix, MESS_matrix)
- DEFBINOP (mm_oscm_add, MESS_matrix, octave_sparse_complex_matrix)
- DEFBINOP (mm_oscm_sub, MESS_matrix, octave_sparse_complex_matrix)
- DEFBINOP (mm_oscm_mul, MESS_matrix, octave_sparse_complex_matrix)
- DEFBINOP (mm_oscm_mul_trans, MESS_matrix, octave_sparse_complex_matrix)
- DEFBINOP (mm_oscm_mul_herm, MESS_matrix, octave_sparse_complex_matrix)
- DEFBINOP (mm_oscm_trans_mul, MESS_matrix, octave_sparse_complex_matrix)
- DEFBINOP (mm_oscm_herm_mul, MESS_matrix, octave_sparse_complex_matrix)
- DEFBINOP (mm_oscm_ldiv, MESS_matrix, octave_sparse_complex_matrix)
- DEFBINOP (mm_oscm_trans_ldiv, MESS_matrix, octave_sparse_complex_matrix)
- DEFBINOP (mm_oscm_herm_ldiv, MESS_matrix, octave_sparse_complex_matrix)
- DEFBINOP (oscm_mm_add, octave_sparse_complex_matrix, MESS_matrix)
- DEFBINOP (oscm_mm_sub, octave_sparse_complex_matrix, MESS_matrix)
- DEFBINOP (oscm_mm_mul, octave_sparse_complex_matrix, MESS_matrix)
- DEFBINOP (oscm_mm_mul_trans, octave_sparse_complex_matrix, MESS_matrix)
- DEFBINOP (oscm_mm_mul_herm, octave_sparse_complex_matrix, MESS_matrix)
- DEFBINOP (oscm_mm_trans_mul, octave_sparse_complex_matrix, MESS_matrix)
- DEFBINOP (oscm_mm_herm_mul, octave_sparse_complex_matrix, MESS_matrix)
- DEFBINOP (oscm_mm_ldiv, octave_sparse_complex_matrix, MESS_matrix)
- DEFBINOP (oscm_mm_trans_ldiv, octave_sparse_complex_matrix, MESS_matrix)
- DEFBINOP (oscm_mm_herm_ldiv, octave_sparse_complex_matrix, MESS_matrix)
- DEFBINOP (os_mm_mul, octave_scalar, MESS_matrix)
- DEFBINOP (mm_os_mul, MESS_matrix, octave_scalar)
- DEFBINOP (ocs_mm_mul, octave_complex_scalar, MESS_matrix)
- DEFBINOP (mm_ocs_mul, MESS_matrix, octave_complex_scalar)

10.487.1 Macro Definition Documentation

10.487.1.1 #define OCTMESS_GENERATOR(*SHORTTYPEA, TYPEA, SHORTTYPEB, TYPEB*)

Value:

```
DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _add,          TYPEA, TYPEB){return
      op_add(        dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _sub,          TYPEA, TYPEB){return
      op_sub(        dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _mul,          TYPEA, TYPEB){return
      op_mul(        dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _mul_trans,    TYPEA, TYPEB){return
      op_mul_trans(  dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _mul_herm,     TYPEA, TYPEB){return
      op_mul_herm(   dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _trans_mul,    TYPEA, TYPEB){return
      op_trans_mul(  dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _herm_mul,     TYPEA, TYPEB){return
      op_herm_mul(   dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _ldiv,         TYPEA, TYPEB){return
      op_ldiv(       dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _trans_ldiv,   TYPEA, TYPEB){return
      op_trans_ldiv( dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _herm_ldiv,    TYPEA, TYPEB){return
      op_herm_ldiv(  dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}    \
```

10.487.1.2 #define OCTMESS_GENERATOR(*SHORTTYPEA, TYPEA, SHORTTYPEB, TYPEB*)

Value:

```
INSTALL_BINOP(op_add,            TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _add);          \
    INSTALL_BINOP(op_sub,            TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _sub);          \
    INSTALL_BINOP(op_mul,            TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _mul);          \
    INSTALL_BINOP(op_trans_mul,      TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _trans_mul);
      \
    INSTALL_BINOP(op_herm_mul,       TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _herm_mul);
      \
    INSTALL_BINOP(op_mul_trans,      TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _mul_trans);
      \
    INSTALL_BINOP(op_mul_herm,       TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _mul_herm);
      \
    INSTALL_BINOP(op_ldiv,           TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _ldiv);          \
    INSTALL_BINOP(op_trans_ldiv,     TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _trans_ldiv);
      \
    INSTALL_BINOP(op_herm_ldiv,      TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _herm_ldiv);
      \
```

10.487.2 Function Documentation

10.487.2.1 DEFBINOP (mm_mm_add , MESS_matrix , MESS_matrix)

10.487.2.2 DEFBINOP (mm_mm_mul , MESS_matrix , MESS_matrix)

10.487.2.3 DEFBINOP (mm_mm_mul_trans , MESS_matrix , MESS_matrix)

10.487.2.4 DEFBINOP (mm_mm_mul_herm , MESS_matrix , MESS_matrix)

10.487.2.5 DEFBINOP (mm_mm_trans_mul , MESS_matrix , MESS_matrix)

10.487.2.6 DEFBINOP (mm_mm_sub , MESS_matrix , MESS_matrix)

10.487.2.7 DEFBINOP (mm_mm_herm_mul , MESS_matrix , MESS_matrix)

10.487.2.8 DEFBINOP (mm_mm_ldiv , MESS_matrix , MESS_matrix)

10.487.2.9 DEFBINOP (mm_mm_trans_ldiv , MESS_matrix , MESS_matrix)

10.487.2.10 DEFBINOP (mm_mm_herm_ldiv , MESS_matrix , MESS_matrix)

10.487.2.11 DEFBINOP (mm_om_add , MESS_matrix , octave_matrix)

10.487.2.12 DEFBINOP (mm_om_sub , MESS_matrix , octave_matrix)

10.487.2.13 DEFBINOP (mm_om_mul , MESS_matrix , octave_matrix)

10.487.2.14 DEFBINOP (mm_om_mul_trans , MESS_matrix , octave_matrix)

10.487.2.15 DEFBINOP (mm_om_mul_herm , MESS_matrix , octave_matrix)

10.487.2.16 DEFBINOP (mm_om_trans_mul , MESS_matrix , octave_matrix)

10.487.2.17 DEFBINOP (mm_om_herm_mul , MESS_matrix , octave_matrix)

10.487.2.18 DEFBINOP (mm_om_ldiv , MESS_matrix , octave_matrix)

10.487.2.19 DEFBINOP (mm_om_trans_ldiv , MESS_matrix , octave_matrix)

10.487.2.20 DEFBINOP (mm_om_herm_ldiv , MESS_matrix , octave_matrix)

10.487.2.21 DEFBINOP (om_mm_herm_mul , octave_matrix , MESS_matrix)

10.487.2.22 DEFBINOP (om_mm_ldiv , octave_matrix , MESS_matrix)

10.487.2.23 DEFBINOP (om_mm_trans_ldiv , octave_matrix , MESS_matrix)

10.487.2.24 DEFBINOP (om_mm_herm_ldiv , octave_matrix , MESS_matrix)

10.487.2.25 DEFBINOP (om_mm_add , octave_matrix , MESS_matrix)

10.487.2.26 DEFBINOP (om_mm_sub , octave_matrix , MESS_matrix)

10.487.2.27 DEFBINOP (om_mm_mul , octave_matrix , MESS_matrix)

10.487.2.28 DEFBINOP (om_mm_mul_trans , octave_matrix , MESS_matrix)

10.487.2.29 DEFBINOP (om_mm_mul_herm , octave_matrix , MESS_matrix)

10.487.2.30 DEFBINOP (om_mm_trans_mul , octave_matrix , MESS_matrix)

10.487.2.31 DEFBINOP (mm_ocm_add , MESS_matrix , octave_complex_matrix)

10.487.2.32 DEFBINOP (mm_ocm_sub , MESS_matrix , octave_complex_matrix)

10.487.2.33 DEFBINOP (mm_ocm_mul , MESS_matrix , octave_complex_matrix)

10.487.2.34 DEFBINOP (mm_ocm_mul_trans , MESS_matrix , octave_complex_matrix)

10.487.2.35 DEFBINOP (mm_ocm_mul_herm , MESS_matrix , octave_complex_matrix)

10.487.2.36 DEFBINOP (mm_ocm_trans_mul , MESS_matrix , octave_complex_matrix)

10.487.2.37 DEFBINOP (mm_ocm_herm_mul , MESS_matrix , octave_complex_matrix)

10.487.2.38 DEFBINOP (mm_ocm_ldiv , MESS_matrix , octave_complex_matrix)

10.487.2.39 DEFBINOP (mm_ocm_trans_ldiv , MESS_matrix , octave_complex_matrix)

10.487.2.40 DEFBINOP (mm_ocm_herm_ldiv , MESS_matrix , octave_complex_matrix)

10.487.2.41 DEFBINOP (ocm_mm_add , octave_complex_matrix , MESS_matrix)

10.487.2.42 DEFBINOP (ocm_mm_sub , octave_complex_matrix , MESS_matrix)

10.487.2.43 DEFBINOP (ocm_mm_mul , octave_complex_matrix , MESS_matrix)

10.487.2.44 DEFBINOP (ocm_mm_mul_trans , octave_complex_matrix , MESS_matrix)

10.487.2.45 DEFBINOP (ocm_mm_mul_herm , octave_complex_matrix , MESS_matrix)

10.487.2.46 DEFBINOP (ocm_mm_herm_ldiv , octave_complex_matrix , MESS_matrix)

10.487.2.47 DEFBINOP (ocm_mm_trans_mul , octave_complex_matrix , MESS_matrix)

10.487.2.48 DEFBINOP (ocm_mm_herm_mul , octave_complex_matrix , MESS_matrix)

10.487.2.49 DEFBINOP (ocm_mm_ldiv , octave_complex_matrix , MESS_matrix)

10.487.2.50 DEFBINOP (ocm_mm_trans_ldiv , octave_complex_matrix , MESS_matrix)

10.487.2.51 DEFBINOP (mm_osm_mul_trans , MESS_matrix , octave_sparse_matrix)

10.487.2.52 DEFBINOP (mm_osm_trans_ldiv , MESS_matrix , octave_sparse_matrix)

10.487.2.53 DEFBINOP (mm_osm_herm_ldiv , MESS_matrix , octave_sparse_matrix)

10.487.2.54 DEFBINOP (mm_osm_ldiv , MESS_matrix , octave_sparse_matrix)

10.487.2.55 DEFBINOP (mm_osm_add , MESS_matrix , octave_sparse_matrix)

10.487.2.56 DEFBINOP (mm_osm_sub , MESS_matrix , octave_sparse_matrix)

10.487.2.57 DEFBINOP (mm_osm_mul , MESS_matrix , octave_sparse_matrix)

10.487.2.58 DEFBINOP (mm_osm_mul_herm , MESS_matrix , octave_sparse_matrix)

10.487.2.59 DEFBINOP (mm_osm_trans_mul , MESS_matrix , octave_sparse_matrix)

10.487.2.60 DEFBINOP (mm_osm_herm_mul , MESS_matrix , octave_sparse_matrix)

10.487.2.61 DEFBINOP (osm_mm_add , octave_sparse_matrix , MESS_matrix)

10.487.2.62 DEFBINOP (osm_mm_sub , octave_sparse_matrix , MESS_matrix)

10.487.2.63 DEFBINOP (osm_mm_mul , octave_sparse_matrix , MESS_matrix)

10.487.2.64 DEFBINOP (osm_mm_trans_ldiv , octave_sparse_matrix , MESS_matrix)

10.487.2.65 DEFBINOP (osm_mm_mul_trans , octave_sparse_matrix , MESS_matrix)

10.487.2.66 DEFBINOP (osm_mm_mul_herm , octave_sparse_matrix , MESS_matrix)

10.487.2.67 DEFBINOP (osm_mm_herm_mul , octave_sparse_matrix , MESS_matrix)

10.487.2.68 DEFBINOP (osm_mm_ldiv , octave_sparse_matrix , MESS_matrix)

10.487.2.69 DEFBINOP (osm_mm_trans_mul , octave_sparse_matrix , MESS_matrix)

10.487.2.70 DEFBINOP (osm_mm_herm_ldiv , octave_sparse_matrix , MESS_matrix)

10.487.2.71 DEFBINOP (mm_oscm_add , MESS_matrix , octave_sparse_complex_matrix)

10.487.2.72 DEFBINOP (mm_oscm_mul_trans , MESS_matrix , octave_sparse_complex_matrix)

10.487.2.73 DEFBINOP (mm_oscm_sub , MESS_matrix , octave_sparse_complex_matrix)

10.487.2.74 DEFBINOP (mm_oscm_ldiv , MESS_matrix , octave_sparse_complex_matrix)

10.487.2.75 DEFBINOP (mm_oscm_mul_herm , MESS_matrix , octave_sparse_complex_matrix)

10.487.2.76 DEFBINOP (mm_oscm_trans_mul , MESS_matrix , octave_sparse_complex_matrix)

10.487.2.77 DEFBINOP (mm_oscm_herm_mul , MESS_matrix , octave_sparse_complex_matrix)

10.487.2.78 DEFBINOP (mm_oscm_trans_ldiv , MESS_matrix , octave_sparse_complex_matrix)

10.487.2.79 DEFBINOP (mm_oscm_herm_ldiv , MESS_matrix , octave_sparse_complex_matrix)

10.487.2.80 DEFBINOP (mm_oscm_mul , MESS_matrix , octave_sparse_complex_matrix)

10.487.2.81 DEFBINOP (oscm_mm_mul_trans , octave_sparse_complex_matrix , MESS_matrix)

10.487.2.82 DEFBINOP (oscm_mm_sub , octave_sparse_complex_matrix , MESS_matrix)

10.487.2.83 DEFBINOP (oscm_mm_ldiv , octave_sparse_complex_matrix , MESS_matrix)

10.487.2.84 DEFBINOP (oscm_mm_mul_herm , octave_sparse_complex_matrix , MESS_matrix)

10.487.2.85 DEFBINOP (oscm_mm_trans_mul , octave_sparse_complex_matrix , MESS_matrix)

10.487.2.86 DEFBINOP (oscm_mm_mul , octave_sparse_complex_matrix , MESS_matrix)

10.487.2.87 DEFBINOP (oscm_mm_herm_mul , octave_sparse_complex_matrix , MESS_matrix)

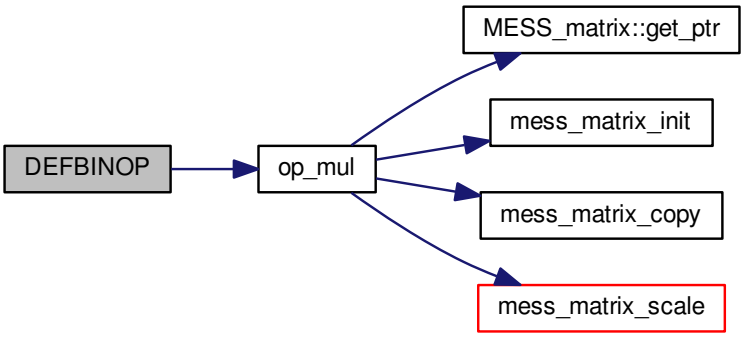
10.487.2.88 DEFBINOP (oscm_mm_trans_ldiv , octave_sparse_complex_matrix , MESS_matrix)

10.487.2.89 DEFBINOP (oscm_mm_herm_ldiv , octave_sparse_complex_matrix , MESS_matrix)

10.487.2.90 DEFBINOP (oscm_mm_add , octave_sparse_complex_matrix , MESS_matrix)

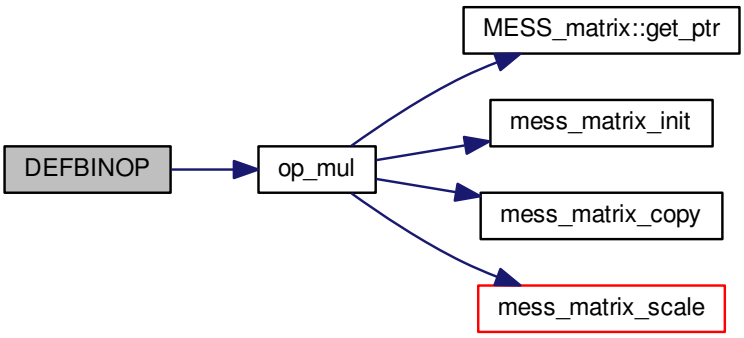
10.487.2.91 DEFBINOP (os_mm_mul , octave_scalar , MESS_matrix)

Here is the call graph for this function:



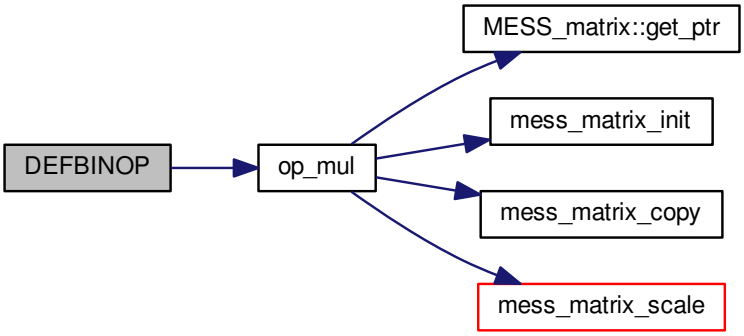
10.487.2.92 DEFBINOP (mm_os_mul , MESS_matrix , octave_scalar)

Here is the call graph for this function:



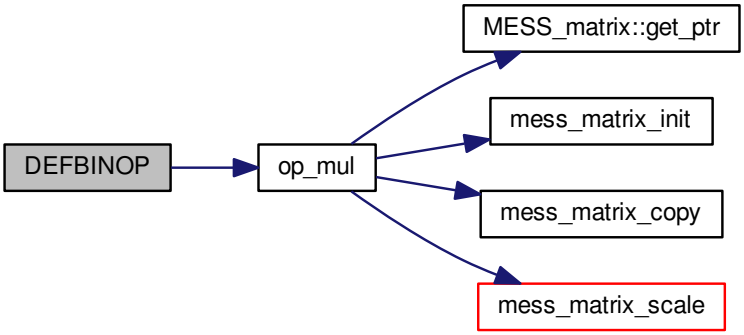
10.487.2.93 DEFBINOP (ocs_mm_mul , octave_complex_scalar , MESS_matrix)

Here is the call graph for this function:



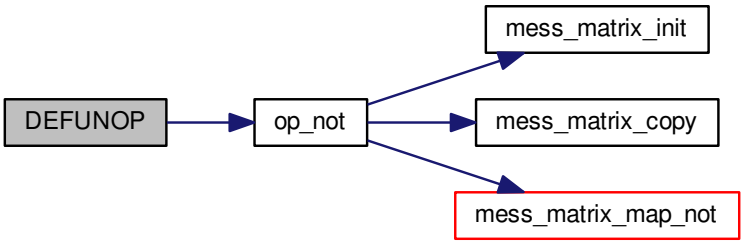
10.487.2.94 DEFBINOP (mm_ocs_mul , MESS_matrix , octave_complex_scalar)

Here is the call graph for this function:



10.487.2.95 DEFUNOP (mm_not , MESS_matrix)

Here is the call graph for this function:

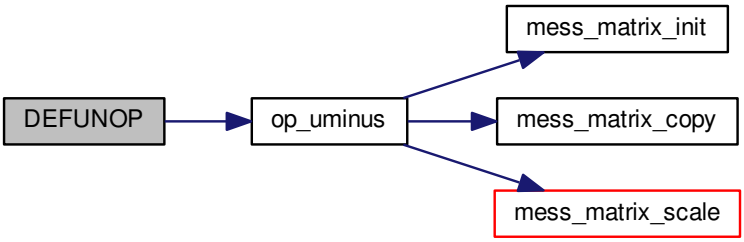


10.487.2.96 DEFUNOP (mm_uplus , MESS_matrix)

Here is the call graph for this function:

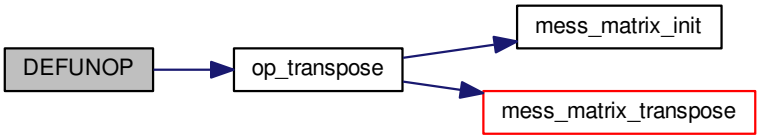
10.487.2.97 DEFUNOP (mm_uminus , MESS_matrix)

Here is the call graph for this function:



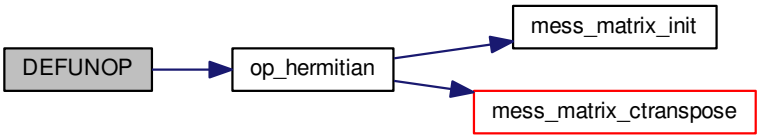
10.487.2.98 DEFUNOP (mm_transpose , MESS_matrix)

Here is the call graph for this function:



10.487.2.99 DEFUNOP (mm_hermitian , MESS_matrix)

Here is the call graph for this function:



10.488 MESS_matrix_ops_mm_generator.h File Reference

Macros

- #define OCTMESS_GENERATOR(OCTAVE_MATRIX_TYPE, OCTAVE_MATRIX_TYPE_VALUE)

10.488.1 Macro Definition Documentation

10.488.1.1 #define OCTMESS_GENERATOR(*OCTAVE_MATRIX_TYPE, OCTAVE_MATRIX_TYPE_VALUE*)

10.489 MESS_matrix_ops_mm_mm.cc File Reference

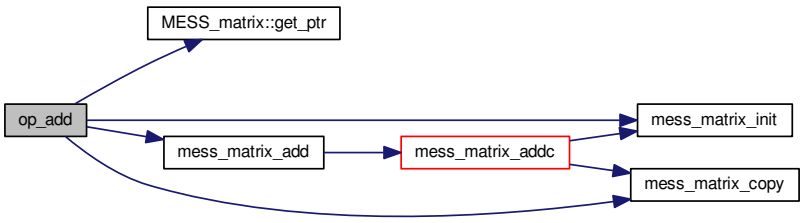
Functions

- octave_value op_not (const MESS_matrix &m)
- octave_value op_uplus (const MESS_matrix &m)
- octave_value op_uminus (const MESS_matrix &m)
- octave_value op_transpose (const MESS_matrix &m)
- octave_value op_hermitian (const MESS_matrix &m)
- octave_value op_add (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_sub (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_xmul (const mess_operation_t opA, const MESS_matrix &m1, const mess_operation_t opB, const MESS_matrix &m2)
- octave_value op_mul (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_trans_mul (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_herm_mul (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_mul_trans (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_mul_herm (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_xldiv (const MESS_matrix &m1, const mess_operation_t opA, const MESS_matrix &m2)
- octave_value op_ldiv (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_trans_ldiv (const MESS_matrix &m1, const MESS_matrix &m2)
- octave_value op_herm_ldiv (const MESS_matrix &m1, const MESS_matrix &m2)

10.489.1 Function Documentation

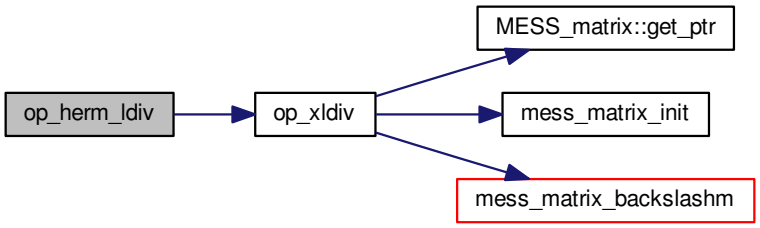
10.489.1.1 octave_value op_add (const MESS_matrix & *m1,* const MESS_matrix & *m2*)

Here is the call graph for this function:



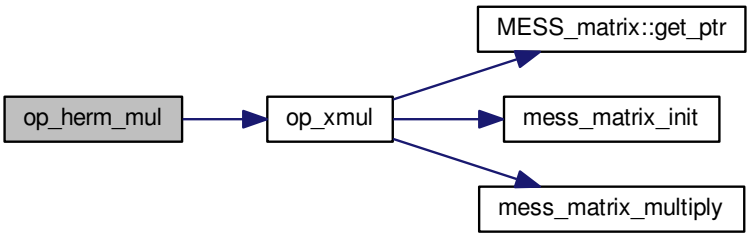
10.489.1.2 octave_value op_herm_ldiv (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:



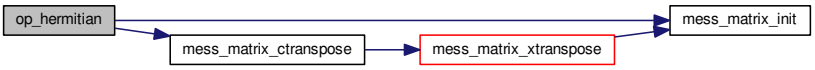
10.489.1.3 octave_value op_herm_mul (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:



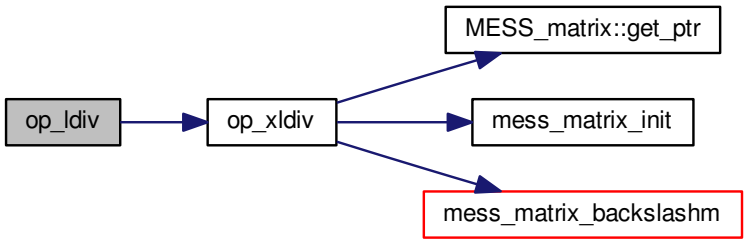
10.489.1.4 octave_value op_hermitian (const MESS_matrix & *m*)

Here is the call graph for this function:



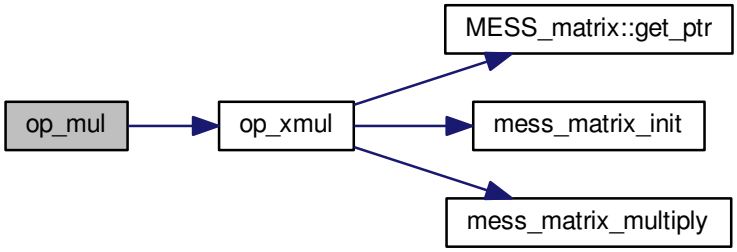
10.489.1.5 octave_value op_ldiv (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:



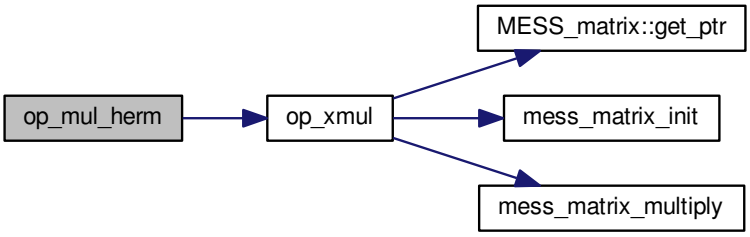
10.489.1.6 octave_value op_mul (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:



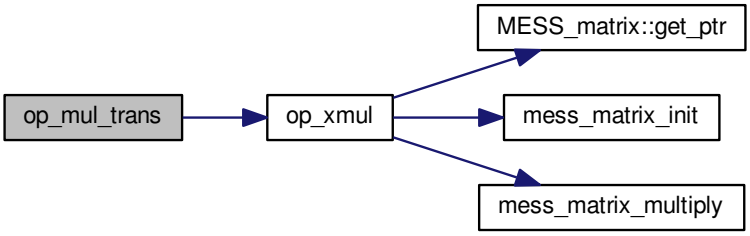
10.489.1.7 octave_value op_mul_herm (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:



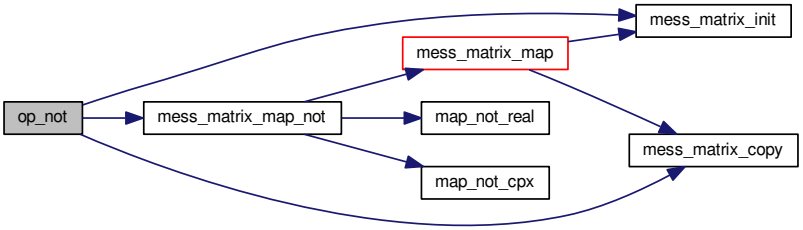
10.489.1.8 octave_value op_mul_trans (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:



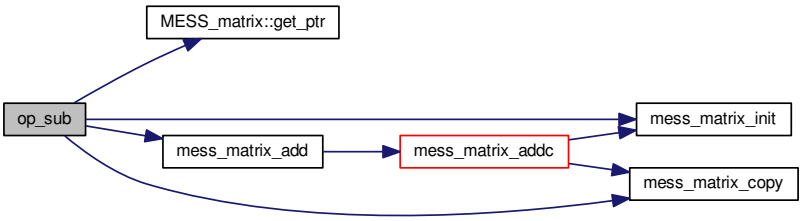
10.489.1.9 octave_value op_not (const MESS_matrix & *m*)

Here is the call graph for this function:



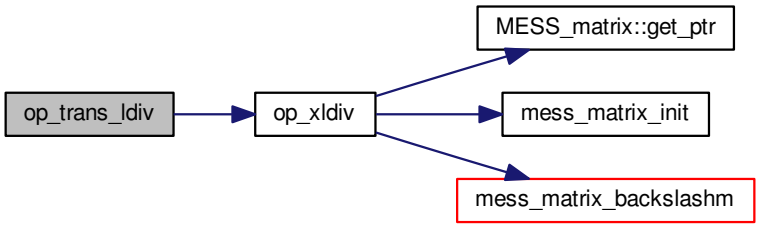
10.489.1.10 octave_value op_sub (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:



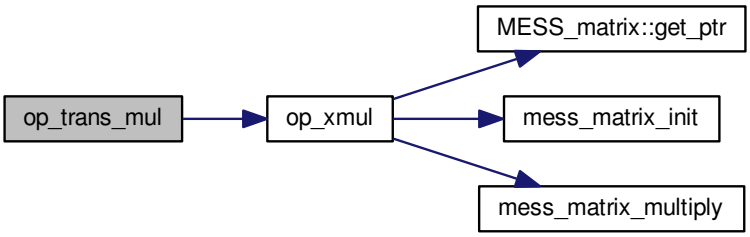
10.489.1.11 octave_value op_trans_ldiv (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:



10.489.1.12 octave_value op_trans_mul (const MESS_matrix & *m1*, const MESS_matrix & *m2*)

Here is the call graph for this function:

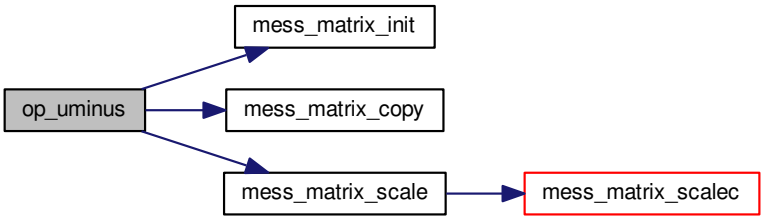


10.489.1.13 octave_value op_transpose (const MESS_matrix & *m*)

Here is the call graph for this function:

10.489.1.14 octave_value op_uminus (const MESS_matrix & *m*)

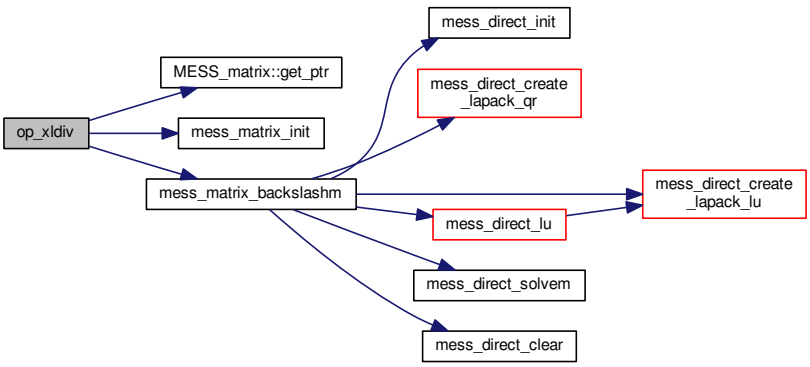
Here is the call graph for this function:



10.489.1.15 octave_value op_uplus (const MESS_matrix & *m*)

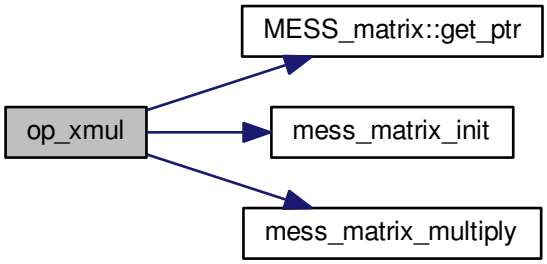
10.489.1.16 octave_value op_xldiv (const MESS_matrix & *m1*, const mess_operation_t *opA*, const MESS_matrix & *m2*)

Here is the call graph for this function:



10.489.1.17 octave_value op_xmul (const mess_operation_t *opA*, const MESS_matrix & *m1*, const mess_operation_t *opB*, const MESS_matrix & *m2*)

Here is the call graph for this function:



10.490 MESS_matrix_ops_mm_ocm.cc File Reference

Functions

- OCTMESS_GENERATOR (octave_complex_matrix, complex_matrix_value)

10.490.1 Function Documentation

10.490.1.1 OCTMESS_GENERATOR (octave_complex_matrix , complex_matrix_value)

10.491 MESS_matrix_ops_mm_om.cc File Reference

Functions

- OCTMESS_GENERATOR (octave_matrix, matrix_value)

10.491.1 Function Documentation

10.491.1.1 OCTMESS_GENERATOR (octave_matrix , matrix_value)

10.492 MESS_matrix_ops_mm_os.cc File Reference

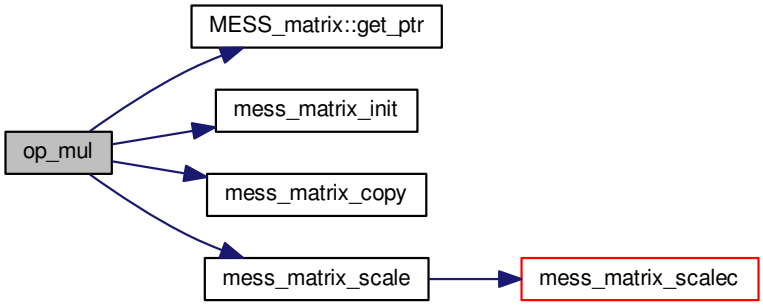
Functions

- octave_value op_mul (const octave_scalar &s, const MESS_matrix &m)
- octave_value op_mul (const MESS_matrix &m, const octave_scalar &s)

10.492.1 Function Documentation

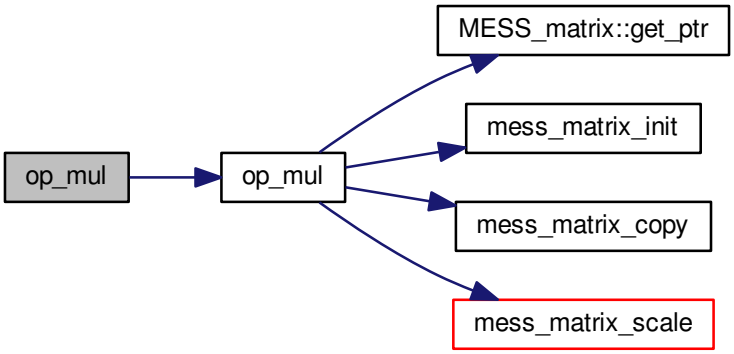
10.492.1.1 octave_value op_mul (const octave_scalar & *s*, const MESS_matrix & *m*)

Here is the call graph for this function:



10.492.1.2 octave_value op_mul (const MESS_matrix & *m*, const octave_scalar & *s*)

Here is the call graph for this function:



10.493 MESS_matrix_ops_mm_osc.cc File Reference

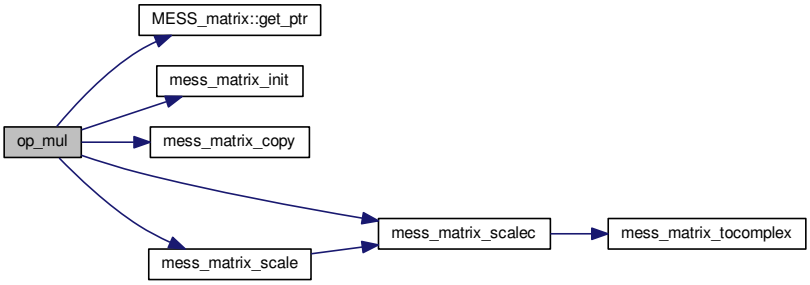
Functions

- octave_value op_mul (const octave_complex_scalar &s, const MESS_matrix &m)
- octave_value op_mul (const MESS_matrix &m, const octave_complex_scalar &s)

10.493.1 Function Documentation

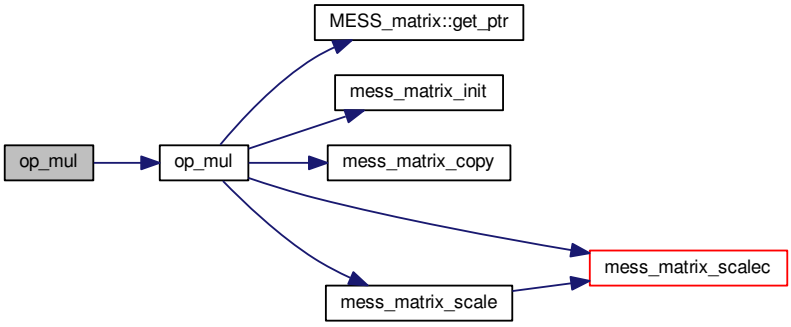
10.493.1.1 octave_value op_mul (const octave_complex_scalar & *s*, const MESS_matrix & *m*)

Here is the call graph for this function:



10.493.1.2 octave_value op_mul (const MESS_matrix & *m*, const octave_complex_scalar & *s*)

Here is the call graph for this function:



10.494 MESS_matrix_ops_mm_oscm.cc File Reference

Functions

- OCTMESS_GENERATOR (octave_sparse_complex_matrix, sparse_complex_matrix_value)

10.494.1 Function Documentation

10.494.1.1 OCTMESS_GENERATOR (octave_sparse_complex_matrix , sparse_complex_matrix_value)

10.495 MESS_matrix_ops_mm_osm.cc File Reference

Functions

- OCTMESS_GENERATOR (octave_sparse_matrix, sparse_matrix_value)

10.495.1 Function Documentation

10.495.1.1 OCTMESS_GENERATOR (octave_sparse_matrix , sparse_matrix_value)

10.496 MESS_misc.cc File Reference

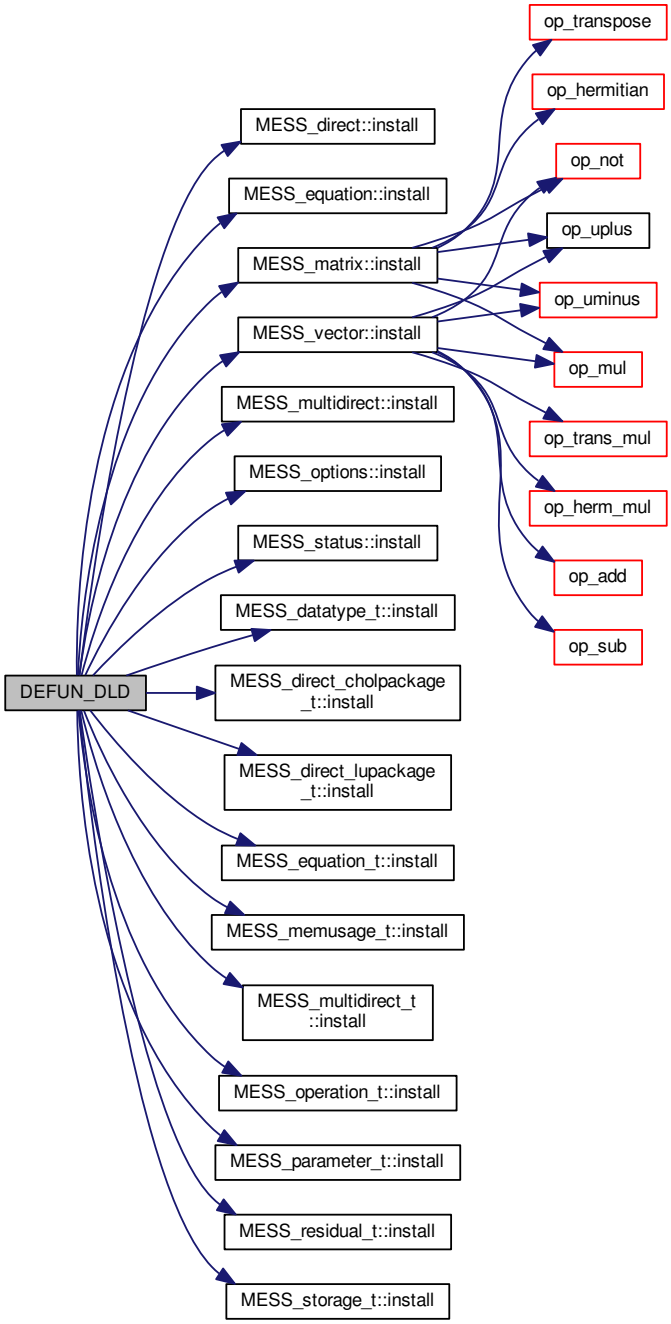
Functions

- DEFUN_DLD (mess_init, args, nargout,"Initializes OCT-M.E.S.S.")
- DEFUN_DLD (mess_eps, args, nargout,"Interface to mess_eps.")
- DEFUN_DLD (mess_version, args, nargout,"Interface to mess_version.")

10.496.1 Function Documentation

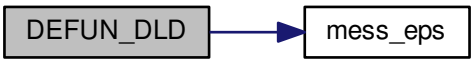
10.496.1.1 DEFUN_DLD (mess_init , args , nargout , "Initializes OCT-M.E.S.S.")

Here is the call graph for this function:



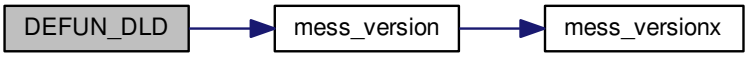
10.496.1.2 DEFUN_DLD (mess_eps , args , nargout , "Interface to mess_eps.")

Here is the call graph for this function:



10.496.1.3 DEFUN_DLD (mess_version , args , nargout , "Interface to mess_version.")

Here is the call graph for this function:



10.497 MESS_multidirect.cc File Reference

Functions

- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_multidirect,"mess_multidirect","mess_multidirect")

10.497.1 Function Documentation

10.497.1.1 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_multidirect , "mess_multidirect" , "mess_multidirect")

10.498 MESS_multidirect.h File Reference

Data Structures

- class MESS_multidirect

10.499 MESS_multidirect_install.cc File Reference

10.500 mess_options.c File Reference

Functions

- mess_options mess_options_from_mexmess (const mxArray ∗instance)
- mxArray ∗ mess_options_to_mexmess (mess_options opt)

10.500.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.501 MESS_options.cc File Reference

Functions

- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_options,"mess_options","mess_options")

Variables

- static const char ∗const fieldnames [ ]
- static const size_t len_fieldnames = 23

10.501.1 Function Documentation

10.501.1.1 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_options , "mess_options" , "mess_options")

10.501.2 Variable Documentation

10.501.2.1 const char∗ const fieldnames[ ] [static]

Initial value:

```
=
{
    "type",

    "adi_shifts_p", "adi_shifts_paratype",
    "adi_shifts_arp_p", "adi_shifts_arp_m", "adi_shifts_l0", "adi_shifts_b0",
    "proj_space",

    "adi_maxit", "adi_res2_tol", "adi_res2c_tol", "adi_rel_change_tol",
    "adi_output", "residual_method",

    "nm_maxit", "nm_res2_tol", "nm_output", "nm_linesearch",
    "nm_singleshifts", "nm_gpStep",

    "memory_usage", "K0", "W"
}
```

10.501.2.2 const size_t len_fieldnames = 23 [static]

10.502 MESS_options.cc File Reference

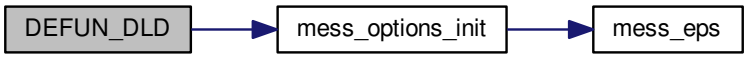
Functions

- DEFUN_DLD (mess_options, args, nargout,"blabla")
- DEFUN_DLD (mess_options_print, args, nargout,"blabla")

10.502.1 Function Documentation

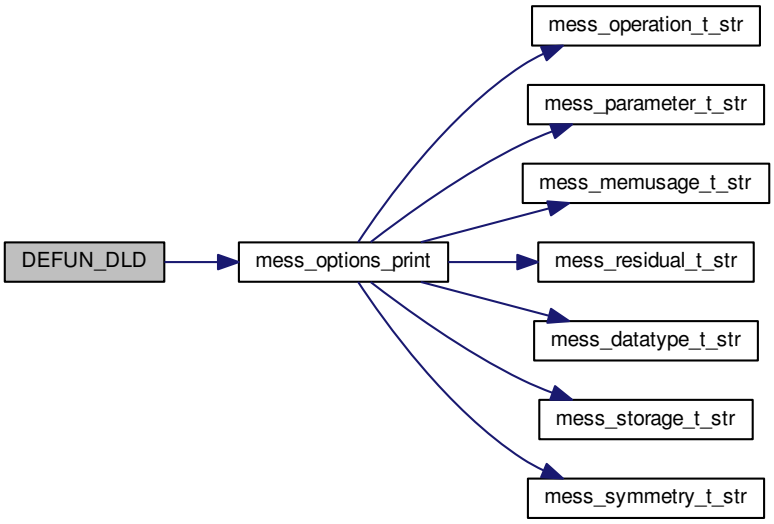
10.502.1.1 DEFUN_DLD (mess_options , args , nargout , "blabla")

Here is the call graph for this function:



10.502.1.2 DEFUN_DLD (mess_options_print , args , nargout , "blabla")

Here is the call graph for this function:



10.503 MESS_options.h File Reference

Data Structures

- class MESS_options

10.504 MESS_options_install.cc File Reference

10.505 mess_set_errorlevel.c File Reference

Functions

- PyObject ∗ pymess_set_errorlevel (PyObject ∗self, PyObject ∗args)

  Wrapper around mess_set_errorlevel for use with `Python`.

10.505.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Nitin Prasad (nitin.prasad93@gmail.com, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.506 mess_status.c File Reference

Functions

- static void set_status_properties (mxArray *m_stat, mess_status stat)
- mxArray * mess_status_to_mexmess (mess_status stat)

10.506.1 Detailed Description

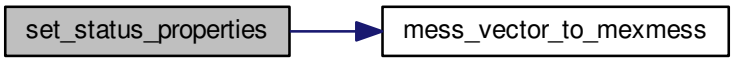
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.506.2 Function Documentation

10.506.2.1 static void set_status_properties (mxArray * *m_stat*, mess_status *stat*) [static]

Here is the call graph for this function:



10.507 MESS_status.cc File Reference

Functions

- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_status,"mess_status","mess_status")

10.507.1 Function Documentation

10.507.1.1 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_status , "mess_status" , "mess_status")

10.508 MESS_status.cc File Reference

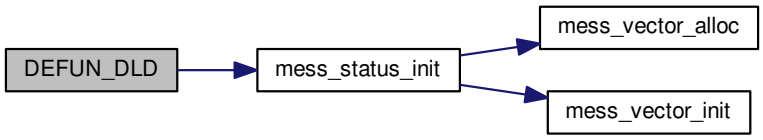
Functions

- DEFUN_DLD (mess_status, args, nargout,"blabla")
- DEFUN_DLD (mess_status_print, args, nargout,"blabla")

10.508.1 Function Documentation

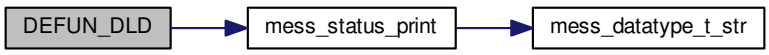
10.508.1.1 DEFUN_DLD (mess_status , args , nargout , "blabla")

Here is the call graph for this function:



10.508.1.2 DEFUN_DLD (mess_status_print , args , nargout , "blabla")

Here is the call graph for this function:



10.509 MESS_status.h File Reference

Data Structures

- class MESS_status

10.510 MESS_status_install.cc File Reference

10.511 MESS_to_oct.cc File Reference

Functions

- DEFUN_DLD (mess_to_oct, args, nargout,"blabla")

10.511.1 Function Documentation

10.511.1.1 DEFUN_DLD (mess_to_oct , args , nargout , "blabla")

10.512 mess_vector.c File Reference

Functions

- mess_vector mess_vector_from_mexmess (const mxArray *Amatlab)

  Copy a vector/matrix from MATLAB *to a column vector in* ***M.E.S.S.***.

- mxArray * mess_vector_to_mexmess (mess_vector A)

  Copy a vector from ***M.E.S.S.*** *to* MATLAB.

10.512.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.513 MESS_vector.cc File Reference

Macros

- #define OCTMESS_GENERATOR(MAPNAME)

Functions

- DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_vector,"mess_vector","mess_vector")

10.513.1 Macro Definition Documentation

10.513.1.1 #define OCTMESS_GENERATOR(*MAPNAME*)

Value:

```
case umap_##MAPNAME:                                                                                    \
        ret = mess_vector_copy(this->ptr, result);
      OCTMESS_ERROR(ret,(ret!=0),mess_vector_copy);            \
        ret = mess_vector_map_##MAPNAME(result);    OCTMESS_ERROR(ret,(ret!=0),
      mess_vector_map_##MAPNAME);  \
        return new MESS_vector(result);
                  \
```

10.513.2 Function Documentation

10.513.2.1 DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA (MESS_vector , "mess_vector" , "mess_vector")

10.514 MESS_vector.cc File Reference

Macros

- #define OCTMESS_MESS_VECTOR_PRINT(PRINTFCT)

Functions

- DEFUN_DLD (mess_vector, args, nargout,"blabla")
- DEFUN_DLD (mess_vector_print, args, nargout,"Interface to ""mess_vector_print")
- DEFUN_DLD (mess_vector_printinfo, args, nargout,"Interface to ""mess_vector_printinfo")
- DEFUN_DLD (mess_vector_printshort, args, nargout,"Interface to ""mess_vector_printshort")

10.514.1 Macro Definition Documentation

10.514.1.1 #define OCTMESS_MESS_VECTOR_PRINT(*PRINTFCT*)

Value:

```
DEFUN_DLD(PRINTFCT, args, nargout, "Interface to " #PRINTFCT){
                                    \
    int ret = 0;
                        \
    if(nargout || args.length()!=1 || args(0).type_id() != MESS_vector::static_type_id()){
                        \
        print_usage();
                        \
    }else{
                        \
        ret = PRINTFCT((mess_vector)const_cast<MESS_vector &> (dynamic_cast<const MESS_vector &>
       (args(0).get_rep())));      \
        OCTMESS_ERROR(ret,(ret!=0),PRINTFCT);
                                            \
    }
                        \
    return octave_value_list();
                        \
}
```

10.514.2 Function Documentation

10.514.2.1 DEFUN_DLD (mess_vector , args , nargout , "blabla")

10.514.2.2 DEFUN_DLD (mess_vector_print , args , nargout , "Interface to ""mess_vector_print")

10.514.2.3 DEFUN_DLD (mess_vector_printinfo , args , nargout , "Interface to ""mess_vector_printinfo")

10.514.2.4 DEFUN_DLD (mess_vector_printshort , args , nargout , "Interface to ""mess_vector_printshort")

10.515 MESS_vector.h File Reference

Data Structures

- class MESS_vector

Macros

- #define OCTMESS_GENERATOR(TYPEA, TYPEB)

Functions

- octave_value op_mul (const octave_scalar &s, const MESS_vector &v)
- octave_value op_mul (const MESS_vector &v, const octave_scalar &s)
- octave_value op_mul (const octave_complex_scalar &s, const MESS_vector &v)
- octave_value op_mul (const MESS_vector &v, const octave_complex_scalar &s)
- octave_value op_not (const MESS_vector &v)
- octave_value op_uplus (const MESS_vector &v)
- octave_value op_uminus (const MESS_vector &v)
- octave_value op_add (const MESS_vector &v1, const MESS_vector &v2)
- octave_value op_sub (const MESS_vector &v1, const MESS_vector &v2)
- octave_value op_trans_mul (const MESS_vector &v1, const MESS_vector &v2)
- octave_value op_herm_mul (const MESS_vector &v1, const MESS_vector &v2)
- octave_value op_xmul (const mess_operation_t opA, MESS_matrix &m1, const MESS_vector &m2)
- octave_value op_mul (const MESS_matrix &m1, const MESS_vector &m2)
- octave_value op_trans_mul (const MESS_matrix &m1, const MESS_vector &m2)
- octave_value op_herm_mul (const MESS_matrix &m1, const MESS_vector &m2)
- octave_value op_xldiv (const MESS_matrix &m1, const mess_operation_t opA, const MESS_vector &m2)
- octave_value op_ldiv (const MESS_matrix &m1, const MESS_vector &m2)
- octave_value op_trans_ldiv (const MESS_matrix &m1, const MESS_vector &m2)
- octave_value op_herm_ldiv (const MESS_matrix &m1, const MESS_vector &m2)
- octave_value op_xmul (const mess_operation_t opA, octave_matrix &m1, const MESS_vector &m2)
- octave_value op_mul (const octave_matrix &m1, const MESS_vector &m2)
- octave_value op_trans_mul (const octave_matrix &m1, const MESS_vector &m2)
- octave_value op_herm_mul (const octave_matrix &m1, const MESS_vector &m2)
- octave_value op_xldiv (const octave_matrix &m1, const mess_operation_t opA, const MESS_vector &m2)
- octave_value op_ldiv (const octave_matrix &m1, const MESS_vector &m2)
- octave_value op_trans_ldiv (const octave_matrix &m1, const MESS_vector &m2)
- octave_value op_herm_ldiv (const octave_matrix &m1, const MESS_vector &m2)
- octave_value op_xmul (const mess_operation_t opA, octave_complex_matrix &m1, const MESS_vector &m2)
- octave_value op_mul (const octave_complex_matrix &m1, const MESS_vector &m2)
- octave_value op_trans_mul (const octave_complex_matrix &m1, const MESS_vector &m2)
- octave_value op_herm_mul (const octave_complex_matrix &m1, const MESS_vector &m2)
- octave_value op_xldiv (const octave_complex_matrix &m1, const mess_operation_t opA, const MESS_vector &m2)
- octave_value op_ldiv (const octave_complex_matrix &m1, const MESS_vector &m2)
- octave_value op_trans_ldiv (const octave_complex_matrix &m1, const MESS_vector &m2)
- octave_value op_herm_ldiv (const octave_complex_matrix &m1, const MESS_vector &m2)
- octave_value op_xmul (const mess_operation_t opA, octave_sparse_matrix &m1, const MESS_vector &m2)
- octave_value op_mul (const octave_sparse_matrix &m1, const MESS_vector &m2)
- octave_value op_trans_mul (const octave_sparse_matrix &m1, const MESS_vector &m2)
- octave_value op_herm_mul (const octave_sparse_matrix &m1, const MESS_vector &m2)
- octave_value op_xldiv (const octave_sparse_matrix &m1, const mess_operation_t opA, const MESS_vector &m2)
- octave_value op_ldiv (const octave_sparse_matrix &m1, const MESS_vector &m2)
- octave_value op_trans_ldiv (const octave_sparse_matrix &m1, const MESS_vector &m2)
- octave_value op_herm_ldiv (const octave_sparse_matrix &m1, const MESS_vector &m2)
- octave_value op_xmul (const mess_operation_t opA, octave_sparse_complex_matrix &m1, const MESS_↩vector &m2)
- octave_value op_mul (const octave_sparse_complex_matrix &m1, const MESS_vector &m2)
- octave_value op_trans_mul (const octave_sparse_complex_matrix &m1, const MESS_vector &m2)
- octave_value op_herm_mul (const octave_sparse_complex_matrix &m1, const MESS_vector &m2)
- octave_value op_xldiv (const octave_sparse_complex_matrix &m1, const mess_operation_t opA, const M↩ESS_vector &m2)
- octave_value op_ldiv (const octave_sparse_complex_matrix &m1, const MESS_vector &m2)
- octave_value op_trans_ldiv (const octave_sparse_complex_matrix &m1, const MESS_vector &m2)
- octave_value op_herm_ldiv (const octave_sparse_complex_matrix &m1, const MESS_vector &m2)

10.515.1 Macro Definition Documentation

10.515.1.1 #define OCTMESS_GENERATOR(*TYPEA*, *TYPEB*)

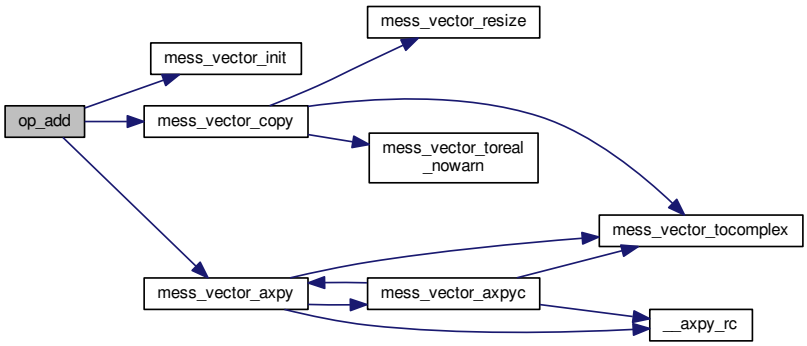
Value:

```
octave_value op_xmul(const mess_operation_t opA, TYPEA & m1, const TYPEB &m2);
                                    \
    octave_value op_mul(const TYPEA & m1, const TYPEB & m2);
                    \
    octave_value op_trans_mul(const TYPEA & m1, const TYPEB & m2);
                            \
    octave_value op_herm_mul(const TYPEA & m1, const TYPEB & m2);
                           \
    octave_value op_xldiv(const TYPEA & m1, const mess_operation_t opA, const TYPEB
       & m2);                                \
    octave_value op_ldiv(const TYPEA & m1, const TYPEB & m2);
                     \
    octave_value op_trans_ldiv(const TYPEA & m1, const TYPEB &m2);
                             \
    octave_value op_herm_ldiv(const TYPEA & m1, const TYPEB &m2);
                            \
```

10.515.2 Function Documentation

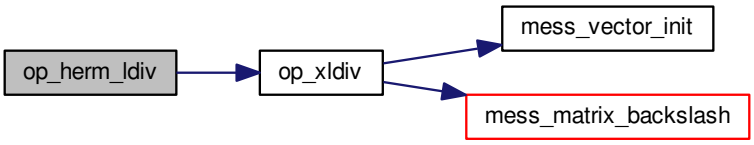
10.515.2.1 octave_value op_add (const MESS_vector & *v1*, const MESS_vector & *v2*)

Here is the call graph for this function:



10.515.2.2 octave_value op_herm_ldiv (const MESS_matrix & *m1*, const MESS_vector & *m2*)

Here is the call graph for this function:



10.515.2.3 octave_value op_herm_ldiv (const octave_matrix & *m1*, const MESS_vector & *m2*)

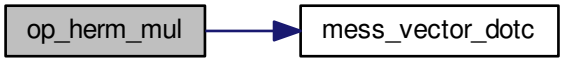
10.515.2.4 octave_value op_herm_ldiv (const octave_complex_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.5 octave_value op_herm_ldiv (const octave_sparse_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.6 octave_value op_herm_ldiv (const octave_sparse_complex_matrix & *m1*, const MESS_vector & *m2*)

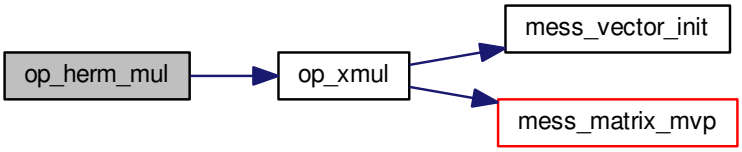
10.515.2.7 octave_value op_herm_mul (const MESS_vector & *v1*, const MESS_vector & *v2*)

Here is the call graph for this function:



10.515.2.8 octave_value op_herm_mul (const MESS_matrix & *m1*, const MESS_vector & *m2*)

Here is the call graph for this function:



10.515.2.9 octave_value op_herm_mul (const octave_matrix & *m1*, const MESS_vector & *m2*)

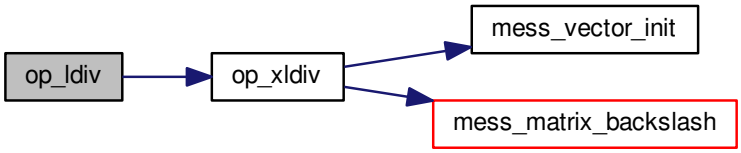
10.515.2.10 octave_value op_herm_mul (const octave_complex_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.11 octave_value op_herm_mul (const octave_sparse_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.12 octave_value op_herm_mul (const octave_sparse_complex_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.13 octave_value op_ldiv (const MESS_matrix & *m1*, const MESS_vector & *m2*)

Here is the call graph for this function:



10.515.2.14 octave_value op_ldiv (const octave_matrix & *m1*, const MESS_vector & *m2*)

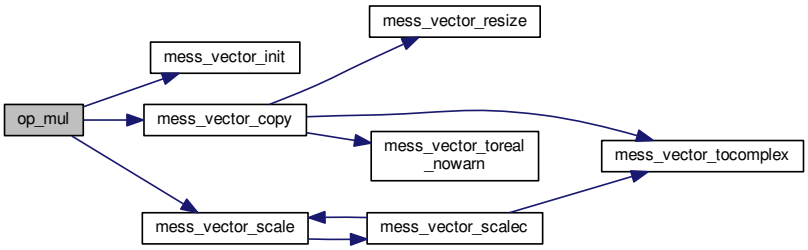
10.515.2.15 octave_value op_ldiv (const octave_complex_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.16 octave_value op_ldiv (const octave_sparse_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.17 octave_value op_ldiv (const octave_sparse_complex_matrix & *m1*, const MESS_vector & *m2*)

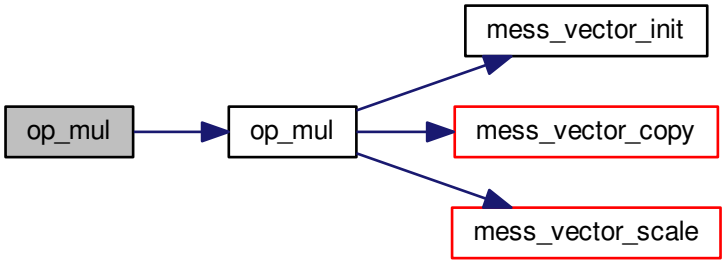
10.515.2.18 octave_value op_mul (const octave_scalar & *s*, const MESS_vector & *v*)

Here is the call graph for this function:



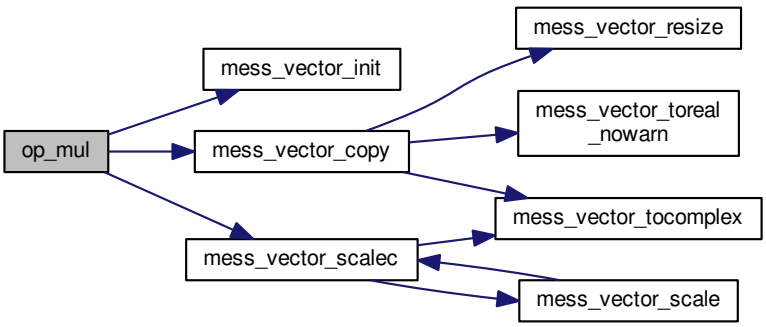
10.515.2.19 octave_value op_mul (const MESS_vector & *v*, const octave_scalar & *s*)

Here is the call graph for this function:



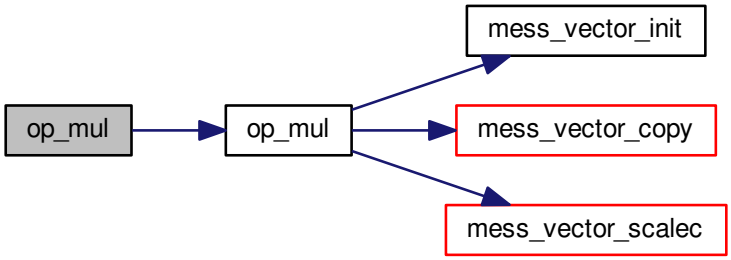
10.515.2.20 octave_value op_mul (const octave_complex_scalar & *s*, const MESS_vector & *v*)

Here is the call graph for this function:



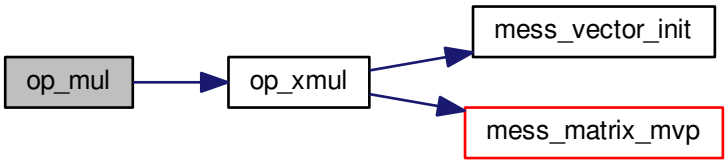
10.515.2.21 octave_value op_mul (const MESS_vector & *v*, const octave_complex_scalar & *s*)

Here is the call graph for this function:



10.515.2.22 octave_value op_mul (const MESS_matrix & *m1*, const MESS_vector & *m2*)

Here is the call graph for this function:



10.515.2.23 octave_value op_mul (const octave_matrix & *m1*, const MESS_vector & *m2*)

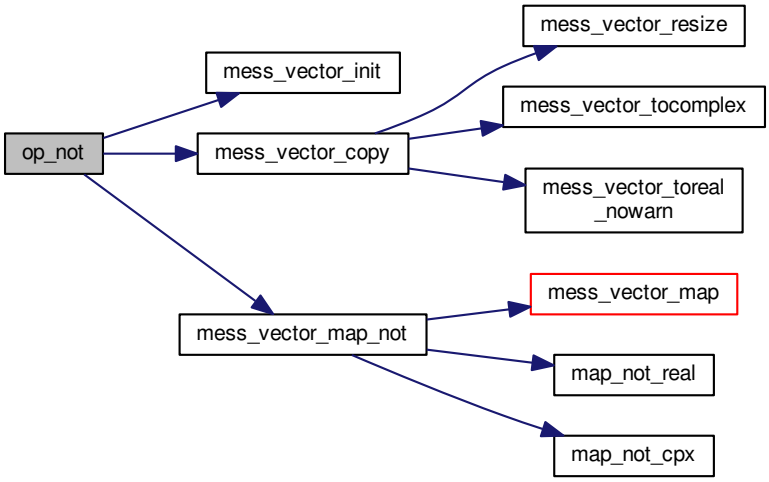
10.515.2.24 octave_value op_mul (const octave_complex_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.25 octave_value op_mul (const octave_sparse_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.26 octave_value op_mul (const octave_sparse_complex_matrix & *m1*, const MESS_vector & *m2*)

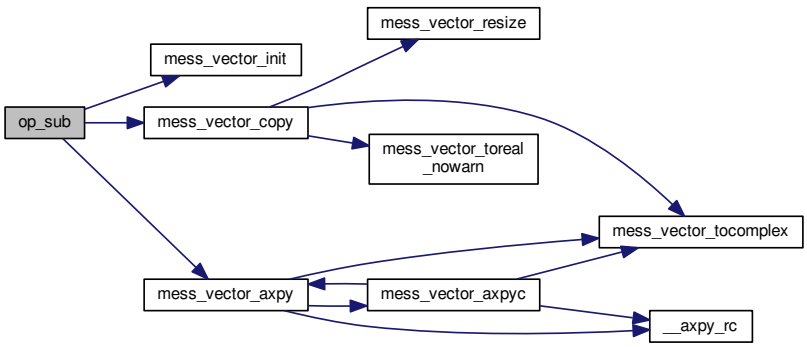
10.515.2.27 octave_value op_not (const MESS_vector & *v*)

Here is the call graph for this function:



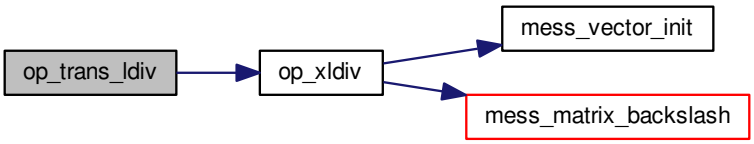
10.515.2.28 octave_value op_sub (const MESS_vector & *v1*, const MESS_vector & *v2*)

Here is the call graph for this function:



10.515.2.29 octave_value op_trans_ldiv (const MESS_matrix & *m1*, const MESS_vector & *m2*)

Here is the call graph for this function:



10.515.2.30 octave_value op_trans_ldiv (const octave_matrix & *m1*, const MESS_vector & *m2*)

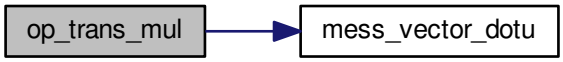
10.515.2.31 octave_value op_trans_ldiv (const octave_complex_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.32 octave_value op_trans_ldiv (const octave_sparse_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.33 octave_value op_trans_ldiv (const octave_sparse_complex_matrix & *m1*, const MESS_vector & *m2*)

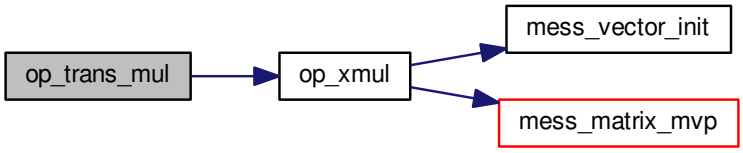
10.515.2.34 octave_value op_trans_mul (const MESS_vector & *v1*, const MESS_vector & *v2*)

Here is the call graph for this function:



10.515.2.35 octave_value op_trans_mul (const MESS_matrix & *m1*, const MESS_vector & *m2*)

Here is the call graph for this function:



10.515.2.36 octave_value op_trans_mul (const octave_matrix & *m1*, const MESS_vector & *m2*)

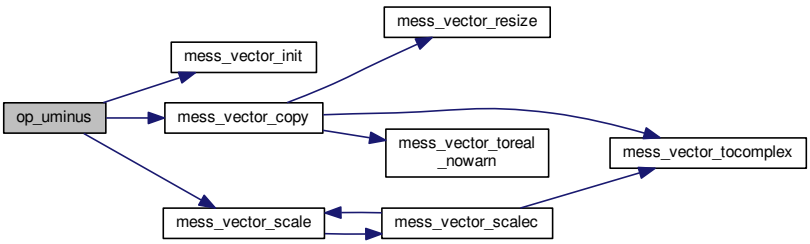
10.515.2.37 octave_value op_trans_mul (const octave_complex_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.38 octave_value op_trans_mul (const octave_sparse_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.39 octave_value op_trans_mul (const octave_sparse_complex_matrix & *m1*, const MESS_vector & *m2*)

10.515.2.40 octave_value op_uminus (const MESS_vector & *v*)

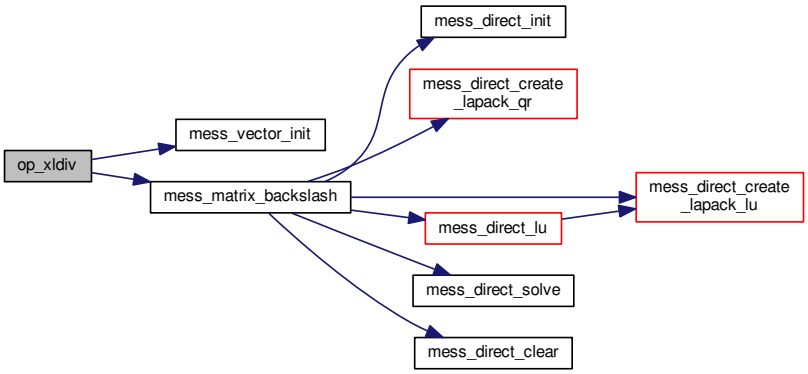
Here is the call graph for this function:



10.515.2.41 octave_value op_uplus (const MESS_vector & *v*)

10.515.2.42 octave_value op_xldiv (const MESS_matrix & *m1*, const mess_operation_t *opA*, const MESS_vector & *m2*)

Here is the call graph for this function:



10.515.2.43 octave_value op_xldiv (const octave_matrix & *m1*, const **mess_operation_t** *opA*, const **MESS_vector** & *m2*)

10.515.2.44 octave_value op_xldiv (const octave_complex_matrix & *m1*, const **mess_operation_t** *opA*, const **MESS_vector** & *m2*)

10.515.2.45 octave_value op_xldiv (const octave_sparse_matrix & *m1*, const **mess_operation_t** *opA*, const **MESS_vector** & *m2*)

10.515.2.46 octave_value op_xldiv (const octave_sparse_complex_matrix & *m1*, const **mess_operation_t** *opA*, const **MESS_vector** & *m2*)

10.515.2.47 octave_value op_xmul (const **mess_operation_t** *opA*, **MESS_matrix** & *m1*, const **MESS_vector** & *m2*)

10.515.2.48 octave_value op_xmul (const **mess_operation_t** *opA*, octave_matrix & *m1*, const **MESS_vector** & *m2*)

10.515.2.49 octave_value op_xmul (const **mess_operation_t** *opA*, octave_complex_matrix & *m1*, const **MESS_vector** & *m2*)

10.515.2.50 octave_value op_xmul (const **mess_operation_t** *opA*, octave_sparse_matrix & *m1*, const **MESS_vector** & *m2*)

10.515.2.51 octave_value op_xmul (const **mess_operation_t** *opA*, octave_sparse_complex_matrix & *m1*, const **MESS_vector** & *m2*)

10.516 MESS_vector_install.cc File Reference

Macros

- #define OCTMESS_GENERATOR(SHORTTYPEA, TYPEA, SHORTTYPEB, TYPEB)
- #define OCTMESS_GENERATOR(SHORTTYPEA, TYPEA, SHORTTYPEB, TYPEB)

Functions

- DEFUNOP (mv_not, MESS_vector)
- DEFUNOP (mv_uplus, MESS_vector)
- DEFUNOP (mv_uminus, MESS_vector)
- DEFBINOP (os_mv_mul, octave_scalar, MESS_matrix)
- DEFBINOP (mv_os_mul, MESS_matrix, octave_scalar)
- DEFBINOP (osc_mv_mul, octave_complex_scalar, MESS_matrix)
- DEFBINOP (mv_osc_mul, MESS_matrix, octave_complex_scalar)
- DEFBINOP (mv_mv_add, MESS_vector, MESS_vector)
- DEFBINOP (mv_mv_sub, MESS_vector, MESS_vector)
- DEFBINOP (mv_mv_trans_mul, MESS_vector, MESS_vector)
- DEFBINOP (mv_mv_herm_mul, MESS_vector, MESS_vector)
- DEFBINOP (mm_mv_mul, MESS_matrix, MESS_vector)
- DEFBINOP (mm_mv_trans_mul, MESS_matrix, MESS_vector)
- DEFBINOP (mm_mv_herm_mul, MESS_matrix, MESS_vector)
- DEFBINOP (mm_mv_ldiv, MESS_matrix, MESS_vector)
- DEFBINOP (mm_mv_trans_ldiv, MESS_matrix, MESS_vector)
- DEFBINOP (mm_mv_herm_ldiv, MESS_matrix, MESS_vector)
- DEFBINOP (om_mv_mul, octave_matrix, MESS_vector)

- DEFBINOP (om_mv_trans_mul, octave_matrix, MESS_vector)
- DEFBINOP (om_mv_herm_mul, octave_matrix, MESS_vector)
- DEFBINOP (om_mv_ldiv, octave_matrix, MESS_vector)
- DEFBINOP (om_mv_trans_ldiv, octave_matrix, MESS_vector)
- DEFBINOP (om_mv_herm_ldiv, octave_matrix, MESS_vector)
- DEFBINOP (ocm_mv_mul, octave_complex_matrix, MESS_vector)
- DEFBINOP (ocm_mv_trans_mul, octave_complex_matrix, MESS_vector)
- DEFBINOP (ocm_mv_herm_mul, octave_complex_matrix, MESS_vector)
- DEFBINOP (ocm_mv_ldiv, octave_complex_matrix, MESS_vector)
- DEFBINOP (ocm_mv_trans_ldiv, octave_complex_matrix, MESS_vector)
- DEFBINOP (ocm_mv_herm_ldiv, octave_complex_matrix, MESS_vector)
- DEFBINOP (osm_mv_mul, octave_sparse_matrix, MESS_vector)
- DEFBINOP (osm_mv_trans_mul, octave_sparse_matrix, MESS_vector)
- DEFBINOP (osm_mv_herm_mul, octave_sparse_matrix, MESS_vector)
- DEFBINOP (osm_mv_ldiv, octave_sparse_matrix, MESS_vector)
- DEFBINOP (osm_mv_trans_ldiv, octave_sparse_matrix, MESS_vector)
- DEFBINOP (osm_mv_herm_ldiv, octave_sparse_matrix, MESS_vector)
- DEFBINOP (oscm_mv_mul, octave_sparse_complex_matrix, MESS_vector)
- DEFBINOP (oscm_mv_trans_mul, octave_sparse_complex_matrix, MESS_vector)
- DEFBINOP (oscm_mv_herm_mul, octave_sparse_complex_matrix, MESS_vector)
- DEFBINOP (oscm_mv_ldiv, octave_sparse_complex_matrix, MESS_vector)
- DEFBINOP (oscm_mv_trans_ldiv, octave_sparse_complex_matrix, MESS_vector)
- DEFBINOP (oscm_mv_herm_ldiv, octave_sparse_complex_matrix, MESS_vector)

10.516.1 Macro Definition Documentation

10.516.1.1 #define OCTMESS_GENERATOR(*SHORTTYPEA, TYPEA, SHORTTYPEB, TYPEB*)

Value:

```
DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _mul,          TYPEA, TYPEB){return
      op_mul(        dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _trans_mul,   TYPEA, TYPEB){return
      op_trans_mul(  dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _herm_mul,    TYPEA, TYPEB){return
      op_herm_mul(   dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _ldiv,        TYPEA, TYPEB){return
      op_ldiv(       dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _trans_ldiv,  TYPEA, TYPEB){return
      op_trans_ldiv( dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}
      \
    DEFBINOP(SHORTTYPEA## _ ##SHORTTYPEB## _herm_ldiv,   TYPEA, TYPEB){return
      op_herm_ldiv(  dynamic_cast<const TYPEA&> (a1), dynamic_cast<const TYPEB&> (a2));}    \
```

10.516.1.2 #define OCTMESS_GENERATOR(*SHORTTYPEA, TYPEA, SHORTTYPEB, TYPEB*)

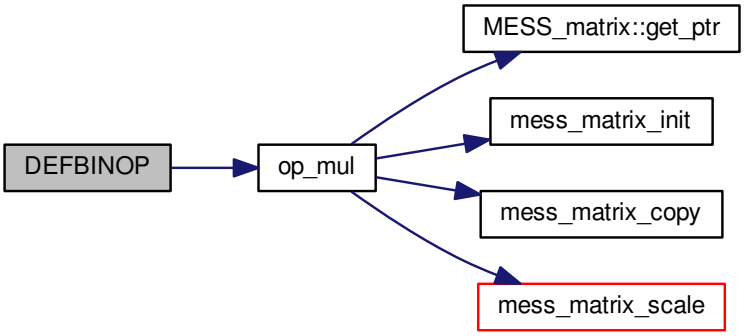
Value:

```
INSTALL_BINOP(op_mul,           TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _mul);          \
    INSTALL_BINOP(op_trans_mul,     TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _trans_mul);
      \
    INSTALL_BINOP(op_herm_mul,      TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _herm_mul);
      \
    INSTALL_BINOP(op_ldiv,          TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _ldiv);         \
    INSTALL_BINOP(op_trans_ldiv,    TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _trans_ldiv);
      \
    INSTALL_BINOP(op_herm_ldiv,     TYPEA, TYPEB,  SHORTTYPEA## _ ##SHORTTYPEB## _herm_ldiv);
      \
```

10.516.2 Function Documentation

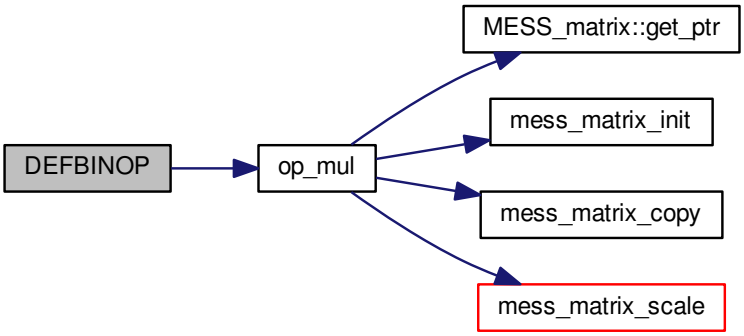
10.516.2.1 DEFBINOP (os_mv_mul , octave_scalar , MESS_matrix)

Here is the call graph for this function:



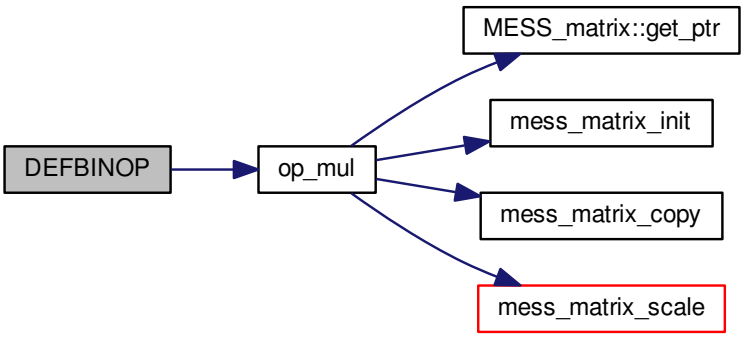
10.516.2.2 DEFBINOP (mv_os_mul , MESS_matrix , octave_scalar)

Here is the call graph for this function:



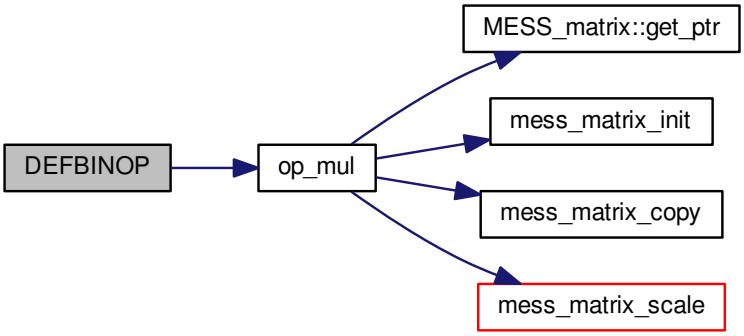
10.516.2.3 DEFBINOP (osc_mv_mul , octave_complex_scalar , MESS_matrix)

Here is the call graph for this function:



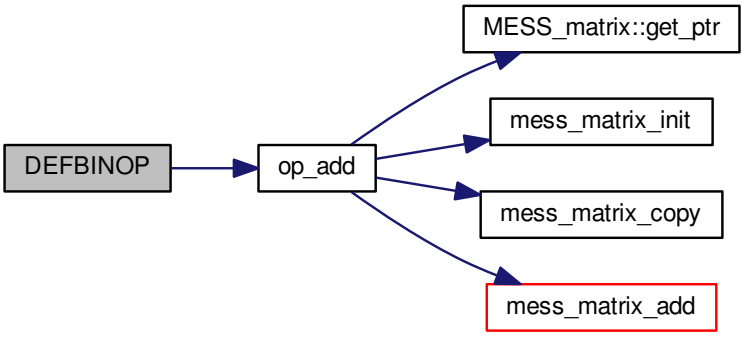
10.516.2.4 DEFBINOP (mv_osc_mul , MESS_matrix , octave_complex_scalar)

Here is the call graph for this function:



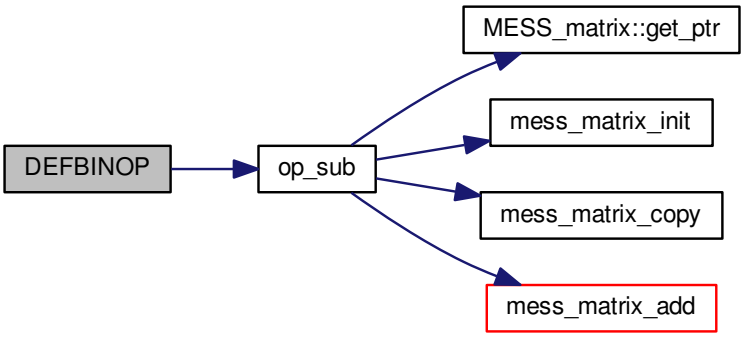
10.516.2.5 DEFBINOP (mv_mv_add , MESS_vector , MESS_vector)

Here is the call graph for this function:



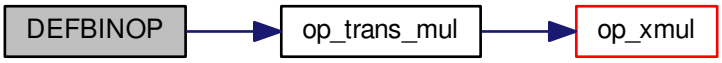
10.516.2.6 DEFBINOP (mv_mv_sub , MESS_vector , MESS_vector)

Here is the call graph for this function:



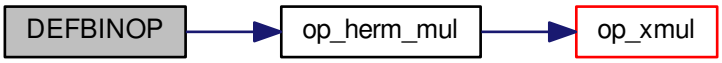
10.516.2.7 DEFBINOP (mv_mv_trans_mul , MESS_vector , MESS_vector)

Here is the call graph for this function:



10.516.2.8 DEFBINOP (mv_mv_herm_mul , MESS_vector , MESS_vector)

Here is the call graph for this function:



10.516.2.9 DEFBINOP (mm_mv_trans_ldiv , MESS_matrix , MESS_vector)

10.516.2.10 DEFBINOP (mm_mv_herm_ldiv , MESS_matrix , MESS_vector)

10.516.2.11 DEFBINOP (mm_mv_mul , MESS_matrix , MESS_vector)

10.516.2.12 DEFBINOP (mm_mv_trans_mul , MESS_matrix , MESS_vector)

10.516.2.13 DEFBINOP (mm_mv_herm_mul , MESS_matrix , MESS_vector)

10.516.2.14 DEFBINOP (mm_mv_ldiv , MESS_matrix , MESS_vector)

10.516.2.15 DEFBINOP (om_mv_mul , octave_matrix , MESS_vector)

10.516.2.16 DEFBINOP (om_mv_trans_mul , octave_matrix , MESS_vector)

10.516.2.17 DEFBINOP (om_mv_herm_mul , octave_matrix , MESS_vector)

10.516.2.18 DEFBINOP (om_mv_ldiv , octave_matrix , MESS_vector)

10.516.2.19 DEFBINOP (om_mv_trans_ldiv , octave_matrix , MESS_vector)

10.516.2.20 DEFBINOP (om_mv_herm_ldiv , octave_matrix , MESS_vector)

10.516.2.21 DEFBINOP (ocm_mv_ldiv , octave_complex_matrix , MESS_vector)

10.516.2.22 DEFBINOP (ocm_mv_mul , octave_complex_matrix , MESS_vector)

10.516.2.23 DEFBINOP (ocm_mv_trans_mul , octave_complex_matrix , MESS_vector)

10.516.2.24 DEFBINOP (ocm_mv_herm_mul , octave_complex_matrix , MESS_vector)

10.516.2.25 DEFBINOP (ocm_mv_trans_ldiv , octave_complex_matrix , MESS_vector)

10.516.2.26 DEFBINOP (ocm_mv_herm_ldiv , octave_complex_matrix , MESS_vector)

10.516.2.27 DEFBINOP (osm_mv_herm_ldiv , octave_sparse_matrix , MESS_vector)

10.516.2.28 DEFBINOP (osm_mv_herm_mul , octave_sparse_matrix , MESS_vector)

10.516.2.29 DEFBINOP (osm_mv_trans_ldiv , octave_sparse_matrix , MESS_vector)

10.516.2.30 DEFBINOP (osm_mv_ldiv , octave_sparse_matrix , MESS_vector)

10.516.2.31 DEFBINOP (osm_mv_mul , octave_sparse_matrix , MESS_vector)

10.516.2.32 DEFBINOP (osm_mv_trans_mul , octave_sparse_matrix , MESS_vector)

10.516.2.33 DEFBINOP (oscm_mv_trans_ldiv , octave_sparse_complex_matrix , MESS_vector)

10.516.2.34 DEFBINOP (oscm_mv_herm_mul , octave_sparse_complex_matrix , MESS_vector)

10.516.2.35 DEFBINOP (oscm_mv_herm_ldiv , octave_sparse_complex_matrix , MESS_vector)

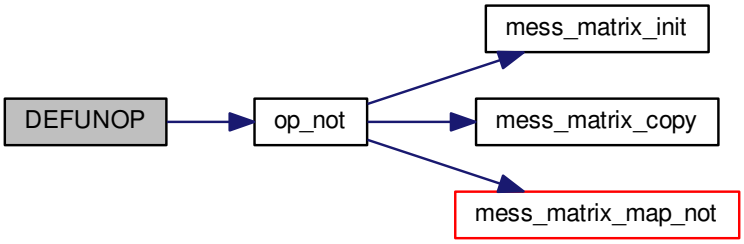
10.516.2.36 DEFBINOP (oscm_mv_ldiv , octave_sparse_complex_matrix , MESS_vector)

10.516.2.37 DEFBINOP (oscm_mv_mul , octave_sparse_complex_matrix , MESS_vector)

10.516.2.38 DEFBINOP (oscm_mv_trans_mul , octave_sparse_complex_matrix , MESS_vector)

10.516.2.39 DEFUNOP (mv_not , MESS_vector)

Here is the call graph for this function:

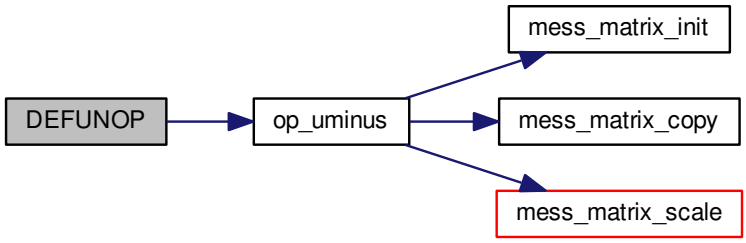


10.516.2.40 DEFUNOP (mv_uplus , MESS_vector)

Here is the call graph for this function:

10.516.2.41 DEFUNOP (mv_uminus , MESS_vector)

Here is the call graph for this function:



10.517 MESS_vector_ops_mm_mv.cc File Reference

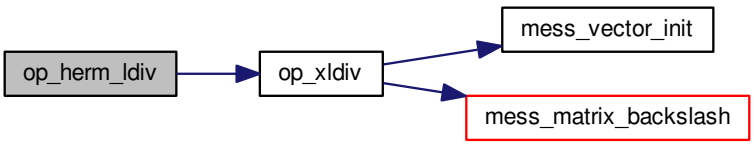
Functions

- octave_value op_xmul (const mess_operation_t opA, const MESS_matrix &m, const MESS_vector &v)
- octave_value op_mul (const MESS_matrix &m, const MESS_vector &v)
- octave_value op_trans_mul (const MESS_matrix &m, const MESS_vector &v)
- octave_value op_herm_mul (const MESS_matrix &m, const MESS_vector &v)
- octave_value op_xldiv (const MESS_matrix &m, const mess_operation_t opA, const MESS_vector &v)
- octave_value op_ldiv (const MESS_matrix &m, const MESS_vector &v)
- octave_value op_trans_ldiv (const MESS_matrix &m, const MESS_vector &v)
- octave_value op_herm_ldiv (const MESS_matrix &m, const MESS_vector &v)

10.517.1 Function Documentation

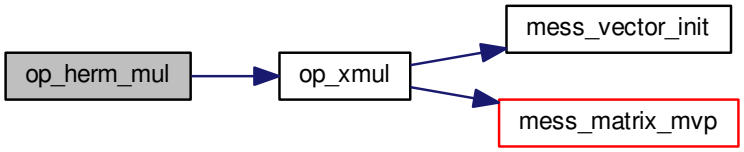
10.517.1.1 octave_value op_herm_ldiv (const MESS_matrix & *m*, const MESS_vector & *v*)

Here is the call graph for this function:



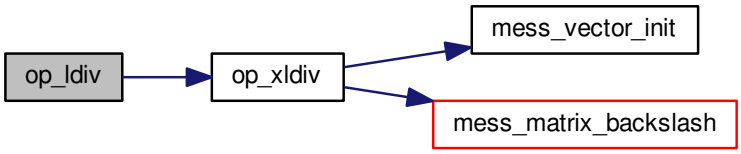
10.517.1.2 octave_value op_herm_mul (const MESS_matrix & *m*, const MESS_vector & *v*)

Here is the call graph for this function:



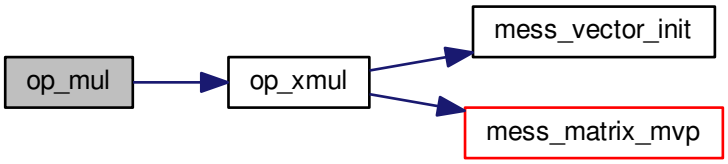
10.517.1.3 octave_value op_ldiv (const MESS_matrix & *m*, const MESS_vector & *v*)

Here is the call graph for this function:



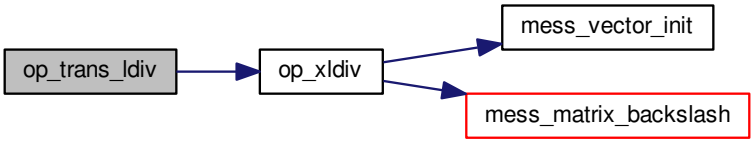
10.517.1.4 octave_value op_mul (const MESS_matrix & *m*, const MESS_vector & *v*)

Here is the call graph for this function:



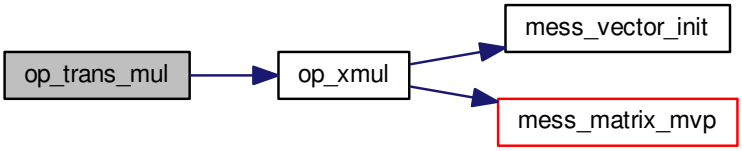
10.517.1.5 octave_value op_trans_ldiv (const MESS_matrix & *m*, const MESS_vector & *v*)

Here is the call graph for this function:



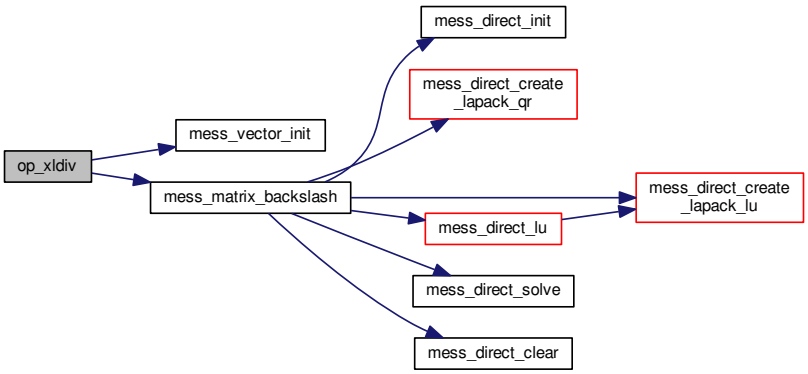
10.517.1.6 octave_value op_trans_mul (const MESS_matrix & *m*, const MESS_vector & *v*)

Here is the call graph for this function:



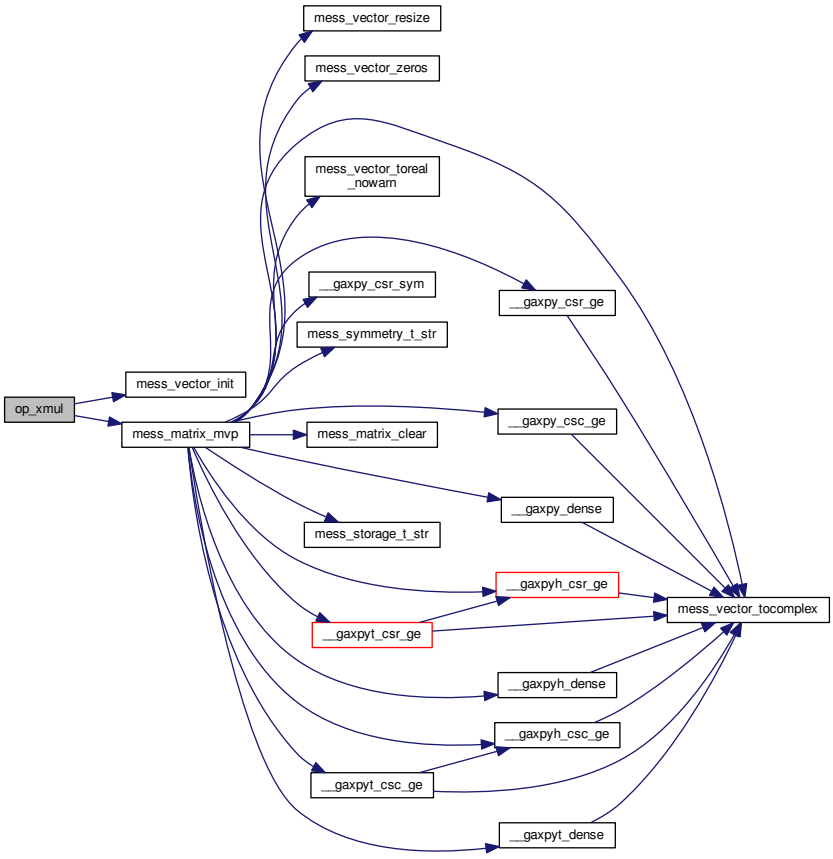
10.517.1.7 octave_value op_xldiv (const MESS_matrix & *m*, const mess_operation_t *opA*, const MESS_vector & *v*)

Here is the call graph for this function:



10.517.1.8 octave_value op_xmul (const mess_operation_t *opA*, const MESS_matrix & *m*, const MESS_vector & *v*)

Here is the call graph for this function:



10.518 MESS_vector_ops_mv_generator.h File Reference

Macros

- #define OCTMESS_GENERATOR(OCTAVE_MATRIX_TYPE, OCTAVE_MATRIX_TYPE_VALUE)

10.518.1 Macro Definition Documentation

10.518.1.1 #define OCTMESS_GENERATOR(*OCTAVE_MATRIX_TYPE*, *OCTAVE_MATRIX_TYPE_VALUE*)

Value:

```
\
    octave_value op_xmul (const mess_operation_t opA, const OCTAVE_MATRIX_TYPE &
      m,  const MESS_vector & v){                                  \
        int ret = 0;
                                        \
        mess_vector result=NULL;
                                                          \
        mess_matrix_st dummy = octmess_matrix_conv(m.
```

```
  OCTAVE_MATRIX_TYPE_VALUE());   /* dummy needs no clear */                                  \
    ret = mess_vector_init(&result);
  OCTMESS_ERROR(ret,(ret!=0),mess_vector_init);           \
    ret = mess_matrix_mvp(opA, &dummy, (mess_vector)v, result);
     OCTMESS_ERROR(ret,(ret!=0),mess_matrix_mvp);            \
    return new MESS_vector(result);
                                                  \
}
                                        \

                                        \

                                        \
octave_value op_mul (const OCTAVE_MATRIX_TYPE & m, const MESS_vector & v){return
  op_xmul(MESS_OP_NONE, m, v);}                           \
octave_value op_trans_mul (const OCTAVE_MATRIX_TYPE & m, const
  MESS_vector & v){return op_xmul(MESS_OP_TRANSPOSE, m, v);}
   \
octave_value op_herm_mul (const OCTAVE_MATRIX_TYPE & m, const
  MESS_vector & v){return op_xmul(MESS_OP_HERMITIAN, m, v);}
    \

                                        \

                                        \
octave_value op_xldiv (const OCTAVE_MATRIX_TYPE & m ,const
  mess_operation_t opA, const MESS_vector & v){                                  \
    int ret = 0;
                                        \
    mess_vector result=NULL;
                                                  \
    mess_matrix_st dummy = octmess_matrix_conv(m.
  OCTAVE_MATRIX_TYPE_VALUE());   /* dummy needs no clear */                                  \
    ret = mess_vector_init(&result);
  OCTMESS_ERROR(ret,(ret!=0),mess_vector_init);           \
    ret = mess_matrix_backslash(opA,&dummy,(mess_vector)v,result);
           OCTMESS_ERROR(ret,(ret!=0),mess_matrix_backslash);      \
    return new MESS_vector(result);
                                                  \
}
                                        \

                                        \

                                        \
octave_value op_ldiv(const OCTAVE_MATRIX_TYPE & m, const MESS_vector & v){return
  op_xldiv(m,MESS_OP_NONE,v);}                            \
octave_value op_trans_ldiv(const OCTAVE_MATRIX_TYPE & m, const
  MESS_vector & v){return op_xldiv(m,MESS_OP_TRANSPOSE,v);}
    \
octave_value op_herm_ldiv(const OCTAVE_MATRIX_TYPE & m, const
  MESS_vector & v){return op_xldiv(m,MESS_OP_HERMITIAN,v);}
     \
```

10.519 MESS_vector_ops_mv_mv.cc File Reference

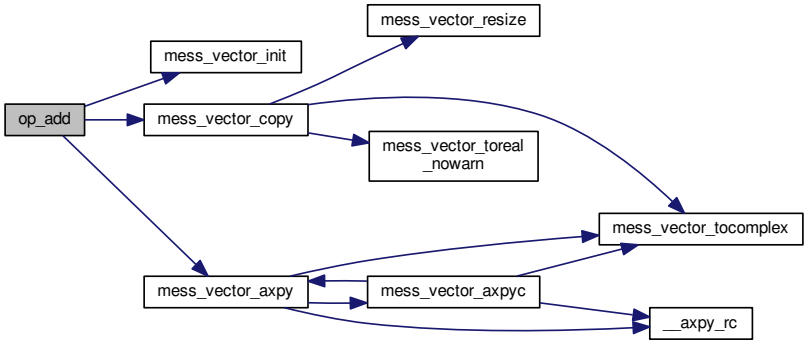
Functions

- octave_value op_not (const MESS_vector &v)
- octave_value op_uplus (const MESS_vector &v)
- octave_value op_uminus (const MESS_vector &v)
- octave_value op_add (const MESS_vector &v1, const MESS_vector &v2)
- octave_value op_sub (const MESS_vector &v1, const MESS_vector &v2)
- octave_value op_trans_mul (const MESS_vector &v1, const MESS_vector &v2)
- octave_value op_herm_mul (const MESS_vector &v1, const MESS_vector &v2)

10.519.1 Function Documentation

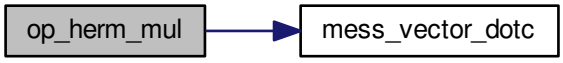
10.519.1.1 octave_value op_add (const MESS_vector & *v1*, const MESS_vector & *v2*)

Here is the call graph for this function:



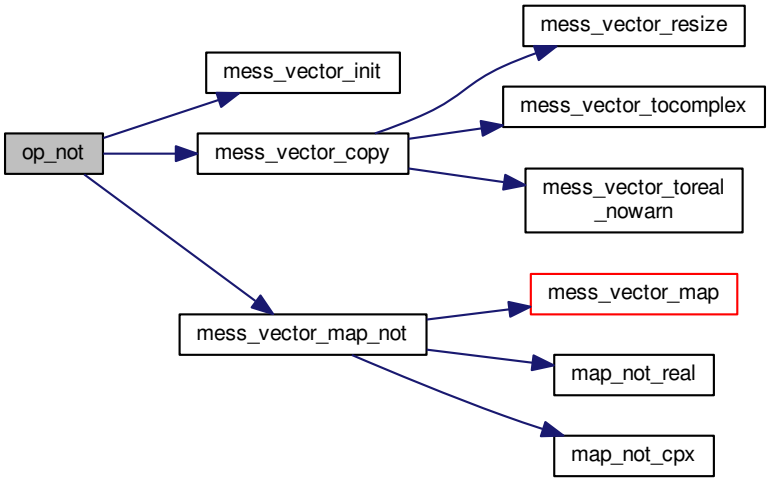
10.519.1.2 octave_value op_herm_mul (const MESS_vector & *v1*, const MESS_vector & *v2*)

Here is the call graph for this function:



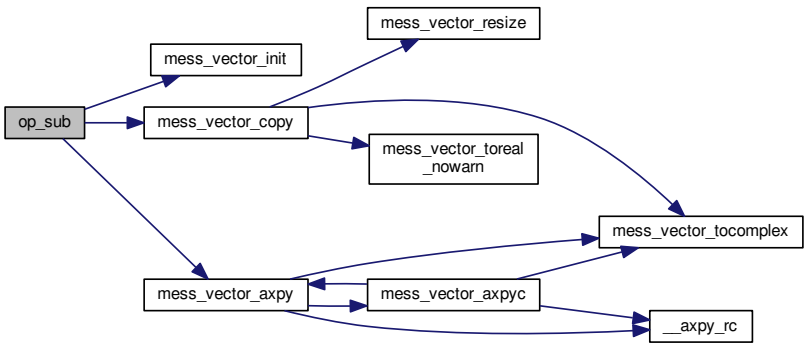
10.519.1.3 octave_value op_not (const MESS_vector & *v*)

Here is the call graph for this function:



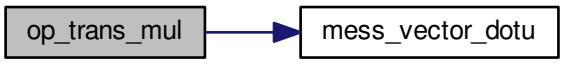
10.519.1.4 octave_value op_sub (const MESS_vector & *v1*, const MESS_vector & *v2*)

Here is the call graph for this function:



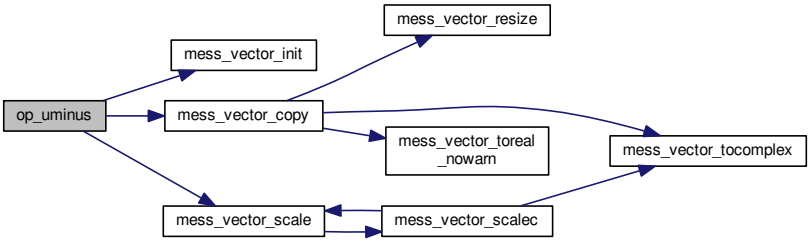
10.519.1.5 octave_value op_trans_mul (const MESS_vector & *v1*, const MESS_vector & *v2*)

Here is the call graph for this function:



10.519.1.6 octave_value op_uminus (const MESS_vector & *v*)

Here is the call graph for this function:



10.519.1.7 octave_value op_uplus (const MESS_vector & *v*)

10.520 MESS_vector_ops_mv_os.cc File Reference

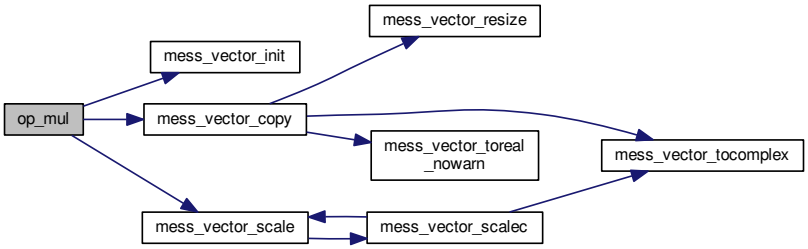
Functions

- octave_value op_mul (const octave_scalar &s, const MESS_vector &v)
- octave_value op_mul (const MESS_vector &m, const octave_scalar &s)

10.520.1 Function Documentation

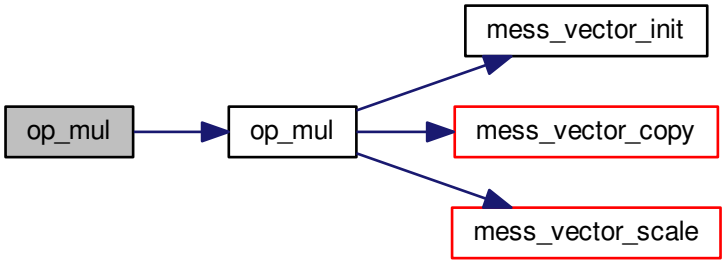
10.520.1.1 octave_value op_mul (const octave_scalar & *s*, const MESS_vector & *v*)

Here is the call graph for this function:



10.520.1.2 octave_value op_mul (const MESS_vector & *m*, const octave_scalar & *s*)

Here is the call graph for this function:



10.521 MESS_vector_ops_mv_osc.cc File Reference

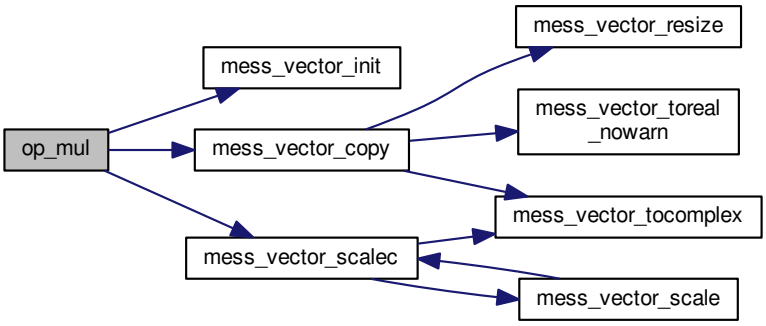
Functions

- octave_value op_mul (const octave_complex_scalar &s, const MESS_vector &v)
- octave_value op_mul (const MESS_vector &m, const octave_complex_scalar &s)

10.521.1 Function Documentation

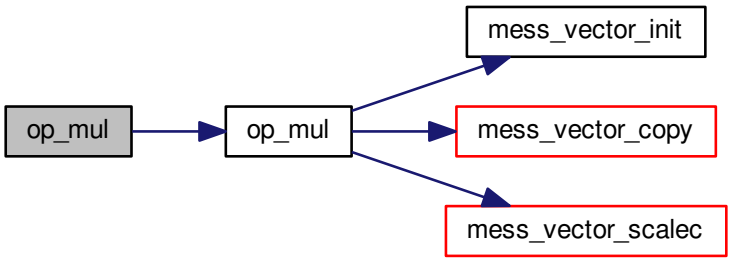
10.521.1.1 octave_value op_mul (const octave_complex_scalar & *s*, const MESS_vector & *v*)

Here is the call graph for this function:



10.521.1.2 octave_value op_mul (const MESS_vector & *m*, const octave_complex_scalar & *s*)

Here is the call graph for this function:



10.522 MESS_vector_ops_ocm_mv.cc File Reference

Functions

- OCTMESS_GENERATOR (octave_complex_matrix, complex_matrix_value)

10.522.1 Function Documentation

10.522.1.1 OCTMESS_GENERATOR (octave_complex_matrix , complex_matrix_value)

10.523 MESS_vector_ops_om_mv.cc File Reference

Functions

- OCTMESS_GENERATOR (octave_matrix, matrix_value)

10.523.1 Function Documentation

10.523.1.1 OCTMESS_GENERATOR (octave_matrix , matrix_value)

10.524 MESS_vector_ops_oscm_mv.cc File Reference

Functions

- OCTMESS_GENERATOR (octave_sparse_complex_matrix, sparse_complex_matrix_value)

10.524.1 Function Documentation

10.524.1.1 OCTMESS_GENERATOR (octave_sparse_complex_matrix , sparse_complex_matrix_value)

10.525 MESS_vector_ops_osm_mv.cc File Reference

Functions

- OCTMESS_GENERATOR (octave_sparse_matrix, sparse_matrix_value)

10.525.1 Function Documentation

10.525.1.1 OCTMESS_GENERATOR (octave_sparse_matrix , sparse_matrix_value)

10.526 mess_version.c File Reference

Macros

- #define PYMESS_MESS_HAVE_GENERATOR(NAME)

Functions

- PyObject ∗ pymess_mess_version (PyObject ∗self, PyObject ∗args)

  Call the mess_version function. The pymess_mess_version is callable as `mess_version` and calls internally the mess_version function.
- PyObject ∗ pymess_mess_version_verbose (PyObject ∗self, PyObject ∗args)

  Call the mess_version_verbose function. The pymess_mess_version_verbose is callable as `mess_version_verbose` and calls internally the mess_version_verbose function.
- PyObject ∗ pymess_mess_version_major (PyObject ∗self, PyObject ∗args)

  Call the mess_version_major function and returns the major version number.
- PyObject ∗ pymess_mess_version_minor (PyObject ∗self, PyObject ∗args)

  Call the mess_version_minor function and returns the minor version number.
- PyObject ∗ pymess_mess_version_patch (PyObject ∗self, PyObject ∗args)

  Call the mess_version_patch function and returns the patch version number.
- PyObject ∗ pymess_mess_git_id (PyObject ∗self, PyObject ∗args)

  Call the mess_git_id function and returns git id, which was used to configure the C-M.E.S.S. build.
- PyObject ∗ pymess_mess_git_branch (PyObject ∗self, PyObject ∗args)

  Call the mess_git_branch function and returns git branch, which was used to configure the C-M.E.S.S. build.

- PyObject * pymess_mess_is_debug (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_zlib (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_bzip2 (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_umfpack (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_amd (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_colamd (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_cholmod (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_csparse (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_superlu (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_mklpardiso (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_arpack (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_matio (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_openmp (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_mess64 (PyObject *self, PyObject *args)

10.526.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.526.2 Macro Definition Documentation

10.526.2.1 #define PYMESS_MESS_HAVE_GENERATOR(*NAME*)

Value:

```
PyObject* pymess_## NAME (PyObject *self, PyObject* args){          \
        if( NAME ()){Py_RETURN_TRUE;}                                \
        Py_RETURN_FALSE;                                             \
    }
```

10.527 mex_mess_care.c File Reference

Functions

- void mex_mess_care (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.527.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.528 mex_mess_dense_nm_gmpare.c File Reference

Functions

- void mex_mess_dense_nm_gmpare (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.528.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Robert Dykstra (dykstra@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.529 mex_mess_glyap.c File Reference

Functions

- void mex_mess_glyap (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.529.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.530 mex_mess_gstein.c File Reference

Functions

- void mex_mess_gstein (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.530.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.531 mex_mess_lradi.c File Reference

Functions

- void mex_mess_lradi (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.531.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.532 mex_mess_lrnm.c File Reference

Functions

- void mex_mess_lrnm (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.532.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.533 mex_mess_lyap.c File Reference

Functions

- void mex_mess_lyap (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.533.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.534 mex_mess_sylvester_dense.c File Reference

Functions

- void mex_mess_sylvester_dense (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.535 mex_mess_sylvester_sparsedense.c File Reference

Functions

- void mex_mess_sylvester_sparsedense (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.535.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.536 mex_test_matrix.c File Reference

Functions

- void mex_test_matrix (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.536.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.537 mex_test_options.c File Reference

Functions

- void mex_test_options (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.537.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.538 mex_test_status.c File Reference

Functions

- void mex_test_status (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.538.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.539 mex_test_vector.c File Reference

Functions

- void mex_test_vector (int nlhs, mxArray *plhs[ ], int nrhs, const mxArray *prhs[ ])

10.539.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.540 mgs.c File Reference

Functions

- int mess_matrix_mgs (mess_matrix A, mess_matrix Q, mess_matrix R)

  Perform a modified Gram-Schmidt orthogonalization.
- int mess_matrix_mgs_inplace (mess_matrix Q)

  Perform a modified Gram-Schmidt orthogonalization inplace on a matrix.

10.540.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.541 mgsadd.c File Reference

Functions

- int mess_matrix_mgs_add (mess_matrix Z, mess_matrix V, mess_matrix E)

  Use the modified Gram-Schmidt orthogonalization to add new columns to an orthogonal matrix.

10.541.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.542 minmax.c File Reference

Functions

- int mess_vector_max (mess_vector v, double *maxval, mess_int_t *maxind)

  Return the largest entry of a vector.
- int mess_vector_minvalue (mess_vector v, double *min)

  Get the minimal value in a vector.
- int mess_vector_maxvalue (mess_vector v, double *max)

  Get the maximum value in a vector.

10.542.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.543 misc.c File Reference

Functions

- void mess_versionx (int verbose)

  *Print **M.E.S.S.** version.*
- void mess_version ()

  *Print **M.E.S.S.** version.*
- void mess_version_verbose ()

  *Print **M.E.S.S.** version information.*
- const char * mess_git_id ()

  *Return the git id which was used to configure the **M.E.S.S.** build.*

- const char * mess_git_branch ()

  Return the name of the git branch which was used to configure the **M.E.S.S.** *build.*
- void mess_print_bytes (mess_int_t size)

  Print a memory size.
- double nearbyint (double x)

  Round a value.
- int mess_version_major (void)
- int mess_version_minor (void)
- int mess_version_patch (void)
- int mess_is_debug (void)
- int mess_have_zlib (void)
- int mess_have_bzip2 (void)
- int mess_have_umfpack (void)
- int mess_have_amd (void)
- int mess_have_colamd (void)
- int mess_have_cholmod (void)
- int mess_have_csparse (void)
- int mess_have_superlu (void)
- int mess_have_mklpardiso (void)
- int mess_have_arpack (void)
- int mess_have_matio (void)
- int mess_have_mess64 (void)
- int mess_have_openmp (void)

10.543.1 Function Documentation

10.543.1.1 void mess_versionx (int *verbose*)

Parameters

in	*verbose*	input if nonzero the output is more verbose.

The mess_versionx prints out the **M.E.S.S.** version and the mess_int_t data type. If `verbose` is nonzero the output is more verbose.

See also

mess_version
mess_version_verbose

10.544 misc.c File Reference

Macros

- #define MEXMESS_GENERATOR(CMESS_CALL, MXCREATECALL)
- #define MEXMESS_GENERATOR(CMESS_DIRECT_SELECT, CMESS_DIRECT_TYPE)

Functions

- void mex_mess_have_amd (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_have_arpack (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_have_bzip2 (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_have_cholmod (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_have_colamd (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_have_csparse (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_have_matio (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_have_mess64 (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_have_mklpardiso (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_have_openmp (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_have_superlu (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_have_umfpack (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_have_zlib (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_is_debug (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version_major (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version_minor (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version_patch (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_git_id (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_git_branch (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_version_verbose (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_direct_lu_select (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_direct_chol_select (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
- void mex_mess_multidirect_select (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])

10.544.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.544.2 Macro Definition Documentation

10.544.2.1 #define MEXMESS_GENERATOR(*CMESS_CALL*, *MXCREATECALL*)

Value:

```
void mex_##CMESS_CALL (int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]){
      \
        init_mexmess();
                \
        *plhs = MXCREATECALL( CMESS_CALL ());                                          \
    }
```

10.544.2.2 #define MEXMESS_GENERATOR(*CMESS_DIRECT_SELECT*, *CMESS_DIRECT_TYPE*)

Value:

```
void mex_##CMESS_DIRECT_SELECT(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]){
                        \
        init_mexmess();
                                        \

                        \
        int ret = 0;
                        \
        const int nargs = 1;
                        \

                        \
        /* check input/output arguments */
                        \
        if (nrhs != nargs ) {
                        \
            csc_error_message("The function needs %d input argument but %d are given.\n", (int) nargs ,(int
      ) nrhs);          \
            return;
                        \
        }
                        \

                        \
        /* check if class is correct */
                        \
        if (!mxIsClass(prhs[0], #CMESS_DIRECT_TYPE)){
                        \
            csc_error_message("Expected "#CMESS_DIRECT_TYPE" class.\n");
                        \
            return ;
                        \
        }
                        \

                        \
        /* call conversion function from MEX-M.E.S.S. to C-M.E.S.S. */
                        \
        CMESS_DIRECT_TYPE c_direct = CMESS_DIRECT_TYPE##_from_mexmess(prhs[0]);
                        \

                        \
        /* call direct select function */
                        \
        ret = CMESS_DIRECT_SELECT ( c_direct );
                        \
        if(ret){
                        \
            csc_error_message("C-M.E.S.S. returned an error %s.\n",
      mess_get_error(ret));                                  \
            return;
                        \
        }
                        \
    return;
                        \
    }
```

10.545 misc.h File Reference

Data Structures

- struct mess_timer_st

  Representation of a timer.

Macros

- #define isfinite(x) (sizeof(x) == sizeof(float))?__finitef(x):(sizeof(x)==sizeof(double))?__finite(x):__finitel(x)
- #define MESS_MAX(a, b) (((a)<(b)) ? (b):(a))

  Macro for Maximium of two numbers. The MESS_MAX macro return the maximum of two numbers.
- #define MESS_MIN(a, b) (((a)<(b)) ? (a):(b))

  Macro for Minimum of two numbers. The MESS_MIN macro return the minimum of two numbers.
- #define MESS_INIT_MATRICES(...)

  Macro to init a list of matrices.
- #define MESS_RESET_MATRICES(...)

  Macro to reset a list of matrices.
- #define MESS_CLEAR_MATRICES(...)

  Macro to clear a list of matrices.
- #define MESS_INIT_VECTORS(...)

  Macro to init a list of mess_vector instances.
- #define MESS_CLEAR_VECTORS(...)

  Macro to clear a list of vectors.
- #define MESS_INIT_DIRECTS(...)

  Macro to init a list of direct solvers.
- #define MESS_CLEAR_DIRECTS(...)

  Macro to clear a list of direct solvers.
- #define MESS_INIT_EQUATIONS(...)

  Macro to init a list of mess_equation instances.
- #define MESS_CLEAR_EQUATIONS(...)

  Macro to clear a list of mess_equation instances.
- #define MESS_CLEAR_POINTERS(...)

  Macro to clear a list of pointers.
- #define mess_free(X)

  Macro to free memory.
- #define mess_try_alloc(mem, cast, X)

  Macro to allocate memory (malloc).
- #define mess_try_alloc2(mem, cast, X)

  Macro to allocate memory (malloc).
- #define mess_try_calloc(mem, cast, n, size)

  Macro to allocate memory (calloc).
- #define mess_try_calloc2(mem, cast, n, size)

  Macro to allocate memory (calloc).
- #define mess_try_realloc(mem, cast, X)

  Macro to allocate memory (realloc).
- #define mess_try_realloc2(mem, cast, X)

  Macro to allocate memory (realloc).
- #define mess_try_alloc_omp(mem, cast, X)

  Macro to allocate memory (malloc).
- #define MESS_CALL_IF_EXIST(func, args...)

  Macro to call a static function (if available) with arguments.
- #define mess_check_nullpointer(X) if ((X)==NULL) { MSG_ERROR("%s points to NULL\n", #X); return MESS_ERROR_NULLPOINTER; }

  Macro to check if an object points to `NULL`.
- #define mess_check_notnullpointer(X) if ((X)!=NULL) { MSG_ERROR("%s points not to NULL\n", #X); return MESS_ERROR_NOTNULLPOINTER; }

  Macro to check if an object does not point to `NULL`.

- #define mess_check_real(X) if (!MESS_IS_REAL((X))) { MSG_ERROR("%s have to be real, current = %s \n", #X, mess_datatype_t_str((X)->data_type)); return MESS_ERROR_DATATYPE; }

  Macro to check if a matrix or vector is real.
- #define mess_check_complex(X) if (!MESS_IS_COMPLEX((X))) { MSG_ERROR("%s have to be complex\n", #X); return MESS_ERROR_DATATYPE; }

  Macro to check if a matrix or vector is complex.
- #define mess_check_square(X) if ((X)->rows!=(X)->cols) { MSG_ERROR("%s have to be square, rows = %ld, cols = %ld\n", #X, (long) (X)->rows, (long) (X)->cols); return MESS_ERROR_DIMENSION;}

  Macro to check if a matrix is square.
- #define mess_check_csc(X) if(!MESS_IS_CSC((X))) { MSG_ERROR("%s have to be a CSC stored matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

  Macro to check if a matrix is stored in the CSC storage.
- #define mess_check_csr(X) if(!MESS_IS_CSR((X))) { MSG_ERROR("%s have to be a CSR stored matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

  Macro to check if a matrix is stored in the CSR storage.
- #define mess_check_dense(X) if(!MESS_IS_DENSE((X))) { MSG_ERROR("%s have to be a dense matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

  Macro to check if a matrix is stored in the DENSE storage.
- #define mess_check_sparse(X) if(!(MESS_IS_CSC((X)) || MESS_IS_CSR((X)) || MESS_IS_COORD((X)))) { MSG_ERROR("%s have to be a sparse matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

  Macro to check if a matrix is stored in the sparse storage format.
- #define mess_check_coord(X) if(!MESS_IS_COORD((X))) { MSG_ERROR("%s have to be a coordinate matrix\n", #X); return MESS_ERROR_STORAGETYPE;}

  Macro to check if a matrix is stored in the COORD storage.
- #define mess_check_nocoord(X) if(MESS_IS_COORD((X))) { MSG_ERROR("%s is a coordinate matrix. Coordinate matrices are not supported.\n", #X); return MESS_ERROR_STORAGETYPE;}

  Macro to check if a matrix is not stored in the CSC storage.
- #define mess_check_positive(X) if ((X) <= 0) { MSG_ERROR("%s have to be positive\n", #X); return MESS_ERROR_ARGUMENTS;}

  Macro to check if an object is positive.
- #define mess_check_nonnegative(X) if ((X) < 0) { MSG_ERROR("%s have to be nonnegative\n", #X); return MESS_ERROR_ARGUMENTS;}

  Macro to check if an object is nonnegative.
- #define mess_check_same_size(X, Y) if (((X)->rows != (Y)->rows) || ((X)->cols != (Y)->cols)) { MSG_ERROR("%s and %s must have the same size. (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT ", " MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->rows, Y->cols); return (MESS_ERROR_DIMENSION); }

  Macro to check if two matrices have the same size.
- #define mess_check_same_dim(X, Y) if (((X)->dim != (Y)->dim)) { MSG_ERROR("%s and %s must have the same dim. " MESS_PRINTF_INT " <-> " MESS_PRINTF_INT ")\n",#X,#Y, X->dim, Y->dim); return (MESS_ERROR_DIMENSION); }

  Macro to check if two structs have the same dim.
- #define mess_check_same_rows(X, Y) if (((X)->rows != (Y)->rows)) { MSG_ERROR("%s and %s must have the number of rows. (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT ", " MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->rows, Y->cols); return (MESS_ERROR_DIMENSION); }

  Macro to check if two matrices or vectors have the same number of rows.
- #define mess_check_same_cols(X, Y) if (((X)->cols != (Y)->cols)) { MSG_ERROR("%s and %s must have the number of columns. (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT ", " MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->rows, Y->cols); return (MESS_ERROR_DIMENSION); }

  Macro to check if two matrices or vectors have the same number of columns.

- #define mess_check_same_colsrows(X, Y) if (((X)->cols != (Y)->rows)) { MSG_ERROR("Number of cols of %s must be the same as the number of rows of %s . (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT ", " MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->rows, Y->cols); return (MESS_ERROR_DIMENSION); }

  Macro to check if the number of rows of a matrix is identical to the number of columns of another matrix.
- #define mess_check_same_colsdim(X, Y) if (((X)->cols != (Y)->dim)) { MSG_ERROR("Number of cols of %s must be the same as the dimension of %s . (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->dim); return (MESS_ERROR_DIMENSION); }

  Macro to check if the number of columns of a matrix is identical to the dimension of a vector.
- #define mess_check_same_rowsdim(X, Y) if (((X)->rows != (Y)->dim)) { MSG_ERROR("Number of rows of %s must be the same as the dimension of %s . (" MESS_PRINTF_INT ", " MESS_PRINTF_INT ") <-> (" MESS_PRINTF_INT "))\n",#X,#Y,X->rows,X->cols, Y->dim); return (MESS_ERROR_DIMENSION); }

  Macro to check if the number of rows of a matrix is identical to the dimension of a vector.
- #define mess_check_same_datatype(X, Y) if (((X)->data_type != (Y)->data_type)) { MSG_ERROR("%s and %s must have the same data type.\n",#X,#Y); return(MESS_ERROR_DATATYPE); }

  Macro to check if two objects have the same datatype.
- #define mess_check_same_storetype(X, Y) if (((X)->store_type != (Y)->store_type)) { MSG_ERROR("%s and %s must have the same storage type.\n",#X,#Y); return (MESS_ERROR_STORAGETYPE); }

  Macro to check if two objects have the same storage type.
- #define mess_check_real_or_complex(X) if ((!MESS_IS_REAL(X) && ! MESS_IS_COMPLEX(X))) { MSG_ERROR("%s must be real or complex.\n",#X); return (MESS_ERROR_DATATYPE); }

  Macro to check if a matrix or a vector is real or complex.
- #define mess_check_operation_type(X)

  Macro to check if an operation on a matrix is correct.
- #define mess_check_storage_type(X)

  Macro to check if a given storage type is valid.
- #define mess_check_datatype(DT)

  Macro to check if an datatype is correct.
- #define mess_check_true(X) if ((X) == 0) { MSG_ERROR("Assertion %s == true failed.\n", #X); return MESS_ERROR_ARGUMENTS; }

  Check if a value is true.

Typedefs

- typedef struct mess_timer_st * mess_timer

  Type definition of a timer.

Functions

- int mess_init (void)

  *Initialize **M.E.S.S.**.*
- int mess_exit (void)

  *Clean up some initial **M.E.S.S.** structures.*
- double mess_eps (void)
- int mess_int_t_size (void)

  Return size of mess_int_t.
- void mess_version (void)

  *Print **M.E.S.S.** version.*
- void mess_version_verbose (void)

  *Print **M.E.S.S.** version information.*
- int mess_version_major (void)

- int mess_version_minor (void)
- int mess_version_patch (void)
- const char * mess_git_id (void)

  Return the git id which was used to configure the **M.E.S.S.** *build.*
- const char * mess_git_branch (void)

  Return the name of the git branch which was used to configure the **M.E.S.S.** *build.*
- void mess_openmp_info (void)

  Display information about OpenMP.
- int mess_is_debug (void)
- int mess_have_zlib (void)
- int mess_have_bzip2 (void)
- int mess_have_umfpack (void)
- int mess_have_amd (void)
- int mess_have_colamd (void)
- int mess_have_cholmod (void)
- int mess_have_csparse (void)
- int mess_have_superlu (void)
- int mess_have_mklpardiso (void)
- int mess_have_arpack (void)
- int mess_have_matio (void)
- int mess_have_openmp (void)
- int mess_have_mess64 (void)
- const char * mess_datatype_t_str (const mess_datatype_t data_type)

  Return a human readable name of a data type value.
- const char * mess_storage_t_str (const mess_storage_t store_type)

  Return a human readable name of a storage type value.
- const char * mess_symmetry_t_str (const mess_symmetry_t symmetry)

  Return a human readable name of a symmetry type value.
- const char * mess_operation_t_str (const mess_operation_t op)

  Return the human readable name of an operation type value.
- const char * mess_equation_t_str (const mess_equation_t eqn)

  Convert the mess_equation_t to a human readable string.
- const char * mess_parameter_t_str (const mess_parameter_t para)

  Convert the mess_parameter_t to a human readable string.
- const char * mess_memusage_t_str (const mess_memusage_t mem)

  Convert the mess_memusage_t to a human readable string.
- const char * mess_norm_t_str (const mess_norm_t nrm)

  Convert the mess_norm_t to a human readable string.
- const char * mess_sign_scale_t_str (const mess_sign_scale_t sc_type)

  Convert the mess_sign_scale_t to a human readable string.
- const char * mess_direct_lupackage_t_str (const mess_direct_lupackage_t lu_type)

  Convert the mess_direct_lupackage_t to a human readable string.
- const char * mess_direct_cholpackage_t_str (const mess_direct_cholpackage_t chol_type)

  Convert the mess_direct_cholpackage_t to a human readable string.
- const char * mess_multidirect_t_str (const mess_multidirect_t ml_type)

  Convert the mess_multidirect_t to a human readable string.
- const char * mess_residual_t_str (const mess_residual_t res_type)

  Convert the mess_residual_t to a human readable string.
- int mess_print_format_select (mess_print_format_t print_type)

  Change the default behaviour of printing.
- int mess_print_format_double (double d)

  Print a double value using MSG_PRINT.

- int mess_print_format_double_cpx (mess_double_cpx_t cpx)

  Print a mess_double_cpx_t value using MSG_PRINT.
- void mess_print_bytes (mess_int_t size)

  Print a memory size.
- void * __mess_malloc (size_t size)

  Wrapper around malloc.
- void * __mess_realloc (void *ptr, size_t newsize)

  Wrapper around realloc.
- void * __mess_calloc (size_t nmemb, size_t size)

  Wrapper around calloc.
- void __mess_free (void *ptr)

  Wrapper around free.
- void mess_set_malloc (void *malloc_ptr, void *reallloc_ptr, void *calloc_ptr, void *free_ptr)

  Set the malloc implementation to be used.
- double nearbyint (double x)

  Round a value.
- int mess_matrix_st_offset (int *size, int *offsets, char **fieldnames)

  Information about memory laout of mess_matrix_st.
- int mess_vector_st_offset (int *size, int *offsets, char **fieldnames)

  Information about memory laout of mess_vector_st.
- int mess_status_st_offset (int *size, int *offsets, char **fieldnames)

  Information about memory laout of mess_status_st.
- int mess_options_st_offset (int *size, int *offsets, char **fieldnames)

  Information about memory laout of mess_options_st.
- int mess_equation_st_offset (int *size, int *offsets, char **fieldnames)

  Information about memory laout of mess_equation_st.
- int mess_direct_st_offset (int *size, int *offsets, char **fieldnames)

  Information about memory laout of mess_direct_st.
- double mess_wtime (void)

  Return a timer value.
- double mess_ctime (void)

  Get the current CPU time.
- int mess_timer_init (mess_timer *timer)

  Initialize a timer object.
- int mess_timer_clear (mess_timer *timer)

  Clean up a mess_timer object.
- double mess_timer_get (mess_timer timer)

  Get current elasped time since starting the timer.
- double mess_timer_getandreset (mess_timer timer)

  Get elapsed time since starting the timer and reset.
- double mess_timer_diff (mess_timer timerStart, mess_timer timerEnd)

  Get difference between two timers.

10.545.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.546 mklpardiso.c File Reference

Data Structures

- struct mklpardiso_solver

Typedefs

- typedef struct mklpardiso_solver mklpardiso_solver

Functions

- static mess_int_t mklpardiso_error (mess_int_t error)
- static int mklpardiso_solve_vector (void ∗data, mess_int_t op, mess_vector b, mess_vector x)
- static int mklpardiso_solve_matrix (void ∗data, mess_int_t op, mess_matrix b, mess_matrix x)
- static int mklpardiso_solve (void ∗data, mess_vector b, mess_vector x)

  Solve $Ax = b$.
- static int mklpardiso_solvet (void ∗data, mess_vector b, mess_vector x)

  Solve $A^T x = b$.
- static int mklpardiso_solveh (void ∗data, mess_vector b, mess_vector x)

  Solve $A^H x = b$.
- static int mklpardiso_solvem (void ∗data, mess_matrix b, mess_matrix x)

  Solve $AX = B$ *(matrix version)*
- static int mklpardiso_solvemt (void ∗data, mess_matrix b, mess_matrix x)

  Solve $A^T x = b$.
- static int mklpardiso_solvemh (void ∗data, mess_matrix b, mess_matrix x)

  Solve $A^H x = b$.
- static int mklpardiso_clear (void ∗solver)

  Clear an mklpardiso_solver object.
- int mess_direct_create_mklpardiso_control (mess_matrix matrix, mess_direct direct, mess_int_t ∗iparm, mess_int_t mtype)

  Generate a direct linear solver for standard linear systems $Ax = b$ *with* `MKL-PARDISO`.
- int mess_direct_create_mklpardiso (mess_matrix matrix, mess_direct direct)

  Generate a direct linear solver for standard linear systems $Ax = b$ *with* `MKL-PARDISO`.

10.546.1 Detailed Description

Author

Maximilian Behr (`behr@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.546.2 Typedef Documentation

10.546.2.1 typedef struct mklpardiso_solver mklpardiso_solver

10.546.3 Function Documentation

10.546.3.1 int mess_direct_create_mklpardiso (mess_matrix *matrix,* mess_direct *direct*)

Parameters

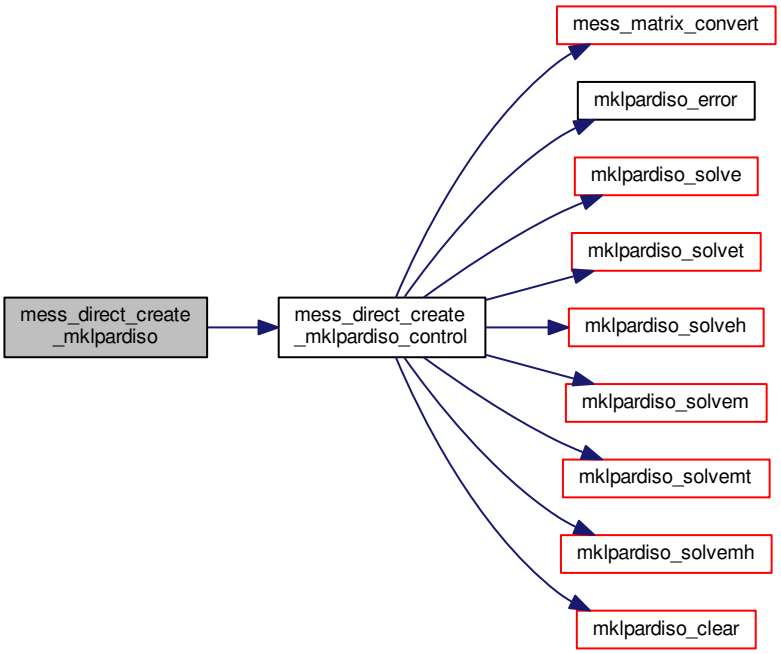
in	*matrix*	matrix to decompose
in,out	*direct*	direct solver

Returns

zero on success or a non zero error value otherwise

The mess_direct_create_mklpardiso function creates a mess_direct solver using MKL-PARDISO.

Here is the call graph for this function:



10.546.3.2 int mess_direct_create_mklpardiso_control (mess_matrix *matrix*, mess_direct *direct*, mess_int_t * *iparm*, mess_int_t *mtype*)

Parameters

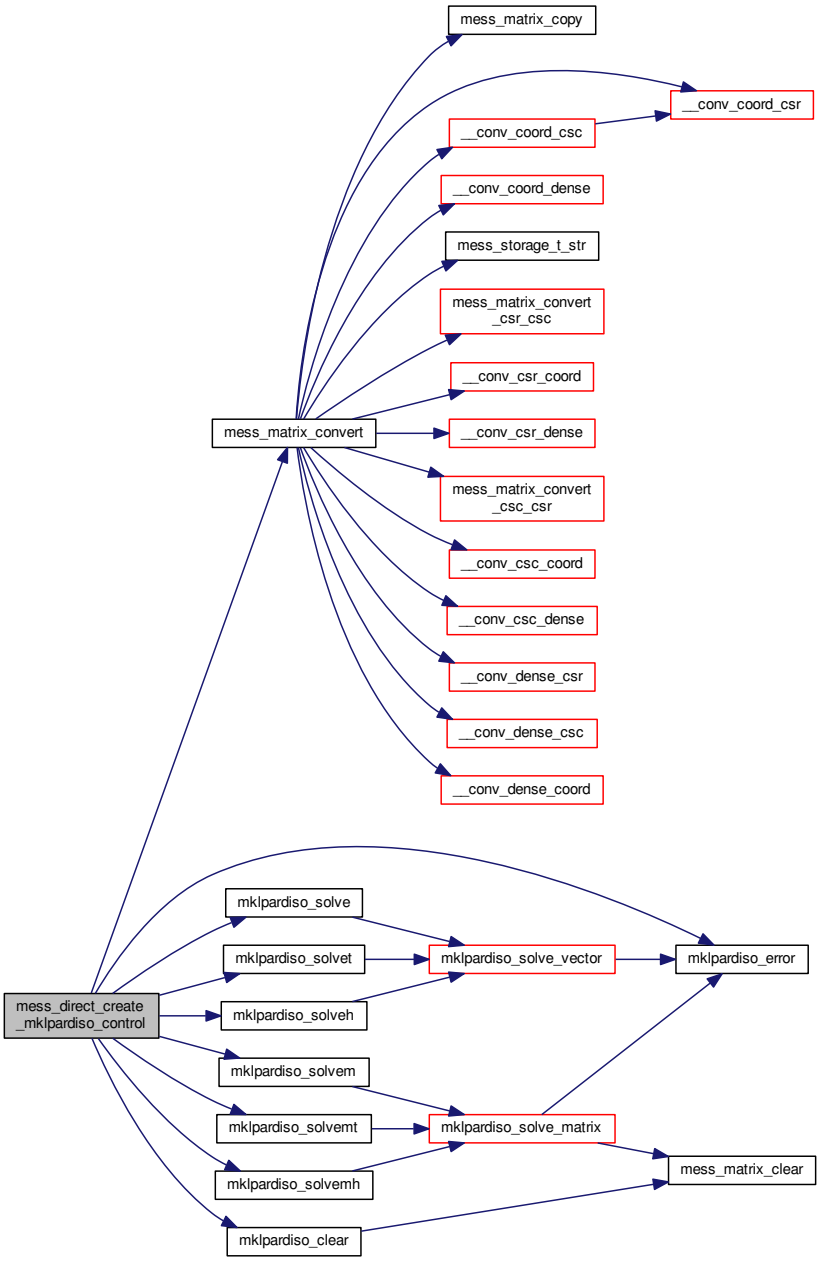
in	*matrix*	input matrix to decompose
in,out	*direct*	input/output direct solver
in	*iparm*	input array of solver parameters
in	*mtype*	input matrix type

Returns

zero on success or a non zero error value otherwise

The mess_direct_create_mklpardiso_control function creates a mess_direct solver using MKL-PARDISO with specified solver parameters in `iparm` and matrix in type specified with `mtype`.

Here is the call graph for this function:



10.546.3.3 static int mklpardiso_clear (void * *solver*) [static]

Parameters

in,out	*solver*	pointer to the data object

Returns

always zero

The mklpardiso_clear function clears an mklpardiso_solver object.

Here is the call graph for this function:

10.546.3.4 static mess_int_t mklpardiso_error (mess_int_t *error*) [static]

10.546.3.5 static int mklpardiso_solve (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

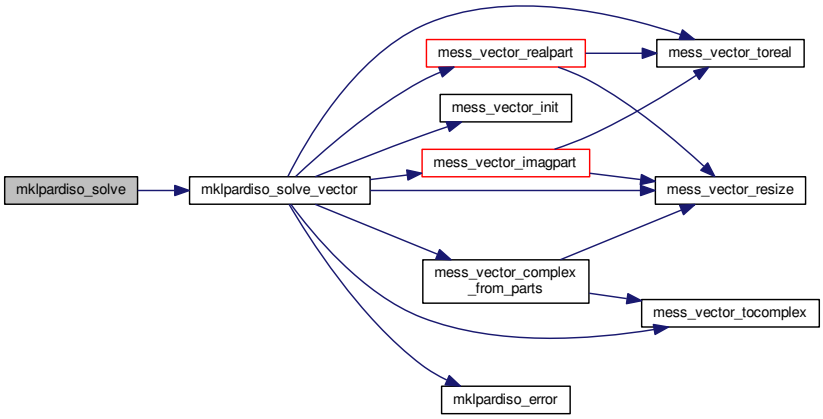
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

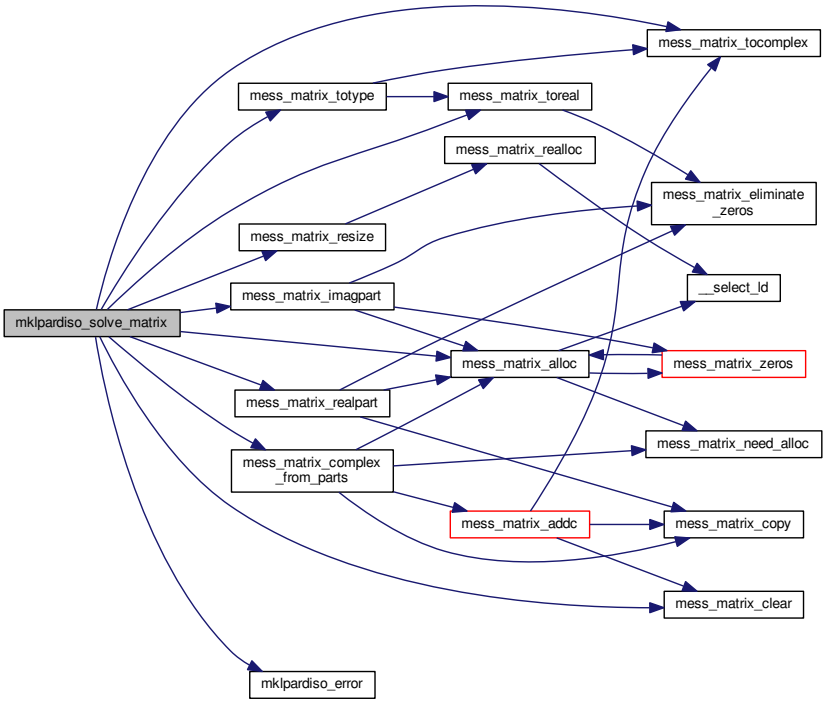
The mklpardiso_solve function solves $Ax = b$.

Here is the call graph for this function:



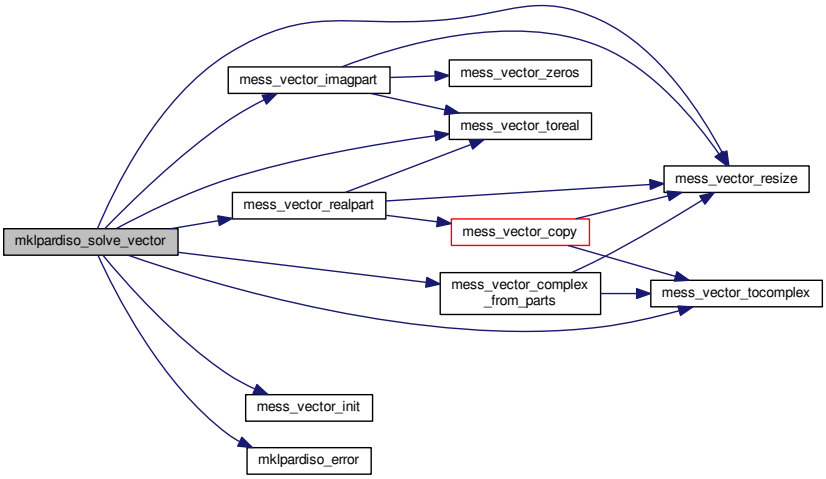
10.546.3.6 static int mklpardiso_solve_matrix (void * *data*, mess_int_t *op*, mess_matrix *b*, mess_matrix *x*) [static]

Here is the call graph for this function:



10.546.3.7 static int mklpardiso_solve_vector (void * *data*, mess_int_t *op*, mess_vector *b*, mess_vector *x*) [static]

Here is the call graph for this function:



10.546.3.8 static int mklpardiso_solveh (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

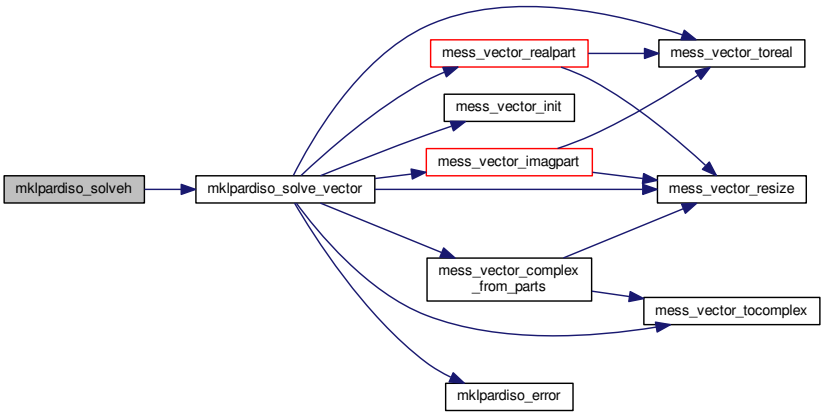
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The mklpardiso_solveh function solves $A^H x = b$.

Here is the call graph for this function:



10.546.3.9 static int mklpardiso_solvem (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

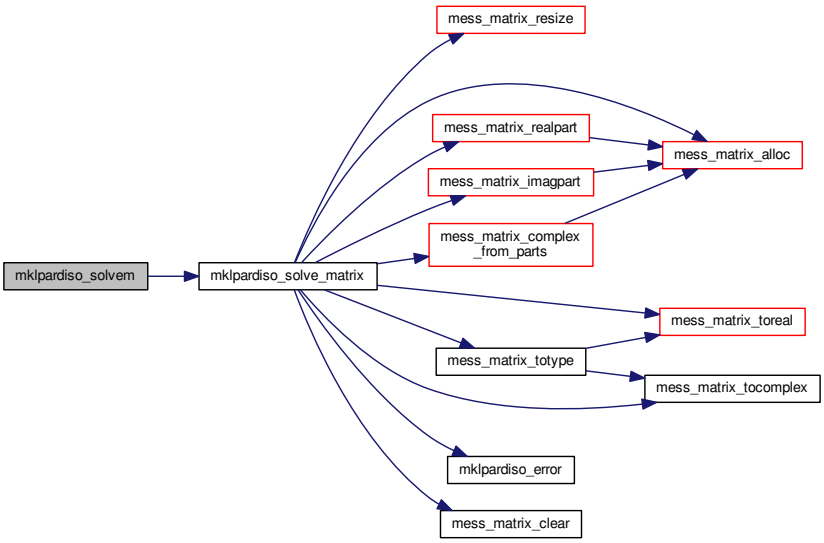
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The mklpardiso_solvem function solves $Ax = b$.

Here is the call graph for this function:



10.546.3.10 static int mklpardiso_solvemh (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

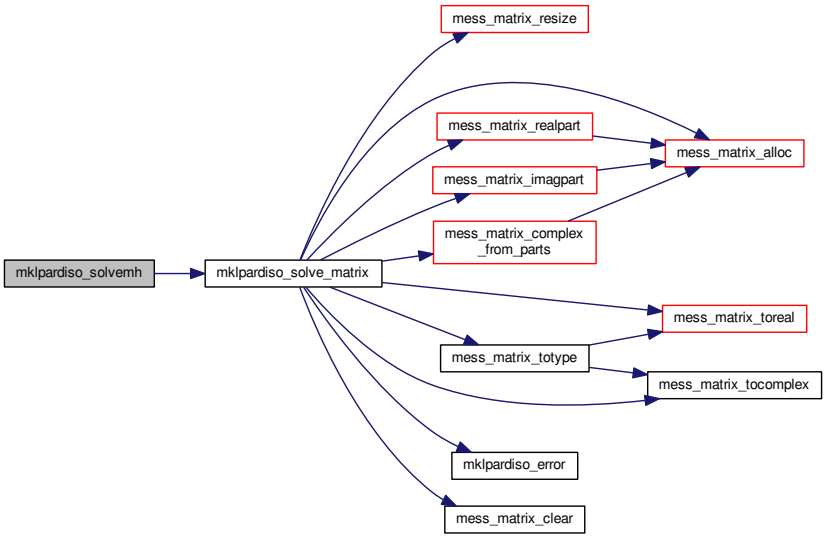
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The mklpardiso_solvemh function solves $A^H x = b$.

Here is the call graph for this function:



10.546.3.11 static int mklpardiso_solvemt (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

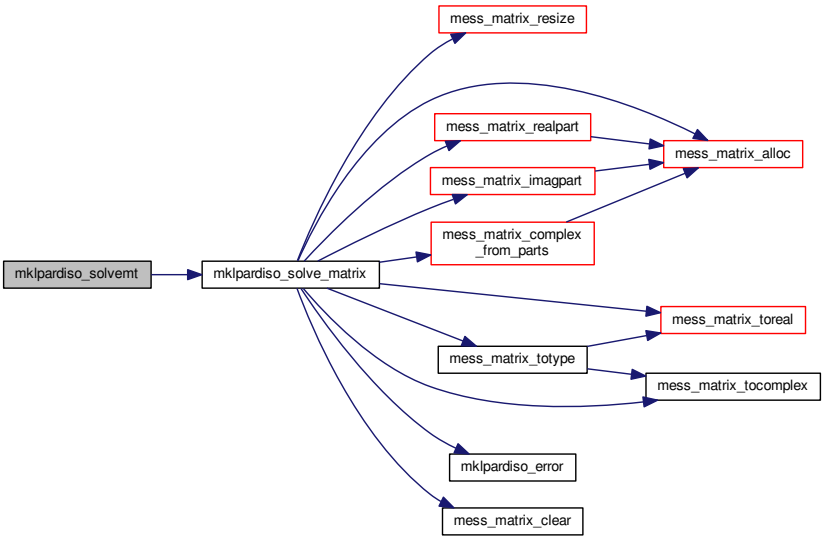
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The mklpardiso_solvemt function solves $A^T x = b$.

Here is the call graph for this function:



10.546.3.12 static int mklpardiso_solvet (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

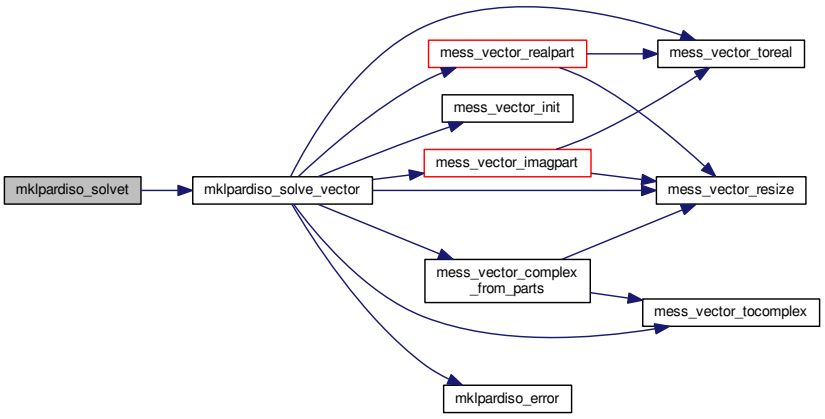
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The mklpardiso_solvet function solves $A^T x = b$.

Here is the call graph for this function:



10.547 mult.c File Reference

Functions

- void F77_GLOBAL (ddgemm_dense_sparsenn, DDGEMM_DENSE_SPARSENN)

10.547.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

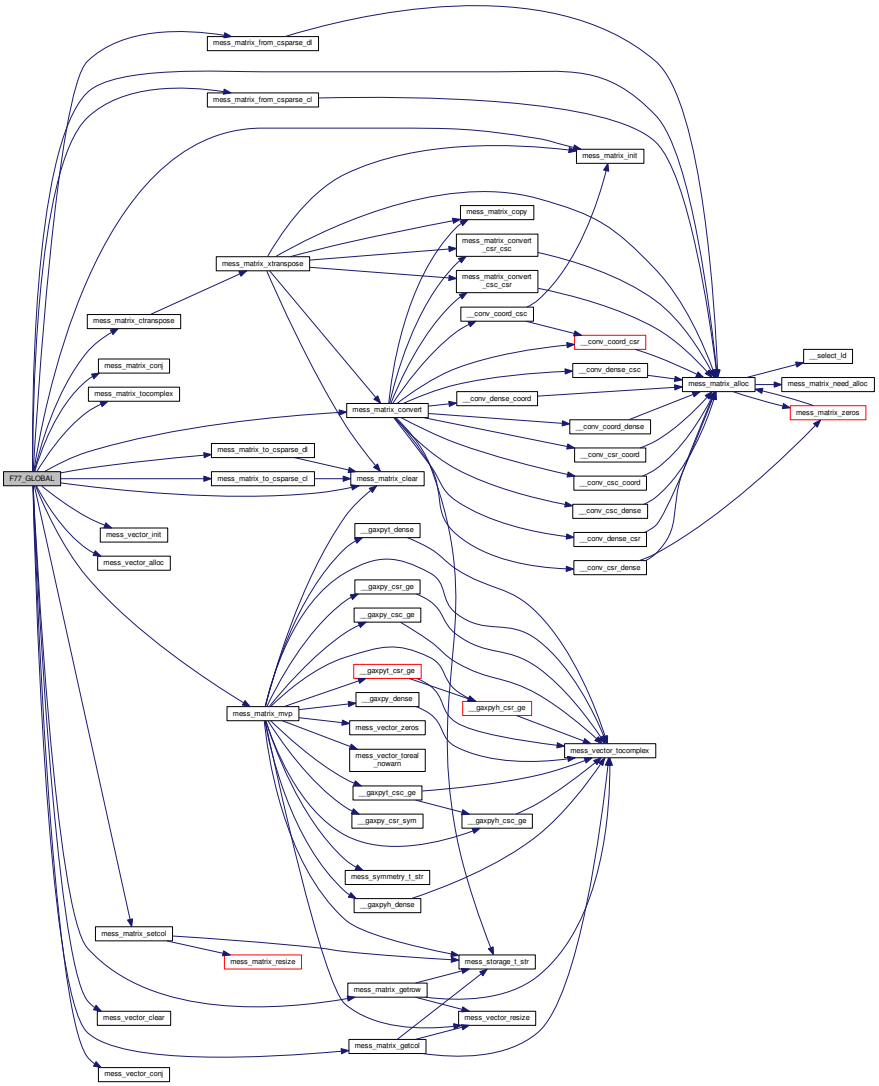
This file implements $C = op(A) * op(B)$ where $op(X)$ can be

Operation Type	op
MESS_OP_NONE	$op(A) = A$
MESS_OP_TRANSPOSE	$op(A) = A^T$
MESS_OP_HERMITIAN	$op(A) = A^H$

10.547.2 Function Documentation

10.547.2.1 void F77_GLOBAL (ddgemm_dense_sparsenn , DDGEMM_DENSE_SPARSENN)

Here is the call graph for this function:



10.548 multidirect.c File Reference

Macros

- #define DTINIT(X)
- #define DTCHECK(X)

Functions

- int mess_multidirect_init (mess_multidirect *direct)

  Initialize a mess_multidirect object.
- int mess_multidirect_clear (mess_multidirect *direct)

  Clear a mess_multidirect object.

- int mess_multidirect_solve (mess_operation_t op, mess_multidirect solver, mess_int_t ind, mess_vector b, mess_vector x)

  Solve $op(A(i))x = b$.
- int mess_multidirect_solvem (mess_operation_t op, mess_multidirect solver, mess_int_t ind, mess_matrix b, mess_matrix x)

  Solve $op(A(i))X = B$.
- int mess_multidirect_create (mess_matrix matrix, mess_vector shiftsl, mess_vector shiftsr, mess_multidirect mlu, mess_direct base, mess_matrix shiftmatrix)

  Create a multisolver for shifted linear systems $(sA + pI)$ *or* $(sA + pE)$.
- int mess_multidirect_select (mess_multidirect_t solver)

  Select the multisolver which is used by mess_multidirect_create.

Variables

- static mess_multidirect_t selected_solver = MESS_MULTIDIRECT_SPARSE_LU
- static pthread_mutex_t selected_solver_mutex = PTHREAD_MUTEX_INITIALIZER

10.548.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file implements the interface to the multidirect solvers.

10.548.2 Macro Definition Documentation

10.548.2.1 #define DTCHECK(X)

10.548.2.2 #define DTINIT(X)

10.548.3 Variable Documentation

10.548.3.1 mess_multidirect_t selected_solver = MESS_MULTIDIRECT_SPARSE_LU [static]

10.548.3.2 pthread_mutex_t selected_solver_mutex = PTHREAD_MUTEX_INITIALIZER [static]

10.549 multidirect_select.c File Reference

Functions

- PyObject * pymess_multidirect_select (PyObject *self, PyObject *args)

  Wrapper around mess_multidirect_select for use with Python.

10.549.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.550 multilu_adavanced.c File Reference

Data Structures

- struct multilu

  Internal structure to store a multilu.
- struct jobdata_st

Macros

- #define WORK_BACKGROUND 0
- #define DTINIT(X)
- #define DTCHECK(X)
- #define IS_REAL(X) (fabs(cimag((X))) $<$ 1.1$*$eps)
- #define IS_COMPLEX(X) (fabs(cimag((X))) $>$= 1.1$*$eps)

Typedefs

- typedef struct jobdata_st jobdata

Functions

- static size_t multilu_memsize (void $*$data)

  Get the size of the data element.
- static mess_int_t $*$ pinv (mess_int_t const $*$p, mess_int_t n)

  Invert a permutation.
- static int multilu_init (struct multilu $*$mlu)

  Initialize a "single-pattern-multi-value" LU decompositon.
- static int multilu_clear (void $*$data)

  Clean up a multilu.
- static int multilu_getL (void $*$data, mess_int_t ind, mess_matrix L)

  Get the L factor of a decomposition.
- static int multilu_getU (void $*$data, mess_int_t ind, mess_matrix U)

  Get the U factor of a decomposition.
- static int multilu_getp (void $*$data, mess_int_t ind, mess_int_t $*$p)

  Get the row permutation from the solver.
- static int multilu_getq (void $*$data, mess_int_t ind, mess_int_t $*$q)

  Get the column permutation from the solver.
- static int multilu_solve (void $*$data, mess_int_t n, mess_vector b, mess_vector x)

  Solve the system $A(n)x = b$.
- static int multilu_solvem (void $*$data, mess_int_t n, mess_matrix b, mess_matrix x)

  Solve the system $A(n)X = B$.
- static int multilu_solvet (void $*$data, mess_int_t n, mess_vector b, mess_vector x)

  Solve the system $A(n)^T x = b$.
- static int multilu_solveh (void $*$data, mess_int_t n, mess_vector b, mess_vector x)

  Solve the system $A(n)^H x = b$.
- static int multilu_solvemt (void $*$data, mess_int_t n, mess_matrix b, mess_matrix x)

  Solve the system $A(n)^T X = B$.
- static int multilu_solvemh (void $*$data, mess_int_t n, mess_matrix b, mess_matrix x)

  Solve the system $A(n)^H X = B$.
- static void job_decompose_real (void $*$data)
- static void job_decompose_complex (void $*$data)
- int mess_multidirect_create_sparse_lu (mess_matrix matrix, mess_vector shiftsl, mess_vector shiftsr, mess_multidirect mlu, mess_direct base_input, mess_matrix shiftmatrix_input)

  Create a single-pattern-multi-value LU decomposition of $(sA + pI)$ or $(sA + pE)$.

10.550.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.550.2 Macro Definition Documentation

10.550.2.1 #define DTCHECK(X)

10.550.2.2 #define DTINIT(X)

10.550.2.3 #define IS_COMPLEX(X) (fabs(cimag((X))) >= 1.1∗eps)

10.550.2.4 #define IS_REAL(X) (fabs(cimag((X))) < 1.1∗eps)

10.550.2.5 #define WORK_BACKGROUND 0

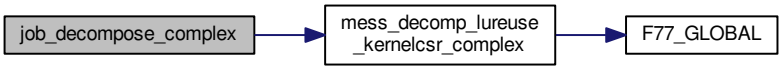
10.550.3 Typedef Documentation

10.550.3.1 typedef struct jobdata_st jobdata

10.550.4 Function Documentation

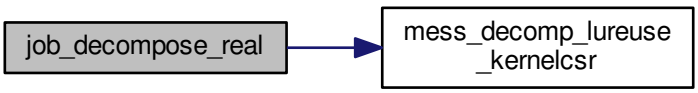
10.550.4.1 static void job_decompose_complex (void ∗ *data*) [static]

Here is the call graph for this function:



10.550.4.2 static void job_decompose_real (void ∗ *data*) [static]

Here is the call graph for this function:



10.550.4.3 static int multilu_clear (void ∗ *data*) [static]

Parameters

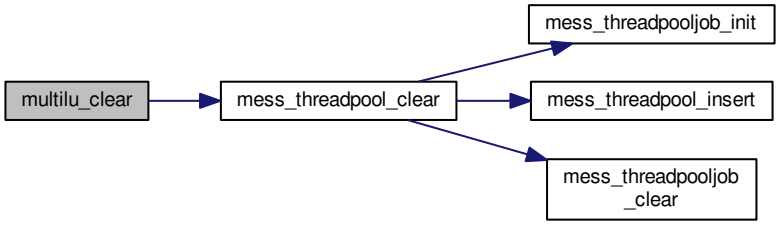
in	*data*	input pointer to the internal data structure

Returns

always zero

The multilu_clear function is a internal function of the multilu-solver to clean up when mess_multidirect_clear is called.

Here is the call graph for this function:



10.550.4.4 static int multilu_getL (void * *data*, mess_int_t *ind*, mess_matrix *L*) [static]

Parameters

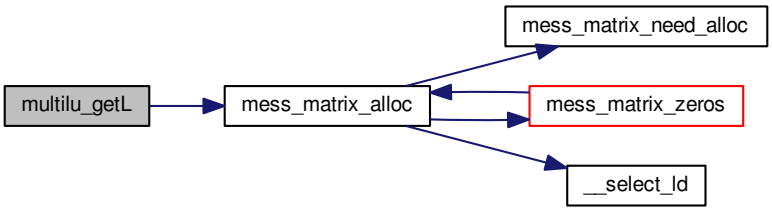
in	*data*	input pointer to the internal data structure
in	*ind*	input index to get
out	*L*	output L factor

Returns

zero on success or a non-zero error value otherwise

The multilu_getL function gets the L factor from a decomposition.

Here is the call graph for this function:



10.550.4.5 static int multilu_getp (void * *data*, mess_int_t *ind*, mess_int_t * *p*) [static]

Parameters

in	*data*	input pointer to internal data structure
in	*ind*	input index of the solver
out	*p*	output row permutation

Returns

zero on success or a non-zero error value otherwise

The multilu_getp function gets the row permutation of a solver. p must be preallocated.

10.550.4.6 static int multilu_getq (void * *data*, mess_int_t *ind*, mess_int_t * *q*) [static]

Parameters

in	*data*	input pointer to internal data structure
in	*ind*	input index of the solver
out	*q*	output column permutation

Returns

zero on success or a non-zero error value otherwise

The multilu_getq function gets the row permutation of a solver. q must be preallocated.

10.550.4.7 static int multilu_getU (void * *data*, mess_int_t *ind*, mess_matrix *U*) [static]

Parameters

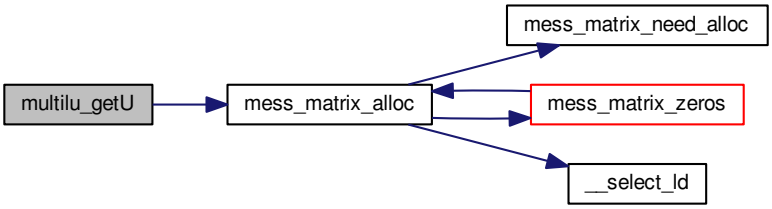
in	*data*	input pointer to the internal data structure
in	*ind*	input index to get
out	*U*	output U factor

Returns

zero on success or a non-zero error value otherwise

The multilu_getU function gets the U factor from a decomposition.

Here is the call graph for this function:



10.550.4.8 static int multilu_init (struct multilu * *mlu*) [static]

Parameters

out	*mlu*	the spmv to initialize

Returns

always zero

The multilu_init function initializes a single-patter-multi-value LU decompositon.

10.550.4.9 static size_t multilu_memsize (void * *data*) [static]

Parameters

in	*data*	input pointer to the data

Returns

size of data, otherwise zero

The multilu_memsize function gets the size of data.

10.550.4.10 static int multilu_solve (void * *data*, mess_int_t *n*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input pointer to a multilu
in	*n*	input index of the decomposition
in	*b*	input right hand side
out	*x*	solution vector

Returns

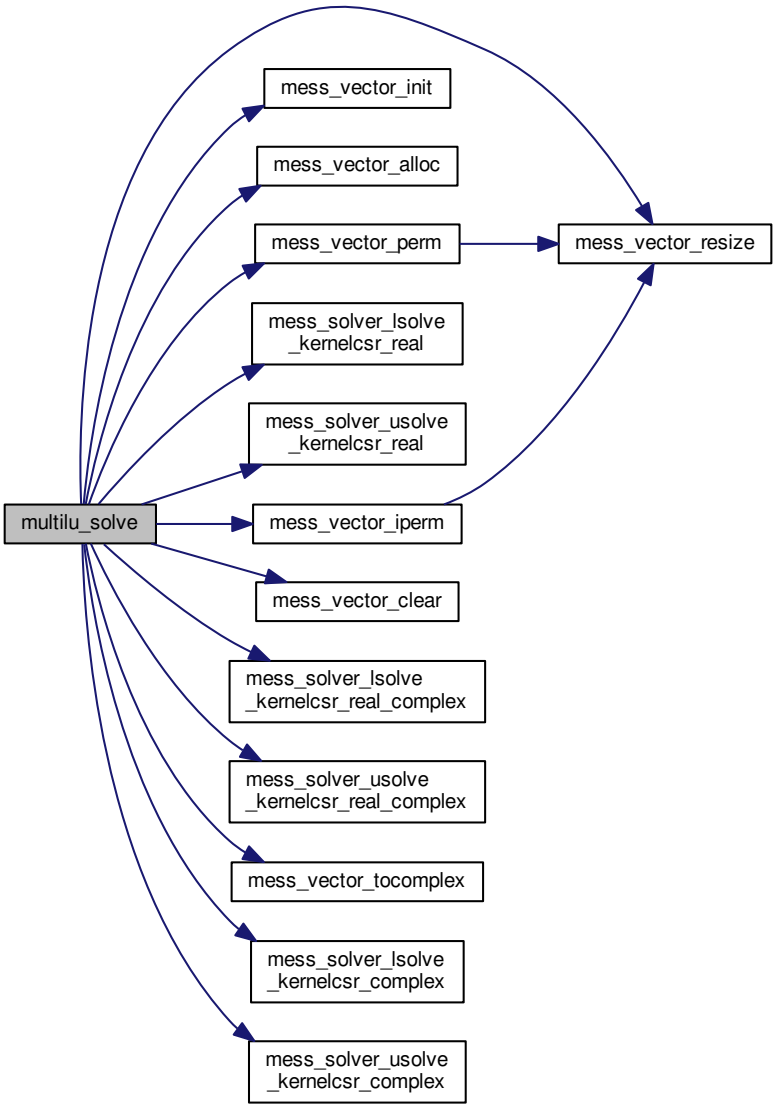
zero on success or a non-zero error value otherwise

The multilu_solve function solves the system

$$A(n)x = b,$$

where b and x are vectors.

Here is the call graph for this function:



10.550.4.11 static int multilu_solveh (void * *data*, mess_int_t *n*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input pointer to a multilu

Parameters

in	*n*	input index of the decomposition
in	*b*	input right hand side
out	*x*	solution

Returns

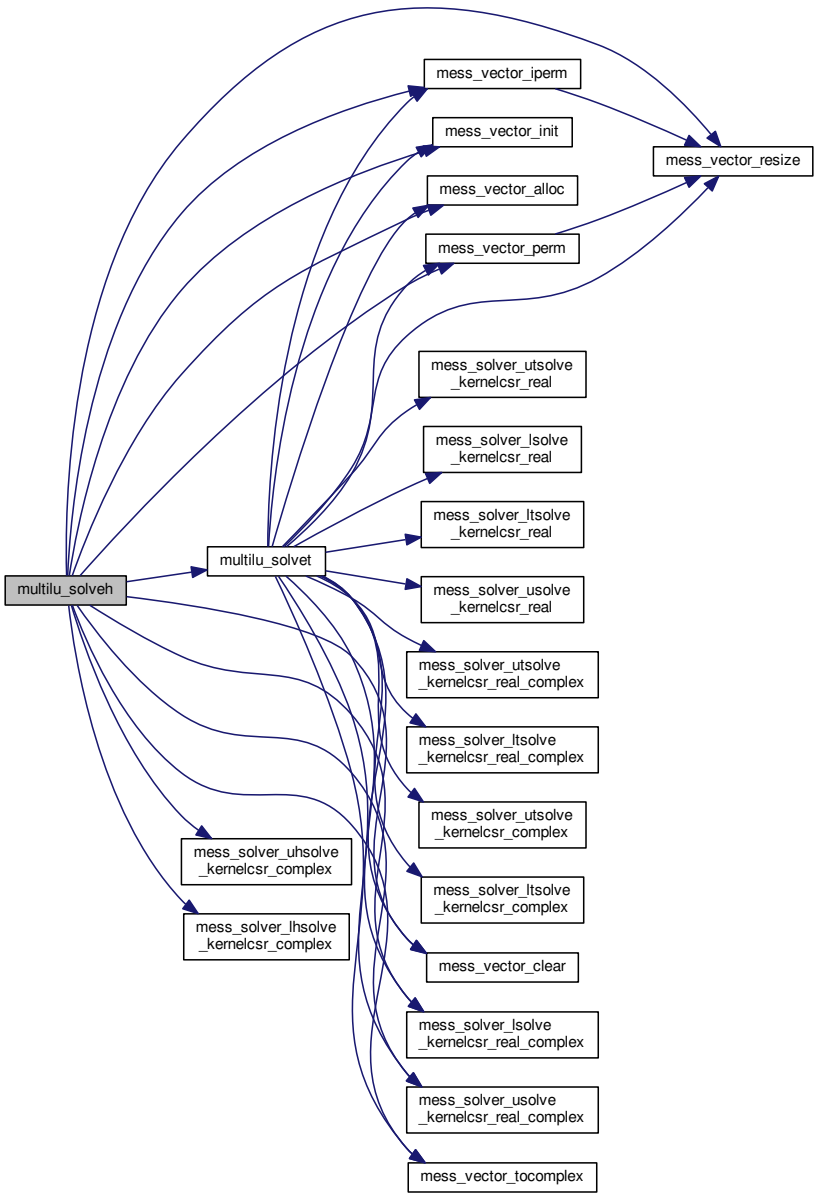
zero on success or a non-zero error value otherwise

The multilu_solveh function solves the system

$$A(n)^H x = b,$$

where b and x are vectors.

Here is the call graph for this function:



10.550.4.12 static int multilu_solvem (void * *data*, mess_int_t *n*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input pointer to a multilu
in	*n*	input index of the decomposition
in	*b*	input right hand side
out	*x*	solution matrix

Returns

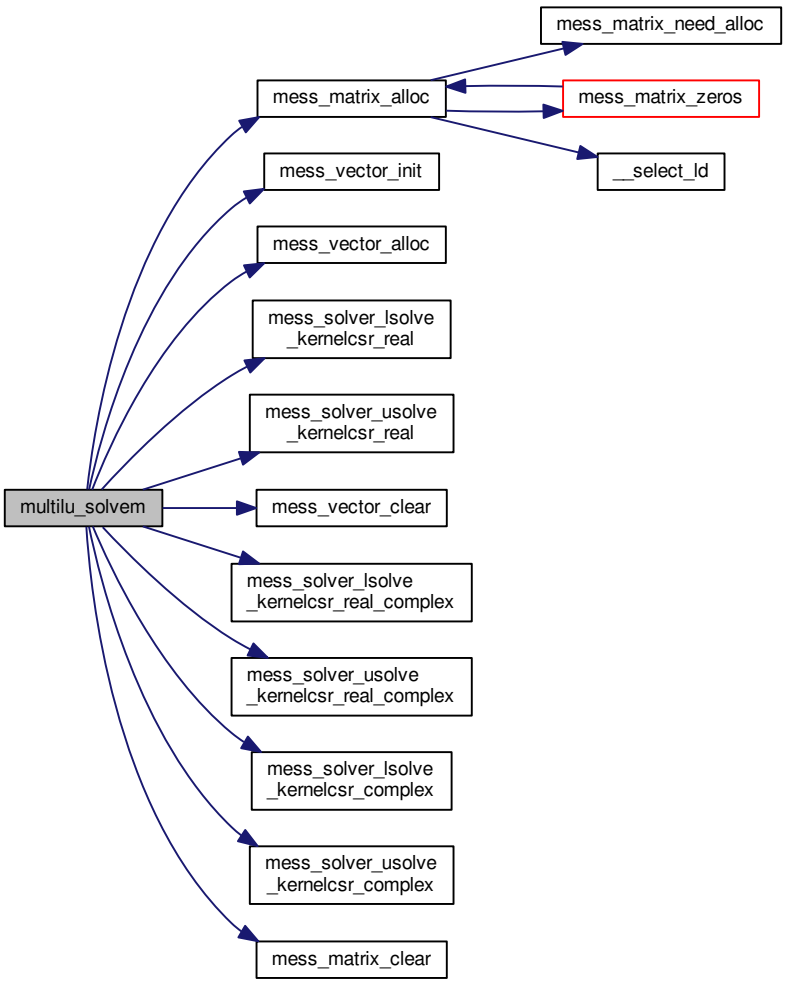
zero on success or a non-zero error value otherwise

The multilu_solvem function solves the system

$$A(n)X = B,$$

where B and X are matrices.

Here is the call graph for this function:



10.550.4.13 static int multilu_solvemh (void * *data*, mess_int_t *n*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input pointer to a multilu
in	*n*	input index of the decomposition
in	*b*	input right hand side
out	*x*	solution matrix

Returns

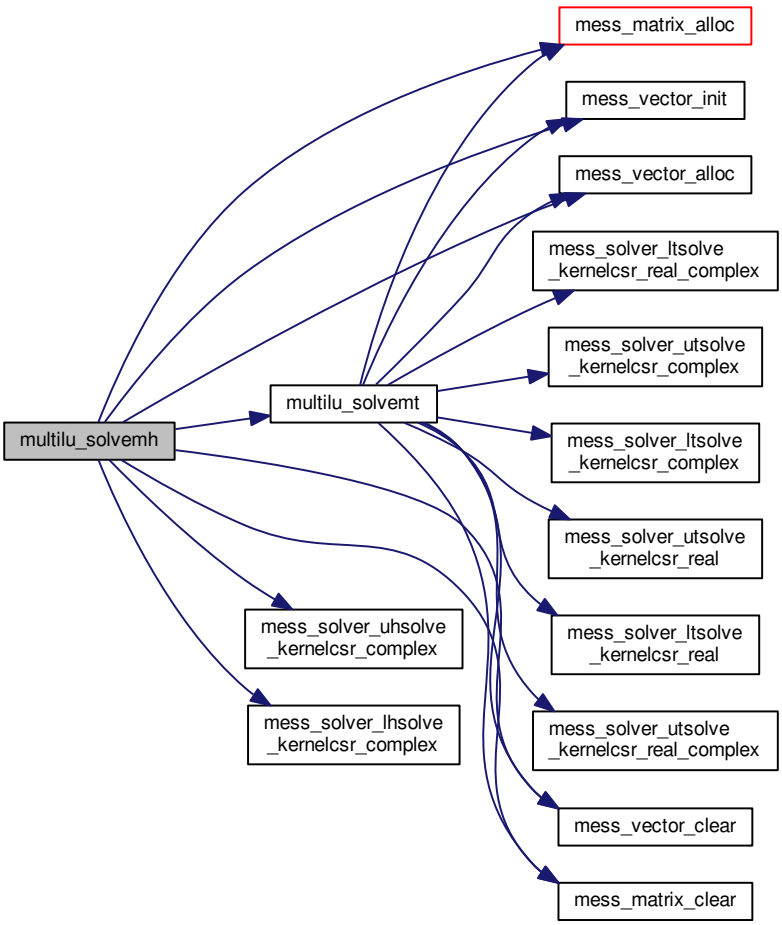
zero on success or a non-zero error value otherwise

The multilu_solvemh function solves the system

$$A(n)^H X = B,$$

where B and X are matrices.

Here is the call graph for this function:



10.550.4.14 static int multilu_solvemt (void * *data*, mess_int_t *n*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input pointer to a multilu
in	*n*	input index of the decomposition
in	*b*	input right hand side
out	*x*	solution matrix

Returns

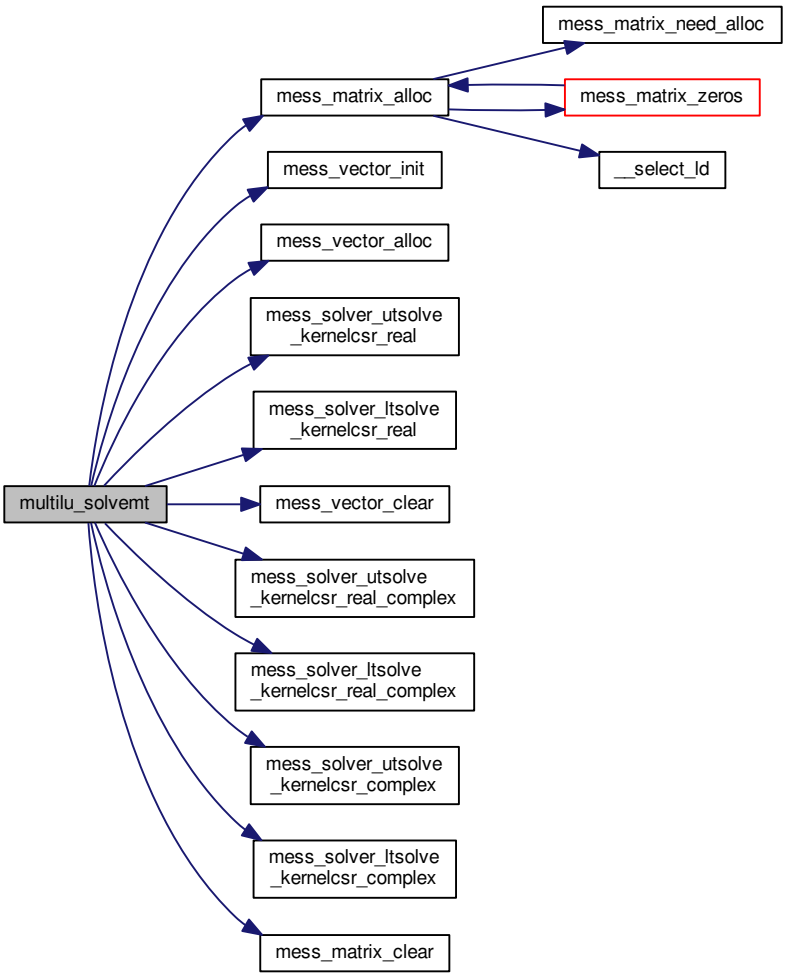
zero on success or a non-zero error value otherwise

The multilu_solvemt function solves the system

$$A(n)^T X = B,$$

where B and X are matrices.

Here is the call graph for this function:



10.550.4.15 static int multilu_solvet (void * *data*, mess_int_t *n*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input pointer to a multilu
in	*n*	input index of the decomposition
in	*b*	input right hand side
out	*x*	solution

Returns

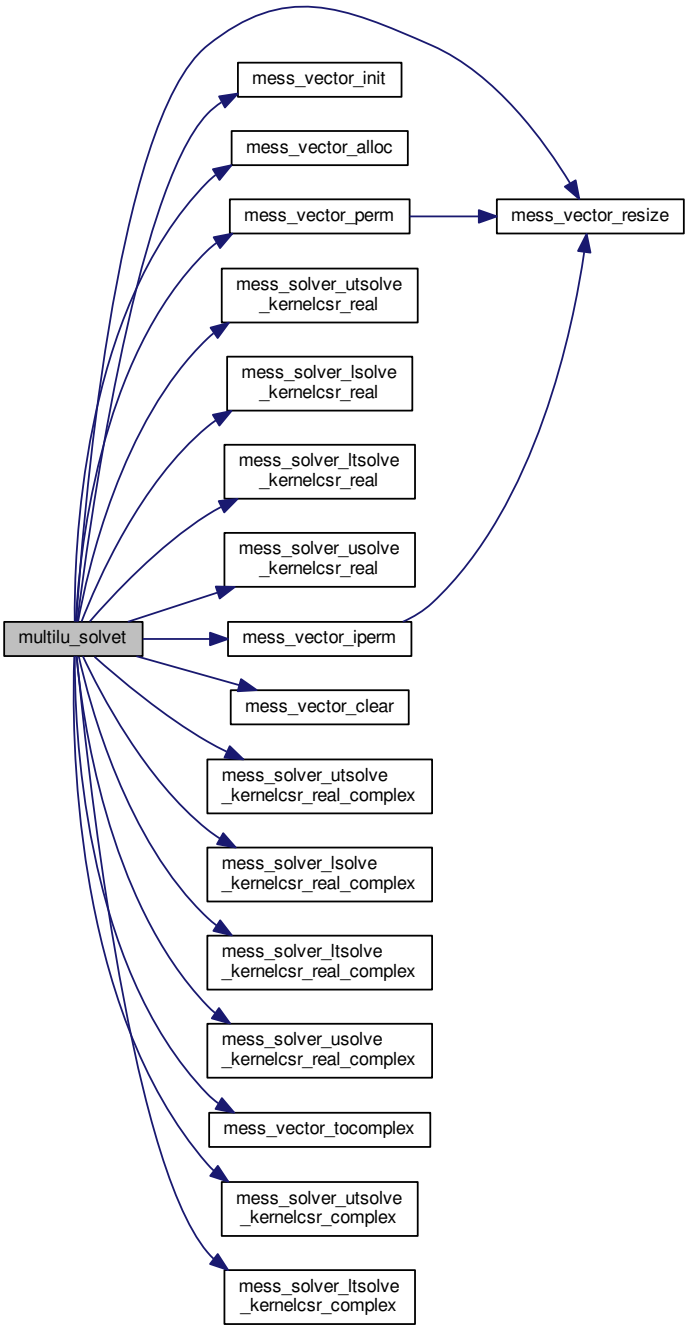
zero on success or a non-zero error value otherwise

The multilu_solvet function solves the system

$$A(n)^T x = b,$$

where b and x are vectors.

Here is the call graph for this function:



10.550.4.16 static mess_int_t* pinv (mess_int_t const * *p*, mess_int_t *n*) [static]

Parameters

in	*p*	input permutation
in	*n*	input length of the permutation

Returns

inverse permutation

The pinv function returns the inverted permution of p.

10.551 multilu_umfpack.c File Reference

Data Structures

- struct multi_umfpack

Macros

- #define IS_REAL(X) (fabs(cimag((X))) $<$ eps)
- #define IS_COMPLEX(X) (fabs(cimag((X))) $>=$ eps)
- #define umfpack_dl_solve umfpack_di_solve
- #define umfpack_zl_solve umfpack_zi_solve
- #define umfpack_zl_load_numeric umfpack_zi_load_numeric
- #define umfpack_dl_load_numeric umfpack_di_load_numeric
- #define umfpack_zl_defaults umfpack_zi_defaults
- #define umfpack_dl_defaults umfpack_di_defaults
- #define umfpack_dl_save_numeric umfpack_di_save_numeric
- #define umfpack_zl_save_numeric umfpack_zi_save_numeric
- #define umfpack_zl_free_numeric umfpack_zi_free_numeric
- #define umfpack_dl_free_numeric umfpack_di_free_numeric
- #define umfpack_dl_symbolic umfpack_di_symbolic
- #define umfpack_dl_numeric umfpack_di_numeric
- #define umfpack_zl_symbolic umfpack_zi_symbolic
- #define umfpack_zl_numeric umfpack_zi_numeric
- #define umfpack_dl_report_info umfpack_di_report_info
- #define umfpack_dl_report_numeric umfpack_di_report_numeric
- #define umfpack_dl_free_symbolic umfpack_di_free_symbolic
- #define umfpack_zl_report_info umfpack_zi_report_info
- #define umfpack_zl_report_numeric umfpack_zi_report_numeric
- #define umfpack_zl_free_symbolic umfpack_zi_free_symbolic

Functions

- static int multiumf_solve (void $*$data, mess_int_t n, mess_vector b, mess_vector x)

  Provide the solve function for $A(n)x = b$.
- static int multiumf_solvem (void $*$data, mess_int_t n, mess_matrix b, mess_matrix x)

  Provide the solve function for $A(n)X = B$.
- static int multiumf_solvet (void $*$data, mess_int_t n, mess_vector b, mess_vector x)

  Provide the solve function for $A(n)^T x = b$.
- static int multiumf_solveh (void $*$data, mess_int_t n, mess_vector b, mess_vector x)

  Provide the solve function for $A(n)^H x = b$.
- static int multiumf_solvemt (void $*$data, mess_int_t n, mess_matrix b, mess_matrix x)

  Provide the solve function for $A(n)^T X = B$.
- static int multiumf_solvemh (void $*$data, mess_int_t n, mess_matrix b, mess_matrix x)

  Provide the solve function for $A(n)^H X = B$.
- static int multiumf_clear (void $*$data)

  Clear an umfpack multisolver.
- int mess_multidirect_create_umfpack (mess_matrix matrix, mess_vector shiftsl, mess_vector shiftsr, mess_multidirect mlu, mess_direct base, mess_matrix shiftmatrix_input)

  Create a multisolver on base of UMFPACK.

10.551.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.551.2 Macro Definition Documentation

10.551.2.1 #define IS_COMPLEX(X) (fabs(cimag((X))) >= eps)

10.551.2.2 #define IS_REAL(X) (fabs(cimag((X))) < eps)

10.551.2.3 #define umfpack_dl_defaults umfpack_di_defaults

10.551.2.4 #define umfpack_dl_free_numeric umfpack_di_free_numeric

10.551.2.5 #define umfpack_dl_free_symbolic umfpack_di_free_symbolic

10.551.2.6 #define umfpack_dl_load_numeric umfpack_di_load_numeric

10.551.2.7 #define umfpack_dl_numeric umfpack_di_numeric

10.551.2.8 #define umfpack_dl_report_info umfpack_di_report_info

10.551.2.9 #define umfpack_dl_report_numeric umfpack_di_report_numeric

10.551.2.10 #define umfpack_dl_save_numeric umfpack_di_save_numeric

10.551.2.11 #define umfpack_dl_solve umfpack_di_solve

10.551.2.12 #define umfpack_dl_symbolic umfpack_di_symbolic

10.551.2.13 #define umfpack_zl_defaults umfpack_zi_defaults

10.551.2.14 #define umfpack_zl_free_numeric umfpack_zi_free_numeric

10.551.2.15 #define umfpack_zl_free_symbolic umfpack_zi_free_symbolic

10.551.2.16 #define umfpack_zl_load_numeric umfpack_zi_load_numeric

10.551.2.17 #define umfpack_zl_numeric umfpack_zi_numeric

10.551.2.18 #define umfpack_zl_report_info umfpack_zi_report_info

10.551.2.19 #define umfpack_zl_report_numeric umfpack_zi_report_numeric

10.551.2.20 #define umfpack_zl_save_numeric umfpack_zi_save_numeric

10.551.2.21 #define umfpack_zl_solve umfpack_zi_solve

10.551.2.22 #define umfpack_zl_symbolic umfpack_zi_symbolic

10.551.3 Function Documentation

10.551.3.1 static int multiumf_clear (void * *data*) [static]

Parameters

in	*data*	input pointer to the data structure

Returns

always zero

The multiumf_clear function clears an umfpack multisolver object.

10.551.3.2 static int multiumf_solve (void * *data*, mess_int_t *n*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input the solver data
in	*n*	input index of the matrix
in	*b*	input right hand side
out	*x*	solution

Returns

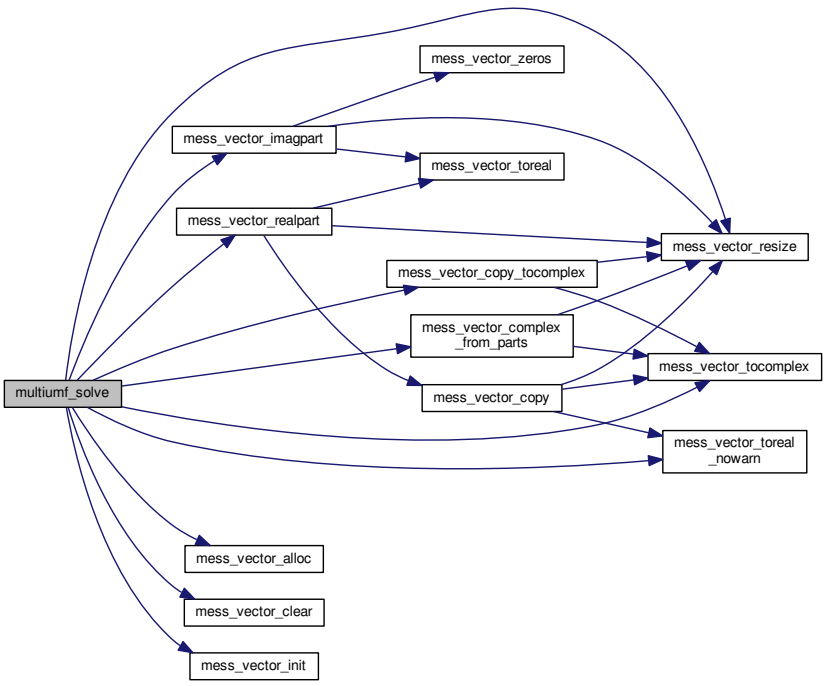
zero on success or a non-zero error value

The multiumf_solve function solve the system

$$A(n)x = b$$

for a single right-hand-side.

Here is the call graph for this function:



10.551.3.3 static int multiumf_solveh (void * *data*, mess_int_t *n*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input the solver data
in	*n*	input index of the matrix
in	*b*	input right hand side
out	*x*	solution

Returns

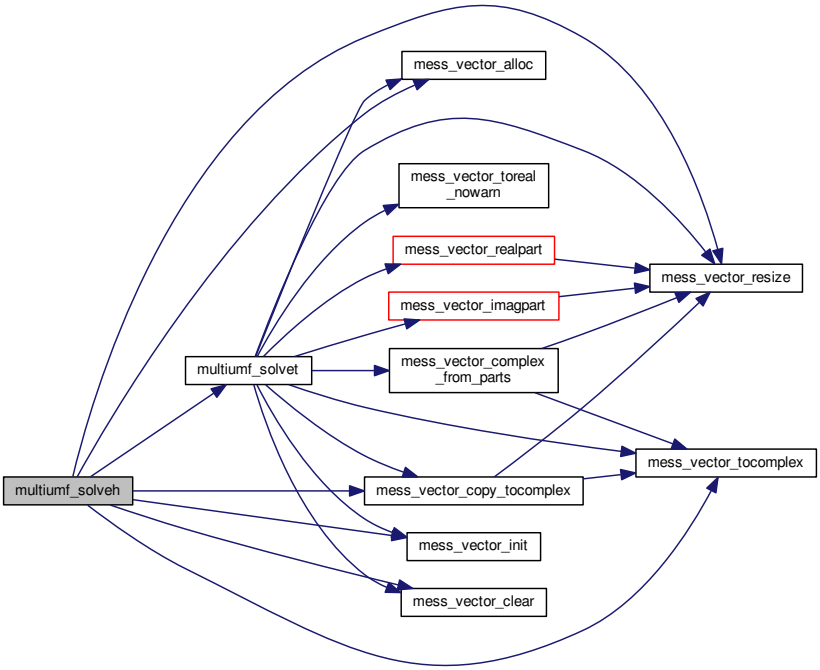
zero on success or a non-zero error value

The multiumf_solvet function solve the system

$$A(n)^H x = b$$

for a single right-hand-side.

Here is the call graph for this function:



10.551.3.4 static int multiumf_solvem (void * *data*, mess_int_t *n*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input the solver data
in	*n*	input index of the matrix
in	*b*	input right hand sides
out	*x*	solution

Returns

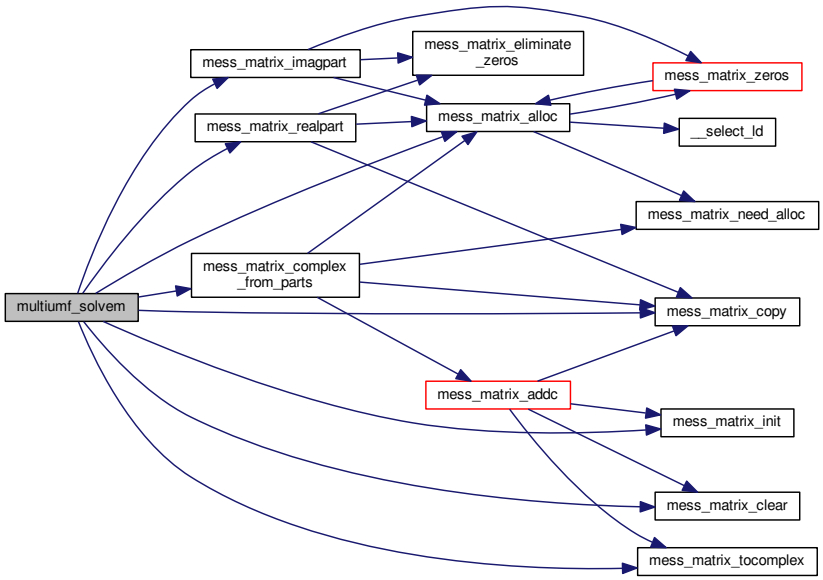
zero on success or a non-zero error value

The multiumf_solvem function solve the system

$$A(n)X = B$$

for a multiple right-hand-sides.

Here is the call graph for this function:



10.551.3.5 static int multiumf_solvemh (void * *data*, mess_int_t *n*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input the solver data
in	*n*	input index of the matrix
in	*b*	input right hand sides
out	*x*	solution

Returns

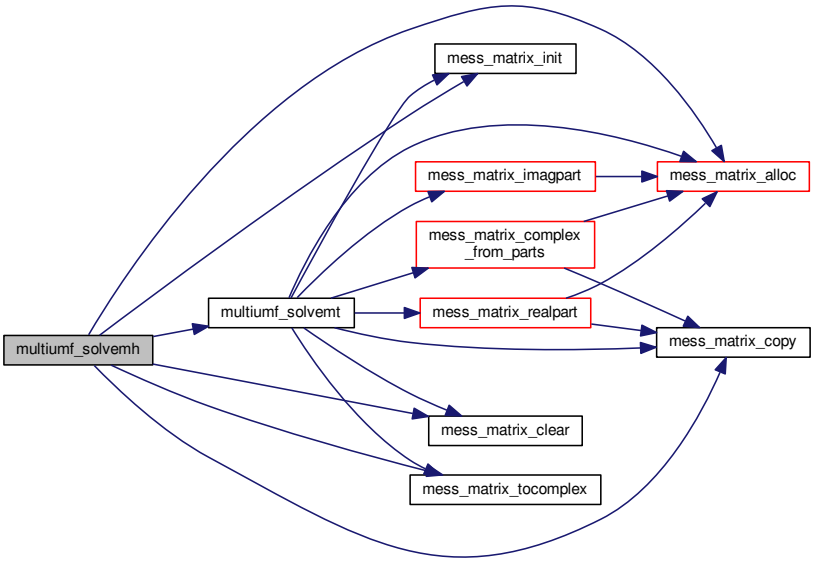
zero on success or a non-zero error value

The multiumf_solvemh function solve the system

$$A(n)^H X = B$$

for a multiple right-hand-sides.

Here is the call graph for this function:



10.551.3.6 static int multiumf_solvemt (void * *data*, mess_int_t *n*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

in	*data*	input the solver data
in	*n*	input index of the matrix
in	*b*	input right hand sides
out	*x*	solution

Returns

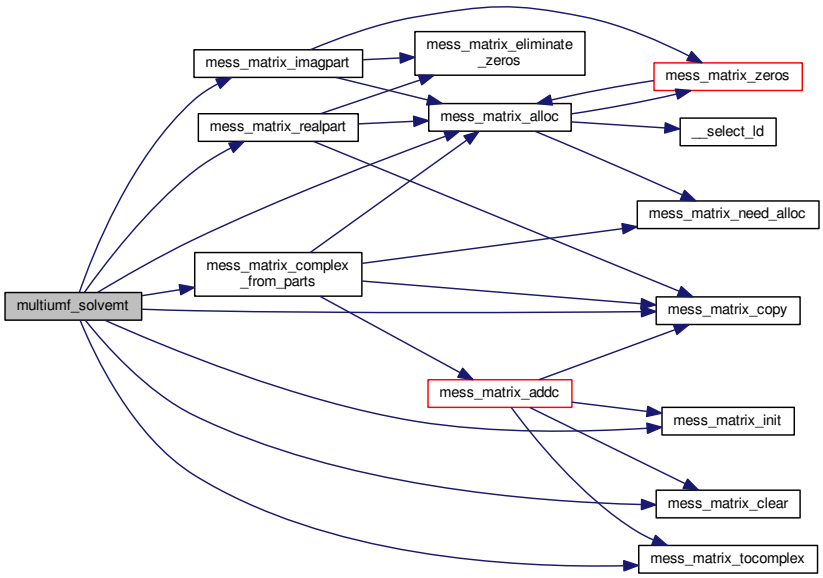
zero on success or a non-zero error value

The multiumf_solvemt function solve the system

$$A(n)^T X = B$$

for a multiple right-hand-sides.

Here is the call graph for this function:



10.551.3.7 static int multiumf_solvet (void * *data*, mess_int_t *n*, mess_vector *b*, mess_vector *x*) [static]

Parameters

in	*data*	input the solver data
in	*n*	input index of the matrix
in	*b*	input right hand side
out	*x*	solution

Returns

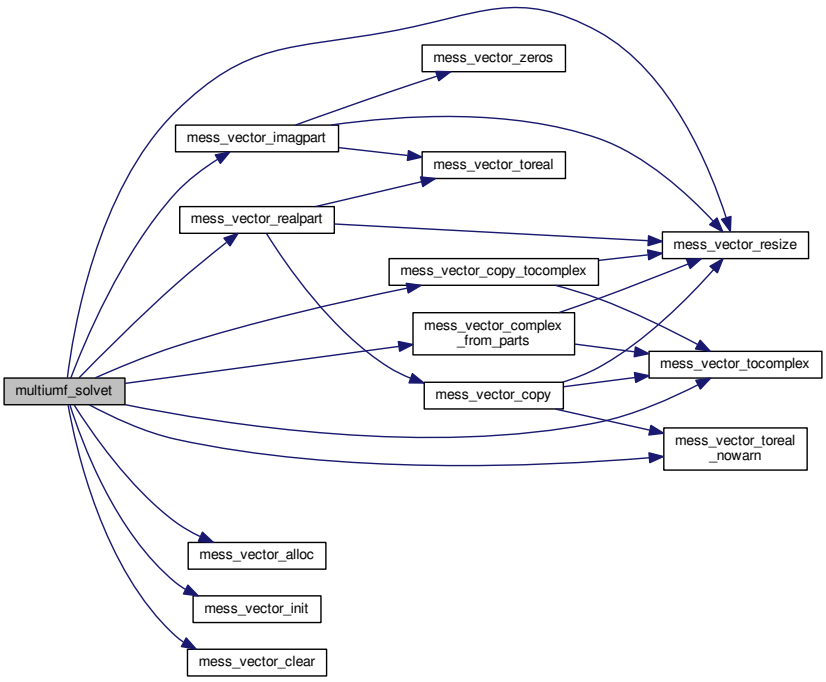
zero on success or a non-zero error value

The multiumf_solvet function solve the system

$$A(n)^T x = b$$

for a single right-hand-side.

Here is the call graph for this function:



10.552 mvp.c File Reference

Functions

- int mess_matrix_mvp (mess_operation_t op, mess_matrix A, mess_vector x, mess_vector y)

  Compute a matrix-vector product.

10.552.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.553 mvpcall.c File Reference

Functions

- static int mvp_normal (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)
- static int mvp_transpose (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)
- static int mvp_hermitian (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)
- int mess_mvpcall_matrix (mess_mvpcall ∗mvpcall, mess_operation_t op, mess_matrix A)

  Generate a mess_mvpcall object for a simple matrix.
- int mess_mvpcall_operator (mess_mvpcall ∗mvpcall, mess_int_t dim, mess_datatype_t data_type, int(∗mvp)(void ∗data, mess_operation_t op, mess_vector x, mess_vector y), void ∗data)

  Generate a mess_mvpcall object for a complex operator.
- int mess_mvpcall_apply (mess_mvpcall mvpcall, mess_operation_t op, mess_vector x, mess_vector y)

  Apply a generic matrix-vector product.
- int mess_mvpcall_clear (mess_mvpcall ∗mvpcall)

  Clean a mess_mvpcall object .

10.553.1 Detailed Description

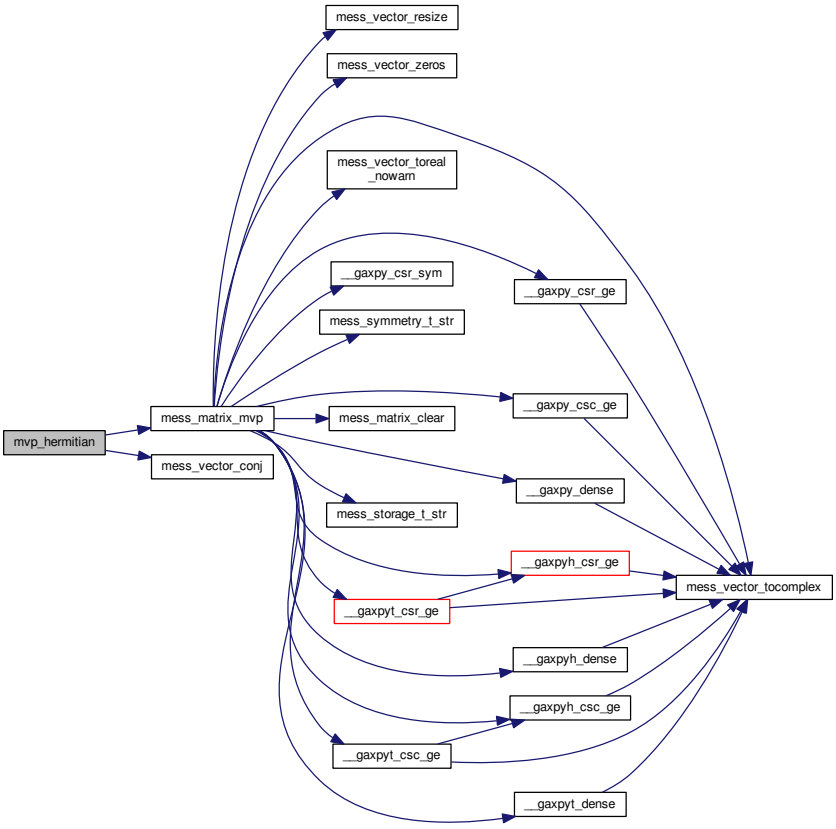
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.553.2 Function Documentation

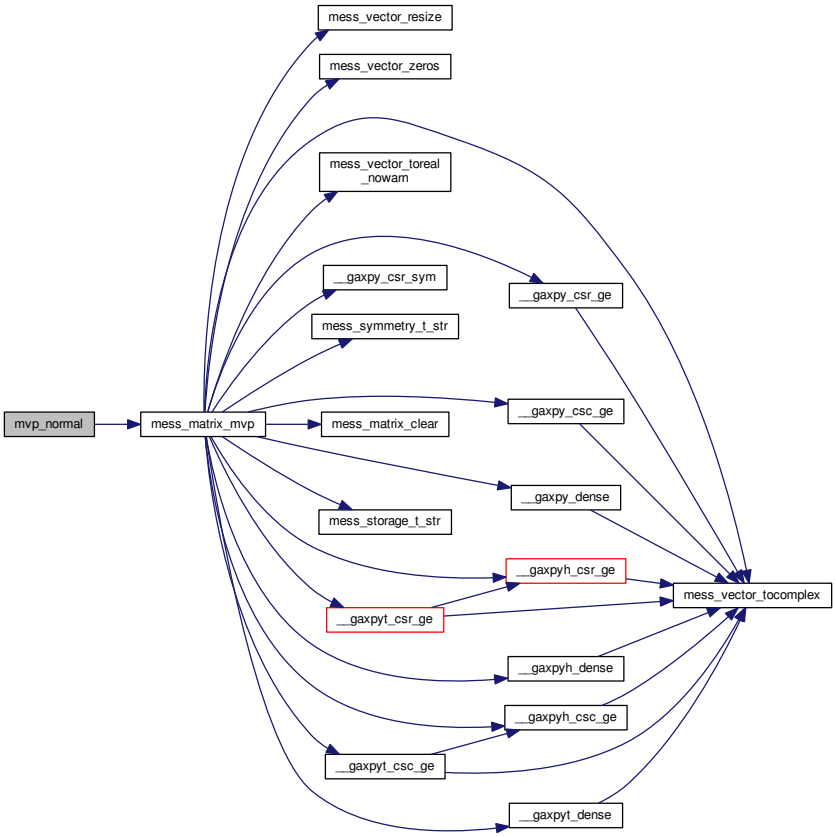
10.553.2.1 static int mvp_hermitian (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



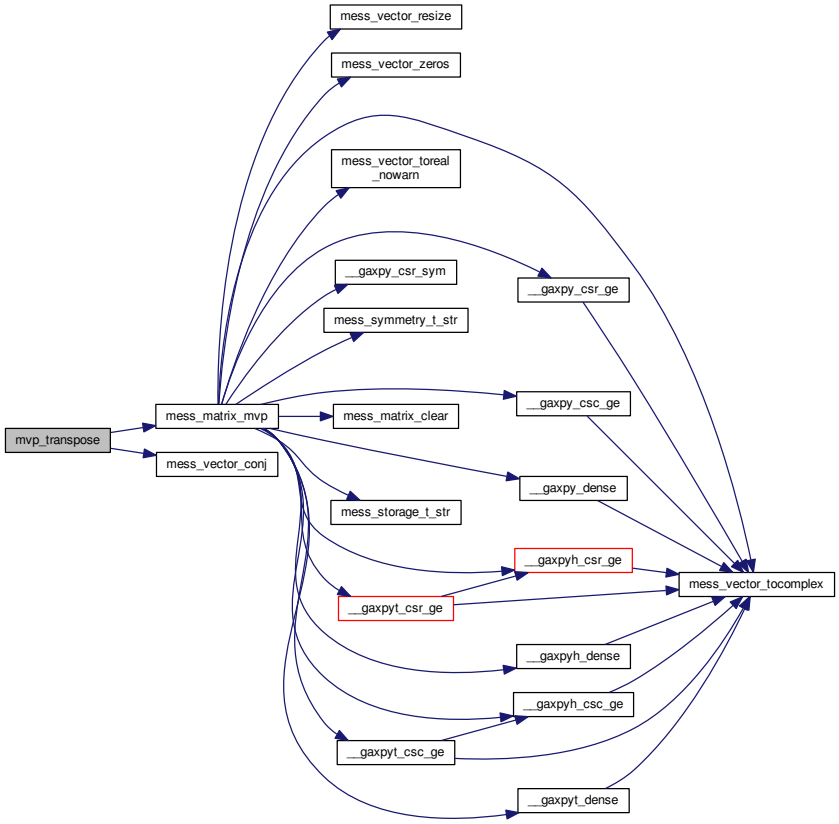
10.553.2.2 static int mvp_normal (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



10.553.2.3 static int mvp_transpose (void ∗ *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



10.554 mxIsSubClass.c File Reference

Functions

- int mxIsSubClass (const mxArray ∗instance, char ∗class)

10.554.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.555 MY_MODULE_mp_MY_SUB-UPPER.c File Reference

Functions

- const char ∗ MY_MODULE_mp_MY_SUB (void)

10.555.1 Function Documentation

10.555.1.1 const char* MY_MODULE_mp_MY_SUB (void)

10.556 MY_MODULE_mp_MY_SUB-UPPER.c File Reference

Functions

- const char * MY_MODULE_mp_MY_SUB (void)

10.556.1 Function Documentation

10.556.1.1 const char* MY_MODULE_mp_MY_SUB (void)

10.557 my_module_MP_my_sub.c File Reference

Functions

- const char * my_module_MP_my_sub (void)

10.557.1 Function Documentation

10.557.1.1 const char* my_module_MP_my_sub (void)

10.558 my_module_MP_my_sub.c File Reference

Functions

- const char * my_module_MP_my_sub (void)

10.558.1 Function Documentation

10.558.1.1 const char* my_module_MP_my_sub (void)

10.559 my_module_mp_my_sub_.c File Reference

Functions

- const char * my_module_mp_my_sub_ (void)

10.559.1 Function Documentation

10.559.1.1 const char* my_module_mp_my_sub_ (void)

10.560 my_module_mp_my_sub_.c File Reference

Functions

- const char * my_module_mp_my_sub_ (void)

10.560.1 Function Documentation

10.560.1.1 const char* my_module_mp_my_sub_ (void)

10.561 my_module_my_sub_.c File Reference

Functions

- const char * my_module_my_sub_ (void)

10.561.1 Function Documentation

10.561.1.1 const char* my_module_my_sub_ (void)

10.562 my_module_my_sub_.c File Reference

Functions

- const char * my_module_my_sub_ (void)

10.562.1 Function Documentation

10.562.1.1 const char* my_module_my_sub_ (void)

10.563 my_moduleSmy_sub.c File Reference

Functions

- const char *my_module $my_sub (void)

10.563.1 Function Documentation

10.563.1.1 const char* my_module $my_sub (void)

10.564 my_moduleSmy_sub.c File Reference

Functions

- const char *my_module $my_sub (void)

10.564.1 Function Documentation

10.564.1.1 const char* my_module $my_sub (void)

10.565 MY_SUB-UPPER.c File Reference

Functions

- const char * MY_SUB (void)

10.565.1 Function Documentation

10.565.1.1 const char* MY_SUB (void)

10.566 MY_SUB-UPPER.c File Reference

Functions

- const char * MY_SUB (void)

10.566.1 Function Documentation

10.566.1.1 const char* MY_SUB (void)

10.567 my_sub.c File Reference

Functions

- const char * my_sub (void)

10.567.1 Function Documentation

10.567.1.1 const char* my_sub (void)

10.568 my_sub.c File Reference

Functions

- const char * my_sub (void)

10.568.1 Function Documentation

10.568.1.1 const char* my_sub (void)

10.569 my_sub_.c File Reference

Functions

- const char * my_sub_ (void)

10.569.1 Function Documentation

10.569.1.1 const char* my_sub_ (void)

10.570 my_sub_.c File Reference

Functions

- const char * my_sub_ (void)

10.570.1 Function Documentation

10.570.1.1 const char* my_sub_ (void)

10.571 my_sub__.c File Reference

Functions

- const char * my_sub__ (void)

10.571.1 Function Documentation

10.571.1.1 const char* my_sub__ (void)

10.572 my_sub__.c File Reference

Functions

- const char * my_sub__ (void)

10.572.1 Function Documentation

10.572.1.1 const char* my_sub__ (void)

10.573 MYMODULE_mp_MYSUB-UPPER.c File Reference

Functions

- const char * MYMODULE_mp_MYSUB (void)

10.573.1 Function Documentation

10.573.1.1 const char* MYMODULE_mp_MYSUB (void)

10.574 MYMODULE_mp_MYSUB-UPPER.c File Reference

Functions

- const char * MYMODULE_mp_MYSUB (void)

10.574.1 Function Documentation

10.574.1.1 const char* MYMODULE_mp_MYSUB (void)

10.575 mymodule_MP_mysub.c File Reference

Functions

- const char * mymodule_MP_mysub (void)

10.575.1 Function Documentation

10.575.1.1 const char∗ mymodule_MP_mysub (void)

10.576 mymodule_MP_mysub.c File Reference

Functions

- const char ∗ mymodule_MP_mysub (void)

10.576.1 Function Documentation

10.576.1.1 const char∗ mymodule_MP_mysub (void)

10.577 mymodule_mp_mysub_.c File Reference

Functions

- const char ∗ mymodule_mp_mysub_ (void)

10.577.1 Function Documentation

10.577.1.1 const char∗ mymodule_mp_mysub_ (void)

10.578 mymodule_mp_mysub_.c File Reference

Functions

- const char ∗ mymodule_mp_mysub_ (void)

10.578.1 Function Documentation

10.578.1.1 const char∗ mymodule_mp_mysub_ (void)

10.579 mymodule_mysub_.c File Reference

Functions

- const char ∗ mymodule_mysub_ (void)

10.579.1 Function Documentation

10.579.1.1 const char∗ mymodule_mysub_ (void)

10.580 mymodule_mysub_.c File Reference

Functions

- const char ∗ mymodule_mysub_ (void)

10.580.1 Function Documentation

10.580.1.1 const char* mymodule_mysub_ (void)

10.581 mymoduleSmysub.c File Reference

Functions

- const char *mymodule $mysub (void)

10.581.1 Function Documentation

10.581.1.1 const char* mymodule $mysub (void)

10.582 mymoduleSmysub.c File Reference

Functions

- const char *mymodule $mysub (void)

10.582.1 Function Documentation

10.582.1.1 const char* mymodule $mysub (void)

10.583 MYSUB-UPPER.c File Reference

Functions

- const char * MYSUB (void)

10.583.1 Function Documentation

10.583.1.1 const char* MYSUB (void)

10.584 MYSUB-UPPER.c File Reference

Functions

- const char * MYSUB (void)

10.584.1 Function Documentation

10.584.1.1 const char* MYSUB (void)

10.585 mysub.c File Reference

Functions

- const char * mysub (void)

10.585.1 Function Documentation

10.585.1.1 const char* mysub (void)

10.586 mysub.c File Reference

Functions

- const char * mysub (void)

10.586.1 Function Documentation

10.586.1.1 const char* mysub (void)

10.587 mysub_.c File Reference

Functions

- const char * mysub_ (void)

10.587.1 Function Documentation

10.587.1.1 const char* mysub_ (void)

10.588 mysub_.c File Reference

Functions

- const char * mysub_ (void)

10.588.1 Function Documentation

10.588.1.1 const char* mysub_ (void)

10.589 newlu.c File Reference

Data Structures

- struct newlu_data

Macros

- #define FLIP(x) (-(x)-2)
- #define UNFLIP(x) (((x) < 0)? FLIP(x):(x))
- #define MARKED(w, x) (w[(x)] < 0)
- #define MARK(w, x) { w[(x)] = FLIP(w[(x)]); }

Functions

- static mess_int_t * perm_inv (mess_int_t const *p, mess_int_t n)
- static int newlu_getL (void *data, mess_matrix L)
- static int newlu_getU (void *data, mess_matrix U)
- static int newlu_getpermp (void *data, mess_int_t *p)
- static int newlu_getpermq (void *data, mess_int_t *q)
- static int newlu_clear (void *data)
- static int lsolve (mess_matrix L, double *x)
- static int lsolverc (mess_matrix L, mess_double_cpx_t *x)
- static int lsolvec (mess_matrix L, mess_double_cpx_t *x)
- static int ltsolve (mess_matrix L, double *x)
- static int ltsolverc (mess_matrix L, mess_double_cpx_t *x)
- static int ltsolvec (mess_matrix L, mess_double_cpx_t *x)
- static int lhsolvec (mess_matrix L, mess_double_cpx_t *x)
- static int usolve (mess_matrix U, double *x)
- static int usolverc (mess_matrix U, mess_double_cpx_t *x)
- static int usolvec (mess_matrix U, mess_double_cpx_t *x)
- static int utsolve (mess_matrix U, double *x)
- static int utsolverc (mess_matrix U, mess_double_cpx_t *x)
- static int utsolvec (mess_matrix U, mess_double_cpx_t *x)
- static int uhsolvec (mess_matrix U, mess_double_cpx_t *x)
- static int newlu_solve (void *data, mess_vector b, mess_vector x)
- static int newlu_solvem (void *data, mess_matrix b, mess_matrix x)
- static int newlu_solvet (void *data, mess_vector b, mess_vector x)
- static int newlu_solvemt (void *data, mess_matrix b, mess_matrix x)
- static int newlu_solveh (void *data, mess_vector b, mess_vector x)
- static int newlu_solvemh (void *data, mess_matrix b, mess_matrix x)
- static int newlu_inverse (void *data, mess_matrix inv)
- int mess_direct_create_sparse_lu (mess_matrix A, mess_direct sol)

  Compute a sparse left-looking LU decomposition.

10.589.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.589.2 Macro Definition Documentation

10.589.2.1 #define FLIP(x) (-(x)-2)

10.589.2.2 #define MARK(w, x) { w[(x)] = FLIP(w[(x)]); }

10.589.2.3 #define MARKED(w, x) (w[(x)] < 0)

10.589.2.4 #define UNFLIP(x) (((x) < 0)? FLIP(x):(x))

10.589.3 Function Documentation

10.589.3.1 static int lhsolvec (mess_matrix *L*, mess_double_cpx_t * *x*) [static]

10.589.3.2 static int lsolve (mess_matrix *L*, double * *x*) [static]

10.589.3.3 static int lsolvec (mess_matrix *L*, mess_double_cpx_t * *x*) [static]

10.589.3.4 static int lsolverc (mess_matrix *L*, mess_double_cpx_t * *x*) [static]

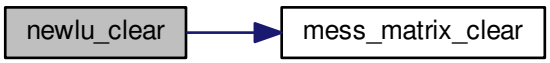
10.589.3.5 static int ltsolve (mess_matrix *L*, double * *x*) [static]

10.589.3.6 static int ltsolvec (mess_matrix *L*, mess_double_cpx_t * *x*) [static]

10.589.3.7 static int ltsolverc (mess_matrix *L*, mess_double_cpx_t * *x*) [static]

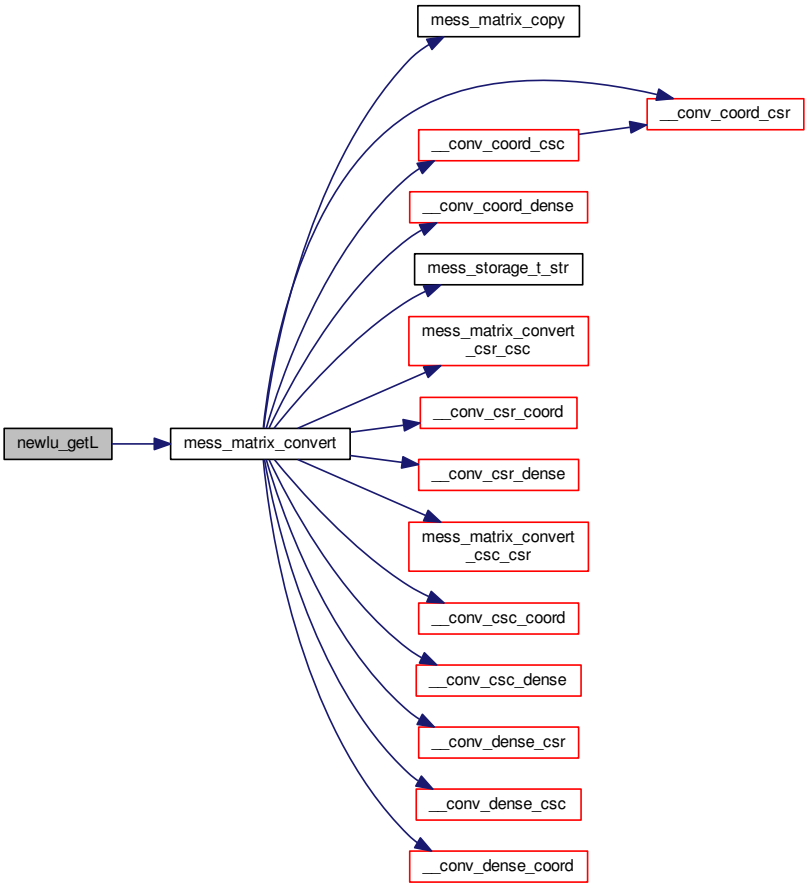
10.589.3.8 static int newlu_clear (void * *data*) [static]

Here is the call graph for this function:



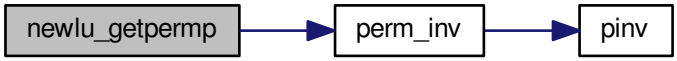
10.589.3.9 static int newlu_getL (void * *data*, mess_matrix *L*) [static]

Here is the call graph for this function:



10.589.3.10 static int newlu_getpermp (void * *data*, mess_int_t * *p*) [static]

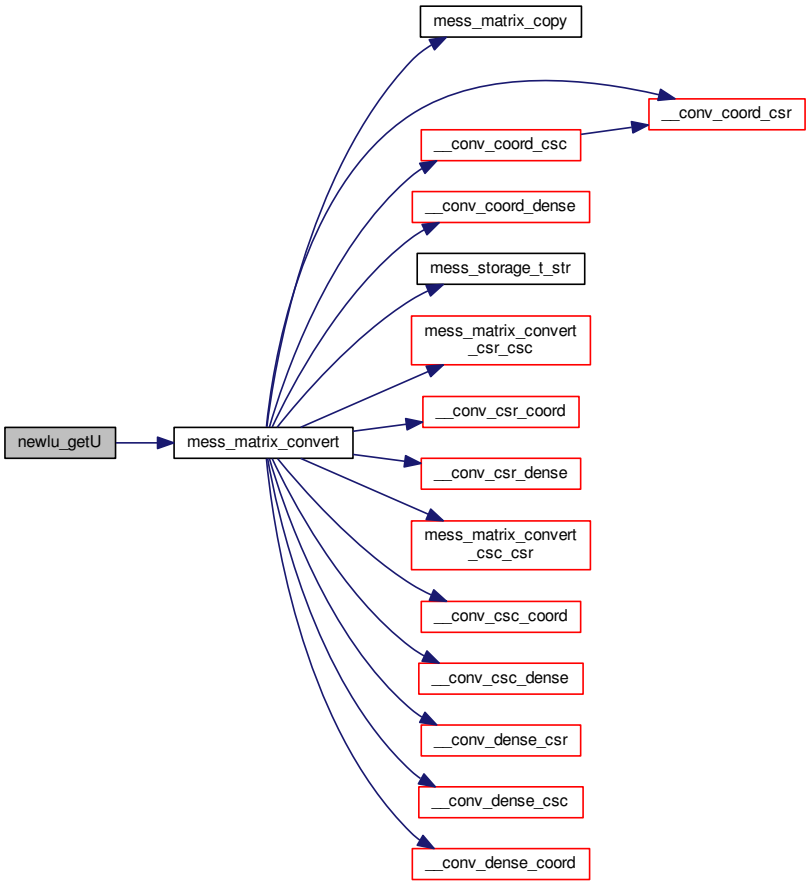
Here is the call graph for this function:



10.589.3.11 static int newlu_getpermq (void * *data*, mess_int_t * *q*) [static]

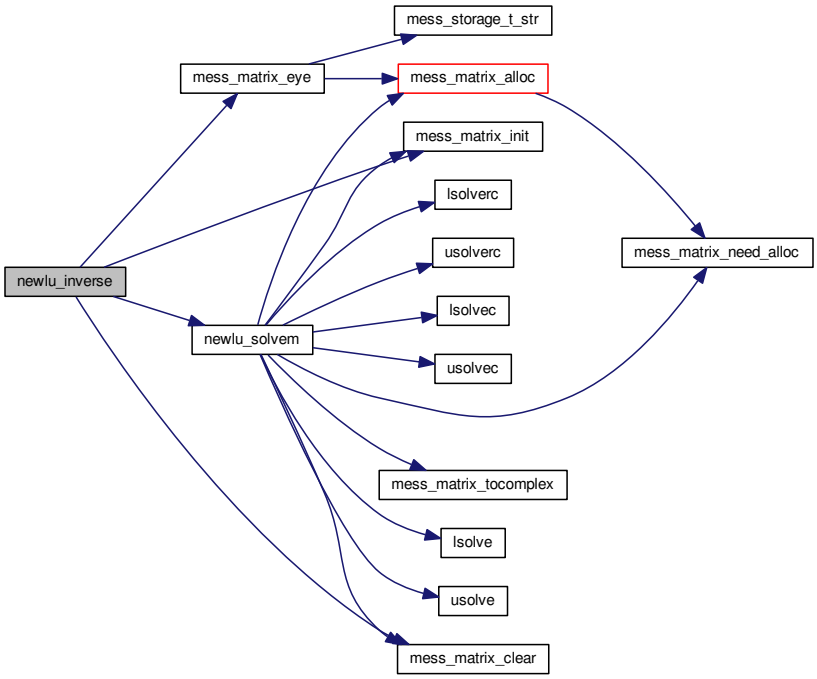
10.589.3.12 static int newlu_getU (void * *data,* mess_matrix *U*) [static]

Here is the call graph for this function:



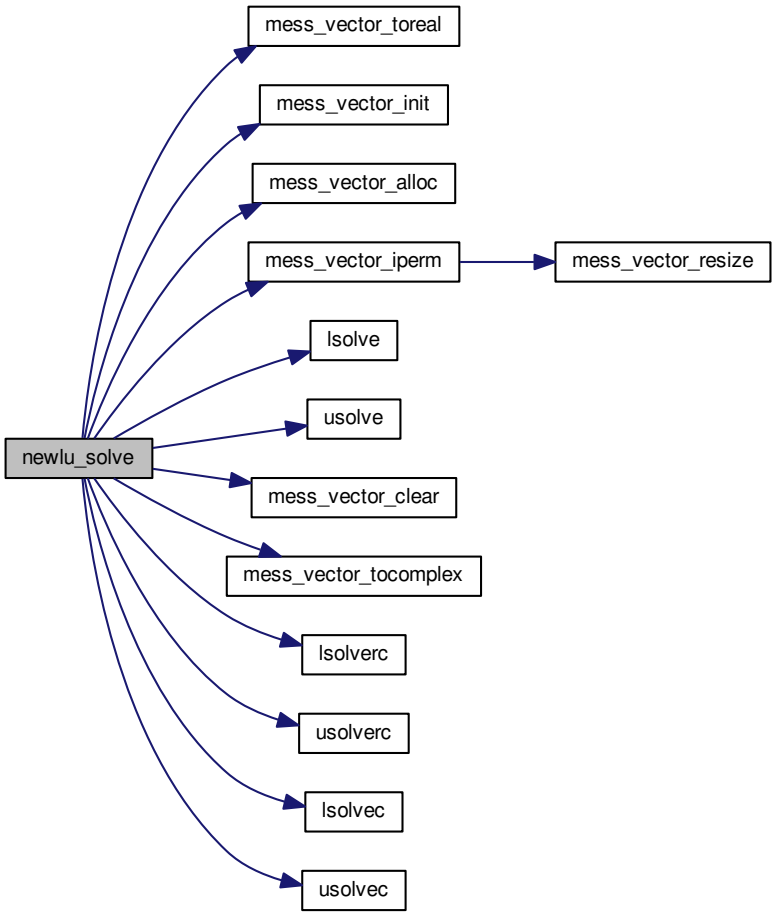
10.589.3.13 static int newlu_inverse (void * *data*, mess_matrix *inv*) [static]

Here is the call graph for this function:



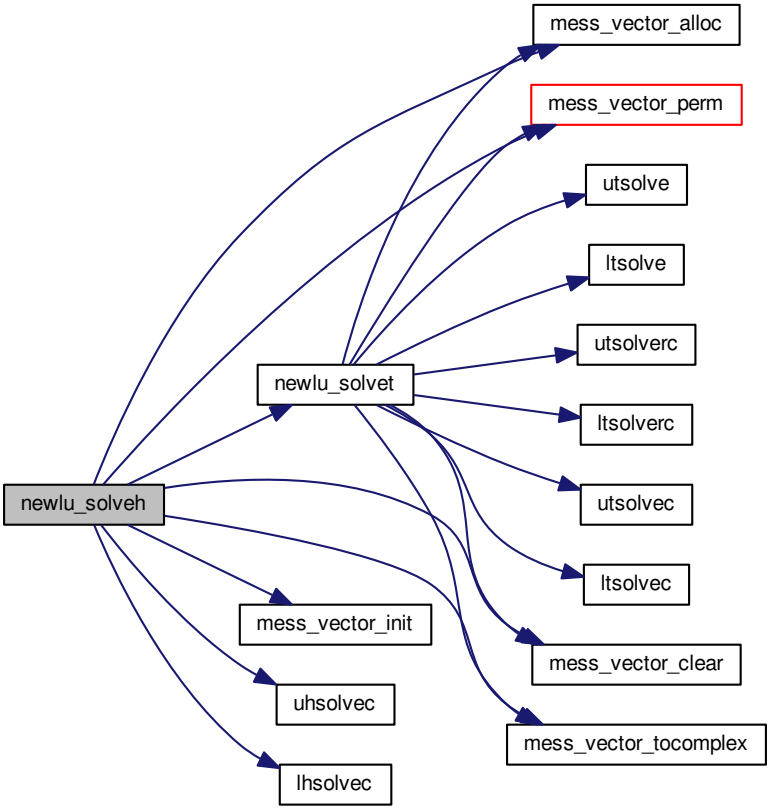
10.589.3.14 static int newlu_solve (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Here is the call graph for this function:



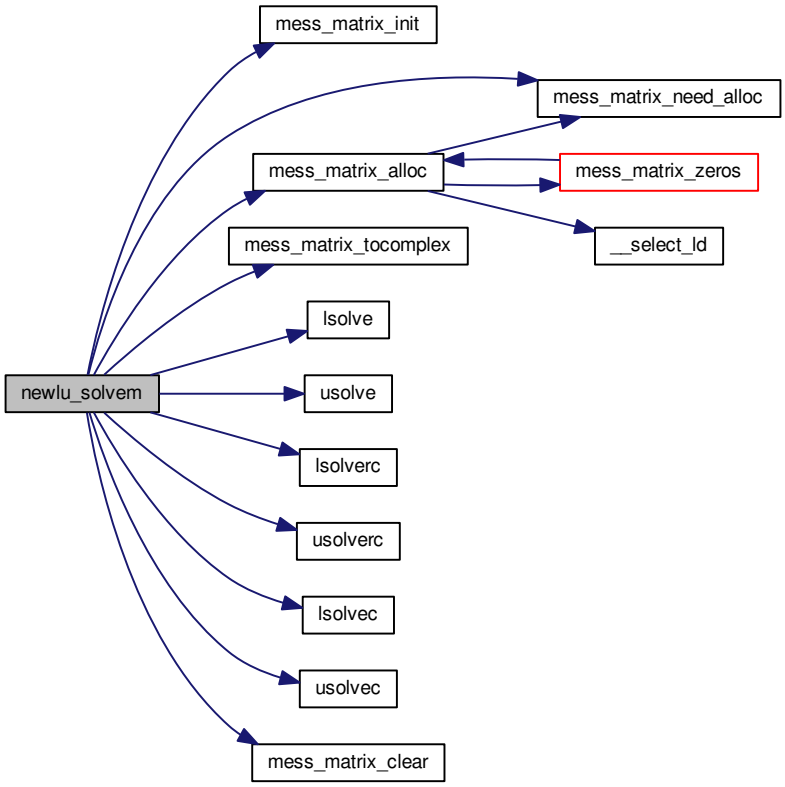
10.589.3.15 static int newlu_solveh (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Here is the call graph for this function:



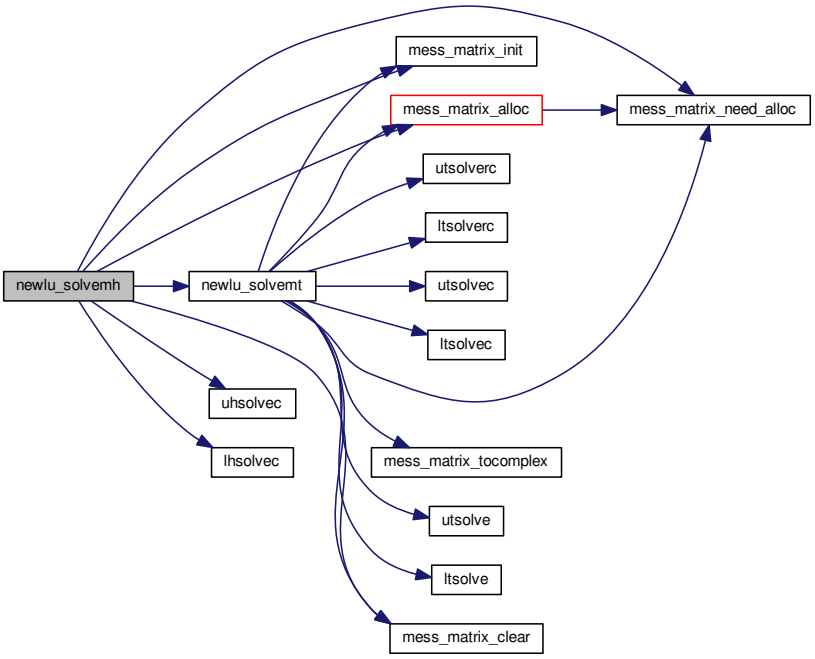
10.589.3.16 static int newlu_solvem (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Here is the call graph for this function:



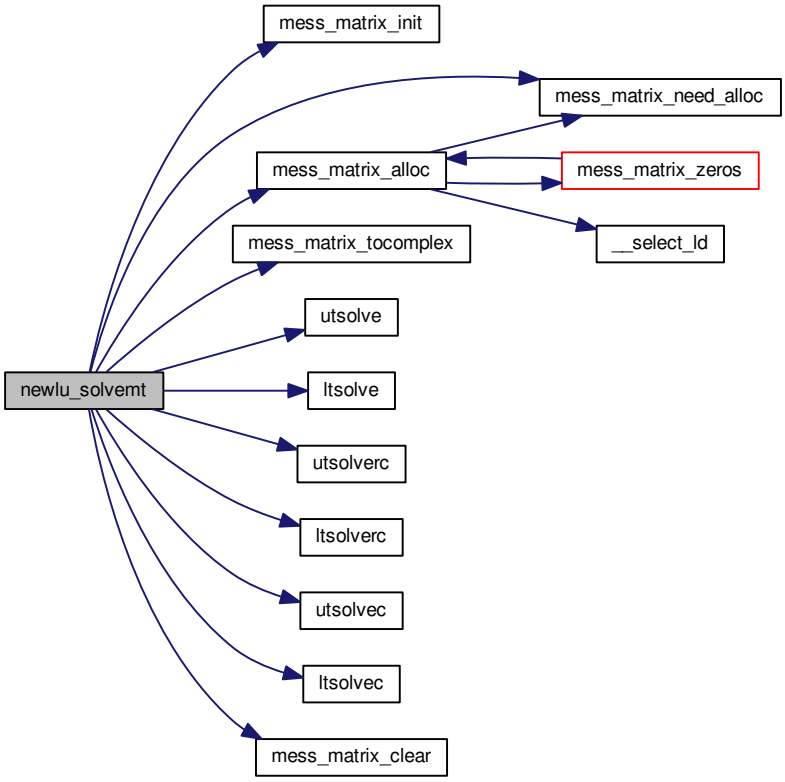
10.589.3.17 static int newlu_solvemh (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Here is the call graph for this function:



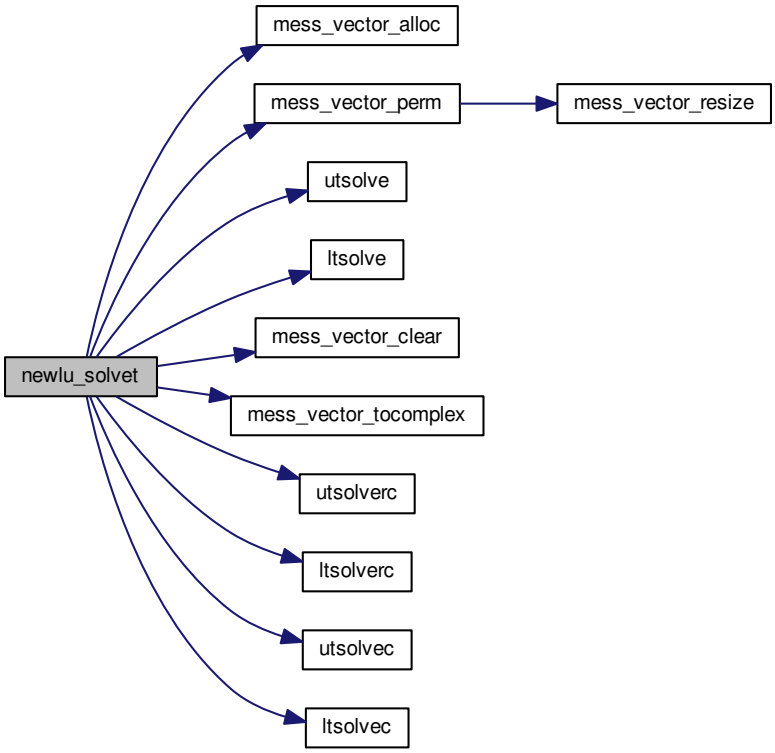
10.589.3.18 static int newlu_solvemt (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Here is the call graph for this function:



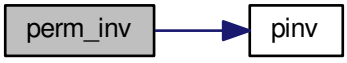
10.589.3.19 static int newlu_solvet (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Here is the call graph for this function:



10.589.3.20 static mess_int_t* perm_inv (mess_int_t const * *p*, mess_int_t *n*) [static]

Here is the call graph for this function:



10.589.3.21 static int uhsolvec (mess_matrix *U*, mess_double_cpx_t * *x*) [static]

10.589.3.22 static int usolve (mess_matrix *U*, double * *x*) [static]

10.589.3.23 static int usolvec (mess_matrix *U*, mess_double_cpx_t * *x*) [static]

10.589.3.24 static int usolverc (mess_matrix *U*, mess_double_cpx_t * *x*) [static]

10.589.3.25 static int utsolve (mess_matrix *U*, double * *x*) [static]

10.589.3.26 static int utsolvec (mess_matrix *U*, mess_double_cpx_t * *x*) [static]

10.589.3.27 static int utsolverc (mess_matrix *U*, mess_double_cpx_t * *x*) [static]

10.590 norm.c File Reference

Data Structures

- struct norm2data
- struct norminv2data

Functions

- int mess_matrix_norm1 (mess_matrix A, double *nrm)

  Compute the 1-norm of a matrix.
- int mess_matrix_norminf (mess_matrix A, double *nrm)

  Compute the ∞-norm of a matrix.
- int mess_matrix_normf2 (mess_matrix A, double *nrm)

  Compute the square of the Frobenius-norm of a matrix.
- int mess_matrix_mulnormf2 (mess_operation_t op1, mess_matrix A, mess_operation_t op2, mess_matrix B, double *nrm)

  Compute the square of the Frobenius-norm of a $op1(A) * op2(B)$.
- int mess_matrix_normf (mess_matrix A, double *nrm)

  Compute the Frobenius-norm of a matrix.
- static int norm2mvp (void *data, mess_operation_t op, mess_vector x, mess_vector y)
- int mess_matrix_norm2 (mess_matrix A, double *nrm)

  Compute the 2-norm of a matrix.
- static int norminv2mvp (void *data, mess_operation_t op, mess_vector x, mess_vector y)
- int mess_matrix_norm2inv (mess_matrix A, double *nrm)

  Compute the 2-norm of the inverse of a matrix.
- int mess_matrix_norm (mess_matrix A, mess_norm_t nrm_t, double *nrm)

  Compute the norm of a matrix.

10.590.1 Detailed Description

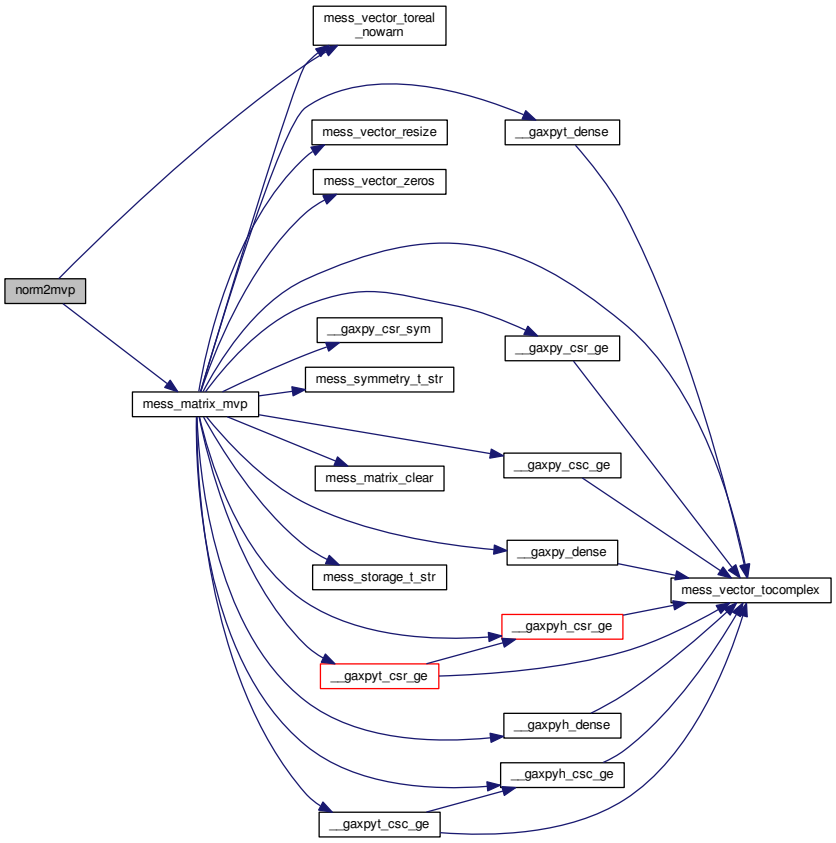
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.590.2 Function Documentation

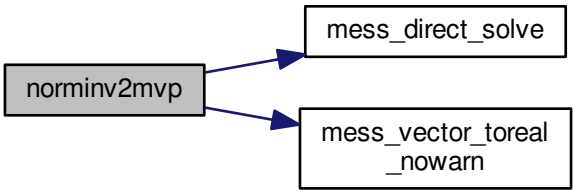
10.590.2.1 static int norm2mvp (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



10.590.2.2 static int norminv2mvp (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



10.591 norm.c File Reference

Functions

- int mess_vector_norm2 (mess_vector x, double ∗nrm)

  Compute the 2-norm of a vector.
- int mess_vector_norm1 (mess_vector x, double ∗nrm)

  Compute the 1-norm of a vector.
- int mess_vector_norminf (mess_vector x, double ∗nrm)

  Compute the ∞-norm of a vector.
- int mess_vector_norm (mess_vector x, mess_norm_t nrm_t, double ∗nrm)

  Compute the norm of a vector.
- int mess_vector_dot (mess_vector x, mess_vector y, double ∗dot)

  Compute the dot product of two vectors.
- int mess_vector_dotc (mess_vector x, mess_vector y, mess_double_cpx_t ∗dot)

  Compute the dot product of two vectors (complex).
- int mess_vector_dotu (mess_vector x, mess_vector y, mess_double_cpx_t ∗dot)

  Compute the dot product of two vectors (complex).
- int mess_vector_diffnorm (mess_vector x1, mess_vector x2, double ∗diff)

  Compute $\|x_1 - x_2\|_2$.
- int mess_vector_diffnorminf (mess_vector x1, mess_vector x2, double ∗diff)

  Compute $\|x_1 - x_2\|_\infty$.

Variables

- static mess_int_t __ONE = 1

10.591.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.591.2 Variable Documentation

10.591.2.1 mess_int_t __ONE = 1 [static]

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

10.592 offsets.c File Reference

Macros

- #define MESS_OFFSET(TYPE, FIELD, COUNTER, OFFSETPTR, FIELDNAMESPTR)

Functions

- int mess_int_t_size (void)

  Return size of mess_int_t.
- int mess_matrix_st_offset (int ∗size, int ∗offsets, char ∗∗fieldnames)

  Information about memory laout of mess_matrix_st.
- int mess_vector_st_offset (int ∗size, int ∗offsets, char ∗∗fieldnames)

  Information about memory laout of mess_vector_st.
- int mess_status_st_offset (int ∗size, int ∗offsets, char ∗∗fieldnames)

  Information about memory laout of mess_status_st.
- int mess_options_st_offset (int ∗size, int ∗offsets, char ∗∗fieldnames)

  Information about memory laout of mess_options_st.
- int mess_equation_st_offset (int ∗size, int ∗offsets, char ∗∗fieldnames)

  Information about memory laout of mess_equation_st.
- int mess_direct_st_offset (int ∗size, int ∗offsets, char ∗∗fieldnames)

  Information about memory laout of mess_direct_st.

10.592.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.592.2 Macro Definition Documentation

10.592.2.1 #define MESS_OFFSET(*TYPE, FIELD, COUNTER, OFFSETPTR, FIELDNAMESPTR*)

Value:

```
{                   \
    if(OFFSETPTR){                                                          \
        OFFSETPTR[COUNTER] = (int) offsetof(TYPE,FIELD);                    \
    }                                                                       \
    if(FIELDNAMESPTR){                                                      \
        FIELDNAMESPTR[COUNTER] = #FIELD;                                    \
    }                                                                       \
    COUNTER=COUNTER+1;                                                      \
}
```

10.593 ones.c File Reference

Functions

- int mess_matrix_one_value (mess_matrix matrix, double value)

  Set all entries of a matrix to one value.
- int mess_matrix_one_valuec (mess_matrix matrix, mess_double_cpx_t value)

  Set all entries of a matrix to one value (complex version).
- int mess_matrix_ones (mess_matrix matrix)

  Set all entries in a matrix to one.
- int mess_matrix_zeros (mess_matrix matrix)

  Set all entries in a matrix to zero.

10.593.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.594 ones.c File Reference

Functions

- int mess_vector_ones (mess_vector v)
  Fill a vector with ones.
- int mess_vector_zeros (mess_vector v)
  Fill a vector with zeros.

10.594.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.595 openmp.c File Reference

Macros

- #define _GNU_SOURCE 1

Functions

- void mess_openmp_info ()
  Display information about OpenMP.

10.595.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.595.2 Macro Definition Documentation

10.595.2.1 #define _GNU_SOURCE 1

10.596 options.c File Reference

Functions

- int mess_bt_options_init (mess_bt_options *opt)
  Initialize a mess_bt_options object.
- int mess_bt_options_clear (mess_bt_options *opt)
  Clear a mess_bt_options object.
- int mess_bt_options_print (mess_bt_options opt)
  Print a mess_bt_options object to stdout.
- const char * mess_bt_getsotypestr (unsigned short so_type)
  Get the string of the so_type value.

10.596.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.597 options.c File Reference

Functions

- int mess_options_init (mess_options *opt)

  Initialize LRCF-ADI options with default values.
- int mess_options_copy (mess_options src, mess_options dest)

  Copy a mess_options object to another one.
- int mess_options_clear (mess_options *opt)

  Clear a mess_options object.
- mess_int_t mess_options_memsize (mess_options opt)

  Return the size of a mess_options in bytes.
- int mess_options_print (mess_options opt)

  Display a mess_options object on standard output.
- int mess_options_set_proj_space (mess_options opt, mess_matrix proj_space)

  Copy the mess_matrix `proj_space`.
- int mess_options_unset_proj_space (mess_options opt)

  Delete the field `proj_space` from mess_options instance.
- int mess_options_set_K0 (mess_options opt, mess_matrix K0)

  Copy the mess_matrix `K0`.
- int mess_options_unset_K0 (mess_options opt)

  Delete the field `K0` from mess_options instance.
- int mess_options_set_W (mess_options opt, mess_matrix W)

  Copy the mess_matrix `W`.
- int mess_options_unset_W (mess_options opt)

  Delete the field `W` from mess_options instance.

10.597.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.598 options.c File Reference

Macros

- #define NPY_NO_DEPRECATED_API NPY_1_7_API_VERSION
- #define PY_ARRAY_UNIQUE_SYMBOL MESS_VECTOR_MATRIX_PYTHON_C_API
- #define NO_IMPORT_ARRAY
- #define ConvStringtoC(X) PyBytes_AsString(X)
- #define NPY_ARRAY_OWNDATA NPY_OWNDATA
- #define PyArray_ENABLEFLAGS(arr, flag) ((PyArrayObject*)(arr))->flags |= (flag)
- #define PyArray_CALL_ISSIGNED(x) PyArray_ISSIGNED((PyArrayObject*)(x))
- #define PyArray_CALL_ISUNSIGNED(x) PyArray_ISUNSIGNED((PyArrayObject*)(x))
- #define PyArray_CALL_DATA(x) PyArray_DATA((PyArrayObject*)(x))

Functions

- mess_options mess_options_from_python (PyObject *options)

  Parse the attributes for the lrcfadi algorithm from PyObject options argument.

10.598.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Nitin Prasad (nitin.prasad93@gmail.com, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.598.2 Macro Definition Documentation

10.598.2.1 #define ConvStringtoC(X) PyBytes_AsString(X)

10.598.2.2 #define NO_IMPORT_ARRAY

10.598.2.3 #define NPY_ARRAY_OWNDATA NPY_OWNDATA

10.598.2.4 #define NPY_NO_DEPRECATED_API NPY_1_7_API_VERSION

10.598.2.5 #define PY_ARRAY_UNIQUE_SYMBOL MESS_VECTOR_MATRIX_PYTHON_C_API

10.598.2.6 #define PyArray_CALL_DATA(x) PyArray_DATA((PyArrayObject*)(x))

10.598.2.7 #define PyArray_CALL_ISSIGNED(x) PyArray_ISSIGNED((PyArrayObject*)(x))

10.598.2.8 #define PyArray_CALL_ISUNSIGNED(x) PyArray_ISUNSIGNED((PyArrayObject*)(x))

10.598.2.9 #define PyArray_ENABLEFLAGS(*arr, flag*) ((PyArrayObject*)(arr))->flags |= (flag)

10.599 orth.c File Reference

Functions

- int mess_matrix_orth (mess_matrix A, mess_matrix Q)

  Compute an orthogonal basis for the range of a matrix.
- int mess_matrix_null (mess_matrix A, mess_matrix Z)

  Compute an orthogonal basis for the nullspace of a matrix.

10.599.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.600 output.txt File Reference

10.601 para.c File Reference

Data Structures

- struct parameter_mvp_call

Functions

- int mess_lrcfadi_lp_s (mess_double_cpx_t *p, mess_int_t lp, mess_double_cpx_t *set, mess_int_t lset, double *max_r, mess_int_t *ind)

  Compute the maximum magnitude of the rational ADI function.
- int mess_lrcfadi_lp_mnmx (mess_double_cpx_t *rw, mess_int_t lrw, mess_int_t adi_shifts_l0, mess_double_cpx_t *p, mess_int_t *lp)

  Compute ADI parameters solving the ADI minimax problem suboptimal.
- static int mvp_parameter (void *aux, mess_operation_t op, mess_vector in, mess_vector out)

  Operator times vector for the Arnoldi process.
- static int mvp_parameterinv (void *aux, mess_operation_t op, mess_vector in, mess_vector out)

  Inverse Operator times vector for the Arnoldi process.
- int mess_lrcfadi_getritzvalues (mess_equation eqn, mess_options opt, mess_vector ritz)

  Compute a set of Ritz values from a given equation.
- int mess_lrcfadi_parameter (mess_equation eqn, mess_options opt, mess_status stat)

  Compute parameters for the low-rank Cholesky factor ADI iteration.
- int mess_parameter (mess_equation eqn, mess_options opt, mess_status stat)

  Wrapper around mess_lrcfadi_parameter except that mess_equation.parameter is preferred (if available).

10.601.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.601.2 Function Documentation

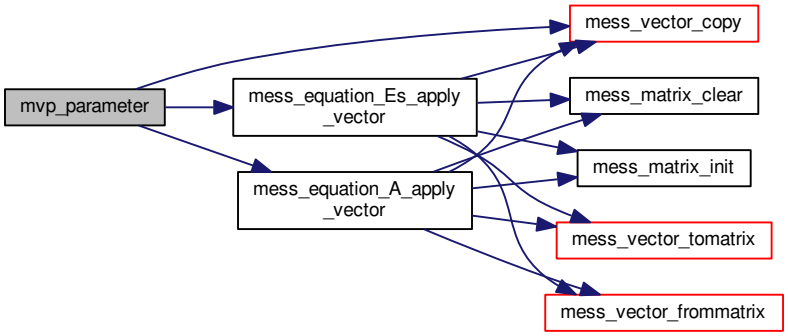
10.601.2.1 static int mvp_parameter (void * *aux*, mess_operation_t *op*, mess_vector *in*, mess_vector *out*) [static]

Parameters

in	*aux*	input pointer to the data of the operator
in	*op*	input (not used)
in	*in*	input vector
out	*out*	output vector

The mvp_parameter function computes out = Operator*in. The in is not guaranteed to be used read only.

Here is the call graph for this function:



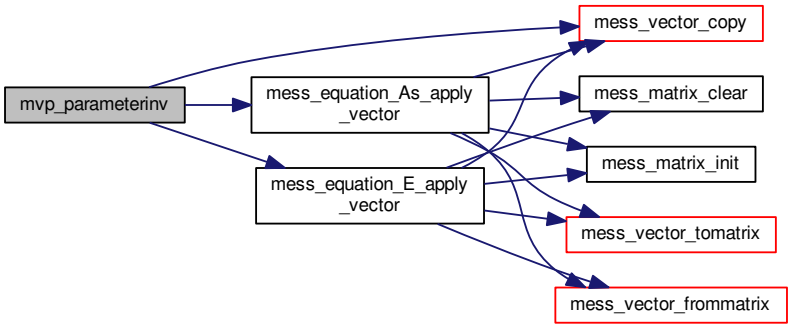
10.601.2.2 static int mvp_parameterinv (void ∗ *aux*, mess_operation_t *op*, mess_vector *in*, mess_vector *out*) [static]

Parameters

in	*aux*	input pointer to the data of the operator
in	*op*	input (not used)
in	*in*	input vector
out	*out*	output vector

The mvp_parameterinv function computes Operator∗out = in. The in is not guaranteed to be used read only.

Here is the call graph for this function:



10.602 perm.c File Reference

Functions

- static mess_int_t ∗ perm_inv (const mess_int_t ∗p, mess_int_t n)

Local function to invert a permutation.

- static int __mess_matrix_perm_csr (mess_matrix matrix, const mess_int_t *p, const mess_int_t *qinv)

  Permute a Compressed Sparse Row matrix.
- static int __mess_matrix_perm_csc (mess_matrix matrix, const mess_int_t *pinv, const mess_int_t *q)

  Permute a Compressed Sparse Column matrix (local function).
- static int __mess_matrix_perm_dense (mess_matrix matrix, mess_int_t *p, mess_int_t *qinv)

  Permute a dense matrix.
- int mess_matrix_perm (mess_matrix matrix, mess_int_t *p, mess_int_t *q)

  Permute a general matrix.
- int mess_matrix_permcopy (mess_matrix matrix, mess_int_t *p, mess_int_t *q, mess_matrix out)

  Copy and permute a matrix.
- static void ex_col_real (double *col1, double *col2, mess_int_t N)

  Exchange two vectors/columns (real valued).
- static void ex_col_cpx (mess_double_cpx_t *col1, mess_double_cpx_t *col2, mess_int_t N)

  Exchange two vectors/columns (complex valued).
- int mess_matrix_colperm (mess_matrix A, mess_int_t *perm)

  Permute the columns of a matrix inplace.
- int mess_matrix_colpermcopy (mess_matrix A, mess_int_t *perm, mess_matrix B)

  Permute the columns of a matrix while copying.

10.602.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.602.2 Function Documentation

10.602.2.1 static int __mess_matrix_perm_csc (mess_matrix *matrix*, const mess_int_t * *pinv*, const mess_int_t * *q*) [static]

Parameters

in,out	*matrix*	matrix to permute
in	*pinv*	input row permutation
in	*q*	input inverse column permutation

Returns

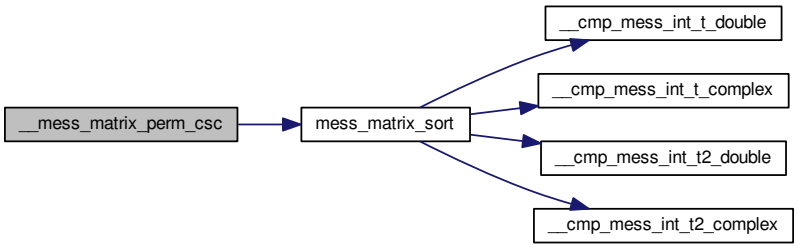
zero on success or a non zero error code otherwise

The __mess_matrix_perm_csc function permutes a Compressed Sparse Column matrix. The algorithm works not in-place.

Attention

Internal use only.

Here is the call graph for this function:



10.602.2.2 static int __mess_matrix_perm_csr (mess_matrix *matrix*, const mess_int_t * *p*, const mess_int_t * *qinv*) [static]

Parameters

in,out	*matrix*	matrix to permute
in	*p*	input row permutation
in	*qinv*	input inverse column permutation

Returns

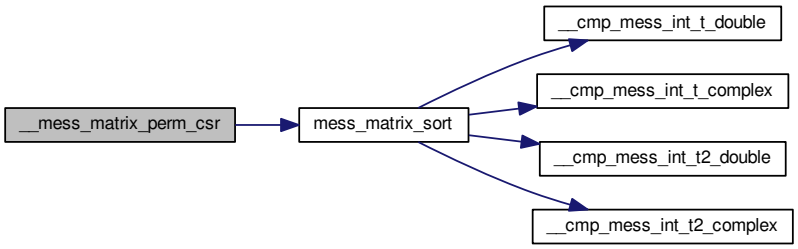
zero on success or a non zero error code otherwise

The __mess_matrix_perm_csr function permutes a Compressed Sparse Row matrix. The algorithm works not in-place.

Attention

Internal use only.

Here is the call graph for this function:



10.602.2.3 static int __mess_matrix_perm_dense (mess_matrix *matrix*, mess_int_t * *p*, mess_int_t * *qinv*) [static]

Parameters

in,out	*matrix*	matrix to permute
in	*p*	input row permutation
in	*qinv*	input column permutation

Returns

zero on success or a non zero error code otherwise

The __mess_matrix_perm_dense function permutes a dense matrix. The algorithm does not work in-place but using OpenMP

Attention

Internal use only.

10.602.2.4 static void ex_col_cpx (mess_double_cpx_t * *col1*, mess_double_cpx_t * *col2*, mess_int_t *N*) [static]

Parameters

in,out	*col1*	pointer to the first column
in,out	*col2*	pointer to the second column
in	*N*	input length of col1 and col2

Returns

Nothing.

The ex_col_real function exchanges two complex valued columns or arrays of length N.

Attention

Internal use only.

10.602.2.5 static void ex_col_real (double * *col1*, double * *col2*, mess_int_t *N*) [static]

Parameters

in,out	*col1*	pointer to the first column
in,out	*col2*	pointer to the second column
in	*N*	input length of col1 and col2

Returns

Nothing.

The ex_col_real function exchanges two real valued columns or arrays of length N.

Attention

Internal use only.

10.602.2.6 static mess_int_t* perm_inv (const mess_int_t * *p*, mess_int_t *n*) [static]

Parameters

in	*p*	input integer array to be permuted
in	*n*	input number of elements in the permutation

Returns

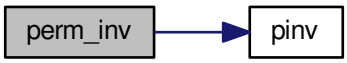
the permuted array, allocated by malloc

The perm_inv function inverts a permutation.

Attention

Interal use only.

Here is the call graph for this function:



10.603 perm.c File Reference

Functions

- static int __compare_complex (const void *pc1, const void *pc2)
  Compare two complex numbers for `qsort`.
- static int __compare_real (const void *pc1, const void *pc2)
  Compare two real numbers for `qsort`.
- static int __compare_realpart (const void *pc1, const void *pc2)
  Compare real part of two complex numbers for `qsort`.
- static int __compare_imagpart (const void *pc1, const void *pc2)
  Compare imaginary part of two complex numbers for qsort.
- int mess_vector_perm (mess_vector in, mess_int_t *perm, mess_vector out)
  Permute of a vector.
- int mess_vector_perm_inplace (mess_vector v, mess_int_t *perm)
  Permute a vector inplace.
- int mess_vector_iperm (mess_vector in, mess_int_t *iperm, mess_vector out)
  Inverse permutation of a vector.

- int mess_vector_iperm_split (mess_vector in, mess_int_t *iperm, mess_vector reout, mess_vector imout)

  Inverse permutation of a vector and spilts real and imaginary part.
- int mess_vector_iperm_combine (mess_vector rein, mess_vector imin, mess_int_t *iperm, mess_vector out)

  Inverse permutation of a vector and spilts real and imaginary part.
- int mess_vector_perm_split (mess_vector in, mess_int_t *perm, mess_vector reout, mess_vector imout)

  Permutate a vector and split in real and imaginary part.
- int mess_vector_perm_combine (mess_vector rein, mess_vector imin, mess_int_t *perm, mess_vector out)

  Inverse permutation of a vector and combination of real and imaginary part.
- int mess_vector_iperm_inplace (mess_vector v, mess_int_t *perm)

  Permute a vector inplace.
- int mess_vector_sort_realpart (mess_vector vect)

  Sort the components of a vector by real part in ascending order.
- int mess_vector_sort_imagpart (mess_vector vect)

  Sort the components of a vector by imaginary part in ascending order.
- int mess_vector_sort (mess_vector vect)

  Sort the components of a vector in ascending order.

10.603.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.603.2 Function Documentation

10.603.2.1 static int __compare_complex (const void * *pc1*, const void * *pc2*) [static]

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Parameters

in	*pc1*	input pointer to the first number
in	*pc2*	input pointer to the second number

Returns

-1, 0, 1 if `cabs(*pc1)` is smaller, equal, bigger than `cabs(*pc2)`

The __compare_complex function compares two complex numbers for the `stdlibc qsort` function.

Attention

Internal use only.

10.603.2.2 static int __compare_imagpart (const void * *pc1*, const void * *pc2*) [static]

Parameters

in	*pc1*	input pointer to the first number
in	*pc2*	input pointer to the second number

Returns

-1, 0, 1 if imaginary part of `*pc1` is smaller, equal, bigger than imaginary part of `*pc2`

The __compare_imagpart function compares two complex numbers by imaginary part for the `stdlibc qsort` function.

Attention

Internal use only.

10.603.2.3 static int __compare_real (const void * *pc1*, const void * *pc2*) [static]

Parameters

in	*pc1*	input pointer to the first number
in	*pc2*	input pointer to the second number

Returns

-1, 0, 1 if `*pc1` is smaller, equal, bigger than `*pc2`

The __compare_real function compares two real numbers for the `stdlibc qsort` function.

Attention

Internal use only.

10.603.2.4 static int __compare_realpart (const void * *pc1*, const void * *pc2*) [static]

Parameters

in	*pc1*	input pointer to the first number
in	*pc2*	input pointer to the second number

Returns

-1, 0, 1 if real part of `*pc1` is smaller, equal, bigger than real part of `*pc2`

The __compare_realpart function compares two complex numbers by real part for the `stdlibc qsort` function.

Attention

Internal use only.

10.604 pinv.c File Reference

Functions

- int mess_matrix_pinv (mess_matrix A, mess_matrix Pinv)

  Compute the pseudoinverse of a dense matrix A.

10.604.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.605 plot_scriptExporter.c File Reference

Functions

- int mess_plotExport_init (mess_plotExport *p, char *title, mess_plot_axis_scale_t xscale, mess_plot_axis↩_scale_t yscale, char *xLabel, char *yLabel, int data)

  Initialize all data structures and variables.
- int mess_plotExport_clear (mess_plotExport *pl)

  Clear all data in a plot.
- int mess_plotExport_clearData (mess_plotExport p, int data)

  Clear all data of a data element in a plot.
- int mess_plotExport_addData (mess_plotExport p, int data, double x, double y)

  Add new data values to a plot.
- int mess_plotExport_setLegendPos (mess_plotExport p, char *orientation)

  Set position of the legend.
- int mess_plotExport_setLabel (mess_plotExport p, int data, char *label)

  Set label of a data element.
- int mess_plotExport_setColor (mess_plotExport p, int data, char *color)

  Set the color of a plot.
- int mess_plotExport_setType (mess_plotExport p, int data, char *type)

  Set the style of a plot.
- int mess_plotExport_save (mess_plotExport p, const char *path)

  Save all data to a .dat file.
- static int mess_plotExport_createGnuScript_int (mess_plotExport p, const char *path)

  Create a GNU plot script file.
- int mess_plotExport_createGnuScript (mess_plotExport p)

  Create a GNU plot script file.
- int mess_plotExport_createGnuScript_filename (mess_plotExport p, const char *path)

  Create a GNU plot script file.
- static int mess_plotExport_createTikzScript_int (mess_plotExport p, const char *path)

  Generate a TikZ-diagram and store it in a .tex-file.
- int mess_plotExport_createTikzScript (mess_plotExport p)

  Generate a TikZ-diagram and store it in a .tex-file.
- int mess_plotExport_createTikzScript_filename (mess_plotExport p, const char *path)

  Generate a TikZ-diagram and store it in a .tex-file.

10.605.1 Detailed Description

Author

Majan Weiss

10.605.2 Function Documentation

10.605.2.1 static int mess_plotExport_createGnuScript_int (mess_plotExport *p*, const char * *path*) [static]

Parameters

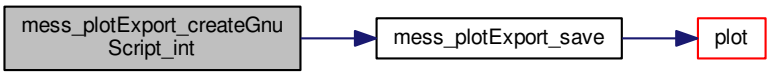
in	*p*	input plot
in	*path*	input destination folder to store the GNU plot script

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_createGnuScript_int function will use all known data from the plot to create a script file. This script file will be saved as a _script.gnu file with the plots title as a prefix.

Here is the call graph for this function:



10.605.2.2 static int mess_plotExport_createTikzScript_int (mess_plotExport *p*, const char * *path*) [static]

Parameters

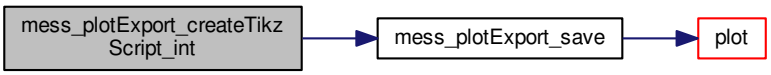
in	*p*	input plot with data
in	*path*	input destination folder to store the TikZ Script

Returns

zero on success or a non-zero error value otherwise

The mess_plotExport_createTikzScript_int evaluates information stored in a plot and creates a TikZ-picture. This will be saved as a .tex-file and can be included to a TeX-document.

Here is the call graph for this function:



10.606 plot_scriptExporter.h File Reference

Data Structures

- struct __mess_plotExportData

  Structure for one data element in a plot.
- struct __mess_plotterScript

  Structure to hold a complete plotter.

Typedefs

- typedef struct __mess_plotExportData mess_plotExportData

  Structure for one data element in a plot.
- typedef struct __mess_plotterScript * mess_plotExport

  Type definition for the plotter structure.

Enumerations

Functions

- int mess_plotExport_init (mess_plotExport *p, char *title, mess_plot_axis_scale_t xscale, mess_plot_axis↩_scale_t yscale, char *xLabel, char *yLabel, int data)

  Initialize all data structures and variables.
- int mess_plotExport_addData (mess_plotExport p, int data, double x, double y)

  Add new data values to a plot.
- int mess_plotExport_clearData (mess_plotExport p, int data)

  Clear all data of a data element in a plot.
- int mess_plotExport_setLegendPos (mess_plotExport p, char *orientation)

  Set position of the legend.
- int mess_plotExport_setLabel (mess_plotExport p, int data, char *label)

  Set label of a data element.
- int mess_plotExport_setColor (mess_plotExport p, int data, char *color)

  Set the color of a plot.
- int mess_plotExport_setType (mess_plotExport p, int data, char *type)

  Set the style of a plot.
- int mess_plotExport_save (mess_plotExport p, const char *path)

  Save all data to a .dat file.
- int mess_plotExport_createGnuScript (mess_plotExport p)

Create a GNU plot script file.

- int mess_plotExport_createGnuScript_filename (mess_plotExport p, const char *path)

  Create a GNU plot script file.
- int mess_plotExport_createTikzScript (mess_plotExport p)

  Generate a TikZ-diagram and store it in a .tex-file.
- int mess_plotExport_createTikzScript_filename (mess_plotExport p, const char *path)

  Generate a TikZ-diagram and store it in a .tex-file.
- int mess_plotExport_clear (mess_plotExport *plot)

  Clear all data in a plot.

10.606.1 Detailed Description

Author

Majan Weiss

10.607 plot_x11.h File Reference

Data Structures

- struct __mess_plotdata

  Structure for one data element in a plot.
- struct __mess_plotter

  Structure to hold a complete plotter.

Typedefs

- typedef struct __mess_plotdata mess_plotdata

  Structure for one data element in a plot.
- typedef struct __mess_plotter * mess_plot

  Type definition for the plotter structure.

Functions

- int mess_plot_create (mess_plot *p, mess_int_t width, mess_int_t height, char *title, int xscale, int yscale, int data)

  Create a plotting window.
- int mess_plot_addData (mess_plot p, int data, double x, double y)

  Add new data values to a plot.
- int mess_plot_clearData (mess_plot p, int data)

  Clear all data of a data element in a plot.
- int mess_plot_setLabel (mess_plot p, int data, char *label)

  Set label of a data element.
- int mess_plot_setColor (mess_plot p, int data, char *color)

  Set color of a data element.
- int mess_plot_save (mess_plot p, char *filename)

  Save a plot to a xpm image.
- int mess_plot_update (mess_plot p)

  Update / redraw a plot.
- int mess_plot_close (mess_plot *p)

  Close a plotting window and clean up all internal data.

10.607.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.608 precond.c File Reference

Functions

- int mess_precond_init (mess_precond *precond)

  Initialize a mess_precond object.
- int mess_precond_clear (mess_precond *precond)

  Clean up the mess_precond structure to use it again.

10.608.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.609 print.c File Reference

Functions

- int mess_print_format_select (mess_print_format_t p)

  Change the default behaviour of printing.
- int mess_print_format_double (double d)

  Print a double value using MSG_PRINT.
- int mess_print_format_double_cpx (mess_double_cpx_t cpx)

  Print a mess_double_cpx_t value using MSG_PRINT.

Variables

- static mess_print_format_t print_format = MESS_PRINT_FORMAT_SHORT
- static pthread_mutex_t print_format_mutex = PTHREAD_MUTEX_INITIALIZER

10.609.1 Variable Documentation

10.609.1.1 mess_print_format_t print_format = MESS_PRINT_FORMAT_SHORT [static]

10.609.1.2 pthread_mutex_t print_format_mutex = PTHREAD_MUTEX_INITIALIZER [static]

10.610 profile.txt File Reference

10.611 proj_sym.c File Reference

Functions

- int mess_matrix_proj_sym (mess_matrix matrix)

  Get projection of a matrix on symmetric matrix space.

10.611.1 Detailed Description

Author

Robert Dykstra (dykstra@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.612 project.c File Reference

Functions

- int mess_dynsys_project_to_lti (mess_dynsys sys, mess_matrix V, mess_matrix W, mess_dynsys red)

  Project a dynamical system (LTI or GLTI) to a LTI system.
- int mess_dynsys_project_to_glti (mess_dynsys sys, mess_matrix V, mess_matrix W, mess_dynsys red)

  Project a GLTI system to a GLTI system.
- int mess_dynsys_project_2nd_to_1st (mess_dynsys sys2nd, mess_matrix V, mess_matrix W, mess_dynsys sys1st)

  Project a second order system to a first order system.
- int mess_dynsys_project_2nd_to_2nd (mess_dynsys sys, mess_matrix V, mess_matrix W, mess_dynsys red)

  Project a second order system to a second order system.

10.612.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.613 pymess.c File Reference

Data Structures

- struct module_state

Macros

- #define NPY_NO_DEPRECATED_API NPY_1_7_API_VERSION
- #define PY_ARRAY_UNIQUE_SYMBOL MESS_VECTOR_MATRIX_PYTHON_C_API
- #define ConvStringtoC(X) PyBytes_AsString(X)
- #define GETSTATE(m) (&_state)
- #define INITERROR return

Functions

- static void init_numpy (void)
- PyObject * pymess_mess_is_debug (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_zlib (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_bzip2 (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_umfpack (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_amd (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_colamd (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_cholmod (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_csparse (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_superlu (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_mklpardiso (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_arpack (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_matio (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_openmp (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_mess64 (PyObject *self, PyObject *args)
- PyObject * pymess_mess_git_id (PyObject *self, PyObject *args)

  Call the mess_git_id function and returns git id, which was used to configure the C-M.E.S.S. build.
- PyObject * pymess_mess_git_branch (PyObject *self, PyObject *args)

  Call the mess_git_branch function and returns git branch, which was used to configure the C-M.E.S.S. build.
- void init_c_interface (void)

Variables

- static struct module_state _state
- static PyMethodDef interface_python_methods [ ]

  Function which are callable from `Python`*. Edit below function if you need to add more functions that you wish to be called from* `Python` *Avoid editing any other piece of code in the Initialisation section.*

10.613.1 Detailed Description

Author

Nitin Prasad (`nitin.prasad93@gmail.com`, MPI Magdeburg, CSC)
Maximilian Behr (`behr@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.614 pymess.h File Reference

Macros

- #define PYCMESS_ERROR(ret, message) { if ((ret) != 0) { PyErr_SetString(PyExc_RuntimeError,message); return NULL; } }
- #define MESS_Py_XDECREF(...)
- #define PY_SET_LONG(obj, name, value)
- #define PY_SET_DOUBLE(obj, name, value)
- #define PY_SET_VECTOR(obj, name, value)
- #define PY_SET_BOOL(obj, name, value)
- #define PY_GET_LONG(dest, from, name)
- #define PY_GET_DOUBLE(dest, from, name)
- #define PY_GET_VECTOR(dest, from, name)
- #define PY_GET_MATRIX(dest, from, name)
- #define PYMESS_READ_MATRIX(mem, cmatrix, pymatrix, name, errormark)

Functions

- int pymess_is_reimplemented (PyObject *base, PyObject *instance, const char *attr)

  Check if a function is reimplemented by a derived object or equal to the one contained in the base class.
- mess_equation eqn_conv_lyap (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_lyap_so1 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati_so1 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_lyap_so2 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati_so2 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_lyap_dae1 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati_dae1 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_lyap_dae2 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati_dae2 (PyObject *obj, mess_freelist mem)
- PyObject * pymess_test_matrix (PyObject *self, PyObject *args)

  Test functions that can be called from `Python` *to ensure that all the conversions are working.*
- PyObject * pymess_test_vector (PyObject *self, PyObject *args)

  Test functions that can be called from `Python` *to ensure that all the conversions are working.*
- PyObject * pymess_test_status (PyObject *self, PyObject *args)
- PyObject * pymess_test_equation (PyObject *self, PyObject *args)
- PyObject * pymess_lrnm (PyObject *self, PyObject *args)
- PyObject * pymess_lradi (PyObject *self, PyObject *args)
- PyObject * pymess_dense_nm_gmpare (PyObject *self, PyObject *args)

  Wrapper around the interface mess_dense_nm_gmpare for use with `Python`.
- PyObject * pymess_lyap (PyObject *self, PyObject *args)

  Wrapper around the easy to use interface mess_lyap for use with `Python`.
- PyObject * pymess_sylvester_sparsedense (PyObject *self, PyObject *args)

  Wrapper around the interface mess_direct_create_sylvester_sparsedense for use with `Python`.
- PyObject * pymess_care (PyObject *self, PyObject *args)

  Wrapper around the easy to use interface mess_care for use with `Python`.
- PyObject * pymess_glyap (PyObject *self, PyObject *args)

  Interface to mess_glyap for use with `Python`.
- PyObject * pymess_gstein (PyObject *self, PyObject *args)

  Interface to mess_gstein for use with `Python`.
- PyObject * pymess_eps (PyObject *self, PyObject *args)

  Return machine epsilon. The pymess_eps is callable as `eps` *and calls internally the mess_eps function and returns the machine epsilon.*
- PyObject * pymess_mess_version (PyObject *self, PyObject *args)

  Call the mess_version function. The pymess_mess_version is callable as `mess_version` *and calls internally the mess_version function.*
- PyObject * pymess_mess_version_verbose (PyObject *self, PyObject *args)

  Call the mess_version_verbose function. The pymess_mess_version_verbose is callable as `mess_version_verbose` *and calls internally the mess_version_verbose function.*
- PyObject * pymess_mess_version_major (PyObject *self, PyObject *args)

  Call the mess_version_major function and returns the major version number.
- PyObject * pymess_mess_version_minor (PyObject *self, PyObject *args)

  Call the mess_version_minor function and returns the minor version number.
- PyObject * pymess_mess_version_patch (PyObject *self, PyObject *args)

  Call the mess_version_patch function and returns the patch version number.
- PyObject * pymess_mess_is_debug (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_zlib (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_bzip2 (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_umfpack (PyObject *self, PyObject *args)

- PyObject * pymess_mess_have_amd (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_colamd (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_cholmod (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_csparse (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_superlu (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_mklpardiso (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_arpack (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_matio (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_openmp (PyObject *self, PyObject *args)
- PyObject * pymess_mess_have_mess64 (PyObject *self, PyObject *args)
- PyObject * pymess_direct_select (PyObject *self, PyObject *args)

  Wrapper around mess_direct_lu_select for use with `Python`.
- PyObject * pymess_direct_chol_select (PyObject *self, PyObject *args)

  Wrapper around mess_direct_chol_select for use with `Python`.
- PyObject * pymess_multidirect_select (PyObject *self, PyObject *args)

  Wrapper around mess_multidirect_select for use with `Python`.
- mess_matrix matrix_to_c (PyObject *data)

  Converts `Python` *matrix to mess_matrix.*
- PyObject * matrix_to_python (mess_matrix c_matrix)

  Converts mess_matrix to `Python` *matrix.*
- mess_vector vector_to_c (PyObject *data)

  Converts `Python` *vector to mess_vector.*
- PyObject * vector_to_python (mess_vector c_vector, int copy)

  Converts mess_vector to `Python` *vector.*
- mess_options mess_options_from_python (PyObject *options)

  Parse the attributes for the lrcfadi algorithm from PyObject options argument.
- PyObject * mess_status_to_python (mess_status stat)

  Convert a mess_status object from C to `Python`.
- PyObject * pymess_set_errorlevel (PyObject *self, PyObject *args)

  Wrapper around mess_set_errorlevel for use with `Python`.
- mess_equation mess_equation_from_python (PyObject *obj, mess_freelist mem, mess_equation_t eqntype, int *need_callback)

10.615 rail_371_c60.C File Reference

10.616 rand.c File Reference

Macros

- #define IDX(i, j, LDA) ((j)*(LDA)+(i))

Functions

- static double __drand ()

  Generate a positive random floating point number between -1 and 1.
- int mess_matrix_rand_init (mess_int_t *seed)

  Init random number generator with given seed.
- int mess_matrix_rand_dense (mess_matrix mat, mess_int_t rows, mess_int_t cols, mess_datatype_t dt)

  Generate a MESS_REAL MESS_COMPLEX random dense matrix.

- int mess_matrix_rand_dense_uniform (mess_matrix mat, mess_int_t *seed, mess_int_t rows, mess_int_t cols)

  Generate a random MESS_REAL dense matrix with a uniform value distribution $(-1, 1)$.
- int mess_matrix_rand_csr (mess_matrix mat, mess_int_t rows, mess_int_t cols, double p, mess_datatype_t dt)

  Generate a MESS_REAL/MESS_COMPLEX random MESS_CSR matrix.
- int mess_matrix_rand_csc (mess_matrix mat, mess_int_t rows, mess_int_t cols, double p, mess_datatype_t dt)

  Generate a MESS_REAL/MESS_COMPLEX random MESS_CSC matrix.
- int mess_matrix_rand_coord (mess_matrix mat, mess_int_t rows, mess_int_t cols, double p, mess_datatype_t dt)

  Generate a MESS_REAL/MESS_COMPLEX random COORD matrix.
- int mess_matrix_rand (mess_matrix mat, mess_int_t rows, mess_int_t cols, mess_storage_t storetype, mess_datatype_t dt, double p)

  Wrapper around mess_matrix_rand_csr, mess_matrix_rand_csc, mess_matrix_rand_dense and mess_matrix_rand_coord.

10.616.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.616.2 Macro Definition Documentation

10.616.2.1 #define IDX(*i, j,* LDA) ((j)*(LDA)+(i))

10.616.3 Function Documentation

10.616.3.1 static double __drand ( ) [static]

Returns

A positive random number between -1 and 1

The __drand function generates an uniformly distributed random number between -1 and 1.

Attention

mess_matrix_rand_coord depends on the scaling to [-1;1].

10.617 rand.c File Reference

Functions

- static double __drand ()

  Generate a positive random floating point number between -1 and 1.
- int mess_vector_rand_init (mess_int_t *seed)

  Init random number generator with given seed.
- int mess_vector_rand (mess_vector v)

  Fill a vector with random numbers.

10.617.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.617.2 Function Documentation

10.617.2.1 static double __drand ( ) [static]

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

Returns

A positive random number between -1 and 1

The __drand function generates an uniformly distributed random number between -1 and 1.

10.618 rank.c File Reference

Functions

- int mess_matrix_rank (mess_matrix A, mess_int_t *rank)
  Estimate the rank of a matrix.

10.618.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.619 rc2norm.c File Reference

Data Structures

- struct rc2data

Macros

- #define IT_MAX 20

Functions

- static int rc2mvp (void *data, mess_operation_t op, mess_vector x, mess_vector y)
- int mess_lrcfadi_fac2diff (mess_matrix Zold, mess_matrix Z, double *chg)
  Compute 2-norm difference between two factored solutions of a matrix equation.

10.619.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

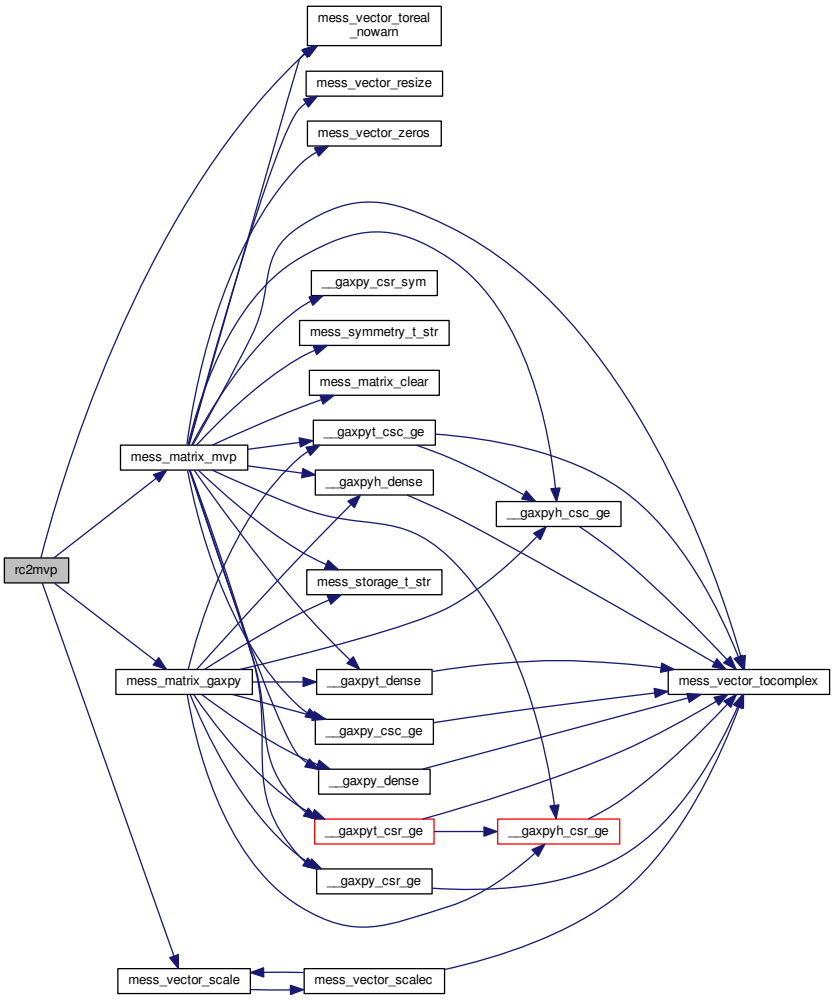
10.619.2 Macro Definition Documentation

10.619.2.1 #define IT_MAX 20

10.619.3 Function Documentation

10.619.3.1 static int rc2mvp (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



10.620 rcFnorm.c File Reference

Functions

- int mess_lrcfadi_facFdiff (mess_matrix Zold, mess_matrix Z, double *chg)

  Compute the relative Frobenius-norm difference between two factored solutions of a matrix equation.

10.620.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file contains various functions to perform matrix-matrix additions and similar problems.

10.621 read.c File Reference

Macros

- #define LINE_LENGTH 1025
- #define TOKEN_LENGTH 30
- #define SET_LAST(array, len, value) (array)[(len)-1] = (value)
- #define MM_BANNER "%%MatrixMarket"
- #define MM_STR_MATRIX "matrix"
- #define MM_STR_COORD "coordinate"
- #define MM_STR_ARR "array"
- #define MM_STR_REAL "real"
- #define MM_STR_INT "integer"
- #define MM_STR_CPX "complex"
- #define MM_STR_PAT "pattern"
- #define MM_STR_GE "general"
- #define MM_STR_SYM "symmetric"
- #define MM_STR_SKSYM "skew-symmetric"
- #define MM_STR_HER "hermitian"

Functions

- static void _lowercase (char *str)

  Convert a string to lower case.
- static void _uppercase (char *str)

  Convert a string to upper case.
- static char * _substr (const char *S, const int pos, const int len)

  Copy a sub-string out of a given one.
- static int _ParseIfmt (char *fmt, int *perline, int *width)
- static int _ParseRfmt (char *fmt, int *perline, int *width, int *prec, int *flag)
- static int __mm_read_info (csc_io_file_t *file, mess_matrix matrix_info)

  Read the header of a Matrix Market file. (internal version)
- static int __mm_read_data (csc_io_file_t *f, mess_matrix matrix)

  Read the data part of a `MatrixMarket` *matrix.*

- static int __hb_read_info (csc_io_file_t *in_file, char *Type, int *Nrow, int *Ncol, int *Nnzero, int *Nrhs, char *Ptrfmt, char *Indfmt, char *Valfmt, char *Rhsfmt, int *Ptrcrd, int *Indcrd, int *Valcrd, int *Rhscrd, char *Rhstype)

  Read the header information out of a Harwell-Boeing matrix.
- static int __hb_read_data (csc_io_file_t *in_file, char *Type, mess_int_t Nrow, mess_int_t Ncol, mess_int_t Nnzero, mess_int_t Nrhs, char *Ptrfmt, char *Indfmt, char *Valfmt, char *Rhsfmt, mess_int_t Ptrcrd, mess_int_t Indcrd, mess_int_t Valcrd, mess_int_t Rhscrd, char *Rhstype, mess_int_t *colptr, mess_int_t *rowind, double *val)
- static int __hb_read (csc_io_file_t *in_file, mess_matrix matrix)
- int mess_matrix_read (const char *filename, mess_matrix matrix)

  Read a matrix from a file into a mess_matrix structure.
- int mess_matrix_read_formated (const char *filename, mess_matrix matrix, mess_storage_t format)

  Read a matrix from a file and convert it to the desired format.

10.621.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.621.2 Macro Definition Documentation

10.621.2.1 #define LINE_LENGTH 1025

10.621.2.2 #define MM_BANNER "%%MatrixMarket"

10.621.2.3 #define MM_STR_ARR "array"

10.621.2.4 #define MM_STR_COORD "coordinate"

10.621.2.5 #define MM_STR_CPX "complex"

10.621.2.6 #define MM_STR_GE "general"

10.621.2.7 #define MM_STR_HER "hermitian"

10.621.2.8 #define MM_STR_INT "integer"

10.621.2.9 #define MM_STR_MATRIX "matrix"

10.621.2.10 #define MM_STR_PAT "pattern"

10.621.2.11 #define MM_STR_REAL "real"

10.621.2.12 #define MM_STR_SKSYM "skew-symmetric"

10.621.2.13 #define MM_STR_SYM "symmetric"

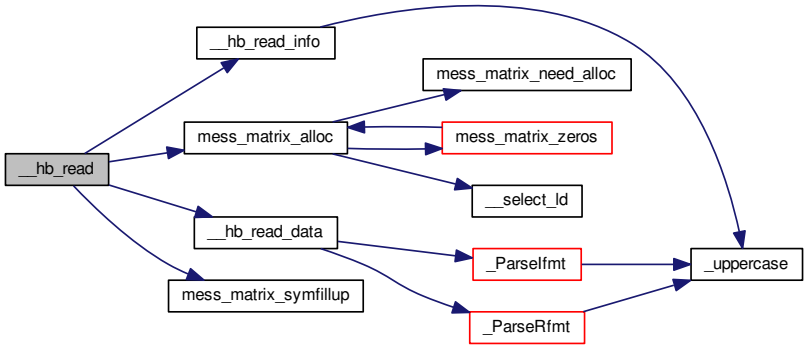
10.621.2.14 #define SET_LAST(*array, len, value*) (array)[(len)-1] = (value)

10.621.2.15 #define TOKEN_LENGTH 30

10.621.3 Function Documentation

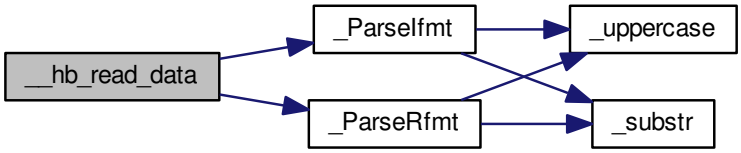
10.621.3.1 static int __hb_read (csc_io_file_t * *in_file*, mess_matrix *matrix*) [static]

Here is the call graph for this function:



10.621.3.2 static int __hb_read_data (csc_io_file_t * *in_file*, char * *Type*, mess_int_t *Nrow*, mess_int_t *Ncol*, mess_int_t *Nnzero*, mess_int_t *Nrhs*, char * *Ptrfmt*, char * *Indfmt*, char * *Valfmt*, char * *Rhsfmt*, mess_int_t *Ptrcrd*, mess_int_t *Indcrd*, mess_int_t *Valcrd*, mess_int_t *Rhscrd*, char * *Rhstype*, mess_int_t * *colptr*, mess_int_t * *rowind*, double * *val*) [static]

Here is the call graph for this function:



10.621.3.3 static int __hb_read_info (csc_io_file_t * *in_file*, char * *Type*, int * *Nrow*, int * *Ncol*, int * *Nnzero*, int * *Nrhs*, char * *Ptrfmt*, char * *Indfmt*, char * *Valfmt*, char * *Rhsfmt*, int * *Ptrcrd*, int * *Indcrd*, int * *Valcrd*, int * *Rhscrd*, char * *Rhstype*) [static]

Parameters

in	*in_file*	input file to read from
	Type	
	Nrow	

Parameters

	Ncol	
	Nnzero	
	Nrhs	
	Ptrfmt	
	Indfmt	
	Valfmt	
	Rhsfmt	
	Ptrcrd	
	Indcrd	
	Valcrd	
	Rhscrd	
	Rhstype	

Returns

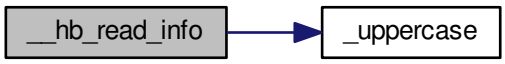
zero on success or a non zero error code otherwise

The __hb_read_info function reads the header information out of a Harwell-Boeing matrix.

Attention

Internal use only.

Here is the call graph for this function:



10.621.3.4 static int __mm_read_data (csc_io_file_t * *f*, mess_matrix *matrix*) [static]

Parameters

in	*f*	input File to read from.
in,out	*matrix*	Matrix object to read.

Returns

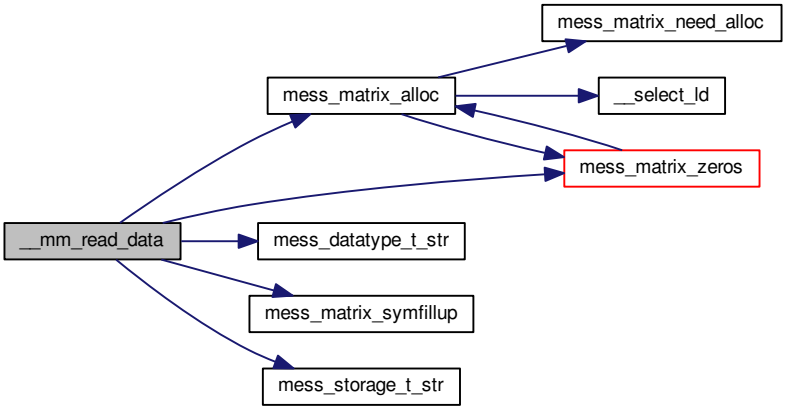
zero on success or a non zero error code otherwise

The __mm_read_data function reads the data part out of a `MatrixMarket` file. The `matrix` object has to be filled up with the values read by __mm_read_info.

Attention

Internal use only.

Here is the call graph for this function:



10.621.3.5 static int __mm_read_info (csc_io_file_t * *file*, mess_matrix *matrix_info*) [static]

Parameters

in	*file*	input filepointer
out	*matrix_info*	mess_matrix where the information should be saved

Returns

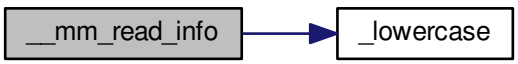
zero on success, otherwise a non zero error code.

The __mm_read_info function reads the header of a `MatrixMarket` file and writes the basic information to matrix_info.

Attention

Internal use only.

Here is the call graph for this function:



10.621.3.6 static void _lowercase (char * *str*) [static]

Parameters

in,out	*str*	input/output string to be converted to lower case

Returns

Nothing.

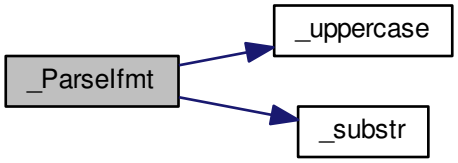
Convert a string to lowercase. This is done inplace.

Attention

Internal use only.

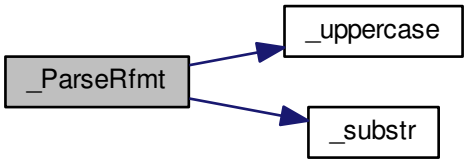
10.621.3.7 static int _ParseIfmt (char * *fmt*, int * *perline*, int * *width*) [static]

Here is the call graph for this function:



10.621.3.8 static int _ParseRfmt (char * *fmt*, int * *perline*, int * *width*, int * *prec*, int * *flag*) [static]

Here is the call graph for this function:



10.621.3.9 static char* _substr (const char * *S*, const int *pos*, const int *len*) [static]

Parameters

in	*S*	input Input string
in	*pos*	input Start position of the requested sub-string
in	*len*	input Length of the requested sub-string

Returns

A copy of the sub-string or `NULL` in case of an error.

The _substr function copies a sub-string beginning at `pos` with `len` characters from `S` to a newly allocated string. The returned string must be freed after it is not longer used.

Attention

Internal use only.

10.621.3.10 static void _uppercase (char * *str*) [static]

Parameters

in,out	*str*	inout/output String to be converted to upper case

Returns

Nothing.

Convert a string to upper case. This is done inplace.

Attention

Internal use only.

10.622 real_complex.c File Reference

Functions

- double * mess_vector_valuereal (mess_vector vec, mess_int_t index)

  Get a real value from a vector.
- mess_double_cpx_t * mess_vector_valuecomplex (mess_vector vec, mess_int_t index)

  Get a complex value from a vector.
- int mess_vector_realpart (mess_vector in, mess_vector out)

  Get the real part of a vector.
- int mess_vector_imagpart (mess_vector in, mess_vector out)

  Get the imaginary part of a vector.
- int mess_vector_complex_from_parts (mess_vector xr, mess_vector xc, mess_vector x)

  Generate a complex vector from real and imaginary part.
- int mess_vector_isreal (mess_vector v)

  Check if a vector is real or if the imaginary part is 0.
- int mess_vector_conj (mess_vector vector)

  Complex conjugatation of a vector.

10.622.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.622.2 Function Documentation

10.622.2.1 int mess_vector_isreal (mess_vector *v*)

Parameters

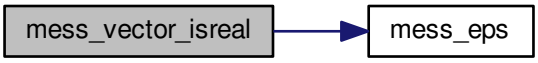
in	*v*	input vector

Returns

one if v is real, otherwise zero

The mess_vector_isreal function checks if a vector is real or if the imaginary part is 0.

Here is the call graph for this function:



10.623 realimag.c File Reference

Functions

- int mess_matrix_realpart (mess_matrix matrix, mess_matrix realpart)
  Get the real part of a matrix.
- int mess_matrix_imagpart (mess_matrix matrix, mess_matrix imagpart)
  Get the imaginary part of a matrix.
- int mess_matrix_conj (mess_matrix A)
  Conjugate a matrix.
- int mess_matrix_complex_from_parts (mess_matrix xr, mess_matrix xc, mess_matrix x)
  Create a complex matrix from real and imaginary parts.

10.623.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.624 reimplemented.c File Reference

Functions

- int pymess_is_reimplemented (PyObject ∗base, PyObject ∗instance, const char ∗attr)

  Check if a function is reimplemented by a derived object or equal to the one contained in the base class.

10.624.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.625 res2.c File Reference

Functions

- int mess_direct_res2 (mess_operation_t op, mess_matrix A, mess_vector x, mess_vector b, double ∗resid, double ∗relresid)

  Compute the (relative) residual of $Ax = b$.
- int mess_direct_res2m (mess_operation_t op, mess_matrix A, mess_matrix x, mess_matrix b, double ∗resid, double ∗relresid)

  Compute the (relative) residual of $Ax = b$.

10.625.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.626 res2.c File Reference

Data Structures

- struct res2_data_st
- struct res2nm_data_st

Macros

- #define VECTOR_SCALE(factor, vector)
- #define VECTOR_AXPY(factor, v1, v2)
- #define MATRIX_ADD(alpha, A, beta, B)
- #define IT_MAX 20

Typedefs

- typedef struct res2_data_st res2_data
- typedef struct res2nm_data_st res2nm_data

Functions

- static int res2_mvp (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)
- static int res2_mvpt (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)
- int res2_mvpBK (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)

  Computes a matrix-vector product of a Lyapunov Equation and a given vector.
- static int res2g_mvp (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)
- static int res2gt_mvp (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)
- int res2g_mvpBK (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)

  Computes a matrix-vector product of a generalized Lyapunov Equation and a given vector.
- int mess_lrcfadi_res2 (mess_matrix F, mess_matrix G, mess_matrix Z, double ∗nrm)

  Compute 2-norm of the Lyapunov residual via an Arnoldi process.
- int mess_lrcfadi_res2t (mess_matrix F, mess_matrix G, mess_matrix Z, double ∗nrm)

  Compute 2 norm of the Lyapunov residual (transposed version).
- int mess_lrcfadi_res2BK (mess_matrix F, mess_matrix G, mess_matrix Z, mess_matrix B, mess_matrix K, double ∗nrm)

  Compute the 2-norm of the Lyapunov residual (BK version).
- int mess_lrcfadi_res2g (mess_matrix F, mess_matrix M, mess_matrix G, mess_matrix Z, double ∗nrm)

  Compute the 2-norm of the generalized Lyapunov residual.
- int mess_lrcfadi_res2gt (mess_matrix F, mess_matrix M, mess_matrix G, mess_matrix Z, double ∗nrm)

  Compute the 2-norm of the generalized Lyapunov residual (transposed version).
- int mess_lrcfadi_res2gBK (mess_matrix F, mess_matrix M, mess_matrix G, mess_matrix Z, mess_matrix B, mess_matrix K, double ∗nrm)

  Compute the 2-norm of the generalized Lyapunov residual (BK-Version).
- static int res2nm_mvp (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)
- static int res2nmg_mvp (void ∗data, mess_operation_t op, mess_vector x, mess_vector y)
- int mess_lrcfadi_res2nm (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix Z, double ∗nrm)

  Compute the 2-norm of the Riccati residual.
- int mess_lrcfadi_res2nmg (mess_matrix A, mess_matrix B, mess_matrix C, mess_matrix M, mess_matrix Z, double ∗nrm)

  Compute the 2-norm of the generalized Riccati residual.

10.626.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.626.2 Macro Definition Documentation

10.626.2.1 #define IT_MAX 20

10.626.2.2 #define MATRIX_ADD(alpha, A, beta, B)

Value:

```
if ( MESS_IS_COMPLEX((B))){ \
    mess_matrix_addc((alpha),(A),(beta),(B));\
} else { \
    mess_matrix_add((alpha),(A),(beta), (B));\
}
```

10.626.2.3 #define VECTOR_AXPY(*factor, v1, v2*)

Value:

```
if (MESS_IS_REAL(v1)) {\
    mess_vector_axpy((factor),(v1),(v2)); \
} else { \
    mess_vector_axpyc((factor),(v1),(v2)); \
}
```

10.626.2.4 #define VECTOR_SCALE(*factor, vector*)

Value:

```
if (MESS_IS_REAL((vector))) { \
    mess_vector_scale((factor), (vector));\
} else { \
    mess_vector_scalec((factor), (vector));\
}
```

10.626.3 Typedef Documentation

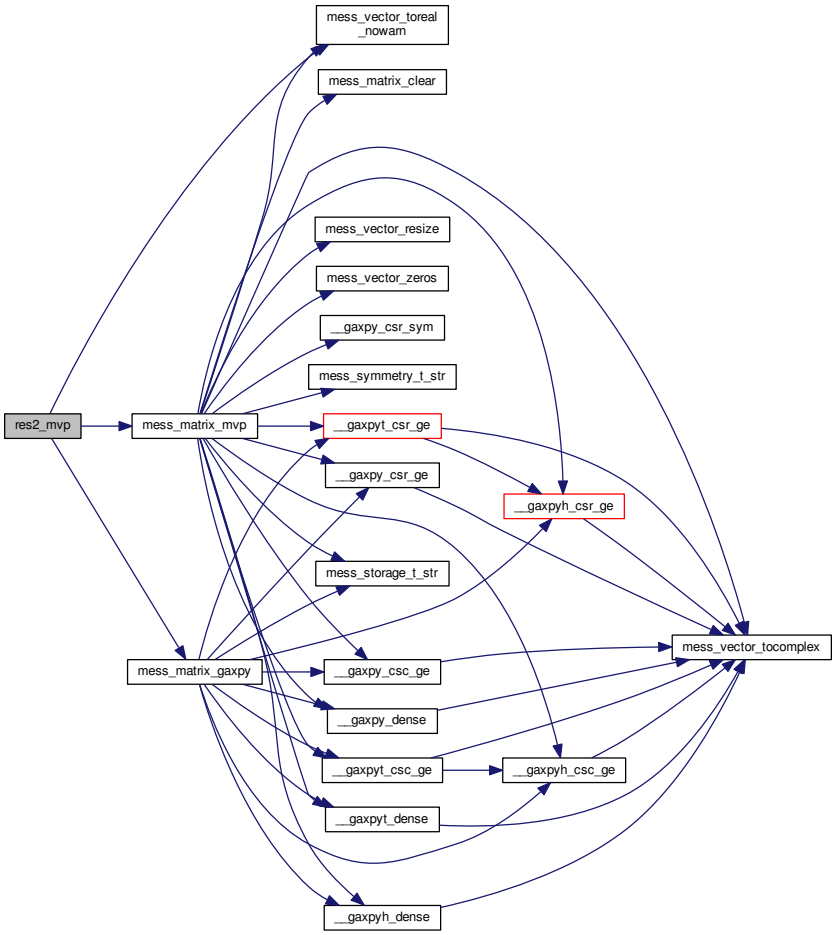
10.626.3.1 typedef struct res2_data_st res2_data

10.626.3.2 typedef struct res2nm_data_st res2nm_data

10.626.4 Function Documentation

10.626.4.1 static int res2_mvp (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



10.626.4.2 int res2_mvpBK (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*)

Parameters

in	*data*	input data
in	*op*	input operation
in	*x*	input vector
out	*y*	vector

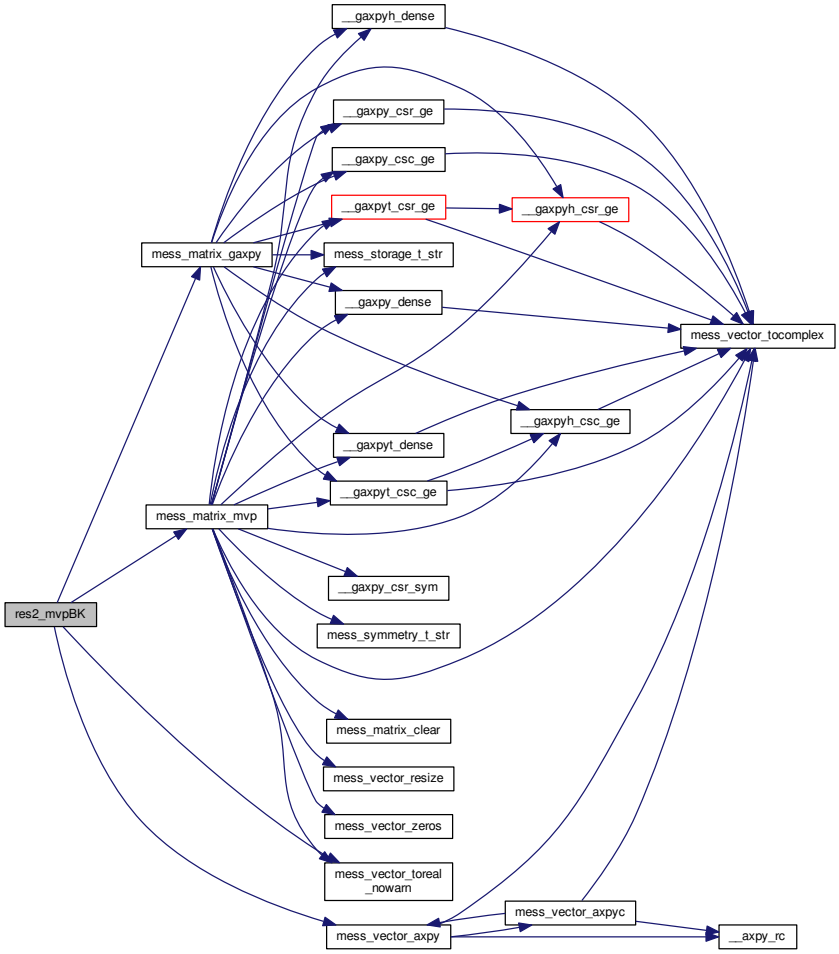
Returns

zero on succes or a non zero error value

The res2_mvpBK function computes a vector y defined by a matrix-vector product of a Lyapunov Equation and a given input vector x:

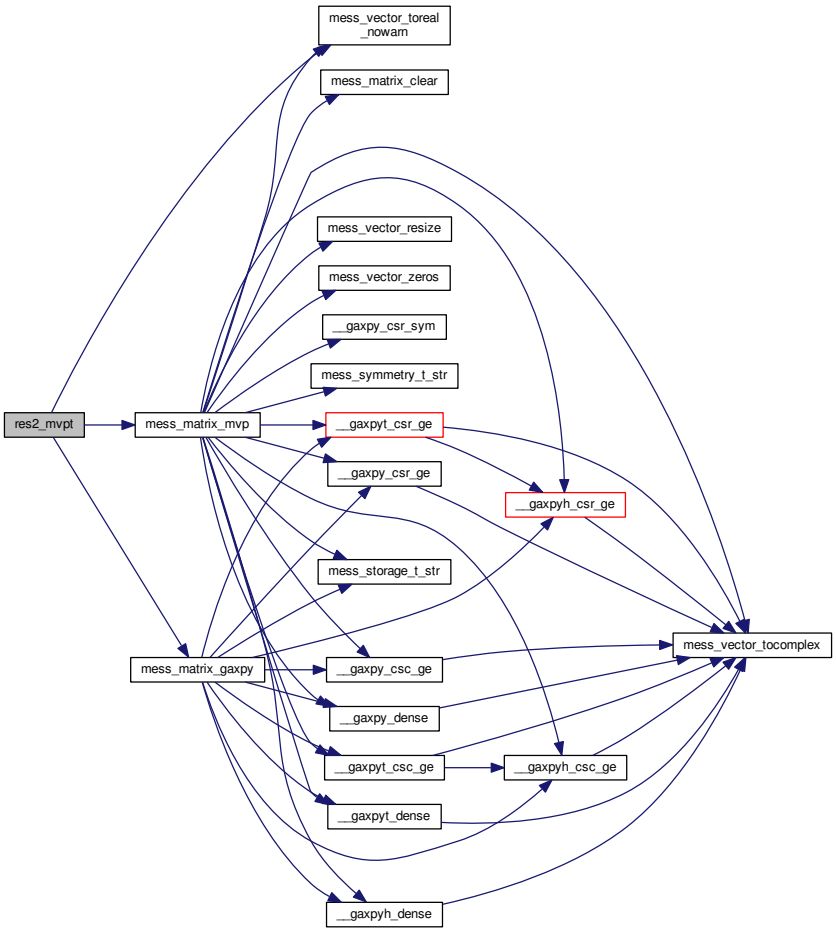
$$y = \left((F - BK)ZZ^T + ZZ^T(F - BK)^T + GG^T\right).$$

Here is the call graph for this function:



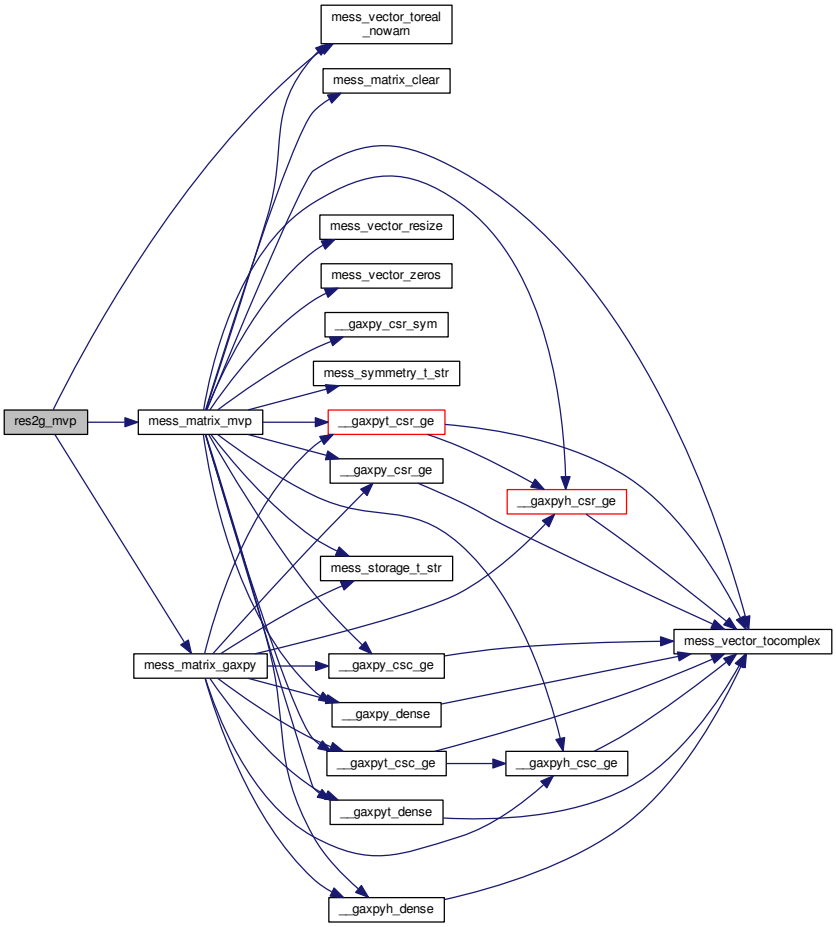
10.626.4.3 static int res2_mvpt (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



10.626.4.4 static int res2g_mvp (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



10.626.4.5 int res2g_mvpBK (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*)

Parameters

in	*data*	input data
in	*op*	input operation
in	*x*	input vector
out	*y*	vector

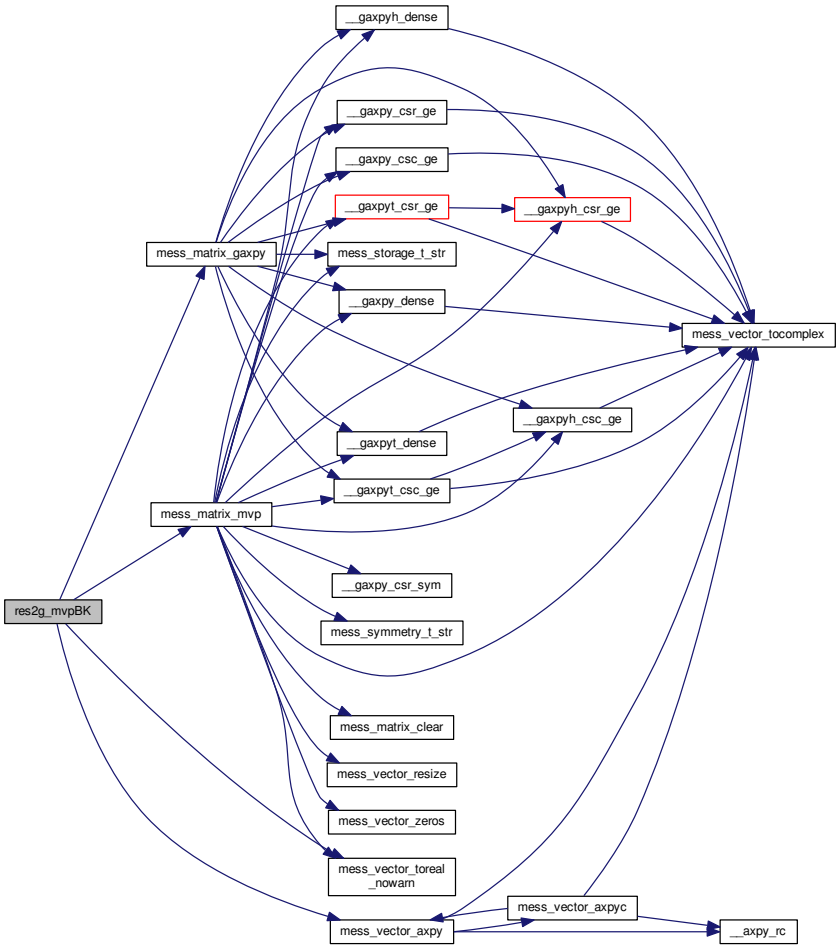
Returns

zero on succes or a non zero error value

The res2_mvpBK function computes a vector y defined by a matrix-vector product of a generalized Lyapunov Equation and a given input vector x:

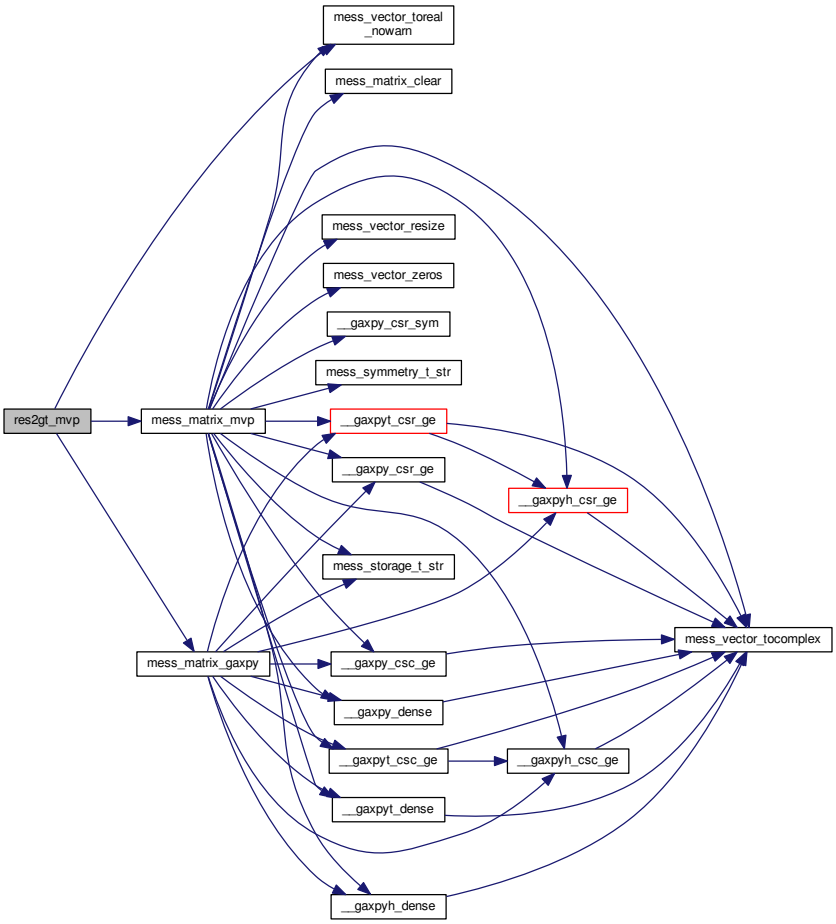
$$y = \left((F - BK)ZZ^T M + MZZ^T (F - BK)^T + GG^T\right).$$

Here is the call graph for this function:



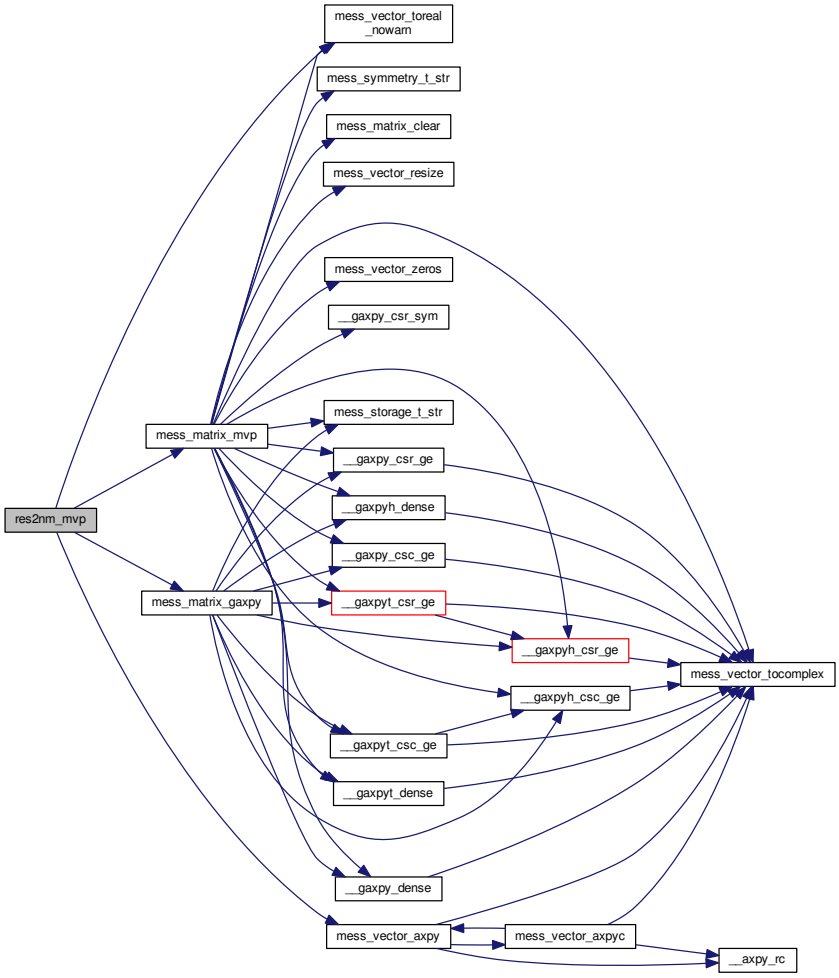
10.626.4.6 static int res2gt_mvp (void * *data,* mess_operation_t *op,* mess_vector *x,* mess_vector *y*) [static]

Here is the call graph for this function:



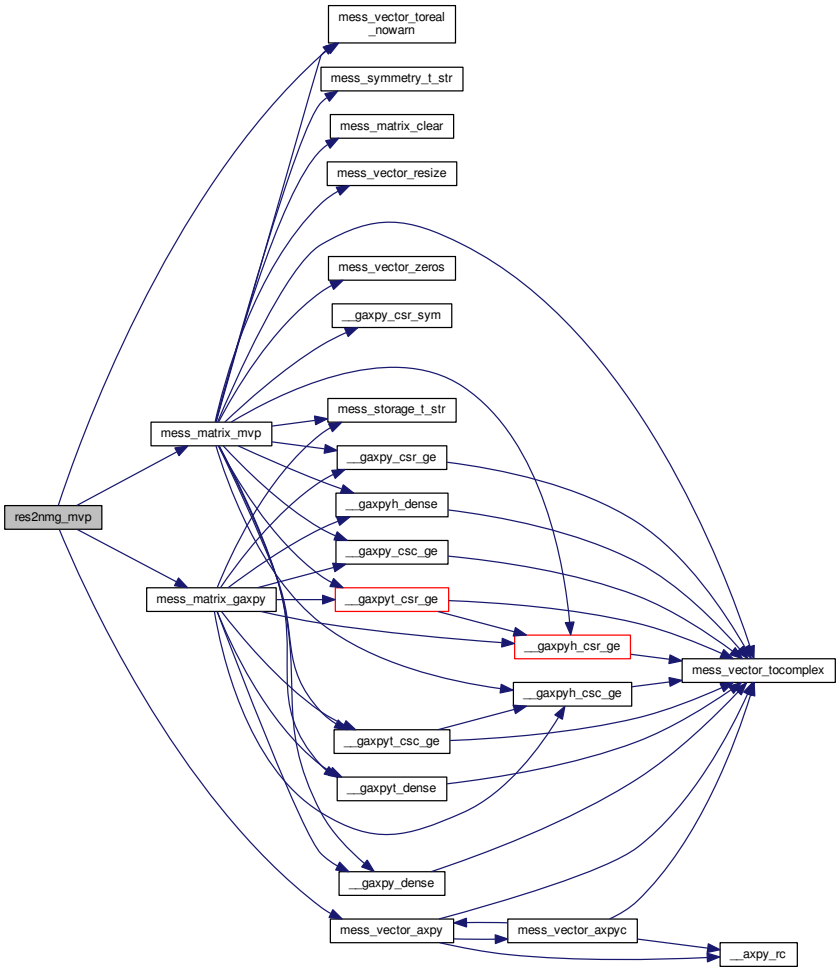
10.626.4.7 static int res2nm_mvp (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



10.626.4.8 static int res2nmg_mvp (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



10.627 res2_template.c File Reference

Data Structures

- struct res2_data_st
- struct res2nm_data_st

Macros

- #define ARNOLDI_ITERATIONS 100
- #define ARPACK_NCV 10

Typedefs

- typedef struct res2_data_st res2_data
- typedef struct res2nm_data_st res2nm_data

Functions

- static int mess_lrcfadi_residual_lyap (mess_equation eqn, mess_options opt, mess_matrix Z, double *nrm)
- static int mess_lrcfadi_residual_nm (mess_equation eqn, mess_options opt, mess_matrix Z, double *nrm)
- int mess_lrcfadi_residual (mess_equation eqn, mess_options opt, mess_matrix Z, double *nrm)

  Compute 2-norm residual of an equation.
- static int res2_mvp (void *data, mess_operation_t op, mess_vector x, mess_vector y)
- static int res2t_mvp (void *data, mess_operation_t op, mess_vector x, mess_vector y)
- static int res2g_mvp (void *data, mess_operation_t op, mess_vector x, mess_vector y)
- static int res2gt_mvp (void *data, mess_operation_t op, mess_vector x, mess_vector y)
- static int res2nm_mvp_none (void *data, mess_operation_t op, mess_vector x, mess_vector y)
- static int res2nm_mvp_transposed (void *data, mess_operation_t op, mess_vector x, mess_vector y)
- static int res2nmg_mvp_none (void *data, mess_operation_t op, mess_vector x, mess_vector y)
- static int res2nmg_mvp_transposed (void *data, mess_operation_t op, mess_vector x, mess_vector y)

10.627.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.627.2 Macro Definition Documentation

10.627.2.1 #define ARNOLDI_ITERATIONS 100

10.627.2.2 #define ARPACK_NCV 10

10.627.3 Typedef Documentation

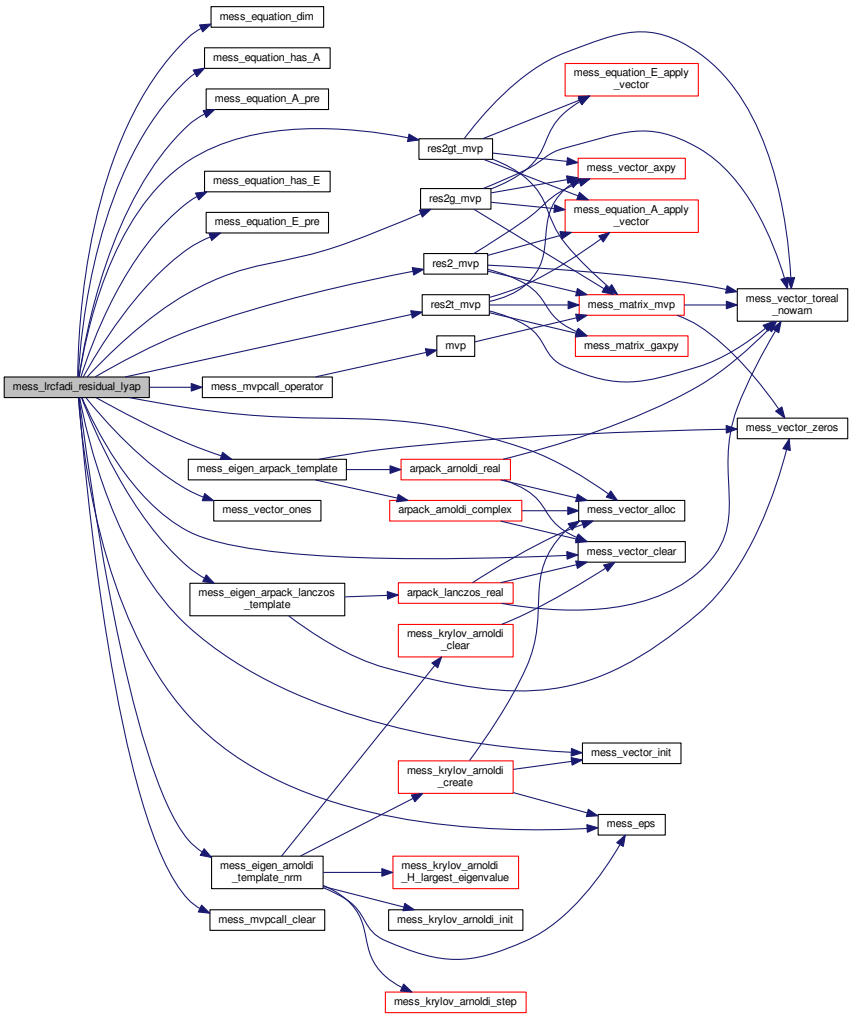
10.627.3.1 typedef struct res2_data_st res2_data

10.627.3.2 typedef struct res2nm_data_st res2nm_data

10.627.4 Function Documentation

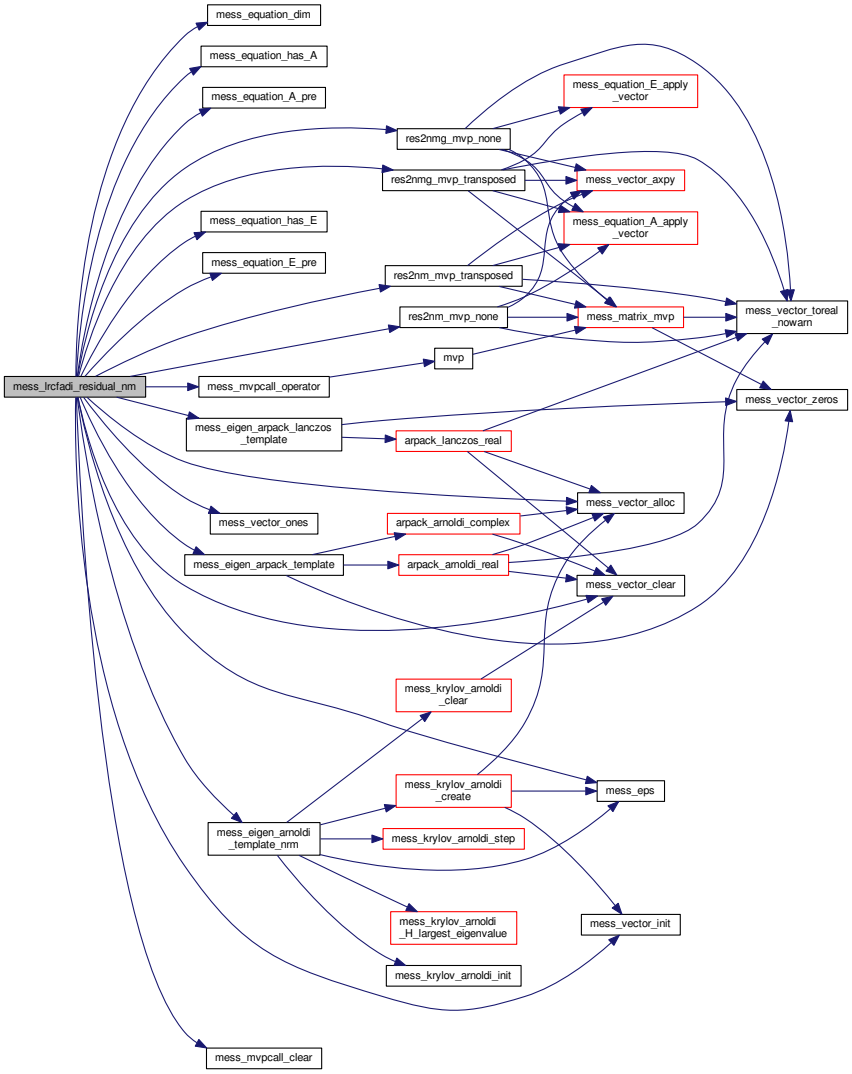
10.627.4.1 static int mess_lrcfadi_residual_lyap (mess_equation *eqn*, mess_options *opt*, mess_matrix *Z*, double * *nrm*) [static]

Here is the call graph for this function:



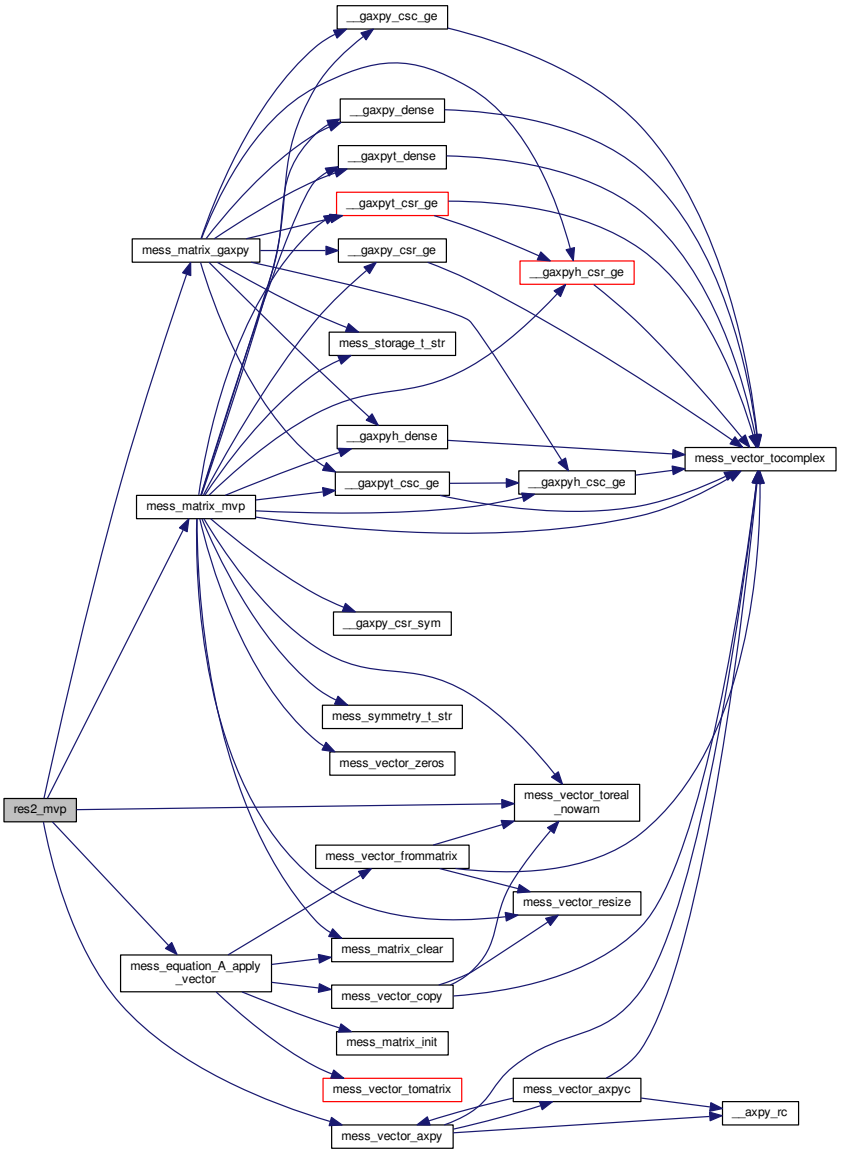
10.627.4.2 static int mess_lrcfadi_residual_nm (mess_equation *eqn*, mess_options *opt*, mess_matrix *Z*, double * *nrm*) [static]

Here is the call graph for this function:



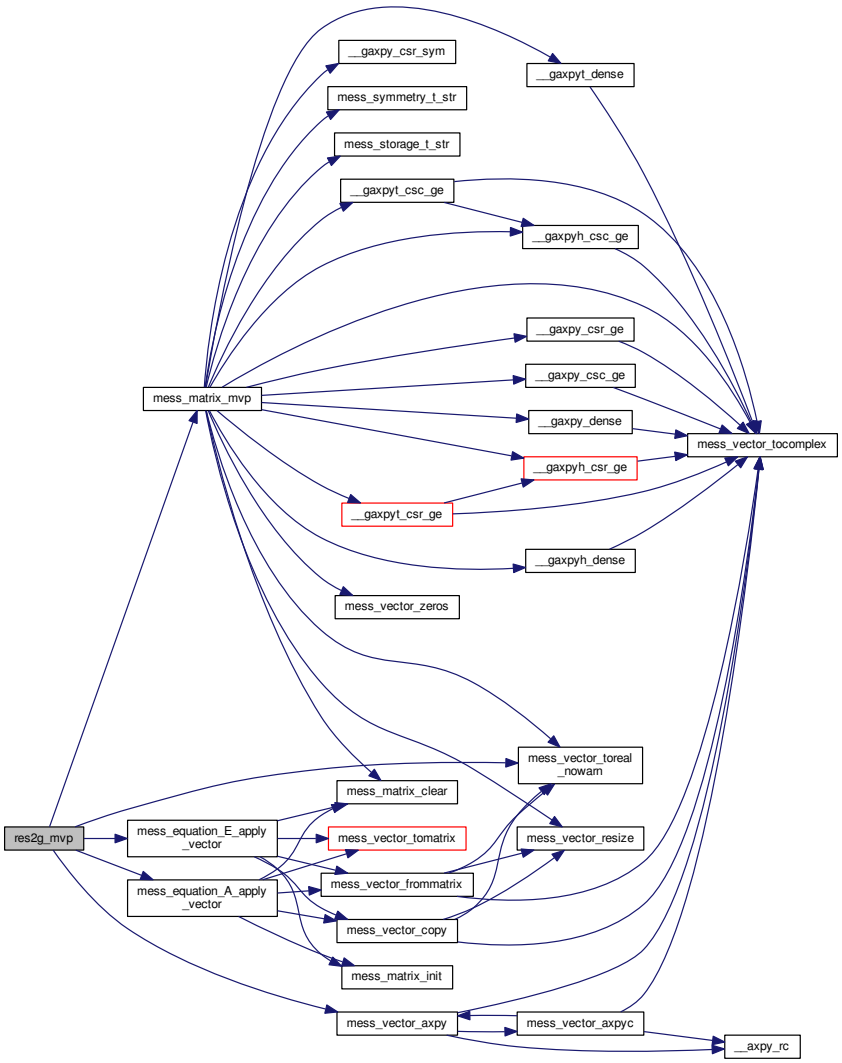
10.627.4.3 static int res2_mvp (void * *data,* mess_operation_t *op,* mess_vector *x,* mess_vector *y*) [static]

Here is the call graph for this function:



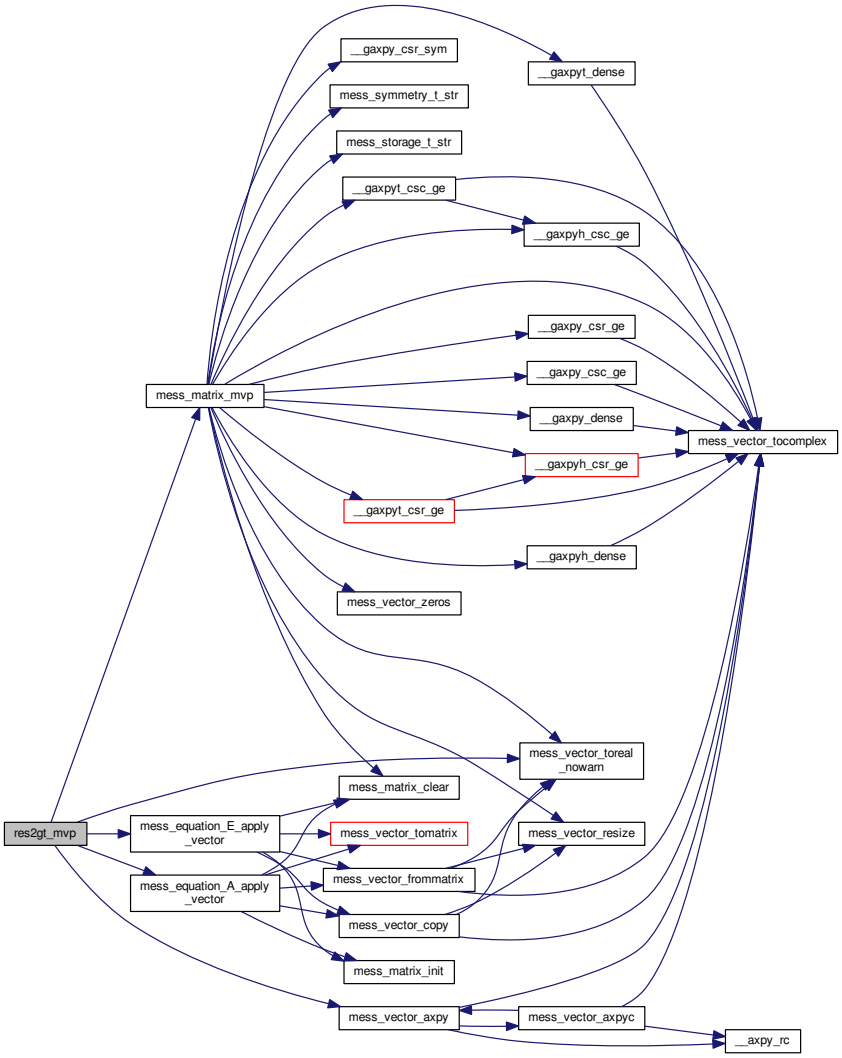
10.627.4.4 static int res2g_mvp (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



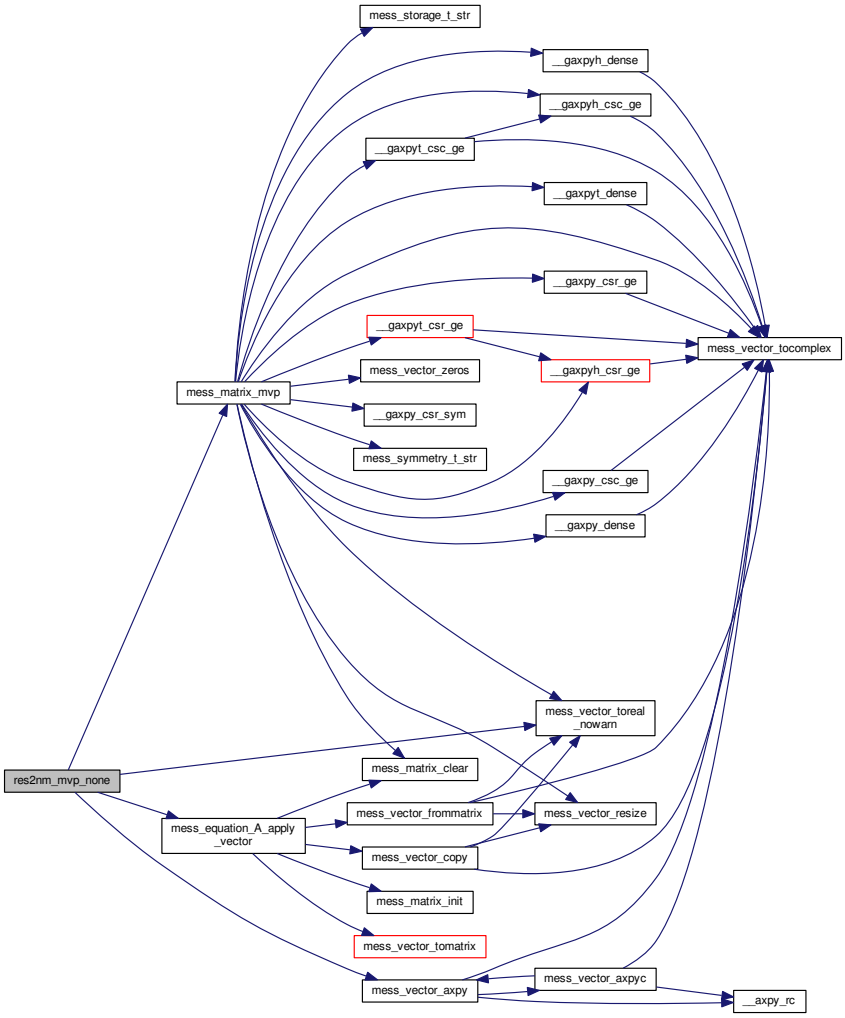
10.627.4.5 static int res2gt_mvp (void * *data,* mess_operation_t *op,* mess_vector *x,* mess_vector *y*) [static]

Here is the call graph for this function:



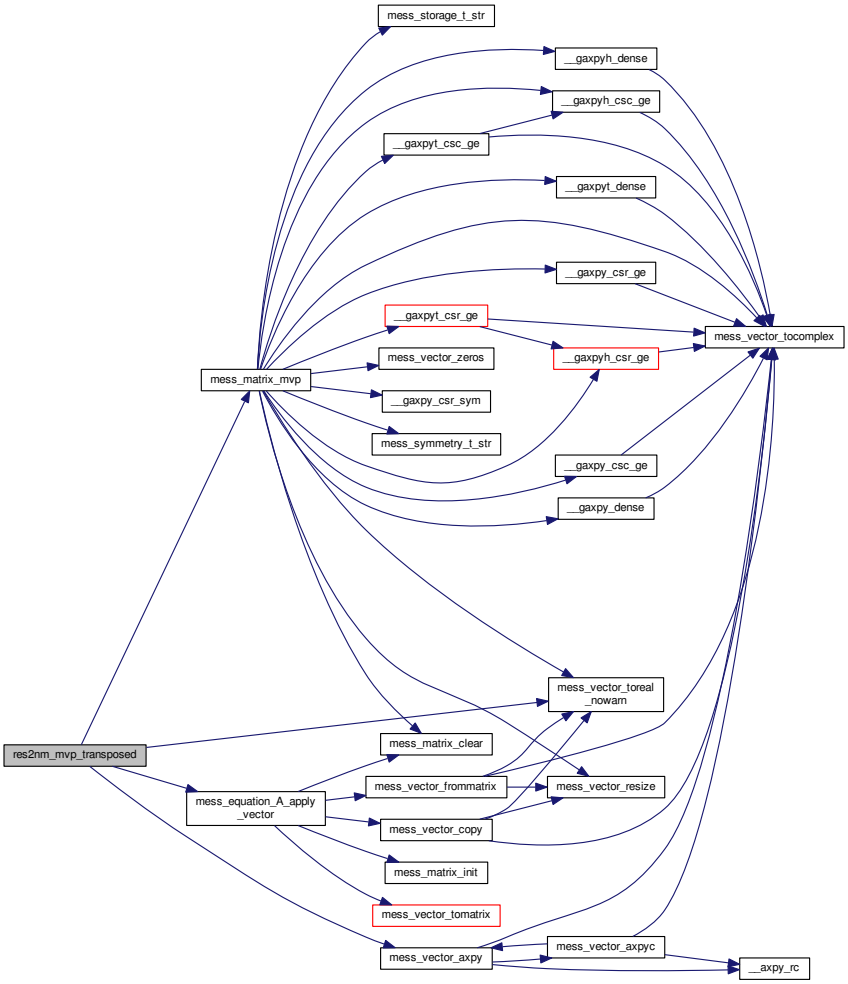
10.627.4.6 static int res2nm_mvp_none (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



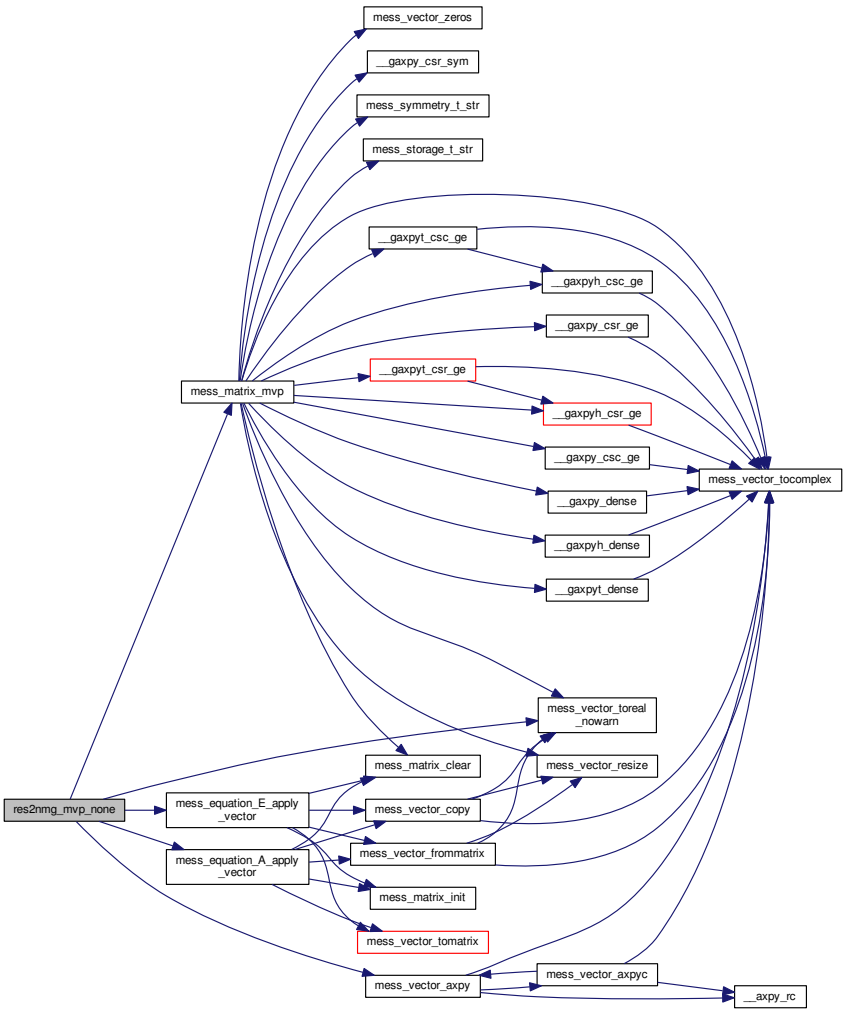
10.627.4.7 static int res2nm_mvp_transposed (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



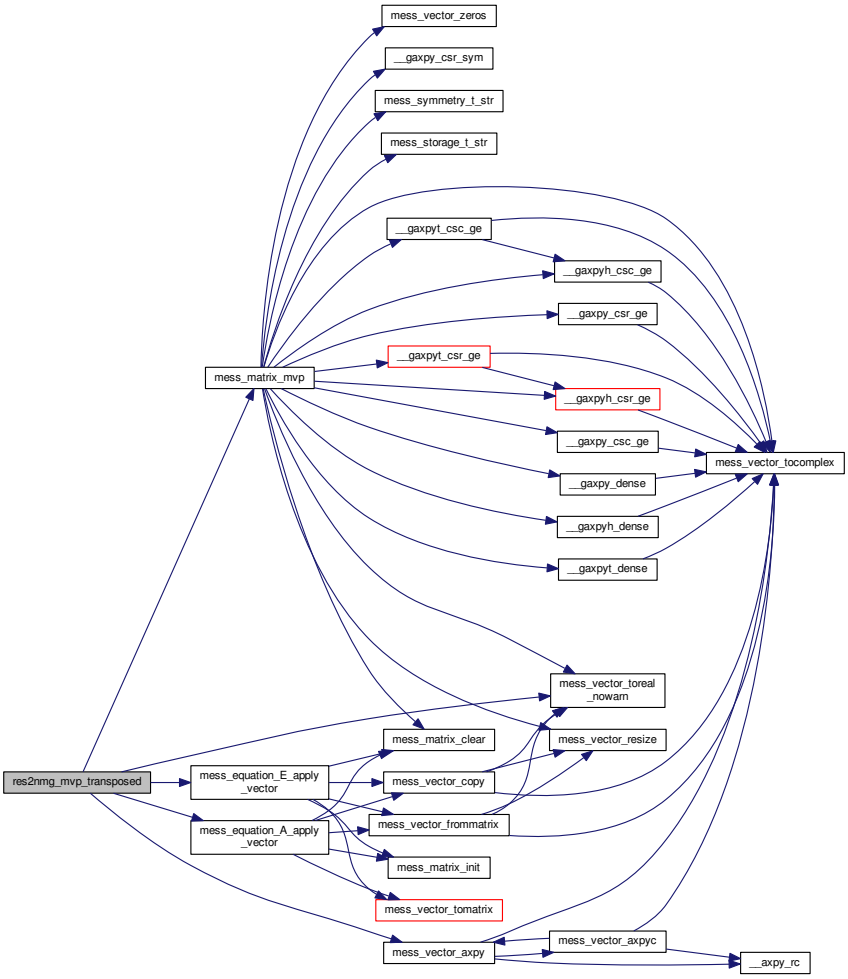
10.627.4.8 static int res2nmg_mvp_none (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



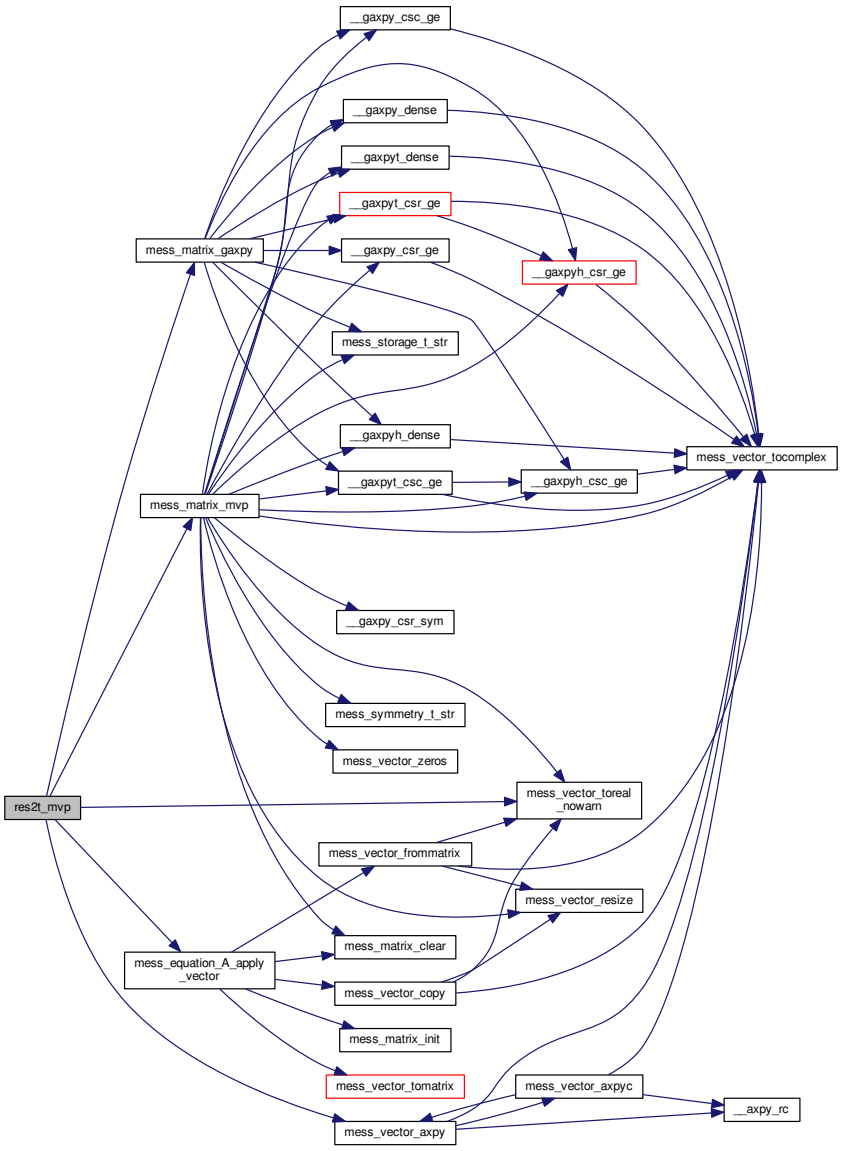
10.627.4.9 static int res2nmg_mvp_transposed (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



10.627.4.10 static int res2t_mvp (void * *data*, mess_operation_t *op*, mess_vector *x*, mess_vector *y*) [static]

Here is the call graph for this function:



10.628 scal.c File Reference

Functions

- int mess_matrix_scale (double alpha, mess_matrix A)

  Scale a real matrix by a scalar.
- int mess_matrix_scalec (mess_double_cpx_t alpha, mess_matrix A)

  Scale a complex matrix by a scalar.
- int mess_matrix_rowscalem (mess_vector r, mess_matrix A)

  Scale rows of a matrix.
- int mess_matrix_colscalem (mess_vector c, mess_matrix A)

  Scale cols of a matrix.

10.628.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.629 scalars.txt File Reference

10.630 scale.c File Reference

Functions

- int mess_vector_scale (double a, mess_vector x)

  Scale a vector with a real scalar.
- int mess_vector_scalec (mess_double_cpx_t a, mess_vector x)

  Scale a vector with a complex scalar.

Variables

- static mess_int_t __ONE = 1

10.630.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.630.2 Variable Documentation

10.630.2.1 mess_int_t __ONE = 1 [static]

On MacOS X 10.6 the zdotc implementation is defect, this is the work around

10.631 schur.c File Reference

Macros

- #define COMPLEX_ZGEES_IF_REAL 1

Functions

- int mess_eigen_schur_complex (mess_matrix A, mess_matrix T, mess_matrix U, mess_vector EV)

  Wrapper around both complex Schur decomposition routines.
- int mess_eigen_schur_complex_zgees (mess_matrix A, mess_matrix T, mess_matrix U, mess_vector EV)

  Compute the complex Schur decomposition of a matrix.
- static int applyUe_right (mess_int_t n, mess_int_t ld, mess_double_cpx_t *U, mess_double_cpx_t ue11, mess_double_cpx_t ue21, mess_double_cpx_t ue12, mess_double_cpx_t ue22)
- static int applyUe_left (mess_int_t n, mess_int_t ld, mess_double_cpx_t *U, mess_double_cpx_t ue11, mess_double_cpx_t ue12, mess_double_cpx_t ue21, mess_double_cpx_t ue22)
- int mess_eigen_schur_complex_post (mess_matrix A, mess_matrix T, mess_matrix U, mess_vector EV)

  Compute the complex Schur decomposition of a matrix using post processing.
- int mess_eigen_schur (mess_matrix A, mess_matrix T, mess_matrix U, mess_vector EV)

  Compute the real Schur decomposition of a matrix.
- int mess_eigen_gschur (mess_matrix A, mess_matrix B, mess_matrix S, mess_matrix T, mess_matrix U, mess_matrix V, mess_vector EVA, mess_vector EVB, mess_vector EV)

  Compute the real generalized Schur decomposition of a matrix pair (A, B).
- int mess_eigen_gschur_complex (mess_matrix A, mess_matrix B, mess_matrix S, mess_matrix T, mess_matrix U, mess_matrix V, mess_vector EVA, mess_vector EVB, mess_vector EV)

  Compute the generalized Schur decomposition of a matrix pair (A, B).

10.631.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.631.2 Macro Definition Documentation

10.631.2.1 #define COMPLEX_ZGEES_IF_REAL 1

10.631.3 Function Documentation

10.631.3.1 static int applyUe_left (mess_int_t *n*, mess_int_t *ld*, mess_double_cpx_t * *U*, mess_double_cpx_t *ue11*, mess_double_cpx_t *ue12*, mess_double_cpx_t *ue21*, mess_double_cpx_t *ue22*) [static]

10.631.3.2 static int applyUe_right (mess_int_t *n*, mess_int_t *ld*, mess_double_cpx_t * *U*, mess_double_cpx_t *ue11*, mess_double_cpx_t *ue21*, mess_double_cpx_t *ue12*, mess_double_cpx_t *ue22*) [static]

10.632 schur_to_evd.c File Reference

Functions

- int mess_eigen_schur_to_evd (mess_matrix T, mess_matrix Q, mess_matrix V)

  Compute eigenvectors from a given Schur decomposition-.

10.632.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.633 select.c File Reference

Functions

- int mess_matrix_reorder (mess_reorder_t reorder, mess_matrix A, mess_int_t *p, mess_int_t *q)

  Select and compute a matrix reordering.

10.633.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.634 setelement.c File Reference

Data Structures

- struct pair_mess_int_t_double
- struct pair_mess_int_t_complex

Functions

- static int __cmp_mess_int_t_double (const void *left, const void *right)

  Compare function for qsort (mess_int_t,double).
- static int __cmp_mess_int_t_complex (const void *left, const void *right)

  Compare function for qsort (mess_int_t,complex).
- int mess_matrix_setelement_complex (mess_matrix matrix, mess_int_t row, mess_int_t col, mess_double↩_cpx_t value)

  Set a complex entry in a matrix.
- int mess_matrix_setelement (mess_matrix matrix, mess_int_t row, mess_int_t col, double value)

  Set a real entry in a matrix.

10.634.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.634.2 Function Documentation

10.634.2.1 static int __cmp_mess_int_t_complex (const void * *left*, const void * *right*) [static]

Parameters

in	*left*	input left hand side
in	*right*	input right hand side

Returns

-1, 0 or 1 to indicate if left<right, left == right or left > right.

The __cmp_mess_int_t_complex function is a compare function to sort pairs (mess_int_t, complex) using qsort.

10.634.2.2 static int __cmp_mess_int_t_double (const void * *left*, const void * *right*) [static]

Parameters

in	*left*	input left hand side
in	*right*	input right hand side

Returns

-1, 0 or 1 to indicate if left<right, left == right or left > right.

The __cmp_mess_int_t_double function is a compare function to sort pairs (mess_int_t, double) using qsort.

10.635 shellsort.c File Reference

Functions

- static void __shellsort (mess_int_t *ind, double *values, mess_double_cpx_t *values_cpx, mess_int_t l, mess_int_t r, unsigned short type)

  Shellsort algorithm for matrix structures.

10.635.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.635.2 Function Documentation

10.635.2.1 static void __shellsort (mess_int_t * *ind*, double * *values*, mess_double_cpx_t * *values_cpx*, mess_int_t *l*, mess_int_t *r*, unsigned short *type*) [static]

Parameters

in,out	*ind*	index array
in,out	*values*	real value array
in,out	*values_cpx*	complex value array
in	*l*	input first position to sort
in	*r*	input last postion to sort
in	*type*	input selects the data type

Perform a shell soft of parts if the ind and values/values_cpx array. This file does not provide a user callable routine. It is only included by other files which needs this algorithm and where using the qsort from the standard libc is too expensive because of the restructuring of the input data.

10.636 sign.c File Reference

Functions

- void F77_GLOBAL (dgesgnnone, DGESGNNONE)
- int mess_eigen_gsign (mess_matrix A, mess_matrix B, mess_matrix Z, mess_sign_scale_t scale)

  Compute the sign of a matrix pencil.

10.636.1 Detailed Description

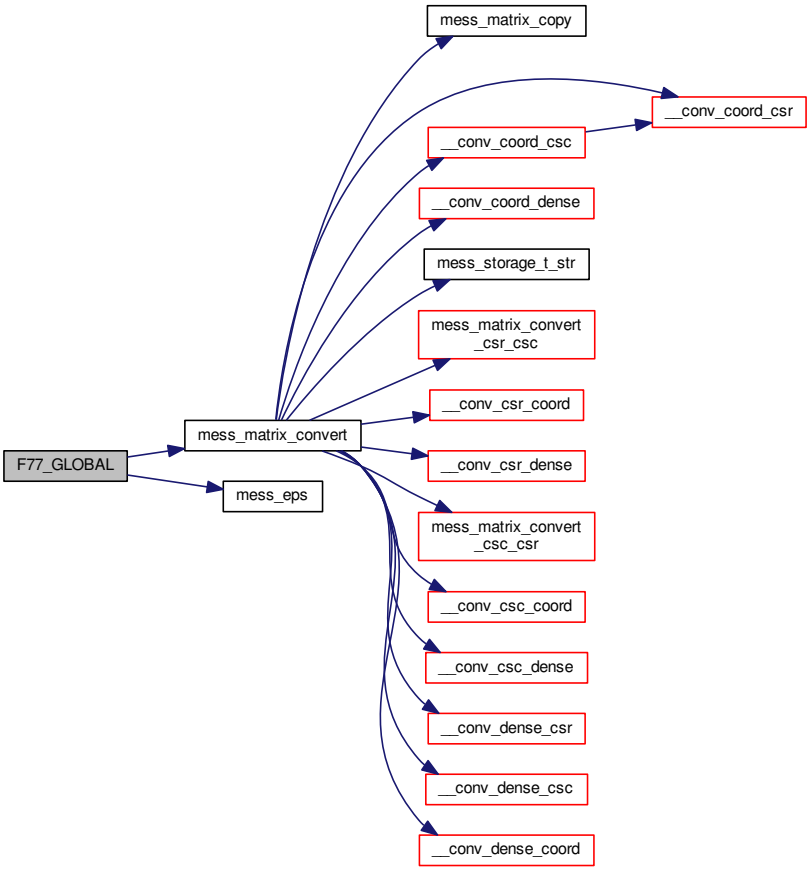
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.636.2 Function Documentation

10.636.2.1 void F77_GLOBAL (dgesgnnone , DGESGNNONE)

Here is the call graph for this function:



10.637 so1.c File Reference

Functions

- mess_equation eqn_conv_lyap_so1 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric_so1 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)

10.637.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.638 so1.c File Reference

Functions

- mess_equation eqn_conv_lyap_so1 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati_so1 (PyObject *obj, mess_freelist mem)

10.638.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.639 so2.c File Reference

Functions

- mess_equation eqn_conv_lyap_so2 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric_so2 (const mxArray *m_eqn, const mxArray *m_opt, mess_freelist mem)

10.639.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.640 so2.c File Reference

Functions

- mess_equation eqn_conv_lyap_so2 (PyObject *obj, mess_freelist mem)
- mess_equation eqn_conv_riccati_so2 (PyObject *obj, mess_freelist mem)

10.640.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.641 solver.h File Reference

Data Structures

- struct __mess_solver_status

  Representation of status information of an iterative solvers.
- struct __mess_solver_options

  Representation of options for iterative solvers.
- struct mess_precond_st

  Representation of a preconditioner.

Macros

- #define MESS_PRECOND_DIAG 1

  Definition of a diagonal preconditioner.
- #define MESS_PRECOND_ILU0 2

  Definition of a $ILU(0)$ preconditioner.
- #define MESS_PRECOND_ILUK 3

  Definition of a $ILU(k)$ preconditioner.

Typedefs

- typedef struct __mess_solver_status * mess_solver_status

  Type definition for status information of an iterative solvers.
- typedef struct __mess_solver_options * mess_solver_options

  Type definition for options of an iterative solvers.
- typedef struct mess_precond_st * mess_precond

  Type definition for a preconditioner.

Functions

- int mess_solver_options_init (mess_solver_options *opt)

  Initialize mess_solver_options.
- int mess_solver_options_print (mess_solver_options opt)

  Print a mess_solver_options object.
- int mess_solver_options_clear (mess_solver_options *opt)

  Clear a mess_solver_options object.
- int mess_solver_status_init (mess_solver_status *opt)

  Initialize a mess_solver_status object.
- int mess_solver_status_print (mess_solver_status opt)

  Print a mess_solver_status object to stdout.
- int mess_solver_status_clear (mess_solver_status *opt)

  Clear a mess_solver_status object.

- int mess_solver_status_clean (mess_solver_status stat)

  Reset values in mess_solver_status.
- int mess_solver_gmres (mess_mvpcall matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with (preconditioned) generalized minimal residual algorithm.
- int mess_solver_gmres_restart (mess_mvpcall matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with restarted GMRES.
- int mess_solver_bicgstab (mess_mvpcall matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with BICGSTAB algorithm.
- int mess_solver_cg (mess_mvpcall matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations using conjugate gradients.
- int mess_solver_sor (mess_matrix matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with successive overrelaxation.
- int mess_solver_ssor (mess_matrix matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with symmetric successive overrelaxation).
- int mess_solver_gaussseidel (mess_matrix matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with the Gauss-Seidel method.
- int mess_solver_jacobi (mess_matrix matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with the Jacobi method.
- int mess_solver_jacobi_convergence (mess_matrix matrix, int *convergence)

  Check if the Jacobi iteration is guaranteed to converge.
- int mess_precond_init (mess_precond *precond)

  Initialize a mess_precond object.
- int mess_precond_clear (mess_precond *precond)

  Clean up the mess_precond structure to use it again.
- int mess_precond_diag (mess_precond pre, mess_matrix mat)

  Setup a diagonal preconditioner.
- int mess_precond_ilu0 (mess_precond pre, mess_matrix matrix)

  Compute an $ILU(0)$ preconditioner.
- int mess_precond_iluk (mess_precond pre, mess_matrix matrix, mess_int_t leveloffill)

  Setup an $ILU(k)$ preconditioner from a matrix.
- int mess_precond_sor (mess_precond pre, mess_matrix matrix, mess_int_t it, double omega)

  Compute a SOR preconditioner.
- int mess_precond_ssor (mess_precond pre, mess_matrix matrix, mess_int_t it, double omega)

  Compute a SSOR preconditioner.

10.641.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.642 solvername.h File Reference

Macros

- #define SET_SOLVERNAME(X, Y)

10.642.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.642.2 Macro Definition Documentation

10.642.2.1 #define SET_SOLVERNAME(X, Y)

Value:

```
do { \
    mess_try_alloc(X, char *, (strlen(Y)+1) * sizeof(char)); \
    strncpy(X, Y, strlen(Y)+1); \
} while(0)
```

10.643 sor.c File Reference

Functions

- int mess_solver_sor (mess_matrix matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with successive overrelaxation.
- int mess_solver_ssor (mess_matrix matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with symmetric successive overrelaxation).
- int mess_solver_gaussseidel (mess_matrix matrix, mess_precond pre, mess_vector b, mess_vector x, mess_solver_options opt, mess_solver_status stat)

  Solve a system of linear equations with the Gauss-Seidel method.

10.643.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.644 sor.c File Reference

Data Structures

- struct __presordata

Functions

- static int __mess_precond_sor_solve (mess_precond myself, mess_solver_options opt, mess_vector b, mess_vector x)
- static int __mess_precond_ssor_solve (mess_precond myself, mess_solver_options opt, mess_vector b, mess_vector x)
- static int __mess_precond_sor_clear (mess_precond myself)
- int mess_precond_sor (mess_precond pre, mess_matrix matrix, mess_int_t it, double omega)

  Compute a SOR preconditioner.
- int mess_precond_ssor (mess_precond pre, mess_matrix matrix, mess_int_t it, double omega)

  Compute a SSOR preconditioner.

10.644.1 Detailed Description

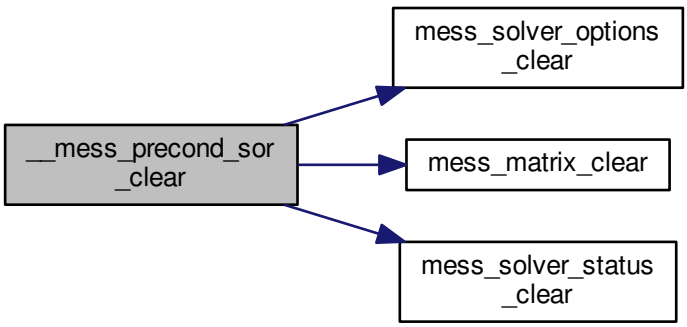
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.644.2 Function Documentation

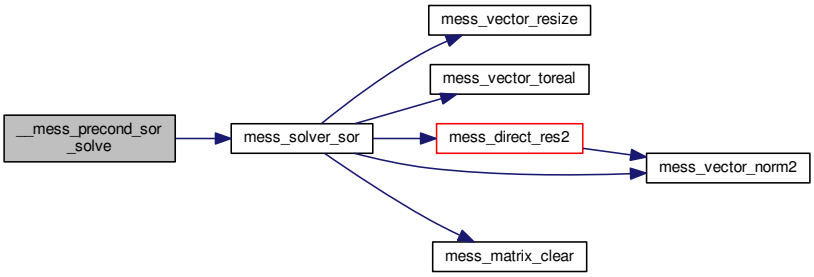
10.644.2.1 static int __mess_precond_sor_clear (mess_precond *myself*) [static]

Here is the call graph for this function:



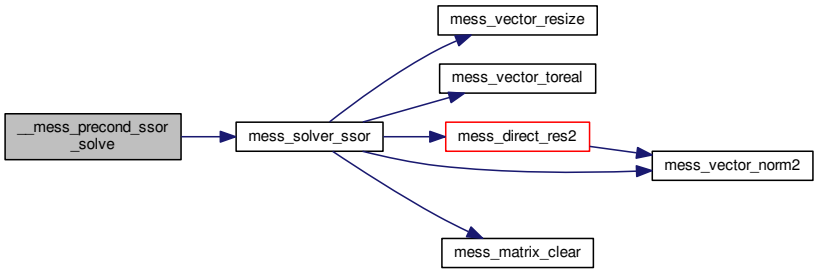
10.644.2.2 static int __mess_precond_sor_solve (mess_precond *myself*, mess_solver_options *opt*, mess_vector *b*, mess_vector *x*) [static]

Here is the call graph for this function:



10.644.2.3 static int __mess_precond_ssor_solve (mess_precond *myself*, mess_solver_options *opt*, mess_vector *b*, mess_vector *x*) [static]

Here is the call graph for this function:



10.645 sort.c File Reference

Data Structures

- struct pair_mess_int_t_double
- struct pair_mess_int_t_complex
- struct pair_mess_int_t2_double
- struct pair_mess_int_t2_complex
- struct pair_mess_int_t2

Functions

- static int __cmp_mess_int_t_double (const void *left, const void *right)

  Compare function for qsort (mess_int_t,double).
- static int __cmp_mess_int_t_complex (const void *left, const void *right)

  Compare function for qsort (mess_int_t, complex).
- static int __cmp_mess_int_t2_double (const void *left, const void *right)

  Compare function for qsort (mess_int_t,mess_int_t, double).
- static int __cmp_mess_int_t2_complex (const void *left, const void *right)

  Compare function for qsort (mess_int_t,mess_int_t,double complex).
- int mess_matrix_sort (mess_matrix mat)

  Sort the internal structure of a sparse matrix.

10.645.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file implements the data structure sorting inside the matrix data structures.

10.645.2 Function Documentation

10.645.2.1 static int __cmp_mess_int_t2_complex (const void * *left*, const void * *right*) [static]

Parameters

in	*left*	input left side
in	*right*	input right side

Returns

-1 if left<right, 0 if left==right, 1 if left > right

The __cmp_mess_int_t2_complex function is a compare function to sort triples (mess_int_t, mess_int_t, double complex).

10.645.2.2 static int __cmp_mess_int_t2_double (const void * *left*, const void * *right*) [static]

Parameters

in	*left*	input left side
in	*right*	input right side

Returns

-1 if left<right, 0 if left==right, 1 if left > right

The __cmp_mess_int_t2_double function is a compare function to sort triples (mess_int_t, mess_int_t,double) using lexicographic order on the first and second entry.

10.645.2.3 static int __cmp_mess_int_t_complex (const void * *left*, const void * *right*) [static]

Parameters

in	*left*	input left side
in	*right*	input right side

Returns

-1 if left<right, 0 if left==right, 1 if left > right

The __cmp_mess_int_t_complex function is a compare function to sort pairs (mess_int_t, `double complex`) using lexicographic order on the first entry.

10.645.2.4 static int __cmp_mess_int_t_double (const void * *left*, const void * *right*) [static]

Parameters

in	*left*	input left side
in	*right*	input right side

Returns

-1 if left<right, 0 if left==right, 1 if left > right

The __cmp_mess_int_t_double function is a compare function to sort pairs (mess_int_t, `double`) using lexicographic order on the first entry.

10.646 source.txt File Reference

10.647 source.txt File Reference

Functions

- mmwrite ('E11.mtx', E11)
- mmwrite ('A11.mtx', A11)
- mmwrite ('A12.mtx', A12)
- mmwrite ('A21.mtx', A21)
- mmwrite ('A22.mtx', A22)
- mmwrite ('B.mtx', B)
- mmwrite ('C.mtx', C)

Variables

- EXAMPLES TAKEN FROM
- np = find(diag(E) == 0)
- pp = [p
- E11 = E(p,p)
- A11 = A(p,p)-1e+4∗E(p,p)
- A12 = A(p,np)
- A21 = A(np,p)
- A22 = A(np,np)
- B = b(pp,:)
- C = c(:,pp)

10.647.1 Function Documentation

10.647.1.1 mmwrite ('E11.mtx' , E11)

10.647.1.2 mmwrite ('A11.mtx' , A11)

10.647.1.3 mmwrite ('A12.mtx' , A12)

10.647.1.4 mmwrite ('A21.mtx' , A21)

10.647.1.5 mmwrite ('A22.mtx' , A22)

10.647.1.6 mmwrite ('B.mtx' , B)

10.647.1.7 mmwrite ('C.mtx' , C)

10.647.2 Variable Documentation

10.647.2.1 A11 = A(p,p)-1e+4∗E(p,p)

10.647.2.2 A12 = A(p,np)

10.647.2.3 A21 = A(np,p)

10.647.2.4 A22 = A(np,np)

10.647.2.5 B = b(pp,:)

10.647.2.6 C = c(:,pp)

10.647.2.7 E11 = E(p,p)

10.647.2.8 EXAMPLES TAKEN FROM

10.647.2.9 np = find(diag(E) == 0)

10.647.2.10 pp = [p

10.648 source.txt File Reference

10.649 source.txt File Reference

10.650 source.txt File Reference

10.651 source.txt File Reference

10.652 source.txt File Reference

10.653 source.txt File Reference

Functions

- write matrices mmwrite ('E11.mtx', E11)
- mmwrite ('A11.mtx', A11)
- mmwrite ('A12.mtx', A12)
- mmwrite ('A21.mtx', A21)
- mmwrite ('A22.mtx', A22)
- mmwrite ('B.mtx', B)
- mmwrite ('C.mtx', C)

Variables

- EXAMPLES TAKEN FROM
- p = find(diag(E))
- np = find(diag(E) == 0)
- pp = [p
- E11 = E(p,p)
- A11 = A(p,p)
- A12 = A(p,np)
- A21 = A(np,p)
- A22 = A(np,np)
- B = b(pp, :)
- C = c(pp,:)'

10.653.1 Function Documentation

10.653.1.1 write matrices mmwrite ('E11.mtx' , E11)

10.653.1.2 mmwrite ('A11.mtx' , A11)

10.653.1.3 mmwrite ('A12.mtx' , A12)

10.653.1.4 mmwrite ('A21.mtx' , A21)

10.653.1.5 mmwrite ('A22.mtx' , A22)

10.653.1.6 mmwrite ('B.mtx' , B)

10.653.1.7 mmwrite ('C.mtx' , C)

10.653.2 Variable Documentation

10.653.2.1 A11 = A(p,p)

10.653.2.2 A12 = A(p,np)

10.653.2.3 A21 = A(np,p)

10.653.2.4 A22 = A(np,np)

10.653.2.5 B = b(pp, :)

10.653.2.6 C = c(pp,:)'

10.653.2.7 E11 = E(p,p)

10.653.2.8 EXAMPLES TAKEN FROM

10.653.2.9 np = find(diag(E) == 0)

10.653.2.10 p = find(diag(E))

10.653.2.11 pp = [p

10.654 source.txt File Reference

Typedefs

- using n1 = 100

Variables

- alpha =2
- beta =5
- v =5

10.654.1 Typedef Documentation

10.654.1.1 using n1 = 100

10.654.2 Variable Documentation

10.654.2.1 alpha =2

10.654.2.2 beta =5

10.654.2.3 v =5

10.655 splsolve.c File Reference

Functions

- int mess_direct_splsolve (mess_matrix G, mess_matrix B, mess_int_t k, mess_int_t *top, mess_int_t *xi, double *x, const mess_int_t *pinv)

  Solve $x = G^{-1}B(:,k)$ with all sparse.
- int mess_direct_splsolvec (mess_matrix G, mess_matrix B, mess_int_t k, mess_int_t *top, mess_int_t *xi, mess_double_cpx_t *x, const mess_int_t *pinv)

  Solve $x = G^{-1}B(:,k)$ with all sparse with complex data.

10.655.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.656 spy.c File Reference

Functions

- int mess_matrix_spy (mess_matrix matrix, const char *filename, int width, int height)

  Generate a bitmap file from the non zero structure of a matrix.

10.656.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.657 stable.c File Reference

Functions

- int mess_dynsys_isstable (mess_dynsys sys, int *stable)

  Check if a dynamical system is stable.
- int mess_matrix_isstable (mess_matrix A, int *stable)

  Check if a matrix is stable.
- int mess_matrix_isstableg (mess_matrix A, mess_matrix E, int *stable)

  Check if a matrix pair is stable.

10.657.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.658 status.c File Reference

Functions

- int mess_bt_status_init (mess_bt_status *status)

  Initialize a mess_bt_status object.
- int mess_bt_status_clear (mess_bt_status *status)

  Clear a mess_bt_status object.
- int mess_bt_status_print (mess_bt_status status)

  Print a mess_bt_status object to stdout.

10.658.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Description:

10.659 status.c File Reference

Functions

- int mess_solver_options_init (mess_solver_options *opt)

  Initialize mess_solver_options.
- int mess_solver_options_clear (mess_solver_options *opt)

  Clear a mess_solver_options object.
- int mess_solver_options_print (mess_solver_options opt)

  Print a mess_solver_options object.
- int mess_solver_status_init (mess_solver_status *opt)

  Initialize a mess_solver_status object.
- int mess_solver_status_clear (mess_solver_status *opt)

  Clear a mess_solver_status object.
- int mess_solver_status_print (mess_solver_status stat)

  Print a mess_solver_status object to stdout.
- int mess_solver_status_clean (mess_solver_status stat)

  Reset values in mess_solver_status.

10.659.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.660 status.c File Reference

Functions

- int mess_status_init (mess_status *status)

  Initialize a mess_status object.
- int mess_status_clear (mess_status *status)

  Clear a mess_status object.
- int mess_status_copy (mess_status src, mess_status dest)

  Copy a mess_status object to another one.
- mess_int_t mess_status_memsize (mess_status stat)

  Return the size of a mess_status in bytes.
- int mess_status_print (mess_status stat)

  Print a mess_status object to the standard output.
- int mess_status_printfull (mess_status stat)

  Print a mess_status object to the standard output including embbed objects.
- int mess_status_write (mess_status stat, const char *filename)

  Save the information of a mess_status object to a text file.

10.660.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.661 status.c File Reference

Functions

- PyObject * mess_status_to_python (mess_status status)

  Convert a mess_status object from C to Python.

10.661.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Nitin Prasad (nitin.prasad93@gmail.com, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.662 statusopt.c File Reference

Functions

- int mess_h2_options_init (mess_h2_options ∗opt)

  Initialize a mess_h2_options object.
- int mess_h2_options_clear (mess_h2_options ∗opt)

  Clean up a mess_h2_options object.
- int mess_h2_options_print (mess_h2_options opt)

  Print a mess_h2_options object to screen.
- int mess_h2_status_init (mess_h2_status ∗status)

  Initialize a mess_h2_status object.
- int mess_h2_status_clear (mess_h2_status ∗status)

  Clean up a mess_h2_status object.
- int mess_h2_status_print (mess_h2_status status)

  Print a mess_h2_status object.
- int mess_h2_status_printfull (mess_h2_status status)

  Print a mess_h2_status object (full output).
- int mess_h2_status_write_mfile (mess_h2_status status, const char ∗filename)

  Write a mfile to plot in `MATLAB` .

10.662.1 Detailed Description

Author

Martin Köhler (`koehlerm@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.663 std.c File Reference

Functions

- mess_equation eqn_conv_lyap (const mxArray ∗m_eqn, const mxArray ∗m_opt, mess_freelist mem)
- mess_equation eqn_conv_ric (const mxArray ∗m_eqn, const mxArray ∗m_opt, mess_freelist mem)

10.663.1 Detailed Description

Author

Maximilian Behr (`behr@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.664 std.c File Reference

Functions

- mess_equation eqn_conv_lyap (PyObject ∗obj, mess_freelist mem)
- mess_equation eqn_conv_riccati (PyObject ∗obj, mess_freelist mem)

10.664.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.665 stein_res2.c File Reference

Functions

- int mess_direct_stein_res2 (mess_operation_t op, mess_matrix A, mess_matrix M, mess_matrix X, double *res)

  Compute the 2-norm residual of a Stein Equation (dense).
- int mess_direct_generalized_stein_res2 (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix M, mess_matrix X, double *res2)

  Compute the 2-norm residual of a generalized Stein Equation.

10.665.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.666 sub.c File Reference

Functions

- int mess_matrix_rowsub (mess_matrix input, mess_int_t srow, mess_int_t erow, mess_matrix out)

  Copy a row-block out of a matrix.
- int mess_matrix_colsub (mess_matrix input, mess_int_t scol, mess_int_t ecol, mess_matrix out)

  Copy a column block out of a matrix.
- int mess_matrix_sub (mess_matrix in, mess_int_t rowS, mess_int_t rowE, mess_int_t colS, mess_int_t colE, mess_matrix out)

  Copy a sub matrix out of a matrix.

10.666.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.667 superlu.c File Reference

10.667.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.668 superlu_refinement_bug.c File Reference

Functions

- int main (void)

10.668.1 Function Documentation

10.668.1.1 int main (void)

10.669 superlu_refinement_bug_43.c File Reference

Functions

- int main (void)

10.669.1 Function Documentation

10.669.1.1 int main (void)

10.670 superlu_refinement_bug_50.c File Reference

Functions

- int main (void)

10.670.1 Function Documentation

10.670.1.1 int main (void)

10.671 suppress_cppcheck.txt File Reference

10.672 svd.c File Reference

Functions

- int mess_eigen_svd (mess_matrix A, mess_vector S, mess_matrix U, mess_matrix V)

  Compute the singular value decompositon (SVD) of a matrix.
- int mess_eigen_svd_econ (mess_matrix A, mess_vector S, mess_matrix U, mess_matrix V)

  Compute the singular value decompositon (SVD) of a matrix.

10.672.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.673 sylv_res2.c File Reference

Functions

- int mess_direct_sylvester_res2 (mess_operation_t op, mess_matrix A, mess_matrix H, mess_matrix M, mess_matrix X, double *res)

  Compute the 2-norm residual of a standard Sylvester equation.
- int mess_direct_sylvestersg_res2 (mess_operation_t op, mess_matrix A, mess_matrix E, mess_matrix H, mess_matrix M, mess_matrix X, double *res)

  Compute the 2-norm residual of a semi-generalized Sylvester equation.
- int mess_direct_generalized_sylvester_res2 (mess_operation_t op, mess_matrix A, mess_matrix F, mess_matrix E, mess_matrix H, mess_matrix M, mess_matrix X, double *res)

  Compute the 2-norm residual of a generalized Sylvester equation.

10.673.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.674 sylvester_dense.c File Reference

Data Structures

- struct _sylv_solver_d_st

Typedefs

- typedef struct _sylv_solver_d_st _sylv_solver_d

Functions

- static int sylvester_d_clear (void *solver)

  Clean up handler for mess_direct_create_sylvester_dense.
- static int sylvester_d_solvem_standard (void *datain, mess_matrix M, mess_matrix X)

  Solve $AX + XH + M = 0$.
- static int sylvester_d_solvemh_standard (void *datain, mess_matrix M, mess_matrix X)

  Solve $A^H X + XH^H + M = 0$.
- static int sylvester_d_solvemt_standard (void *datain, mess_matrix M, mess_matrix X)

  Solve $A^T X + XH^T + M = 0$.
- static int sylvester_d_solvem_generalized (void *datain, mess_matrix M, mess_matrix X)

  Solve $AXF + EXH + M = 0$.
- static int sylvester_d_solvemh_generalized (void *datain, mess_matrix M, mess_matrix X)

  Solve $A^H XF^H + E^H XH^H + M = 0$.
- static int sylvester_d_solvemt_generalized (void *datain, mess_matrix M, mess_matrix X)

  Solve $A^T XF^T + E^T XH^T + M = 0$.
- int mess_direct_create_sylvester_dense (mess_matrix A, mess_matrix F, mess_matrix E, mess_matrix H, mess_direct solver)

  Generate a solver for a dense Sylvester equation using LAPACK.

10.674.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Robert Dykstra (dykstra@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This code implements a sparse-dense Sylvester equation solver for the following Sylvester equations:

- Standard: $AX + XH + M = 0$, $A^T X + XH^T + M = 0$, $A^H X + XH^H + M = 0$
- Generalized Sylvester: $AXF + EXH + M = 0$, $A^T XF^T + E^T XH^T + M = 0$, $A^H XF^H + E^H XH^H + M = 0$

with A, F, E, H small and dense matrices.

10.674.2 Typedef Documentation

10.674.2.1 typedef struct _sylv_solver_d_st _sylv_solver_d

Attention

Internal use only.

10.674.3 Function Documentation

10.674.3.1 static int sylvester_d_clear (void * *solver*) [static]

Parameters

in	*solver*	input pointer to the solver data

Returns

always zero

The sylvester_d_clear function is the clean up function.

Attention

Internal use only.

Here is the call graph for this function:

10.674.3.2 static int sylvester_d_solvem_generalized (void * *datain*, mess_matrix *M*, mess_matrix *X*) [static]

Parameters

in	*datain*	input solver data
in	*M*	input right hand sides
out	*X*	output solution

Returns

zero on success or a non-zero error value otherwise

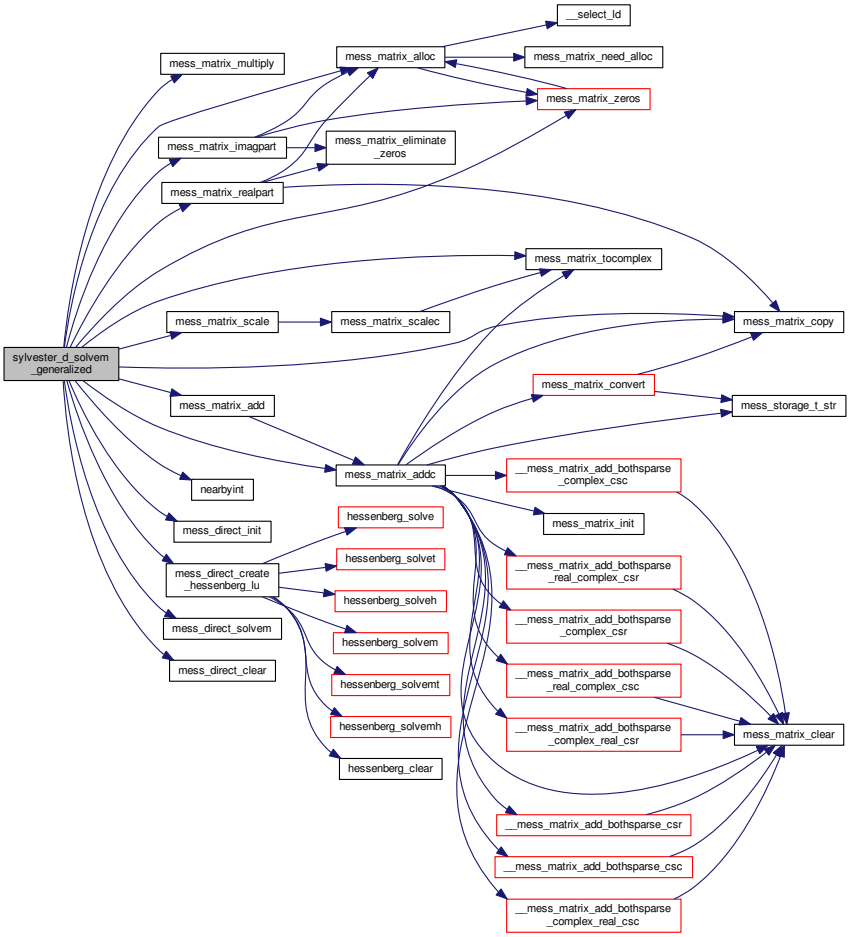
The sylvester_d_solvem_generalized function solves dense Sylvester equations:

- $AXF + EXH + M = 0$

Attention

Internal use only.

Here is the call graph for this function:



10.674.3.3 static int sylvester_d_solvem_standard (void * *datain*, mess_matrix *M*, mess_matrix *X*) [static]

Parameters

in	*datain*	input solver data
in	*M*	input right hand sides
out	*X*	output solution

Returns

zero on success or a non-zero error value otherwise

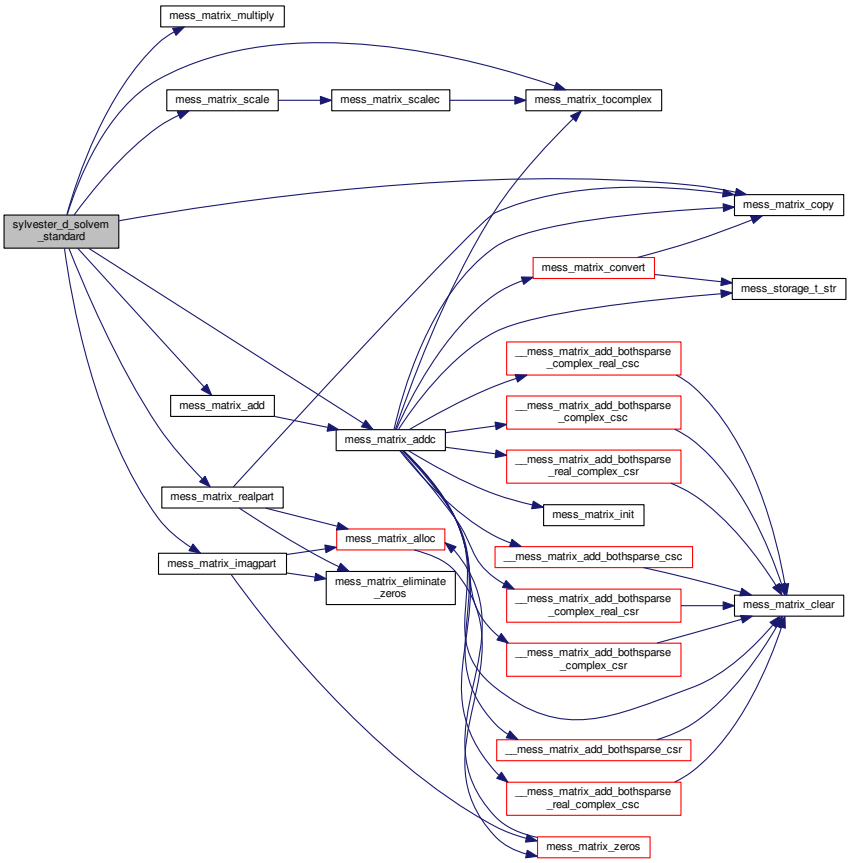
The sylvester_d_solvem_standard function solves dense Sylvester equations:

- $AX + XH + M = 0$

Attention

Internal use only.

Here is the call graph for this function:



10.674.3.4 static int sylvester_d_solvemh_generalized (void * *datain*, mess_matrix *M*, mess_matrix *X*) [static]

Parameters

in	*datain*	input solver data
in	*M*	input right hand sides
out	*X*	output solution

Returns

zero on success or a non-zero error value otherwise

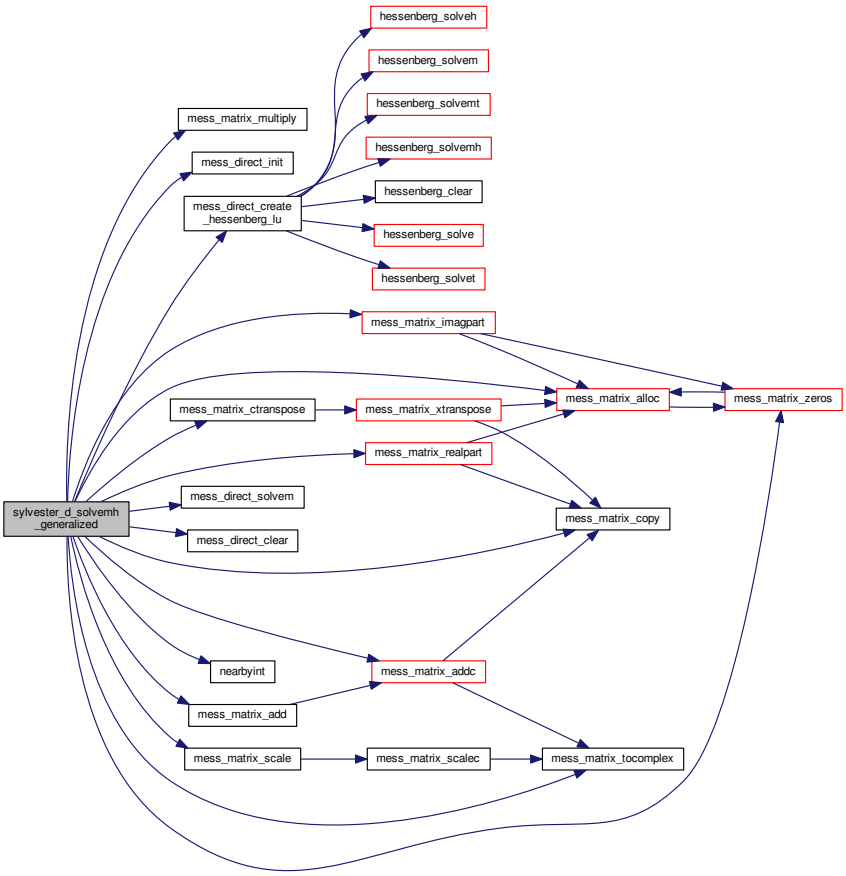
The sylvester_d_solvemh_generalized function solves dense Sylvester equations:

- $A^H X F^H + E^H X H^H + M = 0$

Attention

Internal use only.

Here is the call graph for this function:



10.674.3.5 static int sylvester_d_solvemh_standard (void * *datain*, mess_matrix *M*, mess_matrix *X*) [static]

Parameters

in	*datain*	input solver data
in	*M*	input right hand sides
out	*X*	output solution

Returns

zero on success or a non-zero error value otherwise

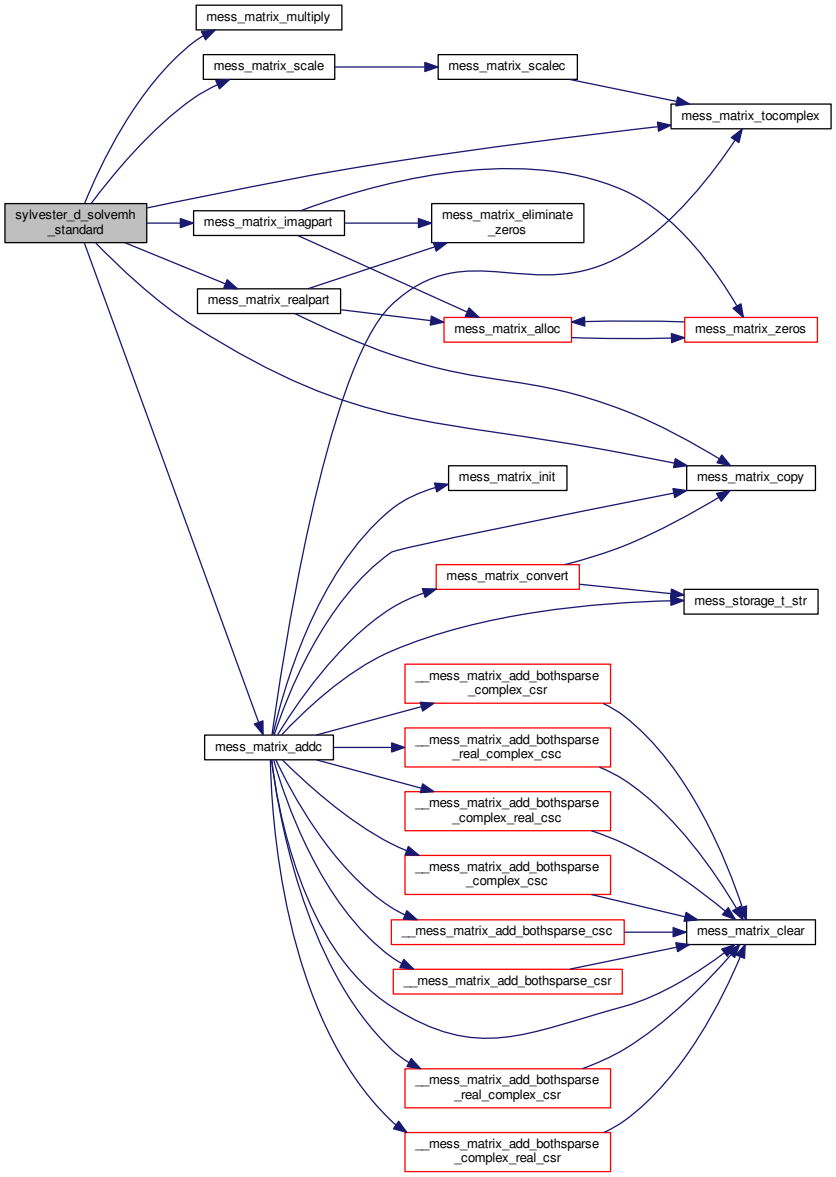
The sylvester_d_solvemh_standard function solves dense Sylvester equations:

- $A^H X + X H^H + M = 0$

Attention

Internal use only.

Here is the call graph for this function:



10.674.3.6 static int sylvester_d_solvemt_generalized (void * *datain*, mess_matrix *M*, mess_matrix *X*) [static]

Parameters

in	*datain*	input solver data
in	*M*	input right hand sides
out	*X*	output solution

Returns

zero on success or a non-zero error value otherwise

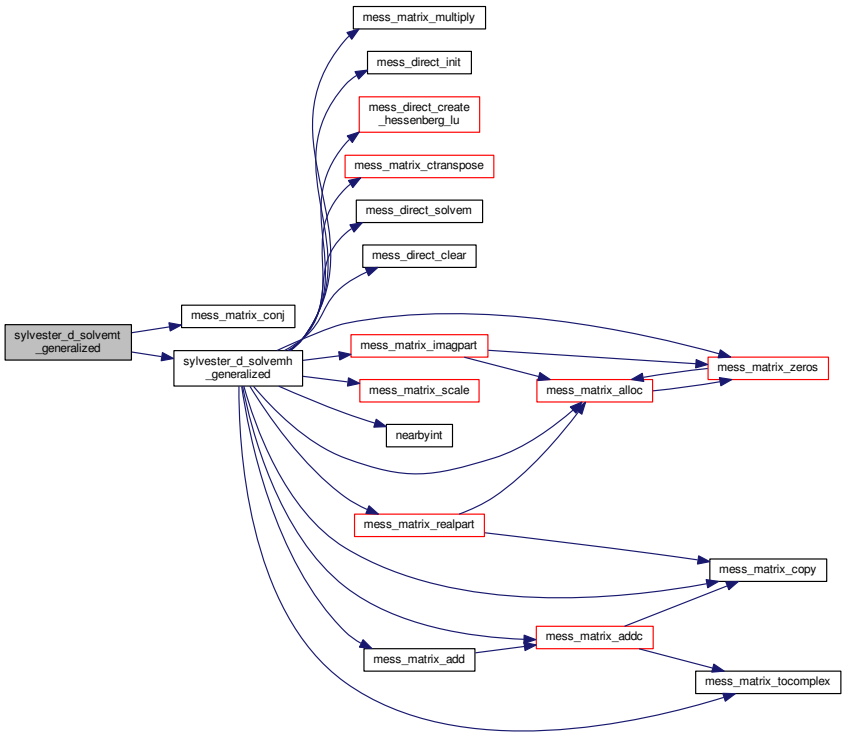
The sylvester_d_solvemt_generalized function solves dense Sylvester equations:

- $A^T X F^T + E^T X H^T + M = 0$

Attention

Internal use only.

Here is the call graph for this function:



10.674.3.7 static int sylvester_d_solvemt_standard (void * *datain*, mess_matrix *M*, mess_matrix *X*) [static]

Parameters

in	*datain*	input solver data
in	*M*	input right hand sides
out	*X*	output solution

Returns

zero on success or a non-zero error value otherwise

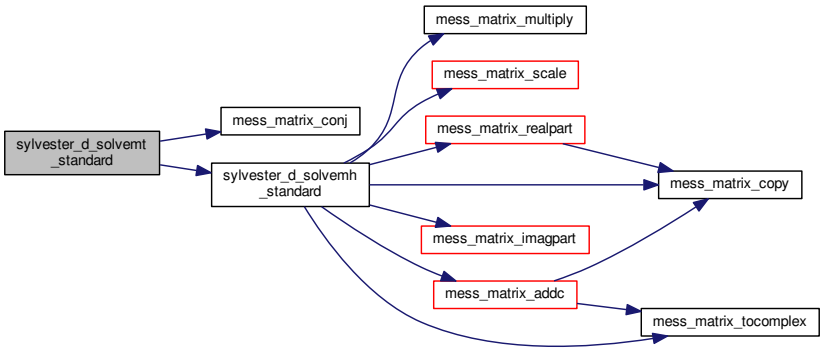
The sylvester_d_solvemt_standard function solves dense Sylvester equations:

- $A^T X + X H^T + M = 0$

Attention

Internal use only.

Here is the call graph for this function:



10.675 sylvester_dense.c File Reference

Functions

- void test_sylvester_info (mess_direct sylv, mess_operation_t op, mess_matrix A, mess_matrix F, mess_matrix E, mess_matrix H, mess_matrix M, double res2, double res2_0)

  Print information about test instance.$.
- int main (int argc, char ∗∗argv)

10.675.1 Detailed Description

- $$AX + XH + M = 0$$

  ,

  $$A^T X + X H^T + M = 0$$

  ,

  $$A^H X + X H^H + M = 0$$

- $$AXF + EXH + M = 0$$

  ,

  $$A^T X F^T + E^T X H^T + M = 0$$

  ,

  $$A^T X F^H + E^H X H^H + M = 0$$

correctly.
All matrix A, E, F, H, M must be dense.

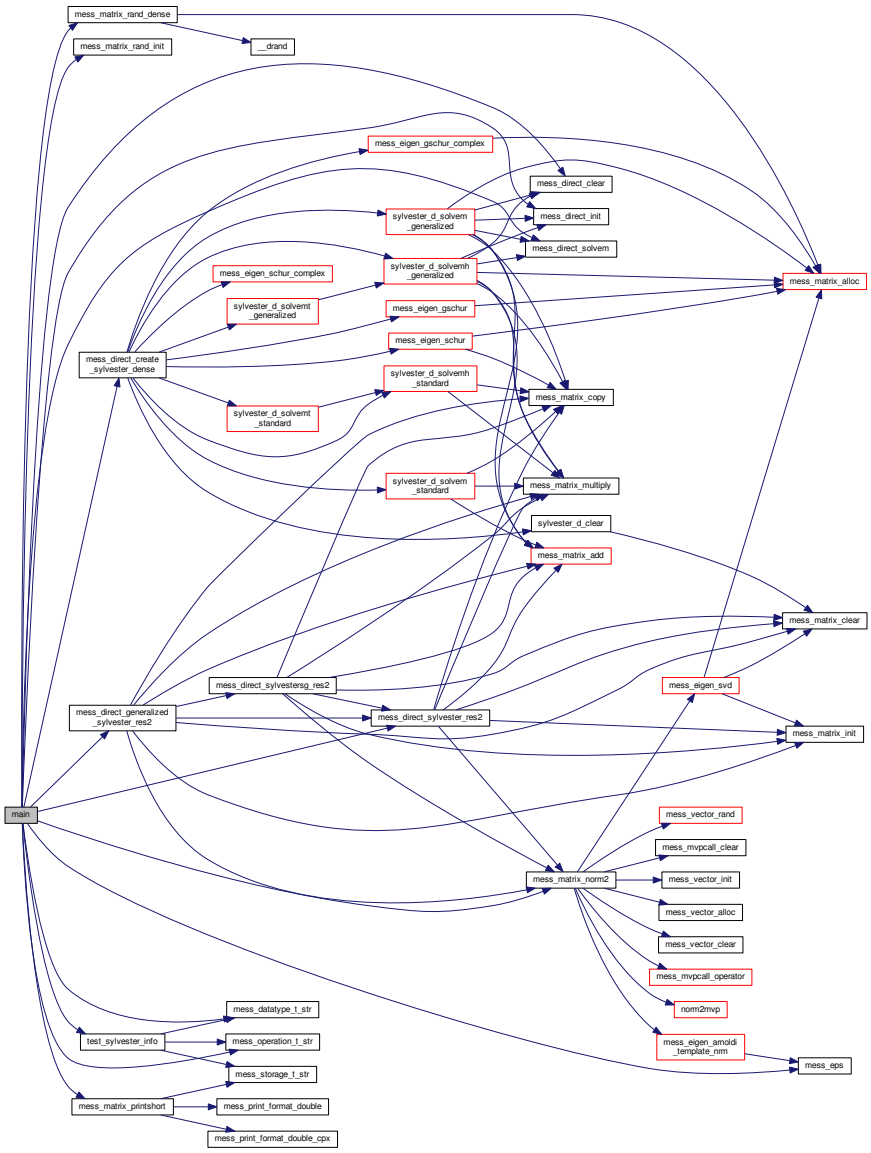
Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.675.2 Function Documentation

10.675.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:

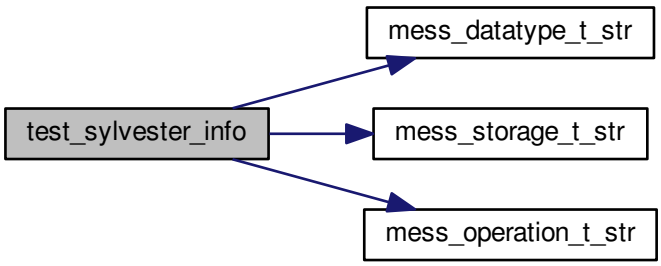


10.675.2.2 void test_sylvester_info (mess_direct *sylv*, mess_operation_t *op*, mess_matrix *A*, mess_matrix *F*, mess_matrix *E*, mess_matrix *H*, mess_matrix *M*, double *res2*, double *res2_0*)

Attention

Internal use only.

Here is the call graph for this function:



10.676 sylvester_sparsedense.c File Reference

Data Structures

- struct _sylv_solver_sd_st

  Internal structure for sparse-dense Sylvester equation solver.

Typedefs

- typedef struct _sylv_solver_sd_st _sylv_solver_sd

  Internal structure for sparse-dense Sylvester equation solver.

Enumerations

Functions

- static int sylvester_sd_clear (void ∗solver)

  Clear a _sylv_solver_sd struct.
- static int sylvester_sd_solvem (void ∗datain, mess_matrix M, mess_matrix X)

  Solve $AX + XH + M = 0$, $AX + EXH + M = 0$, $AXF + EXH + M = 0$.
- static int sylvester_sd_solvemh (void ∗datain, mess_matrix M, mess_matrix X)

  Solve $A^H X + XH^H + M = 0$, $A^H X + E^H XH^H + M = 0$, $A^H XF^H + E^H XH^H + M = 0$.
- static int sylvester_sd_solvemt (void ∗datain, mess_matrix M, mess_matrix X)

  Solve $A^T X + XH^T + M = 0$, $A^T X + E^T XH^T + M = 0$, $A^T XF^T + E^T XH^T + M = 0$.
- int mess_direct_create_sylvester_sparsedense (mess_matrix A, mess_matrix F, mess_matrix E, mess_matrix H, mess_direct solver)

  Generate a solver for a sparse-dense Sylvester equation with a small second matrix.

10.676.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Robert Dykstra (dykstra@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This code implements a sparse-dense Sylvester equation solver for the following Sylvester equations:

- Standard: $AX + XH + M = 0$, $A^T X + XH^T + M = 0$, $A^H X + XH^H + M = 0$
- Half Generalized Sylvester: $AX + EXH + M = 0$, $A^T X + E^T XH^T + M = 0$, $A^H X + E^H XH^H + M = 0$
- Generalized Sylvester: $AXF + EXH + M = 0$, $A^T XF^T + E^T XH^T + M = 0$, $A^H XF^H + E^H XH^H + M = 0$

with A and E sparse and large and F and H are small and dense.

See [14] and [8] for references.

10.676.2 Typedef Documentation

10.676.2.1 typedef struct _sylv_solver_sd_st _sylv_solver_sd

Attention

Internal use only.

10.676.3 Enumeration Type Documentation

10.676.3.1 enum _sylv_equation_sd_t

The _sylv_equation_sd_t enumeration is used to represent different sparse-dense Sylvester equations.

Attention

Internal use only.

Enumerator

SYLV_SPARSE_DENSE_STANDARD Represents a Sylvester Equation $AX + XH + M = 0$.

SYLV_SPARSE_DENSE_HALF_GENERALIZED Represents a Sylvester Equation $AX + EXH + M = 0$.

SYLV_SPARSE_DENSE_GENERALIZED Represents a Sylvester Equation $AXF + EXH + M = 0$.

10.676.4 Function Documentation

10.676.4.1 static int sylvester_sd_clear (void * *solver*) [static]

Parameters

in,out	*solver*	solver data

Returns

always zero

The sylvester_sd_clear function is clears a _sylv_solver_sd.

Attention

Internal use only.

Here is the call graph for this function:

10.676.4.2 static int sylvester_sd_solvem (void * *datain*, mess_matrix *M*, mess_matrix *X*) [static]

Parameters

in	*datain*	input solver data
in	*M*	input right hand sides dense matrix.
out	*X*	solution

Returns

zero on success or a non-zero error value otherwise

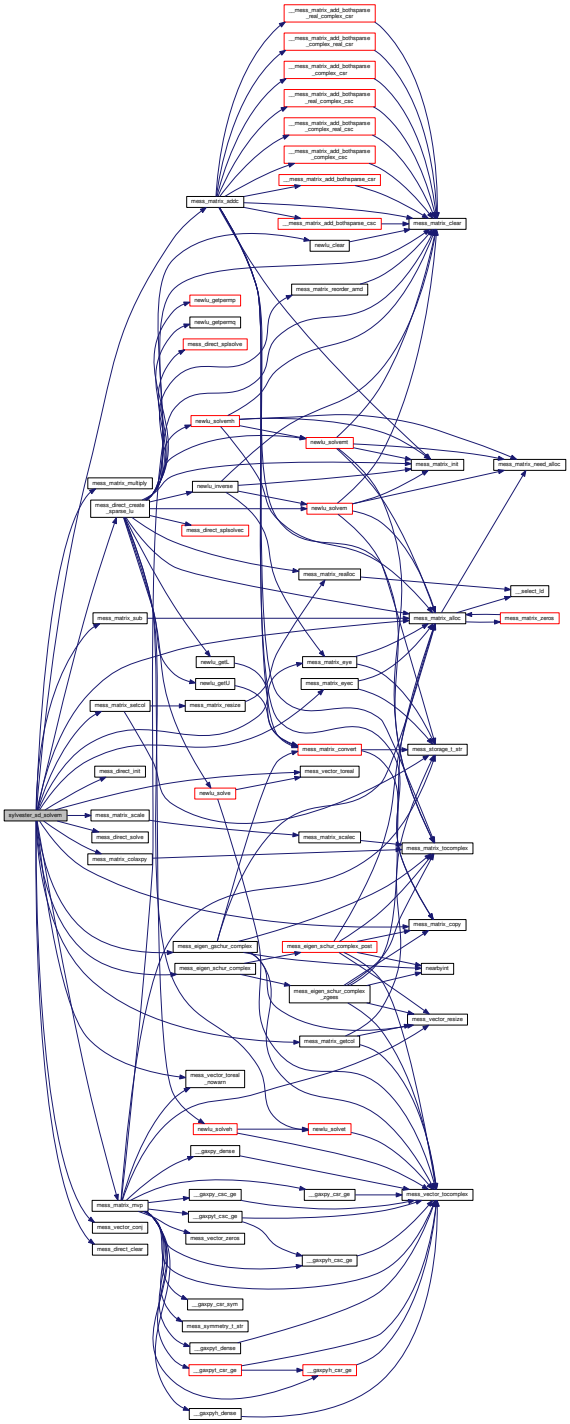
The sylvester_sd_solvem function solves sparse-dense Sylvester equations:

- $AX + XH + M = 0$
- $AX + EXH + M = 0$
- $AXF + EXH + M = 0$

Attention

Internal use only.

Here is the call graph for this function:



10.676.4.3 static int sylvester_sd_solvemh (void * *datain*, mess_matrix *M*, mess_matrix *X*) [static]

Parameters

in	*datain*	input solver data
in	*M*	input right hand sides dense matrix.
out	*X*	solution

Returns

zero on success or a non-zero error value otherwise

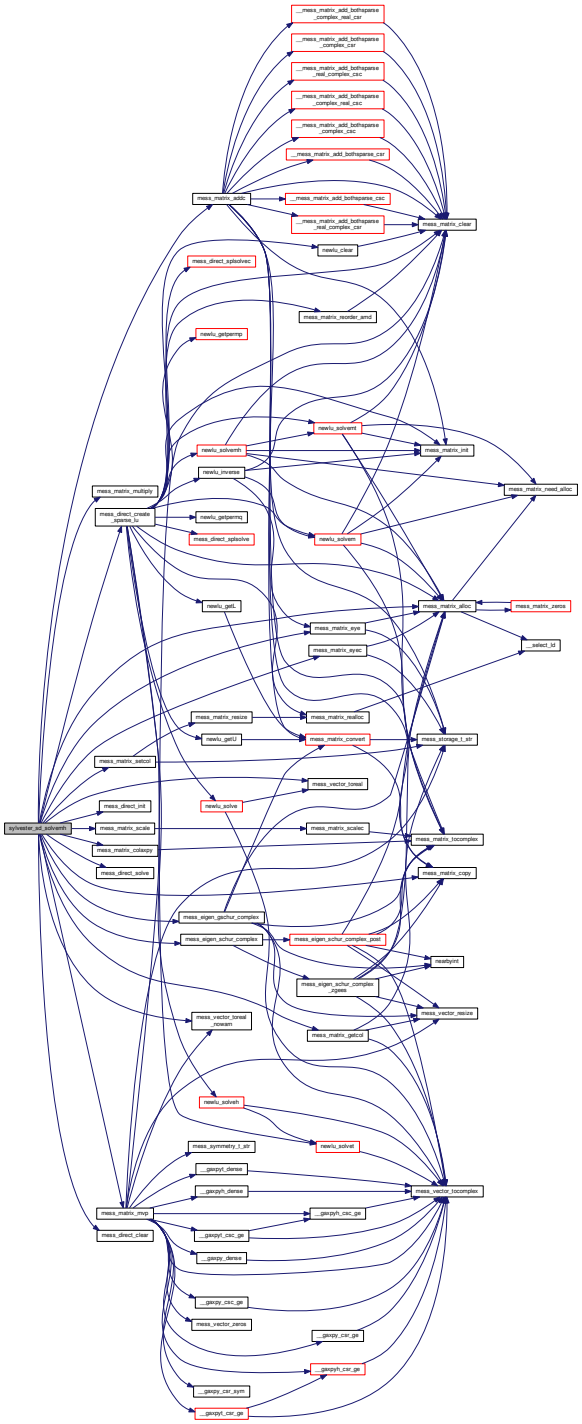
The sylvester_sd_solvemh function solves sparse-dense Sylvester equations:

- $A^H X + X H^H + M = 0$
- $A^H X + E^H X H^H + M = 0$
- $A^H X F^H + E^H X H^H + M = 0$

Attention

Internal use only.

Here is the call graph for this function:



10.676.4.4 static int sylvester_sd_solvemt (void * *datain*, mess_matrix *M*, mess_matrix *X*) [static]

Parameters

in	*datain*	input solver data
in	*M*	input right hand sides dense matrix.
out	*X*	solution

Returns

zero on success or a non-zero error value otherwise

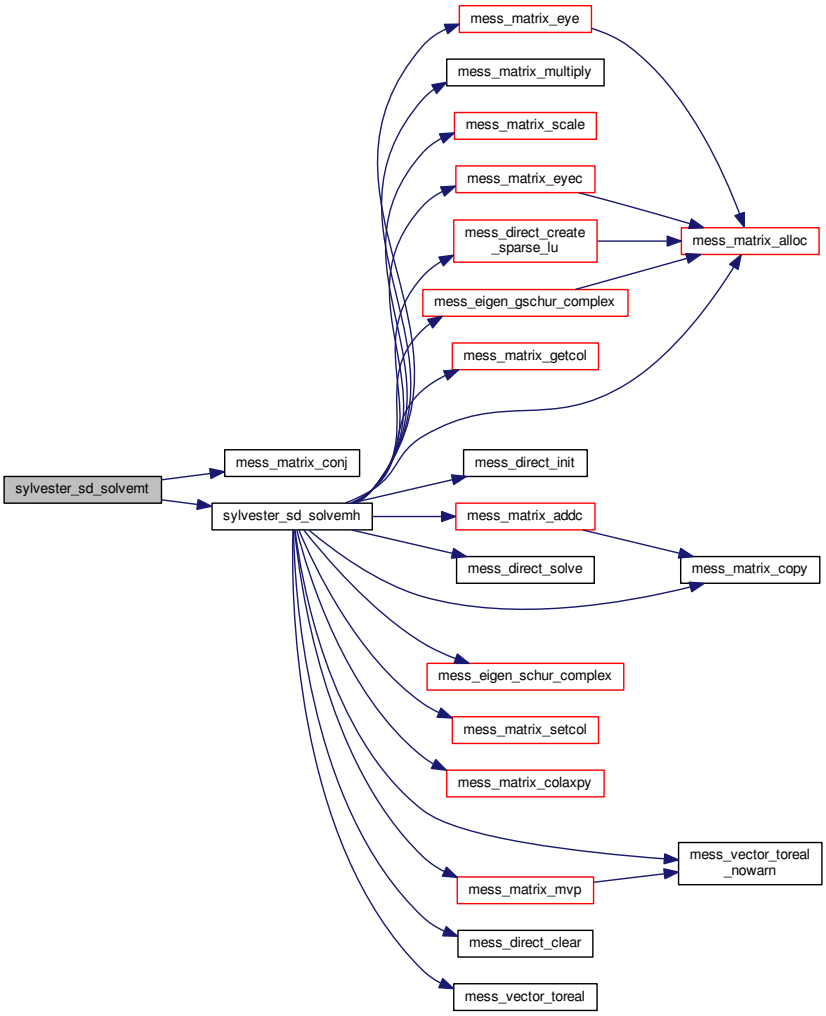
The sylvester_sd_solvemt function solves sparse-dense Sylvester equations:

- $A^T X + X H^T + M = 0$
- $A^T X + E^T X H^T + M = 0$
- $A^T X F^T + E^T X H^T + M = 0$

Attention

Internal use only.

Here is the call graph for this function:



10.677 sylvester_sparsedense.c File Reference

Functions

- PyObject ∗ pymess_sylvester_sparsedense (PyObject ∗self, PyObject ∗args)

  Wrapper around the interface mess_direct_create_sylvester_sparsedense for use with `Python`.

10.677.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Robert Dykstra (dykstra@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.678 sylvester_sparsedense.c File Reference

Functions

- void test_sylvester_info (mess_direct sylv, mess_operation_t op, mess_matrix A, mess_matrix F, mess_matrix E, mess_matrix H, mess_matrix M, double res2, double res2_0)

  Print information about test instance.$.
- int main (int argc, char ∗∗argv)

10.678.1 Detailed Description

- $$AX + XH + M = 0$$

  ,

  $$A^T X + XH^T + M = 0$$

  ,

  $$A^H X + XH^H + M = 0$$
- $$AX + EXH + M = 0$$

  ,

  $$A^T X + E^T XH^T + M = 0$$

  ,

  $$A^T X + E^H XH^H + M = 0$$
- $$AXF + EXH + M = 0$$

  ,

  $$A^T XF^T + E^T XH^T + M = 0$$

  ,

  $$A^T XF^H + E^H XH^H + M = 0$$

are solved correctly with a solver created by the mess_direct_create_sylvester_sparsedense.
A, E is a sparse or dense matrix and F, H is a small and dense matrix.

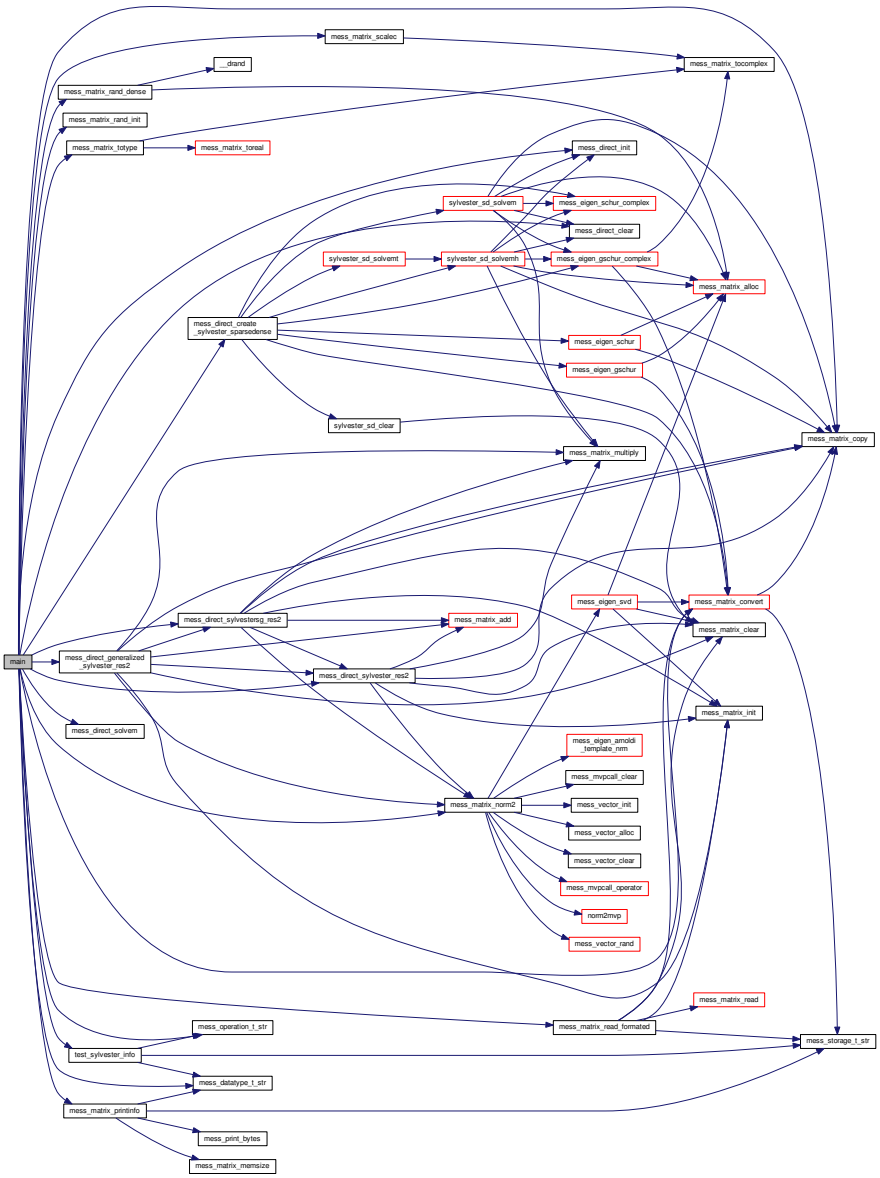
Author

Robert Dykstra (dykstra@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.678.2 Function Documentation

10.678.2.1 int main (int *argc*, char ∗∗ *argv*)

Here is the call graph for this function:

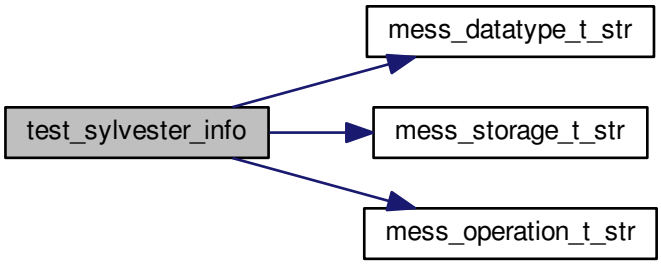


10.678.2.2 void test_sylvester_info (mess_direct *sylv*, mess_operation_t *op*, mess_matrix *A*, mess_matrix *F*, mess_matrix *E*, mess_matrix *H*, mess_matrix *M*, double *res2*, double *res2_0*)

Attention

Internal use only.

Here is the call graph for this function:



10.679 TargetDirectories.txt File Reference

10.680 TargetDirectories.txt File Reference

10.681 TargetDirectories.txt File Reference

10.682 TargetDirectories.txt File Reference

10.683 test.c File Reference

Functions

- PyObject * pymess_test_matrix (PyObject *self, PyObject *args)

  Test functions that can be called from `Python` *to ensure that all the conversions are working.*
- PyObject * pymess_test_vector (PyObject *self, PyObject *args)

  Test functions that can be called from `Python` *to ensure that all the conversions are working.*
- PyObject * pymess_test_equation (PyObject *self, PyObject *args)

10.683.1 Detailed Description

Author

Nitin Prasad (`nitin.prasad93@gmail.com`, MPI Magdeburg, CSC)
Maximilian Behr (`behr@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.684 TestRestrict.c File Reference

Functions

- int foo (int ∗TESTRESTRICTDEF i, int ∗TESTRESTRICTDEF j)
- int main (int argc, char ∗argv[ ])

10.684.1 Function Documentation

10.684.1.1 int foo (int ∗TESTRESTRICTDEF *i*, int ∗TESTRESTRICTDEF *j*)

10.684.1.2 int main (int *argc*, char ∗ *argv[ ]*)

Here is the call graph for this function:

10.685 threadpool.c File Reference

Data Structures

- struct th_done_entry

Functions

- static char ∗ th_ht_getname (csc_ds_object_t obj)
- static void th_ht_free (csc_ds_object_t obj)
- static size_t hash (const char ∗name, size_t size)
- int mess_threadpool_init (mess_threadpool ∗pool, int threads, int max_size)

  Create a loader pool.
- void ∗ mess_threadpool_worker (void ∗arg)

  Implementation of the loader thread.
- int mess_threadpool_clear (mess_threadpool ∗pool)

  Stop and clean up a loader pool.
- int mess_threadpool_insert (mess_threadpool pool, mess_threadpooljob job, mess_int_t ∗jobid)

  Insert a task in a loader pool.
- int mess_threadpooljob_init (mess_threadpooljob ∗job)

  Create a job for a loader pool.
- int mess_threadpooljob_clear (mess_threadpooljob ∗job)

  Stop and clean up a loader pool job.
- int mess_threadpool_isdone (mess_threadpool pool, mess_int_t id)

  Check if a job is done.
- int mess_threadpool_waitdone (mess_threadpool pool, mess_int_t id)

  Wait until a job is done.

10.685.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.685.2 Function Documentation

10.685.2.1 static size_t hash (const char * *name*, size_t *size*) [static]

10.685.2.2 static void th_ht_free (csc_ds_object_t *obj*) [static]

10.685.2.3 static char* th_ht_getname (csc_ds_object_t *obj*) [static]

10.686 threadpool.h File Reference

Data Structures

- struct mess_threadpooljob_st

  Representation of a thread pool job.
- struct mess_threadpool_st

  Representation of a thread pool.

Typedefs

- typedef struct mess_threadpooljob_st * mess_threadpooljob

  Type definition of a thread pool job.
- typedef struct mess_threadpool_st * mess_threadpool

  Type definition of a thread pool.

Functions

- int mess_threadpool_init (mess_threadpool *pool, int threads, int max_size)

  Create a loader pool.
- void * mess_threadpool_worker (void *arg)

  Implementation of the loader thread.
- int mess_threadpool_clear (mess_threadpool *pool)

  Stop and clean up a loader pool.
- int mess_threadpool_insert (mess_threadpool pool, mess_threadpooljob job, mess_int_t *jobid)

  Insert a task in a loader pool.
- int mess_threadpool_isdone (mess_threadpool pool, mess_int_t id)

  Check if a job is done.
- int mess_threadpool_waitdone (mess_threadpool pool, mess_int_t id)

  Wait until a job is done.
- int mess_threadpooljob_init (mess_threadpooljob *job)

  Create a job for a loader pool.
- int mess_threadpooljob_clear (mess_threadpooljob *job)

  Stop and clean up a loader pool job.

10.686.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.687 time.c File Reference

Macros

- #define __USE_POSIX199309
- #define MESS_ERROR

Functions

- double mess_wtime ()

  Return a timer value.
- double mess_ctime ()

  Get the current CPU time.
- int mess_timer_init (mess_timer *timer)

  Initialize a timer object.
- int mess_timer_clear (mess_timer *timer)

  Clean up a mess_timer object.
- double mess_timer_get (mess_timer timer)

  Get current elasped time since starting the timer.
- double mess_timer_getandreset (mess_timer timer)

  Get elapsed time since starting the timer and reset.
- double mess_timer_diff (mess_timer timerStart, mess_timer timerEnd)

  Get difference between two timers.

10.687.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file contains some nice functions for easy time measuring. If your system has POSIX-RT capabilities, they will be used.

10.687.2 Macro Definition Documentation

10.687.2.1 #define __USE_POSIX199309

10.687.2.2 #define MESS_ERROR

10.688 trace.c File Reference

Functions

- int mess_matrix_trace (mess_matrix A, double *tr)

  Compute the trace of a real matrix.
- int mess_matrix_tracec (mess_matrix A, mess_double_cpx_t *tr)

  Compute the trace of a complex matrix.
- int mess_matrix_fro_inn (mess_operation_t op, mess_matrix A, mess_matrix B, void *fro)

  Compute frobenius inner product of two dense matrices.

10.688.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Robert Dykstra (dykstra@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.689 transfer.c File Reference

Functions

- static mess_int_t ∗ pinv (mess_int_t const ∗p, mess_int_t n)

  Invert a permutation.
- int mess_dynsys_evaltransfer (mess_dynsys sys, double aa, double bb, mess_int_t nsample, mess_vector omega, mess_vector G, mess_matrix ∗Gout)

  Evaluate a transfer function.
- int mess_dynsys_transfer_diff (mess_int_t nsample, mess_matrix ∗Go, mess_matrix ∗Gr, double ∗err2, double ∗rel2, double ∗errinf, double ∗relinf, mess_vector diffvec)

  Compute difference and relative difference between two transfer functions.

10.689.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.689.2 Function Documentation

10.689.2.1 static mess_int_t∗ pinv (mess_int_t const ∗ *p*, mess_int_t *n*) [static]

Parameters

in	*p*	input permutation
in	*n*	input length of the permutation

Returns

inverted permutation

The pinv function returns the inverted permution of p.

10.690 transpose.c File Reference

Functions

- int mess_matrix_xtranspose (mess_matrix A, mess_matrix AT, int hermitian)

  Transpose a matrix.
- int mess_matrix_ctranspose (mess_matrix A, mess_matrix AH)

  Transpose a matrix.
- int mess_matrix_transpose (mess_matrix A, mess_matrix AT)

  Transpose a matrix.

10.690.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.691 tridiag.c File Reference

Functions

- int mess_matrix_tridiag (mess_matrix matrix, mess_int_t rows, mess_int_t cols, mess_storage_t store_t, mess_datatype_t data_t, mess_double_cpx_t lower, mess_double_cpx_t diag, mess_double_cpx_t upper)

  Create a tridiagonal matrix.

10.691.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.692 trisolve.c File Reference

Functions

- int mess_solver_lusolvem_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *lval, mess_int_t *lcolptr, mess_int_t *lrowptr, mess_double_cpx_t *uval, mess_int_t *ucolptr, mess_int_t *urowptr, mess_int_t *p, mess_int_t *q, mess_matrix b, mess_matrix x)

  LU solver for complex CSR stored L and U.
- int mess_solver_lsolve_kernelcsr_real (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, double *y)

  Solve $Lx = b$ (CSR, kernel only, real).
- int mess_solver_lsolve_kernelcsr_real_complex (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $Lx = b$ (CSR, kernel only, real L, complex b).
- int mess_solver_lsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $Lx = b$ (CSR, kernel only, complex).
- int mess_solver_lsolve (mess_matrix L, mess_vector y)

  Solve $Lx = b$ (user interface).
- int mess_solver_lsolvem (mess_matrix L, mess_matrix Y)

  Solve $LX = B$ (user interface).
- int mess_solver_usolve_kernelcsr_real (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, double *y)

  Solve $Ux = b$ (CSR, kernel only, real).
- int mess_solver_usolve_kernelcsr_real_complex (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $Ux = b$ (CSR, kernel only, real U, complex y).
- int mess_solver_usolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $Ux = b$ (CSR, kernel only, complex).

- int mess_solver_usolve (mess_matrix U, mess_vector y)

  Solve $Ux = b$ (user interface).
- int mess_solver_usolvem (mess_matrix U, mess_matrix Y)

  Solve $UX = B$ (user interface).
- int mess_solver_ltsolve_kernelcsr_real (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, double *y)

  Solve $L^T x = b$ (CSR, kernel only, real).
- int mess_solver_ltsolve_kernelcsr_real_complex (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $L^T x = b$ (CSR, kernel only, real L, complex y).
- int mess_solver_ltsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $L^T x = b$ (CSR, kernel only, complex).
- int mess_solver_lhsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $L^H x = b$ (CSR, kernel only, complex).
- int mess_solver_lcsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $\bar{L}x = b$ (CSR, kernel only, complex).
- int mess_solver_ucsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *y)

  Solve $\bar{U}x = b$ (CSR, kernel only, complex).
- int mess_solver_ltsolve (mess_matrix L, mess_vector y)

  Solve $L^T x = b$ (user interface).
- int mess_solver_lhsolve (mess_matrix L, mess_vector y)

  Solve $L^H x = b$ (user interface).
- int mess_solver_ltsolvem (mess_matrix L, mess_matrix Y)

  Solve $L^T X = B$ (user interface).
- int mess_solver_lhsolvem (mess_matrix L, mess_matrix Y)

  Solve $L^H X = B$ (user interface).
- int mess_solver_utsolve_kernelcsr_real (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, double *x)

  Solve $U^T x = b$ (CSR, kernel only, real).
- int mess_solver_utsolve_kernelcsr_real_complex (mess_int_t dim, double *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *x)

  Solve $U^T x = b$ (CSR, kernel only, real U, complex y).
- int mess_solver_utsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *x)

  Solve $U^T x = b$ (CSR, kernel only, complex).
- int mess_solver_uhsolve_kernelcsr_complex (mess_int_t dim, mess_double_cpx_t *values, mess_int_t *colptr, mess_int_t *rowptr, mess_double_cpx_t *x)

  Solve $U^H x = b$ (CSR, kernel only, complex).
- int mess_solver_utsolve (mess_matrix U, mess_vector y)

  Solve $U^T x = b$ (user interface).
- int mess_solver_uhsolve (mess_matrix U, mess_vector y)

  Solve $U^H x = b$ (user interface).
- int mess_solver_utsolvem (mess_matrix U, mess_matrix Y)

  Solve $U^T X = B$ (user interface).
- int mess_solver_uhsolvem (mess_matrix U, mess_matrix Y)

  Solve $U^H X = B$ (user interface).

10.692.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.692.2 Function Documentation

10.692.2.1 int mess_solver_lcsolve_kernelcsr_complex (mess_int_t *dim*, mess_double_cpx_t * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_double_cpx_t * *y*)

Parameters

in	*dim*	input row dimension of L
in	*values*	input values of L
in	*colptr*	input column pointer for L
in	*rowptr*	input row pointer for L
in,out	*y*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_lcsolve_kernelcsr_complex function solves the system

$$\bar{L}x = b$$

where L is a complex lower triangular matrix stored in Compressed Row Storage.

Attention

Internal use only.

10.692.2.2 int mess_solver_ucsolve_kernelcsr_complex (mess_int_t *dim*, mess_double_cpx_t * *values*, mess_int_t * *colptr*, mess_int_t * *rowptr*, mess_double_cpx_t * *y*)

Parameters

in	*dim*	input row dimension of U
in	*values*	input values of U
in	*colptr*	input column pointer for U
in	*rowptr*	input row pointer for U
in,out	*y*	on input: right hand side b on output: solution x

Returns

zero on success or a non zero error code otherwise

The mess_solver_ucsolve_kernelcsr_complex function solves the system

$$\bar{U}x = b$$

where U is a complex upper triangular matrix stored in Compressed Row Storage.

Attention

Internal use only.

10.693 triul.c File Reference

Functions

- int mess_matrix_triu (mess_matrix mat, mess_int_t k)

  Upper triangular part of mess_matrix.
- int mess_matrix_tril (mess_matrix mat, mess_int_t k)

  Lower triangular part of mess_matrix.

10.693.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.694 tsia.c File Reference

Macros

- #define MAT_INIT(A) ret = mess_matrix_init(&(A)); FUNCTION_FAILURE_HANDLE(ret,(ret!=0), mess_matrix_init);

Functions

- int mess_h2_tsia (mess_dynsys orig, mess_vector sigmaEX, mess_h2_options opt, mess_dynsys redu, mess_matrix V, mess_matrix W, mess_h2_status status)

  Compute a $\mathcal{H}_2$ reduced model via the TSIA algorithm.

10.694.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.694.2 Macro Definition Documentation

10.694.2.1 #define MAT_INIT(A) ret = mess_matrix_init(&(A)); FUNCTION_FAILURE_HANDLE(ret,(ret!=0), mess_matrix_init);

10.695 tsiag.c File Reference

Macros

- #define MAT_INIT(A) ret = mess_matrix_init(&(A)); FUNCTION_FAILURE_HANDLE(ret,(ret!=0), mess_matrix_init);

Functions

- int mess_h2_tsiag (mess_dynsys orig, mess_vector sigmaEX, mess_h2_options opt, mess_dynsys redu, mess_matrix V, mess_matrix W, mess_h2_status status)

  Compute a $\mathcal{H}_2$ reduced model via the TSIA algorithm.

10.695.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.695.2 Macro Definition Documentation

10.695.2.1 #define MAT_INIT(A) ret = mess_matrix_init(&(A)); FUNCTION_FAILURE_HANDLE(ret,(ret!=0), mess_matrix_init);

10.696 tutorial_backslash.c File Reference

10.696.1 Detailed Description

See also

mess_matrix_backslash

Tutorial: Use mess_matrix_backslash to solve a linear system.

This function demonstrates how to solve a linear system

$$Ax = b$$

or

$$A^T x = b$$

using a backslash operator.

```
#include <stdio.h>
#include <stdlib.h>
#include "mess/mess.h"

int main ( int argc, char **argv) {
    mess_matrix A;
    mess_vector x,x2,b,b2;

    mess_version();

    /*---------------------------------------------------------------------------
     *  check input arguments
     *---------------------------------------------------------------------------*/
    if ( argc != 2 ) {
        printf("Usage: %s file\n", argv[0]);
        return 1;
    }
    mess_set_errorlevel(3);

    /*---------------------------------------------------------------------------
     *  init and read matrix
     *---------------------------------------------------------------------------*/
    MESS_INIT_MATRICES(&A);
    mess_matrix_read_formated(argv[1],A,MESS_DENSE);

    /*---------------------------------------------------------------------------
     *  init vectors
     *---------------------------------------------------------------------------*/
    MESS_INIT_VECTORS(&x,&x2,&b,&b2);
    mess_vector_alloc(x,  A->cols, A->data_type);
    mess_vector_alloc(x2, A->rows, A->data_type);
    mess_vector_alloc(b,  A->rows, A->data_type);
    mess_vector_alloc(b2, A->cols, A->data_type);

    /*---------------------------------------------------------------------------
     *  create right hand side
     *---------------------------------------------------------------------------*/
    mess_matrix_rowsums(A,b);
    mess_matrix_colsums(A,b2);

    /*---------------------------------------------------------------------------
     *  solve systems
     *---------------------------------------------------------------------------*/
    mess_matrix_backslash(MESS_OP_NONE, A, b, x);
    mess_matrix_backslash(MESS_OP_TRANSPOSE,A, b2, x2);

    /*---------------------------------------------------------------------------
     *  print solution should be [1,..,1]
     *---------------------------------------------------------------------------*/
    printf("Ax=b:\n");
    mess_vector_printshort(x);
    printf("A^Tx=b:\n");
    mess_vector_printshort(x2);

    /*---------------------------------------------------------------------------
     *  clear data
     *---------------------------------------------------------------------------*/
    MESS_CLEAR_MATRICES(&A);
    MESS_CLEAR_VECTORS(&x,&x2,&b,&b2);

    return 0;
}
```

10.697 tutorial_bt.c File Reference

10.697.1 Detailed Description

Author

mbehr
koehlerm

Tutorial: Balanced Truncation Model Order Reduction of LTI System

In this Tutorial we show how **M.E.S.S.** is used for Model Order Reduction using Balanced Truncation.

Therefore, we consider a large scale generalized LTI system:

$$\begin{array}{rcll} E\dot{x}(t) & = & Ax(t) & +Bu(t) \quad , \\ y(t) & = & Cx(t) & \quad , \end{array}$$

where $E,\ A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{n\times p}$ and $C \in \mathbb{R}^{q\times n}$. Furthermore, the matrix E is nonsingular and the pencil $(A,\ E)$ is stable. For this LTI system we compute factors of the Controllability Gramian $Z_C Z_C^T \approx X_C$ and the Observability Gramian $Z_O Z_O^T \approx X_O$, which are the solutions of the generalized Lyapunov Equations

$$\begin{array}{rrcl} AX_CE^T+ & EX_CA^T & = & -BB^T \quad , \\ A^T\hat{X_O}E+ & E^T\hat{X_O}A & = & -CC^T \quad , \\ & X_O & = & E^T\hat{X_O}E \quad . \end{array}$$

Using this factors we compute the SVD [23] of

$$U_O \Sigma U_C^T = Z_O^T Z_C$$

to get the balancing tranformation matrices

$$S_C = Z_C U_C(:,1:k)\Sigma(1:k,1:k)^{-\frac{1}{2}} \quad \text{and} \quad S_O = Z_O U_O(:,1:k)\Sigma(1:k,1:k)^{-\frac{1}{2}},$$

where k is selected such that the first k singluar values in Σ fulfill a given threshold. Using the transformation matrices S_C and S_O we compute the reduced order model

$$E := I,\ A_r := S_O^T E^{-1} A S_C,\ B_r := S_O^T B,\ \text{and } C_r = C S_C.$$

The threshold for the selection of k using the following approximation for the $\mathcal{H}\infty$-error:

$$\|H - H_r\|_\infty < 2 \sum_{i=k+1}^{n} \sigma_i < \tau,$$

where τ is a given upper bound for the error and σ_i are the singular values contained in Σ.

1. Include Header Files

We include standard C header files and in addtion the `mess.h` header file to make the **M.E.S.S.** library available.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <math.h>
#include "mess/mess.h"
```

2. Declare Matices, Dynamical Systems and Options Structures

We use the mess_matrix structure to declare our matrices A, B, C, E. Furthermore, we need the matrices S_C and S_O containing the left and the right projection matrix of the Balanced Truncation process. The balanced truncation process needs options for solving the Lyapunov equation and for the truncation process. These are represendted by mess_options, mess_bt_options and mess_bt_status. Because the four matrices A, B, C, E represent an LTI system we need two intances of mess_dynsys for the original and the reduced order model.

```
mess_matrix A, B, C, E,SO, SC;
mess_options adiopt;
mess_bt_options btopt;
mess_bt_status  btstat;
mess_dynsys lti, red;
double tol;
mess_int_t maxr;
```

By using the `argc` variable we can check the number of input arguments and set the error level to 1 to removing debug and warnings from the output.

```
if ( argc != 7 ) {
    printf("usage: %s E.mtx A.mtx B.mtx C.mtx tol maxr\n", argv[0]);
    return 0;
}
mess_set_errorlevel(1);
```

3. Init Matrices and Read Data

Before we are able to use the matrices we have to init the fields of the mess_matrix structure variables using mess↩_matrix_init and read the matrices from `MatrixMarket` file format. The remaining command line parameters are used for the tolerance τ and the maximum order of the reduced order model.

```
mess_matrix_init(&A);
mess_matrix_init(&B);
mess_matrix_init(&C);
mess_matrix_init(&E);

mess_matrix_read_formated(argv[1],E,
MESS_CSR);
mess_matrix_read_formated(argv[2],A,
MESS_CSR);
mess_matrix_read_formated(argv[3],B,
MESS_DENSE);
mess_matrix_read_formated(argv[4],C,
MESS_DENSE);

tol  = atof(argv[5]);
maxr = atoi(argv[6]);
```

4. Converting the Input Matrices into a (Generalized) LTI System

Because the matrices represent a generalized LTI system we have to transfer them into a mess_dynsys object. This either be done using a flat copy by mess_dynsys_glti or as a deep copy by mess_dynsys_glti_copy. Deep copy means that the matrices a copied into the object and not only referenced. Otherwise if one changes the matrix A it will be change in the mess_dynsys object as well.

```
mess_dynsys_init(&lti);
mess_dynsys_glti_copy(lti, A, B, C, E);

printf("Dimension: " MESS_PRINTF_INT "\n", A->rows);
printf("Inputs:    " MESS_PRINTF_INT "\n", B->cols);
printf("Outputs:   " MESS_PRINTF_INT "\n", C->rows);
```

5. Setup the Balanced Truncation and Compute S_C and S_O

We set the options to the Balanced Truncation process and compute the transformation matrices. The mess_bt↩_gslrsrm computes transformation matrices such that the input is a generalized LTI system and the reduced order model will be a standard state space system. The function internally solves the two Lyapunov equations described above with a low-rank solver after wards it computes the SVD of the product of both low rank factors and selects the order k of the reduced order model by the singular values of the SVD as described above. Furthermore, a maximum oder is defined in the Balanced Truncation options. If the matrix E is the identity the mess_bt_lrsrm function can be used. If the reduced order model should have a matrix which is not the identity one has to use mess_bt_gglrsrm.

```
        mess_bt_status_init(&btstat);
        mess_bt_options_init(&btopt);
        btopt->rdim =  maxr;
        btopt->tol  = tol;

        printf("\nBalanced Truncation Options:\n");
        mess_bt_options_print(btopt);
        printf("\n");

        mess_options_init(&adiopt);
        adiopt->adi_output = 0;
        adiopt->adi_shifts_paratype =
MESS_LRCFADI_PARA_MINMAX;

        mess_matrix_init(&SO);
        mess_matrix_init(&SC);

        mess_bt_gslrsrm(lti,btopt, adiopt, SO,SC, btstat);

        printf("Balanced Truncation Status:\n");
        mess_bt_status_print(btstat);
        printf("\n");
```

6. Compute the Reduced Order Model using S_C and S_O

After computing the transformation matrices S_C and S_O the reduced order model

$$\begin{array}{rcll} \dot{x_r}(t) & = & A_r x_r(t) & +B_r u(t) \quad , \\ y_r(t) & = & C_r x_r(t) & \quad , \end{array}$$

is computed by

$$A_r := S_O^T E^{-1} A S_C, \; B_r := S_O^T B, \text{ and } C_r = C S_C.$$

```
        double time_pro = mess_wtime();
        mess_dynsys_init(&red);
        mess_dynsys_project_to_lti(lti, SO,SC,red);
        time_pro = mess_wtime()-time_pro;
```

7. Compute the $\mathcal{H}_2$

Although the $\mathcal{H}_2$ norm does not correspond to the Balanced Truncation process it is a good identifier how good the reduce order model approximates the original model. Computing the distance or the error between two LTI systems is done using the mess_h2_error function.

```
        double err = 0;
        mess_h2_error(lti, red, &err);
        printf("H2 error = %lg\n", err);
```

8. Evaluate the Transferfunction of the Original and the Reduced Order Model

Because the transferfunction is a nice tool to visualize the input-output behaviour of an LTI in the frequency domain, we want to evaluate it for the original and the reduced order model in order to check if the computed model still shows the same input-output behaviour. Therefore, we sample the transferfunction of both models over a interval $\omega \in \left[10^{-1}, 10^{6}\right]$ and compute $max(svd(H(i \cdot \omega)))$. The whole procedure is done by mess_dynsys_evaltransfer.

```
double time_tf = mess_wtime();
mess_vector omega, G1, G2;
MESS_INIT_VECTORS(&omega,&G1,&G2);
mess_vector_alloc(omega,100,MESS_REAL);
mess_vector_alloc(G1, 100, MESS_REAL);
mess_vector_alloc(G2, 100, MESS_REAL);

mess_dynsys_evaltransfer(lti,-1,6,100,omega,G1,NULL);
mess_dynsys_evaltransfer(red,-1,6,100,omega,G2,NULL);

mess_vector_diffnorm(G1,G2,&err);
printf("error discrete ||G1-G2||_2 = %lg\n", err);
mess_vector_diffnorminf(G1,G2,&err);
printf("error discrete ||G1-G2||_infty = %lg\n", err);
time_tf = mess_wtime()-time_tf;
```

9. Clear memory

Print the final status and clear all matrices and data structures.

```
printf("\nTime BT: %lg\nTime Projection: %lg\nTime Transfer Function: %lg\n", btstat->
time, time_pro, time_tf);

mess_vector_clear(&omega);
mess_vector_clear(&G1);
mess_vector_clear(&G2);

mess_matrix_clear(&A);
mess_matrix_clear(&B);
mess_matrix_clear(&C);
mess_matrix_clear(&E);
mess_matrix_clear(&SO);
mess_matrix_clear(&SC);

mess_dynsys_clear(&lti);
mess_dynsys_clear(&red);

mess_options_clear(&adiopt);
mess_bt_options_clear(&btopt);
mess_bt_status_clear(&btstat);
return 0;
```

```
@sa @ref tutorials
```

10.698 tutorial_bt_gglrsrm.c File Reference

10.698.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

See also

mess_bt_gglrsrm

Tutorial: Reduce a First Order Generalized LTI System Using a Low-rank Square Root Balanced Trunction Method.

This function demonstrates how to convert a first order generalized linear time invariant (GLTI) system

$$\begin{aligned} E\dot{x} &= Ax \quad +Bu \\ y &= Cx \end{aligned}$$

to another reduced first order GLTI system using a generalized low-rank square root balanced truncation.

```
#include <stdlib.h>
#include "mess/mess.h"

#define NSAMPLE 25
#define S_LOW -2
#define S_HIGH 6


int main ( int argc, char **argv){
    mess_matrix A,B,C,E;
    mess_matrix V,W;
    mess_options adiopt;
    mess_bt_options btopt;
    mess_bt_status  btstat;
    mess_dynsys lti, red;
    mess_int_t maxr = 0 ;
    double tol = 0;
    mess_vector omega,G1, G2;

    printf("Balanced Truncation Example for generalized LTI Systems\n");
    printf("=======================================================\n");


    /*------------------------------------------------------------------------
     *  Read and init matrices
     *------------------------------------------------------------------------*/
    if ( argc != 7 ) {
        printf("usage: %s A.mtx B.mtx C.mtx E.mtx tol maxr\n", argv[0]);
        return 0;
    }

    MESS_INIT_MATRICES(&A,&B,&C,&E);

    mess_matrix_read_formated(argv[1], A, MESS_CSR);
    mess_matrix_read_formated(argv[2], B, MESS_DENSE);
    mess_matrix_read_formated(argv[3], C, MESS_DENSE);
    mess_matrix_read_formated(argv[4], E, MESS_CSR);
    tol  = atof(argv[5]);
    maxr = atoi(argv[6]);


    mess_dynsys_init(&lti);
    mess_set_errorlevel(1);


    printf("Dimension: " MESS_PRINTF_INT "\n", A->rows);
    printf("Inputs:    " MESS_PRINTF_INT "\n", B->cols);
    printf("Outputs:   " MESS_PRINTF_INT "\n", C->rows);

    /*------------------------------------------------------------------------
     *  Setup BT
     *------------------------------------------------------------------------*/
    mess_dynsys_glti_copy(lti, A, B, C, E);

    mess_options_init(&adiopt);
    mess_bt_options_init(&btopt);
    mess_bt_status_init(&btstat);
    printf("maxr_: " MESS_PRINTF_INT "\n", maxr);
    btopt->rdim =  maxr;
    btopt->tol = tol;

    adiopt->adi_output = 0;

    MESS_INIT_MATRICES(&V,&W);
    printf("ADI options\n");
    mess_options_print(adiopt);
    printf("\nBT options\n");
    mess_bt_options_print(btopt);


    /*------------------------------------------------------------------------
     *  Compute the Projection Matrices
     *------------------------------------------------------------------------*/
```

```
    mess_bt_gglrsrm(lti,btopt, adiopt, V,W, btstat);

    printf("\nStatus of mess_bt_lrsrm:\n");
    mess_bt_status_print(btstat);
    printf("\n");

    /*-----------------------------------------------------------------------------
     *  Compute the reduced order model
     *-----------------------------------------------------------------------------*/
    mess_dynsys_init(&red);
    mess_dynsys_project_to_glti(lti, V,W,red);

    /*-----------------------------------------------------------------------------
     *  Evaluate the Transferfunction
     *-----------------------------------------------------------------------------*/
    MESS_INIT_VECTORS(&omega,&G1,&G2);
    mess_vector_alloc(omega,100,MESS_REAL);
    mess_vector_alloc(G1, 100, MESS_REAL);
    mess_vector_alloc(G2, 100, MESS_REAL);

    int i;
    mess_matrix *Go,*Gr;
    Gr = malloc (sizeof(mess_matrix)*NSAMPLE);
    Go = malloc (sizeof(mess_matrix)*NSAMPLE);
    for ( i = 0 ; i < NSAMPLE; i++){
        mess_matrix_init(&Gr[i]);
        mess_matrix_init(&Go[i]);
    }
    mess_dynsys_evaltransfer(lti,S_LOW,S_HIGH,NSAMPLE,omega,G1,Go);
    mess_dynsys_evaltransfer(red,S_LOW,S_HIGH,NSAMPLE,omega,G2,Gr);


    double err2, rel2, errinf, relinf;
    mess_vector diff;
    MESS_INIT_VECTORS(&diff);
    mess_vector_alloc(diff, NSAMPLE, MESS_REAL);
    mess_dynsys_transfer_diff(NSAMPLE, Go, Gr,&err2, &rel2, &errinf, &relinf, diff
      );
    for ( i = 0; i < NSAMPLE; i++){
        mess_matrix_clear(&Gr[i]);
        mess_matrix_clear(&(Go[i]));
    }
    free(Go); Go=NULL;
    free(Gr); Gr=NULL;

    printf("error discrete ||G1-G2||_2           = %lg\n", err2);
    printf("error discrete ||G1-G2||_2 rel.      = %lg\n", rel2);
    printf("error discrete ||G1-G2||_infty       = %lg\n", errinf);
    printf("error discrete ||G1-G2||_infty rel.  = %lg\n", relinf);

    if(err2 > 1e-4)  { printf("FAILED:error discrete ||G1-G2||_2 to large.\n"); return 1; }
    if(rel2 > 1e-8)  { printf("FAILED:error discrete ||G1-G2||_2_rel to large.\n"); return 1; }
    if(errinf > 1e-5){ printf("FAILED:error discrete ||G1-G2||_2_infty to large.\n"); return 1; }
    if(relinf > 1e-8){ printf("FAILED:error discrete ||G1-G2||_2_infty rel. to large.\n"); return 1; }

    /*-----------------------------------------------------------------------------
     *  Clean up
     *-----------------------------------------------------------------------------*/
    MESS_CLEAR_VECTORS(&G1,&G2,&omega,&diff);
    MESS_CLEAR_MATRICES(&A,&B,&C,&E,&V,&W);

    mess_dynsys_clear(&lti);
    mess_dynsys_clear(&red);
    mess_options_clear(&adiopt);
    mess_bt_options_clear(&btopt);
    mess_bt_status_clear(&btstat);
    return 0;
}
```

10.699 tutorial_bt_gslrsrm.c File Reference

10.699.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

See also

mess_bt_gslrsrm

Tutorial: Reduce a First Order Generalized LTI System to a First Order LTI System Using a Low-rank Square Root Balanced Truncation Method

This function demonstrates how to convert a first order generalized linear time invariant (GLTI) system

$$\begin{aligned} E\dot{x} &= Ax \quad +Bu \\ y &= Cx \end{aligned}$$

to a reduced first order linear time invariant (LTI) system

$$\begin{aligned} \dot{x} &= Ax \quad +Bu \\ y &= Cx \end{aligned}$$

using a generalized low-rank square root balanced truncation.

```
#include <stdlib.h>
#include <stdarg.h>
#include "mess/mess.h"

int main ( int argc, char **argv){
    mess_matrix A,B,C,E;
    mess_matrix V,W;
    mess_options adiopt;
    mess_bt_options btopt;
    mess_bt_status  btstat;
    mess_dynsys lti, red;
    mess_int_t maxr = 0 ;
    double tol = 0, err1, err2;
    mess_vector omega,G1, G2;

    printf("Balanced Truncation Example for generalized LTI Systems\n");
    printf("=====================================================\n");


    /*---------------------------------------------------------------------------
     *  Read matrices
     *---------------------------------------------------------------------------*/
    if ( argc != 7 ) {
        printf("usage: %s A.mtx B.mtx C.mtx E.mtx tol maxr\n", argv[0]);
        return 0;
    }

    MESS_INIT_MATRICES(&A,&B,&C,&E);

    mess_matrix_read_formated(argv[1], A, MESS_CSR);
    mess_matrix_read_formated(argv[2], B, MESS_DENSE);
    mess_matrix_read_formated(argv[3], C, MESS_DENSE);
    mess_matrix_read_formated(argv[4], E, MESS_CSR);
    tol  = atof(argv[5]);
    maxr = atoi(argv[6]);

    mess_dynsys_init(&lti);
    mess_set_errorlevel(1);

    printf("Dimension: " MESS_PRINTF_INT "\n", A->rows);
    printf("Inputs:    " MESS_PRINTF_INT "\n", B->cols);
    printf("Outputs:   " MESS_PRINTF_INT "\n\n", C->rows);

    /*---------------------------------------------------------------------------
     *  Setup BT
     *---------------------------------------------------------------------------*/
    mess_dynsys_glti_copy(lti, A, B, C, E);

    mess_options_init(&adiopt);
    mess_bt_options_init(&btopt);
    mess_bt_status_init(&btstat);
    printf("maxr_: " MESS_PRINTF_INT "\n", maxr);
    btopt->rdim =  maxr;
    btopt->tol = tol;

    adiopt->adi_output = 0;

    mess_matrix_init(&V);
    mess_matrix_init(&W);
    printf("ADI options\n");
    mess_options_print(adiopt);
    printf("\nBT options\n");
    mess_bt_options_print(btopt);

    /*---------------------------------------------------------------------------
```

```
     *  Compute the Projection Matrices
     *-----------------------------------------------------------------------------*/
    mess_bt_gslrsrm(lti,btopt, adiopt, V,W, btstat);

    printf("\nStatus of mess_bt_lrsrm:\n");
    mess_bt_status_print(btstat);
    printf("\n");


    /*-----------------------------------------------------------------------------
     *  Compute the reduced order model
     *-----------------------------------------------------------------------------*/
    mess_dynsys_init(&red);
    mess_dynsys_project_to_lti(lti, V,W,red);

    /*-----------------------------------------------------------------------------
     *  Evaluate the Transferfunctions
     *-----------------------------------------------------------------------------*/
    MESS_INIT_VECTORS(&omega,&G1,&G2);
    mess_vector_alloc(omega,100,MESS_REAL);
    mess_vector_alloc(G1, 100, MESS_REAL);
    mess_vector_alloc(G2, 100, MESS_REAL);

    mess_dynsys_evaltransfer(lti,-1,6,100,omega,G1,NULL);
    mess_dynsys_evaltransfer(red,-1,6,100,omega,G2,NULL);

    mess_vector_diffnorm(G1,G2,&err1);
    mess_vector_diffnorminf(G1,G2,&err2);
    printf("error discrete ||G1-G2||_2 = %lg\n", err1);
    printf("error discrete ||G1-G2||_infty = %lg\n", err2);

    if(err2 > 1e-4)  { printf("FAILED:error discrete ||G1-G2||_2 to large.\n"); return 1; }
    if(err2 > 1e-5){ printf("FAILED:error discrete ||G1-G2||_2_infty to large.\n"); return 1; }

    /*-----------------------------------------------------------------------------
     *  Clean up
     *-----------------------------------------------------------------------------*/
    MESS_CLEAR_VECTORS(&G1,&G2,&omega);

    MESS_CLEAR_MATRICES(&A,&B,&C,&E,&V,&W);
    mess_dynsys_clear(&lti);
    mess_dynsys_clear(&red);
    mess_options_clear(&adiopt);
    mess_bt_options_clear(&btopt);
    mess_bt_status_clear(&btstat);
    return 0;
}
```

10.700 tutorial_bt_lrsrm.c File Reference

10.700.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

See also

mess_bt_lrsrm

Tutorial: Low-rank Square Root Balanced Trunctation First Order LTI to First Order LTI System

This function demonstrates how to convert a first order linear time invariant (LTI) system

$$\begin{aligned} \dot{x} &= Ax \quad +Bu \\ y &= Cx \end{aligned}$$

to another reduced first order LTI system using a low-rank square root balanced truncation.

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "mess/mess.h"


int main ( int argc, char **argv){
    mess_matrix A,B,C;
    mess_matrix V,W;
    mess_options adiopt;
    mess_bt_options btopt;
    mess_bt_status  btstat;
    mess_dynsys lti, red;
    mess_int_t maxr = 0 ;
    double tol = 0, err1, err2;
    mess_vector omega,G1, G2;


    printf("Balanced Truncation Example for LTI Systems\n");
    printf("==========================================\n");


    /*---------------------------------------------------------------------------
     *  Read Matrices
     *---------------------------------------------------------------------------*/
    if ( argc != 6 ) {
        printf("usage: %s A.mtx B.mtx C.mtx tol maxr\n", argv[0]);
        return 0;
    }
    mess_init();

    mess_matrix_init(&A);
    mess_matrix_init(&B);
    mess_matrix_init(&C);

    mess_set_errorlevel(1);

    mess_matrix_read_formated(argv[1], A, MESS_CSR);
    mess_matrix_read_formated(argv[2], B, MESS_DENSE);
    mess_matrix_read_formated(argv[3], C, MESS_DENSE);

    tol  = atof(argv[4]);
    maxr = atoi(argv[5]);

    printf("Dimension: " MESS_PRINTF_INT "\n", A->rows);
    printf("Inputs:    " MESS_PRINTF_INT "\n", B->cols);
    printf("Outputs:   " MESS_PRINTF_INT "\n\n", C->rows);

    /*---------------------------------------------------------------------------
     *  Setup BT
     *---------------------------------------------------------------------------*/

    mess_dynsys_init(&lti);
    mess_dynsys_lti_copy(lti, A, B, C);

    mess_options_init(&adiopt);
    mess_bt_options_init(&btopt);
    mess_bt_status_init(&btstat);
    btopt->rdim =  maxr;
    btopt->tol = tol;

    adiopt->adi_output = 0;
    mess_matrix_init(&V);
    mess_matrix_init(&W);
    printf("BT options\n");
    mess_bt_options_print(btopt);

    /*---------------------------------------------------------------------------
     *  Compute the Projection Matrices
     *---------------------------------------------------------------------------*/
    mess_bt_lrsrm(lti,btopt, adiopt, V,W, btstat);

    printf("\nStatus of mess_bt_lrsrm:\n");
    mess_bt_status_print(btstat);
    printf("\n");


    /*---------------------------------------------------------------------------
     *  Compute the Reduced Order Model
     *---------------------------------------------------------------------------*/
    mess_dynsys_init(&red);
    mess_dynsys_project_to_lti(lti, V,W,red);

    /*---------------------------------------------------------------------------
     *  Evaluate the Transferfunctions
     *---------------------------------------------------------------------------*/
    MESS_INIT_VECTORS(&omega,&G1,&G2);
```

```
    mess_vector_alloc(omega,100,MESS_REAL);
    mess_vector_alloc(G1, 100, MESS_REAL);
    mess_vector_alloc(G2, 100, MESS_REAL);

    mess_dynsys_evaltransfer(lti,-1,6,100,omega,G1,NULL);
    mess_dynsys_evaltransfer(red,-1,6,100,omega,G2,NULL);

    mess_vector_diffnorm(G1,G2,&err1);
    mess_vector_diffnorminf(G1,G2,&err2);
    printf("error discrete ||G1-G2||_2 = %lg\n", err1);
    printf("error discrete ||G1-G2||_infty = %lg\n", err2);


    /*--------------------------------------------------------------------------
     *  Clean up
     *--------------------------------------------------------------------------*/

    mess_vector_clear(&G1);
    mess_vector_clear(&G2);
    mess_vector_clear(&omega);

    mess_matrix_clear(&A);
    mess_matrix_clear(&B);
    mess_matrix_clear(&C);
    mess_matrix_clear(&V);
    mess_matrix_clear(&W);
    mess_dynsys_clear(&lti);
    mess_dynsys_clear(&red);
    mess_options_clear(&adiopt);
    mess_bt_options_clear(&btopt);
    mess_bt_status_clear(&btstat);
    return 0;
}
```

10.701 tutorial_bt_lrsrm_somor.c File Reference

10.701.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

See also

mess_bt_lrsrm_somor

Tutorial: Low-rank Square Root Balanced Trunctation Second Order LTI to First Order LTI System

This function demonstrates how to convert a second order linear time invariant (LTI) system

$$\begin{array}{rcl} M\ddot{x}(t) + G\dot{x}(t) + \quad Kx(t) & = & Bu(t) \\ y(t) & = & C_p x(t) \quad + C_v \dot{x}(t) \end{array}$$

to a reduced first order LTI system

$$\begin{array}{rl} \dot{x} & = Ax \quad + Bu \\ y & = Cx \end{array}$$

using a low-rank square root balanced truncation.

```
#include <stdlib.h>
#include <stdarg.h>
#include "mess/mess.h"

int main ( int argc, char **argv){
    mess_matrix M,K,G,B,Cv,Cp;
    mess_matrix V,W;
    mess_options adiopt;
    mess_bt_options btopt;
    mess_bt_status  btstat;
```

```
    mess_dynsys sys, red;
    mess_int_t maxr = 0 ;
    double tol = 0, err1, err2;
    mess_vector omega,G1, G2;

    printf("Balanced Truncation Example for second order LTI Systems\n");
    printf("=======================================================\n");

    mess_set_errorlevel(1);

    /*-----------------------------------------------------------------------
     *  Read matrices
     *-----------------------------------------------------------------------*/
    if ( argc != 9 ) {
        printf("usage: %s M.mtx G.mtx K.mtx B.mtx Cp.mtx Cv.mtx tol maxr\n", argv[0]);
        return 0;
    }
    mess_init();


    mess_matrix_init(&M);
    mess_matrix_init(&G);
    mess_matrix_init(&K);
    mess_matrix_init(&B);
    mess_matrix_init(&Cv);
    mess_matrix_init(&Cp);
    mess_dynsys_init(&sys);
    mess_dynsys_init(&red);

    mess_matrix_read_formated(argv[1], M, MESS_CSR);
    mess_matrix_read_formated(argv[2], G, MESS_CSR);
    mess_matrix_read_formated(argv[3], K, MESS_CSR);
    mess_matrix_read_formated(argv[4], B, MESS_DENSE);
    mess_matrix_read_formated(argv[5], Cp, MESS_DENSE);
    mess_matrix_read_formated(argv[6], Cv, MESS_DENSE);

    tol = atof(argv[7]);
    maxr = atoi(argv[8]);

    printf("Dimension: " MESS_PRINTF_INT "\n", M->rows);
    printf("Inputs:    " MESS_PRINTF_INT "\n", B->cols);
    printf("Outputs:   " MESS_PRINTF_INT "\n\n", Cp->rows);


    /*-----------------------------------------------------------------------
     *  Setup BT
     *-----------------------------------------------------------------------*/
    mess_dynsys_2nd(sys,M,G,K,B,Cp,Cv);

    mess_options_init(&adiopt);
    mess_bt_options_init(&btopt);
    mess_bt_status_init(&btstat);
    btopt->rdim =  maxr;
    btopt->tol = tol;
    adiopt->adi_output = 0;
    adiopt->adi_maxit = 100;

    mess_matrix_init(&V);
    mess_matrix_init(&W);
    printf("BT options\n");
    mess_bt_options_print(btopt);


    /*-----------------------------------------------------------------------
     *  Compute the Projection matrices
     *-----------------------------------------------------------------------*/
    mess_bt_lrsrm_somor(sys,btopt, adiopt, V,W, btstat);

    printf("\nStatus of mess_bt_lrsrm_somor:\n");
    mess_bt_status_print(btstat);
    printf("\n");


    /*-----------------------------------------------------------------------
     *  Compute the Reduced Order Model
     *-----------------------------------------------------------------------*/
    mess_dynsys_project_2nd_to_1st(sys, V,W,red);


    /*-----------------------------------------------------------------------
     *  Evaluate the Transferfunction
     *-----------------------------------------------------------------------*/
#define NSAMPLE 200

    MESS_INIT_VECTORS(&omega,&G1,&G2);
    mess_vector_alloc(omega,NSAMPLE,MESS_REAL);
    mess_vector_alloc(G1, NSAMPLE, MESS_REAL);
```

```
    mess_vector_alloc(G2, NSAMPLE, MESS_REAL);

    mess_dynsys_evaltransfer(sys,0,5,NSAMPLE,omega,G1,NULL);
    mess_dynsys_evaltransfer(red,0,5,NSAMPLE,omega,G2,NULL);

    mess_vector_diffnorm(G1,G2,&err1);
    mess_vector_diffnorminf(G1,G2,&err2);

    printf("error discrete ||G1-G2||_2 = %lg\n", err1);
    printf("error discrete ||G1-G2||_infty = %lg\n", err2);


    /*------------------------------------------------------------------------
     *  Cleanup
     *------------------------------------------------------------------------*/
    mess_vector_clear(&G1);
    mess_vector_clear(&G2);
    mess_vector_clear(&omega);

    mess_matrix_clear(&V);
    mess_matrix_clear(&W);
    mess_dynsys_clear(&sys);
    mess_dynsys_clear(&red);
    mess_options_clear(&adiopt);
    mess_bt_options_clear(&btopt);
    mess_bt_status_clear(&btstat);
    mess_exit();
    return 0;
}
```

10.702 tutorial_bt_lrsrm_somor_so.c File Reference

10.702.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

See also

mess_bt_lrsrm_somor_so_gram

Tutorial: Reduce a Second Order LTI to a Second Order LTI System Using a Low-rank Square Root Balanced Trunction Method

This function demonstrates how to convert a second order linear time invariant (LTI) system

$$\begin{array}{rcl} M\ddot{x}(t) + G\dot{x}(t) + \quad Kx(t) & = & Bu(t) \\ y(t) & = & C_p x(t) \quad + C_v \dot{x}(t) \end{array}$$

to a another reduced second order LTI system using a low-rank square root balanced truncation.

```
#include <stdlib.h>
#include <stdarg.h>
#include "mess/mess.h"

#define NSAMPLE 200

int main ( int argc, char **argv){
    mess_matrix M,K,G,B,Cv,Cp;
    mess_matrix *Go, *Gr;
    mess_matrix V,W;
    mess_options adiopt;
    mess_bt_options btopt;
    mess_bt_status  btstat;
    mess_dynsys sys, red;
    mess_int_t maxr = 0 ;
    double tol = 0;
    mess_vector omega,G1, G2;
    mess_int_t i;
    double err2, rel2, errinf, relinf;
```

```
mess_vector diff;

printf("mess - BT LRSRM\n");
printf("================\n");

if ( argc != 9 ) {
    printf("usage: %s M.mtx G.mtx K.mtx B.mtx Cp.mtx Cv.mtx tol maxr\n", argv[0]);
    return 0;
}

/*-----------------------------------------------------------------------------
 *  Init Data
 *-----------------------------------------------------------------------------*/

mess_matrix_init(&M);
mess_matrix_init(&G);
mess_matrix_init(&K);
mess_matrix_init(&B);
mess_matrix_init(&Cv);
mess_matrix_init(&Cp);
mess_dynsys_init(&sys);
mess_matrix_init(&V);
mess_matrix_init(&W);
mess_dynsys_init(&red);

mess_set_errorlevel(1);


/*-----------------------------------------------------------------------------
 *  Read Data
 *-----------------------------------------------------------------------------*/

mess_matrix_read_formated(argv[1], M, MESS_CSR);
mess_matrix_read_formated(argv[2], G, MESS_CSR);
mess_matrix_read_formated(argv[3], K, MESS_CSR);
mess_matrix_read_formated(argv[4], B, MESS_DENSE);
mess_matrix_read_formated(argv[5], Cp, MESS_DENSE);
mess_matrix_read_formated(argv[6], Cv, MESS_DENSE);
tol = atof(argv[7]);
maxr = atoi(argv[8]);

printf("Dimension: " MESS_PRINTF_INT "\n", M->rows);
printf("Inputs:    " MESS_PRINTF_INT "\n", B->cols);
printf("Outputs:   " MESS_PRINTF_INT "\n\n", Cp->rows);

mess_dynsys_2nd(sys,M,G,K,B,Cp,Cv);


/*-----------------------------------------------------------------------------
 *  Evaluate original transferfunction
 *-----------------------------------------------------------------------------*/
MESS_INIT_VECTORS(&omega,&G1);
mess_vector_alloc(omega,NSAMPLE,MESS_REAL);
mess_vector_alloc(G1, NSAMPLE, MESS_REAL);
Go = malloc (sizeof(mess_matrix)*NSAMPLE);
for ( i = 0 ; i < NSAMPLE; i++){
    mess_matrix_init(&Go[i]);
}
mess_dynsys_evaltransfer(sys,-2,4,NSAMPLE,omega,G1,Go);

/*-----------------------------------------------------------------------------
 *  setup BT
 *-----------------------------------------------------------------------------*/
mess_bt_options_init(&btopt);
mess_bt_status_init(&btstat);
mess_options_init(&adiopt);
adiopt->adi_output = 0 ;

btopt->rdim =  maxr;
btopt->tol = tol;
btopt->so_type = MESS_BT_POSITION_VELOCITY;

btopt->chooseorder = mess_bt_chooseorder_default;

/*-----------------------------------------------------------------------------
 *  compute V and W
 *-----------------------------------------------------------------------------*/
printf("\nBT options:\n");
mess_bt_options_print(btopt);
mess_bt_lrsrm_somor_so(sys,btopt,adiopt,  V,W, btstat);
printf("\nStatus of mess_bt_lrsrm_somor:\n");
mess_bt_status_print(btstat);
printf("\n");

/*-----------------------------------------------------------------------------
 *  project the system
 *-----------------------------------------------------------------------------*/
```

```
    mess_dynsys_project_2nd_to_2nd(sys, V,W,red);

    /*-----------------------------------------------------------------------
     *  eval red. TFM
     *-----------------------------------------------------------------------*/
    MESS_INIT_VECTORS(&G2);
    mess_vector_alloc(G2, NSAMPLE, MESS_REAL);
    Gr = malloc (sizeof(mess_matrix)*NSAMPLE);
    for ( i = 0 ; i < NSAMPLE; i++){
        mess_matrix_init(&Gr[i]);
    }
    mess_dynsys_evaltransfer(red,-2,4,NSAMPLE,omega,G2,Gr);


    /*-----------------------------------------------------------------------
     *  Compute the error
     *-----------------------------------------------------------------------*/
    MESS_INIT_VECTORS(&diff);
    mess_vector_alloc(diff, NSAMPLE, MESS_REAL);
    mess_dynsys_transfer_diff(NSAMPLE, Go, Gr,&err2, &rel2, &errinf, &relinf, diff
      );
    mess_vector_clear(&diff);

    printf("error discrete ||G1-G2||_2           = %lg\n", err2);
    printf("error discrete ||G1-G2||_2 rel.      = %lg\n", rel2);
    printf("error discrete ||G1-G2||_infty       = %lg\n", errinf);
    printf("error discrete ||G1-G2||_inftyi rel. = %lg\n", relinf);


    mess_vector_clear(&G2);
    mess_matrix_clear(&V);
    mess_matrix_clear(&W);
    mess_dynsys_clear(&red);

    for ( i = 0; i < NSAMPLE; i++){
        mess_matrix_clear(&Go[i]);
        mess_matrix_clear(&Gr[i]);
    }
    free(Go); Go=NULL;
    free(Gr); Gr=NULL;


    mess_vector_clear(&G1);
    mess_vector_clear(&omega);
    mess_dynsys_clear(&sys);
    mess_options_clear(&adiopt);
    mess_bt_options_clear(&btopt);
    mess_bt_status_clear(&btstat);
    return 0;
}
```

10.703 tutorial_care.c File Reference

10.703.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Tutorial Solving a (generalized) Riccati Equation

In this tutorial we solve the (generalized) Riccati Equation, i.e. $A^TX + XA - XBB^TX + C^TC = 0$ or $A^TXE + E^TXA - E^TXBB^TXE + C^TC = 0$.
We assume that $A, E \in \mathbb{R}^{n\times n}$ is large and sparse and the spectrum $\Lambda(A) \subseteq \mathbb{C}^-$ or $\Lambda(A, E) \subseteq \mathbb{C}^-$ is contained in the left open complex half-plane. Under stability and detectability assumption theory admits existence of a unique solution of the (generalized) algebraic Riccati Equation and ensures convergence of the algorithm.
The right-hand side is defined by $C \in \mathbb{R}^{p\times n}$ and $B \in \mathbb{R}^{n\times m}$. Furthermore the number of outputs and inputs should be small compared to the size of the system $p, m \ll n$.
The solver mess_care returns the solution X as a low-rank factor approximation Z such that $X \approx ZZ^T$.
In the last step we compute the relative residual $\|A^TZZ^T + ZZ^TA - ZZ^TBB^TZZ^T + C^TC\|_2/\|C^TC\|_2$ or $\|\|A^TZZ^TE + E^TZZ^TA - E^TZZ^TBB^TZZ^TE + C^TC\|_2/\|C^TC\|_2$ and clear the memory.

1. Include Header Files

We include standard C header files and in addtion the mess.h header file to make the **M.E.S.S.** library available.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <math.h>
#include "mess/mess.h"
```

2. Init Matrices and Read Data

Before we are able to use we have to init the fields of the mess_matrix structure variables using mess_matrix_init

```
    mess_matrix_init(&A);
    mess_matrix_init(&B);
    mess_matrix_init(&C);
    if(argc==5){mess_matrix_init(&E);}
    mess_matrix_init(&Z);
```

Now we can use mess_matrix_read_formated to read our matrices from a Matrix Market File Format and convert it to the storage type MESS_CSR and MESS_DENSE.
mess_matrix_read_formated dynamical allocates memory at this point.

```
    mess_matrix_read_formated(argv[1],A,MESS_CSR);
    mess_matrix_read_formated(argv[2],B,MESS_DENSE);
    mess_matrix_read_formated(argv[3],C,MESS_DENSE);
    if(argc==5){
        mess_matrix_read_formated(argv[4],E,MESS_CSR);
    }
```

3. Solve the (Generalized) Riccati Equation

As already announced mess_care solves the (generalized) Riccati Equation. We can indicate the case of the nongeneralized Riccati Equation by providing NULL as the second argument to mess_care.

```
    if(argc==4){
        mess_care(A,NULL,B,C,Z);
    }else{
        mess_care(A,E,B,C,Z);
    }
```

4. Compute the Relative Residual

We declare two `double` variables for storing the right-hand side norm and the absolute residual. mess_matrix_dynorm2 computes the norm of the right-hand side and mess_lrcfadi_res2nm or mess_lrcfadi_res2nmg are helpfull to compute the desired 2-norms.

```
double nrmRHS=1, nrm=1;
mess_matrix_dynorm2(C,&nrmRHS);
if(argc==4){
    mess_lrcfadi_res2nm(A,B,C,Z,&nrm);
}else{
    mess_lrcfadi_res2nmg(A,B,C,E,Z,&nrm);
}
printf("relative Residual= %10.15e\n",nrm/nrmRHS);
```

5. Don't Forget to Clear Memory

A rule of thumb is that every structure which needs an `init` call before usage must be cleared at the at the end of your code. In our cases we have used mess_matrix_init which means we have to use mess_matrix_clear to clear dynamical allocated memory.

```
mess_matrix_clear(&A);
mess_matrix_clear(&B);
mess_matrix_clear(&C);
mess_matrix_clear(&Z);
if(argc==5)
    mess_matrix_clear(&E);

return nrm/nrmRHS>1e-9;
```

See also

Tutorials

10.704 tutorial_direct_solvers.c File Reference

10.704.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Tutorial About Linear Equation Systems

In this Tutorial we present functions for solving linear equation systems.

1. Include Header Files

We include standard C header files and in addtion the mess.h header file to make the **M.E.S.S.** library available.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <math.h>
#include "mess/mess.h"
```

2. Generate Sparse Random Matrix

At first we want to generate a sparse diagonal dominant matrix A, to make sure that A has full rank. We perform the following steps

$$A \leftarrow I \in \mathbb{R}^{n\times n},$$
$$A \leftarrow (n+1)\cdot A + rand \in \mathbb{R}^{n\times n}.$$

```
mess_int_t n = 1000;
double fill_rate = 0.05;
mess_matrix A,rand;
mess_matrix_init(&A);
mess_matrix_init(&rand);
mess_matrix_eye(A,n,n,MESS_CSR);
mess_matrix_rand(rand,n,n,MESS_CSR,MESS_REAL,fill_rate);
mess_matrix_add(1.0,rand,n+1,A);
```

3. Solve Sparse Linear Systems

In this section we show different functions to solve a sparse linear system.

3.1 Solve Sparse Linear Systems with mess_matrix_backslash

The simplest way to solve a linear equation system is to use mess_matrix_backslash. We generate a random right hand side $b \in \mathbb{R}^n$ and solve $Ax_0 = b$ and $A^T x_0 = b$. After that we want to check our solution x and compute the residual

$$y \leftarrow Ax,$$
$$y \leftarrow A^T x,$$
$$res \leftarrow \|y-b\|_2.$$

We also measure the wall-clock-time.

```
double res, walltime;
mess_vector b, x, y;
MESS_INIT_VECTORS(&b,&x,&y);
mess_vector_alloc(b,n,MESS_REAL);
mess_vector_alloc(x,n,MESS_REAL);
mess_vector_alloc(y,n,MESS_REAL);
mess_vector_rand(b);

walltime = mess_wtime();
mess_matrix_backslash(MESS_OP_NONE,A,b,x);
mess_matrix_mvp(MESS_OP_NONE,A,x,y);
mess_vector_diffnorm(y,b,&res);
printf("||A*x-b||_2 \t= %10.5e\n",res);

mess_matrix_backslash(MESS_OP_TRANSPOSE,A,b,x);
mess_matrix_mvp(MESS_OP_TRANSPOSE,A,x,y);
mess_vector_diffnorm(y,b,&res);
printf("||A^T*x-b||_2 \t= %10.5e\n",res);

printf("Wall Time (mess_matrix_backslash) = %10.5f\n\n",mess_wtime()-walltime);
```

3.2 Solve Sparse Linear Systems with mess_direct structure

mess_matrix_backslash and mess_matrix_backslashm recompute the decomposition of A in every call. If you want to solve a lot of systems with different right hand sides you should use mess_direct structure.

```
walltime = mess_wtime();

mess_direct solver;
mess_direct_init(&solver);
mess_direct_lu(A,solver);

mess_direct_solve(MESS_OP_NONE,solver,b,x);
mess_matrix_mvp(MESS_OP_NONE,A,x,y);
mess_vector_diffnorm(y,b,&res);
printf("||A*x-b||_2 \t= %10.5e\n",res);

mess_direct_solve(MESS_OP_TRANSPOSE,solver,b,x);
mess_matrix_mvp(MESS_OP_TRANSPOSE,A,x,y);
mess_vector_diffnorm(y,b,&res);
printf("||A^T*x-b||_2 \t= %10.5e\n",res);

mess_direct_clear(&solver);
printf("Wall Time (mess_matrix_backslash) = %10.5f\n\n",mess_wtime()-walltime);
```

At this point please compare the wall-clock-time between mess_direct_lu and and mess_matrix_backslash.

3.3 Change LU solver

You can change the direct solver with mess_direct_lu_select. The residual and relative residual is easily computed by mess_direct_res2.

```
double rel_res;

mess_direct_lu_select(MESS_DIRECT_SPARSE_LU);
mess_direct_init(&solver);
mess_direct_lu(A,solver);
mess_direct_solve(MESS_OP_NONE,solver,b,x);
mess_direct_res2(MESS_OP_NONE,A,x,b,&res,&rel_res);
printf("MESS_DIRECT_SPARSE_LU\n");
printf("||A*x-b||_2 \t= %10.5e\n",res);
printf("||A*x-b||_2/||b||_2 \t= %10.5e\n",rel_res);
printf("\n");
mess_direct_clear(&solver);


mess_direct_lu_select(MESS_DIRECT_UMFPACK_LU);
mess_direct_init(&solver);
mess_direct_lu(A,solver);
mess_direct_solve(MESS_OP_NONE,solver,b,x);
mess_direct_res2(MESS_OP_NONE,A,x,b,&res,&rel_res);
printf("MESS_DIRECT_UMFPACK_LU\n");
printf("||A*x-b||_2 \t= %10.5e\n",res);
printf("||A*x-b||_2/||b||_2 \t= %10.5e\n",rel_res);
printf("\n");
mess_direct_clear(&solver);


mess_direct_lu_select(MESS_DIRECT_CSPARSE_LU);
mess_direct_init(&solver);
mess_direct_lu(A,solver);
mess_direct_solve(MESS_OP_NONE,solver,b,x);
mess_direct_res2(MESS_OP_NONE,A,x,b,&res,&rel_res);
printf("MESS_DIRECT_CSPARSE_LU\n");
printf("||A*x-b||_2 \t= %10.5e\n",res);
printf("||A*x-b||_2/||b||_2 \t= %10.5e\n",rel_res);
printf("\n");
mess_direct_clear(&solver);
```

4. Symmetric Positive Definite System

Remember that A is diagonal dominant and all diagonal entries are real and positive. We compute

$$A \leftarrow A + A^T$$

and obtain a symmetric positive definite matrix. Generating a cholesky decomposition based direct solver is similiar to the previous part.

```
mess_matrix_ctranspose(A,rand);
mess_matrix_add(1.0,rand,1.0,A);

mess_direct_chol_select(MESS_DIRECT_CSPARSE_CHOLESKY
  );
mess_direct_init(&solver);
mess_direct_chol(A,solver);
mess_direct_solve(MESS_OP_NONE,solver,b,x);
mess_direct_res2(MESS_OP_NONE,A,x,b,&res,&rel_res);
printf("MESS_DIRECT_CSPARSE_CHOLESKY\n");
printf("||A*x-b||_2 \t= %10.5e\n",res);
printf("||A*x-b||_2/||b||_2 \t= %10.5e\n",rel_res);
printf("\n");
mess_direct_clear(&solver);
```

5. Dense Systems

In the case of a dense matrix, solving linear system works almost the same in **M.E.S.S.**.

```
mess_int_t size2 = 50;
mess_matrix_rand(A,size2,size2,MESS_DENSE,
  MESS_REAL,0.0);

mess_direct_lu_select(MESS_DIRECT_LAPACK_LU);
mess_direct_init(&solver);
mess_direct_lu(A,solver);

mess_vector_resize(b,size2);
mess_vector_resize(x,size2);
mess_direct_solve(MESS_OP_NONE,solver,b,x);
mess_direct_res2(MESS_OP_NONE,A,x,b,&res,&rel_res);
printf("MESS_DIRECT_LAPACK_LU\n");
printf("||A*x-b||_2 \t= %10.5e\n",res);
printf("||A*x-b||_2/||b||_2 \t= %10.5e\n",rel_res);
printf("\n");
mess_direct_clear(&solver);
```

If you want to perform a QR decomposition feel free to take a look at mess_direct_create_lapack_qr.

6. Clear Memory

```
mess_matrix_clear(&A);
mess_matrix_clear(&rand);
mess_vector_clear(&b);
mess_vector_clear(&x);
mess_vector_clear(&y);
```

See also

Tutorials

10.705 tutorial_eig.c File Reference

10.705.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Tutorial Eigenvalue Computation and Singular Value Decomposition

In this tutorial we show how to solve the eigenvalue problem

$$Ax = \lambda x,$$

where A is a matrix and λ is an eigenvalue with $x \neq 0$ as its corresponding eigenvector. Furthermore we show how to compute the singular value decomposition $A = U\Sigma V^H$. We also plot the eigenvalues as well as the singular values.

1. Include Header Files

We include standard C header files and in addtion the mess.h header file to make the **M.E.S.S.** library available.

```
#include <stdio.h>
#include <stdlib.h>
#include "mess/mess.h"
```

2. Check Input arguments, Init Matrix and Read Data

We check the number of input arguments by using the argc variable. We further assume that both input argument are paths to the MatrixMarket file that is needed for reading a matrix.

```
    if(argc!=2){
        printf("usage: %s A.mtx",argv[0]);
        return 1;
    }

    MESS_INIT_MATRICES(&A);
    mess_matrix_read_formated(argv[1],A,MESS_DENSE);
    if(A->rows!=A->cols){
        mess_matrix_printinfo(A);
        MESS_CLEAR_MATRICES(&A);
        printf("First matrix has wrong size.\n");
    }
```

3. Compute the Eigenvalues and Singular Values

The function mess_eigen_eig is able to compute the eigenvalues and mess_eigen_svd computes the singular value decomposition of A. We do not want to compute the eigenvectors and singularvectors, therefore we supress the computation by inserting a `NULL` pointer to the corresponding argument.

```
mess_vector_init(&evals);
mess_vector_alloc(evals,A->rows,MESS_COMPLEX);
mess_eigen_eig(A,evals,NULL);
mess_vector_init(&svals);
mess_vector_alloc(svals,A->rows,MESS_REAL);
mess_eigen_svd(A,svals,NULL,NULL);
```

4. Plotting the Eigenvalues and Singular Values

Now we want to plot the eigenvalues and the singular values on the complex plane. Therefore we use a mess_plotExport instance. The function mess_plotExport_init set the name of the plot the scaling of the axes, the labels for each axes and the number of datasets. We label every dataset with help of mess_plotExport_setLabel. Since we want to plot discrete points we set the type to `"points"`.

5. Add data to the plot

We add the singular as well as the eigenvalues to the plot. In general the eigenvalues are complex. Therefore we have to check the type of the mess_vector which contains the eigenvalues. Adding singular values works in a similar fashion and it is easier because singular values are real.

6. Create a gnuplot script

In order to make the results viewable we export the plot to `gnuplot` with mess_plotExport_createGnuScript. After execution of the programm a file `Eigenvalues_and_Singular_Values_script.gnu` is created. We can check our results by using `gnuplot`.

```
mess_plotExport plot;
mess_plotExport_init(&plot, "Eigenvalues_and_Singular_Values",
  MESS_PLOT_LIN, MESS_PLOT_LIN, "Real", "Imag", 2);
mess_plotExport_setLegendPos(plot, "belowCenter");
mess_plotExport_setLabel(plot, 0, "Eigenvalues");
mess_plotExport_setLabel(plot, 1, "Singular Values");
mess_plotExport_setType(plot, 0, "points");
mess_plotExport_setType(plot, 1, "points");
mess_plotExport_setColor(plot, 0, "red");
mess_plotExport_setColor(plot, 1, "blue");

//add eigenvalues
mess_int_t i;
if(MESS_IS_REAL(evals)){
    for(i=0;i<evals->dim;++i){ mess_plotExport_addData(plot,0,evals->values[i],0
  .0); }
}else{
    for(i=0;i<evals->dim;++i){ mess_plotExport_addData(plot,0,creal(evals->
  values_cpx[i]),cimag(evals->values_cpx[i])); }
}

//add singular values, singular values are in every case real
for(i=0;i<svals->dim;++i){ mess_plotExport_addData(plot,1,svals->values[i],0.0);
   }

// save plot
mess_plotExport_createGnuScript(plot);
```

7. Clear everything

We can clear the plot by mess_plotExport_clear. Another usefull way to clear mess_matrix and mess_vector instances is to use the macros MESS_CLEAR_MATRICES and MESS_CLEAR_VECTORS. The advantage is that these macros are able to handle several instances in one call. However using mess_matrix_clear and mess_vector_clear is also possible.

```
mess_plotExport_clear(&plot);
MESS_CLEAR_MATRICES(&A);
MESS_CLEAR_VECTORS(&evals,&svals);
```

See also

Tutorials

10.706 tutorial_eigs.c File Reference

10.706.1 Detailed Description

Tutorial: Eigenvalue Computation for Sparse Matrices

This function demonstrates how to approximate some eigenvalues of the eigenvalue problem

$$Ax = \lambda x,$$

where A is a matrix, λ an eigenvalue and x its corresponding eigenvector using the Arnoldi process.

See also

mess_eigen_arpack_template
mess_eigen_eigs

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "mess/mess.h"

#ifdef MESS_HAVE_ARPACK

int mvp(void * data, mess_vector x, mess_vector y) {
    mess_matrix A = (mess_matrix) data;
    mess_matrix_mvp(MESS_OP_NONE, A, x, y);
    return 0;
}
#endif

int main (int argc, char *argv[])
{
    mess_matrix matrix;
    mess_vector ev;

    /*-----------------------------------------------------------------------
     *  check input arguments
     *-----------------------------------------------------------------------*/
    if ( argc != 2 ) {
        printf ("Usage: %s file.mtx\n", argv[0]) ;
        exit(1);
    }

    /*-----------------------------------------------------------------------
     *  init and read matrics
     *-----------------------------------------------------------------------*/
    mess_matrix_init(&matrix);
    mess_matrix_read_formated(argv[1], matrix, MESS_CSR);
    if(matrix->rows!=matrix->cols){
        MESS_CLEAR_MATRICES(&matrix);
```

```
        printf("Failed:Matrix has to be square.\n");
        return 1;
    }

    /*-------------------------------------------------------------------------
     *  compute and print eigenvalues
     *-------------------------------------------------------------------------*/
    MESS_INIT_VECTORS(&ev);
    mess_vector_alloc(ev, matrix->rows, MESS_REAL);

    mess_eigen_eigs(matrix,30, 0, NULL, ev);
    printf("Eigenvalues:\n");
    mess_vector_print(ev);


    /*-------------------------------------------------------------------------
     *  compute eigenvalues with arpack if available
     *-------------------------------------------------------------------------*/
#ifdef MESS_HAVE_ARPACK
    mess_matrix V;
    mess_vector EV,R;
    mess_int_t i;
    mess_int_t err = 0;
    double norm, res;
    mess_eigen_arpack_options_t opt =
      MESS_EIGEN_ARPACK_DEFAULT;
    mess_mvpcall mvpcall;
    mess_mvpcall_matrix(&mvpcall, MESS_OP_NONE, matrix);
    opt.which = MESS_EIGEN_ARPACK_LARGE_MAGNITUDE;

    mess_matrix_init(&V);
    mess_vector_init(&EV);
    mess_vector_init(&R);
    mess_vector_alloc(EV,matrix->rows,MESS_REAL);
    mess_vector_alloc(R,matrix->rows,MESS_REAL);

    printf("ARPACK:\n");
    mess_eigen_arpack_template( mvpcall , 5, opt, ev,V);
    mess_mvpcall_clear(&mvpcall);
    mess_vector_print(ev);
    for (i = 0; i < ev->dim; i++) {
        mess_matrix_getcol(V,i,EV);
        mess_vector_norm2(EV,&norm);
        mess_matrix_mvp(MESS_OP_NONE, matrix, EV, R);
        if ( MESS_IS_REAL(ev)) {
            mess_vector_axpyc(-ev->values[i],EV,R);
        } else {
            mess_vector_axpyc(-ev->values_cpx[i],EV,R);
        }
        mess_vector_norm2(R,&res);
        printf("%ld \t %lg\n",(long) i,res/norm);
        if ( res / norm > sqrt(mess_eps()*matrix->rows)) err ++;
    }
    mess_matrix_clear(&V);
    mess_vector_clear(&EV);
    mess_vector_clear(&R);
    if ( err > 0 ) {
        printf("Eigenvalue Computations may be wrong\n");
        return 1;
    } else {
        printf("Eigenvalue Computations are correct.\n");
    }
#endif

    /*-------------------------------------------------------------------------
     *  clear data
     *-------------------------------------------------------------------------*/
    mess_matrix_clear(&matrix);
    mess_vector_clear(&ev);

    return 0;
}
```

10.707 tutorial_gen_fdm_cd.c File Reference

10.707.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

See also

mess_matgen_fdmmatrix

Tutorial: Matrix Generators

This function demonstrates how a convection-diffusion operator can be discretized on the unit square.

```
#include <stdio.h>
#include <stdlib.h>
#include "mess/mess.h"

double fxfun(double x, double y ) {
    return 10.0*x;
}

double fyfun(double x, double y) {
    return 100.0 * y;
}

int main ( int argc, char ** argv) {
    mess_matrix A;
    mess_int_t n;
    double ts, te;

    if (argc !=3 ) {
        printf("FDM Generator: LAPLACE on Unit Square\n");
        printf("Usage: %s n0 output.mtx\n", argv[0]);
        return -1;
    }
    n = atoi ( argv[1] );
    mess_matrix_init(&A);
    printf("Generate Convection - Diffusion on (0,1) x (0,1)...");
    ts = mess_wtime();
    mess_matgen_fdmmatrix(A, n, fxfun, fyfun, NULL);
    te = mess_wtime();
    printf("%lgs\n", te-ts);
    printf("Write it to %s ...", argv[2]);
    mess_matrix_write(argv[2], A);
    printf("done.\n");
    mess_matrix_clear(&A);
    return 0;
}
```

10.708 tutorial_gen_laplace.c File Reference

10.708.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

See also

mess_matgen_fdmmatrix

Tutorial: Matrix Generators

This function demonstrates how a convection-diffusion operator can be discretized on the unit square.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "mess/mess.h"

int main ( int argc, char ** argv) {
    mess_matrix A;
    mess_int_t n;
    double ts, te;

    if (argc !=3 ) {
        printf("FDM Generator: LAPLACE on Unit Square\n");
        printf("Usage: %s n0 output.mtx\n", argv[0]);
        return -1;
    }
    n = atoi ( argv[1] );
    mess_matrix_init(&A);
    printf("Generate LAPLACE on (0,1) x (0,1)...");
    ts = mess_wtime();
    mess_matgen_fdmmatrix(A, n, NULL, NULL, NULL);
    te = mess_wtime();
    printf("%lgs\n", te-ts);
    printf("Write it to %s ...", argv[2]);
    mess_matrix_write(argv[2], A);
    printf("done.\n");
    mess_matrix_clear(&A);
    return 0;
}
```

10.709 tutorial_gschur.c File Reference

10.709.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

See also

mess_eigen_gschur
mess_eigen_gschur_complex

Tutorial: Computing generalized Schur Decomposition

This function demonstrates computing the generalized Schur decomposition of two matrices A and B

$$\begin{aligned} S &= U^T A V, \\ T &= U^T B V, \end{aligned}$$

where S is an upper quasi-triangular matrix and T is an upper triangular matrix and the eigenvalues of the generalized eigenvalue problem:

$$Ax = \lambda B x.$$

10.710 tutorial_h2norm.c File Reference

10.710.1 Detailed Description

Tutorial: Compute $\mathcal{H}_2$ error.

This function demonstrates computing the $\mathcal{H}_2$-norm of a linear time invariant system

$$\begin{aligned} \dot{x} &= Ax &+Bu \\ y &= Cx \end{aligned} .$$

See also

mess_h2_norm

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <math.h>
#include <unistd.h>
#include <string.h>
#include "mess/config.h"
#include "mess/mess.h"
#include "mess/error_macro.h"
#include <complex.h>


int main ( int argc, char **argv){
    mess_matrix A,B,C;
    mess_dynsys lti;
    double norm = 0;
    int ret = 0;
    double ts, te;
    mess_version();
    mess_set_errorlevel(2);

    printf("mess - h2norm\n");
    printf("==============\n");


    /*-------------------------------------------------------------------------
     *  check input arguments
     *-------------------------------------------------------------------------*/
    if ( argc!=4) {
        printf("usage: %s A.mtx B.mtx C.mtx \n", argv[0]);
        return 0;
    }

    /*-------------------------------------------------------------------------
     *  init matrices
     *-------------------------------------------------------------------------*/
    MESS_INIT_MATRICES(&A,&B,&C);

    mess_matrix_read_formated(argv[1], A, MESS_CSR);
    mess_matrix_read_formated(argv[2], B, MESS_DENSE);
    mess_matrix_read_formated(argv[3], C, MESS_DENSE);


    printf("Dimension: " MESS_PRINTF_INT "\n", A->rows);
    printf("Inputs:    " MESS_PRINTF_INT "\n", B->cols);
    printf("Outputs:   " MESS_PRINTF_INT "\n", C->rows);
    mess_dynsys_init(&lti);
    mess_dynsys_lti_copy(lti, A, B, C);

    ts = mess_wtime();
    ret = mess_h2_norm(lti, &norm);
    te = mess_wtime();
    if ( ret !=0) {
        printf("error computing h2 norm: %s\n", mess_get_error(ret));
        return 1;
    } else {
        printf("h2 norm: %lg\n", norm);
    }
    printf("time: %lg\n", te-ts);
```

```
    /*-----------------------------------------------------------------------
     *  clear
     *-----------------------------------------------------------------------*/
    MESS_CLEAR_MATRICES(&A,&B,&C);
    mess_dynsys_clear(&lti);

    return 0;
}
```

10.711 tutorial_irka_biorth.c File Reference

10.711.1 Detailed Description

See also

mess_h2_irka_biorth

Tutorial: Irka Biorth algorithm

This function demonstrates computing a $\mathcal{H}_2$ reduced (generalized) linear time invariant ((G)LTI) system using the IRKA algorithm.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <math.h>
#include <unistd.h>
#include <string.h>
#include "mess/mess.h"
#include <complex.h>


void irka_stepdebug(void *data, mess_int_t it, double ds, double h2err){
    mess_plot p = (mess_plot) data;
    mess_plot_addData(p, 0, it, ds);
    if ( !isinf(h2err) ) mess_plot_addData(p, 1, it, h2err);
    mess_plot_update(p);
}

//#define PLOT
#undef PLOT

int main ( int argc, char **argv){
    mess_int_t r0, kp,km, ret = 0,useE=0;
    mess_vector sigma;
    mess_matrix A,B,C,E=NULL,V,W;
    mess_direct Asolver;
    mess_dynsys orig, reduced;
    mess_h2_options opt;
    mess_h2_status status;

#ifdef PLOT
    mess_plot plot;
#endif

    mess_version();

    printf("mess - IRKA BiOrth\n");
    printf("===================\n");


    /*-----------------------------------------------------------------------
     *  check input arguments
     *-----------------------------------------------------------------------*/
    if ( argc!=8 && argc!=9) {
        printf("usage: %s A.mtx B.mtx C.mtx (E.mtx) r0 maxit tol h2\n", argv[0]);      return 1;
    }
    if ( argc==9){
        printf(" -> for generalized LTI systems.\n");
        mess_matrix_init(&E);
    }
    mess_init();
    MESS_INIT_MATRICES(&A,&B,&C,&V,&W);
```

```
    /*-----------------------------------------------------------------------------
     *  init dynamical system strucutre and H2 options
     *-----------------------------------------------------------------------------*/
    mess_dynsys_init(&orig);
    mess_h2_options_init(&opt);
    mess_h2_status_init(&status);


    /*-----------------------------------------------------------------------------
     *  read matrices
     *-----------------------------------------------------------------------------*/
    mess_matrix_read_formated(argv[1], A, MESS_CSR);
    mess_matrix_read_formated(argv[2], B, MESS_DENSE);
    mess_matrix_read_formated(argv[3], C, MESS_DENSE);
    if (E){
        mess_matrix_read_formated(argv[4],E,MESS_CSR);
        useE=1;
    }
    if (B->cols!=1){
        //only siso systems
        mess_matrix tmp; MESS_INIT_MATRICES(&tmp);
      mess_matrix_colsub(B,0,0,tmp);
        mess_matrix_copy(tmp,B); mess_matrix_clear(&tmp);
    }

    if (C->rows!=1){
        mess_matrix tmp; MESS_INIT_MATRICES(&tmp);
      mess_matrix_rowsub(C,0,0,tmp);
        mess_matrix_copy(tmp,C); mess_matrix_clear(&tmp);
    }

#ifdef PLOT
    /*-----------------------------------------------------------------------------
     *  plot
     *-----------------------------------------------------------------------------*/
    mess_plot_create(&plot,800,600,"IRKA BiOrth", MESS_PLOT_LIN,
      MESS_PLOT_LOG, 2);
    mess_plot_setLabel(plot,0, "rel |sigma-sigmaold|");
    mess_plot_setLabel(plot,1, "h2 error");

    mess_plot_setColor(plot,0, "red");
    mess_plot_setColor(plot,1, "blue");

    mess_plot_update(plot);

#endif

    /*-----------------------------------------------------------------------------
     *  H2 options
     *-----------------------------------------------------------------------------*/
#ifdef PLOT
    opt->stepdebug = irka_stepdebug;
    opt->stepdebug_data = (void*) plot;
#endif
    r0=atoi(argv[4+useE]);
    opt->maxit = atoi(argv[5+useE]);
    opt->tol=atof(argv[6+useE]);
    opt->calc_h2err = atoi(argv[7+useE]);
    opt->calc_finalh2  = 0;
    opt->output = 1;

    kp=3*r0;
    km=2*r0;


    printf("Dimension: " MESS_PRINTF_INT "\n", A->rows);
    printf("wanted number of parameters: " MESS_PRINTF_INT "\n", r0);
    printf("Arnoldi steps w.r.t. A: " MESS_PRINTF_INT "\n", kp);
    printf("Arnoldi steps w.r.t. inv(A): " MESS_PRINTF_INT "\n", km);
    printf("\nIRKA-options:\n");
    mess_h2_options_print(opt);
    printf("\n");


    /*-----------------------------------------------------------------------------
     *  compute irka parameters
     *-----------------------------------------------------------------------------*/
    mess_direct_init(&Asolver);
    mess_direct_lu(A,Asolver);

    mess_vector_init(&sigma);
    mess_vector_alloc(sigma, r0, MESS_COMPLEX);
    ret = mess_h2_irka_init(A,Asolver, E, &r0, kp,km, sigma);
    mess_direct_clear(&Asolver);

    if ( ret !=0) {
```

```
        printf("error computing IRKA parameters: %s\n", mess_get_error(ret));
    } else {
        printf("number of parameters: " MESS_PRINTF_INT "\n", r0);
        mess_vector_print(sigma);
    }

    if  (useE)  mess_dynsys_glti(orig, A,B,C,E);
    else mess_dynsys_lti(orig, A,B,C);

    /*------------------------------------------------------------------------
     *  reduce system
     *------------------------------------------------------------------------*/
    mess_dynsys_init(&reduced);
    mess_h2_irka_biorth(orig,sigma,opt,reduced,V,W, status);
    mess_dynsys_clear(&reduced);

    /*------------------------------------------------------------------------
     *  write status
     *------------------------------------------------------------------------*/
    printf("status: \n");
    mess_h2_status_print(status);
    mess_h2_status_write_mfile(status,"irka_output.m");


    /*------------------------------------------------------------------------
     *  evaluate transfer function
     *------------------------------------------------------------------------*/
    {
        mess_vector omega, G;
        MESS_INIT_VECTORS(&omega,&G);
        mess_vector_alloc(omega,200,MESS_REAL);
        mess_vector_alloc(G,200, MESS_REAL);
        mess_dynsys_evaltransfer(orig,-1,6,200,omega,G,NULL);

        mess_vector_printshort(omega);
        mess_vector_printshort(G);

        mess_vector_clear(&G);
        mess_vector_clear(&omega);
    }

#ifdef PLOT
    /*------------------------------------------------------------------------
     *  save plot
     *------------------------------------------------------------------------*/
    sleep(5);
    mess_plot_save(plot,"plot.xpm");
#endif

    /*------------------------------------------------------------------------
     *  clear
     *------------------------------------------------------------------------*/
#ifdef PLOT
    mess_plot_close(&plot);
#endif
    MESS_CLEAR_MATRICES(&V,&W);
    //MESS_CLEAR_MATRICES(&A,&B,&C,&E); cleared by dynsys_clear
    mess_dynsys_clear(&orig);
    mess_h2_options_clear(&opt);
    mess_h2_status_clear(&status);
    mess_vector_clear(&sigma);
    mess_exit();

    return 0;
}
```

10.712 tutorial_it_unsym.c File Reference

10.712.1 Detailed Description

See also

mess_solver_bicgstab
mess_solver_gmres
mess_solver_gmres_restart
mess_solver_sor
mess_precond_ilu0
mess_precond_iluk
mess_precond_diag

Tutorial: Iterative Solvers

This function demonstrates how to use different iterative solvers to solve

$$Ax = b$$

for an unsymmetric matrix A.
Demonstrated iterative solvers are

- BICGSTAB,
- generalized minimal residual (GMRES),
- restarted GMRES,
- successive overrelaxation (SOR).

Restarted GMRES and BICGSTAB are demonstrated using different preconditioners:

- diagonal preconditioner,
- $ILU(0)$ preconditioner (only used for BICGSTAB),
- $IlU(k)$ preconditioner with a level of fill $= 3$.

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "mess/mess.h"


int main ( int argc, char **argv )
{
    mess_matrix matrix = NULL;
    mess_matrix input = NULL;
    mess_vector x = NULL;
    mess_vector b = NULL;
    mess_solver_options opt;
    mess_solver_status  stat;
    mess_mvpcall mvpcall;
    mess_precond pre_diag = NULL;
    mess_precond pre_ilu0 = NULL;
    mess_precond pre_sor  = NULL;
    mess_precond pre_iluk3 = NULL;
    int use_ilu0 = 0;
    int use_diag = 1;
    int use_ilu3 = 1;
    // int ret = 0;

    printf("mess - demo - iterative solvers for general systems\n");
    printf("=====================================================\n");
    mess_version();


    mess_set_errorlevel(3);

    if ( argc != 5 ) {
        printf (" Usage: %s file.mtx maxit restarts tol\n", argv[0]) ;
        return 1;
    }

    /*-----------------------------------------------------------------------------
     *  read matrix and prepare
     *-----------------------------------------------------------------------------*/
    mess_matrix_init(&input);
    mess_matrix_init(&matrix);
    if (mess_matrix_read(argv[1], input)) {
        printf("error reading matrix\n");
        return 1;
    }
    mess_matrix_convert(input, matrix, MESS_CSR);
    mess_matrix_clear(&input);
```

```
if (matrix->cols != matrix->rows){
    printf("Only with square matrices possible.\n");
}

MESS_INIT_VECTORS(&x,&b);
mess_vector_alloc(x, matrix->rows, MESS_REAL);
mess_vector_alloc(b, matrix->rows, MESS_REAL);
mess_matrix_rowsums(matrix,b);


/*-------------------------------------------------------------------------
 *  Setup the preconditioners
 *-------------------------------------------------------------------------*/
mess_precond_init(&pre_diag);
mess_precond_init(&pre_ilu0);
mess_precond_init(&pre_sor);
mess_precond_init(&pre_iluk3);

if ( mess_precond_diag(pre_diag, matrix) !=0 ){
    printf("-> diagonal preconditioner can not be created. Not used.\n");
    use_diag = 0;
}
if ( mess_precond_ilu0(pre_ilu0, matrix) !=0) {
    printf("-> ilu 0 preconditioner can not be created. Not used.\n");
    use_ilu0 = 0;
}
if ( mess_precond_iluk(pre_iluk3, matrix, 3) !=0) {
    printf("-> ilul preconditioner can not be created. Not used.\n");
    use_ilu3 = 0;
}

/*-------------------------------------------------------------------------
 *  fill options
 *-------------------------------------------------------------------------*/
mess_solver_options_init(&opt);
opt->tol = atof(argv[4]);
opt->maxit = atoi(argv[2]);
opt->restarts = atoi(argv[3]);

printf("\noptions:\n");
mess_solver_options_print(opt);

mess_solver_status_init(&stat);

mess_mvpcall_matrix(&mvpcall, MESS_OP_NONE, matrix);

/*-------------------------------------------------------------------------
 *  GMRES
 *-------------------------------------------------------------------------*/
printf("\n\n-> GMRES\n");
mess_vector_zeros(x);
mess_solver_gmres(mvpcall , NULL, b, x, opt, stat);
printf("Status:\n");
mess_solver_status_print(stat);
printf("solution:\n");
mess_vector_printshort(x);
mess_solver_status_clean(stat);

/*-------------------------------------------------------------------------
 *  restarted GMRES
 *-------------------------------------------------------------------------*/
printf("\n\n-> restarted GMRES\n");
mess_vector_zeros(x);
mess_solver_gmres_restart(mvpcall, NULL, b, x, opt, stat);
printf("Status:\n");
mess_solver_status_print(stat);
printf("solution:\n");
mess_vector_printshort(x);
if(stat->relres>opt->tol){printf("FAILED\n"); return 1;}
mess_solver_status_clean(stat);

/*-------------------------------------------------------------------------
 *  restarted GMRES with Diag
 *-------------------------------------------------------------------------*/
if ( use_diag) {
    printf("\n\n-> restarted GMRES with diagonal preconditioner\n");
    mess_vector_zeros(x);
    mess_solver_gmres_restart(mvpcall , pre_diag, b, x, opt, stat);
    printf("Status:\n");
    mess_solver_status_print(stat);
    printf("solution:\n");
    mess_vector_printshort(x);
    if(stat->relres>opt->tol){printf("FAILED\n"); return 1;}
    mess_solver_status_clean(stat);
}

/*-------------------------------------------------------------------------
```

```
 *  restarted GMRES with ilu(3)
 *-----------------------------------------------------------------------------*/
if ( use_ilu3) {
    printf("\n\n-> restarted GMRES with ilu(3) preconditioner\n");
    mess_vector_zeros(x);
    mess_solver_gmres_restart(mvpcall, pre_iluk3, b, x, opt, stat);
    printf("Status:\n");
    mess_solver_status_print(stat);
    printf("solution:\n");
    mess_vector_printshort(x);
    if(stat->relres>opt->tol){printf("FAILED\n"); return 1;}
    mess_solver_status_clean(stat);
}

/*-----------------------------------------------------------------------------
 * BICGSTAB
 *-----------------------------------------------------------------------------*/
opt->maxit = opt->maxit * opt->restarts;
printf("\n\n-> BiCGStab\n");
mess_vector_zeros(x);
mess_solver_bicgstab(mvpcall, NULL, b, x, opt, stat);
printf("Status:\n");
mess_solver_status_print(stat);
printf("solution:\n");
mess_vector_printshort(x);
if(stat->relres>opt->tol){printf("FAILED\n"); return 1;}
mess_solver_status_clean(stat);

/*-----------------------------------------------------------------------------
 * BiCGStab with Diag
 *-----------------------------------------------------------------------------*/
if ( use_diag) {
    printf("\n\n-> BiCGStab with diagonal preconditioner\n");
    mess_vector_zeros(x);
    mess_solver_bicgstab(mvpcall, pre_diag, b, x, opt, stat);
    printf("Status:\n");
    mess_solver_status_print(stat);
    printf("solution:\n");
    mess_vector_printshort(x);
    if(stat->relres>opt->tol){printf("FAILED\n"); return 1;}
    mess_solver_status_clean(stat);
}

/*-----------------------------------------------------------------------------
 *  BiCGStab with ilu(0)
 *-----------------------------------------------------------------------------*/
if ( use_ilu0) {
    printf("\n\n-> BiCGStab with ilu(0) preconditioner\n");
    mess_vector_zeros(x);
    mess_solver_bicgstab(mvpcall, pre_ilu0, b, x, opt, stat);
    printf("Status:\n");
    mess_solver_status_print(stat);
    printf("solution:\n");
    mess_vector_printshort(x);
    if(stat->relres>opt->tol){printf("FAILED\n"); return 1;}
    mess_solver_status_clean(stat);
}
/*-----------------------------------------------------------------------------
 *  BiCGStab with iluk(3)
 *-----------------------------------------------------------------------------*/
if ( use_ilu3) {
    printf("\n\n-> BiCGStab with ilu(3) preconditioner\n");
    mess_vector_zeros(x);
    mess_solver_bicgstab(mvpcall, pre_iluk3, b, x, opt, stat);
    printf("Status:\n");
    mess_solver_status_print(stat);
    printf("solution:\n");
    mess_vector_printshort(x);
    if(stat->relres>opt->tol){printf("FAILED\n"); return 1;}
    mess_solver_status_clean(stat);
}

/*-----------------------------------------------------------------------------
 *  SOR omega = 1
 *-----------------------------------------------------------------------------*/
printf("\n\n-> SOR omega = 1\n");
mess_vector_zeros(x);
opt->omega = 1;
mess_solver_sor(matrix, pre_ilu0, b, x, opt, stat);
printf("Status:\n");
mess_solver_status_print(stat);
printf("solution:\n");
mess_vector_printshort(x);
if(stat->relres>opt->tol){printf("FAILED\n"); return 1;}
mess_solver_status_clean(stat);

/*-----------------------------------------------------------------------------
```

```
     *  SOR omega = 1.5
     *-----------------------------------------------------------------------------*/
    printf("\n\n-> SOR omega = 1.1\n");
    mess_vector_zeros(x);
    opt->omega = 1.1;
    mess_solver_sor(matrix, pre_ilu0, b, x, opt, stat);
    printf("Status:\n");
    mess_solver_status_print(stat);
    printf("solution:\n");
    mess_vector_printshort(x);
    if(stat->relres>opt->tol){printf("FAILED\n"); return 1;}
    mess_solver_status_clean(stat);

    /*-----------------------------------------------------------------------------
     *  SOR omega = 0.5
     *-----------------------------------------------------------------------------*/
    printf("\n\n-> SOR omega = 0.5\n");
    mess_vector_zeros(x);
    opt->omega = 0.5;
    mess_solver_sor(matrix, pre_ilu0, b, x, opt, stat);
    printf("Status:\n");
    mess_solver_status_print(stat);
    printf("solution:\n");
    mess_vector_printshort(x);
    if(stat->relres>opt->tol){printf("FAILED\n"); return 1;}
    mess_solver_status_clean(stat);



    /*-----------------------------------------------------------------------------
     *  cleanup
     *-----------------------------------------------------------------------------*/

    mess_mvpcall_clear(&mvpcall);
    mess_solver_options_clear(&opt);
    mess_solver_status_clear(&stat);
    mess_matrix_clear(&matrix);
    mess_precond_clear(&pre_ilu0);
    mess_precond_clear(&pre_diag);
    mess_precond_clear(&pre_sor);
    mess_precond_clear(&pre_iluk3);
    mess_vector_clear(&b);
    mess_vector_clear(&x);
    return 0;
}
```

10.713 tutorial_lradi.c File Reference

10.713.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Tutorial Solving a (generalized) Lyapunov Equation

In this tutorial we show solve the (generalized) Lyapunov Equation, i.e. $FX + XF^T = -GG^T$ or $FXE^T + EXF^T = -GG^T$.
We assume that $F, E \in \mathbb{R}^{n\times n}$ is large and sparse and the spectrum $\Lambda(F) \subseteq \mathbb{C}^-$ or $\Lambda(F, E) \subseteq \mathbb{C}^-$ is contained in the left open complex half-plane.
The right-hand side is defined by $G \in \mathbb{R}^{p\times n}$ and $p \ll n$.
The solver mess_lradi returns the solution X as a low-rank factor approximation Z such that $X \approx ZZ^T$.
In the last step we compute the relative residual $\|FZZ^T + ZZ^TF^T + GG^T\|_2/\|GG^T\|_2$ or $\|FZZ^TE^T + EZZ^TF^T + GG^T\|_2/\|GG^T\|_2$ and clear the memory.

1. Include Header Files

We include standard C header files and in addtion the `mess.h` header file to make the **M.E.S.S.** library available.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <math.h>
#include "mess/mess.h"
```

2. Declare Matrices and Check the Number of Input Arguments.

We use the mess_matrix structure to declare our matrices F, G, E, Z. Furthermore we need a mess_equation, mess_options and mess_status structure.

```
mess_matrix F,G,E,Z;
mess_equation eqn;
mess_options opt;
mess_status stat;
```

By using the `argc` variable we can check the number of input arguments.

```
if ( argc !=3&& argc != 4 ) {
    printf("usage: %s F.mtx G.mtx [E.mtx] \n", argv[0]);
    return 1;
}
```

3. Init Matrices and Read Data

Before we are able to use the matrices we have to init the fields of the mess_matrix structure variables using mess_matrix_init and read the matrices from `MatrixMarket` file format.

```
mess_matrix_init(&F);
mess_matrix_init(&G);
mess_matrix_init(&Z);
if(argc==4)
    mess_matrix_init(&E);

mess_matrix_read(argv[1],F);
mess_matrix_read(argv[2],G);
printf("Matrix F:\n"); mess_matrix_printinfo(F);
printf("Matrix G:\n"); mess_matrix_printinfo(G);
if(argc==4){
    mess_matrix_read(argv[3],E);
    printf("Matrix E:\n");
    mess_matrix_printinfo(E);
}
```

4. Init Options, Status and Equations

We use mess_options_init to init the mess_options structure. The mess_options::adi_shifts_paratype describes the shift parameter strategy. mess_options::adi_output makes mess_lradi verbose and with mess_options::type we can switch between the transposed and non-transposed Lyapunov Equation.

Operation Type	Lyapunov Equation	generalized Lyapunov Equation
MESS_OP_NONE	$FX + XF^T = -GG^T$	$FXE^T + EXF^T = -GG^T$
MESS_OP_TRANSPOSE	$F^TX + XF = -G^TG$	$F^TXE + E^TXF = -G^TG$

```
mess_options_init(&opt);
opt->adi_shifts_paratype = MESS_LRCFADI_PARA_MINMAX;
opt->adi_output = 1;
opt->type = MESS_OP_NONE;
mess_options_print(opt);
```

Now we init the mess_status structure.

```
mess_status_init(&stat);
```

mess_equation_lyap builds an mess_equation structure for the (generalized) Lyapunov Equation.

```
mess_equation_init(&eqn);
if(argc==3){
    mess_equation_lyap(eqn,opt,F,NULL,G);
}else{
    mess_equation_lyap(eqn,opt,F,E,G);
}
```

5. Solve the (Generalized) Lyapunov Equation

Now we use mess_lrcfadi_parameter to compute a shift parameter set. mess_status structure `stat` is used to collect runtime and residual information. mess_lradi solves the Lyapunov Equation and after convergence the low-rank factor is stored in `Z`.

```
mess_parameter(eqn,opt,stat);
mess_status_print(stat);
mess_lradi(eqn,opt,stat,Z);
mess_status_print(stat);
```

6. Compute the Residual

We declare two `double` variables for storing the right-hand side norm and the absolute residual. mess_matrix_↩dynorm2 computes the norm of the right-hand side and mess_lrcfadi_residual computes the desired 2-norms.

```
double nrm=1, nrmRHS=1;
mess_matrix_dynorm2(G,&nrmRHS);
mess_lrcfadi_residual(eqn,opt,Z,&nrm);
printf("relative Residual= %10.15e\n",nrm/nrmRHS);
```

7. Clear Memory

Clearing the mess_status, mess_options, mess_equation is done by the corresponding functions.

```
    mess_status_clear(&stat);
    mess_options_clear(&opt);
    mess_equation_clear(&eqn);
    if(argc==3){
        mess_matrix_clear(&F);
        mess_matrix_clear(&G);
        mess_matrix_clear(&Z);
    }else{
        mess_matrix_clear(&F);
        mess_matrix_clear(&G);
        mess_matrix_clear(&E);
        mess_matrix_clear(&Z);
    }
    return nrm/nrmRHS > 1e-9;
```

See also

Tutorials

10.714 tutorial_lrnm-g.c File Reference

10.714.1 Detailed Description

Tutorial: Solve generalized algebraic Riccati Equatio, Plot and Stepdebug Function

This function demonstrates computing the solution factor Z of the solution $X \approx ZZ^T$ of a generalized algebraic Riccati Equation

$$A^T XE + E^T XA - C^T C + E^T XBB^T XE = 0$$

using the low-rank Cholesky factor alternate direction implicit Newton method (LRCF-ADI-NM).

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <math.h>
#include <unistd.h>
#include <string.h>
#include "mess/config.h"
#include "mess/mess.h"
#include "mess/error_macro.h"

//#define PLOT
#undef PLOT

#ifdef PLOT
void lrcf_stepfunction(void *data,mess_equation eqn, mess_options opt,
      mess_lrcfadi_step * step){
    mess_plot p = (mess_plot) data;
    if ( step->it == 0) {
        mess_plot_clearData(p,0);
        mess_plot_clearData(p,1);
        mess_plot_clearData(p,2);
        mess_plot_update(p);
    }
    mess_plot_addData(p, 0, step->it, step->res2);
    mess_plot_addData(p, 1, step->it, step->rel_change);
    mess_plot_addData(p, 2, step->it, (step->res2_change));
    mess_plot_update(p);
}
void lrnm_stepfunction(void *data, mess_equation eqn, mess_equation lyap,
```

```
      mess_options opt, mess_lrcfadi_step * step){
    mess_plot p = (mess_plot) data;
    if ( step->it == 0) {
        mess_plot_clearData(p,0);
        mess_plot_clearData(p,1);
        mess_plot_update(p);
    }
    mess_plot_addData(p, 0, step->it, step->res2);
    mess_plot_addData(p, 1, step->it, (step->res2_change));
    mess_plot_update(p);
}
#endif


int main ( int argc, char **argv){
    mess_matrix A,B,C,E,Z;
    mess_options opt;
    mess_equation eqn;
    mess_status stat;
#ifdef PLOT
    mess_plot plot;
    mess_plot plot_nm;
#endif

    mess_version();
    printf("mess - low-rank-cholesky-factor-nm-g  demo plotting\n");
    printf("=====================================================\n");


    /*----------------------------------------------------------------------------
     *  check input arguments
     *----------------------------------------------------------------------------*/
    if ( argc!=5) {
        printf("usage: %s A.mtx B.mtx C.mtx E.mtx\n", argv[0]);
        return 1;
    }

    mess_init();

    /*----------------------------------------------------------------------------
     *  init and read matrices
     *----------------------------------------------------------------------------*/
    MESS_INIT_MATRICES(&A,&B,&C,&E,&Z);

    mess_matrix_read_formated(argv[1],A, MESS_CSR);
    mess_matrix_read_formated(argv[2],B, MESS_DENSE);
    mess_matrix_read_formated(argv[3],C, MESS_DENSE);
    mess_matrix_read_formated(argv[4],E, MESS_CSR);

    //scale to force more lrnm steps
    mess_matrix_scale(1e+2,C);
    /*----------------------------------------------------------------------------
     *  Plot for LRCFADI
     *----------------------------------------------------------------------------*/
#ifdef PLOT
    mess_plot_create(&plot,800,600,"LRCF-ADI residual",
      MESS_PLOT_LIN, MESS_PLOT_LOG, 3);
    mess_plot_setLabel(plot,0, "res2-norm");
    mess_plot_setLabel(plot,1, "rel-change");
    mess_plot_setLabel(plot,2, "res2-change");
    mess_plot_setColor(plot,0, "red");
    mess_plot_setColor(plot,1, "blue");
    mess_plot_setColor(plot,2, "green");
    mess_plot_update(plot);

    /*----------------------------------------------------------------------------
     *  PLOT for LRCF-NM
     *----------------------------------------------------------------------------*/
    mess_plot_create(&plot_nm,800,600,"LRCF-NM residual",
      MESS_PLOT_LIN, MESS_PLOT_LOG, 3);
    mess_plot_setLabel(plot_nm,0, "res2-norm");
    mess_plot_setLabel(plot_nm,1, "rel-change");
    mess_plot_setLabel(plot_nm,2, "res2-change");
    mess_plot_setColor(plot_nm,0, "red");
    mess_plot_setColor(plot_nm,1, "blue");
    mess_plot_setColor(plot_nm,2, "green");
    mess_plot_update(plot_nm);
#endif

    /*----------------------------------------------------------------------------
     *  init status and options
     *----------------------------------------------------------------------------*/
    mess_status_init(&stat);
    mess_options_init(&opt);
    opt->nm_gpStep=0;
    mess_options_print(opt);
```

```
#ifdef PLOT
    opt->stepfunction           = lrcf_stepfunction;
    opt->stepfunction_aux       = plot;
    opt->nm_stepfunction        = lrnm_stepfunction;
    opt->nm_stepfunction_aux    = plot_nm;
#endif

    /*----------------------------------------------------------------------------
     *  create equation
     *----------------------------------------------------------------------------*/
    mess_equation_init(&eqn);
    mess_equation_riccati(eqn,opt,A,E,B,C);

    /*----------------------------------------------------------------------------
     *  solve riccati equation
     *----------------------------------------------------------------------------*/
    mess_lrcfadi_nm(eqn,opt,stat,Z);

    /*----------------------------------------------------------------------------
     *  print status information
     *----------------------------------------------------------------------------*/
    printf("\n state structure :\n");
    mess_status_print(stat);

    /*----------------------------------------------------------------------------
     *  write matrix
     *----------------------------------------------------------------------------*/
    printf("-> writing Z factor to %s\n","Z.mtx" );
    mess_matrix_write( "Z.mtx" , Z);

    /*----------------------------------------------------------------------------
     *  plot
     *----------------------------------------------------------------------------*/
#ifdef PLOT
    sleep(10);
    mess_plot_save(plot,"plot.xpm");
    mess_plot_close(&plot);
    mess_plot_close(&plot_nm);
#endif

    /*----------------------------------------------------------------------------
     *  clear matrices
     *----------------------------------------------------------------------------*/
    MESS_CLEAR_MATRICES(&A,&B,&C,&E,&Z);
    mess_equation_clear(&eqn);
    mess_status_clear(&stat);
    mess_options_clear(&opt);

    mess_exit();
    return 0;
}
```

10.715 tutorial_lrnm.c File Reference

10.715.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Tutorial Solving a (generalized) Riccati Equation

In this tutorial we solve the (generalized) Riccati Equation, i.e. $A^TX + XA - XBB^TX + C^TC = 0$ or $A^TXE + E^TXA - E^TXBB^TXE + C^TC = 0$.
We assume that $A, E \in \mathbb{R}^{n\times n}$ is large and sparse and the spectrum $\Lambda(A) \subseteq \mathbb{C}^-$ or $\Lambda(A,E) \subseteq \mathbb{C}^-$ is contained in the left open complex half-plane. Under stability and detectability assumption theory admits existence of a unique solution of the (generalized) algebraic Riccati Equation and ensures convergence of the algorithm.
The right-hand side is defined by $C \in \mathbb{R}^{p\times n}$ and $B \in \mathbb{R}^{n\times m}$. Furthermore the number of outputs and inputs should be small compared to the size of the system $p, m \ll n$.
The solver mess_lrnm returns the solution X as a low-rank factor approximation Z such that $X \approx ZZ^T$.
In the last step we compute the relative residual $\|A^TZZ^T + ZZ^TA - ZZ^TBB^TZZ^T + C^TC\|_2/\|C^TC\|_2$ or $\|\|A^TZZ^TE + E^TZZ^TA - E^TZZ^TBB^TZZ^TE + C^TC\|_2/\|C^TC\|_2$ and clear the memory.

1. Include Header Files

We include standard C header files and in addtion the mess.h header file to make the **M.E.S.S.** library available.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <math.h>
#include "mess/mess.h"
```

2. Declare Matrices and Check the Number of Input Arguments.

We use the mess_matrix structure to declare our matrices A, B, C, E, Z. Furthermore we need a mess_equation, mess_options and mess_status structure.

```
    mess_matrix A,B,C,E,Z;
    mess_equation eqn;
    mess_options opt;
    mess_status stat;
```

By using the argc variable we can check the number of input arguments.

```
    if ( argc !=4 && argc != 5 ) {
        printf("usage: %s A.mtx B.mtx C.mtx [E.mtx] \n", argv[0]);
        return 1;
    }
```

3. Init Matrices and Read Data

Before we are able to use the matrices we have to init the fields of the mess_matrix structure variables using mess_matrix_init and read the matrices from MatrixMarket file format.

```
    mess_matrix_init(&A);
    mess_matrix_init(&B);
    mess_matrix_init(&C);
    mess_matrix_init(&Z);
    if(argc==5)
        mess_matrix_init(&E);

    mess_matrix_read_formated(argv[1],A,MESS_CSR);
    mess_matrix_read_formated(argv[2],B,MESS_DENSE);
    mess_matrix_read_formated(argv[3],C,MESS_DENSE);
    if(argc==5){
        mess_matrix_read_formated(argv[4],E,MESS_CSR);
    }
```

4. Init Options, Status and Equations

We use mess_options_init to init the mess_options structure. The mess_options::adi_shifts_paratype describes the shift parameter strategy. mess_options::adi_output makes mess_lrnm verbose and with mess_options::type we can switch between the transposed and non-transposed Riccati Equation.

Operation Type	Riccati Equation	generalized Riccati Equation
MESS_OP_NONE	$AX + XA^T - XBB^TX + C^TC = 0$	$AXE^T + EXA^T - EXBB^TX^T + C^TC = 0$
MESS_OP_TRANSPOSE	$A^TX + XA - XC^TCX + BB^T = 0$	$A^TXE + E^TXA - E^TXC^TCXE + BB^T = 0$

```
mess_options_init(&opt);
opt->adi_shifts_paratype = MESS_LRCFADI_PARA_MINMAX;
opt->adi_output = 1;
opt->type = MESS_OP_TRANSPOSE;
mess_options_print(opt);
```

Now we init the mess_status structure.

```
mess_status_init(&stat);
```

mess_equation_riccati builds an mess_equation structure for the (generalized) Riccati Equation.

```
mess_equation_init(&eqn);
if(argc==4){
    mess_equation_riccati(eqn,opt,A,NULL,B,C);
}else{
    mess_equation_riccati(eqn,opt,A,E,B,C);
}
```

5. Solve the (Generalized) Riccati Equation

Now we use mess_lrcfadi_parameter to compute a shift parameter set. mess_status structure `stat` is used to collect runtime and residual information. mess_lrnm solves the Riccati Equation and after convergence the low-rank factor is stored in `Z`.

```
mess_parameter(eqn,opt,stat);
mess_status_print(stat);
mess_lrnm(eqn,opt,stat,Z);
mess_status_print(stat);
```

6. Compute the Residual

We declare two `double` variables for storing the right-hand side norm and the absolute residual. mess_matrix_↩dynorm2 computes the norm of the right-hand side and mess_lrcfadi_residual computes the desired 2-norms.

```
double nrm=1, nrmRHS=1;
mess_matrix_dynorm2(C,&nrmRHS);
mess_lrcfadi_residual(eqn,opt,Z,&nrm);
printf("relative Residual= %10.15e\n",nrm/nrmRHS);
```

7. Clear Memory

Clearing the mess_status, mess_options, mess_equation is done by the corresponding functions.

```
mess_status_clear(&stat);
mess_options_clear(&opt);
mess_equation_clear(&eqn);
if(argc==4){
    mess_matrix_clear(&A);
    mess_matrix_clear(&B);
    mess_matrix_clear(&C);
    mess_matrix_clear(&Z);
}else{
    mess_matrix_clear(&A);
    mess_matrix_clear(&B);
    mess_matrix_clear(&C);
    mess_matrix_clear(&E);
    mess_matrix_clear(&Z);
}
return nrm/nrmRHS > 1e-9;
```

See also

Tutorials

10.716 tutorial_lyap.c File Reference

10.716.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Tutorial Solving a (generalized) Lyapunov Equation

In this tutorial we solve the (generalized) Lyapunov Equation, i.e. $FX + XF^T = -GG^T$ or $FXE^T + EXF^T = -GG^T$.
We assume that $F, E \in \mathbb{R}^{n\times n}$ is large and sparse and the spectrum $\Lambda(F) \subseteq \mathbb{C}^-$ or $\Lambda(F, E) \subseteq \mathbb{C}^-$ is contained in the left open complex half-plane.
The right-hand side is defined by $G \in \mathbb{R}^{n\times p}$ and $p \ll n$.
The solver mess_lyap returns the solution X as a low-rank factor approximation Z such that $X \approx ZZ^T$.
In the last step we compute the relative residual $\|FZZ^T + ZZ^TF^T + GG^T\|_2/\|GG^T\|_2$ or $\|FZZ^TE^T + EZZ^TF^T + GG^T\|_2/\|GG^T\|_2$ and clear the memory.

1. Include Header Files

We include standard C header files and in addtion the mess.h header file to make the **M.E.S.S.** library available.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <math.h>
#include "mess/mess.h"
```

2. Declare Matrices and Check the Number of Input Arguments.

We use the mess_matrix structure to declare our matrices F, G, E, Z.

```
mess_matrix F,G,E,Z;
```

By using the `argc` variable we can check the number of input arguments.

```
if(argc!=3 && argc!=4){
    printf("usage: %s F.mtx G.mtx [E.mtx]\n",argv[0]);
    return 1;
}
```

3. Init Matrices and Read Data

Before we are able to use we have to init the fields of the mess_matrix structure variables using mess_matrix_init

```
mess_matrix_init(&F);
mess_matrix_init(&G);
if(argc==4){ mess_matrix_init(&E);}
mess_matrix_init(&Z);
```

Now we can use mess_matrix_read_formated to read our matrices from a Matrix Market File Format and convert it to the storage type MESS_CSR and MESS_DENSE.
mess_matrix_read_formated dynamical allocates memory at this point.

```
mess_matrix_read_formated(argv[1],F,MESS_CSR);
mess_matrix_read_formated(argv[2],G,MESS_DENSE);
if(argc==4){ mess_matrix_read_formated(argv[3],E,
  MESS_CSR);}
```

4. Solve the (Generalized) Lyapunov Equation

As already announced mess_lyap will solve the (generalized) Lyapunov Equation. We can indicate the case of the normal Lyapunov Equation by providing `NULL` as the second argument to mess_lyap.

```
if(argc==3){
    mess_lyap(F,NULL,G,Z);
}else{
    mess_lyap(F,E,G,Z);
}
```

5. Compute the Relative Residual

We declare two `double` variables for storing the right-hand side norm and the absolute residual. mess_matrix_dynorm2 computes the norm of the right-hand side and mess_lrcfadi_res2 or mess_lrcfadi_res2g are helpfull to compute the desired 2-norms.

```
double nrmRHS=1, nrm=1;
mess_matrix_dynorm2(G,&nrmRHS);
if(argc==3){
    mess_lrcfadi_res2(F,G,Z,&nrm);
}else{
    mess_lrcfadi_res2g(F,E,G,Z,&nrm);
}
printf("relative Residual= %10.15e\n",nrm/nrmRHS);
```

6. Don't Forget to Clear Memory

A rule of thumb is that every structure which needs an `init` call before usage must be cleared at the at the end of your code. In our cases we have used mess_matrix_init which means we have to use mess_matrix_clear to clear dynamical allocated memory.

```
mess_matrix_clear(&F);
mess_matrix_clear(&G);
mess_matrix_clear(&Z);
if(argc==4)
    mess_matrix_clear(&E);
```

See also

Tutorials

10.717 tutorial_matrix_operations.c File Reference

10.717.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Tutorial on Matrix Operations

In this tutorial we show some algebraic matrix operations. We will perform the following steps

$$B \leftarrow A^T,$$
$$B \leftarrow 2A + A^T,$$
$$B \leftarrow (1+i)B,$$
$$B \leftarrow \overline{B},$$
$$C \leftarrow A * B^H,$$
$$D \leftarrow \mathrm{Re}(C),$$
$$D \leftarrow \mathrm{Im}(C).$$
$$||A||_F,\ ||A||_\infty, \ldots$$

1. Include Header Files

We include standard C header files and in addtion the mess.h header file to make the **M.E.S.S.** library available.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "mess/mess.h"
```

2. (Optional) Print M.E.S.S. Status Informations

You can use mess_version to print status informations about your **M.E.S.S.** version.

```
    mess_version();
```

3. Declare, Init and Read Matrices

At first we declare some mess_matrix structures.

```
    mess_matrix input, A, B, C, D;
```

Now we check the number of input arguments and init the mess_matrix structures.

```
    if ( argc != 2){
        printf("usage: %s <file.mtx>\n", argv[0]);
        return 1;
    }
    mess_matrix_init(&A);
    mess_matrix_init(&B);
    mess_matrix_init(&C);
    mess_matrix_init(&D);
    mess_matrix_init(&input);
```

Reading from MatrixMarket file format works the following way.

```
    mess_matrix_read(argv[1], input);
    mess_matrix_convert(input, A, MESS_CSR);
```

Note that the required memory is allocated by mess_matrix_read. mess_matrix_convert converts to MESS_CSR storage format and the required memory is allocated by mess_matrix_convert.

4. Transpose, Add and Print

Transposing, Adding, Scaling and Printing matrices is performed by mess_matrix_ctranspose, mess_matrix_add, mess_matrix_scale / mess_matrix_scalec and mess_matrix_print. Most functions of **M.E.S.S.** follow the convention that the *"input"* or *"source"* argument(s) is (are) the first one(s) and the *"output"* or *"target"* argument(s) is (are) the last one(s). The prefix *mess_matrix* indicates that the function is related to mess_matrix structures. Now let us do the following operations:

$$B \leftarrow A^T,$$
$$B \leftarrow 2A + A^T.$$

```
mess_matrix_ctranspose(A, B);

mess_matrix_add(2.0, A, 1.0, B);
```

5. Scale, Complex Conjugate, Multiply and Real/Imaginary Part

Next, we perform

$$B \leftarrow (1+i)B,$$
$$B \leftarrow \overline{B},$$
$$C \leftarrow A * B^H,$$
$$A \leftarrow \mathrm{Re}(C),$$
$$A \leftarrow \mathrm{Im}(C).$$

```
mess_matrix_scalec(1+1*I,B);

mess_matrix_conj(B);

mess_matrix_multiply(MESS_OP_NONE,A,
  MESS_OP_HERMITIAN, B, C);

mess_matrix_realpart(C,D);

mess_matrix_imagpart(C,D);
```

6. Compute Martix Norms

M.E.S.S. contains the following functions to compute matrix norms:

- mess_matrix_norm1, to compute the 1-norm,
- mess_matrix_norm2, to compute the 2-norm,
- mess_matrix_norminf, to compute the ∞-norm and
- mess_matrix_normf, to compute the Frobenius norm. Depending on the storage scheme of the matrix the 2-norm computation may get expensive. The example computes all four norms of a previously generated matrix C.

```
double nrm1,nrm2,nrmf, nrminf;
mess_matrix_norm1(A, &nrm1);
mess_matrix_norminf(A, &nrminf);
mess_matrix_norm2(A, &nrm2);
mess_matrix_normf(A, &nrmf);

printf (" 1-Norm:  \t %g\n", nrm1);
printf (" Inf-Norm:\t %g\n", nrminf);
printf (" F-Norm:  \t %g\n", nrmf);
printf (" 2-Norm:  \t %g\n", nrm2);
```

7. Clear Memory

```
mess_matrix_clear(&A);
mess_matrix_clear(&B);
mess_matrix_clear(&C);
mess_matrix_clear(&D);
mess_matrix_clear(&input);
return 0;
```

See also

Tutorials

10.718 tutorial_norms.c File Reference

10.718.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Tutorial for Computing Different Norms of a Matrix

This function demonstrates computing different matrix norms for matrix A

- 1-norm (column-sum-norm)
- 2-norm (largest eigenvalue of $A^H A$)
- ∞-norm (row-sum-norm)
- Frobenius-norm

with different storage formats (CSC, CSC, COORD).
Additionally the rank of a matrix is computed for CSR storage format.

See also

mess_matrix_norm1
mess_matrix_norm2
mess_matrix_norminf
mess_matrix_normf
mess_matrix_rank

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "mess/mess.h"

int main (int argc, char *argv[])
{
    mess_matrix mat_coord;
    mess_matrix mat_csr;
    mess_matrix mat_csc;
    double nrm1,nrm2,nrmf, nrminf;
    mess_int_t rank =0;

    MESS_INIT_MATRICES(&mat_coord,&mat_csr,&mat_csc);

    mess_error_level = 1;


    if ( argc != 2) {
        printf("compute the norms of a matrix\n");
        printf("usage: %s matrix.mtx\n", argv[0]);
    }

    mess_matrix_read(argv[1], mat_coord);
    mess_matrix_convert(mat_coord, mat_csr, MESS_CSR);
    mess_matrix_convert(mat_csr, mat_csc, MESS_CSC);


    printf ("CSR: \n");
    mess_matrix_norm1(mat_csr, &nrm1);
    mess_matrix_norminf(mat_csr, &nrminf);
    mess_matrix_rank(mat_csr, &rank);
    mess_matrix_norm2(mat_csr, &nrm2);
    mess_matrix_normf(mat_csr, &nrmf);
    printf (" 1-Norm:  \t %g\n", nrm1);
    printf (" Inf-Norm:\t %g\n", nrminf);
    printf (" F-Norm:  \t %g\n", nrmf);
    printf (" 2-Norm:  \t %g\n", nrm2);
    printf (" Rank:    \t " MESS_PRINTF_INT "\n", rank);

    printf ("CSC: \n");
    mess_matrix_norm1(mat_csc, &nrm1);
    mess_matrix_norminf(mat_csc, &nrminf);
    mess_matrix_norm2(mat_csc, &nrm2);
    mess_matrix_normf(mat_csc, &nrmf);

    printf (" 1-Norm:  \t %g\n", nrm1);
    printf (" Inf-Norm:\t %g\n", nrminf);
    printf (" F-Norm:  \t %g\n", nrmf);
    printf (" 2-Norm:  \t %g\n", nrm2);


    printf ("COORD: \n");
    mess_matrix_norm1(mat_coord, &nrm1);
    mess_matrix_norminf(mat_coord, &nrminf);
    mess_matrix_norm2(mat_coord, &nrm2);
    mess_matrix_normf(mat_coord, &nrmf);

    printf (" 1-Norm:  \t %g\n", nrm1);
    printf (" Inf-Norm:\t %g\n", nrminf);
    printf (" F-Norm:  \t %g\n", nrmf);
    printf (" 2-Norm:  \t %g\n", nrm2);

    printf("\n2-Norm inv(A)\n");
    mess_matrix_norm2inv(mat_csr, &nrm2);
    printf(" = %lg\n", nrm2);

    MESS_CLEAR_MATRICES(&mat_coord,&mat_csr,&mat_csc);

    return 0;
}
```

10.719 tutorial_plot.c File Reference

10.719.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Tutorial Plotting Interface

In this tutorial we use the plotting interface to plot the sine and cosine function.
More precisely we plot $20\sin(x),\ 20\cos(x),\ 20\sin(x)+20\cos(x)$. We export the graphics to a `gnuplot` script as well as a `TikZ` script.

1. Include Header Files

We include standard `C` header files and in addtion the `mess.h` header file to make the **M.E.S.S.** library available.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <math.h>
#include "mess/mess.h"
```

2. Declare mess_plotExport and additional variables.

We use the mess_plotExport type to generate a plot interface object and some additional variables to generate the data.

```
mess_int_t i, N=100;
double x, y1, y2, h = 2.0*3.14159/N;
mess_plotExport plot;
```

3. Init mess_plotExport Instance

Before we are able to use the mess_plotExport instance we have to init the fields of the structure. Therefore we use mess_plotExport_init. We name our plot `TestPlot` choose MESS_PLOT_LIN for linear scaling the `x-axis` as well as the `y-axis`. For logarithmic scaling use MESS_PLOT_LOG. The label of the `x-axis` is `x` and the lable fo the `y-axis` is `y`.
The last argument indicates that we want to store `3` datasets.

```
mess_plotExport_init(&plot, "TestPlot", MESS_PLOT_LIN,
  MESS_PLOT_LIN, "x", "y", 3);
```

4. Adjust the Plot

Now we are ready to use the `plot` instance of mess_plotExport. We use the functions mess_plotExport_set↩LegendPos, mess_plotExport_setLabel, mess_plotExport_setType to specify the position of the legend and for each dataset the label, type and color of the plot. The second argument is used to reference to the dataset.
You can find more options by reading the corresponding documentation of the functions.

```
mess_plotExport_setLegendPos(plot, "belowCenter");
mess_plotExport_setLabel(plot,  0, "20\\sin(x)");
mess_plotExport_setColor(plot,  0, "blue");
mess_plotExport_setType(plot,   0, "lines");
mess_plotExport_setLabel(plot,  1, "20\\cos(x)");
mess_plotExport_setColor(plot,  1, "yellow");
mess_plotExport_setType(plot,   1, "lines");
mess_plotExport_setLabel(plot,  2, "20\\sin(x)+20\\cos(x)");
mess_plotExport_setColor(plot,  2, "green");
mess_plotExport_setType(plot,   2, "lines");
printf("\nNew Plot initialised: \"TestPlot\" with 3 datasets. Keybox orientation: belowCenter\n");
```

5. Fill data

To fill the `plot` instance with data we use a simple `for` loop compute the function values of the functions and use mess_plotExport_addData to add the arguments as well as the function values to the corresponding data set.

```
for ( i = 0; i <= N; i++){
    x = h*i;
    y1 = sin(x)*20;
    y2 = cos(x)*20;
    mess_plotExport_addData(plot,0, x,y1 );
    mess_plotExport_addData(plot,1, x,y2 );
    mess_plotExport_addData(plot,2, x,y1+y2 );
}
```

6. Export

Exporting the plot is very easy using mess_plotExport_createGnuScript and mess_plotExport_createTikzScript to export the data. The function create a `gnuplot` script and a datafile and a `TikZ` file with corresponding data file. The name of the generated files starts with `TestPlot*`. You can also use mess_plotExport_createTikzScript_filename or mess_plotExport_createGnuScript_filename to specify the name of the output.

```
printf("\ngnu_plot_script runs\n");
mess_plotExport_createGnuScript(plot);

printf("Tikz_plot_script runs\n");
mess_plotExport_createTikzScript(plot);
```

7. Clear mess_plotExport instance.

Do not forget to release the memory allocated by mess_plotExport_init by calling mess_plotExport_clear.

```
mess_plotExport_clear(&plot);
```

See also

Tutorials

10.720 tutorial_qr.c File Reference

10.720.1 Detailed Description

See also

mess_direct_create_lapack_qr
mess_direct_solve
mess_direct_solvem

Tutorial: QR solver

This function demonstrates how to generate a QR solver to solve equation

$$Ax = b$$

or

$$AX = B,$$

where A and B are matrices and b is a vector.

```
#include <stdio.h>
#include <stdlib.h>
#include "mess/mess.h"

int main ( int argc, char **argv) {
    mess_matrix A,RHS,X;
    mess_vector b,x;
    mess_direct qr;

    mess_set_errorlevel(3);
    mess_version();

    /*-----------------------------------------------------------------------------
     *  check input arguments
     *-----------------------------------------------------------------------------*/
    if ( argc != 2 ) {
        printf("Usage: %s file\n", argv[0]);
        return -1;
    }

    /*-----------------------------------------------------------------------------
     *  init matrices
     *-----------------------------------------------------------------------------*/
    MESS_INIT_MATRICES(&A,&RHS,&X);

    mess_matrix_read_formated(argv[1],A,MESS_DENSE);

    /*-----------------------------------------------------------------------------
     *  init vectors
     *-----------------------------------------------------------------------------*/
    MESS_INIT_VECTORS(&x,&b);
    mess_vector_alloc(x, A->rows, A->data_type);
    mess_vector_alloc(b, A->rows, A->data_type);

    mess_matrix_rowsums(A,b);

    /*-----------------------------------------------------------------------------
     *  create qr solver
     *-----------------------------------------------------------------------------*/
    mess_direct_init(&qr);
    mess_direct_create_lapack_qr(A,qr);

    /*-----------------------------------------------------------------------------
     *  solve system and plot solution
     *-----------------------------------------------------------------------------*/
    mess_direct_solve(MESS_OP_NONE,qr, b, x);
    printf("Ax=b:\n");
    mess_vector_printshort(x);

    /*-----------------------------------------------------------------------------
     *  create rhs and plot solution
     *-----------------------------------------------------------------------------*/
    mess_matrix_alloc(RHS, A->rows, 2, A->rows*2,
      MESS_DENSE, A->data_type);
    mess_matrix_setcol(RHS, 0, b);
    mess_vector_scalee(2.0, b);
    mess_matrix_setcol(RHS, 1, b);
    mess_direct_solvem(MESS_OP_NONE,qr,RHS, X);

    printf("AX=B:\n");
    mess_matrix_print(X);

    /*-----------------------------------------------------------------------------
     *  clear matrices
     *-----------------------------------------------------------------------------*/
    MESS_CLEAR_MATRICES(&A,&X,&RHS);
    MESS_CLEAR_VECTORS(&x,&b);
    mess_direct_clear(&qr);

    return 0;
}
```

10.721 tutorial_schur.c File Reference

10.721.1 Detailed Description

See also

mess_eigen_schur
mess_eigen_schur_complex

Tutorial: Compute Schur Decomposition

This function demonstrates computing the Schur decomposition of a matrices A

$$T = U^T AU$$

where T is the Schur form of A and U is an orthogonal matrix and the eigenvalues of the eigenvalue problem:

$$Ax = \lambda x.$$

using LAPACK's DGEES subroutine.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include "mess/mess.h"
#include <complex.h>

int main ( int argc, char **argv){
    mess_matrix A, T, U;
    mess_vector EV;
    int ret = 0;
    int type = 1;
    mess_version();

    printf("mess schur decomposition demo\n");
    printf("============================\n");

    if ( argc!=3){
        printf("Usage: %s matrix.mtx type\n", argv[0]);
        printf(" type = 1 real schur form\n");
        printf(" type = 2 complex schur form\n");
        exit(1);
    }
    type = atoi(argv[2]);
    mess_matrix_init(&A);
    mess_matrix_init(&T);
    mess_matrix_init(&U);
    mess_vector_init(&EV);
    mess_vector_alloc(EV, 1, MESS_COMPLEX);

    if ( mess_matrix_read(argv[1], A) != 0){
        printf("error reading matrix\n");
    }
    if ( type == 1) {
        ret = mess_eigen_schur(A,T,U, EV);
    } else if ( type ==2){
        ret = mess_eigen_schur_complex(A,T,U, EV);
    }else{
        ret =1;
    }
    if ( ret !=0) {
        printf("an error occured. ret = %d\n", ret);
        return ret;
    } else{
        printf("T=\n");
        mess_matrix_print(T);
        printf("\nU=\n");
        mess_matrix_print(U);
        printf("\nEigenvalues:\n");
        mess_vector_print(EV);
    }

    mess_matrix_clear(&A);
    mess_matrix_clear(&T);
    mess_matrix_clear(&U);
    mess_vector_clear(&EV);
    return 0;
}
```

10.722 tutorial_sign.c File Reference

10.722.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Tutorial for computing the sign of a matrix.

In this tutorial we show how to use the mess_eigen_sign function to compute the sign of a function. We compute the eigenvalues of $Z = sign(A)$. We also check if $Z^2 = I_n$.

1. Include Header Files

We include standard C header files and in addtion the mess.h header file to make the **M.E.S.S.** library available.

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "mess/mess.h"
```

2. Check Input arguments, Init Matrix and Read Data

We declare four mess_matrix instances, one mess_vector instance and three double value variables.
We check the number of input arguments by using the argc variable and use mess_matrix_read_formated to read from a MatrixMarket file directly into a mess_matrix with storage type MESS_DENSE.

```
    mess_matrix A, Z, Z2, eye;
    mess_vector ev;
    double ts, te, diff;

    if ( argc != 2 ) {
        printf("usage: %s <FILE>\n", argv[0]);
        exit(1);
    }

    MESS_INIT_MATRICES(&A,&Z,&Z2,&eye);
    mess_matrix_read_formated(argv[1],A,MESS_DENSE);
```

3. Compute the sign

The method mess_eigen_sign takes the matrix A as argument, stores the result $Z = sign(A)$ and the last argument is the mess_sign_scale_t enum type to speedup the the computation.
Using mess_wtime we can measure the elapsed wall clock time.

```
    ts = mess_wtime();
    mess_eigen_sign(A,Z,MESS_SIGN_SCALE_NONE);
    te = mess_wtime();
    printf("Sign function tooks %lg seconds\n", te-ts);
```

4. CHECK

The eigenvalues of Z should be either 1 or -1 and Z should fullfill $Z^2 - I_n = 0_n$.
We want to check these properties. Therefore we allocate a mess_vector instance for mess_eigen_eig to compute the eigenvalues. Since we are not interested in the eigenvectors the last argument of mess_eigen_eig is `NULL`.
In order to check if $Z^2 - I_n = 0_n$, we use mess_matrix_eye to create a unit matrix `eye`. The second step is to compute Z^2 and store the result in `Z2`. We check the identity $Z^2 - I_n$ by computing $||Z^2 - I_n||_2$. The spectral norm of the difference of two matrices can be computed with mess_matrix_diffnorm and the frobenius norm of the difference with mess_matrix_diffnormf.

```
mess_vector_init(&ev);
mess_vector_alloc(ev,A->rows,MESS_COMPLEX);
mess_eigen_eig(Z,ev,NULL);
printf("Eigenvalues of A\n");
mess_vector_printshort(ev);

mess_matrix_eye(eye,Z->rows,Z->cols,MESS_DENSE);
mess_matrix_multiply(MESS_OP_NONE,Z,
  MESS_OP_NONE,Z,Z2);
mess_matrix_diffnorm(Z2,eye,&diff);
printf("||sign(A)^2-I||_2=%e\n",diff);
```

5. Clear the memory

Allocated memory has to be cleared. MESS_CLEAR_MATRICES and MESS_CLEAR_VECTORS do this job.

```
MESS_CLEAR_MATRICES(&A,&Z,&Z2,&eye);
MESS_CLEAR_VECTORS(&ev);
```

See also

Tutorials

10.723 tutorial_spy.c File Reference

10.723.1 Detailed Description

Author

Maximilian Behr (`behr@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)
Martin Köhler (`koehlerm@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

Tutorial Plot Sparsity Pattern of a Matrix

In this tutorial we show how to use the mess_matrix_spy function to plot the sparsity pattern of a matrix. We use mess_matrix_read_formated to read a matrix from a `MatrixMarket` file.

1. Include Header Files

We include standard C header files and in addtion the `mess.h` header file to make the **M.E.S.S.** library available.

```
#include <stdio.h>
#include <stdlib.h>
#include "mess/mess.h"
```

2. Check Input arguments, Init Matrix and Read Data

We check the number of input arguments by using the `argc` variable. We further assume that the first input argument is the path to the `MatrixMarket` file, the second and third one is the width and height of the resulting bitmap plot and the last one is the path to the resulting bitmap file.
The second step is to initialize the mess_matrix structure using mess_matrix_init and use mess_matrix_read_formated to read directly into MESS_CSR format. We use `atoi` to read the width and height from the command line.

```
if(argc!=5){
    printf("usage: %s A.mtx width height output.bmp",argv[0]);
    return 1;
}

mess_matrix_init(&A);
mess_matrix_read_formated(argv[1],A,MESS_CSR);
w = atoi(argv[2]);
h = atoi(argv[3]);
```

3. Generate Spy Plot

The calling sequence of mess_matrix_spy is easy. The first input argument is the desired matrix, the second and third one is the width and height of the resulting bitmap image. The last input argument indicates the path to the bitmap image.

```
mess_matrix_spy(A, argv[4], w, h);
```

4. Clear Memory

We allocated memory by mess_matrix_init and mess_matrix_read_formated, so we have to clear it and use mess_matrix_clear.

```
mess_matrix_clear(&A);
```

See also

Tutorials

10.724 tutorial_sylvester.c File Reference

10.724.1 Detailed Description

Author

Robert Dykstra (dykstra@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

Tutorial Solving a (generalized) sparse-dense Sylvester Equation

This tutorial demonstrates how to solve sparse-dense Sylvester equations

$$op(A)X[op(F)] + [op(E)]Xop(H) + M = 0,$$

where A and E are large and sparse matrices, F and H are small and dense matrices and M is a matrix, by generating a solver and using the mess_direct_solvem function.
E and F may or may not be used, there are solvers for three cases (further info below).
Operations $op(.)$ on matrices A and H can be

- MESS_OP_NONE ($op(C) = C$),
- MESS_OP_TRANSPOSE ($op(C) = C^T$),
- MESS_OP_HERMITIAN ($op(C) = C^H$).

In this program we have to differentiate between the three cases of sparse-dense Sylvester equations the mess_direct_create_sylvester_sparsedense function can generate solvers for :

- $op(A)X + Xop(H) + M = 0$
- $op(A)X + op(E)Xop(H) + M = 0$
- $op(A)Xop(F) + op(E)Xop(H) + M = 0$

After solving for X we compute the relative residual $\|op(A)X[op(F)] + [op(E)]Xop(H) + M\|_2 / \|\|M\|_2$ and clear the memory.

1. Include Header Files

We include standard `C` header files and in addtion the `mess.h` header file to make the **`M.E.S.S.`** library available and the `complex.h` header file for handling of complex matrices.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include "mess/mess.h"
#include <complex.h>
```

2. Print Title (optional)

Print the M.E.S.S. version and the title of the demo.

```
mess_version();
printf("mess sylvester solver demo\n");
printf("=========================\n\n");
```

3. Declare Matrices and Check the Number of Input Arguments.

We use the mess_matrix structure to declare our matrices A, F, E, H, M and $X1, X2$ for operations MESS_OP_NONE and MESS_OP_HERMITIAN. We also declare a mess_direct structure to store the generated Sylvester solver.

```
mess_matrix A,F=NULL,E=NULL,H,M,X1,X2;
mess_direct sylv;
```

By using the `argc` variable we can check the number of input arguments.

```
if ( !( 3 <= argc && argc <= 5)){
    printf("usage: %s A.mtx [F.mtx] [E.mtx] H.mtx\n", argv[0]);
    return -1;
}
```

4. Init Matrices and Read Data

Before we are able to use we have to init the fields of the mess_matrix structure variables using mess_matrix_init. Be careful to catch the different input cases.

```
mess_matrix_init(&A);
mess_matrix_init(&F);
mess_matrix_init(&E);
mess_matrix_init(&H);
mess_matrix_init(&M);
mess_matrix_init(&X1);
mess_matrix_init(&X2);
```

Now we can use mess_matrix_read or mess_matrix_read_formated to read our matrices from a Matrix Market File Format and convert A and E to the storage type MESS_CSR if needed. Also differentiate the input cases here.

```
mess_matrix_read_formated(argv[1], A, MESS_CSR);
if( argc == 3 ){
    mess_matrix_read_formated(argv[2], H,
  MESS_DENSE);
} else if( argc == 4 ){
    mess_matrix_read_formated(argv[2], E, MESS_CSR);
    mess_matrix_read(argv[3], H);
} else {
    mess_matrix_read_formated(argv[2], F,
  MESS_DENSE);
    mess_matrix_read_formated(argv[3], E, MESS_CSR);
    mess_matrix_read_formated(argv[4], H,
  MESS_DENSE);
}
mess_int_t seed = 1234;
mess_matrix_rand_init(&seed);
mess_matrix_rand_dense(M, A->rows, H->rows,
  MESS_REAL);
```

5. Generate the Sylvester Solver and Solve

As already announced mess_direct_create_sylvester_sparsedense creates a solver for the Sylvester equation. We can use this solver with mess_direct_solvem to solve the (generalized) Sylvester Equation.
We measure the time using mess_wtime.
The solver is initialized using mess_direct_init and created using mess_direct_create_sylvester_sparsedense. We can indicate the case of the standard Sylvester Equation by providing `NULL` as the second and third argument to mess_direct_create_sylvester_sparsedense, as well as the semi-generalized case $op(A)X + op(E)Xop(H) + M = 0$ by providing NULL as second argument.

```
double ts, te;
ts = mess_wtime();
mess_direct_init(&sylv);
mess_direct_create_sylvester_sparsedense(A,NULL,NULL,H,sylv);
mess_direct_solvem(MESS_OP_NONE,sylv,M,X1);
mess_direct_solvem(MESS_OP_HERMITIAN,sylv,M,X2);
mess_direct_clear(&sylv);
te = mess_wtime();
printf("time: %.3f\n\n", te-ts);
```

6. Compute the Relative Residual

mess_matrix_norm2 computes the norm of M, which we divide from mess_direct_sylvester_res2, mess_direct↩_sylvestersg_res2 or mess_direct_generalized_sylvester_res2 to get the absolute residual. The relative residual residual can easily obtained by computing the norm of the right hand side.

```
double res = 0, res0 = 0;
mess_matrix_norm2(M, &res0);
if(argc == 3){
    mess_direct_sylvester_res2(MESS_OP_NONE,A,H,M,X1,&res);
    printf("||AX+XH+M||_2 / ||M||_2 = %.10e\n", res/res0);
    mess_direct_sylvester_res2(MESS_OP_HERMITIAN,A,H,M,X2,&
  res);
    printf("||A^HX+XH^H+M||_2 / ||M||_2 = %.10e\n", res/res0);
} else if (argc == 4){
    mess_direct_sylvestersg_res2(MESS_OP_NONE,A,E,H,M,X1,&res);
    printf("||AX+EXH+M||_2 / ||M||_2 = %.10e\n", res/res0);
    mess_direct_sylvestersg_res2(
  MESS_OP_HERMITIAN,A,E,H,M,X2,&res);
    printf("||A^HX+E^HXH^H+M||_2 / ||M||_2 = %.10e\n", res/res0);
} else {
    mess_direct_generalized_sylvester_res2(
  MESS_OP_NONE,A,F,E,H,M,X1,&res);
    printf("||AXF+EXH+M||_2 || / ||M||_2 = %.10e\n", res/res0);
    mess_direct_generalized_sylvester_res2(
  MESS_OP_HERMITIAN,A,F,E,H,M,X2,&res);
    printf("||A^HXF^H+E^HXH^H+M||_2 / ||M||_2 = %.10e\n", res/res0);
}
```

7. Don't Forget to Clear Memory

A rule of thumb is that every structure which needs an `init` call before usage must be cleared at the at the end of your code. In our cases we have used mess_matrix_init which means we have to use mess_matrix_clear to clear dynamical allocated memory (we already cleared the mess_direct instance.

```
    mess_matrix_init(&A);
    if(F)mess_matrix_init(&F);
    if(E)mess_matrix_init(&E);
    mess_matrix_init(&H);
    mess_matrix_init(&M);
    mess_matrix_init(&X1);
    mess_matrix_init(&X2);
```

For a special dense Sylvester solve take a look at mess_direct_create_sylvester_dense.

See also

mess_direct_create_sylvester_sparsedense
mess_direct_create_sylvester_sparsedense
mess_direct_solvem

10.725 tutorial_tsia.c File Reference

10.725.1 Detailed Description

See also

mess_h2_tsia

Tutorial: TSIA Algorithm

This function demonstrates computing a $\mathcal{H}_2$ reduced linear time invariant (LTI) system using the TSIA algorithm.

10.726 tutorial_vector_read.c File Reference

10.726.1 Detailed Description

See also

mess_vector_rand
mess_vector_write
mess_vector_read
mess_vector_print

Tutorial: Read, Write and Generate a Random Vector

This function demonstrates how to generate a random vector, to write it to a file, to read this vector from a file and to print it.

```
#include <stdio.h>
#include <stdlib.h>
#include "mess/mess.h"
#include <complex.h>

int main (int argc, char ** argv) {
    mess_vector a, b;
    mess_vector_init(&a);
    mess_vector_init(&b);
    mess_vector_alloc(a, 10, MESS_REAL);
    mess_vector_alloc(b, 10, MESS_REAL);
    mess_vector_rand(a);
    printf("Random vector:\n");
    mess_vector_print(a);
    mess_vector_write("test.mtx", a );
    mess_vector_read("test.mtx", b);
    printf("Read Vector:\n");
    mess_vector_print(b);
    mess_vector_clear(&a);
    mess_vector_clear(&b);
    return 0;
}
```

10.727 tutorials.dox File Reference

10.728 twoside_mgs.c File Reference

Macros

- #define REFINE 1
- #define KAPPA 0.1

Functions

- int mess_matrix_proj_mgs (mess_vector in, mess_matrix Q, mess_matrix QT, mess_int_t colcount, mess_vector out, double *nrmout)

  Skew projection using Q and QT.
- int mess_matrix_proj_gmgs (mess_vector in, mess_matrix Q, mess_matrix QT, mess_matrix E, char *Eflag, mess_int_t colcount, mess_vector out, double *nrmout)

  Generalized skew projection using Q and QT.

10.728.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.728.2 Macro Definition Documentation

10.728.2.1 #define KAPPA 0.1

10.728.2.2 #define REFINE 1

10.729 umfpack.c File Reference

Macros

- #define umfpack_dl_solve umfpack_di_solve
- #define umfpack_zl_solve umfpack_zi_solve
- #define umfpack_zl_load_numeric umfpack_zi_load_numeric
- #define umfpack_dl_load_numeric umfpack_di_load_numeric
- #define umfpack_zl_defaults umfpack_zi_defaults
- #define umfpack_dl_defaults umfpack_di_defaults
- #define umfpack_dl_save_numeric umfpack_di_save_numeric
- #define umfpack_zl_save_numeric umfpack_zi_save_numeric
- #define umfpack_zl_free_numeric umfpack_zi_free_numeric
- #define umfpack_dl_free_numeric umfpack_di_free_numeric
- #define umfpack_dl_symbolic umfpack_di_symbolic
- #define umfpack_dl_numeric umfpack_di_numeric
- #define umfpack_zl_symbolic umfpack_zi_symbolic
- #define umfpack_zl_numeric umfpack_zi_numeric
- #define umfpack_dl_report_info umfpack_di_report_info
- #define umfpack_dl_free_symbolic umfpack_di_free_symbolic
- #define umfpack_zl_report_info umfpack_zi_report_info
- #define umfpack_zl_free_symbolic umfpack_zi_free_symbolic
- #define umfpack_zl_get_lunz umfpack_zi_get_lunz
- #define umfpack_zl_get_numeric umfpack_zi_get_numeric
- #define umfpack_dl_get_lunz umfpack_di_get_lunz
- #define umfpack_dl_get_numeric umfpack_di_get_numeric

Functions

- int mess_direct_create_umfpack (mess_matrix matrix, mess_direct solver)
  Generate a direct linear solver for standard linear systems $Ax = b$ with UMFPACK

10.729.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.729.2 Macro Definition Documentation

10.729.2.1 #define umfpack_dl_defaults umfpack_di_defaults

10.729.2.2 #define umfpack_dl_free_numeric umfpack_di_free_numeric

10.729.2.3 #define umfpack_dl_free_symbolic umfpack_di_free_symbolic

10.729.2.4 #define umfpack_dl_get_lunz umfpack_di_get_lunz

10.729.2.5 #define umfpack_dl_get_numeric umfpack_di_get_numeric

10.729.2.6 #define umfpack_dl_load_numeric umfpack_di_load_numeric

10.729.2.7 #define umfpack_dl_numeric umfpack_di_numeric

10.729.2.8 #define umfpack_dl_report_info umfpack_di_report_info

10.729.2.9 #define umfpack_dl_save_numeric umfpack_di_save_numeric

10.729.2.10 #define umfpack_dl_solve umfpack_di_solve

10.729.2.11 #define umfpack_dl_symbolic umfpack_di_symbolic

10.729.2.12 #define umfpack_zl_defaults umfpack_zi_defaults

10.729.2.13 #define umfpack_zl_free_numeric umfpack_zi_free_numeric

10.729.2.14 #define umfpack_zl_free_symbolic umfpack_zi_free_symbolic

10.729.2.15 #define umfpack_zl_get_lunz umfpack_zi_get_lunz

10.729.2.16 #define umfpack_zl_get_numeric umfpack_zi_get_numeric

10.729.2.17 #define umfpack_zl_load_numeric umfpack_zi_load_numeric

10.729.2.18 #define umfpack_zl_numeric umfpack_zi_numeric

10.729.2.19 #define umfpack_zl_report_info umfpack_zi_report_info

10.729.2.20 #define umfpack_zl_save_numeric umfpack_zi_save_numeric

10.729.2.21 #define umfpack_zl_solve umfpack_zi_solve

10.729.2.22 #define umfpack_zl_symbolic umfpack_zi_symbolic

10.730 umfpack_control.c File Reference

Data Structures

- struct umfpack_control_solver

Macros

- #define umfpack_dl_solve umfpack_di_solve
- #define umfpack_zl_solve umfpack_zi_solve
- #define umfpack_zl_load_numeric umfpack_zi_load_numeric
- #define umfpack_dl_load_numeric umfpack_di_load_numeric
- #define umfpack_zl_defaults umfpack_zi_defaults
- #define umfpack_dl_defaults umfpack_di_defaults
- #define umfpack_dl_save_numeric umfpack_di_save_numeric
- #define umfpack_zl_save_numeric umfpack_zi_save_numeric
- #define umfpack_zl_free_numeric umfpack_zi_free_numeric
- #define umfpack_dl_free_numeric umfpack_di_free_numeric
- #define umfpack_dl_symbolic umfpack_di_symbolic
- #define umfpack_dl_numeric umfpack_di_numeric
- #define umfpack_zl_symbolic umfpack_zi_symbolic
- #define umfpack_zl_numeric umfpack_zi_numeric
- #define umfpack_dl_report_info umfpack_di_report_info
- #define umfpack_dl_free_symbolic umfpack_di_free_symbolic
- #define umfpack_zl_report_info umfpack_zi_report_info
- #define umfpack_zl_free_symbolic umfpack_zi_free_symbolic
- #define umfpack_zl_get_lunz umfpack_zi_get_lunz
- #define umfpack_zl_get_numeric umfpack_zi_get_numeric
- #define umfpack_dl_get_lunz umfpack_di_get_lunz
- #define umfpack_dl_get_numeric umfpack_di_get_numeric
- #define umfpack_dl_get_determinant umfpack_di_get_determinant
- #define umfpack_zl_get_determinant umfpack_zi_get_determinant
- #define UMFPACK_CONTROL_LOG2_10 3.321928094887362347870319429489390175864831393024

Typedefs

- typedef struct umfpack_control_solver umfpack_control_solver

Functions

- static int umfpack_control_getL (void ∗data, mess_matrix L)

  Get the L factor of a matrix.
- static int umfpack_control_getU (void ∗data, mess_matrix U)

  Get the U factor of a matrix.
- static int umfpack_control_getpermp (void ∗data, mess_int_t ∗p)

  Get the row permutation of the matrix.
- static int umfpack_control_getpermq (void ∗data, mess_int_t ∗q)

  Get the column permutation of a matrix.
- static int umfpack_control_getscalerow (void ∗data, mess_vector r)

  Get the row scaling of a matrix.
- static int umfpack_control_solve (void ∗data, mess_vector b, mess_vector x)

  Solve $Ax = b$.
- static int umfpack_control_solvet (void ∗data, mess_vector b, mess_vector x)

  Solve $A^T x = b$.
- static int umfpack_control_solveh (void ∗data, mess_vector b, mess_vector x)

  Solve $A^H x = b$.
- static int umfpack_control_solvem (void ∗data, mess_matrix b, mess_matrix x)

  Solve $AX = B$ *(matrix version)*

- static int umfpack_control_solvemt (void ∗data, mess_matrix b, mess_matrix x)

  Solve $A^T X = B$ (matrix version)
- static int umfpack_control_solvemh (void ∗data, mess_matrix b, mess_matrix x)

  Solve $A^H X = B$ (matrix version)
- static int umfpack_control_clear (void ∗solver)

  Clear an umfpack_control_solver object.
- static int umfpack_control_inverse (void ∗data, mess_matrix inv)

  Compute the inverse of a matrix.
- static int umfpack_control_det (void ∗data, double ∗m, double ∗e)

  Compute the determinant of a matrix using UMFPACK.
- static int umfpack_control_detc (void ∗data, double ∗mr, double ∗mi, double ∗e)

  Compute the determinant of a matrix using UMFPACK.
- int mess_direct_create_umfpack_control (mess_matrix matrix, mess_direct solver, double ∗Control)

  Generate a direct linear solver for standard linear systems $Ax = b$ with UMFPACK.

10.730.1 Detailed Description

Author

Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.730.2 Macro Definition Documentation

10.730.2.1 #define UMFPACK_CONTROL_LOG2_10 3.321928094887362347870319429489390175864831393024

10.730.2.2 #define umfpack_dl_defaults umfpack_di_defaults

10.730.2.3 #define umfpack_dl_free_numeric umfpack_di_free_numeric

10.730.2.4 #define umfpack_dl_free_symbolic umfpack_di_free_symbolic

10.730.2.5 #define umfpack_dl_get_determinant umfpack_di_get_determinant

10.730.2.6 #define umfpack_dl_get_lunz umfpack_di_get_lunz

10.730.2.7 #define umfpack_dl_get_numeric umfpack_di_get_numeric

10.730.2.8 #define umfpack_dl_load_numeric umfpack_di_load_numeric

10.730.2.9 #define umfpack_dl_numeric umfpack_di_numeric

10.730.2.10 #define umfpack_dl_report_info umfpack_di_report_info

10.730.2.11 #define umfpack_dl_save_numeric umfpack_di_save_numeric

10.730.2.12 #define umfpack_dl_solve umfpack_di_solve

10.730.2.13 #define umfpack_dl_symbolic umfpack_di_symbolic

10.730.2.14 #define umfpack_zl_defaults umfpack_zi_defaults

10.730.2.15 #define umfpack_zl_free_numeric umfpack_zi_free_numeric

10.730.2.16 #define umfpack_zl_free_symbolic umfpack_zi_free_symbolic

10.730.2.17 #define umfpack_zl_get_determinant umfpack_zi_get_determinant

10.730.2.18 #define umfpack_zl_get_lunz umfpack_zi_get_lunz

10.730.2.19 #define umfpack_zl_get_numeric umfpack_zi_get_numeric

10.730.2.20 #define umfpack_zl_load_numeric umfpack_zi_load_numeric

10.730.2.21 #define umfpack_zl_numeric umfpack_zi_numeric

10.730.2.22 #define umfpack_zl_report_info umfpack_zi_report_info

10.730.2.23 #define umfpack_zl_save_numeric umfpack_zi_save_numeric

10.730.2.24 #define umfpack_zl_solve umfpack_zi_solve

10.730.2.25 #define umfpack_zl_symbolic umfpack_zi_symbolic

10.730.3 Typedef Documentation

10.730.3.1 typedef struct umfpack_control_solver umfpack_control_solver

10.730.4 Function Documentation

10.730.4.1 static int umfpack_control_clear (void * *solver*) [static]

Parameters

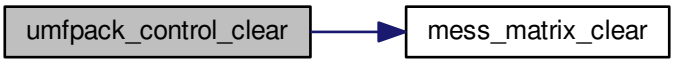
in,out	*solver*	pointer to the data object

Returns

always zero

The umfpack_control_clear function clears an UMFPACK object.

Here is the call graph for this function:



10.730.4.2 static int umfpack_control_det (void * *data*, double * *m*, double * *e*) [static]

Parameters

in	*data*	input pointer to the internal data
out	*m*	output mantissa of determinant
out	*e*	output exponent of determinant

Returns

zero on success or a non-zero error value otherwise

The umfpack_control_det function computes $\det(A)$ using UMFPACK.

10.730.4.3 static int umfpack_control_detc (void * *data*, double * *mr*, double * *mi*, double * *e*) [static]

Parameters

in	*data*	input pointer to the internal data
out	*mr*	output mantissa of real part of determinant
out	*mi*	output mantissa of imaginary part of determinant
out	*e*	output exponent of determinant

Returns

zero on success or a non-zero error value otherwise

The umfpack_control_detc function computes $\det(A)$ using UMFPACK.

10.730.4.4 static int umfpack_control_getL (void * *data*, mess_matrix *L*) [static]

Parameters

in	*data*	input pointer to the data object
in,out	*L*	input/output the L factor

Returns

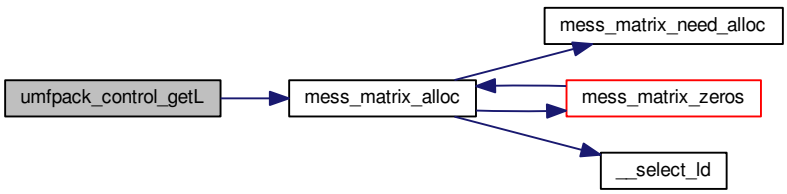
zero on success or a non-zero error value otherwise

The umfpack_control_getL function gets the L factor of a matrix decomposed with UMFPACK. L is in CSR format.

Attention

Internal use only.

Here is the call graph for this function:



10.730.4.5 static int umfpack_control_getpermp (void * *data*, mess_int_t * *p*) [static]

Parameters

in	*data*	input pointer to the data object
in,out	*p*	output permutation

Returns

zero on success or a non-zero error value otherwise

The umfpack_control_getpermp function gets the row perumation P for $PRAQ = LU$.

Attention

Internal use only.

10.730.4.6 static int umfpack_control_getpermq (void * *data*, mess_int_t * *q*) [static]

Parameters

in	*data*	input pointer to the data object
in,out	*q*	output permuataion

Returns

zero on success or a non-zero error value otherwise

The umfpack_control_getpermq function gets the colume permutation Q for $PRAQ = LU$.

Attention

Internal use only.

10.730.4.7 static int umfpack_control_getscalerow (void * *data*, mess_vector *r*) [static]

Parameters

in	*data*	input pointer to the data object
in,out	*r*	output permuataion

Returns

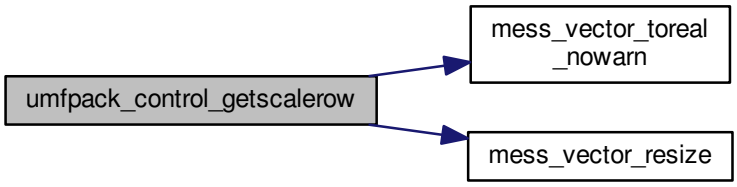
zero on success or a non-zero error value otherwise

The umfpack_control_getscalerow function gets the row scaling diagonal entries of R for $PRAQ = LU$.

Attention

Internal use only.

Here is the call graph for this function:



10.730.4.8 static int umfpack_control_getU (void * *data*, mess_matrix *U*) [static]

Parameters

in	*data*	input pointer to the data object
out	*U*	output U matrix

Returns

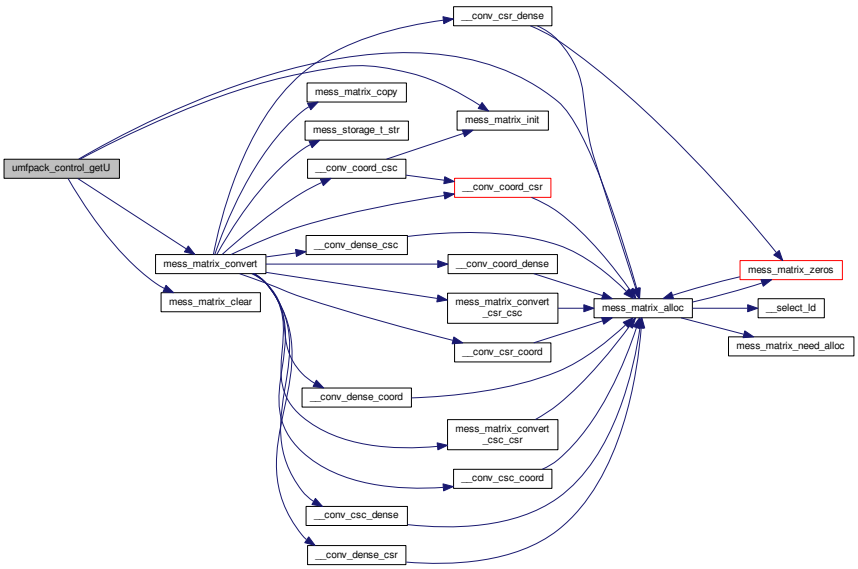
zero on success or a non-zero error value otherwise

The umfpack_control_getU function gets the U factor of a matrix decomposed with UMFPACK. U is in CSR format.

Attention

Internal use only.

Here is the call graph for this function:



10.730.4.9 static int umfpack_control_inverse (void * *data*, mess_matrix *inv*) [static]

Parameters

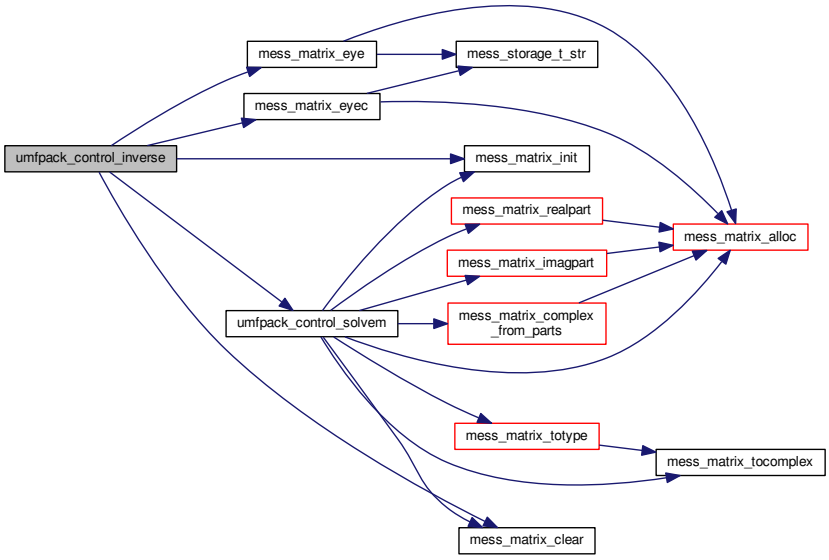
in	*data*	pointer to the internal data.
in,out	*inv*	output inverse

Returns

zero on success or a non zero error value otherwise

The umfpack_control_inverse function computes A^{-1}.

Here is the call graph for this function:



10.730.4.10 static int umfpack_control_solve (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

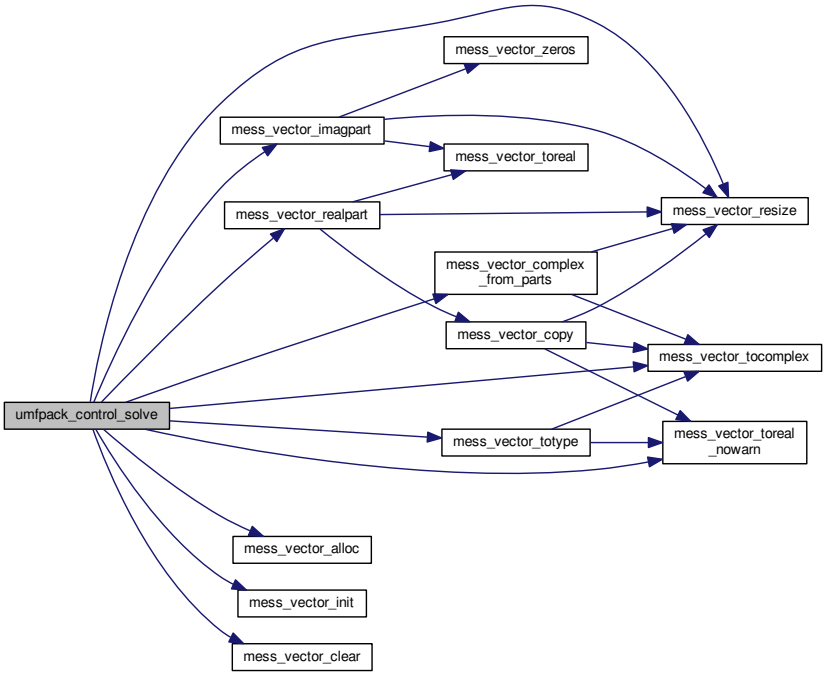
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The umfpack_control_solve function solves $Ax = b$.

Here is the call graph for this function:



10.730.4.11 static int umfpack_control_solveh (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

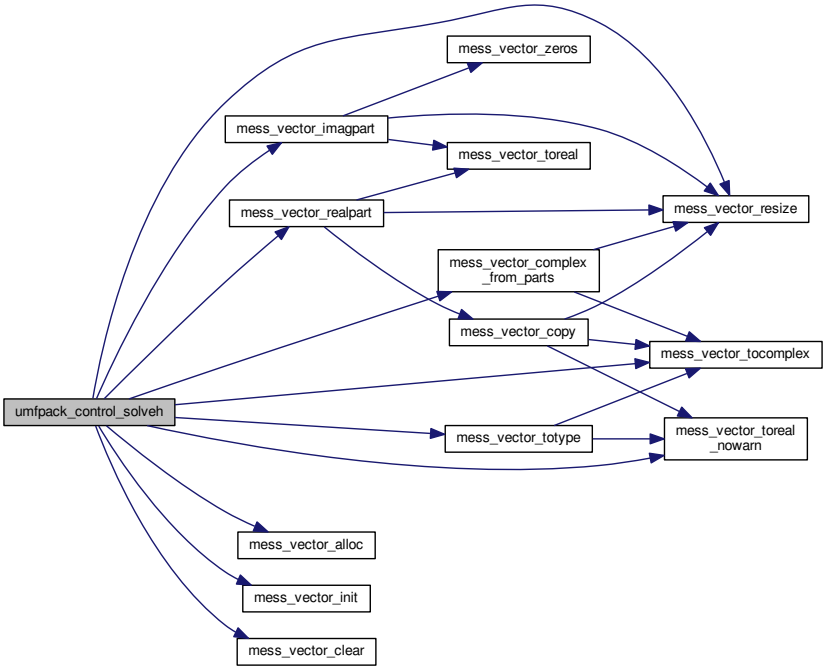
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The umfpack_control_solveh function solves $A^H x = b$.

Here is the call graph for this function:



10.730.4.12 static int umfpack_control_solvem (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

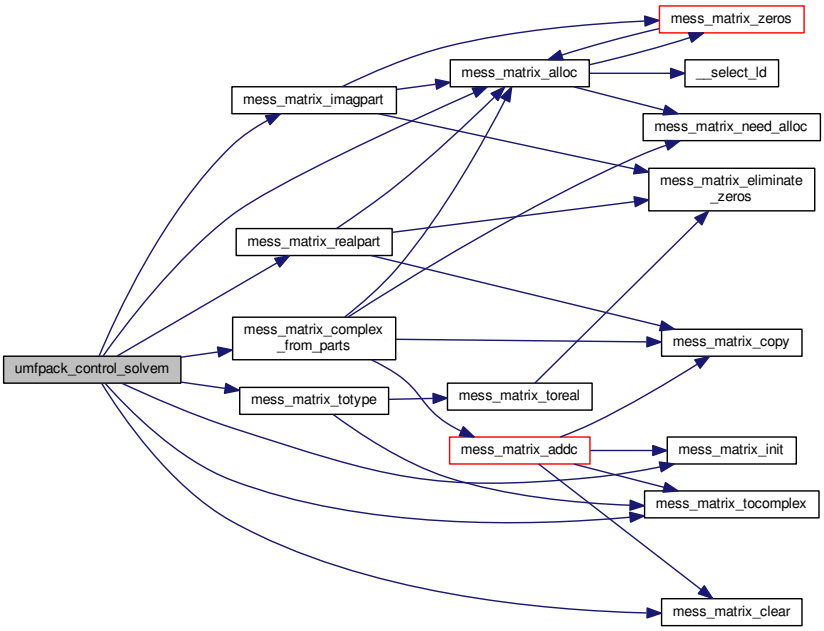
in	*data*	pointer to the data object
in	*b*	right hand side B
in,out	*x*	solution X

Returns

zero on success or a non zero error value otherwise

The umfpack_control_solvem function solves $AX = B$ with B and X matrices.

Here is the call graph for this function:



10.730.4.13 static int umfpack_control_solvemh (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

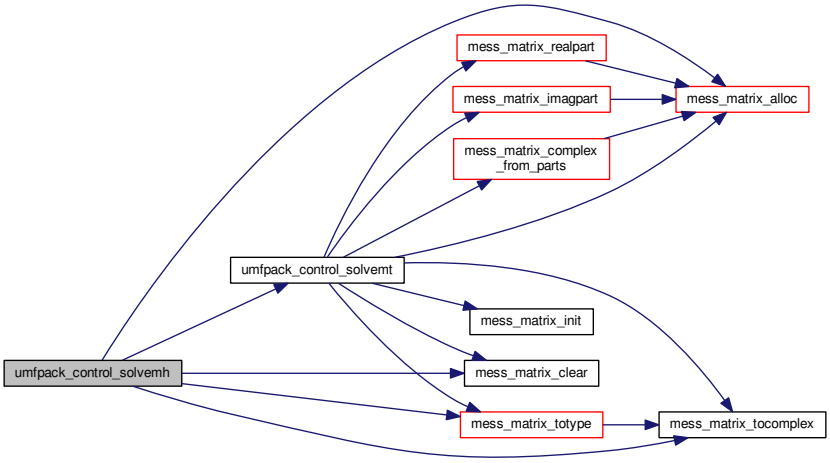
in	*data*	pointer to the data object
in	*b*	right hand side B
in,out	*x*	solution X

Returns

zero on success or a non zero error value otherwise

The umfpack_control_solvemh function solves $A^H X = B$ with B and X matrices.

Here is the call graph for this function:



10.730.4.14 static int umfpack_control_solvemt (void * *data*, mess_matrix *b*, mess_matrix *x*) [static]

Parameters

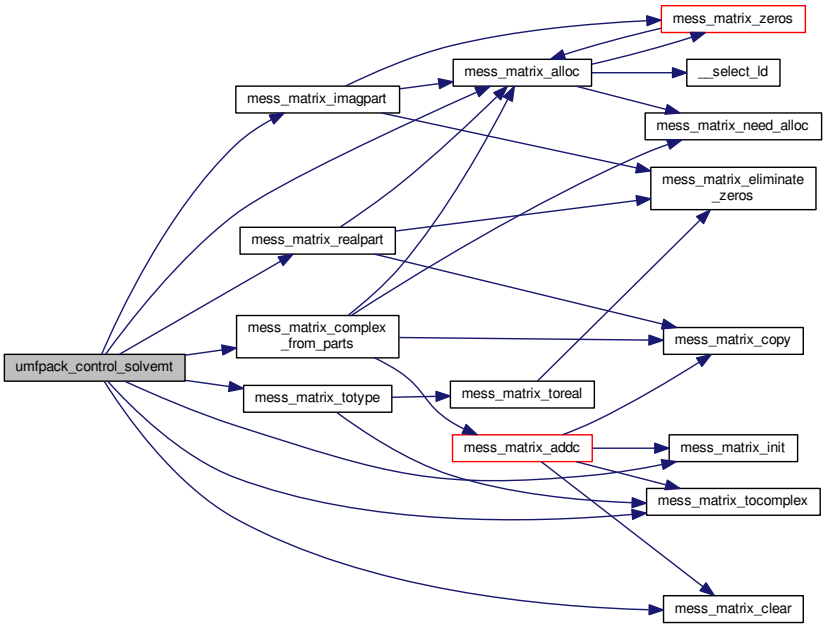
in	*data*	pointer to the data object
in	*b*	right hand side B
in,out	*x*	solution X

Returns

zero on success or a non zero error value otherwise

The umfpack_control_solvemt function solves $A^T X = B$ with B and X matrices.

Here is the call graph for this function:



10.730.4.15 static int umfpack_control_solvet (void * *data*, mess_vector *b*, mess_vector *x*) [static]

Parameters

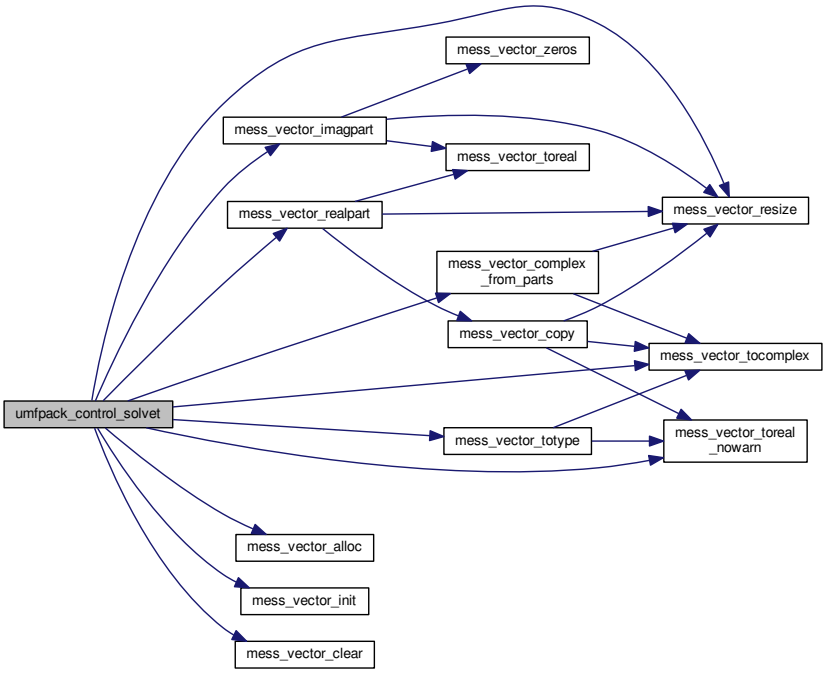
in	*data*	pointer to the data object
in	*b*	right hand side
in,out	*x*	solution

Returns

zero on success or a non zero error value otherwise

The umfpack_control_solvet function solves $A^T x = b$.

Here is the call graph for this function:



10.731 vector.c File Reference

Macros

- #define NPY_NO_DEPRECATED_API NPY_1_7_API_VERSION
- #define PY_ARRAY_UNIQUE_SYMBOL MESS_VECTOR_MATRIX_PYTHON_C_API
- #define NO_IMPORT_ARRAY
- #define ConvStringtoC(X) PyBytes_AsString(X)
- #define NPY_ARRAY_OWNDATA NPY_OWNDATA
- #define PyArray_ENABLEFLAGS(arr, flag) ((PyArrayObject∗)(arr))->flags |= (flag)
- #define PyArray_CALL_ISSIGNED(x) PyArray_ISSIGNED((PyArrayObject∗)(x))
- #define PyArray_CALL_ISUNSIGNED(x) PyArray_ISUNSIGNED((PyArrayObject∗)(x))
- #define PyArray_CALL_DATA(x) PyArray_DATA((PyArrayObject∗)(x))

Functions

- mess_vector vector_to_c (PyObject ∗data)

  Converts `Python` *vector to mess_vector.*
- PyObject ∗ vector_to_python (mess_vector c_vector, int copy)

  Converts mess_vector to `Python` *vector.*

10.731.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Nitin Prasad (nitin.prasad93@gmail.com, MPI Magdeburg, CSC)
Maximilian Behr (behr@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.731.2 Macro Definition Documentation

10.731.2.1 #define ConvStringtoC(X) PyBytes_AsString(X)

10.731.2.2 #define NO_IMPORT_ARRAY

10.731.2.3 #define NPY_ARRAY_OWNDATA NPY_OWNDATA

10.731.2.4 #define NPY_NO_DEPRECATED_API NPY_1_7_API_VERSION

10.731.2.5 #define PY_ARRAY_UNIQUE_SYMBOL MESS_VECTOR_MATRIX_PYTHON_C_API

10.731.2.6 #define PyArray_CALL_DATA(x) PyArray_DATA((PyArrayObject∗)(x))

10.731.2.7 #define PyArray_CALL_ISSIGNED(x) PyArray_ISSIGNED((PyArrayObject∗)(x))

10.731.2.8 #define PyArray_CALL_ISUNSIGNED(x) PyArray_ISUNSIGNED((PyArrayObject∗)(x))

10.731.2.9 #define PyArray_ENABLEFLAGS(*arr, flag*) ((PyArrayObject∗)(arr))->flags |= (flag)

10.732 vector.h File Reference

Data Structures

- struct mess_vector_st

  Representation of a real or complex vector.

Macros

- #define mess_vector_scalee(val, vec) if (MESS_IS_REAL(vec)) { mess_vector_scale((val),(vec)); } else { mess_vector_scalec((val), (vec));}

  Macro to scale a vector.

Typedefs

- typedef struct mess_vector_st mess_vector_st

  Representation of a real or complex vector.
- typedef mess_vector_st ∗ mess_vector

  Type definition of a real or complex vector.

Functions

- int mess_vector_init (mess_vector *vect)

  Initialize the mess_vector structure.
- int mess_vector_alloc (mess_vector vect, mess_int_t dim, mess_datatype_t dtype)

  Allocate a MESS_REAL or MESS_COMPLEX the mess_vector structure.
- int mess_vector_reset (mess_vector vect)

  Resets a mess_vector object.
- int mess_vector_clear (mess_vector *vect)

  Clean up the mess_vector structure to use it again.
- int mess_vector_copy (mess_vector x, mess_vector y)

  Copy mess_vector structure x to a mess_vector y structure.
- int mess_vector_copy_tocomplex (mess_vector in, mess_vector out)

  Copy a mess_vector strucutre into a complex mess_vector structure.
- int mess_vector_tocomplex (mess_vector v)

  Convert a vector to complex one.
- int mess_vector_toreal (mess_vector v)

  Convert a vector to real one.
- int mess_vector_toreal_nowarn (mess_vector v)

  Convert a vector to real values (without any warnings).
- int mess_vector_totype (mess_vector v, mess_datatype_t dt)

  Convert a vector to the given datatype.
- int mess_vector_from_farray (mess_vector v, mess_int_t dim, double *vals, mess_double_cpx_t *vals_cpx)

  Create a mess_vector from a `Fortran` array.
- int mess_vector_from_lapack (mess_vector v, mess_int_t dim, double *vals_re, double *vals_im)

  Create a mess_vector from two `Fortran` array.
- int mess_vector_realpart (mess_vector in, mess_vector out)

  Get the real part of a vector.
- int mess_vector_imagpart (mess_vector in, mess_vector out)

  Get the imaginary part of a vector.
- int mess_vector_complex_from_parts (mess_vector xr, mess_vector xc, mess_vector x)

  Generate a complex vector from real and imaginary part.
- int mess_vector_print (mess_vector vect)

  Print out a mess_vector structure.
- int mess_vector_printshort (mess_vector vect)

  Print out a mess_vector structure in a short form.
- int mess_vector_write (const char *filename, mess_vector vect)

  Write a vector to a matrix market vector file.
- int mess_vector_read (const char *filename, mess_vector vect)

  Read a vector from a matrix market vector file.
- int mess_vector_printinfo (mess_vector v)

  Print dimension and datatype of a mess_vector structure on stdout.
- int mess_vector_resize (mess_vector x, mess_int_t dim)

  Resize a vector.
- int mess_vector_zeros (mess_vector v)

  Fill a vector with zeros.
- int mess_vector_ones (mess_vector v)

  Fill a vector with ones.
- int mess_vector_rand (mess_vector v)

  Fill a vector with random numbers.
- int mess_vector_get (mess_vector v, mess_int_t i, mess_double_cpx_t *val)

Get entry i of a mess_vector instance.

- int mess_vector_rand_init (mess_int_t *seed)

  Init random number generator with given seed.
- int mess_vector_iperm (mess_vector in, mess_int_t *iperm, mess_vector out)

  Inverse permutation of a vector.
- int mess_vector_perm (mess_vector in, mess_int_t *perm, mess_vector out)

  Permute of a vector.
- int mess_vector_perm_inplace (mess_vector v, mess_int_t *p)

  Permute a vector inplace.
- int mess_vector_iperm_inplace (mess_vector v, mess_int_t *q)

  Permute a vector inplace.
- int mess_vector_iperm_split (mess_vector in, mess_int_t *iperm, mess_vector reout, mess_vector imout)

  Inverse permutation of a vector and spilts real and imaginary part.
- int mess_vector_iperm_combine (mess_vector rein, mess_vector imin, mess_int_t *iperm, mess_vector out)

  Inverse permutation of a vector and spilts real and imaginary part.
- int mess_vector_perm_split (mess_vector in, mess_int_t *perm, mess_vector reout, mess_vector imout)

  Permutate a vector and split in real and imaginary part.
- int mess_vector_perm_combine (mess_vector rein, mess_vector imin, mess_int_t *perm, mess_vector out)

  Inverse permutation of a vector and combination of real and imaginary part.
- double * mess_vector_valuereal (mess_vector vec, mess_int_t index)

  Get a real value from a vector.
- mess_double_cpx_t * mess_vector_valuecomplex (mess_vector vec, mess_int_t index)

  Get a complex value from a vector.
- int mess_vector_sort (mess_vector vect)

  Sort the components of a vector in ascending order.
- int mess_vector_sort_realpart (mess_vector vect)

  Sort the components of a vector by real part in ascending order.
- int mess_vector_sort_imagpart (mess_vector vect)

  Sort the components of a vector by imaginary part in ascending order.
- int mess_vector_logspace10 (mess_vector vect, double a, double b, mess_int_t nsample)

  Create a $\log_{10}$ space vector.
- int mess_vector_logspacee (mess_vector vect, double a, double b, mess_int_t nsample)

  Create a $\log_{\mathrm{e}}$ space.
- int mess_vector_logspace2 (mess_vector vect, double a, double b, mess_int_t nsample)

  Create a $\log_2$ space.
- int mess_vector_linspace (mess_vector vect, double a, double b, mess_int_t nsample)

  Create a linspace vector.
- int mess_vector_cat (mess_vector x1, mess_vector x2, mess_vector x)

  Concatenate two vectors.
- int mess_vector_split (mess_vector input, mess_int_t n, mess_vector x1, mess_vector x2)

  Split a vector after the n-th entry.
- int mess_vector_lift (mess_vector in, mess_int_t n, mess_vector out)

  Add n zeros to vector.
- int mess_vector_kron (mess_vector in1, mess_vector in2, mess_vector out)

  Compute Kronecker product for two vectors.
- int mess_vector_filter_stable (mess_vector in)

  Filter values with positive realpart from vector.
- int mess_vector_filter (mess_vector in, int(*filter_real)(const double *), int(*filter_complex)(const mess_double_cpx_t *))

  Filter values from vector.
- int mess_vector_convert_if_real (mess_vector v)

Convert complex vector to real vector if imaginary part of every entry is small.

- int mess_vector_map (mess_vector v, double(∗f_real)(double), mess_double_cpx_t(∗f_cpx)(mess_double↩_cpx_t))

  Map a scalar function to a vector.
- int mess_vector_map_abs (mess_vector)
- int mess_vector_map_acos (mess_vector)
- int mess_vector_map_acosh (mess_vector)
- int mess_vector_map_arg (mess_vector)
- int mess_vector_map_asin (mess_vector)
- int mess_vector_map_asinh (mess_vector)
- int mess_vector_map_atan (mess_vector)
- int mess_vector_map_atanh (mess_vector)
- int mess_vector_map_ceil (mess_vector)
- int mess_vector_map_conj (mess_vector)
- int mess_vector_map_cos (mess_vector)
- int mess_vector_map_cosh (mess_vector)
- int mess_vector_map_exp (mess_vector)
- int mess_vector_map_expm1 (mess_vector)
- int mess_vector_map_floor (mess_vector)
- int mess_vector_map_log (mess_vector)
- int mess_vector_map_not (mess_vector)
- int mess_vector_map_round (mess_vector)
- int mess_vector_map_sin (mess_vector)
- int mess_vector_map_sinh (mess_vector)
- int mess_vector_map_sqrt (mess_vector)
- int mess_vector_map_tan (mess_vector)
- int mess_vector_map_tanh (mess_vector)
- int mess_vector_map_isfinite (mess_vector)
- int mess_vector_map_isinf (mess_vector)
- int mess_vector_map_isnan (mess_vector)
- int mess_vector_any (mess_vector mat, mess_int_t(∗f_real)(double), mess_int_t(∗f_cpx)(mess_double_↩cpx_t), mess_int_t ∗anyval)

  Checks a mess_vector if a given predicate is true.
- mess_int_t mess_vector_memsize (mess_vector v)

  Return the size of a mess_vector in bytes.
- int mess_vector_axpy (double a, mess_vector x, mess_vector y)

  Compute $y \leftarrow ax + y$ (real).
- int mess_vector_norm2 (mess_vector x, double ∗nrm)

  Compute the 2-norm of a vector.
- int mess_vector_norm1 (mess_vector x, double ∗nrm)

  Compute the 1-norm of a vector.
- int mess_vector_norminf (mess_vector x, double ∗nrm)

  Compute the ∞-norm of a vector.
- int mess_vector_norm (mess_vector x, mess_norm_t nrm_t, double ∗nrm)

  Compute the norm of a vector.
- int mess_vector_scale (double alpha, mess_vector x)

  Scale a vector with a real scalar.
- int mess_vector_dot (mess_vector x, mess_vector y, double ∗dot)

  Compute the dot product of two vectors.
- int mess_vector_minvalue (mess_vector v, double ∗min)

  Get the minimal value in a vector.
- int mess_vector_maxvalue (mess_vector v, double ∗max)

  Get the maximum value in a vector.

- int mess_vector_scalec (mess_double_cpx_t alpha, mess_vector x)
  Scale a vector with a complex scalar.
- int mess_vector_dotc (mess_vector x, mess_vector y, mess_double_cpx_t *dot)
  Compute the dot product of two vectors (complex).
- int mess_vector_axpyc (mess_double_cpx_t a, mess_vector x, mess_vector y)
  Compute $y \leftarrow ax + y$ *(complex).*
- int mess_vector_dotu (mess_vector x, mess_vector y, mess_double_cpx_t *dot)
  Compute the dot product of two vectors (complex).
- int mess_vector_diffnorm (mess_vector x1, mess_vector x2, double *diff)
  Compute $\|x_1 - x_2\|_2$.
- int mess_vector_diffnorminf (mess_vector x1, mess_vector x2, double *diff)
  Compute $\|x_1 - x_2\|_\infty$.
- int mess_vector_max (mess_vector v, double *maxval, mess_int_t *maxind)
  Return the largest entry of a vector.
- int mess_vector_conj (mess_vector vector)
  Complex conjugatation of a vector.

10.732.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.733 vector_print.c File Reference

Functions

- int mess_vector_print (mess_vector vect)
  Print out a mess_vector structure.
- int mess_vector_printshort (mess_vector vect)
  Print out a mess_vector structure in a short form.
- int mess_vector_printinfo (mess_vector v)
  Print dimension and datatype of a mess_vector structure on stdout.

10.733.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.734 vector_read.c File Reference

Macros

- #define SET_LAST(array, len, value) (array)[(len)-1] = (value)
- #define LINE_LENGTH 1025
- #define TOKEN_LENGTH 30
- #define MM_BANNER "%%MatrixMarket"
- #define MM_STR_VECTOR "vector"
- #define MM_STR_REAL "real"
- #define MM_STR_INT "integer"
- #define MM_STR_CPX "complex"

Functions

- static void _lowercase (char *str)

  Convert a string to lower case.
- static int __mm_read_vector_info (csc_io_file_t *file, mess_vector vect)

  Reads the header of a Matrix Market vector file. (internal version)
- int mess_vector_read (const char *filename, mess_vector vect)

  Read a vector from a matrix market vector file.
- int mess_vector_write (const char *filename, mess_vector vect)

  Write a vector to a matrix market vector file.
- int mess_vector_mwrite (const char *filename, const char *name, mess_vector v)

  Write a vector into a `MATLAB` *-mfile.*

10.734.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.734.2 Macro Definition Documentation

10.734.2.1 #define LINE_LENGTH 1025

10.734.2.2 #define MM_BANNER "%%MatrixMarket"

10.734.2.3 #define MM_STR_CPX "complex"

10.734.2.4 #define MM_STR_INT "integer"

10.734.2.5 #define MM_STR_REAL "real"

10.734.2.6 #define MM_STR_VECTOR "vector"

10.734.2.7 #define SET_LAST(*array, len, value*) (array)[(len)-1] = (value)

10.734.2.8 #define TOKEN_LENGTH 30

10.734.3 Function Documentation

10.734.3.1 static int __mm_read_vector_info (csc_io_file_t * *file,* mess_vector *vect*) [static]

Parameters

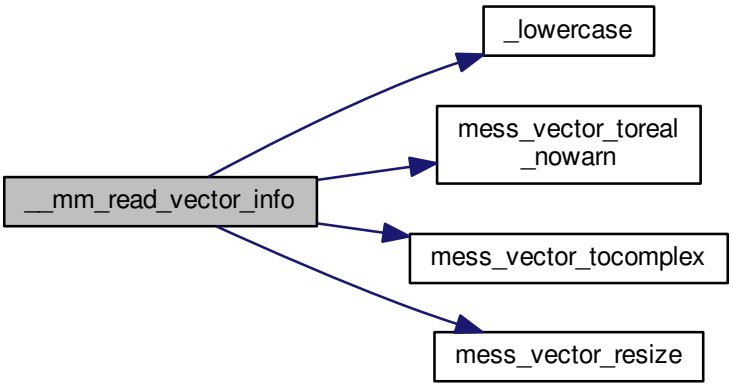
in	*file*	input filepointer
out	*vect*	mess_vector to store the read information

Returns

zero on success or otherwise a non zero error code.

The __mm_read_vector_info function reads the header of a matrix market vector file and writes the basic information to matrix_vect.

Here is the call graph for this function:



10.734.3.2 static void _lowercase (char * *str*) [static]

Parameters

in,out	*str*	String to be converted to lower case

Convert a string to lowercase.

10.734.3.3 int mess_vector_mwrite (const char * *filename*, const char * *name*, mess_vector *v*)

Parameters

in	*filename*	input file to write
in	*name*	input name of variable in mfile
in	*v*	input vector to write to the mfile

Returns

zero on success or a non zero error code otherwise

The mess_vector_mwrite function writes a vector as a MATLAB column vector in to a file.
It could be loaded into MATLAB by calling the script.

10.735 VerifyFortran.h File Reference

Macros

- #define FortranCInterface_GLOBAL(name, NAME) name##_
- #define FortranCInterface_GLOBAL_(name, NAME) name##_
- #define FortranCInterface_MODULE(mod_name, name, mod_NAME, NAME) __##mod_name##_MOD_↩ ##name
- #define FortranCInterface_MODULE_(mod_name, name, mod_NAME, NAME) __##mod_name##_MOD_↩ ##name
- #define VerifyFortran FortranCInterface_GLOBAL(verifyfortran, VERIFYFORTRAN)

10.735.1 Macro Definition Documentation

10.735.1.1 #define FortranCInterface_GLOBAL(*name, NAME*) name##_

10.735.1.2 #define FortranCInterface_GLOBAL_(*name, NAME*) name##_

10.735.1.3 #define FortranCInterface_MODULE(*mod_name, name, mod_NAME, NAME*) __##mod_name##_MOD_##name

10.735.1.4 #define FortranCInterface_MODULE_(*mod_name, name, mod_NAME, NAME*) __##mod_name##_MOD_##name

10.735.1.5 #define VerifyFortran FortranCInterface_GLOBAL(verifyfortran, VERIFYFORTRAN)

10.736 wachspress.c File Reference

Functions

- static double min (double x, double y)
- static double dn (double u, double k, double TOL)
- static double ellipK (double k, double TOL)
- static double F (double phi, double k, double TOL)
- int mess_lrcfadi_para_wachspress (double a, double b, double alpha, double tol, double *p, mess_int_t *lp)

  Compute Wachspress parameters.

10.736.1 Detailed Description

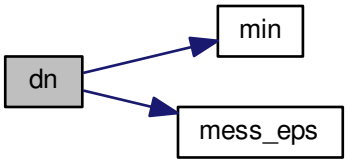
Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)
Jens Saak (saak@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.736.2 Function Documentation

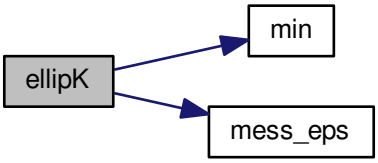
10.736.2.1 static double dn (double *u*, double *k*, double *TOL*) [static]

Here is the call graph for this function:



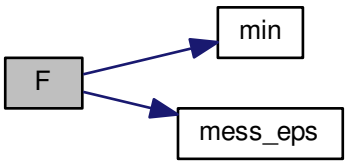
10.736.2.2 static double ellipK (double *k*, double *TOL*) [static]

Here is the call graph for this function:



10.736.2.3 static double F (double *phi*, double *k*, double *TOL*) [static]

Here is the call graph for this function:



10.736.2.4 static double min (double *x*, double *y*) [static]

10.737 write.c File Reference

Macros

- #define CHECK_NULL(ret) if ((ret) == 0) { MSG_ERROR("Writing to file failed.\n"); csc_io_close(f); return MESS_ERROR_FILEIO; }

Functions

- int mess_matrix_write (const char *filename, mess_matrix matrix)

  Write a mess_matrix to a `MatrixMarket` *file.*

10.737.1 Detailed Description

Author

Martin Köhler (`koehlerm@mpi-magdeburg.mpg.de`, MPI Magdeburg, CSC)

10.737.2 Macro Definition Documentation

10.737.2.1 #define CHECK_NULL(*ret*) if ((ret) == 0) { MSG_ERROR("Writing to file failed.\n"); csc_io_close(f); return MESS_ERROR_FILEIO; }

10.738 x11init.c File Reference

Data Structures

- struct _scalepoints

Typedefs

- typedef struct _scalepoints ScalePoints

Functions

- static double map_to_win (double a, double b, double c, double d, double x)

  Map a point x from an interval $[c, d]$ to a point in $[a, b]$.
- static int drawaxis (mess_plot win, double xmin, double xmax, double ymin, double ymax, ScalePoints *X↩ Scale, int XP, ScalePoints *YScale, int YP, int axis_pos)

  Draw x and y axis on a plot.
- static void SetColor (mess_plot win, char *colorname)

  Set drawing color.
- static void plot (mess_plot p)

  Draw a complete plot.
- static void * __plotter (void *arg)

  Mainloop of the plot-window thread.
- int mess_plot_create (mess_plot *pl, mess_int_t width, mess_int_t height, char *title, int xscale, int yscale, int data)

  Create a plotting window.
- int mess_plot_addData (mess_plot p, int data, double x, double y)

  Add new data values to a plot.
- int mess_plot_clearData (mess_plot p, int data)

  Clear all data of a data element in a plot.
- int mess_plot_setLabel (mess_plot p, int data, char *label)

  Set label of a data element.
- int mess_plot_setColor (mess_plot p, int data, char *color)

  Set color of a data element.
- int mess_plot_update (mess_plot p)

  Update / redraw a plot.
- static int save_as_xpm_file (Display *display, int screen, XWindowAttributes *win_info, Pixmap pixmap, unsigned int width, unsigned int height, unsigned int depth, char *filename)

  Internal helper function to save a plot as xpm.
- int mess_plot_save (mess_plot p, char *filename)

  Save a plot to a xpm image.
- int mess_plot_close (mess_plot *p)

  Close a plotting window and clean up all internal data.

10.738.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

This file provides a small and easy plotting interface on top of the X11 library.

10.738.2 Typedef Documentation

10.738.2.1 typedef struct _scalepoints ScalePoints

Displaypoints for the X and Y axis

10.738.3 Function Documentation

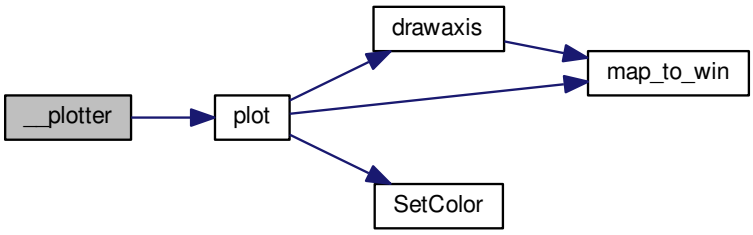
10.738.3.1 static void* __plotter (void * *arg*) [static]

Parameters

in	*arg*	input plot to work on

The __plotter function is the mainloop and the event handler of the plotting window.

Here is the call graph for this function:



10.738.3.2 static int drawaxis (mess_plot *win*, double *xmin*, double *xmax*, double *ymin*, double *ymax*, ScalePoints * *XScale*, int *XP*, ScalePoints * *YScale*, int *YP*, int *axis_pos*) [static]

Parameters

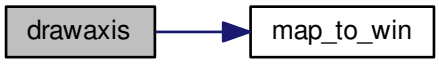
in,out	*win*	window to draw on
in	*xmin*	input minimal value on x axis
in	*xmax*	input maximal value on x axis
in	*ymin*	input minimal value on y axis
in	*ymax*	input maximal value on y axis
in	*XScale*	input points and labels on x axis
in	*XP*	input number of labels on x axis
in	*YScale*	input points and labels on y axis
in	*YP*	input number of labels on y axis
in	*axis_pos*	input position of axis (0 if they should go through (0,0) or 1 if they should be on the left and bottom)

Returns

zero on success or a non-zero error value otherwise

The drawaxis function draws the axis on a plot.

Here is the call graph for this function:



10.738.3.3 static double map_to_win (double *a*, double *b*, double *c*, double *d*, double *x*) [static]

Parameters

in	*a*	input lower bound of destination interval
in	*b*	input upper bound of destination interval
in	*c*	input lower bound of source interval
in	*d*	input upper bound of source interval
in	*x*	input point to map in $[a, b]$

Returns

mapped point in a, b

The map_to_win function computes the linear mapping of a point x from an interval $[c, d]$ to a point in the interval $[a, b]$ and prints out the mapped point.

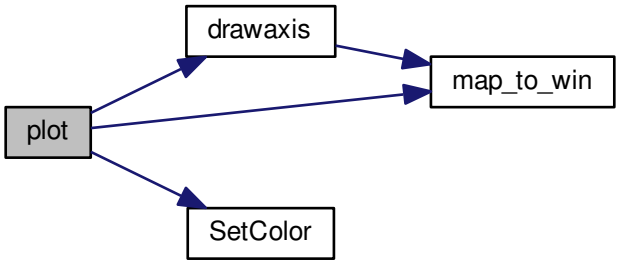
10.738.3.4 static void plot (mess_plot *p*) [static]

Parameters

in,out	*p*	plot to draw

The Plotting Interface function is the internal main function of the plotter.

Here is the call graph for this function:



10.738.3.5 static int save_as_xpm_file (Display * *display*, int *screen*, XWindowAttributes * *win_info*, Pixmap *pixmap*, unsigned int *width*, unsigned int *height*, unsigned int *depth*, char * *filename*) [static]

Parameters

in	*display*	input pointer to the display
in	*screen*	input screennumber (unused)
in	*win_info*	input Window information
in	*pixmap*	input pixmap to save
in	*width*	input width of image
in	*height*	input height of image
in	*depth*	input color depth of image (unused)
in	*filename*	input filename of the output image

Returns

zero on success input or a non-zero error value otherwise

The save_as_xpm_file function saves a given pixmap to a file.
It is only a internal helper function.

10.738.3.6 static void SetColor (mess_plot *win*, char * *colorname*) [static]

Parameters

in,out	*win*	plotting window
in	*colorname*	input name of the color (X11 color names)

The SetColor function sets the foreground color in a plot.
Allowed colornames are all that are used by X11.

10.739 xerbla.c File Reference

Functions

- void xerbla_ (char *fname, mess_int_t *info)

  Override LAPACK *xerbla function.*

10.739.1 Detailed Description

Author

Martin Köhler (koehlerm@mpi-magdeburg.mpg.de, MPI Magdeburg, CSC)

10.739.2 Function Documentation

10.739.2.1 void xerbla_ (char * *fname*, mess_int_t * *info*)

Parameters

in	*fname*	input function name
in	*info*	input info code

The xerbla_ function overrides LAPACK xerbla function.

References

[1] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*, volume 6 of *Adv. Des. Control*. SIAM Publications, Philadelphia, PA, 2005. 374, 415

[2] E. Bänsch, P. Benner, J. Saak, and H. K. Weichelt. Optimal control-based feedback stabilization of multi-field flow problems. In G. Leugering, P. Benner, S. Engell, A. Griewank, H. Harbrecht, M. Hinze, R. Rannacher, S. Ulbrich, G. Leugering, P. Benner, S. Engell, A. Griewank, H. Harbrecht, M. Hinze, R. Rannacher, and S. Ulbrich, editors, *Trends in PDE Constrained Optimization*, volume 165 of *Internat. Ser. Numer. Math. 165*, pages 173–188. Birkhäuser, 2014. 508, 518, 1151

[3] E. Bänsch, P. Benner, J. Saak, and H. K. Weichelt. Riccati-based boundary feedback stabilization of incompressible Navier-Stokes flows. *SIAM J. Sci. Comput.*, 37(2):A832–A858, 2015. 508, 518

[4] S. Barrachina, P. Benner, and E. S. Quintana-Ortí. Efficient algorithms for generalized algebraic Bernoulli equations based on the matrix sign function. *Numer. Algorithms*, 46(4):351–368, 2007. 561, 565

[5] P. Benner. Numerical solution of special algebraic Riccati equations via an exact line search method. In *Proc. European Control Conf. ECC 97* (CD-ROM), Paper 786. BELWARE Information Technology, Waterloo, Belgium, 1997. 557, 558

[6] P. Benner and R. Byers. An exact line search method for solving generalized continuous-time algebraic Riccati equations. *IEEE Trans. Autom. Control*, 43(1):101–107, 1998. 557, 558

[7] P. Benner, M. Heinkenschloss, J. Saak, and H. K. Weichelt. An inexact low-rank Newton-ADI method for large-scale algebraic Riccati equations. *Appl. Numer. Math.*, 108:125–142, October 2016. 493, 494, 495, 1606

[8] P. Benner, M. Köhler, and J. Saak. Sparse-dense Sylvester equations in H_2-model order reduction. Preprint MPIMD/11-11, Max Planck Institute Magdeburg, December 2011. 295, 1874

[9] P. Benner, M. Köhler, and J. Saak. Fast approximate solution of the non-symmetric generalized eigenvalue problem on multicore architectures. In Michael Bader, Arndt Bodeand Hans-Joachim Bungartz, Michael Gerndt, Gerhard R. Joubert, and Frans Peters, editors, *Parallel Computing: Accelerating Computational Science and Engineering (CSE)*, volume 25 of *Advances in Parallel Computing*, pages 143–152. IOS Press, 2014. 496

[10] P. Benner, P. Kürschner, and J. Saak. Avoiding complex arithmetic in the low-rank ADI method efficiently. *Proc. Appl. Math. Mech.*, 12:639–640, 2012. 487

[11] P. Benner and E. S. Quintana-Ortí. Solving stable generalized Lyapunov equations with the matrix sign function. *Numer. Algorithms*, 20(1):75–100, 1999. 563, 567

[12] P. Benner and J. Saak. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Preprint SPP 1253-090, DFG-SPP153, January 2010. Available from: http://www.am.uni-erlangen.de/home/spp1253/wiki/index.php/Preprints. 491

[13] T. A. Davis. *Direct Methods for Sparse Linear Systems*. Number 2 in Fundamentals of Algorithms. SIAM, Philadelphia, PA, USA, 2006. 92, 108, 254, 255, 315, 316, 596, 597, 1326

[14] J. D. Gardiner, A. J. Laub, J. J. Amato, and C. B. Moler. Solution of the Sylvester matrix equation $AXB + CXD = E$. *ACM Trans. Math. Software*, 18:223–231, 1992. 295, 1874

[15] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, fourth edition, 2013. 257

[16] S. Gugercin, A. C. Antoulas, and C. Beattie. $\mathcal{H}_2$ model reduction for large-scale linear dynamical systems. *SIAM J. Matrix Anal. Appl.*, 30(2):609–638, 2008. 395

[17] S. Gugercin, T. Stykel, and S. Wyatt. Model reduction of descriptor systems by interpolatory projection methods. *SIAM J. Sci. Comput.*, 35(5):B1010–B1033, 2013. 506, 516

[18] G. F. Jónsson and S. Vavasis. Solving polynomials with small leading coefficients. *SIAM J. Matrix Anal. Appl.*, 26(2):400–414, July 2006. 557, 558

[19] M. Köhler and J. Saak. Efficiency Improving Implementation Techniques for Large Scale Matrix Equation Solvers. Chemnitz Scientific Computing Prep. CSC 09-10, Technische Universität Chemnitz, 2009. 487

[20] M. Köhler and J. Saak. On BLAS level-3 implementations of common solvers for (quasi-) triangular generalized Lyapunov equations. SLICOT Working Note 1, NICONET e.V., 2014. 1435

[21] R. B. Lehoucq and D. C. Sorensen. Deflation techniques for an implicitly restarted Arnoldi iteration. *SIAM J. Matrix Anal. Appl.*, 17(4):789–821, 1996. 454, 456, 457, 458

[22] A. Meister. *Numerik linearer Gleichungssysteme. Eine Einführung in moderne Verfahren.* Vieweg+Teubner, Wiesbaden, 4th revised ed. edition, 2011. 335, 340, 341

[23] B. C. Moore. Principal component analysis in linear systems: controllability, observability, and model reduction. *IEEE Trans. Autom. Control*, AC–26(1):17–32, 1981. 1893

[24] T. Penzl. Numerical solution of generalized Lyapunov equations. *Adv. Comp. Math.*, 8:33–48, 1997. 1435

[25] T. Penzl. A cyclic low rank Smith method for large, sparse Lyapunov equations with applications in model reduction and optimal control. Technical Report SFB393/98-6, Fakultät für Mathematik, TU Chemnitz, 09107 Chemnitz, FRG, 1998. Available from `http://www.tu-chemnitz.de/sfb393/sfb98pr.html`. 495, 574, 575

[26] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press, Manchester, UK, 1992. 971

[27] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, USA, second edition, 2003. 31

[28] J. Saak. *Efficient Numerical Solution of Large Scale Algebraic Matrix Equations in PDE Control and Model Order Reduction*. Dissertation, Technische Universität Chemnitz, Chemnitz, Germany, July 2009. 377, 379, 381, 487, 491, 495, 571, 574, 575

[29] E. L. Wachspress. The ADI model problem, 1995. Available from the author. 496, 576

[30] Yuesheng Xu and Taishan Zeng. Optimal $\mathcal{H}_2$ model reduction for large scale MIMO systems via tangential interpolation. *Int. J. Numer. Anal. Model.*, 8(1):174–188, 2011. 395

Index

$my_sub
symbols/my_moduleSmy_sub.c, 1767
VerifyC/CMakeFiles/FortranCInterface/symbols/my↩
_moduleSmy_sub.c, 1767
$mysub
symbols/mymoduleSmysub.c, 1771
VerifyC/CMakeFiles/FortranCInterface/symbols/mymodule↩
Smysub.c, 1771
_GNU_SOURCE
openmp.c, 1788
_ParseIfmt
read.c, 1817
_ParseRfmt
read.c, 1817
__ONE
axpy.c, 978
copy.c, 1275
scale.c, 1842
vector/cat.c, 1020
vector/norm.c, 1786
__USE_POSIX199309
time.c, 1885
__axpy_rc
axpy.c, 978
__cmp_mess_int_t2_complex
sort.c, 1853
__cmp_mess_int_t2_double
sort.c, 1853
__cmp_mess_int_t_complex
setelement.c, 1844
sort.c, 1853
__cmp_mess_int_t_double
setelement.c, 1845
sort.c, 1854
__compare_complex
vector/perm.c, 1797
__compare_imagpart
vector/perm.c, 1797
__compare_real
vector/perm.c, 1798
__compare_realpart
vector/perm.c, 1798
__constructVW
H2 Model Order Reduction, 396
__constructVW_pre
H2 Model Order Reduction, 397
__constructVW_pthread
H2 Model Order Reduction, 398
__constructVW_sec
H2 Model Order Reduction, 399
__constructVW_seq
H2 Model Order Reduction, 400
__constructVWmat
H2 Model Order Reduction, 401
__conv_coord_csc
matrix/convert.c, 1268
__conv_coord_csr
matrix/convert.c, 1268
__conv_coord_dense
matrix/convert.c, 1269
__conv_csc_coord
matrix/convert.c, 1269
__conv_csc_dense
matrix/convert.c, 1270
__conv_csr_coord
matrix/convert.c, 1271
__conv_csr_dense
matrix/convert.c, 1271
__conv_dense_coord
matrix/convert.c, 1272
__conv_dense_csc
matrix/convert.c, 1273
__conv_dense_csr
matrix/convert.c, 1273
__drand
matrix/rand.c, 1808
vector/rand.c, 1809
__filter_linesearch_cpx
lrnm_linesearch.c, 1606
__filter_linesearch_real
lrnm_linesearch.c, 1606
__filter_stable_complex
filter.c, 1427
__filter_stable_real
filter.c, 1427
__gaxpy_csc_ge
csc.c, 1275
__gaxpy_csr_ge
csr.c, 1311
__gaxpy_csr_sym
csr.c, 1311
__gaxpy_dense
dense.c, 1322
__gaxpyh_csc_ge
csc.c, 1276
__gaxpyh_csr_ge
csr.c, 1312
__gaxpyh_dense
dense.c, 1323
__gaxpyt_csc_ge
csc.c, 1277
__gaxpyt_csr_ge
csr.c, 1312
__gaxpyt_dense
dense.c, 1324
__glyap_dae1, 764
A11, 765
A12, 765
A21, 765
A22, 765

A22sol, 765
Amsolver, 765
E11, 765
E11sol, 765
fullAsol, 766
fullA, 766
fullE, 766
n1, 766
n2, 766
__glyap_dae2, 766
A, 768
applyPIsolver, 768
delta, 768
fullAsolver, 768
fullMsolver, 768
fullA, 768
fullM, 768
G, 768
M, 769
MM, 769
num_solvers, 769
shiftsolvers, 769
__glyap_so1, 769
D, 771
K, 771
Ksolver, 771
lowerbound, 771
M, 771
Msolver, 771
shifts, 771
shiftsolver, 771
upperbound, 771
__glyap_so2, 772
D, 773
K, 773
Ksolver, 773
lowerbound, 773
M, 773
Msolver, 773
shifts, 773
shiftsolver, 773
upperbound, 773
__hb_read
read.c, 1813
__hb_read_data
read.c, 1813
__hb_read_info
read.c, 1813
__ilu0_data, 774
mat, 775
uptr, 775
__iluk, 775
D, 776
data_type, 776
Dx, 776
L, 776
n, 776
U, 776
__iluk_clear
iluk.c, 1462
__iluk_solve
iluk.c, 1462
__iluk_solve_complex
iluk.c, 1462
__lyap, 776
A, 777
Amsolver, 777
ApEX_tmp, 777
Asolver, 777
B, 778
E, 778
Esolver, 778
__lyap_AINV_apply
equation_lyap.c, 1405
__lyap_AINV_clear
equation_lyap.c, 1406
__lyap_AINV_generate
equation_lyap.c, 1406
__lyap_A_multiplym
equation_lyap.c, 1404
__lyap_ApEX_apply
equation_lyap.c, 1409
__lyap_ApEX_clear
equation_lyap.c, 1410
__lyap_ApEX_generate
equation_lyap.c, 1411
__lyap_ApE_clear
equation_lyap.c, 1407
__lyap_ApE_generate
equation_lyap.c, 1408
__lyap_ApE_solve
equation_lyap.c, 1408
__lyap_clear
equation_lyap.c, 1411
__matlab_equation, 778
data, 779
function_AINV_apply, 779
function_AINV_clear, 779
function_AINV_generate, 779
function_AX_apply, 779
function_AX_clear, 779
function_AX_generate, 779
function_ApEINV_apply, 779
function_ApEINV_clear, 779
function_ApEINV_generate, 779
function_ApEX_apply, 779
function_ApEX_clear, 779
function_ApEX_generate, 779
function_EINV_apply, 780
function_EINV_clear, 780
function_EINV_generate, 780
function_EX_apply, 780
function_EX_clear, 780
function_EX_generate, 780
function_parameter, 780
options, 780

__mess_calloc
Miscellaneous Helper Functions, 675
__mess_free
Miscellaneous Helper Functions, 676
__mess_lrcfadi_step, 780
it, 781
rel_change, 781
res2, 781
res2_change, 781
stop_user, 781
Z, 781
__mess_malloc
Miscellaneous Helper Functions, 676
__mess_matrix_add_bothsparse_complex_csc
add.c, 951
__mess_matrix_add_bothsparse_complex_csr
add.c, 952
__mess_matrix_add_bothsparse_complex_real_csc
add.c, 952
__mess_matrix_add_bothsparse_complex_real_csr
add.c, 953
__mess_matrix_add_bothsparse_csc
add.c, 954
__mess_matrix_add_bothsparse_csr
add.c, 955
__mess_matrix_add_bothsparse_real_complex_csc
add.c, 955
__mess_matrix_add_bothsparse_real_complex_csr
add.c, 956
__mess_matrix_alloc_setld
matrix/alloc.c, 960
__mess_matrix_cat_LABC
matrix/cat.c, 1016
__mess_matrix_cat_LACD
matrix/cat.c, 1017
__mess_matrix_cat_RABD
matrix/cat.c, 1018
__mess_matrix_cat_RBCD
matrix/cat.c, 1019
__mess_matrix_cat_antidiag
matrix/cat.c, 1012
__mess_matrix_cat_diag
matrix/cat.c, 1013
__mess_matrix_cat_firstcol
matrix/cat.c, 1013
__mess_matrix_cat_firstrow
matrix/cat.c, 1014
__mess_matrix_cat_full
matrix/cat.c, 1015
__mess_matrix_perm_csc
matrix/perm.c, 1793
__mess_matrix_perm_csr
matrix/perm.c, 1794
__mess_matrix_perm_dense
matrix/perm.c, 1794
__mess_plotExportData, 783
color, 784
label, 784
len, 784
type, 784
x, 784
y, 784
__mess_plotdata, 782
color, 782
label, 782
len, 782
x, 782
y, 783
__mess_plotter, 784
delete_atom, 786
display, 786
done, 786
editing, 786
gc, 786
height, 786
lock, 786
plot, 786
plot_len, 786
screen_num, 787
sig, 787
thread, 787
title, 787
width, 787
win, 787
xscale, 787
yscale, 787
__mess_plotterScript, 788
legendGnuPos, 789
legendTikzPos, 789
plot, 789
plot_len, 789
title, 789
xLabel, 789
xscale, 789
yLabel, 789
yscale, 789
__mess_precond_diag_clear
precond/diag.c, 1327
__mess_precond_diag_complexclear
precond/diag.c, 1328
__mess_precond_diag_solve
precond/diag.c, 1328
__mess_precond_diag_solvecomplex
precond/diag.c, 1329
__mess_precond_ilu0_clear
ilu0.c, 1460
__mess_precond_ilu0_solve
ilu0.c, 1460
__mess_precond_sor_clear
precond/sor.c, 1851
__mess_precond_sor_solve
precond/sor.c, 1851
__mess_precond_ssor_solve
precond/sor.c, 1852
__mess_realloc
Miscellaneous Helper Functions, 676

__mess_solver_options, 789
aux_stepdebug, 790
maxit, 790
omega, 790
restarts, 790
stepdebug, 790
tol, 790
__mess_solver_status, 791
converged, 791
it, 791
need_restart, 791
num_mvp, 791
relres, 792
res, 792
restarts, 792
__mm_read_data
read.c, 1814
__mm_read_info
read.c, 1815
__mm_read_vector_info
vector_read.c, 1972
__my_module_MOD_my_sub
symbols/__my_module_MOD_my_sub.c, 948
VerifyC/CMakeFiles/FortranCInterface/symbols/↩ __my_module_MOD_my_sub.c, 948
__my_module_MOD_my_sub.c, 948
__my_module_NMOD_my_sub
symbols/__my_module_NMOD_my_sub.c, 948
VerifyC/CMakeFiles/FortranCInterface/symbols/↩ __my_module_NMOD_my_sub.c, 949
__my_module_NMOD_my_sub.c, 948
__my_module__my_sub
symbols/__my_module__my_sub.c, 948
VerifyC/CMakeFiles/FortranCInterface/symbols/↩ __my_module__my_sub.c, 948
__my_module__my_sub.c, 947, 948
__mymodule_MOD_mysub
symbols/__mymodule_MOD_mysub.c, 949
VerifyC/CMakeFiles/FortranCInterface/symbols/↩ __mymodule_MOD_mysub.c, 949
__mymodule_MOD_mysub.c, 949
__mymodule_NMOD_mysub
symbols/__mymodule_NMOD_mysub.c, 950
VerifyC/CMakeFiles/FortranCInterface/symbols/↩ __mymodule_NMOD_mysub.c, 950
__mymodule_NMOD_mysub.c, 949, 950
__mymodule__mysub
symbols/__mymodule__mysub.c, 949
VerifyC/CMakeFiles/FortranCInterface/symbols/↩ __mymodule__mysub.c, 949
__mymodule__mysub.c, 949
__pad0__
CMakeCache.txt, 1256
__pad1__
CMakeCache.txt, 1256
__pad2__
CMakeFiles/FortranCInterface/CMakeCache.txt, 1256
__pad5__
CMakeFiles/FortranCInterface/VerifyC/CMake↩ Files/FortranCInterface/CMakeCache.txt, 1257
__plotter
x11init.c, 1977
__presordata, 792
matrix, 793
opt, 793
stat, 793
__project_A
H2 Model Order Reduction, 402
__project_B
H2 Model Order Reduction, 403
__project_C
H2 Model Order Reduction, 405
__project_Bmat
H2 Model Order Reduction, 404
__project_Cmat
H2 Model Order Reduction, 406
__select_ld
matrix/alloc.c, 961
__shellsort
shellsort.c, 1845
__stable_lyap, 793
ApEX_tmp, 794
AX, 794
Bhat, 794
Khat, 794
KX, 794
x2, 794
z2, 794
_equations_t, 794
converter, 795
name, 795
_glyap3
glyap3.c, 1436
_glyap3_eqn_t
glyap3.c, 1436
_glyap3_st, 795
Ahat, 796
Ehat, 796
eqn_type, 796
QA, 796
QE, 796
semidefinite, 796
_lowercase
read.c, 1815
vector_read.c, 1973
_mess_call, 796
func, 797
name, 797
_scalecat
lib/lrcf_adi/lrnm.c, 1604
_scalepoints, 797
label, 797
val, 797
_state

Interface to Python, 655
_substr
read.c, 1817
_sylv_eqn_sd_t
_sylv_solver_sd_st, 801
_sylv_equation_sd_t
lib/direct/singlesolver/sylvester_sparsedense.↩
c, 1874
_sylv_solver_d
lib/direct/singlesolver/sylvester_dense.c, 1864
_sylv_solver_d_st, 798
Ahat, 799
colsX, 799
Ehat, 799
Fhat, 799
Hhat, 799
isreal, 799
QA, 799
QE, 799
QF, 799
QH, 799
rowsX, 799
_sylv_solver_sd
lib/direct/singlesolver/sylvester_sparsedense.↩
c, 1874
_sylv_solver_sd_st, 800
_sylv_eqn_sd_t, 801
A, 801
colsX, 801
E, 801
Fhat, 801
Hhat, 801
isreal, 801
Q, 801
rowsX, 801
Z, 801
_uppercase
read.c, 1818
_w
mess_krylov_arnoldi_st, 858
_wtmp
mess_krylov_arnoldi_st, 858
~MESS_direct
MESS_direct, 830
~MESS_equation
MESS_equation, 845
~MESS_matrix
MESS_matrix, 863
~MESS_multidirect
MESS_multidirect, 877
~MESS_options
MESS_options, 889
~MESS_status
MESS_status, 903
~MESS_vector
MESS_vector, 918

A
__glyap_dae2, 768
__lyap, 777
_sylv_solver_sd_st, 801
bicgstab_solver, 808
blas_defs.h, 998
mat_diff, 822
mess_dynsys_st, 841
mvpdata, 929
mvpdatag, 931
res2nm_data_st, 944
a
blas_defs.h, 998
A Compact Loader Pool Implementation, 691
mess_threadpool, 692
mess_threadpool_clear, 692
mess_threadpool_init, 692
mess_threadpool_insert, 693
mess_threadpool_isdone, 693
mess_threadpool_waitdone, 694
mess_threadpool_worker, 694
mess_threadpooljob, 692
mess_threadpooljob_clear, 694
mess_threadpooljob_init, 694
A11
__glyap_dae1, 765
tests/data/bips98_606/source.txt, 1855
tests/data/siso_ww_vref_6405/source.txt, 1856
A12
__glyap_dae1, 765
tests/data/bips98_606/source.txt, 1855
tests/data/siso_ww_vref_6405/source.txt, 1856
A21
__glyap_dae1, 765
tests/data/bips98_606/source.txt, 1855
tests/data/siso_ww_vref_6405/source.txt, 1856
A22
__glyap_dae1, 765
tests/data/bips98_606/source.txt, 1855
tests/data/siso_ww_vref_6405/source.txt, 1856
A22sol
__glyap_dae1, 765
AB_cpx
banded_lapack, 807
AB_real
banded_lapack, 807
ABSRES_TOL
dense_nm.c, 1325
ADVANCED
CMakeCache.txt, 1256
AINV_apply
callback_equation.c, 1009
equation_glyap.c, 1350
equation_glyap_dae1.c, 1357
equation_glyap_so1.c, 1382
equation_glyap_so2.c, 1393
equation_stable.c, 1414
AINV_apply_shifts
equation_glyap_dae2.c, 1367
AINV_clear

callback_equation.c, 1009
equation_glyap.c, 1350
equation_stable.c, 1415
AINV_generate
callback_equation.c, 1009
equation_glyap.c, 1350
equation_stable.c, 1415
AINV
mess_equation_st, 848
ALPHAI
blas_defs.h, 998
ALPHAR
blas_defs.h, 999
AMD_ORDER
amd.c, 962
AMD_PINFO
amd.c, 962
AMD_VALID
amd.c, 962
APPLY_FUNCTION2
callback_equation.c, 1007
equations.c, 1419
APPLY_FUNCTION
callback_equation.c, 1007
equations.c, 1419
AR_W_VA
H2 Model Order Reduction, 395
ARCHITECTURE_ID
CMakeCCompilerId.c, 1257
CMakeCXXCompilerId.cpp, 1259
ARNOLDI_BREAKDOWN_TOL
arnoldi_template.c, 968
ARNOLDI_ITERATIONS
res2_template.c, 1831
ARNOLDI_RESIZE_CHUNK
arnoldi_template.c, 968
ARPACK_IPARAM_SIZE
arpack.c, 975
ARPACK_IPNTR_SIZE
arpack.c, 975
ARPACK_NCV
res2_template.c, 1831
AX_apply
callback_equation.c, 1009
equation_glyap.c, 1353
equation_glyap_dae1.c, 1360
equation_glyap_so1.c, 1387
equation_glyap_so2.c, 1398
equation_stable.c, 1416
AX_apply_N
equation_glyap_dae2.c, 1373
AX_apply_shifts
equation_glyap_dae2.c, 1373
AX_apply_T
equation_glyap_dae2.c, 1374
AX_clear
callback_equation.c, 1009
equation_stable.c, 1417
AX_generate
callback_equation.c, 1009
equation_stable.c, 1417
AB
blas_defs.h, 998
add.c, 950
__mess_matrix_add_bothsparse_complex_csc, 951
__mess_matrix_add_bothsparse_complex_csr, 952
__mess_matrix_add_bothsparse_complex_real↩_csc, 952
__mess_matrix_add_bothsparse_complex_real↩_csr, 953
__mess_matrix_add_bothsparse_csc, 954
__mess_matrix_add_bothsparse_csr, 955
__mess_matrix_add_bothsparse_real_complex↩_csc, 955
__mess_matrix_add_bothsparse_real_complex↩_csr, 956
add_kernel_complex, 957
add_kernel_complex_real, 957
add_kernel_real, 958
add_kernel_real_complex, 959
add_kernel_complex
add.c, 957
add_kernel_complex_real
add.c, 957
add_kernel_real
add.c, 958
add_kernel_real_complex
add.c, 959
addcol.c, 959
Additional Matrix Equation Solvers, 553
MESS_SIGN_SCALE_DET, 554
MESS_SIGN_SCALE_FRO, 554
MESS_SIGN_SCALE_NONE, 554
mess_dense_nm_gmare, 555
mess_dense_nm_gmpare, 556
mess_dense_nm_gpare, 558
mess_dense_res_gmpare, 559
mess_lrcf_gsignber, 561
mess_lrcf_gsignfac, 563
mess_lrcf_signber, 564
mess_lrcf_signfac, 566
mess_sign_scale_t, 554
adi_maxit
mess_options_st, 894
adi_output
mess_options_st, 894
adi_rel_change_tol
mess_options_st, 894
adi_res2_tol
mess_options_st, 894
adi_res2c_tol
mess_options_st, 895
adi_shifts_arp_m
mess_options_st, 895

adi_shifts_arp_p
mess_options_st, 895
adi_shifts_b0
mess_options_st, 895
adi_shifts_l0
mess_options_st, 895
adi_shifts_p
mess_options_st, 895
adi_shifts_paratype
mess_options_st, 895
Ahat
_glyap3_st, 796
_sylv_solver_d_st, 799
alloc.c, 960, 961
Allocate, Convert and Access to Matrices, 109
mess_matrix_addcols, 112
mess_matrix_addcols1, 113
mess_matrix_addcols2p, 114
mess_matrix_alloc, 115
mess_matrix_alloc_setld, 115
mess_matrix_bandwidth, 115
mess_matrix_cat, 116
mess_matrix_colscalem, 117
mess_matrix_colsub, 118
mess_matrix_complex_from_parts, 119
mess_matrix_convert, 120
mess_matrix_convert_csc_csr, 121
mess_matrix_convert_csr_csc, 122
mess_matrix_coord, 123
mess_matrix_copy, 124
mess_matrix_csc, 124
mess_matrix_csr, 125
mess_matrix_dense_from_carray, 126
mess_matrix_dense_from_farray, 127
mess_matrix_diag, 127
mess_matrix_diag_from_vector, 128
mess_matrix_diagpos, 129
mess_matrix_dupl, 129
mess_matrix_eliminate_zeros, 130
mess_matrix_eye, 130
mess_matrix_eyec, 131
mess_matrix_getcol, 131
mess_matrix_getelement, 132
mess_matrix_getrow, 133
mess_matrix_imagpart, 134
mess_matrix_lift, 134
mess_matrix_need_alloc, 135
mess_matrix_one_value, 136
mess_matrix_one_valuec, 136
mess_matrix_ones, 137
mess_matrix_rand, 137
mess_matrix_rand_coord, 138
mess_matrix_rand_csc, 139
mess_matrix_rand_csr, 140
mess_matrix_rand_dense, 141
mess_matrix_rand_dense_uniform, 142
mess_matrix_rand_init, 142
mess_matrix_realloc, 143
mess_matrix_realpart, 143
mess_matrix_reset, 144
mess_matrix_resize, 144
mess_matrix_rowscalem, 145
mess_matrix_rowsub, 145
mess_matrix_setcol, 146
mess_matrix_setelement, 147
mess_matrix_setelement_complex, 147
mess_matrix_setrow, 148
mess_matrix_sort, 149
mess_matrix_sub, 150
mess_matrix_symfillup, 150
mess_matrix_tocomplex, 151
mess_matrix_toreal, 151
mess_matrix_totype, 151
mess_matrix_tridiag, 152
mess_matrix_tril, 153
mess_matrix_triu, 153
mess_matrix_view, 154
mess_matrix_zeros, 154
alpha
blas_defs.h, 998
tests/data/TripleChain/source.txt, 1857
amd.c, 961
AMD_ORDER, 962
AMD_PINFO, 962
AMD_VALID, 962
COLAMD_RECOMMENDED, 962
COLAMD_REPORT, 962
COLAMD_SET_DEFAULTS, 962
COLAMD, 962
Amsolver
__glyap_dae1, 765
__lyap, 777
any.c, 962
ApEINV2_apply
equation_glyap_dae1.c, 1358
ApEINV2_clear
equation_glyap_dae1.c, 1359
ApEINV2_generate
equation_glyap_dae1.c, 1360
ApEINV_apply
callback_equation.c, 1009
equation_glyap.c, 1350
equation_glyap_so1.c, 1383
equation_glyap_so2.c, 1394
equation_stable.c, 1415
ApEINV_apply_memory0
equation_glyap_dae2.c, 1367
ApEINV_apply_memory1
equation_glyap_dae2.c, 1369
ApEINV_clear
callback_equation.c, 1009
equation_glyap.c, 1351
equation_glyap_so1.c, 1385
equation_glyap_so2.c, 1396
equation_stable.c, 1416
ApEINV_clear_memory0

equation_glyap_dae2.c, 1370
ApEINV_clear_memory1
equation_glyap_dae2.c, 1371
ApEINV_generate
callback_equation.c, 1009
equation_glyap.c, 1351
equation_glyap_so1.c, 1386
equation_glyap_so2.c, 1397
equation_stable.c, 1416
ApEINV_generate_memory0
equation_glyap_dae2.c, 1371
ApEINV_generate_memory1
equation_glyap_dae2.c, 1372
ApEINV
mess_equation_st, 848
ApEX_apply
callback_equation.c, 1009
equation_glyap.c, 1351
ApEX_clear
callback_equation.c, 1009
equation_glyap.c, 1352
ApEX_generate
callback_equation.c, 1009
equation_glyap.c, 1352
ApEX_tmp
__lyap, 777
__stable_lyap, 794
ApEX
mess_equation_st, 848
apply
mess_equation_st::AINV_st, 802
mess_equation_st::AX_st, 805
mess_equation_st::ApEINV_st, 803
mess_equation_st::ApEX_st, 804
mess_equation_st::EINV_st, 812
mess_equation_st::EX_st, 813
applyPIsolver
__glyap_dae2, 768
applyUe_left
schur.c, 1843
applyUe_right
schur.c, 1843
Arithmetic Operations on Matrices, 50
mess_matrix_add, 52
mess_matrix_addc, 53
mess_matrix_any, 54
mess_matrix_biorth, 55
mess_matrix_colsums, 56
mess_matrix_conj, 57
mess_matrix_ctranspose, 58
mess_matrix_fro_inn, 58
mess_matrix_gaxpy, 59
mess_matrix_gbiorth, 60
mess_matrix_map, 61
mess_matrix_map_abs, 62
mess_matrix_map_acos, 63
mess_matrix_map_acosh, 63
mess_matrix_map_arg, 63
mess_matrix_map_asin, 63
mess_matrix_map_asinh, 63
mess_matrix_map_atan, 63
mess_matrix_map_atanh, 63
mess_matrix_map_ceil, 63
mess_matrix_map_conj, 63
mess_matrix_map_cos, 63
mess_matrix_map_cosh, 63
mess_matrix_map_exp, 63
mess_matrix_map_expm1, 63
mess_matrix_map_floor, 63
mess_matrix_map_isfinite, 63
mess_matrix_map_isinf, 63
mess_matrix_map_isnan, 63
mess_matrix_map_log, 63
mess_matrix_map_not, 63
mess_matrix_map_round, 63
mess_matrix_map_sin, 64
mess_matrix_map_sinh, 64
mess_matrix_map_sqrt, 64
mess_matrix_map_tan, 64
mess_matrix_map_tanh, 64
mess_matrix_mgs, 64
mess_matrix_mgs_add, 65
mess_matrix_mgs_inplace, 66
mess_matrix_multiply, 67
mess_matrix_mvp, 67
mess_matrix_null, 68
mess_matrix_orth, 69
mess_matrix_proj_gmgs, 70
mess_matrix_proj_mgs, 71
mess_matrix_proj_sym, 72
mess_matrix_rank, 73
mess_matrix_rowsums, 73
mess_matrix_scale, 74
mess_matrix_scalec, 75
mess_matrix_trace, 75
mess_matrix_tracec, 76
mess_matrix_transpose, 77
mess_matrix_xtranspose, 77
Arithmetic Operations on Matrix Columns, 79
mess_matrix_colaxpy, 80
mess_matrix_colaxpy2, 80
mess_matrix_coldot, 81
mess_matrix_coldotc, 82
mess_matrix_coldotcE, 83
mess_matrix_coldotE, 84
mess_matrix_colnorm, 85
mess_matrix_colnormE, 86
mess_matrix_colscale, 87
mess_matrix_colvecaxpy, 88
mess_matrix_colvecdot, 89
mess_matrix_colvecdotc, 90
Arithmetic Operations with Vectors, 197
mess_vector_axpy, 198
mess_vector_axpyc, 199
mess_vector_conj, 199
mess_vector_diffnorm, 200

mess_vector_diffnorminf, 200
mess_vector_dot, 200
mess_vector_dotc, 201
mess_vector_dotu, 201
mess_vector_max, 201
mess_vector_maxvalue, 202
mess_vector_minvalue, 202
mess_vector_norm, 202
mess_vector_norm1, 203
mess_vector_norm2, 203
mess_vector_norminf, 204
mess_vector_scale, 204
mess_vector_scalec, 204
mess_vector_scalee, 198
arnoldi.c, 963
mvp_arnoldi, 964
mvp_arnoldi_inv, 964
mvp_arnoldig, 965
mvp_arnoldig_inv, 966
arnoldi_template.c, 967
ARNOLDI_BREAKDOWN_TOL, 968
ARNOLDI_RESIZE_CHUNK, 968
mess_krylov_arnoldi, 968
mess_krylov_arnoldi_H_largest_eigenvalue, 971
mess_krylov_arnoldi_clear, 968
mess_krylov_arnoldi_create, 969
mess_krylov_arnoldi_getH, 970
mess_krylov_arnoldi_getV, 970
mess_krylov_arnoldi_init, 972
mess_krylov_arnoldi_number_arnoldi_vectors, 972
mess_krylov_arnoldi_print, 972
mess_krylov_arnoldi_step, 973
arpack.c, 974
ARPACK_IPARAM_SIZE, 975
ARPACK_IPNTR_SIZE, 975
arpack_arnoldi_complex, 975
arpack_arnoldi_real, 975
arpack_lanczos_real, 976
arpack_print_iparam, 977
F77_GLOBAL, 977
arpack_arnoldi_complex
arpack.c, 975
arpack_arnoldi_real
arpack.c, 975
arpack_lanczos_real
arpack.c, 976
arpack_print_iparam
arpack.c, 977
Asolver
__lyap, 777
mvpdatag, 931
VWjob, 947
Atilde
mvpdatag, 931
aux
mess_equation_st, 848
aux_stepdebug
__mess_solver_options, 790
AX
__stable_lyap, 794
mess_equation_st, 848
axpy.c, 977
__ONE, 978
__axpy_rc, 978

B
__lyap, 778
blas_defs.h, 999
mat_diff, 822
mess_dynsys_st, 841
mess_equation_st, 848
res2_data_st, 942
res2nm_data_st, 944
tests/data/bips98_606/source.txt, 1855
tests/data/siso_ww_vref_6405/source.txt, 1856
b
blas_defs.h, 999
VWjob, 947
b0
mess_eigen_arpack_options_t, 843
BANDWIDHT_SWITCH_THRESHOLD
banded.c, 980
BETA
blas_defs.h, 999
BTRSIZE
error_macro.h, 1424
BWORK
blas_defs.h, 999
background
multilu, 927
backslash.c, 978
Balanced Truncation, 372
MESS_BT_POSITION_VELOCITY, 374
MESS_BT_POSITION, 374
MESS_BT_VELOCITY_POSITION, 374
MESS_BT_VELOCITY, 374
mess_bt_chooseorder_default, 375
mess_bt_chooseorder_func, 374
mess_bt_chooseorder_minreal, 375
mess_bt_getsotypestr, 376
mess_bt_gglrsrm, 376
mess_bt_gslrsrm, 378
mess_bt_lrsrm, 380
mess_bt_lrsrm_somor, 382
mess_bt_lrsrm_somor_so, 384
mess_bt_lrsrm_somor_so_gram, 386
mess_bt_options, 374
mess_bt_options_clear, 387
mess_bt_options_init, 389
mess_bt_options_print, 389
mess_bt_status, 375
mess_bt_status_clear, 389
mess_bt_status_init, 390
mess_bt_status_print, 390
banded.c, 979
BANDWIDHT_SWITCH_THRESHOLD, 980

banded_lapack_clear, 980
banded_lapack_solve, 980
banded_lapack_solveh, 981
banded_lapack_solvem, 982
banded_lapack_solvemh, 982
banded_lapack_solvemt, 983
banded_lapack_solvet, 984
IDX2F, 980
LAPACK_ERROR, 980
umfpack_dl_defaults, 980
umfpack_zl_defaults, 980
banded_lapack, 806
AB_cpx, 807
AB_real, 807
info, 807
invperm, 807
ipiv, 807
kl, 807
ku, 807
n, 807
nband, 807
perm, 807
banded_lapack_clear
banded.c, 980
banded_lapack_solve
banded.c, 980
banded_lapack_solveh
banded.c, 981
banded_lapack_solvem
banded.c, 982
banded_lapack_solvemh
banded.c, 982
banded_lapack_solvemt
banded.c, 983
banded_lapack_solvet
banded.c, 984
bandw_gipost
cuthill.c, 1315
bandw_level_struct
cuthill.c, 1315
bandw_mark_connected
cuthill.c, 1316
bandw_number_adj_vert
cuthill.c, 1317
bandw_red_perm
cuthill.c, 1318
bandwidth.c, 985
Basic Matrix and Vector Operations, 714
bc
VWjob, 947
ber_sgn_fac.c, 985
beta
blas_defs.h, 999
tests/data/TripleChain/source.txt, 1857
Bhat
__stable_lyap, 794
bicgstab.c, 985, 988
bicgstab_clear
direct/singlesolver/bicgstab.c, 986
bicgstab_solve
direct/singlesolver/bicgstab.c, 986
bicgstab_solvem
direct/singlesolver/bicgstab.c, 987
bicgstab_solver, 807
A, 808
opt, 808
p, 808
pre, 808
q, 808
stat, 808
biorth.c, 989
blas_defs.h, 989
A, 998
a, 998
ALPHAI, 998
ALPHAR, 999
AB, 998
alpha, 998
B, 999
b, 999
BETA, 999
BWORK, 999
beta, 999
C, 999
COMPZ, 999
D, 999
DIFF, 999
DWORK, 999
diag, 999
DX, 999
DY, 999
E, 999
EIGSRC, 999
F, 999
F77_GLOBAL, 994–998
FACT, 999
FERR, 1000
ferr, 999
H, 1000
HOWMNY, 1000
how, 1000
IFAILL, 1000
IFAILR, 1000
IHI, 1000
ILO, 1000
INCX, 1000
INCY, 1000
INFO, 1000
INITV, 1000
IPIV, 1000
ISGN, 1000
ISOLVE, 1000
IWORK, 1001
ijob, 1000
incx, 1000
incy, 1000

info, 1000
ipiv, 1000
lwork, 1000
JOBVSR, 1001
JOBVR, 1001
JOBVT, 1001
JOBR, 1001
JOB, 1001
JPVT, 1001
K, 1001
k, 1001
KL, 1001
KU, 1001
LDAB, 1001
LDVSL, 1002
LDVSR, 1002
LDVL, 1002
LDVR, 1002
LDVS, 1002
LDVT, 1002
LDWORK, 1002
LDA, 1001
LDB, 1001
LDC, 1001
LDD, 1001
LDE, 1001
LDF, 1001
LDH, 1002
LDQ, 1002
LDT, 1002
LDU, 1002
LDX, 1002
LDZ, 1002
LVDS, 1002
LWORK, 1002
lda, 1001
ldb, 1001
ldc, 1001
lwork, 1002
M, 1002
m, 1002
MESS_LAPACK_LOGICAL, 994
MM, 1003
N, 1003
n, 1003
n2, 1003
NRHS, 1003
NB, 1003
nrhs, 1003
Q, 1003
RWORK, 1003
rwork, 1003
S, 1003
SCALE, 1003
SDIM, 1003
SDA, 1003
SELECT, 1003
SEP, 1003
SORT, 1003
scale, 1003
seed, 1003
sep, 1003
T, 1003
TAU, 1004
TRANSA, 1004
TRANS, 1004
Trans, 1004
trans, 1004
transA, 1004
transB, 1004
U, 1004
UNIT, 1004
UPLO, 1004
uplo, 1004
VSL, 1004
VSR, 1004
values, 1004
VL, 1004
VR, 1004
VS, 1004
VT, 1004
W, 1004
WORK, 1005
WI, 1004
work, 1005
WR, 1005
X, 1005
x, 1005
y, 1005
Z, 1005
ZX, 1005
ZY, 1005
blockpart.c, 1005
lblock_dfs, 1005
swap, 1006
breakdown
mess_krylov_arnoldi_st, 858
breakdown_tol
mess_krylov_arnoldi_st, 858
byte_size
MESS_datatype_t, 828
MESS_direct_cholpackage_t, 833
MESS_direct_lupackage_t, 836
MESS_equation_t, 851
MESS_matrix, 863
MESS_memusage_t, 874
MESS_multidirect_t, 883
MESS_operation_t, 887
MESS_options, 890
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_status, 904
MESS_storage_t, 909
MESS_vector, 918

C
blas_defs.h, 999

mess_dynsys_st, 841
mess_equation_st, 848
norm2data, 934
res2nm_data_st, 944
tests/data/bips98_606/source.txt, 1855
tests/data/siso_ww_vref_6405/source.txt, 1856
c
VWjob, 947
CALL2
check_cat.c, 1043
check_coldot.c, 1054
check_colnorm.c, 1060
check_diag.c, 1087
check_mul.c, 1175
check_mul_small.c, 1181
check_mul_tall.c, 1187
CALL
call_macro.h, 1006
CCSRFmt, 808
colptr, 809
data_type, 809
n, 809
nnz, 809
rowptr, 809
values, 809
values_cpx, 809
CCSRMat
iluk.c, 1462
CHECK_ADD
check_add.c, 1021
CHECK_ALL_STORAGETYPEM
check_trisolve.c, 1243
CHECK_ALL_STORAGETYPE
check_trisolve.c, 1243
CHECK_ANY
check_any.c, 1024
CHECK_ARNOLDI
check_arnoldi_template.c, 1025
CHECK_BACKSLASHM
check_backslash.c, 1036
CHECK_BACKSLASH
check_backslash.c, 1036
CHECK_CONVERTCHOLMOD_DENSEMATRIX
check_cholmod.c, 1047
CHECK_CONVERTCHOLMOD_SPARSEMATRIX
check_cholmod.c, 1047
CHECK_CONVERTCHOLMOD_VECTOR
check_cholmod.c, 1048
CHECK_FUNCTION
check_dae2_handles_adi.c, 1076
check_dae2_handles_para.c, 1078
check_so1_handles.c, 1226
check_so2_handles.c, 1228
CHECK_KRON
check_kron.c, 1130
CHECK_MESS_MATRIX_NULL
check_matrix_null.c, 1165
CHECK_NULL
write.c, 1976
CHECK_TRISOLVEM
check_trisolve.c, 1244
CHECK_TRISOLVE
check_trisolve.c, 1243
CHECKADDCOL2P
check_addcol2p.c, 1022
CHECKCOLDOTCE
check_coldotcE.c, 1057
CHECKCOLSCALE
check_colscale.c, 1062
CHECKCOLVCAXPY
check_colvecaxpy.c, 1066
CHECKCOLVDOTE
check_coldotE.c, 1059
CHECKCONDEST
check_condest.c, 1071
CHECKGAXPY
check_gaxpy.c, 1117
CHECKMARTIXBIORTH
check_matrix_biorth.c, 1160
CHECKMARTIXGBIORTH
check_matrix_gbiorth.c, 1163
CHECKMATRIXLIFT
check_lift.c, 1137
CHECKMATRIXOCLAXPY
check_matrix_colaxpy.c, 1161
CHECKMATRIXSUM
check_matrix_sum.c, 1166
CHECKMGSADDE
check_mgsadd.c, 1172
CHECKMGSADD
check_mgsadd.c, 1172
CHECKMGSINPLACE
check_mgs_inplace.c, 1171
CHECKMGS
check_mgs.c, 1169
CHECKNORM
check_norm.c, 1196
CHECKORTH
check_orth.c, 1200
CHECKPERM
check_perm.c, 1202
CHECKPROSYM
check_proj_sym.c, 1212
CHECKROWSUB
check_colsub.c, 1064
check_rowsub.c, 1216
CHECKSUB
check_sub.c, 1230
CHECKTRACE
check_trace.c, 1238
CHECKTRANSPOSE
check_transpose.c, 1240
CHECKVECTORLIFT
check_lift.c, 1137
CLEAR_FUNCTION
callback_equation.c, 1007

CMakeCCompilerId.c, 1257
ARCHITECTURE_ID, 1257
COMPILER_ID, 1257
DEC, 1257
HEX, 1257
info_arch, 1258
info_compiler, 1258
info_language_dialect_default, 1258
info_platform, 1258
main, 1258
PLATFORM_ID, 1258
STRINGIFY_HELPER, 1258
STRINGIFY, 1258
CMakeCXXCompilerId.cpp, 1258
ARCHITECTURE_ID, 1259
COMPILER_ID, 1259
DEC, 1259
HEX, 1259
info_arch, 1260
info_compiler, 1260
info_language_dialect_default, 1260
info_platform, 1260
main, 1260
PLATFORM_ID, 1259
STRINGIFY_HELPER, 1260
STRINGIFY, 1260
CMakeCache.txt, 1255, 1256
__pad0__, 1256
__pad1__, 1256
ADVANCED, 1256
cscutils, 1256
general, 1256
lgfortran, 1256
lpthread, 1256
m, 1256
so, 1256
CMakeFiles/FortranCInterface/CMakeCache.txt
__pad2__, 1256
myfort, 1256
CMakeFiles/FortranCInterface/VerifyC/CMakeFiles/↩
FortranCInterface/CMakeCache.txt
__pad5__, 1257
myfort, 1257
CMakeFortranCompilerId.F, 1260
cmakefortrancompilerid, 1261
DEC_1, 1261
DEC_2, 1261
DEC_3, 1261
DEC_4, 1261
DEC_5, 1261
DEC_6, 1261
DEC_7, 1261
DEC_8, 1261
HEX_1, 1261
HEX_2, 1261
HEX_3, 1261
HEX_4, 1261
HEX_5, 1261
HEX_6, 1261
HEX_7, 1261
HEX_8, 1261
CMakeRuleHashes.txt, 1261
COLAMD_RECOMMENDED
amd.c, 962
COLAMD_REPORT
amd.c, 962
COLAMD_SET_DEFAULTS
amd.c, 962
COLAMD
amd.c, 962
COMPILER_ID
CMakeCCompilerId.c, 1257
CMakeCXXCompilerId.cpp, 1259
COMPLEX_ZGEES_IF_REAL
schur.c, 1843
COMPUTESOL
check_colvecaxpy.c, 1066
COMPZ
blas_defs.h, 999
CSPARSE_ORDER
csparse.c, 1279
csparse_chol.c, 1298
CSPARSE_TOL
csparse.c, 1279
calc_finalh2
mess_h2_options_st, 854
calc_h2err
mess_h2_options_st, 854
call_arnoldi
check_arpack.c, 1028
call_lanczos
check_arpack_lanczos.c, 1032
call_macro.h, 1006
CALL, 1006
callback_AINV_apply
equations.c, 1420
callback_AINV_clear
equations.c, 1420
callback_AINV_generate
equations.c, 1420
callback_AX_apply
equations.c, 1420
callback_AX_clear
equations.c, 1420
callback_AX_generate
equations.c, 1420
callback_ApEINV_apply
equations.c, 1420
callback_ApEINV_clear
equations.c, 1420
callback_ApEINV_generate
equations.c, 1420
callback_ApEX_apply
equations.c, 1420
callback_ApEX_clear
equations.c, 1420

callback_ApEX_generate
equations.c, 1420
callback_EINV_apply
equations.c, 1420
callback_EINV_clear
equations.c, 1421
callback_EINV_generate
equations.c, 1421
callback_EX_apply
equations.c, 1421
callback_EX_clear
equations.c, 1421
callback_EX_generate
equations.c, 1421
callback_equation.c, 1006
AINV_apply, 1009
AINV_clear, 1009
AINV_generate, 1009
APPLY_FUNCTION2, 1007
APPLY_FUNCTION, 1007
AX_apply, 1009
AX_clear, 1009
AX_generate, 1009
ApEINV_apply, 1009
ApEINV_clear, 1009
ApEINV_generate, 1009
ApEX_apply, 1009
ApEX_clear, 1009
ApEX_generate, 1009
CLEAR_FUNCTION, 1007
clear, 1009
EINV_apply, 1009
EINV_clear, 1009
EINV_generate, 1009
EX_apply, 1009
EX_clear, 1009
EX_generate, 1009
GENERATE_FUNCTION2, 1008
GENERATE_FUNCTION, 1008
parameter, 1009
calloc_call_t
malloc.c, 1610
cancel_it
mess_h2_status_st, 856
cancel_sigma
mess_h2_status_st, 856
care.c, 1010, 1011
cat.c, 1011, 1020
category.dox, 1020
cc
VWjob, 947
ccsrptr
iluk.c, 1462
cg.c, 1020
check
check_convert.c, 1073
check_add.c, 1021
CHECK_ADD, 1021
main, 1021
check_addcol2p.c, 1022
CHECKADDCOL2P, 1022
main, 1022
check_any.c, 1023
CHECK_ANY, 1024
f_cpx, 1024
f_real, 1024
main, 1024
check_arnoldi_template.c, 1025
CHECK_ARNOLDI, 1025
main, 1025
check_arnoldi_template_nrm.c, 1026
main, 1027
check_arpack.c, 1028
call_arnoldi, 1028
check_ev, 1029
main, 1030
check_arpack_lanczos.c, 1031
call_lanczos, 1032
check_ev, 1032
main, 1033
mvp, 1034
check_backslash.c, 1035
CHECK_BACKSLASHM, 1036
CHECK_BACKSLASH, 1036
main, 1037
check_bandwidth.c, 1037
main, 1038
check_ber_sign_fac.c, 1038
main, 1039
check_biorth.c, 1040
main, 1041
check_cat
check_cat.c, 1043
check_cat.c, 1042
CALL2, 1043
check_cat, 1043
data_ABCD, 1045
data_ABC, 1045
data_ABD, 1046
data_ACD, 1046
data_AB, 1045
data_AC, 1046
data_AD, 1046
data_BCD, 1046
data_BC, 1046
data_A, 1045
data_B, 1046
data_C, 1046
data_D, 1047
main, 1044
check_cholmod.c, 1047
CHECK_CONVERTCHOLMOD_DENSEMATRIX, 1047
CHECK_CONVERTCHOLMOD_SPARSEMATRIX, 1047
CHECK_CONVERTCHOLMOD_VECTOR, 1048

cholmod_print_dense_all, 1048
cholmod_print_sparse_all, 1048
main, 1048
check_cimport.c, 1049
main, 1050
check_cimport_c2d.c, 1051
check_complex, 1051
check_real, 1051
main, 1052
check_colaxpy2.c, 1053
main, 1053
check_coldot.c, 1053
CALL2, 1054
main, 1054
matrix_data, 1055
check_coldotE.c, 1058
CHECKCOLVDOTE, 1059
main, 1059
check_coldotc.c, 1055
main, 1056
matrix_data, 1056
check_coldotcE.c, 1056
CHECKCOLDOTCE, 1057
main, 1057
check_colnorm.c, 1059
CALL2, 1060
main, 1060
matrix_data, 1061
check_colscale.c, 1061
CHECKCOLSCALE, 1062
complex_matrix_data, 1063
main, 1062
real_matrix_data, 1063
check_colsub.c, 1063
CHECKROWSUB, 1064
main, 1064
check_colvecaxpy.c, 1065
CHECKCOLVCAXPY, 1066
COMPUTESOL, 1066
main, 1066
check_colvecdot.c, 1067
main, 1068
matrix_data, 1069
check_colvecdotc.c, 1069
main, 1070
matrix_datac, 1070
matrix_datar, 1070
check_complex
check_cimport_c2d.c, 1051
check_condest.c, 1071
CHECKCONDEST, 1071
main, 1071
rand50x50condest, 1072
check_convert.c, 1073
check, 1073
main, 1074
check_dae2_handles_adi.c, 1075
CHECK_FUNCTION, 1076
main, 1076
check_dae2_handles_para.c, 1077
CHECK_FUNCTION, 1078
main, 1078
check_decomp.c, 1079
main, 1080
check_dense_nm.c, 1081
main, 1081
check_dense_nm_tridiag.c, 1082
main, 1083
check_det.c, 1083
main, 1084
test_det, 1084
check_dgeqp3.c, 1085
main, 1085
check_diag.c, 1086
CALL2, 1087
main, 1087
matrix_data, 1088
matrix_datac, 1088
check_direct_care.c, 1088
main, 1089
check_dynorm2.c, 1089
fdmcol, 1089
main, 1090
check_dynorm2t.c, 1091
fdmcol, 1091
main, 1091
check_dynorm_indefinite2.c, 1092
main, 1092
test_cases, 1093
check_dynorm_indefinitef.c, 1094
main, 1095
test_cases, 1095
check_dynormf.c, 1096
fdmcol, 1096
main, 1097
check_dynormft.c, 1097
fdmcol, 1097
main, 1098
check_eig.c, 1098
check_eigen_vector, 1099
main, 1100
check_eig2.c, 1101
check_eigen_vector, 1102
main, 1103
check_eig_complex.c, 1104
check_eigen_vector, 1105
main, 1106
check_eig_hessenberg_schur.c, 1107
main, 1108
check_eig_hessenberg_schur_largest_ev.c, 1109
main, 1109
check_eigen_vector
check_eig.c, 1099
check_eig2.c, 1102
check_eig_complex.c, 1105
check_eliminate_zeros.c, 1110

f_cpx, 1111
f_real, 1111
iszero_cpx, 1111
iszero_real, 1111
main, 1111
check_eps.c, 1112
main, 1113
check_ev
check_arpack.c, 1029
check_arpack_lanczos.c, 1032
check_fputs_block.c, 1113
main, 1114
check_freelist.c, 1114
main, 1114
check_fro_inn.c, 1115
main, 1116
matrix_data1, 1116
matrix_data2, 1116
check_gaxpy.c, 1116
CHECKGAXPY, 1117
main, 1117
check_getL_getU
check_lapackqr.c, 1133
check_getLU.c, 1120
main, 1120
select_solver, 1121
testmat, 1122
check_getelement.c, 1118
main, 1119
check_gsign.c, 1123
main, 1124
check_hessenberg.c, 1125
main, 1126
check_itsolver.c, 1127
check_solver, 1127
main, 1128
check_kron.c, 1129
CHECK_KRON, 1130
main, 1130
testkron, 1131
check_lapackqr.c, 1132
check_getL_getU, 1133
check_solve, 1133
check_solvem, 1134
main, 1135
check_lift.c, 1136
CHECKMATRIXLIFT, 1137
CHECKVECTORLIFT, 1137
main, 1138
check_linesearch.c, 1138
main, 1139
SCALAR_ERROR_MACRO, 1139
check_lradi_dae1.c, 1140
main, 1140
check_lradi_dae2.c, 1141
main, 1142
check_lradi_glyap.c, 1143
main, 1143
check_lradi_so.c, 1144
main, 1145
check_lradi_unsym.c, 1146
fdmcol, 1146
fx, 1146
fy, 1146
main, 1147
check_lrnm_dae1.c, 1148
main, 1149
check_lrnm_dae2.c, 1150
main, 1151
check_lrnm_g.c, 1152
main, 1152
check_lrnm_so.c, 1153
main, 1154
check_lrnm_unsym.c, 1155
fdmcol, 1155
fdmcol2, 1155
fx, 1155
fy, 1156
main, 1156
check_lyap_sign_fac.c, 1157
main, 1158
check_matrix_biorth.c, 1159
CHECKMARTIXBIORTH, 1160
main, 1160
check_matrix_colaxpy.c, 1161
CHECKMATRIXOCLAXPY, 1161
main, 1162
check_matrix_gbiorth.c, 1162
CHECKMARTIXGBIORTH, 1163
generatespd, 1163
main, 1163
check_matrix_null.c, 1164
CHECK_MESS_MATRIX_NULL, 1165
main, 1165
check_matrix_sum.c, 1165
CHECKMATRIXSUM, 1166
main, 1166
check_matrixio.c, 1167
main, 1168
check_mgs.c, 1169
CHECKMGS, 1169
main, 1169
check_mgs_inplace.c, 1170
CHECKMGSINPLACE, 1171
main, 1171
check_mgsadd.c, 1171
CHECKMGSADDE, 1172
CHECKMGSADD, 1172
generatespd, 1173
main, 1173
check_mul
check_mul.c, 1175
check_mul_small.c, 1181
check_mul_tall.c, 1187
check_mul.c, 1174
CALL2, 1175

check_mul, 1175
main, 1175
matrix_HiHi, 1176
matrix_HiNi, 1176
matrix_HiTi, 1177
matrix_HiN, 1176
matrix_HiT, 1176
matrix_NHi, 1177
matrix_NNi, 1178
matrix_NTi, 1178
matrix_NiHi, 1177
matrix_NiNi, 1177
matrix_NiTi, 1178
matrix_NiN, 1177
matrix_NiT, 1177
matrix_NN, 1178
matrix_NT, 1178
matrix_THi, 1178
matrix_TNi, 1179
matrix_TTi, 1180
matrix_TiHi, 1178
matrix_TiNi, 1179
matrix_TiTi, 1179
matrix_TiN, 1179
matrix_TiT, 1179
matrix_TN, 1179
matrix_TT, 1179
matrix_data, 1176
matrixi_data, 1180
check_mul_small.c, 1180
CALL2, 1181
check_mul, 1181
main, 1182
matrix_ANAH, 1183
matrix_ANAiH, 1184
matrix_ANAiT, 1184
matrix_ANAT, 1184
matrix_ANBiN, 1184
matrix_ANBN, 1184
matrix_Ai, 1183
matrix_AiNAH, 1183
matrix_AiNAiH, 1183
matrix_AiNAiT, 1183
matrix_AiNAT, 1183
matrix_AiNBiN, 1183
matrix_AiNBN, 1183
matrix_BHAH, 1184
matrix_BHAiH, 1184
matrix_BHAiT, 1184
matrix_BHAT, 1184
matrix_BHBiN, 1184
matrix_BHBN, 1184
matrix_BTAH, 1185
matrix_BTAiH, 1185
matrix_BTAiT, 1185
matrix_BTAT, 1185
matrix_BTBiN, 1185
matrix_BTBN, 1185
matrix_Bi, 1184
matrix_BiHAH, 1184
matrix_BiHAiH, 1184
matrix_BiHAiT, 1184
matrix_BiHAT, 1184
matrix_BiHBiN, 1184
matrix_BiHBN, 1184
matrix_BiTAH, 1184
matrix_BiTAiH, 1184
matrix_BiTAiT, 1184
matrix_BiTAT, 1184
matrix_BiTBiN, 1184
matrix_BiTBN, 1185
matrix_A, 1183
matrix_B, 1184
check_mul_tall.c, 1185
CALL2, 1187
check_mul, 1187
main, 1188
matrix_ANAiH, 1190
matrix_ANAiT, 1190
matrix_ANAT, 1190
matrix_ANBiN, 1190
matrix_ANBN, 1190
matrix_Ai, 1189
matrix_AiNAH, 1189
matrix_AiNAiH, 1189
matrix_AiNAiT, 1189
matrix_AiNAT, 1189
matrix_AiNBiN, 1189
matrix_AiNBN, 1189
matrix_BHAiH, 1190
matrix_BHAiT, 1190
matrix_BHBiN, 1190
matrix_BTAiH, 1191
matrix_BTAiT, 1191
matrix_BTAT, 1191
matrix_BTBiN, 1191
matrix_BTBN, 1191
matrix_Bi, 1190
matrix_BiHAH, 1190
matrix_BiHAiH, 1190
matrix_BiHAiT, 1190
matrix_BiHAT, 1190
matrix_BiHBiN, 1190
matrix_BiHBN, 1191
matrix_BiTAH, 1191
matrix_BiTAiH, 1191
matrix_BiTAiT, 1191
matrix_BiTAT, 1191
matrix_BiTBiN, 1191
matrix_BiTBN, 1191
matrix_A, 1189
matrix_B, 1190
check_mulnormf2.c, 1191
main, 1192
check_mvp.c, 1192
main, 1193

check_mvp_call_matrix
check_mvpcall_matrix.c, 1194
check_mvpcall_matrix.c, 1194
check_mvp_call_matrix, 1194
main, 1194
check_norm.c, 1195
CHECKNORM, 1196
issAnorm2inv, 1197
main, 1196
rand50x50norm1, 1197
rand50x50norm2, 1197
rand50x50norm2inv, 1197
rand50x50normf, 1197
rand50x50norminf, 1198
randc50x50norm1, 1198
randc50x50norm2, 1198
randc50x50norm2inv, 1198
randc50x50normf, 1198
randc50x50norminf, 1198
check_one_value.c, 1198
f_one_value, 1198
f_one_valuec, 1198
main, 1198
VAL, 1198
check_orth.c, 1199
CHECKORTH, 1200
main, 1200
check_perm
check_vector_perm.c, 1246
check_perm.c, 1201
CHECKPERM, 1202
main, 1202
permute, 1203
check_perm_combine
check_vector_perm.c, 1246
check_perm_inplace
check_vector_perm.c, 1247
check_perm_split
check_vector_perm.c, 1248
check_pinv.c, 1203
check_pseudo1, 1204
check_pseudo2, 1205
check_pseudo3, 1206
check_pseudo4, 1206
main, 1207
check_precond
check_precond.c, 1209
check_precond.c, 1208
check_precond, 1209
main, 1210
check_proj_sym.c, 1211
CHECKPROSYM, 1212
main, 1212
matrix_AS, 1213
matrix_A, 1213
check_pseudo1
check_pinv.c, 1204
check_pseudo2
check_pinv.c, 1205
check_pseudo3
check_pinv.c, 1206
check_pseudo4
check_pinv.c, 1206
check_rank.c, 1213
main, 1214
check_real
check_cimport_c2d.c, 1051
check_real_imag_part.c, 1215
main, 1215
check_rowsub.c, 1215
CHECKROWSUB, 1216
main, 1216
check_scalem.c, 1217
main, 1218
check_schur.c, 1218
main, 1219
check_set_get_col_row.c, 1220
main, 1220
v_map_cpx, 1221
v_map_real, 1221
check_setelement.c, 1222
main, 1222
check_sign.c, 1223
main, 1224
check_so1_handles.c, 1225
CHECK_FUNCTION, 1226
main, 1226
check_so2_handles.c, 1227
CHECK_FUNCTION, 1228
main, 1228
check_solve
check_lapackqr.c, 1133
check_solvem
check_lapackqr.c, 1134
check_solver
check_itsolver.c, 1127
check_sub.c, 1229
CHECKSUB, 1230
main, 1230
check_svd.c, 1231
main, 1232
check_svd_complex.c, 1233
main, 1233
check_svd_econ.c, 1234
main, 1235
check_svd_econ_complex.c, 1236
main, 1237
check_trace.c, 1238
CHECKTRACE, 1238
main, 1239
trcpx, 1239
trreal, 1239
check_transpose.c, 1239
CHECKTRANSPOSE, 1240
main, 1240
matrix_AT, 1241

matrix_A, 1241
check_tridiag.c, 1241
main, 1242
check_trisolve.c, 1242
CHECK_ALL_STORAGETYPEM, 1243
CHECK_ALL_STORAGETYPE, 1243
CHECK_TRISOLVEM, 1244
CHECK_TRISOLVE, 1243
main, 1244
check_vector_perm.c, 1245
check_perm, 1246
check_perm_combine, 1246
check_perm_inplace, 1247
check_perm_split, 1248
main, 1248
permute, 1249
check_vectorio.c, 1249
main, 1250
child
mess_equation_st, 849
chol_clear
cholesky.c, 1251
chol_getL
cholesky.c, 1251
chol_getU
cholesky.c, 1252
chol_inverse
cholesky.c, 1252
chol_solve
cholesky.c, 1253
chol_solvem
cholesky.c, 1253
chol_type
direct.c, 1331
chol_type_mutex
direct.c, 1331
cholesky.c, 1250
chol_clear, 1251
chol_getL, 1251
chol_getU, 1252
chol_inverse, 1252
chol_solve, 1253
chol_solvem, 1253
cholesky_solver, 809
cpx, 809
n, 809
val, 810
val_cpx, 810
cholfactor.c, 1254
cholmod.c, 1254
cholmod_chol.c, 1254
cholmod_print_dense_all
check_cholmod.c, 1048
cholmod_print_sparse_all
check_cholmod.c, 1048
chooseorder
mess_bt_options_st, 824
chooseorder_aux
mess_bt_options_st, 824
cimport.c, 1255
clean_ccsrmat
iluk.c, 1463
clear
callback_equation.c, 1009
equation_glyap.c, 1353
equation_glyap_dae1.c, 1361
equation_glyap_dae2.c, 1375
equation_glyap_so1.c, 1388
equation_glyap_so2.c, 1399
equation_stable.c, 1417
equations.c, 1421
mess_direct_st, 838
mess_equation_st, 849
mess_equation_st::AINV_st, 802
mess_equation_st::AX_st, 805
mess_equation_st::ApEINV_st, 803
mess_equation_st::ApEX_st, 804
mess_equation_st::EINV_st, 812
mess_equation_st::EX_st, 813
mess_multidirect_st, 880
mess_precond_st, 899
clearRHS
mess_equation_st, 849
clearB
mess_equation_st, 849
clearC
mess_equation_st, 849
cmakefortrancompilerid
CMakeFortranCompilerId.F, 1261
col_compress.c, 1261
colops.c, 1262
color
__mess_plotExportData, 784
__mess_plotdata, 782
colptr
CCSRFmt, 809
mess_matrix_st, 872
colrowsums.c, 1262
cols
jobdata_st, 818
lapackqr_solver, 821
mess_direct_st, 838
mess_matrix_file_st, 870
mess_matrix_st, 872
mess_multidirect_st, 880
multi_umfpack, 925
multilu, 927
newlu_data, 933
colsX
_sylv_solver_d_st, 799
_sylv_solver_sd_st, 801
comp_mess_int_t
cuthill.c, 1318
compare_complex
irka_para_init.c, 1476
complex_matrix_data

check_colscale.c, 1063
complex_matrix_value
MESS_matrix, 863
compmess_int_t
csparse.c, 1279
Computation of Matrix Norms, 91
mess_matrix_condest, 92
mess_matrix_diffnorm, 93
mess_matrix_diffnormf, 94
mess_matrix_dynorm2, 95
mess_matrix_dynormf, 95
mess_matrix_dynormf2, 96
mess_matrix_indefinite_dynorm2, 97
mess_matrix_indefinite_dynormf, 98
mess_matrix_indefinite_dynormf2, 99
mess_matrix_mulnormf2, 100
mess_matrix_norm, 101
mess_matrix_norm1, 102
mess_matrix_norm2, 103
mess_matrix_norm2inv, 104
mess_matrix_normf, 105
mess_matrix_normf2, 106
mess_matrix_norminf, 107
mess_matrix_norminvest, 108
condest.c, 1263
INFINITY, 1263
config.h, 1263
LD_MULTIPLICITY, 1265
LOGLEVEL, 1265
MESS_ARPACK_LIBRARIES, 1265
MESS_BLA_VENDOR, 1265
MESS_BLAS_LIBRARIES, 1265
MESS_CMAKE_BUILD_TYPE, 1265
MESS_CMAKE_C_COMPILER, 1265
MESS_CMAKE_C_FLAGS_DEBUG, 1265
MESS_CMAKE_C_FLAGS_RELEASE, 1265
MESS_CMAKE_C_FLAGS, 1265
MESS_CMAKE_CXX_COMPILER, 1265
MESS_CMAKE_CXX_FLAGS_DEBUG, 1265
MESS_CMAKE_CXX_FLAGS_RELEASE, 1265
MESS_CMAKE_CXX_FLAGS, 1265
MESS_CMAKE_Fortran_COMPILER, 1265
MESS_CMAKE_Fortran_FLAGS_DEBUG, 1265
MESS_CMAKE_Fortran_FLAGS_RELEASE, 1265
MESS_CMAKE_Fortran_FLAGS, 1265
MESS_CMAKE_HOST_SYSTEM, 1265
MESS_CMAKE_VERSION, 1265
MESS_CONFIG_TIME, 1265
MESS_GIT_BRANCH, 1265
MESS_GIT_COMMIT_HASH, 1265
MESS_HAVE_AMD, 1266
MESS_HAVE_BZIP2, 1266
MESS_HAVE_CHOLMOD, 1266
MESS_HAVE_CLOCK_GETTIME, 1266
MESS_HAVE_COLAMD, 1266
MESS_HAVE_CSPARSE, 1266
MESS_HAVE_ISFINITE, 1266
MESS_HAVE_NEARBYINT, 1266
MESS_HAVE_PTRDIFF_T, 1266
MESS_HAVE_SIZE_T, 1266
MESS_HAVE_STRNCMP, 1266
MESS_HAVE_STRNDUP, 1266
MESS_HAVE_UMFPACK, 1266
MESS_HAVE_XPM, 1266
MESS_HAVE_ZLIB, 1266
MESS_INCLUDE_DIR, 1266
MESS_LAPACK_LIBRARIES, 1266
MESS_LIBS, 1266
MESS_MAJOR_VERSION, 1267
MESS_MINOR_VERSION, 1267
MESS_PATCH_VERSION, 1267
MESS_SUITESPARSE_INCLUDE_DIR, 1267
MESS_SUITESPARSE_LIBRARIES, 1267
MESS_SUPERLU_INCLUDE_DIR, 1267
MESS_SUPERLU_LIBRARIES, 1267
MESS_USE_OPENBLAS, 1267
MESS_VERSION, 1267
Control
multi_umfpack, 925
umfpack_control_solver, 946
ConvStringtoC
Interface to Python, 623
matrix.c, 1620
python/c_interface/conv/options.c, 1790
vector.c, 1967
converged
__mess_solver_status, 791
convert.c, 1267, 1274
converter
_equations_t, 795
copy.c, 1275
__ONE, 1275
Cp
mess_dynsys_st, 841
cpp_interface/MESS_matrix/MESS_matrix.cc
DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA, 1647
OCTMESS_GENERATOR, 1647
cpp_interface/MESS_options/MESS_options.cc
DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA, 1685
fieldnames, 1685
len_fieldnames, 1685
cpp_interface/MESS_status/MESS_status.cc
DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA, 1687
cpp_interface/MESS_vector/MESS_vector.cc
DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA, 1689
OCTMESS_GENERATOR, 1689
cpx
cholesky_solver, 809
lapack_solver, 819
lapackqr_solver, 821
multi_umfpack, 925
umfpack_control_solver, 946

cs_cl_lttsolve
csparse.c, 1279
csparse_chol.c, 1298
cs_cl_uttsolve
csparse.c, 1279
cs_dl_ipveccombine
csparse.c, 1279
cs_dl_ipvecsplit
csparse.c, 1280
csparse_chol.c, 1298
cs_dl_plsolve
csparse.c, 1280
cs_dl_pveccombine
csparse.c, 1280
csparse_chol.c, 1298
cs_dl_pvecsplit
csparse.c, 1280
cs_dlc_lsolve
csparse.c, 1280
csparse_chol.c, 1298
cs_dlc_ltsolve
csparse.c, 1280
csparse_chol.c, 1298
cs_dlc_usolve
csparse.c, 1280
cs_dlc_utsolve
csparse.c, 1280
cs_lsolve_analyse
csparse.c, 1280
csc.c, 1275
__gaxpy_csc_ge, 1275
__gaxpyh_csc_ge, 1276
__gaxpyt_csc_ge, 1277
cscutils
CMakeCache.txt, 1256
csparse.c, 1278
CSPARSE_ORDER, 1279
CSPARSE_TOL, 1279
compmess_int_t, 1279
cs_cl_lttsolve, 1279
cs_cl_uttsolve, 1279
cs_dl_ipveccombine, 1279
cs_dl_ipvecsplit, 1280
cs_dl_plsolve, 1280
cs_dl_pveccombine, 1280
cs_dl_pvecsplit, 1280
cs_dlc_lsolve, 1280
cs_dlc_ltsolve, 1280
cs_dlc_usolve, 1280
cs_dlc_utsolve, 1280
cs_lsolve_analyse, 1280
csparse_clear, 1280
csparse_clear_complex, 1281
csparse_getL_complex, 1282
csparse_getU_complex, 1285
csparse_getL, 1281
csparse_getpermp, 1283
csparse_getpermp_complex, 1283
csparse_getpermq, 1284
csparse_getpermq_complex, 1284
csparse_getU, 1284
csparse_inverse, 1286
csparse_inverse_complex, 1287
csparse_psolve, 1287
csparse_solve, 1288
csparse_solve_complex, 1288
csparse_solveh_complex, 1289
csparse_solvem, 1290
csparse_solvem_complex, 1290
csparse_solvemh_complex, 1291
csparse_solvemt, 1291
csparse_solvemt_complex, 1293
csparse_solvet, 1294
csparse_solvet_complex, 1295
perm_inv, 1296
csparse_chol.c, 1296
CSPARSE_ORDER, 1298
cs_cl_lttsolve, 1298
cs_dl_ipvecsplit, 1298
cs_dl_pveccombine, 1298
cs_dlc_lsolve, 1298
cs_dlc_ltsolve, 1298
csparse_clear, 1298
csparse_clear_complex, 1298
csparse_getL_complex, 1299
csparse_getU_complex, 1302
csparse_getL, 1298
csparse_getpermp, 1300
csparse_getpermp_complex, 1301
csparse_getU, 1301
csparse_inverse, 1303
csparse_inverse_complex, 1303
csparse_solve, 1304
csparse_solve_complex, 1305
csparse_solveh_complex, 1306
csparse_solvem, 1306
csparse_solvem_complex, 1307
csparse_solvemh_complex, 1307
csparse_solvemt, 1308
csparse_solvemt_complex, 1308
csparse_solvet, 1309
csparse_solvet_complex, 1309
perm_inv, 1310
csparse_clear
csparse.c, 1280
csparse_chol.c, 1298
csparse_clear_complex
csparse.c, 1281
csparse_chol.c, 1298
csparse_getL_complex
csparse.c, 1282
csparse_chol.c, 1299
csparse_getU_complex
csparse.c, 1285
csparse_chol.c, 1302
csparse_getL

csparse.c, 1281
csparse_chol.c, 1298
csparse_getpermp
csparse.c, 1283
csparse_chol.c, 1300
csparse_getpermp_complex
csparse.c, 1283
csparse_chol.c, 1301
csparse_getpermq
csparse.c, 1284
csparse_getpermq_complex
csparse.c, 1284
csparse_getU
csparse.c, 1284
csparse_chol.c, 1301
csparse_inverse
csparse.c, 1286
csparse_chol.c, 1303
csparse_inverse_complex
csparse.c, 1287
csparse_chol.c, 1303
csparse_psolve
csparse.c, 1287
csparse_solve
csparse.c, 1288
csparse_chol.c, 1304
csparse_solve_complex
csparse.c, 1288
csparse_chol.c, 1305
csparse_solveh_complex
csparse.c, 1289
csparse_chol.c, 1306
csparse_solvem
csparse.c, 1290
csparse_chol.c, 1306
csparse_solvem_complex
csparse.c, 1290
csparse_chol.c, 1307
csparse_solvemh_complex
csparse.c, 1291
csparse_chol.c, 1307
csparse_solvemt
csparse.c, 1291
csparse_chol.c, 1308
csparse_solvemt_complex
csparse.c, 1293
csparse_chol.c, 1308
csparse_solver, 810
dim, 811
Llevels, 811
LU, 811
SLU, 811
csparse_solver_complex, 811
dim, 811
LU, 811
SLU, 812
csparse_solvet
csparse.c, 1294
csparse_chol.c, 1309
csparse_solvet_complex
csparse.c, 1295
csparse_chol.c, 1309
csr.c, 1310
__gaxpy_csr_ge, 1311
__gaxpy_csr_sym, 1311
__gaxpyh_csr_ge, 1312
__gaxpyt_csr_ge, 1312
cur_size
mess_threadpool_st, 911
cuthill.c, 1313
bandw_gipost, 1315
bandw_level_struct, 1315
bandw_mark_connected, 1316
bandw_number_adj_vert, 1317
bandw_red_perm, 1318
comp_mess_int_t, 1318
FAIL, 1314
FUNCTION_FAILURE_HANDLE, 1314
int_list_alloc, 1319
int_list_free, 1319
SUCCESS, 1314
TRY_MALLOC, 1314
TRY_REALLOC, 1314
Cv
mess_dynsys_st, 841

D
__glyap_so1, 771
__glyap_so2, 773
__iluk, 776
blas_defs.h, 999
d
pair_mess_int_t2_complex, 936
pair_mess_int_t2_double, 937
pair_mess_int_t_complex, 937
pair_mess_int_t_double, 938
D_FILE_OFFSET_BITS
lib/CMakeFiles/mess.dir/link.txt, 1503
tests/direct/CMakeFiles/check_backslash.↩ dir/link.txt, 1504
tests/direct/CMakeFiles/check_det.dir/link.txt, 1504
tests/direct/CMakeFiles/check_dgeqp3.dir/link.txt, 1505
tests/direct/CMakeFiles/check_direct_care.↩ dir/link.txt, 1505
tests/direct/CMakeFiles/check_getLU.dir/link.txt, 1506
tests/direct/CMakeFiles/check_hessenberg.↩ dir/link.txt, 1506
tests/direct/CMakeFiles/check_lapackqr.dir/link.txt, 1507
tests/direct/CMakeFiles/check_pinv.dir/link.txt, 1507
tests/direct/CMakeFiles/check_trisolve.dir/link.txt, 1508
tests/direct/CMakeFiles/glyap_test.dir/link.txt, 1508

tests/direct/CMakeFiles/gstein_test.dir/link.txt, 1508
tests/direct/CMakeFiles/inverse.dir/link.txt, 1509
tests/direct/CMakeFiles/linsolve.dir/link.txt, 1509
tests/direct/CMakeFiles/linsolve_multi.dir/link.txt, 1509
tests/direct/CMakeFiles/linsolvem.dir/link.txt, 1510
tests/direct/CMakeFiles/linsolvem_multi.dir/link.txt, 1510
tests/direct/CMakeFiles/sylvester_dense.dir/link.↩ txt, 1511
tests/direct/CMakeFiles/sylvester_sparsedense.↩ dir/link.txt, 1511
tests/eigen/CMakeFiles/check_arnoldi_template.↩ dir/link.txt, 1512
tests/eigen/CMakeFiles/check_arnoldi_template↩ _nrm.dir/link.txt, 1512
tests/eigen/CMakeFiles/check_eig.dir/link.txt, 1513
tests/eigen/CMakeFiles/check_eig2.dir/link.txt, 1513
tests/eigen/CMakeFiles/check_eig_complex.↩ dir/link.txt, 1513
tests/eigen/CMakeFiles/check_eig_hessenberg_↩ schur.dir/link.txt, 1514
tests/eigen/CMakeFiles/check_eig_hessenberg_↩ schur_largest_ev.dir/link.txt, 1514
tests/eigen/CMakeFiles/check_gsign.dir/link.txt, 1515
tests/eigen/CMakeFiles/check_schur.dir/link.txt, 1515
tests/eigen/CMakeFiles/check_sign.dir/link.txt, 1516
tests/eigen/CMakeFiles/check_svd.dir/link.txt, 1516
tests/eigen/CMakeFiles/check_svd_complex.↩ dir/link.txt, 1516
tests/eigen/CMakeFiles/check_svd_econ.↩ dir/link.txt, 1517
tests/eigen/CMakeFiles/check_svd_econ_↩ complex.dir/link.txt, 1517
tests/formats/CMakeFiles/check_cholmod.↩ dir/link.txt, 1518
tests/formats/CMakeFiles/check_cimport.dir/link.↩ txt, 1518
tests/formats/CMakeFiles/check_cimport_c2d.↩ dir/link.txt, 1519
tests/formats/CMakeFiles/check_convert.dir/link.↩ txt, 1519
tests/formats/CMakeFiles/check_mvpcall_↩ matrix.dir/link.txt, 1520
tests/formats/CMakeFiles/check_perm.dir/link.txt, 1520
tests/formats/CMakeFiles/check_real_imag_↩ part.dir/link.txt, 1521
tests/io/CMakeFiles/check_fputs_block.dir/link.txt, 1521
tests/io/CMakeFiles/check_matrixio.dir/link.txt, 1522
tests/io/CMakeFiles/check_vectorio.dir/link.txt, 1522
tests/itsolver/CMakeFiles/check_itsolver.dir/link.txt, 1523
tests/itsolver/CMakeFiles/check_precond.↩ dir/link.txt, 1523
tests/lrcf_adi/CMakeFiles/check_ber_sign_fac.↩ dir/link.txt, 1524
tests/lrcf_adi/CMakeFiles/check_dae2_handles_↩ adi.dir/link.txt, 1524
tests/lrcf_adi/CMakeFiles/check_dae2_handles_↩ para.dir/link.txt, 1525
tests/lrcf_adi/CMakeFiles/check_dense_nm.↩ dir/link.txt, 1525
tests/lrcf_adi/CMakeFiles/check_dense_nm_↩ tridiag.dir/link.txt, 1526
tests/lrcf_adi/CMakeFiles/check_linesearch.↩ dir/link.txt, 1526
tests/lrcf_adi/CMakeFiles/check_lradi_dae1.↩ dir/link.txt, 1527
tests/lrcf_adi/CMakeFiles/check_lradi_dae2.↩ dir/link.txt, 1527
tests/lrcf_adi/CMakeFiles/check_lradi_glyap.↩ dir/link.txt, 1528
tests/lrcf_adi/CMakeFiles/check_lradi_so.dir/link.↩ txt, 1528
tests/lrcf_adi/CMakeFiles/check_lradi_unsym.↩ dir/link.txt, 1529
tests/lrcf_adi/CMakeFiles/check_lrnm_dae1.↩ dir/link.txt, 1529
tests/lrcf_adi/CMakeFiles/check_lrnm_dae2.↩ dir/link.txt, 1530
tests/lrcf_adi/CMakeFiles/check_lrnm_g.dir/link.txt, 1530
tests/lrcf_adi/CMakeFiles/check_lrnm_so.↩ dir/link.txt, 1531
tests/lrcf_adi/CMakeFiles/check_lrnm_unsym.↩ dir/link.txt, 1531
tests/lrcf_adi/CMakeFiles/check_lyap_sign_fac.↩ dir/link.txt, 1532
tests/lrcf_adi/CMakeFiles/check_so1_handles.↩ dir/link.txt, 1532
tests/lrcf_adi/CMakeFiles/check_so2_handles.↩ dir/link.txt, 1533
tests/matrix/CMakeFiles/check_add.dir/link.txt, 1533
tests/matrix/CMakeFiles/check_addcol2p.↩ dir/link.txt, 1534
tests/matrix/CMakeFiles/check_any.dir/link.txt, 1534
tests/matrix/CMakeFiles/check_bandwidth.↩ dir/link.txt, 1535
tests/matrix/CMakeFiles/check_biorth.dir/link.txt, 1535
tests/matrix/CMakeFiles/check_cat.dir/link.txt, 1536
tests/matrix/CMakeFiles/check_colaxpy2.dir/link.↩ txt, 1536

tests/matrix/CMakeFiles/check_coldot.dir/link.txt, 1537
tests/matrix/CMakeFiles/check_coldotE.dir/link.txt, 1538
tests/matrix/CMakeFiles/check_coldotc.dir/link.txt, 1537
tests/matrix/CMakeFiles/check_coldotcE.dir/link.↩txt, 1538
tests/matrix/CMakeFiles/check_colnorm.dir/link.txt, 1539
tests/matrix/CMakeFiles/check_colscale.dir/link.txt, 1539
tests/matrix/CMakeFiles/check_colsub.dir/link.txt, 1540
tests/matrix/CMakeFiles/check_colvecaxpy.↩dir/link.txt, 1540
tests/matrix/CMakeFiles/check_colvecdot.↩dir/link.txt, 1541
tests/matrix/CMakeFiles/check_colvecdotc.↩dir/link.txt, 1541
tests/matrix/CMakeFiles/check_condest.dir/link.txt, 1542
tests/matrix/CMakeFiles/check_decomp.dir/link.txt, 1542
tests/matrix/CMakeFiles/check_diag.dir/link.txt, 1543
tests/matrix/CMakeFiles/check_dynorm2.dir/link.↩txt, 1543
tests/matrix/CMakeFiles/check_dynorm2t.↩dir/link.txt, 1544
tests/matrix/CMakeFiles/check_dynorm_indefinite2.↩dir/link.txt, 1544
tests/matrix/CMakeFiles/check_dynorm_indefinitef.↩dir/link.txt, 1545
tests/matrix/CMakeFiles/check_dynormf.dir/link.txt, 1545
tests/matrix/CMakeFiles/check_dynormft.dir/link.↩txt, 1546
tests/matrix/CMakeFiles/check_eliminate_zeros.↩dir/link.txt, 1546
tests/matrix/CMakeFiles/check_eps.dir/link.txt, 1547
tests/matrix/CMakeFiles/check_fro_inn.dir/link.txt, 1547
tests/matrix/CMakeFiles/check_gaxpy.dir/link.txt, 1548
tests/matrix/CMakeFiles/check_getelement.↩dir/link.txt, 1548
tests/matrix/CMakeFiles/check_matrix_biorth.↩dir/link.txt, 1549
tests/matrix/CMakeFiles/check_matrix_colaxpy.↩dir/link.txt, 1549
tests/matrix/CMakeFiles/check_matrix_gbiorth.↩dir/link.txt, 1550
tests/matrix/CMakeFiles/check_matrix_null.↩dir/link.txt, 1550
tests/matrix/CMakeFiles/check_matrix_sum.↩dir/link.txt, 1551
tests/matrix/CMakeFiles/check_mgs.dir/link.txt, 1551
tests/matrix/CMakeFiles/check_mgs_inplace.↩dir/link.txt, 1552
tests/matrix/CMakeFiles/check_mgsadd.dir/link.txt, 1552
tests/matrix/CMakeFiles/check_mul.dir/link.txt, 1553
tests/matrix/CMakeFiles/check_mul_small.↩dir/link.txt, 1553
tests/matrix/CMakeFiles/check_mul_tall.dir/link.txt, 1554
tests/matrix/CMakeFiles/check_mulnormf2.↩dir/link.txt, 1554
tests/matrix/CMakeFiles/check_mvp.dir/link.txt, 1555
tests/matrix/CMakeFiles/check_norm.dir/link.txt, 1555
tests/matrix/CMakeFiles/check_one_value.↩dir/link.txt, 1555
tests/matrix/CMakeFiles/check_orth.dir/link.txt, 1556
tests/matrix/CMakeFiles/check_proj_sym.↩dir/link.txt, 1556
tests/matrix/CMakeFiles/check_rank.dir/link.txt, 1556
tests/matrix/CMakeFiles/check_rowsub.dir/link.txt, 1557
tests/matrix/CMakeFiles/check_scalem.dir/link.txt, 1557
tests/matrix/CMakeFiles/check_set_get_col_↩row.dir/link.txt, 1558
tests/matrix/CMakeFiles/check_setelement.↩dir/link.txt, 1558
tests/matrix/CMakeFiles/check_sub.dir/link.txt, 1559
tests/matrix/CMakeFiles/check_trace.dir/link.txt, 1559
tests/matrix/CMakeFiles/check_transpose.↩dir/link.txt, 1559
tests/matrix/CMakeFiles/check_tridiag.dir/link.txt, 1560
tests/misc/CMakeFiles/check_freelist.dir/link.txt, 1560
tutorials/bt/CMakeFiles/tutorial_bt.dir/link.txt, 1561
tutorials/bt/CMakeFiles/tutorial_bt_gglrsrm.↩dir/link.txt, 1561
tutorials/bt/CMakeFiles/tutorial_bt_gslrsrm.↩dir/link.txt, 1562
tutorials/bt/CMakeFiles/tutorial_bt_lrsrm.dir/link.txt, 1562
tutorials/bt/CMakeFiles/tutorial_bt_lrsrm_somor.↩dir/link.txt, 1563
tutorials/bt/CMakeFiles/tutorial_bt_lrsrm_somor↩_so.dir/link.txt, 1563
tutorials/direct/CMakeFiles/tutorial_backslash.↩dir/link.txt, 1564
tutorials/direct/CMakeFiles/tutorial_direct_↩

solvers.dir/link.txt, 1564
tutorials/direct/CMakeFiles/tutorial_qr.dir/link.txt, 1565
tutorials/direct/CMakeFiles/tutorial_sylvester.↩ dir/link.txt, 1565
tutorials/eig/CMakeFiles/tutorial_eig.dir/link.txt, 1565
tutorials/eig/CMakeFiles/tutorial_eigs.dir/link.txt, 1566
tutorials/eig/CMakeFiles/tutorial_gschur.dir/link.txt, 1566
tutorials/eig/CMakeFiles/tutorial_schur.dir/link.txt, 1567
tutorials/eig/CMakeFiles/tutorial_sign.dir/link.txt, 1567
tutorials/h2/CMakeFiles/tutorial_h2norm.dir/link.txt, 1568
tutorials/h2/CMakeFiles/tutorial_irka_biorth.↩ dir/link.txt, 1568
tutorials/h2/CMakeFiles/tutorial_tsia.dir/link.txt, 1569
tutorials/io/CMakeFiles/tutorial_vector_read.↩ dir/link.txt, 1569
tutorials/itsolver/CMakeFiles/tutorial_it_unsym.↩ dir/link.txt, 1570
tutorials/lrcf_adi/CMakeFiles/tutorial_care.↩ dir/link.txt, 1570
tutorials/lrcf_adi/CMakeFiles/tutorial_lradi.↩ dir/link.txt, 1570
tutorials/lrcf_adi/CMakeFiles/tutorial_lrnm-g.↩ dir/link.txt, 1571
tutorials/lrcf_adi/CMakeFiles/tutorial_lrnm.↩ dir/link.txt, 1571
tutorials/lrcf_adi/CMakeFiles/tutorial_lyap.↩ dir/link.txt, 1571
tutorials/matgen/CMakeFiles/tutorial_gen_fdm_↩ cd.dir/link.txt, 1572
tutorials/matgen/CMakeFiles/tutorial_gen_↩ laplace.dir/link.txt, 1572
tutorials/matrix/CMakeFiles/tutorial_matrix_↩ operations.dir/link.txt, 1573
tutorials/matrix/CMakeFiles/tutorial_norms.↩ dir/link.txt, 1573
tutorials/plot/CMakeFiles/tutorial_plot.dir/link.txt, 1574
tutorials/plot/CMakeFiles/tutorial_spy.dir/link.txt, 1574
DCPX_T.c, 1321
info_size, 1321
main, 1321
SIZE, 1321
DEC_1
CMakeFortranCompilerId.F, 1261
DEC_2
CMakeFortranCompilerId.F, 1261
DEC_3
CMakeFortranCompilerId.F, 1261
DEC_4
CMakeFortranCompilerId.F, 1261
DEC_5
CMakeFortranCompilerId.F, 1261
DEC_6
CMakeFortranCompilerId.F, 1261
DEC_7
CMakeFortranCompilerId.F, 1261
DEC_8
CMakeFortranCompilerId.F, 1261
DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA
MESS_datatype_t, 828
MESS_direct, 832
MESS_direct_cholpackage_t, 834
MESS_direct_lupackage_t, 836
MESS_equation, 847
MESS_equation_t, 851
MESS_matrix, 869
MESS_memusage_t, 875
MESS_multidirect, 879
MESS_multidirect_t, 884
MESS_operation_t, 887
MESS_options, 892
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_status, 905
MESS_storage_t, 910
MESS_vector, 921
DEFBINOP
MESS_enums.cc, 1639, 1640
MESS_matrix_install.cc, 1665–1670
MESS_vector_install.cc, 1702–1706
DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA
cpp_interface/MESS_matrix/MESS_matrix.cc, 1647
cpp_interface/MESS_options/MESS_options.cc, 1685
cpp_interface/MESS_status/MESS_status.cc, 1687
cpp_interface/MESS_vector/MESS_vector.cc, 1689
MESS_direct.cc, 1633
MESS_enums.cc, 1640
MESS_equation.cc, 1643
MESS_multidirect.cc, 1684
DEFUN_DLD
MESS_enum.cc, 1635, 1636
MESS_misc.cc, 1682–1684
MESS_to_oct.cc, 1689
octmess/src/MESS_matrix.cc, 1648
octmess/src/MESS_options.cc, 1686
octmess/src/MESS_status.cc, 1688
octmess/src/MESS_vector.cc, 1690
DEFUNOP
MESS_matrix_install.cc, 1671, 1672
MESS_vector_install.cc, 1706
DEPRECATED
General Type Definitions and Macros, 45
DEC

CMakeCCompilerId.c, 1257
CMakeCXXCompilerId.cpp, 1259
DIFF
blas_defs.h, 999
DTCHECK
multidirect.c, 1742
multilu_adavanced.c, 1744
DTINIT
multidirect.c, 1742
multilu_adavanced.c, 1744
DWORK
blas_defs.h, 999
dae1.c, 1320
dae2.c, 1320
DartConfiguration.tcl, 1321
data
__matlab_equation, 779
mess_direct_st, 838
mess_multidirect_st, 880
mess_mvpcall_st, 885
mess_precond_st, 899
mess_threadpooljob_st, 913
Data Structures and Related Functions, 188
mess_vector, 189
mess_vector_alloc, 189
mess_vector_clear, 190
mess_vector_complex_from_parts, 190
mess_vector_copy, 190
mess_vector_copy_tocomplex, 191
mess_vector_from_farray, 192
mess_vector_from_lapack, 192
mess_vector_imagpart, 193
mess_vector_init, 193
mess_vector_realpart, 194
mess_vector_reset, 194
mess_vector_st, 189
mess_vector_tocomplex, 195
mess_vector_toreal, 195
mess_vector_toreal_nowarn, 195
mess_vector_totype, 196
Data Structures, Types and Macros for Handling Matrices, 177
MESS_COORD, 181
MESS_CSC, 181
MESS_CSR, 181
MESS_DENSE, 181
MESS_GENERAL, 181
MESS_HERMITIAN, 181
MESS_IS_COORD, 178
MESS_IS_CSC, 178
MESS_IS_CSR, 178
MESS_IS_DENSE, 178
MESS_IS_GENERAL, 178
MESS_IS_HERMITIAN, 178
MESS_IS_SKEWSYMMETRIC, 179
MESS_IS_SPARSE, 179
MESS_IS_SYMMETRIC, 179
MESS_IS_UNSYMMETRIC, 179
MESS_MATRIX_CHECKFORMAT, 179
MESS_MATRIX_FILE_MAGIC_CHECK, 180
MESS_MATRIX_FILE_MAGIC_SET, 180
MESS_MATRIX_FILE_MAGIC, 180
MESS_MATRIX_FILE_VERSION, 180
MESS_MATRIX_RESET, 180
MESS_SKEWSYMMETRIC, 181
MESS_SYMMETRIC, 181
MESS_UNKNOWN, 181
mess_matrix, 180
mess_matrix_clear, 181
mess_matrix_file, 181
mess_matrix_init, 182
mess_matrix_st, 181
mess_storage_t, 181
mess_symmetry_t, 181
data_ABCD
check_cat.c, 1045
data_ABC
check_cat.c, 1045
data_ABD
check_cat.c, 1046
data_ACD
check_cat.c, 1046
data_AB
check_cat.c, 1045
data_AC
check_cat.c, 1046
data_AD
check_cat.c, 1046
data_BCD
check_cat.c, 1046
data_BC
check_cat.c, 1046
data_A
check_cat.c, 1045
data_B
check_cat.c, 1046
data_C
check_cat.c, 1046
data_D
check_cat.c, 1047
data_type
__iluk, 776
CCSRFmt, 809
hessenberg_solver, 814
mess_direct_st, 838
mess_krylov_arnoldi_st, 858
mess_matrix_file_st, 870
mess_matrix_st, 872
mess_multidirect_st, 880
mess_mvpcall_st, 885
mess_vector_st, 922
newlu_data, 933
datatype
VWjob, 947
datatypes
multi_umfpack, 925

multilu, 927
decomp.c, 1322
deepcopy
MESS_matrix, 863
MESS_options, 890
MESS_status, 904
MESS_vector, 918
default_rom_chooser.c, 1322
Define own Matrix Generators, 534
mess_equation_A_apply, 535
mess_equation_A_apply_vector, 536
mess_equation_A_post, 537
mess_equation_A_pre, 537
mess_equation_ApE_apply, 538
mess_equation_ApE_apply_vector, 538
mess_equation_ApE_post, 539
mess_equation_ApE_pre, 539
mess_equation_ApEs_apply, 540
mess_equation_ApEs_apply_vector, 540
mess_equation_ApEs_post, 541
mess_equation_ApEs_pre, 542
mess_equation_As_apply, 542
mess_equation_As_apply_vector, 543
mess_equation_As_post, 544
mess_equation_As_pre, 544
mess_equation_E_apply, 545
mess_equation_E_apply_vector, 546
mess_equation_E_post, 547
mess_equation_E_pre, 547
mess_equation_Es_apply, 548
mess_equation_Es_apply_vector, 548
mess_equation_Es_post, 549
mess_equation_Es_pre, 550
mess_equation_dim, 545
mess_equation_has_ApEs, 551
mess_equation_has_ApE, 550
mess_equation_has_As, 551
mess_equation_has_Es, 552
mess_equation_has_A, 550
mess_equation_has_E, 551
delete_atom
__mess_plotter, 786
delta
__glyap_dae2, 768
Dense Eigenvalue Solvers, 425
mess_eigen_eig, 426
mess_eigen_eigg, 427
mess_eigen_gschur, 428
mess_eigen_gschur_complex, 431
mess_eigen_gsign, 432
mess_eigen_hessenberg_abs_largest, 433
mess_eigen_hessenberg_schur, 434
mess_eigen_schur, 435
mess_eigen_schur_complex, 436
mess_eigen_schur_complex_post, 437
mess_eigen_schur_complex_zgees, 438
mess_eigen_schur_to_evd, 439
mess_eigen_sign, 439
mess_eigen_svd, 440
mess_eigen_svd_econ, 440
dense.c, 1322
__gaxpy_dense, 1322
__gaxpyh_dense, 1323
__gaxpyt_dense, 1324
dense_nm.c, 1325
ABSRES_TOL, 1325
LINESEARCH_BREAKDOWN, 1325
MAXIT, 1325
MIN_LINESEARCH_PARA, 1325
RELRES_TOL, 1325
dense_nm_gmpare.c, 1325
det
mess_direct_st, 838
detc
mess_direct_st, 838
dfs.c, 1326
FLIP, 1326
MARKED, 1326
MARK, 1326
UNFLIP, 1326
dgeqp3.c, 1326
diag
blas_defs.h, 999
diag.c, 1327
dim
csparse_solver, 811
csparse_solver_complex, 811
mess_dynsys_st, 841
mess_equation_st, 849
mess_krylov_arnoldi_st, 858
mess_mvpcall_st, 885
mess_vector_st, 922
umfpack_control_solver, 946
dims
MESS_matrix, 864
MESS_vector, 919
Direct Solvers for Linear Systems, 234, 247, 718
MESS_DIRECT_BANDED_LU, 248
MESS_DIRECT_CHOLMOD_CHOLESKY, 248
MESS_DIRECT_CSPARSE_CHOLESKY, 248
MESS_DIRECT_CSPARSE_LU, 248
MESS_DIRECT_DEFAULT_CHOLESKY, 248
MESS_DIRECT_DEFAULT_LU, 248
MESS_DIRECT_LAPACK_CHOLESKY, 248
MESS_DIRECT_LAPACK_LU, 248
MESS_DIRECT_MKLPARDISO_LU, 248
MESS_DIRECT_SPARSE_LU, 248
MESS_DIRECT_SUPERLU_LU, 248
MESS_DIRECT_UMFPACK_LU, 248
mess_direct_chol, 248
mess_direct_chol_select, 249
mess_direct_cholpackage_t, 248
mess_direct_create_banded, 250
mess_direct_create_bicgstab, 251
mess_direct_create_cholesky, 252
mess_direct_create_csparse_cholesky, 253

mess_direct_create_csparse_lu, 254
mess_direct_create_hessenberg_lu, 256
mess_direct_create_lapack_lu, 257
mess_direct_create_lapack_qr, 258
mess_direct_create_sparse_lu, 259
mess_direct_create_umfpack, 260
mess_direct_create_umfpack_control, 261
mess_direct_lu, 262
mess_direct_lu_select, 263
mess_direct_lupackage_t, 248
Direct Solvers for Matrix Equations, 288
mess_direct_care, 288
mess_direct_care_sign, 290
mess_direct_create_generalized_lyapunov, 290
mess_direct_create_generalized_lyapunovchol, 291
mess_direct_create_generalized_stein, 292
mess_direct_create_sylvester_dense, 293
mess_direct_create_sylvester_sparsedense, 294
direct.c, 1330
chol_type, 1331
chol_type_mutex, 1331
lu_type, 1331
lu_type_mutex, 1331
direct.h, 1331
direct/singlesolver/bicgstab.c
bicgstab_clear, 986
bicgstab_solve, 986
bicgstab_solvem, 987
ILU_LEVEL, 986
direct_chol_select.c, 1336
direct_select.c, 1336
display
__mess_plotter, 786
dn
wachspress.c, 1975
done
__mess_plotter, 786
th_done_entry, 945
doxy_main.c, 1337
drawaxis
x11init.c, 1978
dtimer
mess_timer_st, 914
dupl.c, 1337
DX
blas_defs.h, 999
Dx
__iluk, 776
DY
blas_defs.h, 999
Dynamical Systems, 350
dynorm.c, 1337
dynsys.h, 1338
dynsysinit.c, 1339

E
__lyap, 778
_sylv_solver_sd_st, 801
blas_defs.h, 999
mess_dynsys_st, 841
mvpdatag, 931
E11
__glyap_dae1, 765
tests/data/bips98_606/source.txt, 1855
tests/data/siso_ww_vref_6405/source.txt, 1856
E11sol
__glyap_dae1, 765
EIGSRC
blas_defs.h, 999
EINV_apply
callback_equation.c, 1009
equation_glyap.c, 1354
equation_glyap_dae1.c, 1362
equation_glyap_so1.c, 1389
equation_glyap_so2.c, 1399
equation_stable.c, 1417
EINV_apply_shifts
equation_glyap_dae2.c, 1376
EINV_clear
callback_equation.c, 1009
equation_glyap.c, 1354
equation_stable.c, 1417
EINV_generate
callback_equation.c, 1009
equation_glyap.c, 1354
equation_stable.c, 1417
EINV
mess_equation_st, 849
ENDTIME
irka_biorth.c, 1474
EX_apply
callback_equation.c, 1009
equation_glyap.c, 1354
equation_glyap_dae1.c, 1363
equation_glyap_dae2.c, 1377
equation_glyap_so1.c, 1389
equation_glyap_so2.c, 1400
equation_stable.c, 1417
EX_apply_shifts
equation_glyap_dae2.c, 1377
EX_clear
callback_equation.c, 1009
equation_stable.c, 1417
EX_generate
callback_equation.c, 1009
equation_stable.c, 1417
Easy Frontends to Main Functions, 585
mess_care, 585
mess_glyap, 587
mess_gstein, 588
mess_lyap, 590
mess_tglyap, 591
mess_tgstein, 592
easyfrontend.h, 1340
editing
__mess_plotter, 786

Ehat
_glyap3_st, 796
_sylv_solver_d_st, 799
eig.c, 1340
eig_hessenberg.c, 1341
Eigenvalue and Related Problems, 424, 720
eigenvalue.h, 1341
el
int_list, 816
ellipK
wachspress.c, 1975
elnmax
int_list, 816
enum_conversions.c, 1343
MEXMESS_GENERATOR, 1344
eps.c, 1344, 1345
eqn
parameter_mvp_call, 939
res2_data_st, 942
res2nm_data_st, 944
eqn_conv_lyap
Interface to Python, 625
Interfaces_matlab, 730
eqn_conv_lyap_dae1
Interface to Python, 625
Interfaces_matlab, 730
eqn_conv_lyap_dae2
Interface to Python, 626
Interfaces_matlab, 731
eqn_conv_lyap_so1
Interface to Python, 627
Interfaces_matlab, 732
eqn_conv_lyap_so2
Interface to Python, 628
Interfaces_matlab, 733
eqn_conv_ric
Interfaces_matlab, 734
eqn_conv_ric_dae1
Interfaces_matlab, 735
eqn_conv_ric_dae2
Interfaces_matlab, 736
eqn_conv_ric_so1
Interfaces_matlab, 737
eqn_conv_ric_so2
Interfaces_matlab, 738
eqn_conv_riccati
Interface to Python, 629
eqn_conv_riccati_dae1
Interface to Python, 630
eqn_conv_riccati_dae2
Interface to Python, 631
eqn_conv_riccati_so1
Interface to Python, 632
eqn_conv_riccati_so2
Interface to Python, 633
eqn_converter
equations.c, 1420
mess_equation_from_mexmess.c, 1644
eqn_type
_glyap3_st, 796
mess_equation_st, 849
eqns
equations.c, 1422
mess_equation_from_mexmess.c, 1644
mess_freelist_st, 853
equal.c, 1345
mat_diff_mvp, 1345
Equation Generators, 502
mess_equation_glyap, 504
mess_equation_glyap_dae1, 505
mess_equation_glyap_dae2, 507
mess_equation_glyap_so1, 509
mess_equation_glyap_so2, 511
mess_equation_griccati, 513
mess_equation_griccati_dae1, 515
mess_equation_griccati_dae2, 517
mess_equation_griccati_so1, 520
mess_equation_griccati_so2, 522
mess_equation_lyap, 524
mess_equation_riccati, 526
mess_equation_stable, 528
mess_equation_std_lyap, 530
mess_equation_std_riccati, 531
equation.c, 1347
equation_apply.c, 1347
equation_glyap.c, 1349
AINV_apply, 1350
AINV_clear, 1350
AINV_generate, 1350
AX_apply, 1353
ApEINV_apply, 1350
ApEINV_clear, 1351
ApEINV_generate, 1351
ApEX_apply, 1351
ApEX_clear, 1352
ApEX_generate, 1352
clear, 1353
EINV_apply, 1354
EINV_clear, 1354
EINV_generate, 1354
EX_apply, 1354
parameter, 1355
equation_glyap_dae1.c, 1356
AINV_apply, 1357
AX_apply, 1360
ApEINV2_apply, 1358
ApEINV2_clear, 1359
ApEINV2_generate, 1360
clear, 1361
EINV_apply, 1362
EX_apply, 1363
parameter, 1364
equation_glyap_dae2.c, 1365
AINV_apply_shifts, 1367
AX_apply_N, 1373
AX_apply_shifts, 1373

AX_apply_T, 1374
ApEINV_apply_memory0, 1367
ApEINV_apply_memory1, 1369
ApEINV_clear_memory0, 1370
ApEINV_clear_memory1, 1371
ApEINV_generate_memory0, 1371
ApEINV_generate_memory1, 1372
clear, 1375
EINV_apply_shifts, 1376
EX_apply, 1377
EX_apply_shifts, 1377
init_rhs, 1378
mess_matrix_applyPI_dae2, 1379
parameter, 1380
equation_glyap_so1.c, 1382
AINV_apply, 1382
AX_apply, 1387
ApEINV_apply, 1383
ApEINV_clear, 1385
ApEINV_generate, 1386
clear, 1388
EINV_apply, 1389
EX_apply, 1389
parameter, 1390
equation_glyap_so2.c, 1392
AINV_apply, 1393
AX_apply, 1398
ApEINV_apply, 1394
ApEINV_clear, 1396
ApEINV_generate, 1397
clear, 1399
EINV_apply, 1399
EX_apply, 1400
parameter, 1401
equation_lyap.c, 1403
__lyap_AINV_apply, 1405
__lyap_AINV_clear, 1406
__lyap_AINV_generate, 1406
__lyap_A_multiplym, 1404
__lyap_ApEX_apply, 1409
__lyap_ApEX_clear, 1410
__lyap_ApEX_generate, 1411
__lyap_ApE_clear, 1407
__lyap_ApE_generate, 1408
__lyap_ApE_solve, 1408
__lyap_clear, 1411
parameter, 1412
equation_stable.c, 1413
AINV_apply, 1414
AINV_clear, 1415
AINV_generate, 1415
AX_apply, 1416
AX_clear, 1417
AX_generate, 1417
ApEINV_apply, 1415
ApEINV_clear, 1416
ApEINV_generate, 1416
clear, 1417
EINV_apply, 1417
EINV_clear, 1417
EINV_generate, 1417
EX_apply, 1417
EX_clear, 1417
EX_generate, 1417
IS_REAL, 1414
init_rhs, 1417
parameter, 1417
equations.c, 1417
APPLY_FUNCTION2, 1419
APPLY_FUNCTION, 1419
callback_AINV_apply, 1420
callback_AINV_clear, 1420
callback_AINV_generate, 1420
callback_AX_apply, 1420
callback_AX_clear, 1420
callback_AX_generate, 1420
callback_ApEINV_apply, 1420
callback_ApEINV_clear, 1420
callback_ApEINV_generate, 1420
callback_ApEX_apply, 1420
callback_ApEX_clear, 1420
callback_ApEX_generate, 1420
callback_EINV_apply, 1420
callback_EINV_clear, 1421
callback_EINV_generate, 1421
callback_EX_apply, 1421
callback_EX_clear, 1421
callback_EX_generate, 1421
clear, 1421
eqn_converter, 1420
eqns, 1422
equations_t, 1420
op_to_int, 1421
PRE_POST_FUNCTION2, 1419
PRE_POST_FUNCTION, 1419
parameter, 1421
SET_IF_IMPLEMENTED, 1419
equations_t
equations.c, 1420
mess_equation_from_mexmess.c, 1644
error
mklpardiso_solver, 924
module_state, 925
Error Codes and Handling, 605
MESS_ERROR_ARGUMENTS, 606
MESS_ERROR_CHOLMOD, 606
MESS_ERROR_CONVERGE, 606
MESS_ERROR_CONVERT, 606
MESS_ERROR_DATATYPE, 606
MESS_ERROR_DATA, 606
MESS_ERROR_DIMENSION, 606
MESS_ERROR_DYNSYS, 606
MESS_ERROR_EIGENVALUES, 607
MESS_ERROR_FILEIO, 607
MESS_ERROR_GENERAL, 607
MESS_ERROR_LAPACK, 607

MESS_ERROR_MEMLOC, 607
MESS_ERROR_MEMORY, 607
MESS_ERROR_MISSING, 607
MESS_ERROR_MISC, 607
MESS_ERROR_MKLPARDISO, 607
MESS_ERROR_NOCOMPLEX, 607
MESS_ERROR_NOPATTERN, 607
MESS_ERROR_NOSUPPORT, 607
MESS_ERROR_NOTLOAD, 608
MESS_ERROR_NOTNULLPOINTER, 608
MESS_ERROR_NULLPOINTER, 608
MESS_ERROR_PYTHON_SIG, 608
MESS_ERROR_PYTHON, 608
MESS_ERROR_SINGULAR, 608
MESS_ERROR_STORAGETYPE, 608
MESS_ERROR_SYMMETRIC, 608
MESS_ERROR_UMFPACK, 608
MESS_ERROR_WRONG_HEADER, 608
MESS_SUCCESS, 608
mess_error_level, 609
mess_get_error, 608
mess_set_errorlevel, 609
Error Handling, 604
Error Macros, 610
FUNCTION_FAILURE_HANDLE_OMP, 610
FUNCTION_FAILURE_HANDLE, 610
MSG_ERROR, 610
MSG_FNAME, 611
MSG_INFO, 611
MSG_PRINT, 611
MSG_WARN_NOBT, 611
MSG_WARN, 611
error.c, 1423
error_handle
interface_matlab.c, 1467
error_macro.h, 1423
BTRSIZE, 1424
errors.h, 1424
Esolver
__lyap, 778
mvpdatag, 931
esterror
mess_bt_status_st, 826
EX
mess_equation_st, 849
ex_col_cpx
matrix/perm.c, 1795
ex_col_real
matrix/perm.c, 1795
exit
mess_threadpooljob_st, 913
exit_done
misc/init.c, 1464
eye.c, 1425
MIN, 1425

F
blas_defs.h, 999
res2_data_st, 942
wachspress.c, 1975
F77_GLOBAL_
fortran_translate.h, 1427
F77_GLOBAL
arpack.c, 977
blas_defs.h, 994–998
fortran_translate.h, 1427
lib/misc/eps.c, 1344
lu.c, 1608
mult.c, 1740
sign.c, 1846
F77_MODULE_
fortran_translate.h, 1427
F77_MODULE
fortran_translate.h, 1427
f_cpx
check_any.c, 1024
check_eliminate_zeros.c, 1111
f_one_value
check_one_value.c, 1198
f_one_valuec
check_one_value.c, 1198
f_real
check_any.c, 1024
check_eliminate_zeros.c, 1111
FACT
blas_defs.h, 999
FAIL
cuthill.c, 1314
FDM Matrix Generators, 599
mess_matgen_fdm_function, 600
mess_matgen_fdmcolumn, 600
mess_matgen_fdmmatrix, 600
mess_matgen_fdmrow, 601
mess_matgen_fdmvector, 602
FERR
blas_defs.h, 1000
FLIP
dfs.c, 1326
newlu.c, 1773
FROM
tests/data/bips98_606/source.txt, 1855
tests/data/siso_ww_vref_6405/source.txt, 1856
FUNCTION_FAILURE_HANDLE_MEX
Interfaces_matlab, 730
FUNCTION_FAILURE_HANDLE_OMP
Error Macros, 610
FUNCTION_FAILURE_HANDLE
cuthill.c, 1314
Error Macros, 610
fdm_matrix.c, 1425
zero_func, 1426
fdmcol
check_dynorm2.c, 1089
check_dynorm2t.c, 1091
check_dynormf.c, 1096
check_dynormft.c, 1097
check_lradi_unsym.c, 1146

check_lrnm_unsym.c, 1155
fdmcol2
check_lrnm_unsym.c, 1155
feature_tests.c, 1426
features, 1426
main, 1426
feature_tests.cxx, 1426
features, 1426
main, 1426
features
feature_tests.c, 1426
feature_tests.cxx, 1426
ferr
blas_defs.h, 999
Fhat
_sylv_solver_d_st, 799
_sylv_solver_sd_st, 801
fieldnames
cpp_interface/MESS_options/MESS_options.cc, 1685
fill.c, 1426
filter.c, 1427
__filter_stable_complex, 1427
__filter_stable_real, 1427
filter2D.C, 1427
finalh2
mess_h2_status_st, 856
finalsigma
mess_h2_status_st, 856
fn_calloc
malloc.c, 1610
fn_free
malloc.c, 1610
fn_malloc
malloc.c, 1610
fn_realloc
malloc.c, 1610
foo
TestRestrict.c, 1883
fortran_translate.h, 1427
F77_GLOBAL_, 1427
F77_GLOBAL, 1427
F77_MODULE_, 1427
F77_MODULE, 1427
FortranCInterface_GLOBAL_
VerifyFortran.h, 1974
FortranCInterface_GLOBAL
VerifyFortran.h, 1974
FortranCInterface_MODULE_
VerifyFortran.h, 1974
FortranCInterface_MODULE
VerifyFortran.h, 1974
free_call_t
malloc.c, 1610
freelist.c, 1427
MESS_FREELIST_ADD_GENERIC, 1428
freelist.h, 1428
full_value
MESS_matrix, 864
fullAsol
__glyap_dae1, 766
fullAsolver
__glyap_dae2, 768
fullMsolver
__glyap_dae2, 768
fullA
__glyap_dae1, 766
__glyap_dae2, 768
fullE
__glyap_dae1, 766
fullM
__glyap_dae2, 768
func
_mess_call, 797
function_AINV_apply
__matlab_equation, 779
function_AINV_clear
__matlab_equation, 779
function_AINV_generate
__matlab_equation, 779
function_AX_apply
__matlab_equation, 779
function_AX_clear
__matlab_equation, 779
function_AX_generate
__matlab_equation, 779
function_ApEINV_apply
__matlab_equation, 779
function_ApEINV_clear
__matlab_equation, 779
function_ApEINV_generate
__matlab_equation, 779
function_ApEX_apply
__matlab_equation, 779
function_ApEX_clear
__matlab_equation, 779
function_ApEX_generate
__matlab_equation, 779
function_EINV_apply
__matlab_equation, 780
function_EINV_clear
__matlab_equation, 780
function_EINV_generate
__matlab_equation, 780
function_EX_apply
__matlab_equation, 780
function_EX_clear
__matlab_equation, 780
function_EX_generate
__matlab_equation, 780
function_parameter
__matlab_equation, 780
fx
check_lradi_unsym.c, 1146
check_lrnm_unsym.c, 1155
fy

check_lradi_unsym.c, 1146
check_lrnm_unsym.c, 1156

G
__glyap_dae2, 768
mess_dynsys_st, 841
res2_data_st, 942
GENERATE_FUNCTION2
callback_equation.c, 1008
GENERATE_FUNCTION
callback_equation.c, 1008
GETSTATE
Interface to Python, 623
GLYAP3_GEN_LYAP
glyap3.c, 1436
GLYAP3_GEN_STEIN
glyap3.c, 1436
GLYAP3_ISOLVE
glyap3.c, 1436
GLYAP3_NB
glyap3.c, 1436
GLYAP3_STD_LYAP
glyap3.c, 1436
GLYAP3_STD_STEIN
glyap3.c, 1436
galerkin.c, 1429
IS_REAL, 1430
gaxpy.c, 1430
gc
__mess_plotter, 786
general
CMakeCache.txt, 1256
General Type Definitions and Macros, 44
DEPRECATED, 45
MESS_1_NORM, 47
MESS_2_NORM, 47
MESS_COMPLEX, 46
MESS_FROBENIUS_NORM, 47
MESS_INF_NORM, 47
MESS_INTEGER, 45
MESS_IS_COMPLEX, 45
MESS_IS_INTEGER, 45
MESS_IS_REAL, 45
MESS_OP_HERMITIAN, 47
MESS_OP_NONE, 47
MESS_OP_TRANSPOSE, 47
MESS_PRINT_FORMAT_LONG, 47
MESS_PRINT_FORMAT_SHORT, 47
MESS_PRINTF_INT2D, 46
MESS_PRINTF_INT, 45
MESS_REAL, 46
MESS_STORETYPE, 46
MESS_SYMMETRY, 46
mess_datatype_t, 46
mess_double_cpx_t, 46
mess_float_cpx_t, 46
mess_int_t, 46
mess_norm_t, 46
mess_operation_t, 47
mess_print_format_t, 47
generate
mess_equation_st::AINV_st, 802
mess_equation_st::AX_st, 805
mess_equation_st::ApEINV_st, 803
mess_equation_st::ApEX_st, 804
mess_equation_st::EINV_st, 812
mess_equation_st::EX_st, 813
generatespd
check_matrix_gbiorth.c, 1163
check_mgsadd.c, 1173
Generic Functions and Data Structures, 465
MESS_EQN_GLYAP, 467
MESS_EQN_GRICCATI, 467
MESS_EQN_LYAP, 467
MESS_EQN_NONE, 467
MESS_EQN_RICCATI, 467
MESS_LRCFADI_PARA_ADAPTIVE_V, 468
MESS_LRCFADI_PARA_ADAPTIVE_Z, 468
MESS_LRCFADI_PARA_MINMAX_REAL, 468
MESS_LRCFADI_PARA_MINMAX, 468
MESS_LRCFADI_PARA_WACHSPRESS, 468
MESS_MEMORY_HIGH, 468
MESS_MEMORY_LOW, 468
MESS_MEMORY_MID, 468
MESS_RESIDUAL_INDEFINITE, 468
MESS_RESIDUAL_SPECTRAL, 468
mess_equation, 467
mess_equation_clean, 468
mess_equation_clear, 469
mess_equation_init, 469
mess_equation_set_rhs, 470
mess_equation_set_rhs2, 470
mess_equation_t, 467
mess_lrcfadi_step, 467
mess_memusage_t, 467
mess_options, 467
mess_options_clear, 471
mess_options_copy, 471
mess_options_init, 472
mess_options_memsize, 473
mess_options_print, 473
mess_options_set_K0, 474
mess_options_set_proj_space, 475
mess_options_set_W, 476
mess_options_st, 467
mess_options_unset_K0, 476
mess_options_unset_proj_space, 477
mess_options_unset_W, 477
mess_parameter_t, 468
mess_residual_t, 468
mess_status, 467
mess_status_clear, 478
mess_status_copy, 478
mess_status_init, 479
mess_status_memsize, 480
mess_status_print, 480
mess_status_printfull, 481

mess_status_st, 467
mess_status_write, 482
Generic Interface to Direct Solvers, 236
MESS_HAVE_DETC, 237
MESS_HAVE_DET, 237
MESS_HAVE_GETPERMP, 238
MESS_HAVE_GETPERMQ, 238
MESS_HAVE_GETSCALECOL, 238
MESS_HAVE_GETSCALEROW, 238
MESS_HAVE_GETL, 237
MESS_HAVE_GETU, 238
MESS_HAVE_INVERSE, 238
MESS_HAVE_SOLVEMH, 238
MESS_HAVE_SOLVEMT, 238
MESS_HAVE_SOLVEH, 238
MESS_HAVE_SOLVEM, 238
MESS_HAVE_SOLVET, 238
MESS_HAVE_SOLVE, 238
mess_direct, 238
mess_direct_clear, 238
mess_direct_determinant, 239
mess_direct_determinantc, 239
mess_direct_getL, 239
mess_direct_getpermp, 240
mess_direct_getpermq, 240
mess_direct_getscalecol, 240
mess_direct_getscalerow, 240
mess_direct_getU, 241
mess_direct_init, 241
mess_direct_inverse, 241
mess_direct_solve, 241
mess_direct_solvem, 242
mess_matrix_backslash, 242
mess_matrix_backslashm, 244
mess_matrix_pinv, 245
Generic Interface to Multisolvers, 318
MESS_MULTIDIRECT_SPARSE_LU, 319
MESS_MULTIDIRECT_UMFPACK_LU, 319
mess_multidirect, 319
mess_multidirect_clear, 319
mess_multidirect_init, 319
mess_multidirect_solve, 320
mess_multidirect_solvem, 320
mess_multidirect_t, 319
Generic matrix-vector Products, 183
mess_mvpcall, 184
mess_mvpcall_apply, 184
mess_mvpcall_clear, 184
mess_mvpcall_matrix, 184
mess_mvpcall_operator, 185
get_char
H2 Model Order Reduction, 407
get_ptr
MESS_direct, 831
MESS_equation, 846
MESS_matrix, 865
MESS_multidirect, 878
MESS_options, 890
MESS_status, 904
MESS_vector, 919
getcol.c, 1430
getdatatype
mess_multidirect_st, 881
getelement.c, 1430
getL
mess_direct_st, 838
mess_multidirect_st, 881
getp
mess_multidirect_st, 881
getpermp
mess_direct_st, 838
getpermq
mess_direct_st, 839
getq
mess_multidirect_st, 881
getrow.c, 1431
getscalecol
mess_direct_st, 839
mess_multidirect_st, 881
getscalerow
mess_direct_st, 839
mess_multidirect_st, 881
getset.c, 1431
gettypes.c, 1431
MESS_ENUM_STR_1, 1432
MESS_ENUM_STR_2, 1432
MESS_ENUM_STR_3, 1432
MESS_ENUM_STR_4, 1432
MESS_ENUM_STR_5, 1433
MESS_ENUM_STR_6, 1433
MESS_ENUM_STR_7, 1433
MESS_ENUM_STR_8, 1433
MESS_ENUM_STR_9, 1433
mess_print_format_t_str, 1433
getU
mess_direct_st, 839
mess_multidirect_st, 881
gglrsrm.c, 1433
glyap.c, 1433
glyap3.c, 1434
_glyap3, 1436
_glyap3_eqn_t, 1436
GLYAP3_GEN_LYAP, 1436
GLYAP3_GEN_STEIN, 1436
GLYAP3_ISOLVE, 1436
GLYAP3_NB, 1436
GLYAP3_STD_LYAP, 1436
GLYAP3_STD_STEIN, 1436
glyap3_clear, 1436
glyap3_eqn_t_str, 1437
glyap3_solvem, 1437
glyap3_solvemt, 1438
glyap3_solvemx, 1438
glyapchol_solvem, 1440
glyapchol_solvemt, 1441
glyapchol_solvemx, 1442

mess_direct_create_glyap3, 1443
mess_glyap3x, 1444
mess_tglyap3x, 1445
glyap3_clear
glyap3.c, 1436
glyap3_eqn_t_str
glyap3.c, 1437
glyap3_solvem
glyap3.c, 1437
glyap3_solvemt
glyap3.c, 1438
glyap3_solvemx
glyap3.c, 1438
glyap_test.c, 1447
main, 1447
glyapchol_solvem
glyap3.c, 1440
glyapchol_solvemt
glyap3.c, 1441
glyapchol_solvemx
glyap3.c, 1442
gmres.c, 1448
Graph Algorithms, 595
mess_graph_dfs, 595
mess_graph_lblockpart, 596
mess_graph_reach, 596
gslrsrm.c, 1449
gstein.c, 1449
gstein_test.c, 1449
main, 1450

H
blas_defs.h, 1000
mess_krylov_arnoldi_st, 858
h1
mvpdata, 929
mvpdatag, 931
h2
mvpdata, 929
mvpdatag, 931
H2 Model Order Reduction, 392
__constructVW, 396
__constructVW_pre, 397
__constructVW_pthread, 398
__constructVW_sec, 399
__constructVW_seq, 400
__constructVWmat, 401
__project_A, 402
__project_B, 403
__project_C, 405
__project_Bmat, 404
__project_Cmat, 406
AR_W_VA, 395
get_char, 407
MESS_H2NORM_FULL, 395
MESS_H2NORM_NO, 395
MESS_H2NORM_UPDATE, 396
mess_h2_error, 407
mess_h2_error_internal, 408
mess_h2_irka_biorth, 409
mess_h2_irka_init, 411
mess_h2_norm, 413
mess_h2_norm_internal, 414
mess_h2_options, 396
mess_h2_options_clear, 416
mess_h2_options_init, 417
mess_h2_options_print, 417
mess_h2_status, 396
mess_h2_status_clear, 417
mess_h2_status_init, 417
mess_h2_status_print, 418
mess_h2_status_printfull, 418
mess_h2_status_write_mfile, 419
mess_h2_tsia, 419
mess_h2_tsiag, 420
VW_worker, 422
h2.h, 1451
h2_norm.c, 1453
INFINITY, 1453
h2err
mess_h2_status_st, 856
h2error.c, 1453
INFINITY, 1454
HEX_1
CMakeFortranCompilerId.F, 1261
HEX_2
CMakeFortranCompilerId.F, 1261
HEX_3
CMakeFortranCompilerId.F, 1261
HEX_4
CMakeFortranCompilerId.F, 1261
HEX_5
CMakeFortranCompilerId.F, 1261
HEX_6
CMakeFortranCompilerId.F, 1261
HEX_7
CMakeFortranCompilerId.F, 1261
HEX_8
CMakeFortranCompilerId.F, 1261
HEX
CMakeCCompilerId.c, 1257
CMakeCXXCompilerId.cpp, 1259
HOWMNY
blas_defs.h, 1000
hash
threadpool.c, 1884
Hashtable, 690
hashtable_done
mess_threadpool_st, 911
have_mass
parameter_mvp_call, 939
haveUW
mvpdata, 929
mvpdatag, 931
head
mess_threadpool_st, 911
height

__mess_plotter, 786
hessenberg_clear
hessenberg_lu.c, 1455
hessenberg_lu.c, 1454
hessenberg_clear, 1455
hessenberg_solve, 1455
hessenberg_solve_all, 1455
hessenberg_solveh, 1456
hessenberg_solvem, 1456
hessenberg_solvem_all, 1457
hessenberg_solvemh, 1458
hessenberg_solvemt, 1458
hessenberg_solver, 1455
hessenberg_solvet, 1458
hessenberg_solvex, 1458
hessenberg_solve
hessenberg_lu.c, 1455
hessenberg_solve_all
hessenberg_lu.c, 1455
hessenberg_solveh
hessenberg_lu.c, 1456
hessenberg_solvem
hessenberg_lu.c, 1456
hessenberg_solvem_all
hessenberg_lu.c, 1457
hessenberg_solvemh
hessenberg_lu.c, 1458
hessenberg_solvemt
hessenberg_lu.c, 1458
hessenberg_solver, 814
data_type, 814
hessenberg_lu.c, 1455
ipiv, 814
n, 815
val, 815
val_cpx, 815
hessenberg_solvet
hessenberg_lu.c, 1458
hessenberg_solvex
hessenberg_lu.c, 1458
Hhat
_sylv_solver_d_st, 799
_sylv_solver_sd_st, 801
how
blas_defs.h, 1000

i
VWjob, 947
IDX2F
banded.c, 980
IDX
matrix/rand.c, 1808
IFAILL
blas_defs.h, 1000
IFAILR
blas_defs.h, 1000
IHI
blas_defs.h, 1000
ILU_LEVEL
direct/singlesolver/bicgstab.c, 986
ILUK
iluk.c, 1462
ILO
blas_defs.h, 1000
INCX
blas_defs.h, 1000
INCY
blas_defs.h, 1000
INFINITY
condest.c, 1263
h2_norm.c, 1453
h2error.c, 1454
INFO
blas_defs.h, 1000
INITERROR
Interface to Python, 623
INITV
blas_defs.h, 1000
INSTALL.md, 1465
IPIV
blas_defs.h, 1000
IS_COMPLEX
multilu_adavanced.c, 1744
multilu_umfpack.c, 1756
IS_REAL
equation_stable.c, 1414
galerkin.c, 1430
lib/lrcf_adi/lradi.c, 1597
lib/lrcf_adi/lrnm.c, 1604
multilu_adavanced.c, 1744
multilu_umfpack.c, 1756
ISGN
blas_defs.h, 1000
ISOLVE
blas_defs.h, 1000
IT_MAX
lrcf_adi/norms/res2.c, 1821
rc2norm.c, 1810
IWORK
blas_defs.h, 1001
id
mess_threadpooljob_st, 913
th_done_entry, 945
ijob
blas_defs.h, 1000
ilu0.c, 1459
__mess_precond_ilu0_clear, 1460
__mess_precond_ilu0_solve, 1460
ilu0_data, 1460
ilu0_data
ilu0.c, 1460
iluk.c, 1461
__iluk_clear, 1462
__iluk_solve, 1462
__iluk_solve_complex, 1462
CCSRMat, 1462
ccsrptr, 1462

clean_ccsrmat, 1463
ILUK, 1462
ilukptr, 1462
max, 1462
min, 1462
symbolic_iluk, 1463
ilukptr
iluk.c, 1462
incx
blas_defs.h, 1000
incy
blas_defs.h, 1000
index
jobdata_st, 818
indx
mess_multidirect_st, 881
Info
multi_umfpack, 925
umfpack_control_solver, 946
info
banded_lapack, 807
blas_defs.h, 1000
info_arch
CMakeCCompilerId.c, 1258
CMakeCXXCompilerId.cpp, 1260
info_compiler
CMakeCCompilerId.c, 1258
CMakeCXXCompilerId.cpp, 1260
info_handle
interface_matlab.c, 1467
info_language_dialect_default
CMakeCCompilerId.c, 1258
CMakeCXXCompilerId.cpp, 1260
info_platform
CMakeCCompilerId.c, 1258
CMakeCXXCompilerId.cpp, 1260
info_size
DCPX_T.c, 1321
MESS_HAVE_PTRDIFF_T.c, 1645
MESS_HAVE_SIZE_T.c, 1646
init.c, 1463, 1464
init_c_interface
Interface to Python, 634
init_done
misc/init.c, 1464
init_mexmess
Interfaces_matlab, 739
init_numpy
Interface to Python, 634
init_rhs
equation_glyap_dae2.c, 1378
equation_stable.c, 1417
mess_equation_st, 849
Input/Output Operations, 722
Input/Output Operations with Matrices, 155
mess_matrix_display, 155
mess_matrix_print, 156
mess_matrix_printdata, 157
mess_matrix_printinfo, 157
mess_matrix_printshort, 158
mess_matrix_read, 159
mess_matrix_read_formated, 160
mess_matrix_write, 161
inputs
mess_dynsys_st, 842
install
MESS_datatype_t, 828
MESS_direct, 831
MESS_direct_cholpackage_t, 833
MESS_direct_lupackage_t, 836
MESS_equation, 846
MESS_equation_t, 851
MESS_matrix, 865
MESS_memusage_t, 874
MESS_multidirect, 878
MESS_multidirect_t, 883
MESS_operation_t, 887
MESS_options, 890
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_status, 904
MESS_storage_t, 909
MESS_vector, 919
int_list, 815
el, 816
elnmax, 816
ix, 816
ixn, 816
ixnmax, 816
int_list_alloc
cuthill.c, 1319
int_list_free
cuthill.c, 1319
Interface for handling Dynamical Systems, 351
MESS_DYNSYS_2ND, 353
MESS_DYNSYS_GLTI, 353
MESS_DYNSYS_LTI, 353
MESS_IS_DYNSYS_2ND, 353
MESS_IS_DYNSYS_GLTI, 353
MESS_IS_DYNSYS_LTI, 353
MESS_IS_DYNSYS_SISO, 353
mess_dynsys, 353
mess_dynsys_2nd, 353
mess_dynsys_2nd_copy, 354
mess_dynsys_2nd_to_1st, 355
mess_dynsys_clear, 356
mess_dynsys_evaltransfer, 356
mess_dynsys_glti, 358
mess_dynsys_glti_copy, 359
mess_dynsys_init, 360
mess_dynsys_isstable, 360
mess_dynsys_lti, 361
mess_dynsys_lti_copy, 362
mess_dynsys_printinfo, 362
mess_dynsys_project_2nd_to_1st, 363
mess_dynsys_project_2nd_to_2nd, 364

mess_dynsys_project_to_glti, 365
mess_dynsys_project_to_lti, 366
mess_dynsys_transfer_diff, 367
mess_dynsysstr, 368
mess_matrix_isstable, 369
mess_matrix_isstableg, 369
Interface to CSparse, 613
mess_matrix_from_csparse, 613
mess_matrix_from_csparse_ci, 614
mess_matrix_from_csparse_cl, 614
mess_matrix_from_csparse_di, 615
mess_matrix_from_csparse_dl, 615
mess_matrix_to_csparse, 616
mess_matrix_to_csparse_ci, 616
mess_matrix_to_csparse_cl, 617
mess_matrix_to_csparse_di, 617
mess_matrix_to_csparse_dl, 618
Interface to Cholmod, 619
mess_matrix_convert_cholmod_to_csc, 619
mess_matrix_convert_cholmod_to_dense, 619
mess_matrix_convert_csc_to_cholmod, 619
mess_matrix_convert_dense_to_cholmod, 619
mess_vector_convert_cholmod_to_dense, 619
mess_vector_convert_dense_to_cholmod, 619
Interface to Python, 620
_state, 655
ConvStringtoC, 623
eqn_conv_lyap, 625
eqn_conv_lyap_dae1, 625
eqn_conv_lyap_dae2, 626
eqn_conv_lyap_so1, 627
eqn_conv_lyap_so2, 628
eqn_conv_riccati, 629
eqn_conv_riccati_dae1, 630
eqn_conv_riccati_dae2, 631
eqn_conv_riccati_so1, 632
eqn_conv_riccati_so2, 633
GETSTATE, 623
INITERROR, 623
init_c_interface, 634
init_numpy, 634
interface_python_methods, 655
MESS_Py_XDECREF, 623
matrix_to_c, 635
matrix_to_python, 635
mess_equation_from_python, 636
mess_options_from_python, 637
mess_status_to_python, 638
NPY_NO_DEPRECATED_API, 623
PY_ARRAY_UNIQUE_SYMBOL, 623
PY_GET_DOUBLE, 623
PY_GET_LONG, 623
PY_GET_MATRIX, 623
PY_GET_VECTOR, 623
PY_SET_BOOL, 624
PY_SET_DOUBLE, 624
PY_SET_LONG, 624
PY_SET_VECTOR, 624
PYCMESS_ERROR, 624
PYMESS_READ_MATRIX, 625
pymess_care, 638
pymess_dense_nm_gmpare, 639
pymess_direct_chol_select, 640
pymess_direct_select, 641
pymess_eps, 641
pymess_glyap, 642
pymess_gstein, 642
pymess_is_reimplemented, 643
pymess_lradi, 644
pymess_lrnm, 644
pymess_lyap, 645
pymess_mess_git_branch, 647
pymess_mess_git_id, 648
pymess_mess_have_amd, 648
pymess_mess_have_arpack, 648
pymess_mess_have_bzip2, 648
pymess_mess_have_cholmod, 648
pymess_mess_have_colamd, 648
pymess_mess_have_csparse, 648
pymess_mess_have_matio, 648
pymess_mess_have_mess64, 648
pymess_mess_have_mklpardiso, 648
pymess_mess_have_openmp, 648
pymess_mess_have_superlu, 648
pymess_mess_have_umfpack, 648
pymess_mess_have_zlib, 649
pymess_mess_is_debug, 649
pymess_mess_version, 649
pymess_mess_version_major, 649
pymess_mess_version_minor, 649
pymess_mess_version_patch, 649
pymess_mess_version_verbose, 650
pymess_multidirect_select, 650
pymess_set_errorlevel, 651
pymess_sylvester_sparsedense, 651
pymess_test_equation, 652
pymess_test_matrix, 653
pymess_test_status, 653
pymess_test_vector, 653
vector_to_c, 653
vector_to_python, 654
interface_cholmod.h, 1465
interface_csparse.h, 1465
interface_matlab.c, 1466
error_handle, 1467
info_handle, 1467
matlab_calloc, 1466
matlab_free, 1466
matlab_malloc, 1466
matlab_realloc, 1466
mess_int_array_to_matlab, 1466
my_mexError, 1466
my_mexPrintf, 1466
my_mexWarn, 1466
print_handle, 1467
warn_handle, 1467

interface_matlab.h, 1467
MESS64, 1469
MESS_MATLAB, 1469
interface_python_methods
Interface to Python, 655
Interfaces to other Programming Languages and Libraries, 612
Interfaces_matlab, 728
eqn_conv_lyap, 730
eqn_conv_lyap_dae1, 730
eqn_conv_lyap_dae2, 731
eqn_conv_lyap_so1, 732
eqn_conv_lyap_so2, 733
eqn_conv_ric, 734
eqn_conv_ric_dae1, 735
eqn_conv_ric_dae2, 736
eqn_conv_ric_so1, 737
eqn_conv_ric_so2, 738
FUNCTION_FAILURE_HANDLE_MEX, 730
init_mexmess, 739
mess_callback_equation_mexmess, 740
mess_check_struct, 730
mess_classname_from_mexmess, 741
mess_complex_from_mexmess, 742
mess_direct_cholpackage_t_from_mexmess, 742
mess_direct_cholpackage_t_to_mexmess, 742
mess_direct_lupackage_t_from_mexmess, 742
mess_direct_lupackage_t_to_mexmess, 742
mess_double_from_mexmess, 742
mess_equation_from_mexmess, 742
mess_equation_t_from_mexmess, 743
mess_equation_t_to_mexmess, 743
mess_matrix_from_mexmess, 743
mess_matrix_to_mexmess, 745
mess_memusage_t_from_mexmess, 746
mess_memusage_t_to_mexmess, 746
mess_multidirect_t_from_mexmess, 746
mess_multidirect_t_to_mexmess, 746
mess_norm_t_from_mexmess, 746
mess_norm_t_to_mexmess, 746
mess_operation_t_from_mexmess, 746
mess_operation_t_to_mexmess, 746
mess_options_from_mexmess, 746
mess_options_to_mexmess, 747
mess_parameter_t_from_mexmess, 747
mess_parameter_t_to_mexmess, 748
mess_residual_t_from_mexmess, 748
mess_residual_t_to_mexmess, 748
mess_status_to_mexmess, 748
mess_vector_from_mexmess, 748
mess_vector_to_mexmess, 748
mex_mess_care, 749
mex_mess_dense_nm_gmpare, 749
mex_mess_direct_chol_select, 750
mex_mess_direct_lu_select, 750
mex_mess_git_branch, 750
mex_mess_git_id, 750
mex_mess_glyap, 750
mex_mess_gstein, 751
mex_mess_have_amd, 752
mex_mess_have_arpack, 752
mex_mess_have_bzip2, 752
mex_mess_have_cholmod, 752
mex_mess_have_colamd, 752
mex_mess_have_csparse, 752
mex_mess_have_matio, 752
mex_mess_have_mess64, 752
mex_mess_have_mklpardiso, 752
mex_mess_have_openmp, 752
mex_mess_have_superlu, 752
mex_mess_have_umfpack, 753
mex_mess_have_zlib, 753
mex_mess_is_debug, 753
mex_mess_lradi, 753
mex_mess_lrnm, 753
mex_mess_lyap, 754
mex_mess_multidirect_select, 755
mex_mess_sylvester_dense, 755
mex_mess_sylvester_sparsedense, 756
mex_mess_version, 757
mex_mess_version_major, 758
mex_mess_version_minor, 758
mex_mess_version_patch, 758
mex_mess_version_verbose, 758
mex_test_matrix, 759
mex_test_mess_direct_cholpackage_t, 760
mex_test_mess_direct_lupackage_t, 760
mex_test_mess_equation_t, 760
mex_test_mess_memusage_t, 760
mex_test_mess_multidirect_t, 760
mex_test_mess_norm_t, 760
mex_test_mess_operation_t, 760
mex_test_mess_parameter_t, 760
mex_test_mess_residual_t, 760
mex_test_options, 760
mex_test_status, 761
mex_test_vector, 762
mxCreateComplexScalar, 763
mxIsSubClass, 763
internal_status
mess_status_st, 907
inverse
mess_direct_st, 839
inverse.c, 1469
main, 1470
select_solver, 1470
testmat, 1472
invperm
banded_lapack, 807
iparm
mklpardiso_solver, 924
ipiv
banded_lapack, 807
blas_defs.h, 1000
hessenberg_solver, 814
lapack_solver, 819

irka_biorth.c, 1474
ENDTIME, 1474
STARTTIME, 1474
irka_common.c, 1474
irka_para_init.c, 1475
compare_complex, 1476
is_complex_matrix
MESS_matrix, 866
MESS_vector, 919
is_complex_type
MESS_matrix, 866
MESS_vector, 920
is_constant
MESS_datatype_t, 828
MESS_direct, 831
MESS_direct_cholpackage_t, 833
MESS_direct_lupackage_t, 836
MESS_equation, 846
MESS_equation_t, 851
MESS_matrix, 866
MESS_memusage_t, 874
MESS_multidirect, 878
MESS_multidirect_t, 883
MESS_operation_t, 887
MESS_options, 890
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_status, 904
MESS_storage_t, 909
MESS_vector, 920
is_defined
MESS_datatype_t, 828
MESS_direct, 831
MESS_direct_cholpackage_t, 833
MESS_direct_lupackage_t, 836
MESS_equation, 846
MESS_equation_t, 851
MESS_matrix, 866
MESS_memusage_t, 874
MESS_multidirect, 878
MESS_multidirect_t, 883
MESS_operation_t, 887
MESS_options, 890
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_status, 904
MESS_storage_t, 909
MESS_vector, 920
is_double_type
MESS_matrix, 866
MESS_vector, 920
is_empty
MESS_datatype_t, 828
MESS_direct, 831
MESS_direct_cholpackage_t, 833
MESS_direct_lupackage_t, 836
MESS_equation, 846
MESS_equation_t, 851
MESS_matrix, 866
MESS_memusage_t, 874
MESS_multidirect, 878
MESS_multidirect_t, 883
MESS_operation_t, 887
MESS_options, 890
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_status, 904
MESS_storage_t, 909
MESS_vector, 920
is_map
MESS_options, 891
is_matrix_type
MESS_matrix, 866
MESS_vector, 920
is_numeric_type
MESS_matrix, 866
MESS_vector, 920
is_real_matrix
MESS_matrix, 866
MESS_vector, 920
is_real_type
MESS_matrix, 866
MESS_vector, 920
is_single_type
MESS_matrix, 866
MESS_vector, 920
is_sparse_type
MESS_matrix, 866
isfinite
Miscellaneous Helper Functions, 675
isreal
_sylv_solver_d_st, 799
_sylv_solver_sd_st, 801
issAnorm2inv
check_norm.c, 1197
iszero_cpx
check_eliminate_zeros.c, 1111
iszero_real
check_eliminate_zeros.c, 1111
it
__mess_lrcfadi_step, 781
__mess_solver_status, 791
mess_h2_status_st, 856
mess_status_st, 907
Iterative Solvers for Linear Systems, 329, 723
Iwork
blas_defs.h, 1000
ix
int_list, 816
ixn
int_list, 816
ixnmax
int_list, 816

j
mess_krylov_arnoldi_st, 858
JOBVSR

blas_defs.h, 1001
JOBVR
blas_defs.h, 1001
JOBVT
blas_defs.h, 1001
JOBR
blas_defs.h, 1001
JOB
blas_defs.h, 1001
JPVT
blas_defs.h, 1001
jacobi.c, 1476
JaySolver
mvpdatag, 931
job_decompose_complex
multilu_adavanced.c, 1744
job_decompose_real
multilu_adavanced.c, 1744
job_done
mess_threadpool_st, 911
jobdata
multilu_adavanced.c, 1744
jobdata_st, 817
cols, 818
index, 818
lind, 818
lp, 818
lvalues, 818
lvalues_next, 818
mlu, 818
pair, 818
pperm, 818
qperm, 818
ret, 818
rows, 818
uind, 818
up, 818
uvalues, 818
uvalues_next, 818
vwork, 819
workcolptr, 819
workrowptr, 819
workvalues, 819

K
__glyap_so1, 771
__glyap_so2, 773
blas_defs.h, 1001
mess_dynsys_st, 842
mess_equation_st, 849
res2_data_st, 942
k
blas_defs.h, 1001
K0
mess_options_st, 895
KAPPA
twoside_mgs.c, 1951
Khat
__stable_lyap, 794
KL
blas_defs.h, 1001
kl
banded_lapack, 807
kron.c, 1476
Ksolver
__glyap_so1, 771
__glyap_so2, 773
KU
blas_defs.h, 1001
ku
banded_lapack, 807
KX
__stable_lyap, 794

L
__iluk, 776
newlu_data, 933
l
pair_mess_int_t_complex, 937
pair_mess_int_t_double, 938
l1
pair_mess_int_t2, 935
pair_mess_int_t2_complex, 936
pair_mess_int_t2_double, 937
l2
pair_mess_int_t2, 935
pair_mess_int_t2_complex, 936
pair_mess_int_t2_double, 937
LAPACK_ERROR
banded.c, 980
LD_MULTIPLICITY
config.h, 1265
LDAB
blas_defs.h, 1001
LDVSL
blas_defs.h, 1002
LDVSR
blas_defs.h, 1002
LDVL
blas_defs.h, 1002
LDVR
blas_defs.h, 1002
LDVS
blas_defs.h, 1002
LDVT
blas_defs.h, 1002
LDWORK
blas_defs.h, 1002
LDA
blas_defs.h, 1001
LDB
blas_defs.h, 1001
LDC
blas_defs.h, 1001
LDD
blas_defs.h, 1001
LDE
blas_defs.h, 1001

LDF
blas_defs.h, 1001
LDH
blas_defs.h, 1002
LDQ
blas_defs.h, 1002
LDT
blas_defs.h, 1002
LDU
blas_defs.h, 1002
LDX
blas_defs.h, 1002
LDZ
blas_defs.h, 1002
LINE_LENGTH
read.c, 1812
vector_read.c, 1972
LINESEARCH_BREAKDOWN
dense_nm.c, 1325
LOGLEVEL
config.h, 1265
LVDS
blas_defs.h, 1002
LWORK
blas_defs.h, 1002
label
__mess_plotExportData, 784
__mess_plotdata, 782
_scalepoints, 797
lanczos_template_nrm.c, 1477
lapack.c, 1477
lapack_clear, 1478
lapack_det, 1478
lapack_detc, 1478
lapack_getL, 1478
lapack_getpermp, 1479
lapack_getU, 1479
lapack_inverse, 1480
lapack_solve, 1480
lapack_solveh, 1481
lapack_solvem, 1481
lapack_solvemh, 1482
lapack_solvemt, 1483
lapack_solvet, 1483
lapack_clear
lapack.c, 1478
lapack_det
lapack.c, 1478
lapack_detc
lapack.c, 1478
lapack_getL
lapack.c, 1478
lapack_getpermp
lapack.c, 1479
lapack_getU
lapack.c, 1479
lapack_inverse
lapack.c, 1480
lapack_qr.c, 1484
lapackqr_clear, 1485
lapackqr_getL_over, 1485
lapackqr_getL_under, 1486
lapackqr_getU_over, 1487
lapackqr_getU_under, 1487
lapackqr_solve_over, 1488
lapackqr_solve_under, 1489
lapackqr_solveh_over, 1490
lapackqr_solveh_under, 1491
lapackqr_solvem_over, 1492
lapackqr_solvem_under, 1493
lapackqr_solvemh_over, 1494
lapackqr_solvemh_under, 1495
lapackqr_solvemt_over, 1496
lapackqr_solvemt_under, 1497
lapackqr_solvet_over, 1498
lapackqr_solvet_under, 1499
mess_direct_create_lapack_qr_over, 1500
mess_direct_create_lapack_qr_under, 1501
lapack_solve
lapack.c, 1480
lapack_solveh
lapack.c, 1481
lapack_solvem
lapack.c, 1481
lapack_solvemh
lapack.c, 1482
lapack_solvemt
lapack.c, 1483
lapack_solver, 819
cpx, 819
ipiv, 819
n, 820
val, 820
val_cpx, 820
lapack_solvet
lapack.c, 1483
lapackqr_clear
lapack_qr.c, 1485
lapackqr_getL_over
lapack_qr.c, 1485
lapackqr_getL_under
lapack_qr.c, 1486
lapackqr_getU_over
lapack_qr.c, 1487
lapackqr_getU_under
lapack_qr.c, 1487
lapackqr_solve_over
lapack_qr.c, 1488
lapackqr_solve_under
lapack_qr.c, 1489
lapackqr_solveh_over
lapack_qr.c, 1490
lapackqr_solveh_under
lapack_qr.c, 1491
lapackqr_solvem_over
lapack_qr.c, 1492

lapackqr_solvem_under
lapack_qr.c, 1493
lapackqr_solvemh_over
lapack_qr.c, 1494
lapackqr_solvemh_under
lapack_qr.c, 1495
lapackqr_solvemt_over
lapack_qr.c, 1496
lapackqr_solvemt_under
lapack_qr.c, 1497
lapackqr_solver, 820
cols, 821
cpx, 821
rows, 821
tau, 821
tau_cpx, 821
val, 821
val_cpx, 821
lapackqr_solvet_over
lapack_qr.c, 1498
lapackqr_solvet_under
lapack_qr.c, 1499
lblock_dfs
blockpart.c, 1005
ld
mess_matrix_file_st, 870
mess_matrix_st, 872
ld_multiplicity
matrix/alloc.c, 961
lda
blas_defs.h, 1001
ldb
blas_defs.h, 1001
ldc
blas_defs.h, 1001
legendGnuPos
__mess_plotterScript, 789
legendTikzPos
__mess_plotterScript, 789
len
__mess_plotExportData, 784
__mess_plotdata, 782
len_fieldnames
cpp_interface/MESS_options/MESS_options.cc, 1685
length
MESS_matrix, 867
MESS_vector, 920
levelind
mess_direct_levelset_st, 835
levelptr
mess_direct_levelset_st, 835
levels
mess_direct_levelset_st, 835
lgfortran
CMakeCache.txt, 1256
lhsolvec
newlu.c, 1773
lib/CMakeFiles/mess.dir/link.txt
D_FILE_OFFSET_BITS, 1503
soname, 1503
lib/direct/singlesolver/sylvester_dense.c
_sylv_solver_d, 1864
sylvester_d_clear, 1864
sylvester_d_solvem_generalized, 1865
sylvester_d_solvem_standard, 1865
sylvester_d_solvemh_generalized, 1866
sylvester_d_solvemh_standard, 1867
sylvester_d_solvemt_generalized, 1869
sylvester_d_solvemt_standard, 1870
lib/direct/singlesolver/sylvester_sparsedense.c
_sylv_equation_sd_t, 1874
_sylv_solver_sd, 1874
SYLV_SPARSE_DENSE_GENERALIZED, 1874
SYLV_SPARSE_DENSE_HALF_GENERALIZED, 1874
SYLV_SPARSE_DENSE_STANDARD, 1874
sylvester_sd_clear, 1874
sylvester_sd_solvem, 1875
sylvester_sd_solvemh, 1876
sylvester_sd_solvemt, 1878
lib/lrcf_adi/lradi.c
IS_REAL, 1597
RES2_COUNT, 1597
RES_ABORT, 1597
lib/lrcf_adi/lrnm.c
_scalecat, 1604
IS_REAL, 1604
SET_COLOR, 1604
UNSET_COLOR, 1604
lib/misc/eps.c
F77_GLOBAL, 1344
lib/misc/misc.c
mess_versionx, 1723
lift.c, 1502
lind
jobdata_st, 818
multilu, 927
linesearch_poly_eval
lrnm_linesearch.c, 1606
link.txt, 1503–1574
linsolve.c, 1574
main, 1575
select_solver, 1576
testmat, 1578
linsolve_multi.c, 1580
main, 1581
select_solver, 1582
testmat, 1584
linsolvem.c, 1586
main, 1587
select_solver, 1588
testmat, 1590
linsolvem_multi.c, 1592
linsolvem_multi_testmat, 1593
main, 1594

select_solver, 1595
linsolvem_multi_testmat
linsolvem_multi.c, 1593
linspace.c, 1596
Llevels
csparse_solver, 811
lnnz
multilu, 927
lock
__mess_plotter, 786
mess_threadpool_st, 911
Low-Rank Methods for Matrix Equations, 464, 724
lowerbound
__glyap_so1, 771
__glyap_so2, 773
lp
jobdata_st, 818
multilu, 927
lpthread
CMakeCache.txt, 1256
lradi.c, 1597
lrcf_adi.h, 1598
lrcf_adi/norms/res2.c
IT_MAX, 1821
MATRIX_ADD, 1821
res2_data, 1822
res2_mvp, 1822
res2_mvpBK, 1823
res2_mvpt, 1824
res2g_mvp, 1825
res2g_mvpBK, 1826
res2gt_mvp, 1827
res2nm_data, 1822
res2nm_mvp, 1828
res2nmg_mvp, 1829
VECTOR_AXPY, 1821
VECTOR_SCALE, 1822
lrnm.c, 1604, 1605
lrnm_linesearch.c, 1606
__filter_linesearch_cpx, 1606
__filter_linesearch_real, 1606
linesearch_poly_eval, 1606
lrsrm.c, 1606
lrsrm_somor.c, 1607
lrsrm_somor_so.c, 1607
lsolve
newlu.c, 1773
lsolvec
newlu.c, 1774
lsolverc
newlu.c, 1774
lstore_type
multilu, 927
ltsolve
newlu.c, 1774
ltsolvec
newlu.c, 1774
ltsolverc
newlu.c, 1774
LU
csparse_solver, 811
csparse_solver_complex, 811
lu.c, 1607
F77_GLOBAL, 1608
USE_FORTRAN_KERNELS, 1608
lu_type
direct.c, 1331
lu_type_mutex
direct.c, 1331
lvalues
jobdata_st, 818
multilu, 927
lvalues_cpx
multilu, 927
lvalues_next
jobdata_st, 818
lwork
blas_defs.h, 1002
lyap.c, 1608
lyap_res2.c, 1608
lyap_sgn_fac.c, 1609

M
__glyap_dae2, 769
__glyap_so1, 771
__glyap_so2, 773
blas_defs.h, 1002
mess_dynsys_st, 842
res2_data_st, 942
res2nm_data_st, 944
m
blas_defs.h, 1002
CMakeCache.txt, 1256
m_enum
MESS_datatype_t, 828
MESS_direct_cholpackage_t, 834
MESS_direct_lupackage_t, 836
MESS_equation_t, 851
MESS_memusage_t, 875
MESS_multidirect_t, 884
MESS_operation_t, 887
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_storage_t, 910
MARKED
dfs.c, 1326
newlu.c, 1773
MARK
dfs.c, 1326
newlu.c, 1773
MAT_INIT
tsia.c, 1891
tsiag.c, 1891
MATRIX_ADD
lrcf_adi/norms/res2.c, 1821
MAXIT
dense_nm.c, 1325

MESS64
interface_matlab.h, 1469
MESS_1_NORM
General Type Definitions and Macros, 47
MESS_2_NORM
General Type Definitions and Macros, 47
MESS_ARPACK_LIBRARIES
config.h, 1265
MESS_BLA_VENDOR
config.h, 1265
MESS_BLAS_LIBRARIES
config.h, 1265
MESS_BT_POSITION_VELOCITY
Balanced Truncation, 374
MESS_BT_POSITION
Balanced Truncation, 374
MESS_BT_VELOCITY_POSITION
Balanced Truncation, 374
MESS_BT_VELOCITY
Balanced Truncation, 374
MESS_CALL_IF_EXIST
Various Macros, 703
MESS_CLEAR_DIRECTS
Various Macros, 706
MESS_CLEAR_EQUATIONS
Various Macros, 706
MESS_CLEAR_MATRICES
Various Macros, 706
MESS_CLEAR_POINTERS
Various Macros, 707
MESS_CLEAR_VECTORS
Various Macros, 707
MESS_CMAKE_BUILD_TYPE
config.h, 1265
MESS_CMAKE_C_COMPILER
config.h, 1265
MESS_CMAKE_C_FLAGS_DEBUG
config.h, 1265
MESS_CMAKE_C_FLAGS_RELEASE
config.h, 1265
MESS_CMAKE_C_FLAGS
config.h, 1265
MESS_CMAKE_CXX_COMPILER
config.h, 1265
MESS_CMAKE_CXX_FLAGS_DEBUG
config.h, 1265
MESS_CMAKE_CXX_FLAGS_RELEASE
config.h, 1265
MESS_CMAKE_CXX_FLAGS
config.h, 1265
MESS_CMAKE_Fortran_COMPILER
config.h, 1265
MESS_CMAKE_Fortran_FLAGS_DEBUG
config.h, 1265
MESS_CMAKE_Fortran_FLAGS_RELEASE
config.h, 1265
MESS_CMAKE_Fortran_FLAGS
config.h, 1265
MESS_CMAKE_HOST_SYSTEM
config.h, 1265
MESS_CMAKE_VERSION
config.h, 1265
MESS_COMPLEX
General Type Definitions and Macros, 46
MESS_CONFIG_TIME
config.h, 1265
MESS_COORD
Data Structures, Types and Macros for Handling Matrices, 181
MESS_CSC
Data Structures, Types and Macros for Handling Matrices, 181
MESS_CSR
Data Structures, Types and Macros for Handling Matrices, 181
MESS_DENSE
Data Structures, Types and Macros for Handling Matrices, 181
MESS_DIRECT_BANDED_LU
Direct Solvers for Linear Systems, 248
MESS_DIRECT_CHOLMOD_CHOLESKY
Direct Solvers for Linear Systems, 248
MESS_DIRECT_CSPARSE_CHOLESKY
Direct Solvers for Linear Systems, 248
MESS_DIRECT_CSPARSE_LU
Direct Solvers for Linear Systems, 248
MESS_DIRECT_DEFAULT_CHOLESKY
Direct Solvers for Linear Systems, 248
MESS_DIRECT_DEFAULT_LU
Direct Solvers for Linear Systems, 248
MESS_DIRECT_LAPACK_CHOLESKY
Direct Solvers for Linear Systems, 248
MESS_DIRECT_LAPACK_LU
Direct Solvers for Linear Systems, 248
MESS_DIRECT_MKLPARDISO_LU
Direct Solvers for Linear Systems, 248
MESS_DIRECT_SPARSE_LU
Direct Solvers for Linear Systems, 248
MESS_DIRECT_SUPERLU_LU
Direct Solvers for Linear Systems, 248
MESS_DIRECT_UMFPACK_LU
Direct Solvers for Linear Systems, 248
MESS_DYNSYS_2ND
Interface for handling Dynamical Systems, 353
MESS_DYNSYS_GLTI
Interface for handling Dynamical Systems, 353
MESS_DYNSYS_LTI
Interface for handling Dynamical Systems, 353
MESS_EIGEN_ARPACK_BOTH_ENDS
Sparse Eigenvalue Solvers, 445
MESS_EIGEN_ARPACK_DEFAULT
Sparse Eigenvalue Solvers, 444
MESS_EIGEN_ARPACK_LARGE_ALGEBRAIC
Sparse Eigenvalue Solvers, 445
MESS_EIGEN_ARPACK_LARGE_IMAGPART
Sparse Eigenvalue Solvers, 445

MESS_EIGEN_ARPACK_LARGE_MAGNITUDE
Sparse Eigenvalue Solvers, 444
MESS_EIGEN_ARPACK_LARGE_REALPART
Sparse Eigenvalue Solvers, 444
MESS_EIGEN_ARPACK_SMALL_ALGEBRAIC
Sparse Eigenvalue Solvers, 445
MESS_EIGEN_ARPACK_SMALL_IMAGPART
Sparse Eigenvalue Solvers, 445
MESS_EIGEN_ARPACK_SMALL_MAGNITUDE
Sparse Eigenvalue Solvers, 444
MESS_EIGEN_ARPACK_SMALL_REALPART
Sparse Eigenvalue Solvers, 445
MESS_ENUM_CONVERTER
mess_enum_converter.c, 1637
MESS_ENUM_STR_1
gettypes.c, 1432
MESS_ENUM_STR_2
gettypes.c, 1432
MESS_ENUM_STR_3
gettypes.c, 1432
MESS_ENUM_STR_4
gettypes.c, 1432
MESS_ENUM_STR_5
gettypes.c, 1433
MESS_ENUM_STR_6
gettypes.c, 1433
MESS_ENUM_STR_7
gettypes.c, 1433
MESS_ENUM_STR_8
gettypes.c, 1433
MESS_ENUM_STR_9
gettypes.c, 1433
MESS_EQN_GLYAP
Generic Functions and Data Structures, 467
MESS_EQN_GRICCATI
Generic Functions and Data Structures, 467
MESS_EQN_LYAP
Generic Functions and Data Structures, 467
MESS_EQN_NONE
Generic Functions and Data Structures, 467
MESS_EQN_RICCATI
Generic Functions and Data Structures, 467
MESS_ERROR_ARGUMENTS
Error Codes and Handling, 606
MESS_ERROR_CHOLMOD
Error Codes and Handling, 606
MESS_ERROR_CONVERGE
Error Codes and Handling, 606
MESS_ERROR_CONVERT
Error Codes and Handling, 606
MESS_ERROR_DATATYPE
Error Codes and Handling, 606
MESS_ERROR_DATA
Error Codes and Handling, 606
MESS_ERROR_DIMENSION
Error Codes and Handling, 606
MESS_ERROR_DYNSYS
Error Codes and Handling, 606
MESS_ERROR_EIGENVALUES
Error Codes and Handling, 607
MESS_ERROR_FILEIO
Error Codes and Handling, 607
MESS_ERROR_GENERAL
Error Codes and Handling, 607
MESS_ERROR_LAPACK
Error Codes and Handling, 607
MESS_ERROR_MEMLOC
Error Codes and Handling, 607
MESS_ERROR_MEMORY
Error Codes and Handling, 607
MESS_ERROR_MISSING
Error Codes and Handling, 607
MESS_ERROR_MISC
Error Codes and Handling, 607
MESS_ERROR_MKLPARDISO
Error Codes and Handling, 607
MESS_ERROR_NOCOMPLEX
Error Codes and Handling, 607
MESS_ERROR_NOPATTERN
Error Codes and Handling, 607
MESS_ERROR_NOSUPPORT
Error Codes and Handling, 607
MESS_ERROR_NOTLOAD
Error Codes and Handling, 608
MESS_ERROR_NOTNULLPOINTER
Error Codes and Handling, 608
MESS_ERROR_NULLPOINTER
Error Codes and Handling, 608
MESS_ERROR_PYTHON_SIG
Error Codes and Handling, 608
MESS_ERROR_PYTHON
Error Codes and Handling, 608
MESS_ERROR_SINGULAR
Error Codes and Handling, 608
MESS_ERROR_STORAGETYPE
Error Codes and Handling, 608
MESS_ERROR_SYMMETRIC
Error Codes and Handling, 608
MESS_ERROR_UMFPACK
Error Codes and Handling, 608
MESS_ERROR_WRONG_HEADER
Error Codes and Handling, 608
MESS_ERROR
time.c, 1885
MESS_FREELIST_ADD_GENERIC
freelist.c, 1428
MESS_FROBENIUS_NORM
General Type Definitions and Macros, 47
MESS_GENERAL
Data Structures, Types and Macros for Handling Matrices, 181
MESS_GIT_BRANCH
config.h, 1265
MESS_GIT_COMMIT_HASH
config.h, 1265
MESS_H2NORM_FULL

H2 Model Order Reduction, 395
MESS_H2NORM_NO
H2 Model Order Reduction, 395
MESS_H2NORM_UPDATE
H2 Model Order Reduction, 396
MESS_HAVE_AMD
config.h, 1266
MESS_HAVE_BZIP2
config.h, 1266
MESS_HAVE_CHOLMOD
config.h, 1266
MESS_HAVE_CLOCK_GETTIME
config.h, 1266
MESS_HAVE_COLAMD
config.h, 1266
MESS_HAVE_CSPARSE
config.h, 1266
MESS_HAVE_DETC
Generic Interface to Direct Solvers, 237
MESS_HAVE_DET
Generic Interface to Direct Solvers, 237
MESS_HAVE_GETPERMP
Generic Interface to Direct Solvers, 238
MESS_HAVE_GETPERMQ
Generic Interface to Direct Solvers, 238
MESS_HAVE_GETSCALECOL
Generic Interface to Direct Solvers, 238
MESS_HAVE_GETSCALEROW
Generic Interface to Direct Solvers, 238
MESS_HAVE_GETL
Generic Interface to Direct Solvers, 237
MESS_HAVE_GETU
Generic Interface to Direct Solvers, 238
MESS_HAVE_INVERSE
Generic Interface to Direct Solvers, 238
MESS_HAVE_ISFINITE
config.h, 1266
MESS_HAVE_NEARBYINT
config.h, 1266
MESS_HAVE_PTRDIFF_T.c, 1645
info_size, 1645
main, 1645
SIZE, 1645
MESS_HAVE_PTRDIFF_T
config.h, 1266
MESS_HAVE_SIZE_T.c, 1646
info_size, 1646
main, 1646
SIZE, 1646
MESS_HAVE_SIZE_T
config.h, 1266
MESS_HAVE_SOLVEMH
Generic Interface to Direct Solvers, 238
MESS_HAVE_SOLVEMT
Generic Interface to Direct Solvers, 238
MESS_HAVE_SOLVEH
Generic Interface to Direct Solvers, 238
MESS_HAVE_SOLVEM
Generic Interface to Direct Solvers, 238
MESS_HAVE_SOLVET
Generic Interface to Direct Solvers, 238
MESS_HAVE_SOLVE
Generic Interface to Direct Solvers, 238
MESS_HAVE_STRNCMP
config.h, 1266
MESS_HAVE_STRNDUP
config.h, 1266
MESS_HAVE_UMFPACK
config.h, 1266
MESS_HAVE_XPM
config.h, 1266
MESS_HAVE_ZLIB
config.h, 1266
MESS_HERMITIAN
Data Structures, Types and Macros for Handling Matrices, 181
MESS_INCLUDE_DIR
config.h, 1266
MESS_INF_NORM
General Type Definitions and Macros, 47
MESS_INIT_DIRECTS
Various Macros, 708
MESS_INIT_EQUATIONS
Various Macros, 708
MESS_INIT_MATRICES
Various Macros, 708
MESS_INIT_VECTORS
Various Macros, 709
MESS_INTEGER
General Type Definitions and Macros, 45
MESS_IS_COMPLEX
General Type Definitions and Macros, 45
MESS_IS_COORD
Data Structures, Types and Macros for Handling Matrices, 178
MESS_IS_CSC
Data Structures, Types and Macros for Handling Matrices, 178
MESS_IS_CSR
Data Structures, Types and Macros for Handling Matrices, 178
MESS_IS_DENSE
Data Structures, Types and Macros for Handling Matrices, 178
MESS_IS_DYNSYS_2ND
Interface for handling Dynamical Systems, 353
MESS_IS_DYNSYS_GLTI
Interface for handling Dynamical Systems, 353
MESS_IS_DYNSYS_LTI
Interface for handling Dynamical Systems, 353
MESS_IS_DYNSYS_SISO
Interface for handling Dynamical Systems, 353
MESS_IS_GENERAL
Data Structures, Types and Macros for Handling Matrices, 178
MESS_IS_HERMITIAN

Data Structures, Types and Macros for Handling Matrices, 178
MESS_IS_INTEGER
General Type Definitions and Macros, 45
MESS_IS_REAL
General Type Definitions and Macros, 45
MESS_IS_SKEWSYMMETRIC
Data Structures, Types and Macros for Handling Matrices, 179
MESS_IS_SPARSE
Data Structures, Types and Macros for Handling Matrices, 179
MESS_IS_SYMMETRIC
Data Structures, Types and Macros for Handling Matrices, 179
MESS_IS_UNSYMMETRIC
Data Structures, Types and Macros for Handling Matrices, 179
MESS_LAPACK_LIBRARIES
config.h, 1266
MESS_LAPACK_LOGICAL
blas_defs.h, 994
MESS_LIBS
config.h, 1266
MESS_LRCFADI_PARA_ADAPTIVE_V
Generic Functions and Data Structures, 468
MESS_LRCFADI_PARA_ADAPTIVE_Z
Generic Functions and Data Structures, 468
MESS_LRCFADI_PARA_MINMAX_REAL
Generic Functions and Data Structures, 468
MESS_LRCFADI_PARA_MINMAX
Generic Functions and Data Structures, 468
MESS_LRCFADI_PARA_WACHSPRESS
Generic Functions and Data Structures, 468
MESS_MAJOR_VERSION
config.h, 1267
MESS_MAP_STATIC_FUNC2
matrix/map.c, 1612
vector/map.c, 1616
MESS_MAP_STATIC_FUNC3
matrix/map.c, 1612
vector/map.c, 1617
MESS_MAP_STATIC_FUNC
matrix/map.c, 1612
vector/map.c, 1616
MESS_MATLAB
interface_matlab.h, 1469
MESS_MATRIX_CHECKFORMAT
Data Structures, Types and Macros for Handling Matrices, 179
MESS_MATRIX_FILE_MAGIC_CHECK
Data Structures, Types and Macros for Handling Matrices, 180
MESS_MATRIX_FILE_MAGIC_SET
Data Structures, Types and Macros for Handling Matrices, 180
MESS_MATRIX_FILE_MAGIC
Data Structures, Types and Macros for Handling Matrices, 180
MESS_MATRIX_FILE_VERSION
Data Structures, Types and Macros for Handling Matrices, 180
MESS_MATRIX_RESET
Data Structures, Types and Macros for Handling Matrices, 180
MESS_MAX
Various Macros, 709
MESS_MEMORY_HIGH
Generic Functions and Data Structures, 468
MESS_MEMORY_LOW
Generic Functions and Data Structures, 468
MESS_MEMORY_MID
Generic Functions and Data Structures, 468
MESS_MINOR_VERSION
config.h, 1267
MESS_MIN
Various Macros, 709
MESS_MULTIDIRECT_SPARSE_LU
Generic Interface to Multisolvers, 319
MESS_MULTIDIRECT_UMFPACK_LU
Generic Interface to Multisolvers, 319
MESS_OFFSET
offsets.c, 1787
MESS_OP_HERMITIAN
General Type Definitions and Macros, 47
MESS_OP_NONE
General Type Definitions and Macros, 47
MESS_OP_TRANSPOSE
General Type Definitions and Macros, 47
MESS_PATCH_VERSION
config.h, 1267
MESS_PLOT_LIN
Plot Export, 665
MESS_PLOT_LOG
Plot Export, 665
MESS_PRECOND_DIAG
Types and Datastructures, 331
MESS_PRECOND_ILU0
Types and Datastructures, 331
MESS_PRECOND_ILUK
Types and Datastructures, 331
MESS_PRINT_FORMAT_LONG
General Type Definitions and Macros, 47
MESS_PRINT_FORMAT_SHORT
General Type Definitions and Macros, 47
MESS_PRINTF_INT2D
General Type Definitions and Macros, 46
MESS_PRINTF_INT
General Type Definitions and Macros, 45
MESS_Py_XDECREF
Interface to Python, 623
MESS_REAL
General Type Definitions and Macros, 46
MESS_REORDER_AMD
Various Other Operations on Matrices, 164

MESS_REORDER_COLAMD
Various Other Operations on Matrices, 164
MESS_REORDER_NONE
Various Other Operations on Matrices, 164
MESS_REORDER_RCM
Various Other Operations on Matrices, 164
MESS_RESET_MATRICES
Various Macros, 709
MESS_RESIDUAL_INDEFINITE
Generic Functions and Data Structures, 468
MESS_RESIDUAL_SPECTRAL
Generic Functions and Data Structures, 468
MESS_SIGN_SCALE_DET
Additional Matrix Equation Solvers, 554
MESS_SIGN_SCALE_FRO
Additional Matrix Equation Solvers, 554
MESS_SIGN_SCALE_NONE
Additional Matrix Equation Solvers, 554
MESS_SKEWSYMMETRIC
Data Structures, Types and Macros for Handling Matrices, 181
MESS_STORETYPE
General Type Definitions and Macros, 46
MESS_SUCCESS
Error Codes and Handling, 608
MESS_SUITESPARSE_INCLUDE_DIR
config.h, 1267
MESS_SUITESPARSE_LIBRARIES
config.h, 1267
MESS_SUPERLU_INCLUDE_DIR
config.h, 1267
MESS_SUPERLU_LIBRARIES
config.h, 1267
MESS_SYMMETRIC
Data Structures, Types and Macros for Handling Matrices, 181
MESS_SYMMETRY
General Type Definitions and Macros, 46
MESS_UNKNOWN
Data Structures, Types and Macros for Handling Matrices, 181
MESS_USE_OPENBLAS
config.h, 1267
MESS_VERSION
config.h, 1267
MESS_datatype_t, 827
byte_size, 828
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 828
install, 828
is_constant, 828
is_defined, 828
is_empty, 828
m_enum, 828
MESS_datatype_t, 828
operator mess_datatype_t, 828
operator!=, 828
operator==, 828
print, 828
print_as_scalar, 828
print_raw, 828
type_installed, 828
MESS_direct, 829
~MESS_direct, 830
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 832
get_ptr, 831
install, 831
is_constant, 831
is_defined, 831
is_empty, 831
MESS_direct, 830
operator bool, 831
operator mess_direct, 831
print, 831
print_as_scalar, 831
print_name_tag, 832
print_raw, 832
ptr, 832
type_installed, 832
MESS_direct.cc, 1632
DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA, 1633
MESS_direct.h, 1633
MESS_direct_cholpackage_t, 832
byte_size, 833
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 834
install, 833
is_constant, 833
is_defined, 833
is_empty, 833
m_enum, 834
MESS_direct_cholpackage_t, 833
operator mess_direct_cholpackage_t, 833
operator!=, 833
operator==, 834
print, 834
print_as_scalar, 834
print_raw, 834
type_installed, 834
MESS_direct_install.cc, 1633
MESS_direct_lupackage_t, 835
byte_size, 836
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 836
install, 836
is_constant, 836
is_defined, 836
is_empty, 836
m_enum, 836
MESS_direct_lupackage_t, 836
operator mess_direct_lupackage_t, 836
operator!=, 836
operator==, 836
print, 836

print_as_scalar, 836
print_raw, 836
type_installed, 837
MESS_enum.cc, 1633
DEFUN_DLD, 1635, 1636
OCTMESS_GENERATOR, 1634
MESS_enums.cc, 1637
DEFBINOP, 1639, 1640
DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA, 1640
OCTMESS_GENERATOR, 1639
op_eq, 1641
op_ne, 1641
MESS_enums.h, 1641
OCTMESS_GENERATOR, 1642
op_eq, 1642, 1643
op_ne, 1643
MESS_equation, 844
~MESS_equation, 845
DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA, 847
get_ptr, 846
install, 846
is_constant, 846
is_defined, 846
is_empty, 846
MESS_equation, 845
operator bool, 846
operator mess_equation, 846
print, 846
print_as_scalar, 846
print_name_tag, 847
print_raw, 847
ptr, 847
type_installed, 847
MESS_equation.cc, 1643
DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA, 1643
MESS_equation.h, 1643
MESS_equation_install.cc, 1645
MESS_equation_t, 850
byte_size, 851
DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA, 851
install, 851
is_constant, 851
is_defined, 851
is_empty, 851
m_enum, 851
MESS_equation_t, 851
operator mess_equation_t, 851
operator!=, 851
operator==, 851
print, 851
print_as_scalar, 851
print_raw, 851
type_installed, 852
MESS_error.h, 1645
OCTMESS_ERROR, 1645
MESS_matrix, 859
~MESS_matrix, 863
byte_size, 863
complex_matrix_value, 863
DECLARE_OV_TYPEID_FUNCTIONS_AND_DATA, 869
deepcopy, 863
dims, 864
full_value, 864
get_ptr, 865
install, 865
is_complex_matrix, 866
is_complex_type, 866
is_constant, 866
is_defined, 866
is_double_type, 866
is_empty, 866
is_matrix_type, 866
is_numeric_type, 866
is_real_matrix, 866
is_real_type, 866
is_single_type, 866
is_sparse_type, 866
length, 867
MESS_matrix, 861, 862
map, 867
matrix_value, 867
ndims, 867
operator bool, 868
operator mess_matrix, 868
print, 868
print_as_scalar, 868
print_name_tag, 868
print_raw, 868
ptr, 869
sparse_complex_matrix_value, 868
sparse_matrix_value, 868
type_installed, 869
MESS_matrix.cc, 1647
MESS_matrix.h, 1648
OCTMESS_GENERATOR, 1651
octmess_matrix_conv, 1652
op_add, 1652
op_herm_ldiv, 1652, 1653
op_herm_mul, 1653
op_hermitian, 1653
op_ldiv, 1653, 1654
op_mul, 1654–1657
op_mul_herm, 1657
op_mul_trans, 1657, 1658
op_not, 1658
op_sub, 1658, 1659
op_trans_ldiv, 1659
op_trans_mul, 1659, 1660
op_transpose, 1660
op_uminus, 1660
op_uplus, 1661

op_xldiv, 1661, 1662
op_xmul, 1662
MESS_matrix_install.cc, 1662
DEFBINOP, 1665–1670
DEFUNOP, 1671, 1672
OCTMESS_GENERATOR, 1665
MESS_matrix_ops_mm_generator.h, 1672
OCTMESS_GENERATOR, 1673
MESS_matrix_ops_mm_mm.cc, 1673
op_add, 1673
op_herm_ldiv, 1673
op_herm_mul, 1674
op_hermitian, 1674
op_ldiv, 1674
op_mul, 1675
op_mul_herm, 1675
op_mul_trans, 1675
op_not, 1676
op_sub, 1676
op_trans_ldiv, 1676
op_trans_mul, 1677
op_transpose, 1677
op_uminus, 1677
op_uplus, 1678
op_xldiv, 1678
op_xmul, 1678
MESS_matrix_ops_mm_ocm.cc, 1679
OCTMESS_GENERATOR, 1679
MESS_matrix_ops_mm_om.cc, 1679
OCTMESS_GENERATOR, 1679
MESS_matrix_ops_mm_os.cc, 1679
op_mul, 1680
MESS_matrix_ops_mm_osc.cc, 1680
op_mul, 1681
MESS_matrix_ops_mm_oscm.cc, 1681
OCTMESS_GENERATOR, 1681
MESS_matrix_ops_mm_osm.cc, 1681
OCTMESS_GENERATOR, 1682
MESS_memusage_t, 873
byte_size, 874
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 875
install, 874
is_constant, 874
is_defined, 874
is_empty, 874
m_enum, 875
MESS_memusage_t, 874
operator mess_memusage_t, 874
operator!=, 874
operator==, 875
print, 875
print_as_scalar, 875
print_raw, 875
type_installed, 875
MESS_misc.cc, 1682
DEFUN_DLD, 1682–1684
MESS_multidirect, 875
~MESS_multidirect, 877
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 879
get_ptr, 878
install, 878
is_constant, 878
is_defined, 878
is_empty, 878
MESS_multidirect, 877
operator bool, 878
operator mess_multidirect, 878
print, 878
print_as_scalar, 878
print_name_tag, 879
print_raw, 879
ptr, 879
type_installed, 879
MESS_multidirect.cc, 1684
DEFINE_OV_TYPEID_FUNCTIONS_AND_DATA, 1684
MESS_multidirect.h, 1684
MESS_multidirect_install.cc, 1684
MESS_multidirect_t, 882
byte_size, 883
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 884
install, 883
is_constant, 883
is_defined, 883
is_empty, 883
m_enum, 884
MESS_multidirect_t, 883
operator mess_multidirect_t, 883
operator!=, 883
operator==, 884
print, 884
print_as_scalar, 884
print_raw, 884
type_installed, 884
MESS_operation_t, 885
byte_size, 887
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 887
install, 887
is_constant, 887
is_defined, 887
is_empty, 887
m_enum, 887
MESS_operation_t, 887
operator mess_operation_t, 887
operator!=, 887
operator==, 887
print, 887
print_as_scalar, 887
print_raw, 887
type_installed, 887
MESS_options, 887
~MESS_options, 889

byte_size, 890
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 892
deepcopy, 890
get_ptr, 890
install, 890
is_constant, 890
is_defined, 890
is_empty, 890
is_map, 891
MESS_options, 889
map_keys, 891
operator bool, 891
operator mess_options, 891
print, 891
print_as_scalar, 891
print_name_tag, 891
print_raw, 891
ptr, 892
subsasgn, 891
subsref, 891
type_installed, 892
MESS_options.cc, 1685
MESS_options.h, 1686
MESS_options_install.cc, 1686
MESS_parameter_t, 897
byte_size, 898
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 898
install, 898
is_constant, 898
is_defined, 898
is_empty, 898
m_enum, 898
MESS_parameter_t, 898
operator mess_parameter_t, 898
operator!=, 898
operator==, 898
print, 898
print_as_scalar, 898
print_raw, 898
type_installed, 898
MESS_residual_t, 900
byte_size, 901
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 901
install, 901
is_constant, 901
is_defined, 901
is_empty, 901
m_enum, 901
MESS_residual_t, 901
operator mess_residual_t, 901
operator!=, 901
operator==, 901
print, 901
print_as_scalar, 901
print_raw, 901
type_installed, 901
MESS_status, 902
~MESS_status, 903
byte_size, 904
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 905
deepcopy, 904
get_ptr, 904
install, 904
is_constant, 904
is_defined, 904
is_empty, 904
MESS_status, 903
operator bool, 904
operator mess_status, 904
print, 904, 905
print_as_scalar, 905
print_name_tag, 905
print_raw, 905
ptr, 905
type_installed, 905
MESS_status.cc, 1687
MESS_status.h, 1688
MESS_status_install.cc, 1688
MESS_storage_t, 908
byte_size, 909
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 910
install, 909
is_constant, 909
is_defined, 909
is_empty, 909
m_enum, 910
MESS_storage_t, 909
operator mess_storage_t, 909
operator!=, 909
operator==, 910
print, 910
print_as_scalar, 910
print_raw, 910
type_installed, 910
MESS_to_oct.cc, 1688
DEFUN_DLD, 1689
MESS_vector, 914
~MESS_vector, 918
byte_size, 918
DECLARE_OV_TYPEID_FUNCTIONS_AND_D↩ ATA, 921
deepcopy, 918
dims, 919
get_ptr, 919
install, 919
is_complex_matrix, 919
is_complex_type, 920
is_constant, 920
is_defined, 920
is_double_type, 920
is_empty, 920

is_matrix_type, 920
is_numeric_type, 920
is_real_matrix, 920
is_real_type, 920
is_single_type, 920
length, 920
MESS_vector, 917, 918
map, 920
ndims, 920
operator bool, 921
operator mess_vector, 921
print, 921
print_as_scalar, 921
print_name_tag, 921
print_raw, 921
ptr, 921
type_installed, 921
MESS_vector.cc, 1689
MESS_vector.h, 1690
OCTMESS_GENERATOR, 1692
op_add, 1692
op_herm_ldiv, 1692, 1693
op_herm_mul, 1693
op_ldiv, 1693, 1694
op_mul, 1694–1696
op_not, 1696
op_sub, 1697
op_trans_ldiv, 1697, 1698
op_trans_mul, 1698, 1699
op_uminus, 1699
op_uplus, 1699
op_xldiv, 1699, 1700
op_xmul, 1700
MESS_vector_install.cc, 1700
DEFBINOP, 1702–1706
DEFUNOP, 1706
OCTMESS_GENERATOR, 1701
MESS_vector_ops_mm_mv.cc, 1707
op_herm_ldiv, 1707
op_herm_mul, 1707
op_ldiv, 1708
op_mul, 1708
op_trans_ldiv, 1708
op_trans_mul, 1709
op_xldiv, 1709
op_xmul, 1709
MESS_vector_ops_mv_generator.h, 1710
OCTMESS_GENERATOR, 1710
MESS_vector_ops_mv_mv.cc, 1711
op_add, 1711
op_herm_mul, 1712
op_not, 1712
op_sub, 1713
op_trans_mul, 1713
op_uminus, 1714
op_uplus, 1714
MESS_vector_ops_mv_os.cc, 1714
op_mul, 1714, 1715
MESS_vector_ops_mv_osc.cc, 1715
op_mul, 1715, 1716
MESS_vector_ops_ocm_mv.cc, 1716
OCTMESS_GENERATOR, 1716
MESS_vector_ops_om_mv.cc, 1716
OCTMESS_GENERATOR, 1717
MESS_vector_ops_oscm_mv.cc, 1717
OCTMESS_GENERATOR, 1717
MESS_vector_ops_osm_mv.cc, 1717
OCTMESS_GENERATOR, 1717
MEXMESS_GENERATOR
enum_conversions.c, 1344
matlab/c_interface/misc/misc.c, 1724
MIN_LINESEARCH_PARA
dense_nm.c, 1325
MIN
eye.c, 1425
MM_BANNER
read.c, 1812
vector_read.c, 1972
MM_STR_ARR
read.c, 1812
MM_STR_COORD
read.c, 1812
MM_STR_CPX
read.c, 1812
vector_read.c, 1972
MM_STR_GE
read.c, 1812
MM_STR_HER
read.c, 1812
MM_STR_INT
read.c, 1812
vector_read.c, 1972
MM_STR_MATRIX
read.c, 1812
MM_STR_PAT
read.c, 1812
MM_STR_REAL
read.c, 1812
vector_read.c, 1972
MM_STR_SKSYM
read.c, 1812
MM_STR_SYM
read.c, 1812
MM_STR_VECTOR
vector_read.c, 1972
MSG_ERROR
Error Macros, 610
MSG_FNAME
Error Macros, 611
MSG_INFO
Error Macros, 611
MSG_PRINT
Error Macros, 611
MSG_WARN_NOBT
Error Macros, 611
MSG_WARN

Error Macros, 611
MY_MODULE_mp_MY_SUB-UPPER.c, 1765, 1766
MY_MODULE_mp_MY_SUB
symbols/MY_MODULE_mp_MY_SUB-UPPER.c, 1766
VerifyC/CMakeFiles/FortranCInterface/symbols/↩ MY_MODULE_mp_MY_SUB-UPPER.c, 1766
MY_SUB-UPPER.c, 1767, 1768
MY_SUB
symbols/MY_SUB-UPPER.c, 1768
VerifyC/CMakeFiles/FortranCInterface/symbols/↩ MY_SUB-UPPER.c, 1768
MYMODULE_mp_MYSUB-UPPER.c, 1769
MYMODULE_mp_MYSUB
symbols/MYMODULE_mp_MYSUB-UPPER.↩ c, 1769
VerifyC/CMakeFiles/FortranCInterface/symbols/↩ MYMODULE_mp_MYSUB-UPPER.c, 1769
MYSUB-UPPER.c, 1771
MYSUB
symbols/MYSUB-UPPER.c, 1771
VerifyC/CMakeFiles/FortranCInterface/symbols/↩ MYSUB-UPPER.c, 1771
magic
mess_matrix_file_st, 870
main
CMakeCCompilerId.c, 1258
CMakeCXXCompilerId.cpp, 1260
check_add.c, 1021
check_addcol2p.c, 1022
check_any.c, 1024
check_arnoldi_template.c, 1025
check_arnoldi_template_nrm.c, 1027
check_arpack.c, 1030
check_arpack_lanczos.c, 1033
check_backslash.c, 1037
check_bandwidth.c, 1038
check_ber_sign_fac.c, 1039
check_biorth.c, 1041
check_cat.c, 1044
check_cholmod.c, 1048
check_cimport.c, 1050
check_cimport_c2d.c, 1052
check_colaxpy2.c, 1053
check_coldot.c, 1054
check_coldotE.c, 1059
check_coldotc.c, 1056
check_coldotcE.c, 1057
check_colnorm.c, 1060
check_colscale.c, 1062
check_colsub.c, 1064
check_colvecaxpy.c, 1066
check_colvecdot.c, 1068
check_colvecdotc.c, 1070
check_condest.c, 1071
check_convert.c, 1074
check_dae2_handles_adi.c, 1076
check_dae2_handles_para.c, 1078
check_decomp.c, 1080
check_dense_nm.c, 1081
check_dense_nm_tridiag.c, 1083
check_det.c, 1084
check_dgeqp3.c, 1085
check_diag.c, 1087
check_direct_care.c, 1089
check_dynorm2.c, 1090
check_dynorm2t.c, 1091
check_dynorm_indefinite2.c, 1092
check_dynorm_indefinitef.c, 1095
check_dynormf.c, 1097
check_dynormft.c, 1098
check_eig.c, 1100
check_eig2.c, 1103
check_eig_complex.c, 1106
check_eig_hessenberg_schur.c, 1108
check_eig_hessenberg_schur_largest_ev.c, 1109
check_eliminate_zeros.c, 1111
check_eps.c, 1113
check_fputs_block.c, 1114
check_freelist.c, 1114
check_fro_inn.c, 1116
check_gaxpy.c, 1117
check_getLU.c, 1120
check_getelement.c, 1119
check_gsign.c, 1124
check_hessenberg.c, 1126
check_itsolver.c, 1128
check_kron.c, 1130
check_lapackqr.c, 1135
check_lift.c, 1138
check_linesearch.c, 1139
check_lradi_dae1.c, 1140
check_lradi_dae2.c, 1142
check_lradi_glyap.c, 1143
check_lradi_so.c, 1145
check_lradi_unsym.c, 1147
check_lrnm_dae1.c, 1149
check_lrnm_dae2.c, 1151
check_lrnm_g.c, 1152
check_lrnm_so.c, 1154
check_lrnm_unsym.c, 1156
check_lyap_sign_fac.c, 1158
check_matrix_biorth.c, 1160
check_matrix_colaxpy.c, 1162
check_matrix_gbiorth.c, 1163
check_matrix_null.c, 1165
check_matrix_sum.c, 1166
check_matrixio.c, 1168
check_mgs.c, 1169
check_mgs_inplace.c, 1171
check_mgsadd.c, 1173
check_mul.c, 1175
check_mul_small.c, 1182
check_mul_tall.c, 1188
check_mulnormf2.c, 1192
check_mvp.c, 1193

check_mvpcall_matrix.c, 1194
check_norm.c, 1196
check_one_value.c, 1198
check_orth.c, 1200
check_perm.c, 1202
check_pinv.c, 1207
check_precond.c, 1210
check_proj_sym.c, 1212
check_rank.c, 1214
check_real_imag_part.c, 1215
check_rowsub.c, 1216
check_scalem.c, 1218
check_schur.c, 1219
check_set_get_col_row.c, 1220
check_setelement.c, 1222
check_sign.c, 1224
check_so1_handles.c, 1226
check_so2_handles.c, 1228
check_sub.c, 1230
check_svd.c, 1232
check_svd_complex.c, 1233
check_svd_econ.c, 1235
check_svd_econ_complex.c, 1237
check_trace.c, 1239
check_transpose.c, 1240
check_tridiag.c, 1242
check_trisolve.c, 1244
check_vector_perm.c, 1248
check_vectorio.c, 1250
DCPX_T.c, 1321
feature_tests.c, 1426
feature_tests.cxx, 1426
glyap_test.c, 1447
gstein_test.c, 1450
inverse.c, 1470
linsolve.c, 1575
linsolve_multi.c, 1581
linsolvem.c, 1587
linsolvem_multi.c, 1594
MESS_HAVE_PTRDIFF_T.c, 1645
MESS_HAVE_SIZE_T.c, 1646
superlu_refinement_bug.c, 1862
superlu_refinement_bug_43.c, 1862
superlu_refinement_bug_50.c, 1862
TestRestrict.c, 1883
tests/direct/sylvester_dense.c, 1872
tests/direct/sylvester_sparsedense.c, 1881
malloc.c, 1609
calloc_call_t, 1610
fn_calloc, 1610
fn_free, 1610
fn_malloc, 1610
fn_realloc, 1610
free_call_t, 1610
malloc_call_t, 1610
realloc_call_t, 1610
malloc_call_t
malloc.c, 1610
map
MESS_matrix, 867
MESS_vector, 920
map.c, 1610, 1614
map_abs_cpx
matrix/map.c, 1613
vector/map.c, 1617
map_abs_real
matrix/map.c, 1613
vector/map.c, 1617
map_acos_cpx
matrix/map.c, 1613
vector/map.c, 1617
map_acos_real
matrix/map.c, 1613
vector/map.c, 1617
map_acosh_cpx
matrix/map.c, 1613
vector/map.c, 1617
map_acosh_real
matrix/map.c, 1613
vector/map.c, 1617
map_arg_cpx
matrix/map.c, 1613
vector/map.c, 1617
map_arg_real
matrix/map.c, 1613
vector/map.c, 1617
map_asin_cpx
matrix/map.c, 1613
vector/map.c, 1617
map_asin_real
matrix/map.c, 1613
vector/map.c, 1617
map_asinh_cpx
matrix/map.c, 1613
vector/map.c, 1617
map_asinh_real
matrix/map.c, 1613
vector/map.c, 1617
map_atan_cpx
matrix/map.c, 1613
vector/map.c, 1617
map_atan_real
matrix/map.c, 1613
vector/map.c, 1618
map_atanh_cpx
matrix/map.c, 1613
vector/map.c, 1618
map_atanh_real
matrix/map.c, 1613
vector/map.c, 1618
map_ceil_cpx
matrix/map.c, 1613
vector/map.c, 1618
map_ceil_real
matrix/map.c, 1613
vector/map.c, 1618

map_conj_cpx
matrix/map.c, 1613
vector/map.c, 1618
map_conj_real
matrix/map.c, 1613
vector/map.c, 1618
map_cos_cpx
matrix/map.c, 1613
vector/map.c, 1618
map_cos_real
matrix/map.c, 1613
vector/map.c, 1618
map_cosh_cpx
matrix/map.c, 1613
vector/map.c, 1618
map_cosh_real
matrix/map.c, 1613
vector/map.c, 1618
map_exp_cpx
matrix/map.c, 1613
vector/map.c, 1618
map_exp_real
matrix/map.c, 1614
vector/map.c, 1618
map_expm1_cpx
matrix/map.c, 1614
vector/map.c, 1618
map_expm1_real
matrix/map.c, 1614
vector/map.c, 1618
map_floor_cpx
matrix/map.c, 1614
vector/map.c, 1618
map_floor_real
matrix/map.c, 1614
vector/map.c, 1618
map_isfinite_cpx
matrix/map.c, 1614
vector/map.c, 1618
map_isfinite_real
matrix/map.c, 1614
vector/map.c, 1618
map_isinf_cpx
matrix/map.c, 1614
vector/map.c, 1618
map_isinf_real
matrix/map.c, 1614
vector/map.c, 1618
map_isnan_cpx
matrix/map.c, 1614
vector/map.c, 1618
map_isnan_real
matrix/map.c, 1614
vector/map.c, 1618
map_keys
MESS_options, 891
map_log_cpx
matrix/map.c, 1614
vector/map.c, 1618
map_log_real
matrix/map.c, 1614
vector/map.c, 1618
map_not_cpx
matrix/map.c, 1614
vector/map.c, 1619
map_not_real
matrix/map.c, 1614
vector/map.c, 1619
map_round_cpx
matrix/map.c, 1614
vector/map.c, 1619
map_round_real
matrix/map.c, 1614
vector/map.c, 1619
map_sin_cpx
matrix/map.c, 1614
vector/map.c, 1619
map_sin_real
matrix/map.c, 1614
vector/map.c, 1619
map_sinh_cpx
matrix/map.c, 1614
vector/map.c, 1619
map_sinh_real
matrix/map.c, 1614
vector/map.c, 1619
map_sqrt_cpx
matrix/map.c, 1614
vector/map.c, 1619
map_sqrt_real
matrix/map.c, 1614
vector/map.c, 1619
map_tan_cpx
matrix/map.c, 1614
vector/map.c, 1619
map_tan_real
matrix/map.c, 1614
vector/map.c, 1619
map_tanh_cpx
matrix/map.c, 1614
vector/map.c, 1619
map_tanh_real
matrix/map.c, 1614
vector/map.c, 1619
map_to_win
x11init.c, 1979
mat
__ilu0_data, 775
mat_diff, 822
A, 822
B, 822
mat_diff_mvp
equal.c, 1345
matlab/c_interface/misc/misc.c
MEXMESS_GENERATOR, 1724
matlab_calloc

interface_matlab.c, 1466
matlab_free
interface_matlab.c, 1466
matlab_malloc
interface_matlab.c, 1466
matlab_realloc
interface_matlab.c, 1466
matrices
mess_freelist_st, 853
matrix
__presordata, 793
Matrix Decompositions, 297
mess_direct_cholfactor, 297
mess_direct_dgeqp3, 298
Matrix Generators, 598
Matrix Operations, 48
matrix.c, 1619
ConvStringtoC, 1620
NO_IMPORT_ARRAY, 1620
NPY_ARRAY_OWNDATA, 1620
NPY_NO_DEPRECATED_API, 1620
PY_ARRAY_UNIQUE_SYMBOL, 1620
PyArray_CALL_DATA, 1620
PyArray_CALL_ISSIGNED, 1620
PyArray_CALL_ISUNSIGNED, 1620
PyArray_ENABLEFLAGS, 1620
matrix.h, 1620
matrix/alloc.c
__mess_matrix_alloc_setld, 960
__select_ld, 961
ld_multiplicity, 961
NEW_LD, 960
matrix/cat.c
__mess_matrix_cat_LABC, 1016
__mess_matrix_cat_LACD, 1017
__mess_matrix_cat_RABD, 1018
__mess_matrix_cat_RBCD, 1019
__mess_matrix_cat_antidiag, 1012
__mess_matrix_cat_diag, 1013
__mess_matrix_cat_firstcol, 1013
__mess_matrix_cat_firstrow, 1014
__mess_matrix_cat_full, 1015
matrix/convert.c
__conv_coord_csc, 1268
__conv_coord_csr, 1268
__conv_coord_dense, 1269
__conv_csc_coord, 1269
__conv_csc_dense, 1270
__conv_csr_coord, 1271
__conv_csr_dense, 1271
__conv_dense_coord, 1272
__conv_dense_csc, 1273
__conv_dense_csr, 1273
matrix/map.c
MESS_MAP_STATIC_FUNC2, 1612
MESS_MAP_STATIC_FUNC3, 1612
MESS_MAP_STATIC_FUNC, 1612
map_abs_cpx, 1613
map_abs_real, 1613
map_acos_cpx, 1613
map_acos_real, 1613
map_acosh_cpx, 1613
map_acosh_real, 1613
map_arg_cpx, 1613
map_arg_real, 1613
map_asin_cpx, 1613
map_asin_real, 1613
map_asinh_cpx, 1613
map_asinh_real, 1613
map_atan_cpx, 1613
map_atan_real, 1613
map_atanh_cpx, 1613
map_atanh_real, 1613
map_ceil_cpx, 1613
map_ceil_real, 1613
map_conj_cpx, 1613
map_conj_real, 1613
map_cos_cpx, 1613
map_cos_real, 1613
map_cosh_cpx, 1613
map_cosh_real, 1613
map_exp_cpx, 1613
map_exp_real, 1614
map_expm1_cpx, 1614
map_expm1_real, 1614
map_floor_cpx, 1614
map_floor_real, 1614
map_isfinite_cpx, 1614
map_isfinite_real, 1614
map_isinf_cpx, 1614
map_isinf_real, 1614
map_isnan_cpx, 1614
map_isnan_real, 1614
map_log_cpx, 1614
map_log_real, 1614
map_not_cpx, 1614
map_not_real, 1614
map_round_cpx, 1614
map_round_real, 1614
map_sin_cpx, 1614
map_sin_real, 1614
map_sinh_cpx, 1614
map_sinh_real, 1614
map_sqrt_cpx, 1614
map_sqrt_real, 1614
map_tan_cpx, 1614
map_tan_real, 1614
map_tanh_cpx, 1614
map_tanh_real, 1614
matrix/norm.c
norm2mvp, 1785
norminv2mvp, 1785
matrix/perm.c
__mess_matrix_perm_csc, 1793
__mess_matrix_perm_csr, 1794
__mess_matrix_perm_dense, 1794

ex_col_cpx, 1795
ex_col_real, 1795
perm_inv, 1796
matrix/rand.c
__drand, 1808
IDX, 1808
matrix_ANAH
check_mul_small.c, 1183
matrix_ANAiH
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_ANAiT
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_ANAT
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_ANBiN
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_ANBN
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_Ai
check_mul_small.c, 1183
check_mul_tall.c, 1189
matrix_AiNAH
check_mul_small.c, 1183
check_mul_tall.c, 1189
matrix_AiNAiH
check_mul_small.c, 1183
check_mul_tall.c, 1189
matrix_AiNAiT
check_mul_small.c, 1183
check_mul_tall.c, 1189
matrix_AiNAT
check_mul_small.c, 1183
check_mul_tall.c, 1189
matrix_AiNBiN
check_mul_small.c, 1183
check_mul_tall.c, 1189
matrix_AiNBN
check_mul_small.c, 1183
check_mul_tall.c, 1189
matrix_AS
check_proj_sym.c, 1213
matrix_AT
check_transpose.c, 1241
matrix_BHAH
check_mul_small.c, 1184
matrix_BHAiH
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_BHAiT
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_BHAT
check_mul_small.c, 1184
matrix_BHBiN
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_BHBN
check_mul_small.c, 1184
matrix_BTAH
check_mul_small.c, 1185
matrix_BTAiH
check_mul_small.c, 1185
check_mul_tall.c, 1191
matrix_BTAiT
check_mul_small.c, 1185
check_mul_tall.c, 1191
matrix_BTAT
check_mul_small.c, 1185
check_mul_tall.c, 1191
matrix_BTBiN
check_mul_small.c, 1185
check_mul_tall.c, 1191
matrix_BTBN
check_mul_small.c, 1185
check_mul_tall.c, 1191
matrix_Bi
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_BiHAH
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_BiHAiH
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_BiHAiT
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_BiHAT
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_BiHBiN
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_BiHBN
check_mul_small.c, 1184
check_mul_tall.c, 1191
matrix_BiTAH
check_mul_small.c, 1184
check_mul_tall.c, 1191
matrix_BiTAiH
check_mul_small.c, 1184
check_mul_tall.c, 1191
matrix_BiTAiT
check_mul_small.c, 1184
check_mul_tall.c, 1191
matrix_BiTAT
check_mul_small.c, 1184
check_mul_tall.c, 1191
matrix_BiTBiN
check_mul_small.c, 1184
check_mul_tall.c, 1191

matrix_BiTBN
check_mul_small.c, 1185
check_mul_tall.c, 1191
matrix_HiHi
check_mul.c, 1176
matrix_HiNi
check_mul.c, 1176
matrix_HiTi
check_mul.c, 1177
matrix_HiN
check_mul.c, 1176
matrix_HiT
check_mul.c, 1176
matrix_NHi
check_mul.c, 1177
matrix_NNi
check_mul.c, 1178
matrix_NTi
check_mul.c, 1178
matrix_NiHi
check_mul.c, 1177
matrix_NiNi
check_mul.c, 1177
matrix_NiTi
check_mul.c, 1178
matrix_NiN
check_mul.c, 1177
matrix_NiT
check_mul.c, 1177
matrix_NN
check_mul.c, 1178
matrix_NT
check_mul.c, 1178
matrix_THi
check_mul.c, 1178
matrix_TNi
check_mul.c, 1179
matrix_TTi
check_mul.c, 1180
matrix_TiHi
check_mul.c, 1178
matrix_TiNi
check_mul.c, 1179
matrix_TiTi
check_mul.c, 1179
matrix_TiN
check_mul.c, 1179
matrix_TiT
check_mul.c, 1179
matrix_TN
check_mul.c, 1179
matrix_TT
check_mul.c, 1179
matrix_A
check_mul_small.c, 1183
check_mul_tall.c, 1189
check_proj_sym.c, 1213
check_transpose.c, 1241
matrix_B
check_mul_small.c, 1184
check_mul_tall.c, 1190
matrix_data
check_coldot.c, 1055
check_coldotc.c, 1056
check_colnorm.c, 1061
check_colvecdot.c, 1069
check_diag.c, 1088
check_mul.c, 1176
matrix_data1
check_fro_inn.c, 1116
matrix_data2
check_fro_inn.c, 1116
matrix_datac
check_colvecdotc.c, 1070
check_diag.c, 1088
matrix_datar
check_colvecdotc.c, 1070
matrix_print.c, 1629
matrix_to_c
Interface to Python, 635
matrix_to_python
Interface to Python, 635
matrix_value
MESS_matrix, 867
matrixi_data
check_mul.c, 1180
max
iluk.c, 1462
max_size
mess_threadpool_st, 911
maxfct
mklpardiso_solver, 924
maxit
__mess_solver_options, 790
mess_eigen_arpack_options_t, 843
mess_h2_options_st, 854
memory_usage
mess_options_st, 895
memsize
mess_multidirect_st, 881
memsize.c, 1630
mess.h, 1630
mess_bt_chooseorder_default
Balanced Truncation, 375
mess_bt_chooseorder_func
Balanced Truncation, 374
mess_bt_chooseorder_minreal
Balanced Truncation, 375
mess_bt_getsotypestr
Balanced Truncation, 376
mess_bt_gglrsrm
Balanced Truncation, 376
mess_bt_gslrsrm
Balanced Truncation, 378
mess_bt_lrsrm
Balanced Truncation, 380

mess_bt_lrsrm_somor
Balanced Truncation, 382
mess_bt_lrsrm_somor_so
Balanced Truncation, 384
mess_bt_lrsrm_somor_so_gram
Balanced Truncation, 386
mess_bt_options
Balanced Truncation, 374
mess_bt_options_clear
Balanced Truncation, 387
mess_bt_options_init
Balanced Truncation, 389
mess_bt_options_print
Balanced Truncation, 389
mess_bt_options_st, 823
chooseorder, 824
chooseorder_aux, 824
rdim, 824
so_type, 824
tol, 824
mess_bt_status
Balanced Truncation, 375
mess_bt_status_clear
Balanced Truncation, 389
mess_bt_status_init
Balanced Truncation, 390
mess_bt_status_print
Balanced Truncation, 390
mess_bt_status_st, 824
esterror, 826
rdim, 826
statB, 826
statC, 826
time, 826
time_VW, 826
time_lyap, 826
mess_call.c, 1631
mess_call_func, 1631
mess_call_t, 1631
mess_calls, 1632
mexFunction, 1631
mess_call_func
mess_call.c, 1631
mess_call_t
mess_call.c, 1631
mess_callback_equation_mexmess
Interfaces_matlab, 740
mess_calls
mess_call.c, 1632
mess_care
Easy Frontends to Main Functions, 585
mess_check_complex
Various Macros, 703
mess_check_coord
Various Macros, 703
mess_check_csc
Various Macros, 703
mess_check_csr
Various Macros, 703
mess_check_datatype
Various Macros, 703
mess_check_dense
Various Macros, 703
mess_check_nocoord
Various Macros, 703
mess_check_nonnegative
Various Macros, 704
mess_check_notnullpointer
Various Macros, 704
mess_check_nullpointer
Various Macros, 704
mess_check_operation_type
Various Macros, 704
mess_check_positive
Various Macros, 704
mess_check_real
Various Macros, 704
mess_check_real_or_complex
Various Macros, 704
mess_check_same_cols
Various Macros, 704
mess_check_same_colsdim
Various Macros, 704
mess_check_same_colsrows
Various Macros, 705
mess_check_same_datatype
Various Macros, 705
mess_check_same_dim
Various Macros, 705
mess_check_same_rows
Various Macros, 705
mess_check_same_rowsdim
Various Macros, 705
mess_check_same_size
Various Macros, 705
mess_check_same_storetype
Various Macros, 705
mess_check_sparse
Various Macros, 705
mess_check_square
Various Macros, 705
mess_check_storage_type
Various Macros, 706
mess_check_struct
Interfaces_matlab, 730
mess_check_true
Various Macros, 706
mess_classname_from_mexmess
Interfaces_matlab, 741
mess_classname_from_mexmess.c, 1632
mess_complex_from_mexmess
Interfaces_matlab, 742
mess_ctime
Time Measurement, 697
mess_datatype_t
General Type Definitions and Macros, 46

mess_datatype_t_str
Miscellaneous Helper Functions, 676
mess_decomp_lureuse_kernelcsr
Various Other Functions related to Direct Solvers., 315
mess_decomp_lureuse_kernelcsr_complex
Various Other Functions related to Direct Solvers., 315
mess_decomp_lureuse_kernelcsr_flopcount
Various Other Functions related to Direct Solvers., 315
mess_dense_nm_gmare
Additional Matrix Equation Solvers, 555
mess_dense_nm_gmpare
Additional Matrix Equation Solvers, 556
mess_dense_nm_gpare
Additional Matrix Equation Solvers, 558
mess_dense_res_gmpare
Additional Matrix Equation Solvers, 559
mess_direct
Generic Interface to Direct Solvers, 238
mess_direct_care
Direct Solvers for Matrix Equations, 288
mess_direct_care_res2
Residual Computations, 301
mess_direct_care_sign
Direct Solvers for Matrix Equations, 290
mess_direct_chol
Direct Solvers for Linear Systems, 248
mess_direct_chol_select
Direct Solvers for Linear Systems, 249
mess_direct_cholfactor
Matrix Decompositions, 297
mess_direct_cholpackage_t
Direct Solvers for Linear Systems, 248
mess_direct_cholpackage_t_from_mexmess
Interfaces_matlab, 742
mess_direct_cholpackage_t_str
Miscellaneous Helper Functions, 677
mess_direct_cholpackage_t_to_mexmess
Interfaces_matlab, 742
mess_direct_clear
Generic Interface to Direct Solvers, 238
mess_direct_create_banded
Direct Solvers for Linear Systems, 250
mess_direct_create_bicgstab
Direct Solvers for Linear Systems, 251
mess_direct_create_cholesky
Direct Solvers for Linear Systems, 252
mess_direct_create_csparse_cholesky
Direct Solvers for Linear Systems, 253
mess_direct_create_csparse_lu
Direct Solvers for Linear Systems, 254
mess_direct_create_generalized_lyapunov
Direct Solvers for Matrix Equations, 290
mess_direct_create_generalized_lyapunovchol
Direct Solvers for Matrix Equations, 291
mess_direct_create_generalized_stein
Direct Solvers for Matrix Equations, 292
mess_direct_create_glyap3
glyap3.c, 1443
mess_direct_create_hessenberg_lu
Direct Solvers for Linear Systems, 256
mess_direct_create_lapack_lu
Direct Solvers for Linear Systems, 257
mess_direct_create_lapack_qr
Direct Solvers for Linear Systems, 258
mess_direct_create_lapack_qr_over
lapack_qr.c, 1500
mess_direct_create_lapack_qr_under
lapack_qr.c, 1501
mess_direct_create_mklpardiso
mklpardiso.c, 1731
mess_direct_create_mklpardiso_control
mklpardiso.c, 1732
mess_direct_create_sparse_lu
Direct Solvers for Linear Systems, 259
mess_direct_create_sylvester_dense
Direct Solvers for Matrix Equations, 293
mess_direct_create_sylvester_sparsedense
Direct Solvers for Matrix Equations, 294
mess_direct_create_umfpack
Direct Solvers for Linear Systems, 260
mess_direct_create_umfpack_control
Direct Solvers for Linear Systems, 261
mess_direct_determinant
Generic Interface to Direct Solvers, 239
mess_direct_determinantc
Generic Interface to Direct Solvers, 239
mess_direct_dgeqp3
Matrix Decompositions, 298
mess_direct_generalized_lyapunov_res2
Residual Computations, 301
mess_direct_generalized_stein_res2
Residual Computations, 303
mess_direct_generalized_sylvester_res2
Residual Computations, 304
mess_direct_getL
Generic Interface to Direct Solvers, 239
mess_direct_getpermp
Generic Interface to Direct Solvers, 240
mess_direct_getpermq
Generic Interface to Direct Solvers, 240
mess_direct_getscalecol
Generic Interface to Direct Solvers, 240
mess_direct_getscalerow
Generic Interface to Direct Solvers, 240
mess_direct_getU
Generic Interface to Direct Solvers, 241
mess_direct_init
Generic Interface to Direct Solvers, 241
mess_direct_inverse
Generic Interface to Direct Solvers, 241
mess_direct_levelset
Various Other Functions related to Direct Solvers., 315

mess_direct_levelset_st, 834
levelind, 835
levelptr, 835
levels, 835
mess_direct_lu
Direct Solvers for Linear Systems, 262
mess_direct_lu_select
Direct Solvers for Linear Systems, 263
mess_direct_lupackage_t
Direct Solvers for Linear Systems, 248
mess_direct_lupackage_t_from_mexmess
Interfaces_matlab, 742
mess_direct_lupackage_t_str
Miscellaneous Helper Functions, 677
mess_direct_lupackage_t_to_mexmess
Interfaces_matlab, 742
mess_direct_lyapunov_res2
Residual Computations, 305
mess_direct_res2
Residual Computations, 307
mess_direct_res2m
Residual Computations, 308
mess_direct_solve
Generic Interface to Direct Solvers, 241
mess_direct_solvem
Generic Interface to Direct Solvers, 242
mess_direct_splsolve
Various Other Functions related to Direct Solvers., 315
mess_direct_splsolvec
Various Other Functions related to Direct Solvers., 316
mess_direct_st, 837
clear, 838
cols, 838
data, 838
data_type, 838
det, 838
detc, 838
getL, 838
getpermp, 838
getpermq, 839
getscalecol, 839
getscalerow, 839
getU, 839
inverse, 839
name, 839
rows, 839
solve, 839
solveh, 839
solvem, 839
solvemh, 839
solvemt, 839
solvet, 840
mess_direct_st_offset
Miscellaneous Helper Functions, 677
mess_direct_stein_res2
Residual Computations, 309
mess_direct_sylvester_res2
Residual Computations, 310
mess_direct_sylvestersg_res2
Residual Computations, 312
mess_double_cpx_t
General Type Definitions and Macros, 46
mess_double_from_mexmess
Interfaces_matlab, 742
mess_dynsys
Interface for handling Dynamical Systems, 353
mess_dynsys_2nd
Interface for handling Dynamical Systems, 353
mess_dynsys_2nd_copy
Interface for handling Dynamical Systems, 354
mess_dynsys_2nd_to_1st
Interface for handling Dynamical Systems, 355
mess_dynsys_clear
Interface for handling Dynamical Systems, 356
mess_dynsys_evaltransfer
Interface for handling Dynamical Systems, 356
mess_dynsys_glti
Interface for handling Dynamical Systems, 358
mess_dynsys_glti_copy
Interface for handling Dynamical Systems, 359
mess_dynsys_init
Interface for handling Dynamical Systems, 360
mess_dynsys_isstable
Interface for handling Dynamical Systems, 360
mess_dynsys_lti
Interface for handling Dynamical Systems, 361
mess_dynsys_lti_copy
Interface for handling Dynamical Systems, 362
mess_dynsys_printinfo
Interface for handling Dynamical Systems, 362
mess_dynsys_project_2nd_to_1st
Interface for handling Dynamical Systems, 363
mess_dynsys_project_2nd_to_2nd
Interface for handling Dynamical Systems, 364
mess_dynsys_project_to_glti
Interface for handling Dynamical Systems, 365
mess_dynsys_project_to_lti
Interface for handling Dynamical Systems, 366
mess_dynsys_st, 840
A, 841
B, 841
C, 841
Cp, 841
Cv, 841
dim, 841
E, 841
G, 841
inputs, 842
K, 842
M, 842
outputs, 842
type, 842
mess_dynsys_transfer_diff
Interface for handling Dynamical Systems, 367

mess_dynsysstr
Interface for handling Dynamical Systems, 368
mess_eigen_arnoldi
Sparse Eigenvalue Solvers, 445
mess_eigen_arnoldi_inv
Sparse Eigenvalue Solvers, 446
mess_eigen_arnoldi_template
Sparse Eigenvalue Solvers, 447
mess_eigen_arnoldi_template_nrm
Sparse Eigenvalue Solvers, 449
mess_eigen_arnoldig
Sparse Eigenvalue Solvers, 450
mess_eigen_arnoldig_inv
Sparse Eigenvalue Solvers, 451
mess_eigen_arpack
Sparse Eigenvalue Solvers, 453
mess_eigen_arpack_lanczos
Sparse Eigenvalue Solvers, 455
mess_eigen_arpack_lanczos_template
Sparse Eigenvalue Solvers, 456
mess_eigen_arpack_options_t, 842
b0, 843
maxit, 843
ncv, 843
tol, 843
which, 843
mess_eigen_arpack_template
Sparse Eigenvalue Solvers, 458
mess_eigen_arpack_which_t
Sparse Eigenvalue Solvers, 444
mess_eigen_eig
Dense Eigenvalue Solvers, 426
mess_eigen_eigg
Dense Eigenvalue Solvers, 427
mess_eigen_eiggs
Sparse Eigenvalue Solvers, 459
mess_eigen_eigs
Sparse Eigenvalue Solvers, 460
mess_eigen_gschur
Dense Eigenvalue Solvers, 428
mess_eigen_gschur_complex
Dense Eigenvalue Solvers, 431
mess_eigen_gsign
Dense Eigenvalue Solvers, 432
mess_eigen_hessenberg_abs_largest
Dense Eigenvalue Solvers, 433
mess_eigen_hessenberg_schur
Dense Eigenvalue Solvers, 434
mess_eigen_lanczos_template_nrm
Sparse Eigenvalue Solvers, 462
mess_eigen_schur
Dense Eigenvalue Solvers, 435
mess_eigen_schur_complex
Dense Eigenvalue Solvers, 436
mess_eigen_schur_complex_post
Dense Eigenvalue Solvers, 437
mess_eigen_schur_complex_zgees
Dense Eigenvalue Solvers, 438
mess_eigen_schur_to_evd
Dense Eigenvalue Solvers, 439
mess_eigen_sign
Dense Eigenvalue Solvers, 439
mess_eigen_svd
Dense Eigenvalue Solvers, 440
mess_eigen_svd_econ
Dense Eigenvalue Solvers, 440
mess_enum_converter.c, 1636
MESS_ENUM_CONVERTER, 1637
mess_eps
Miscellaneous Helper Functions, 678
mess_equation
Generic Functions and Data Structures, 467
mess_equation_A_apply
Define own Matrix Generators, 535
mess_equation_A_apply_vector
Define own Matrix Generators, 536
mess_equation_A_post
Define own Matrix Generators, 537
mess_equation_A_pre
Define own Matrix Generators, 537
mess_equation_ApE_apply
Define own Matrix Generators, 538
mess_equation_ApE_apply_vector
Define own Matrix Generators, 538
mess_equation_ApE_post
Define own Matrix Generators, 539
mess_equation_ApE_pre
Define own Matrix Generators, 539
mess_equation_ApEs_apply
Define own Matrix Generators, 540
mess_equation_ApEs_apply_vector
Define own Matrix Generators, 540
mess_equation_ApEs_post
Define own Matrix Generators, 541
mess_equation_ApEs_pre
Define own Matrix Generators, 542
mess_equation_As_apply
Define own Matrix Generators, 542
mess_equation_As_apply_vector
Define own Matrix Generators, 543
mess_equation_As_post
Define own Matrix Generators, 544
mess_equation_As_pre
Define own Matrix Generators, 544
mess_equation_E_apply
Define own Matrix Generators, 545
mess_equation_E_apply_vector
Define own Matrix Generators, 546
mess_equation_E_post
Define own Matrix Generators, 547
mess_equation_E_pre
Define own Matrix Generators, 547
mess_equation_Es_apply
Define own Matrix Generators, 548
mess_equation_Es_apply_vector
Define own Matrix Generators, 548

mess_equation_Es_post
Define own Matrix Generators, 549
mess_equation_Es_pre
Define own Matrix Generators, 550
mess_equation_clean
Generic Functions and Data Structures, 468
mess_equation_clear
Generic Functions and Data Structures, 469
mess_equation_dim
Define own Matrix Generators, 545
mess_equation_from_mexmess
Interfaces_matlab, 742
mess_equation_from_mexmess.c, 1643
eqn_converter, 1644
eqns, 1644
equations_t, 1644
mess_equation_from_python
Interface to Python, 636
mess_equation_glyap
Equation Generators, 504
mess_equation_glyap_dae1
Equation Generators, 505
mess_equation_glyap_dae2
Equation Generators, 507
mess_equation_glyap_so1
Equation Generators, 509
mess_equation_glyap_so2
Equation Generators, 511
mess_equation_griccati
Equation Generators, 513
mess_equation_griccati_dae1
Equation Generators, 515
mess_equation_griccati_dae2
Equation Generators, 517
mess_equation_griccati_so1
Equation Generators, 520
mess_equation_griccati_so2
Equation Generators, 522
mess_equation_has_ApEs
Define own Matrix Generators, 551
mess_equation_has_ApE
Define own Matrix Generators, 550
mess_equation_has_As
Define own Matrix Generators, 551
mess_equation_has_Es
Define own Matrix Generators, 552
mess_equation_has_A
Define own Matrix Generators, 550
mess_equation_has_E
Define own Matrix Generators, 551
mess_equation_init
Generic Functions and Data Structures, 469
mess_equation_lyap
Equation Generators, 524
mess_equation_riccati
Equation Generators, 526
mess_equation_set_rhs
Generic Functions and Data Structures, 470
mess_equation_set_rhs2
Generic Functions and Data Structures, 470
mess_equation_st, 847
AINV, 848
ApEINV, 848
ApEX, 848
aux, 848
AX, 848
B, 848
C, 848
child, 849
clear, 849
clearRHS, 849
clearB, 849
clearC, 849
dim, 849
EINV, 849
eqn_type, 849
EX, 849
init_rhs, 849
K, 849
parameter, 849
parent, 849
RHS, 850
mess_equation_st::AINV_st, 802
apply, 802
clear, 802
generate, 802
to_clear, 802
mess_equation_st::AX_st, 805
apply, 805
clear, 805
generate, 805
to_clear, 806
mess_equation_st::ApEINV_st, 803
apply, 803
clear, 803
generate, 803
to_clear, 803
mess_equation_st::ApEX_st, 804
apply, 804
clear, 804
generate, 804
to_clear, 805
mess_equation_st::EINV_st, 812
apply, 812
clear, 812
generate, 812
to_clear, 813
mess_equation_st::EX_st, 813
apply, 813
clear, 813
generate, 813
to_clear, 814
mess_equation_st_offset
Miscellaneous Helper Functions, 678
mess_equation_stable
Equation Generators, 528

mess_equation_std_lyap
Equation Generators, 530
mess_equation_std_riccati
Equation Generators, 531
mess_equation_t
Generic Functions and Data Structures, 467
mess_equation_t_from_mexmess
Interfaces_matlab, 743
mess_equation_t_str
Miscellaneous Helper Functions, 678
mess_equation_t_to_mexmess
Interfaces_matlab, 743
mess_error_level
Error Codes and Handling, 609
mess_exit
Miscellaneous Helper Functions, 679
mess_float_cpx_t
General Type Definitions and Macros, 46
mess_free
Various Macros, 707
mess_freelist
Miscellaneous Helper Functions, 675
mess_freelist_add_mess_equation
Miscellaneous Helper Functions, 679
mess_freelist_add_mess_matrix
Miscellaneous Helper Functions, 679
mess_freelist_add_mess_options
Miscellaneous Helper Functions, 679
mess_freelist_add_mess_status
Miscellaneous Helper Functions, 679
mess_freelist_add_mess_vector
Miscellaneous Helper Functions, 679
mess_freelist_add_ptr
Miscellaneous Helper Functions, 680
mess_freelist_clear
Miscellaneous Helper Functions, 680
mess_freelist_init
Miscellaneous Helper Functions, 680
mess_freelist_print
Miscellaneous Helper Functions, 680
mess_freelist_st, 852
eqns, 853
matrices, 853
n_eqns, 853
n_matrices, 853
n_opts, 853
n_ptrs, 853
n_stats, 853
n_vectors, 853
opts, 853
ptrs, 853
stats, 853
vectors, 853
mess_get_error
Error Codes and Handling, 608
mess_git_branch
Miscellaneous Helper Functions, 681
mess_git_id
Miscellaneous Helper Functions, 681
mess_glyap
Easy Frontends to Main Functions, 587
mess_glyap3x
glyap3.c, 1444
mess_graph_dfs
Graph Algorithms, 595
mess_graph_lblockpart
Graph Algorithms, 596
mess_graph_reach
Graph Algorithms, 596
mess_gstein
Easy Frontends to Main Functions, 588
mess_h2_error
H2 Model Order Reduction, 407
mess_h2_error_internal
H2 Model Order Reduction, 408
mess_h2_irka_biorth
H2 Model Order Reduction, 409
mess_h2_irka_init
H2 Model Order Reduction, 411
mess_h2_norm
H2 Model Order Reduction, 413
mess_h2_norm_internal
H2 Model Order Reduction, 414
mess_h2_options
H2 Model Order Reduction, 396
mess_h2_options_clear
H2 Model Order Reduction, 416
mess_h2_options_init
H2 Model Order Reduction, 417
mess_h2_options_print
H2 Model Order Reduction, 417
mess_h2_options_st, 854
calc_finalh2, 854
calc_h2err, 854
maxit, 854
output, 855
rdim, 855
stepdebug, 855
stepdebug_data, 855
tol, 855
mess_h2_status
H2 Model Order Reduction, 396
mess_h2_status_clear
H2 Model Order Reduction, 417
mess_h2_status_init
H2 Model Order Reduction, 417
mess_h2_status_print
H2 Model Order Reduction, 418
mess_h2_status_printfull
H2 Model Order Reduction, 418
mess_h2_status_st, 855
cancel_it, 856
cancel_sigma, 856
finalh2, 856
finalsigma, 856
h2err, 856

it, 856
sigmadiff, 856
time, 856
mess_h2_status_write_mfile
H2 Model Order Reduction, 419
mess_h2_tsia
H2 Model Order Reduction, 419
mess_h2_tsiag
H2 Model Order Reduction, 420
mess_have_amd
Miscellaneous Helper Functions, 681
mess_have_arpack
Miscellaneous Helper Functions, 682
mess_have_bzip2
Miscellaneous Helper Functions, 682
mess_have_cholmod
Miscellaneous Helper Functions, 682
mess_have_colamd
Miscellaneous Helper Functions, 682
mess_have_csparse
Miscellaneous Helper Functions, 682
mess_have_matio
Miscellaneous Helper Functions, 682
mess_have_mess64
Miscellaneous Helper Functions, 682
mess_have_mklpardiso
Miscellaneous Helper Functions, 682
mess_have_openmp
Miscellaneous Helper Functions, 682
mess_have_superlu
Miscellaneous Helper Functions, 682
mess_have_umfpack
Miscellaneous Helper Functions, 682
mess_have_zlib
Miscellaneous Helper Functions, 682
mess_init
Miscellaneous Helper Functions, 682
mess_int_array_to_matlab
interface_matlab.c, 1466
mess_int_t
General Type Definitions and Macros, 46
mess_int_t_size
Miscellaneous Helper Functions, 682
mess_is_debug
Miscellaneous Helper Functions, 683
mess_krylov_arnoldi
arnoldi_template.c, 968
mess_krylov_arnoldi_H_largest_eigenvalue
arnoldi_template.c, 971
mess_krylov_arnoldi_clear
arnoldi_template.c, 968
mess_krylov_arnoldi_create
arnoldi_template.c, 969
mess_krylov_arnoldi_getH
arnoldi_template.c, 970
mess_krylov_arnoldi_getV
arnoldi_template.c, 970
mess_krylov_arnoldi_init
arnoldi_template.c, 972
mess_krylov_arnoldi_number_arnoldi_vectors
arnoldi_template.c, 972
mess_krylov_arnoldi_print
arnoldi_template.c, 972
mess_krylov_arnoldi_st, 857
_w, 858
_wtmp, 858
breakdown, 858
breakdown_tol, 858
data_type, 858
dim, 858
H, 858
j, 858
V, 858
mess_krylov_arnoldi_step
arnoldi_template.c, 973
mess_lradi
The ADI and the ADI-NM Algorithm, 484
mess_lrcf_gsignber
Additional Matrix Equation Solvers, 561
mess_lrcf_gsignfac
Additional Matrix Equation Solvers, 563
mess_lrcf_signber
Additional Matrix Equation Solvers, 564
mess_lrcf_signfac
Additional Matrix Equation Solvers, 566
mess_lrcfadi_adi
The ADI and the ADI-NM Algorithm, 486
mess_lrcfadi_ccsvd
Miscellaneous Functions, 570
mess_lrcfadi_colcompress
Miscellaneous Functions, 571
mess_lrcfadi_fac2diff
Miscellaneous Functions, 572
mess_lrcfadi_facFdiff
Miscellaneous Functions, 573
mess_lrcfadi_galerkin
The ADI and the ADI-NM Algorithm, 488
mess_lrcfadi_getritzvalues
The ADI and the ADI-NM Algorithm, 489
mess_lrcfadi_lp_mnmx
Miscellaneous Functions, 574
mess_lrcfadi_lp_s
Miscellaneous Functions, 575
mess_lrcfadi_nm
The ADI and the ADI-NM Algorithm, 490
mess_lrcfadi_nm_linesearch
The ADI and the ADI-NM Algorithm, 492
mess_lrcfadi_nm_linesearch_lambda
The ADI and the ADI-NM Algorithm, 493
mess_lrcfadi_nm_linesearch_poly
The ADI and the ADI-NM Algorithm, 494
mess_lrcfadi_para_wachspress
Miscellaneous Functions, 575
mess_lrcfadi_parameter
The ADI and the ADI-NM Algorithm, 495
mess_lrcfadi_res2

Miscellaneous Functions, 576
mess_lrcfadi_res2BK
Miscellaneous Functions, 577
mess_lrcfadi_res2g
Miscellaneous Functions, 578
mess_lrcfadi_res2g_so
Miscellaneous Functions, 579
mess_lrcfadi_res2gBK
Miscellaneous Functions, 579
mess_lrcfadi_res2gt
Miscellaneous Functions, 580
mess_lrcfadi_res2gt_so
Miscellaneous Functions, 581
mess_lrcfadi_res2nm
Miscellaneous Functions, 582
mess_lrcfadi_res2nmg
Miscellaneous Functions, 582
mess_lrcfadi_res2t
Miscellaneous Functions, 583
mess_lrcfadi_residual
The ADI and the ADI-NM Algorithm, 497
mess_lrcfadi_residual_lyap
res2_template.c, 1831
mess_lrcfadi_residual_nm
res2_template.c, 1832
mess_lrcfadi_step
Generic Functions and Data Structures, 467
mess_lrnm
The ADI and the ADI-NM Algorithm, 499
mess_lyap
Easy Frontends to Main Functions, 590
mess_matgen_fdm_function
FDM Matrix Generators, 600
mess_matgen_fdmcolumn
FDM Matrix Generators, 600
mess_matgen_fdmmatrix
FDM Matrix Generators, 600
mess_matgen_fdmrow
FDM Matrix Generators, 601
mess_matgen_fdmvector
FDM Matrix Generators, 602
mess_matrix
Data Structures, Types and Macros for Handling Matrices, 180
mess_matrix.c, 1646
mess_matrix_add
Arithmetic Operations on Matrices, 52
mess_matrix_addc
Arithmetic Operations on Matrices, 53
mess_matrix_addcols
Allocate, Convert and Access to Matrices, 112
mess_matrix_addcols1
Allocate, Convert and Access to Matrices, 113
mess_matrix_addcols2p
Allocate, Convert and Access to Matrices, 114
mess_matrix_alloc
Allocate, Convert and Access to Matrices, 115
mess_matrix_alloc_setld
Allocate, Convert and Access to Matrices, 115
mess_matrix_any
Arithmetic Operations on Matrices, 54
mess_matrix_applyPI_dae2
equation_glyap_dae2.c, 1379
mess_matrix_backslash
Generic Interface to Direct Solvers, 242
mess_matrix_backslashm
Generic Interface to Direct Solvers, 244
mess_matrix_bandwidth
Allocate, Convert and Access to Matrices, 115
mess_matrix_biorth
Arithmetic Operations on Matrices, 55
mess_matrix_cat
Allocate, Convert and Access to Matrices, 116
mess_matrix_clear
Data Structures, Types and Macros for Handling Matrices, 181
mess_matrix_colaxpy
Arithmetic Operations on Matrix Columns, 80
mess_matrix_colaxpy2
Arithmetic Operations on Matrix Columns, 80
mess_matrix_coldot
Arithmetic Operations on Matrix Columns, 81
mess_matrix_coldotc
Arithmetic Operations on Matrix Columns, 82
mess_matrix_coldotcE
Arithmetic Operations on Matrix Columns, 83
mess_matrix_coldotE
Arithmetic Operations on Matrix Columns, 84
mess_matrix_colnorm
Arithmetic Operations on Matrix Columns, 85
mess_matrix_colnormE
Arithmetic Operations on Matrix Columns, 86
mess_matrix_colperm
Various Other Operations on Matrices, 164
mess_matrix_colpermcopy
Various Other Operations on Matrices, 165
mess_matrix_colscale
Arithmetic Operations on Matrix Columns, 87
mess_matrix_colscalem
Allocate, Convert and Access to Matrices, 117
mess_matrix_colsub
Allocate, Convert and Access to Matrices, 118
mess_matrix_colsums
Arithmetic Operations on Matrices, 56
mess_matrix_colvecaxpy
Arithmetic Operations on Matrix Columns, 88
mess_matrix_colvecdot
Arithmetic Operations on Matrix Columns, 89
mess_matrix_colvecdotc
Arithmetic Operations on Matrix Columns, 90
mess_matrix_complex_from_parts
Allocate, Convert and Access to Matrices, 119
mess_matrix_condest
Computation of Matrix Norms, 92
mess_matrix_conj
Arithmetic Operations on Matrices, 57

mess_matrix_convert
Allocate, Convert and Access to Matrices, 120
mess_matrix_convert_cholmod_to_csc
Interface to Cholmod, 619
mess_matrix_convert_cholmod_to_dense
Interface to Cholmod, 619
mess_matrix_convert_csc_csr
Allocate, Convert and Access to Matrices, 121
mess_matrix_convert_csc_to_cholmod
Interface to Cholmod, 619
mess_matrix_convert_csr_csc
Allocate, Convert and Access to Matrices, 122
mess_matrix_convert_dense_to_cholmod
Interface to Cholmod, 619
mess_matrix_coord
Allocate, Convert and Access to Matrices, 123
mess_matrix_copy
Allocate, Convert and Access to Matrices, 124
mess_matrix_csc
Allocate, Convert and Access to Matrices, 124
mess_matrix_csr
Allocate, Convert and Access to Matrices, 125
mess_matrix_ctranspose
Arithmetic Operations on Matrices, 58
mess_matrix_decomp
Various Other Operations on Matrices, 166
mess_matrix_dense_from_carray
Allocate, Convert and Access to Matrices, 126
mess_matrix_dense_from_farray
Allocate, Convert and Access to Matrices, 127
mess_matrix_diag
Allocate, Convert and Access to Matrices, 127
mess_matrix_diag_from_vector
Allocate, Convert and Access to Matrices, 128
mess_matrix_diagpos
Allocate, Convert and Access to Matrices, 129
mess_matrix_diffnorm
Computation of Matrix Norms, 93
mess_matrix_diffnormf
Computation of Matrix Norms, 94
mess_matrix_display
Input/Output Operations with Matrices, 155
mess_matrix_dupl
Allocate, Convert and Access to Matrices, 129
mess_matrix_dynorm2
Computation of Matrix Norms, 95
mess_matrix_dynormf
Computation of Matrix Norms, 95
mess_matrix_dynormf2
Computation of Matrix Norms, 96
mess_matrix_eliminate_zeros
Allocate, Convert and Access to Matrices, 130
mess_matrix_equal
Various Other Operations on Matrices, 166
mess_matrix_equal_verbose
Various Other Operations on Matrices, 167
mess_matrix_equalpattern
Various Other Operations on Matrices, 167
mess_matrix_eye
Allocate, Convert and Access to Matrices, 130
mess_matrix_eyec
Allocate, Convert and Access to Matrices, 131
mess_matrix_file
Data Structures, Types and Macros for Handling Matrices, 181
mess_matrix_file_st, 869
cols, 870
data_type, 870
ld, 870
magic, 870
nnz, 870
rows, 870
sizemess_int_t, 870
store_type, 870
symmetry, 871
version, 871
mess_matrix_fro_inn
Arithmetic Operations on Matrices, 58
mess_matrix_from_csparse
Interface to CSparse, 613
mess_matrix_from_csparse_ci
Interface to CSparse, 614
mess_matrix_from_csparse_cl
Interface to CSparse, 614
mess_matrix_from_csparse_di
Interface to CSparse, 615
mess_matrix_from_csparse_dl
Interface to CSparse, 615
mess_matrix_from_mexmess
Interfaces_matlab, 743
mess_matrix_gaxpy
Arithmetic Operations on Matrices, 59
mess_matrix_gbiorth
Arithmetic Operations on Matrices, 60
mess_matrix_getcol
Allocate, Convert and Access to Matrices, 131
mess_matrix_getelement
Allocate, Convert and Access to Matrices, 132
mess_matrix_getrow
Allocate, Convert and Access to Matrices, 133
mess_matrix_imagpart
Allocate, Convert and Access to Matrices, 134
mess_matrix_indefinite_dynorm2
Computation of Matrix Norms, 97
mess_matrix_indefinite_dynormf
Computation of Matrix Norms, 98
mess_matrix_indefinite_dynormf2
Computation of Matrix Norms, 99
mess_matrix_init
Data Structures, Types and Macros for Handling Matrices, 182
mess_matrix_isstable
Interface for handling Dynamical Systems, 369
mess_matrix_isstableg
Interface for handling Dynamical Systems, 369
mess_matrix_joinpatterns

Various Other Operations on Matrices, 168
mess_matrix_lift
Allocate, Convert and Access to Matrices, 134
mess_matrix_map
Arithmetic Operations on Matrices, 61
mess_matrix_map_abs
Arithmetic Operations on Matrices, 62
mess_matrix_map_acos
Arithmetic Operations on Matrices, 63
mess_matrix_map_acosh
Arithmetic Operations on Matrices, 63
mess_matrix_map_arg
Arithmetic Operations on Matrices, 63
mess_matrix_map_asin
Arithmetic Operations on Matrices, 63
mess_matrix_map_asinh
Arithmetic Operations on Matrices, 63
mess_matrix_map_atan
Arithmetic Operations on Matrices, 63
mess_matrix_map_atanh
Arithmetic Operations on Matrices, 63
mess_matrix_map_ceil
Arithmetic Operations on Matrices, 63
mess_matrix_map_conj
Arithmetic Operations on Matrices, 63
mess_matrix_map_cos
Arithmetic Operations on Matrices, 63
mess_matrix_map_cosh
Arithmetic Operations on Matrices, 63
mess_matrix_map_exp
Arithmetic Operations on Matrices, 63
mess_matrix_map_expm1
Arithmetic Operations on Matrices, 63
mess_matrix_map_floor
Arithmetic Operations on Matrices, 63
mess_matrix_map_isfinite
Arithmetic Operations on Matrices, 63
mess_matrix_map_isinf
Arithmetic Operations on Matrices, 63
mess_matrix_map_isnan
Arithmetic Operations on Matrices, 63
mess_matrix_map_log
Arithmetic Operations on Matrices, 63
mess_matrix_map_not
Arithmetic Operations on Matrices, 63
mess_matrix_map_round
Arithmetic Operations on Matrices, 63
mess_matrix_map_sin
Arithmetic Operations on Matrices, 64
mess_matrix_map_sinh
Arithmetic Operations on Matrices, 64
mess_matrix_map_sqrt
Arithmetic Operations on Matrices, 64
mess_matrix_map_tan
Arithmetic Operations on Matrices, 64
mess_matrix_map_tanh
Arithmetic Operations on Matrices, 64
mess_matrix_memsize
Various Other Operations on Matrices, 169
mess_matrix_memsize_nnz
Various Other Operations on Matrices, 169
mess_matrix_mgs
Arithmetic Operations on Matrices, 64
mess_matrix_mgs_add
Arithmetic Operations on Matrices, 65
mess_matrix_mgs_inplace
Arithmetic Operations on Matrices, 66
mess_matrix_mulnormf2
Computation of Matrix Norms, 100
mess_matrix_multiply
Arithmetic Operations on Matrices, 67
mess_matrix_mvp
Arithmetic Operations on Matrices, 67
mess_matrix_need_alloc
Allocate, Convert and Access to Matrices, 135
mess_matrix_norm
Computation of Matrix Norms, 101
mess_matrix_norm1
Computation of Matrix Norms, 102
mess_matrix_norm2
Computation of Matrix Norms, 103
mess_matrix_norm2inv
Computation of Matrix Norms, 104
mess_matrix_normf
Computation of Matrix Norms, 105
mess_matrix_normf2
Computation of Matrix Norms, 106
mess_matrix_norminf
Computation of Matrix Norms, 107
mess_matrix_norminvest
Computation of Matrix Norms, 108
mess_matrix_null
Arithmetic Operations on Matrices, 68
mess_matrix_one_value
Allocate, Convert and Access to Matrices, 136
mess_matrix_one_valuec
Allocate, Convert and Access to Matrices, 136
mess_matrix_ones
Allocate, Convert and Access to Matrices, 137
mess_matrix_orth
Arithmetic Operations on Matrices, 69
mess_matrix_perm
Various Other Operations on Matrices, 170
mess_matrix_permcopy
Various Other Operations on Matrices, 170
mess_matrix_pinv
Generic Interface to Direct Solvers, 245
mess_matrix_print
Input/Output Operations with Matrices, 156
mess_matrix_printdata
Input/Output Operations with Matrices, 157
mess_matrix_printinfo
Input/Output Operations with Matrices, 157
mess_matrix_printshort
Input/Output Operations with Matrices, 158
mess_matrix_proj_gmgs

Arithmetic Operations on Matrices, 70
mess_matrix_proj_mgs
Arithmetic Operations on Matrices, 71
mess_matrix_proj_sym
Arithmetic Operations on Matrices, 72
mess_matrix_rand
Allocate, Convert and Access to Matrices, 137
mess_matrix_rand_coord
Allocate, Convert and Access to Matrices, 138
mess_matrix_rand_csc
Allocate, Convert and Access to Matrices, 139
mess_matrix_rand_csr
Allocate, Convert and Access to Matrices, 140
mess_matrix_rand_dense
Allocate, Convert and Access to Matrices, 141
mess_matrix_rand_dense_uniform
Allocate, Convert and Access to Matrices, 142
mess_matrix_rand_init
Allocate, Convert and Access to Matrices, 142
mess_matrix_rank
Arithmetic Operations on Matrices, 73
mess_matrix_read
Input/Output Operations with Matrices, 159
mess_matrix_read_formated
Input/Output Operations with Matrices, 160
mess_matrix_realloc
Allocate, Convert and Access to Matrices, 143
mess_matrix_realpart
Allocate, Convert and Access to Matrices, 143
mess_matrix_reorder
Various Other Operations on Matrices, 171
mess_matrix_reorder_amd
Various Other Operations on Matrices, 172
mess_matrix_reorder_colamd
Various Other Operations on Matrices, 173
mess_matrix_reorder_rcm
Various Other Operations on Matrices, 173
mess_matrix_reset
Allocate, Convert and Access to Matrices, 144
mess_matrix_resize
Allocate, Convert and Access to Matrices, 144
mess_matrix_rowscalem
Allocate, Convert and Access to Matrices, 145
mess_matrix_rowsub
Allocate, Convert and Access to Matrices, 145
mess_matrix_rowsums
Arithmetic Operations on Matrices, 73
mess_matrix_scale
Arithmetic Operations on Matrices, 74
mess_matrix_scalec
Arithmetic Operations on Matrices, 75
mess_matrix_setcol
Allocate, Convert and Access to Matrices, 146
mess_matrix_setelement
Allocate, Convert and Access to Matrices, 147
mess_matrix_setelement_complex
Allocate, Convert and Access to Matrices, 147
mess_matrix_setrow
Allocate, Convert and Access to Matrices, 148
mess_matrix_sort
Allocate, Convert and Access to Matrices, 149
mess_matrix_spy
Various Other Operations on Matrices, 174
mess_matrix_st, 871
colptr, 872
cols, 872
Data Structures, Types and Macros for Handling Matrices, 181
data_type, 872
ld, 872
nnz, 872
rowptr, 872
rows, 872
store_type, 872
symmetry, 873
values, 873
values_cpx, 873
mess_matrix_st_offset
Miscellaneous Helper Functions, 683
mess_matrix_sub
Allocate, Convert and Access to Matrices, 150
mess_matrix_symfillup
Allocate, Convert and Access to Matrices, 150
mess_matrix_to_csparse
Interface to CSparse, 616
mess_matrix_to_csparse_ci
Interface to CSparse, 616
mess_matrix_to_csparse_cl
Interface to CSparse, 617
mess_matrix_to_csparse_di
Interface to CSparse, 617
mess_matrix_to_csparse_dl
Interface to CSparse, 618
mess_matrix_to_mexmess
Interfaces_matlab, 745
mess_matrix_tocomplex
Allocate, Convert and Access to Matrices, 151
mess_matrix_toreal
Allocate, Convert and Access to Matrices, 151
mess_matrix_totype
Allocate, Convert and Access to Matrices, 151
mess_matrix_trace
Arithmetic Operations on Matrices, 75
mess_matrix_tracec
Arithmetic Operations on Matrices, 76
mess_matrix_transpose
Arithmetic Operations on Matrices, 77
mess_matrix_tridiag
Allocate, Convert and Access to Matrices, 152
mess_matrix_tril
Allocate, Convert and Access to Matrices, 153
mess_matrix_triu
Allocate, Convert and Access to Matrices, 153
mess_matrix_view
Allocate, Convert and Access to Matrices, 154
mess_matrix_write

Input/Output Operations with Matrices, 161
mess_matrix_xtranspose
Arithmetic Operations on Matrices, 77
mess_matrix_zeros
Allocate, Convert and Access to Matrices, 154
mess_memusage_t
Generic Functions and Data Structures, 467
mess_memusage_t_from_mexmess
Interfaces_matlab, 746
mess_memusage_t_str
Miscellaneous Helper Functions, 683
mess_memusage_t_to_mexmess
Interfaces_matlab, 746
mess_multidirect
Generic Interface to Multisolvers, 319
mess_multidirect_clear
Generic Interface to Multisolvers, 319
mess_multidirect_create
Multisolvers, 322
mess_multidirect_create_sparse_lu
Multisolvers, 324
mess_multidirect_create_umfpack
Multisolvers, 326
mess_multidirect_init
Generic Interface to Multisolvers, 319
mess_multidirect_select
Multisolvers, 327
mess_multidirect_solve
Generic Interface to Multisolvers, 320
mess_multidirect_solvem
Generic Interface to Multisolvers, 320
mess_multidirect_st, 879
clear, 880
cols, 880
data, 880
data_type, 880
getdatatype, 881
getL, 881
getp, 881
getq, 881
getscalecol, 881
getscalerow, 881
getU, 881
indx, 881
memsize, 881
name, 881
rows, 881
solve, 881
solveh, 882
solvem, 882
solvemh, 882
solvemt, 882
solvet, 882
mess_multidirect_t
Generic Interface to Multisolvers, 319
mess_multidirect_t_from_mexmess
Interfaces_matlab, 746
mess_multidirect_t_str
Miscellaneous Helper Functions, 683
mess_multidirect_t_to_mexmess
Interfaces_matlab, 746
mess_mvpcall
Generic matrix-vector Products, 184
mess_mvpcall_apply
Generic matrix-vector Products, 184
mess_mvpcall_clear
Generic matrix-vector Products, 184
mess_mvpcall_matrix
Generic matrix-vector Products, 184
mess_mvpcall_operator
Generic matrix-vector Products, 185
mess_mvpcall_st, 884
data, 885
data_type, 885
dim, 885
mvp, 885
mess_norm_t
General Type Definitions and Macros, 46
mess_norm_t_from_mexmess
Interfaces_matlab, 746
mess_norm_t_str
Miscellaneous Helper Functions, 683
mess_norm_t_to_mexmess
Interfaces_matlab, 746
mess_openmp_info
Miscellaneous Helper Functions, 684
mess_operation_t
General Type Definitions and Macros, 47
mess_operation_t_from_mexmess
Interfaces_matlab, 746
mess_operation_t_str
Miscellaneous Helper Functions, 684
mess_operation_t_to_mexmess
Interfaces_matlab, 746
mess_options
Generic Functions and Data Structures, 467
mess_options.c, 1684
mess_options_clear
Generic Functions and Data Structures, 471
mess_options_copy
Generic Functions and Data Structures, 471
mess_options_from_mexmess
Interfaces_matlab, 746
mess_options_from_python
Interface to Python, 637
mess_options_init
Generic Functions and Data Structures, 472
mess_options_memsize
Generic Functions and Data Structures, 473
mess_options_print
Generic Functions and Data Structures, 473
mess_options_set_K0
Generic Functions and Data Structures, 474
mess_options_set_proj_space
Generic Functions and Data Structures, 475
mess_options_set_W

Generic Functions and Data Structures, 476
mess_options_st, 892
adi_maxit, 894
adi_output, 894
adi_rel_change_tol, 894
adi_res2_tol, 894
adi_res2c_tol, 895
adi_shifts_arp_m, 895
adi_shifts_arp_p, 895
adi_shifts_b0, 895
adi_shifts_l0, 895
adi_shifts_p, 895
adi_shifts_paratype, 895
Generic Functions and Data Structures, 467
K0, 895
memory_usage, 895
nm_gpStep, 895
nm_linesearch, 895
nm_maxit, 895
nm_output, 896
nm_res2_tol, 896
nm_singleshifts, 896
nm_stepfunction, 896
nm_stepfunction_aux, 896
proj_space, 896
residual_method, 896
stepfunction, 896
stepfunction_aux, 896
type, 896
W, 896
mess_options_st_offset
Miscellaneous Helper Functions, 684
mess_options_to_mexmess
Interfaces_matlab, 747
mess_options_unset_K0
Generic Functions and Data Structures, 476
mess_options_unset_proj_space
Generic Functions and Data Structures, 477
mess_options_unset_W
Generic Functions and Data Structures, 477
mess_parameter
The ADI and the ADI-NM Algorithm, 500
mess_parameter_t
Generic Functions and Data Structures, 468
mess_parameter_t_from_mexmess
Interfaces_matlab, 747
mess_parameter_t_str
Miscellaneous Helper Functions, 685
mess_parameter_t_to_mexmess
Interfaces_matlab, 748
mess_plot
X11 Function Plotter, 658
mess_plot_addData
X11 Function Plotter, 658
mess_plot_axis_scale_t
Plot Export, 665
mess_plot_clearData
X11 Function Plotter, 659
mess_plot_close
X11 Function Plotter, 659
mess_plot_create
X11 Function Plotter, 660
mess_plot_save
X11 Function Plotter, 661
mess_plot_setColor
X11 Function Plotter, 661
mess_plot_setLabel
X11 Function Plotter, 661
mess_plot_update
X11 Function Plotter, 662
mess_plotExport
Plot Export, 665
mess_plotExport_addData
Plot Export, 665
mess_plotExport_clear
Plot Export, 665
mess_plotExport_clearData
Plot Export, 666
mess_plotExport_createGnuScript
Plot Export, 666
mess_plotExport_createGnuScript_filename
Plot Export, 666
mess_plotExport_createGnuScript_int
plot_scriptExporter.c, 1800
mess_plotExport_createTikzScript
Plot Export, 667
mess_plotExport_createTikzScript_filename
Plot Export, 667
mess_plotExport_createTikzScript_int
plot_scriptExporter.c, 1800
mess_plotExport_init
Plot Export, 668
mess_plotExport_save
Plot Export, 668
mess_plotExport_setColor
Plot Export, 669
mess_plotExport_setLabel
Plot Export, 669
mess_plotExport_setLegendPos
Plot Export, 670
mess_plotExport_setType
Plot Export, 670
mess_plotExportData
Plot Export, 665
mess_plotdata
X11 Function Plotter, 658
mess_precond
Types and Datastructures, 331
mess_precond_clear
Preconditioners, 344
mess_precond_diag
Preconditioners, 345
mess_precond_ilu0
Preconditioners, 345
mess_precond_iluk
Preconditioners, 346

mess_precond_init
Preconditioners, 347
mess_precond_sor
Preconditioners, 347
mess_precond_ssor
Preconditioners, 348
mess_precond_st, 899
clear, 899
data, 899
solve, 899
type, 899
mess_print_bytes
Miscellaneous Helper Functions, 685
mess_print_format_double
Miscellaneous Helper Functions, 685
mess_print_format_double_cpx
Miscellaneous Helper Functions, 685
mess_print_format_select
Miscellaneous Helper Functions, 685
mess_print_format_t
General Type Definitions and Macros, 47
mess_print_format_t_str
gettypes.c, 1433
mess_reorder_t
Various Other Operations on Matrices, 164
mess_residual_t
Generic Functions and Data Structures, 468
mess_residual_t_from_mexmess
Interfaces_matlab, 748
mess_residual_t_str
Miscellaneous Helper Functions, 686
mess_residual_t_to_mexmess
Interfaces_matlab, 748
mess_set_errorlevel
Error Codes and Handling, 609
mess_set_errorlevel.c, 1686
mess_set_malloc
Miscellaneous Helper Functions, 686
mess_sign_scale_t
Additional Matrix Equation Solvers, 554
mess_sign_scale_t_str
Miscellaneous Helper Functions, 686
mess_solver_bicgstab
Solvers, 334
mess_solver_cg
Solvers, 335
mess_solver_gaussseidel
Solvers, 336
mess_solver_gmres
Solvers, 337
mess_solver_gmres_restart
Solvers, 338
mess_solver_jacobi
Solvers, 339
mess_solver_jacobi_convergence
Solvers, 340
mess_solver_lcsolve_kernelcsr_complex
trisolve.c, 1889
mess_solver_lhsolve
Triangular Solvers, 266
mess_solver_lhsolve_kernelcsr_complex
Triangular Solvers, 267
mess_solver_lhsolvem
Triangular Solvers, 268
mess_solver_lsolve
Triangular Solvers, 269
mess_solver_lsolve_kernelcsr_complex
Triangular Solvers, 269
mess_solver_lsolve_kernelcsr_real
Triangular Solvers, 270
mess_solver_lsolve_kernelcsr_real_complex
Triangular Solvers, 270
mess_solver_lsolvem
Triangular Solvers, 271
mess_solver_ltsolve
Triangular Solvers, 272
mess_solver_ltsolve_kernelcsr_complex
Triangular Solvers, 273
mess_solver_ltsolve_kernelcsr_real
Triangular Solvers, 274
mess_solver_ltsolve_kernelcsr_real_complex
Triangular Solvers, 274
mess_solver_ltsolvem
Triangular Solvers, 275
mess_solver_lusolvem_kernelcsr_complex
Triangular Solvers, 276
mess_solver_options
Types and Datastructures, 331
mess_solver_options_clear
Types and Datastructures, 331
mess_solver_options_init
Types and Datastructures, 332
mess_solver_options_print
Types and Datastructures, 332
mess_solver_sor
Solvers, 341
mess_solver_ssor
Solvers, 342
mess_solver_status
Types and Datastructures, 331
mess_solver_status_clean
Types and Datastructures, 332
mess_solver_status_clear
Types and Datastructures, 332
mess_solver_status_init
Types and Datastructures, 333
mess_solver_status_print
Types and Datastructures, 333
mess_solver_ucsolve_kernelcsr_complex
trisolve.c, 1889
mess_solver_uhsolve
Triangular Solvers, 277
mess_solver_uhsolve_kernelcsr_complex
Triangular Solvers, 278
mess_solver_uhsolvem
Triangular Solvers, 279

mess_solver_usolve
Triangular Solvers, 279
mess_solver_usolve_kernelcsr_complex
Triangular Solvers, 280
mess_solver_usolve_kernelcsr_real
Triangular Solvers, 281
mess_solver_usolve_kernelcsr_real_complex
Triangular Solvers, 281
mess_solver_usolvem
Triangular Solvers, 282
mess_solver_utsolve
Triangular Solvers, 283
mess_solver_utsolve_kernelcsr_complex
Triangular Solvers, 284
mess_solver_utsolve_kernelcsr_real
Triangular Solvers, 285
mess_solver_utsolve_kernelcsr_real_complex
Triangular Solvers, 285
mess_solver_utsolvem
Triangular Solvers, 286
mess_status
Generic Functions and Data Structures, 467
mess_status.c, 1687
set_status_properties, 1687
mess_status_clear
Generic Functions and Data Structures, 478
mess_status_copy
Generic Functions and Data Structures, 478
mess_status_init
Generic Functions and Data Structures, 479
mess_status_memsize
Generic Functions and Data Structures, 480
mess_status_print
Generic Functions and Data Structures, 480
mess_status_printfull
Generic Functions and Data Structures, 481
mess_status_st, 905
Generic Functions and Data Structures, 467
internal_status, 907
it, 907
n_internal_status, 907
rel_change, 907
rel_changes, 907
res2_0, 907
res2_change, 907
res2_norm, 907
res2_norms, 907
stop_rel, 907
stop_res2, 907
stop_res2c, 907
stop_user, 907
time_adi, 908
time_all, 908
unstable, 908
mess_status_st_offset
Miscellaneous Helper Functions, 687
mess_status_to_mexmess
Interfaces_matlab, 748
mess_status_to_python
Interface to Python, 638
mess_status_write
Generic Functions and Data Structures, 482
mess_storage_t
Data Structures, Types and Macros for Handling Matrices, 181
mess_storage_t_str
Miscellaneous Helper Functions, 687
mess_symmetry_t
Data Structures, Types and Macros for Handling Matrices, 181
mess_symmetry_t_str
Miscellaneous Helper Functions, 687
mess_tglyap
Easy Frontends to Main Functions, 591
mess_tglyap3x
glyap3.c, 1445
mess_tgstein
Easy Frontends to Main Functions, 592
mess_threadpool
A Compact Loader Pool Implementation, 692
mess_threadpool_clear
A Compact Loader Pool Implementation, 692
mess_threadpool_init
A Compact Loader Pool Implementation, 692
mess_threadpool_insert
A Compact Loader Pool Implementation, 693
mess_threadpool_isdone
A Compact Loader Pool Implementation, 693
mess_threadpool_st, 910
cur_size, 911
hashtable_done, 911
head, 911
job_done, 911
lock, 911
max_size, 911
nextid, 911
not_empty, 911
not_full, 912
num_threads, 912
tail, 912
threads, 912
mess_threadpool_waitdone
A Compact Loader Pool Implementation, 694
mess_threadpool_worker
A Compact Loader Pool Implementation, 694
mess_threadpooljob
A Compact Loader Pool Implementation, 692
mess_threadpooljob_clear
A Compact Loader Pool Implementation, 694
mess_threadpooljob_init
A Compact Loader Pool Implementation, 694
mess_threadpooljob_st, 912
data, 913
exit, 913
id, 913
next, 913

worker, 913
mess_timer
Time Measurement, 697
mess_timer_clear
Time Measurement, 697
mess_timer_diff
Time Measurement, 697
mess_timer_get
Time Measurement, 697
mess_timer_getandreset
Time Measurement, 698
mess_timer_init
Time Measurement, 698
mess_timer_st, 913
dtimer, 914
timer, 914
mess_try_alloc
Various Macros, 709
mess_try_alloc2
Various Macros, 710
mess_try_alloc_omp
Various Macros, 710
mess_try_calloc
Various Macros, 710
mess_try_calloc2
Various Macros, 710
mess_try_realloc
Various Macros, 711
mess_try_realloc2
Various Macros, 711
mess_vector
Data Structures and Related Functions, 189
mess_vector.c, 1689
mess_vector_alloc
Data Structures and Related Functions, 189
mess_vector_any
Various Other Operations with Vectors, 208
mess_vector_axpy
Arithmetic Operations with Vectors, 198
mess_vector_axpyc
Arithmetic Operations with Vectors, 199
mess_vector_cat
Various Other Operations with Vectors, 209
mess_vector_clear
Data Structures and Related Functions, 190
mess_vector_complex_from_parts
Data Structures and Related Functions, 190
mess_vector_conj
Arithmetic Operations with Vectors, 199
mess_vector_convert_cholmod_to_dense
Interface to Cholmod, 619
mess_vector_convert_dense_to_cholmod
Interface to Cholmod, 619
mess_vector_convert_if_real
Various Other Operations with Vectors, 210
mess_vector_copy
Data Structures and Related Functions, 190
mess_vector_copy_tocomplex
Data Structures and Related Functions, 191
mess_vector_diffnorm
Arithmetic Operations with Vectors, 200
mess_vector_diffnorminf
Arithmetic Operations with Vectors, 200
mess_vector_dot
Arithmetic Operations with Vectors, 200
mess_vector_dotc
Arithmetic Operations with Vectors, 201
mess_vector_dotu
Arithmetic Operations with Vectors, 201
mess_vector_filter
Various Other Operations with Vectors, 210
mess_vector_filter_stable
Various Other Operations with Vectors, 211
mess_vector_from_farray
Data Structures and Related Functions, 192
mess_vector_from_lapack
Data Structures and Related Functions, 192
mess_vector_from_mexmess
Interfaces_matlab, 748
mess_vector_frommatrix
Various Other Operations on Matrices, 174
mess_vector_get
Various Other Operations with Vectors, 211
mess_vector_imagpart
Data Structures and Related Functions, 193
mess_vector_init
Data Structures and Related Functions, 193
mess_vector_iperm
Various Other Operations with Vectors, 212
mess_vector_iperm_combine
Various Other Operations with Vectors, 212
mess_vector_iperm_inplace
Various Other Operations with Vectors, 213
mess_vector_iperm_split
Various Other Operations with Vectors, 214
mess_vector_isreal
real_complex.c, 1819
mess_vector_kron
Various Other Operations with Vectors, 214
mess_vector_lift
Various Other Operations with Vectors, 215
mess_vector_linspace
Various Other Operations with Vectors, 216
mess_vector_logspace10
Various Other Operations with Vectors, 217
mess_vector_logspace2
Various Other Operations with Vectors, 217
mess_vector_logspacee
Various Other Operations with Vectors, 218
mess_vector_map
Various Other Operations with Vectors, 219
mess_vector_map_abs
Various Other Operations with Vectors, 220
mess_vector_map_acos
Various Other Operations with Vectors, 220
mess_vector_map_acosh

Various Other Operations with Vectors, 220
mess_vector_map_arg
Various Other Operations with Vectors, 220
mess_vector_map_asin
Various Other Operations with Vectors, 220
mess_vector_map_asinh
Various Other Operations with Vectors, 220
mess_vector_map_atan
Various Other Operations with Vectors, 220
mess_vector_map_atanh
Various Other Operations with Vectors, 220
mess_vector_map_ceil
Various Other Operations with Vectors, 220
mess_vector_map_conj
Various Other Operations with Vectors, 220
mess_vector_map_cos
Various Other Operations with Vectors, 220
mess_vector_map_cosh
Various Other Operations with Vectors, 220
mess_vector_map_exp
Various Other Operations with Vectors, 220
mess_vector_map_expm1
Various Other Operations with Vectors, 220
mess_vector_map_floor
Various Other Operations with Vectors, 220
mess_vector_map_isfinite
Various Other Operations with Vectors, 220
mess_vector_map_isinf
Various Other Operations with Vectors, 220
mess_vector_map_isnan
Various Other Operations with Vectors, 220
mess_vector_map_log
Various Other Operations with Vectors, 220
mess_vector_map_not
Various Other Operations with Vectors, 221
mess_vector_map_round
Various Other Operations with Vectors, 221
mess_vector_map_sin
Various Other Operations with Vectors, 221
mess_vector_map_sinh
Various Other Operations with Vectors, 221
mess_vector_map_sqrt
Various Other Operations with Vectors, 221
mess_vector_map_tan
Various Other Operations with Vectors, 221
mess_vector_map_tanh
Various Other Operations with Vectors, 221
mess_vector_max
Arithmetic Operations with Vectors, 201
mess_vector_maxvalue
Arithmetic Operations with Vectors, 202
mess_vector_memsize
Various Other Operations with Vectors, 221
mess_vector_minvalue
Arithmetic Operations with Vectors, 202
mess_vector_mwrite
vector_read.c, 1973
mess_vector_norm
Arithmetic Operations with Vectors, 202
mess_vector_norm1
Arithmetic Operations with Vectors, 203
mess_vector_norm2
Arithmetic Operations with Vectors, 203
mess_vector_norminf
Arithmetic Operations with Vectors, 204
mess_vector_ones
Various Other Operations with Vectors, 221
mess_vector_perm
Various Other Operations with Vectors, 222
mess_vector_perm_combine
Various Other Operations with Vectors, 222
mess_vector_perm_inplace
Various Other Operations with Vectors, 223
mess_vector_perm_split
Various Other Operations with Vectors, 224
mess_vector_print
Various Other Operations with Vectors, 225
mess_vector_printinfo
Various Other Operations with Vectors, 225
mess_vector_printshort
Various Other Operations with Vectors, 226
mess_vector_rand
Various Other Operations with Vectors, 226
mess_vector_rand_init
Various Other Operations with Vectors, 227
mess_vector_read
Various Other Operations with Vectors, 227
mess_vector_realpart
Data Structures and Related Functions, 194
mess_vector_reset
Data Structures and Related Functions, 194
mess_vector_resize
Various Other Operations with Vectors, 228
mess_vector_scale
Arithmetic Operations with Vectors, 204
mess_vector_scalec
Arithmetic Operations with Vectors, 204
mess_vector_scalee
Arithmetic Operations with Vectors, 198
mess_vector_sort
Various Other Operations with Vectors, 229
mess_vector_sort_imagpart
Various Other Operations with Vectors, 229
mess_vector_sort_realpart
Various Other Operations with Vectors, 230
mess_vector_split
Various Other Operations with Vectors, 230
mess_vector_st, 921
Data Structures and Related Functions, 189
data_type, 922
dim, 922
values, 922
values_cpx, 922
mess_vector_st_offset
Miscellaneous Helper Functions, 688
mess_vector_to_mexmess

Interfaces_matlab, 748
mess_vector_tocomplex
Data Structures and Related Functions, 195
mess_vector_tomatrix
Various Other Operations on Matrices, 175
mess_vector_toreal
Data Structures and Related Functions, 195
mess_vector_toreal_nowarn
Data Structures and Related Functions, 195
mess_vector_totype
Data Structures and Related Functions, 196
mess_vector_valuecomplex
Various Other Operations with Vectors, 231
mess_vector_valuereal
Various Other Operations with Vectors, 231
mess_vector_write
Various Other Operations with Vectors, 232
mess_vector_zeros
Various Other Operations with Vectors, 233
mess_version
Miscellaneous Helper Functions, 688
mess_version.c, 1717
PYMESS_MESS_HAVE_GENERATOR, 1718
mess_version_major
Miscellaneous Helper Functions, 688
mess_version_minor
Miscellaneous Helper Functions, 688
mess_version_patch
Miscellaneous Helper Functions, 688
mess_version_verbose
Miscellaneous Helper Functions, 688
mess_versionx
lib/misc/misc.c, 1723
mess_wtime
Time Measurement, 699
message_level
mklpardiso_solver, 924
mex_mess_care
Interfaces_matlab, 749
mex_mess_care.c, 1718
mex_mess_dense_nm_gmpare
Interfaces_matlab, 749
mex_mess_dense_nm_gmpare.c, 1718
mex_mess_direct_chol_select
Interfaces_matlab, 750
mex_mess_direct_lu_select
Interfaces_matlab, 750
mex_mess_git_branch
Interfaces_matlab, 750
mex_mess_git_id
Interfaces_matlab, 750
mex_mess_glyap
Interfaces_matlab, 750
mex_mess_glyap.c, 1719
mex_mess_gstein
Interfaces_matlab, 751
mex_mess_gstein.c, 1719
mex_mess_have_amd
Interfaces_matlab, 752
mex_mess_have_arpack
Interfaces_matlab, 752
mex_mess_have_bzip2
Interfaces_matlab, 752
mex_mess_have_cholmod
Interfaces_matlab, 752
mex_mess_have_colamd
Interfaces_matlab, 752
mex_mess_have_csparse
Interfaces_matlab, 752
mex_mess_have_matio
Interfaces_matlab, 752
mex_mess_have_mess64
Interfaces_matlab, 752
mex_mess_have_mklpardiso
Interfaces_matlab, 752
mex_mess_have_openmp
Interfaces_matlab, 752
mex_mess_have_superlu
Interfaces_matlab, 752
mex_mess_have_umfpack
Interfaces_matlab, 753
mex_mess_have_zlib
Interfaces_matlab, 753
mex_mess_is_debug
Interfaces_matlab, 753
mex_mess_lradi
Interfaces_matlab, 753
mex_mess_lradi.c, 1719
mex_mess_lrnm
Interfaces_matlab, 753
mex_mess_lrnm.c, 1719
mex_mess_lyap
Interfaces_matlab, 754
mex_mess_lyap.c, 1720
mex_mess_multidirect_select
Interfaces_matlab, 755
mex_mess_sylvester_dense
Interfaces_matlab, 755
mex_mess_sylvester_dense.c, 1720
mex_mess_sylvester_sparsedense
Interfaces_matlab, 756
mex_mess_sylvester_sparsedense.c, 1720
mex_mess_version
Interfaces_matlab, 757
mex_mess_version_major
Interfaces_matlab, 758
mex_mess_version_minor
Interfaces_matlab, 758
mex_mess_version_patch
Interfaces_matlab, 758
mex_mess_version_verbose
Interfaces_matlab, 758
mex_test_matrix
Interfaces_matlab, 759
mex_test_matrix.c, 1720
mex_test_mess_direct_cholpackage_t

Interfaces_matlab, 760
mex_test_mess_direct_lupackage_t
Interfaces_matlab, 760
mex_test_mess_equation_t
Interfaces_matlab, 760
mex_test_mess_memusage_t
Interfaces_matlab, 760
mex_test_mess_multidirect_t
Interfaces_matlab, 760
mex_test_mess_norm_t
Interfaces_matlab, 760
mex_test_mess_operation_t
Interfaces_matlab, 760
mex_test_mess_parameter_t
Interfaces_matlab, 760
mex_test_mess_residual_t
Interfaces_matlab, 760
mex_test_options
Interfaces_matlab, 760
mex_test_options.c, 1721
mex_test_status
Interfaces_matlab, 761
mex_test_status.c, 1721
mex_test_vector
Interfaces_matlab, 762
mex_test_vector.c, 1721
mexFunction
mess_call.c, 1631
mgs.c, 1721
mgsadd.c, 1722
min
iluk.c, 1462
wachspress.c, 1975
minmax.c, 1722
misc.c, 1722, 1723
misc.h, 1725
misc/init.c
exit_done, 1464
init_done, 1464
print_error, 1464
print_info, 1464
print_print, 1464
print_warn, 1464
Miscellaneous Functions, 569
mess_lrcfadi_ccsvd, 570
mess_lrcfadi_colcompress, 571
mess_lrcfadi_fac2diff, 572
mess_lrcfadi_facFdiff, 573
mess_lrcfadi_lp_mnmx, 574
mess_lrcfadi_lp_s, 575
mess_lrcfadi_para_wachspress, 575
mess_lrcfadi_res2, 576
mess_lrcfadi_res2BK, 577
mess_lrcfadi_res2g, 578
mess_lrcfadi_res2g_so, 579
mess_lrcfadi_res2gBK, 579
mess_lrcfadi_res2gt, 580
mess_lrcfadi_res2gt_so, 581
mess_lrcfadi_res2nm, 582
mess_lrcfadi_res2nmg, 582
mess_lrcfadi_res2t, 583
Miscellaneous Functions and Functions Related to Vectors, 727
Miscellaneous Helper Functions, 672
__mess_calloc, 675
__mess_free, 676
__mess_malloc, 676
__mess_realloc, 676
isfinite, 675
mess_datatype_t_str, 676
mess_direct_cholpackage_t_str, 677
mess_direct_lupackage_t_str, 677
mess_direct_st_offset, 677
mess_eps, 678
mess_equation_st_offset, 678
mess_equation_t_str, 678
mess_exit, 679
mess_freelist, 675
mess_freelist_add_mess_equation, 679
mess_freelist_add_mess_matrix, 679
mess_freelist_add_mess_options, 679
mess_freelist_add_mess_status, 679
mess_freelist_add_mess_vector, 679
mess_freelist_add_ptr, 680
mess_freelist_clear, 680
mess_freelist_init, 680
mess_freelist_print, 680
mess_git_branch, 681
mess_git_id, 681
mess_have_amd, 681
mess_have_arpack, 682
mess_have_bzip2, 682
mess_have_cholmod, 682
mess_have_colamd, 682
mess_have_csparse, 682
mess_have_matio, 682
mess_have_mess64, 682
mess_have_mklpardiso, 682
mess_have_openmp, 682
mess_have_superlu, 682
mess_have_umfpack, 682
mess_have_zlib, 682
mess_init, 682
mess_int_t_size, 682
mess_is_debug, 683
mess_matrix_st_offset, 683
mess_memusage_t_str, 683
mess_multidirect_t_str, 683
mess_norm_t_str, 683
mess_openmp_info, 684
mess_operation_t_str, 684
mess_options_st_offset, 684
mess_parameter_t_str, 685
mess_print_bytes, 685
mess_print_format_double, 685
mess_print_format_double_cpx, 685

mess_print_format_select, 685
mess_residual_t_str, 686
mess_set_malloc, 686
mess_sign_scale_t_str, 686
mess_status_st_offset, 687
mess_storage_t_str, 687
mess_symmetry_t_str, 687
mess_vector_st_offset, 688
mess_version, 688
mess_version_major, 688
mess_version_minor, 688
mess_version_patch, 688
mess_version_verbose, 688
nearbyint, 689
mklpardiso.c, 1731
mess_direct_create_mklpardiso, 1731
mess_direct_create_mklpardiso_control, 1732
mklpardiso_clear, 1733
mklpardiso_error, 1734
mklpardiso_solve, 1734
mklpardiso_solve_matrix, 1735
mklpardiso_solve_vector, 1735
mklpardiso_solveh, 1736
mklpardiso_solvem, 1737
mklpardiso_solvemh, 1738
mklpardiso_solvemt, 1738
mklpardiso_solver, 1731
mklpardiso_solvet, 1739
mklpardiso_clear
mklpardiso.c, 1733
mklpardiso_error
mklpardiso.c, 1734
mklpardiso_solve
mklpardiso.c, 1734
mklpardiso_solve_matrix
mklpardiso.c, 1735
mklpardiso_solve_vector
mklpardiso.c, 1735
mklpardiso_solveh
mklpardiso.c, 1736
mklpardiso_solvem
mklpardiso.c, 1737
mklpardiso_solvemh
mklpardiso.c, 1738
mklpardiso_solvemt
mklpardiso.c, 1738
mklpardiso_solver, 923
error, 924
iparm, 924
maxfct, 924
message_level, 924
mklpardiso.c, 1731
mnum, 924
mtype, 924
num_procs, 924
phase, 924
pt, 924
solver, 924
work, 924
mklpardiso_solvet
mklpardiso.c, 1739
mlu
jobdata_st, 818
MM
__glyap_dae2, 769
blas_defs.h, 1003
mmwrite
tests/data/bips98_606/source.txt, 1855
tests/data/siso_ww_vref_6405/source.txt, 1856
mnum
mklpardiso_solver, 924
Model Order Reduction, 371
module_state, 924
error, 925
Msolver
__glyap_so1, 771
__glyap_so2, 773
msolverid
VWjob, 947
mtype
mklpardiso_solver, 924
mult.c, 1740
F77_GLOBAL, 1740
multi_umfpack, 925
cols, 925
Control, 925
cpx, 925
datatypes, 925
Info, 925
nlu, 925
numeric, 926
rows, 926
multidirect.c, 1741
DTCHECK, 1742
DTINIT, 1742
selected_solver, 1742
selected_solver_mutex, 1742
multidirect_select.c, 1742
multilu, 926
background, 927
cols, 927
datatypes, 927
lind, 927
lnnz, 927
lp, 927
lstore_type, 927
lvalues, 927
lvalues_cpx, 927
mutex_ready, 927
nlu, 928
p, 928
pinv, 928
q, 928
qinv, 928
ready, 928
rows, 928

sig_ready, 928
solverpool, 928
store_type, 928
uind, 928
unnz, 928
up, 928
ustore_type, 928
uvalues, 928
uvalues_cpx, 928
workcolptr, 928
workrowptr, 928
multilu_adavanced.c, 1743
DTCHECK, 1744
DTINIT, 1744
IS_COMPLEX, 1744
IS_REAL, 1744
job_decompose_complex, 1744
job_decompose_real, 1744
jobdata, 1744
multilu_clear, 1744
multilu_getL, 1745
multilu_getp, 1746
multilu_getq, 1746
multilu_getU, 1746
multilu_init, 1747
multilu_memsize, 1747
multilu_solve, 1747
multilu_solveh, 1748
multilu_solvem, 1749
multilu_solvemh, 1750
multilu_solvemt, 1751
multilu_solvet, 1752
pinv, 1754
WORK_BACKGROUND, 1744
multilu_clear
multilu_adavanced.c, 1744
multilu_getL
multilu_adavanced.c, 1745
multilu_getp
multilu_adavanced.c, 1746
multilu_getq
multilu_adavanced.c, 1746
multilu_getU
multilu_adavanced.c, 1746
multilu_init
multilu_adavanced.c, 1747
multilu_memsize
multilu_adavanced.c, 1747
multilu_solve
multilu_adavanced.c, 1747
multilu_solveh
multilu_adavanced.c, 1748
multilu_solvem
multilu_adavanced.c, 1749
multilu_solvemh
multilu_adavanced.c, 1750
multilu_solvemt
multilu_adavanced.c, 1751
multilu_solvet
multilu_adavanced.c, 1752
multilu_umfpack.c, 1755
IS_COMPLEX, 1756
IS_REAL, 1756
multiumf_clear, 1756
multiumf_solve, 1757
multiumf_solveh, 1757
multiumf_solvem, 1758
multiumf_solvemh, 1759
multiumf_solvemt, 1760
multiumf_solvet, 1761
umfpack_dl_defaults, 1756
umfpack_dl_free_numeric, 1756
umfpack_dl_free_symbolic, 1756
umfpack_dl_load_numeric, 1756
umfpack_dl_numeric, 1756
umfpack_dl_report_info, 1756
umfpack_dl_report_numeric, 1756
umfpack_dl_save_numeric, 1756
umfpack_dl_solve, 1756
umfpack_dl_symbolic, 1756
umfpack_zl_defaults, 1756
umfpack_zl_free_numeric, 1756
umfpack_zl_free_symbolic, 1756
umfpack_zl_load_numeric, 1756
umfpack_zl_numeric, 1756
umfpack_zl_report_info, 1756
umfpack_zl_report_numeric, 1756
umfpack_zl_save_numeric, 1756
umfpack_zl_solve, 1756
umfpack_zl_symbolic, 1756
Multisolvers, 322
mess_multidirect_create, 322
mess_multidirect_create_sparse_lu, 324
mess_multidirect_create_umfpack, 326
mess_multidirect_select, 327
multiumf_clear
multilu_umfpack.c, 1756
multiumf_solve
multilu_umfpack.c, 1757
multiumf_solveh
multilu_umfpack.c, 1757
multiumf_solvem
multilu_umfpack.c, 1758
multiumf_solvemh
multilu_umfpack.c, 1759
multiumf_solvemt
multilu_umfpack.c, 1760
multiumf_solvet
multilu_umfpack.c, 1761
mutex_ready
multilu, 927
mvp
check_arpack_lanczos.c, 1034
mess_mvpcall_st, 885
mvp.c, 1762
mvp_arnoldi

arnoldi.c, 964
mvp_arnoldi_inv
arnoldi.c, 964
mvp_arnoldig
arnoldi.c, 965
mvp_arnoldig_inv
arnoldi.c, 966
mvp_hermitian
mvpcall.c, 1763
mvp_normal
mvpcall.c, 1763
mvp_parameter
para.c, 1791
mvp_parameterinv
para.c, 1792
mvp_transpose
mvpcall.c, 1764
mvpcall.c, 1762
mvp_hermitian, 1763
mvp_normal, 1763
mvp_transpose, 1764
mvpdata, 929
A, 929
h1, 929
h2, 929
haveUW, 929
U, 930
W, 930
mvpdatag, 930
A, 931
Asolver, 931
Atilde, 931
E, 931
Esolver, 931
h1, 931
h2, 931
haveUW, 931
JaySolver, 931
U, 931
u, 931
Utilde, 931
v, 931
vj, 931
W, 931
y, 932
z, 932
mxCreateComplexScalar
Interfaces_matlab, 763
mxIsSubClass
Interfaces_matlab, 763
mxIsSubClass.c, 1765
my_mexError
interface_matlab.c, 1466
my_mexPrintf
interface_matlab.c, 1466
my_mexWarn
interface_matlab.c, 1466
my_module_MP_my_sub
symbols/my_module_MP_my_sub.c, 1766
VerifyC/CMakeFiles/FortranCInterface/symbols/my↩
_module_MP_my_sub.c, 1766
my_module_MP_my_sub.c, 1766
my_module_mp_my_sub_
symbols/my_module_mp_my_sub_.c, 1766
VerifyC/CMakeFiles/FortranCInterface/symbols/my↩
_module_mp_my_sub_.c, 1767
my_module_mp_my_sub_.c, 1766
my_module_my_sub_
symbols/my_module_my_sub_.c, 1767
VerifyC/CMakeFiles/FortranCInterface/symbols/my↩
_module_my_sub_.c, 1767
my_module_my_sub_.c, 1767
my_moduleSmy_sub.c, 1767
my_sub
symbols/my_sub.c, 1768
VerifyC/CMakeFiles/FortranCInterface/symbols/my↩
_sub.c, 1768
my_sub.c, 1768
my_sub_
symbols/my_sub_.c, 1768
VerifyC/CMakeFiles/FortranCInterface/symbols/my↩
sub.c, 1769
my_sub_.c, 1768
my_sub__
symbols/my_sub__.c, 1769
VerifyC/CMakeFiles/FortranCInterface/symbols/my↩
_sub__.c, 1769
my_sub__.c, 1769
myfort
CMakeFiles/FortranCInterface/CMakeCache.txt,
1256
CMakeFiles/FortranCInterface/VerifyC/CMake↩
Files/FortranCInterface/CMakeCache.txt,
1257
mymodule_MP_mysub
symbols/mymodule_MP_mysub.c, 1770
VerifyC/CMakeFiles/FortranCInterface/symbols/mymodule↩
_MP_mysub.c, 1770
mymodule_MP_mysub.c, 1769, 1770
mymodule_mp_mysub_
symbols/mymodule_mp_mysub_.c, 1770
VerifyC/CMakeFiles/FortranCInterface/symbols/mymodule↩
_mp_mysub_.c, 1770
mymodule_mp_mysub_.c, 1770
mymodule_mysub_
symbols/mymodule_mysub_.c, 1770
VerifyC/CMakeFiles/FortranCInterface/symbols/mymodule↩
mysub.c, 1771
mymodule_mysub_.c, 1770
mymoduleSmysub.c, 1771
mysub
symbols/mysub.c, 1772
VerifyC/CMakeFiles/FortranCInterface/symbols/mysub.↩
c, 1772
mysub.c, 1771, 1772
mysub_

symbols/mysub_.c, 1772
VerifyC/CMakeFiles/FortranCInterface/symbols/mysub_.c, 1772
mysub_.c, 1772

N
blas_defs.h, 1003
n
__iluk, 776
banded_lapack, 807
blas_defs.h, 1003
CCSRFmt, 809
cholesky_solver, 809
hessenberg_solver, 815
lapack_solver, 820
VWjob, 947
n1
__glyap_dae1, 766
tests/data/TripleChain/source.txt, 1857
n2
__glyap_dae1, 766
blas_defs.h, 1003
n_eqns
mess_freelist_st, 853
n_internal_status
mess_status_st, 907
n_matrices
mess_freelist_st, 853
n_opts
mess_freelist_st, 853
n_ptrs
mess_freelist_st, 853
n_stats
mess_freelist_st, 853
n_vectors
mess_freelist_st, 853
NEW_LD
matrix/alloc.c, 960
NO_IMPORT_ARRAY
matrix.c, 1620
python/c_interface/conv/options.c, 1790
vector.c, 1967
NPY_ARRAY_OWNDATA
matrix.c, 1620
python/c_interface/conv/options.c, 1790
vector.c, 1967
NPY_NO_DEPRECATED_API
Interface to Python, 623
matrix.c, 1620
python/c_interface/conv/options.c, 1790
vector.c, 1967
NRHS
blas_defs.h, 1003
name
_equations_t, 795
_mess_call, 797
mess_direct_st, 839
mess_multidirect_st, 881
NB
blas_defs.h, 1003
nband
banded_lapack, 807
ncv
mess_eigen_arpack_options_t, 843
ndims
MESS_matrix, 867
MESS_vector, 920
nearbyint
Miscellaneous Helper Functions, 689
need_restart
__mess_solver_status, 791
newlu.c, 1772
FLIP, 1773
lhsolvec, 1773
lsolve, 1773
lsolvec, 1774
lsolverc, 1774
ltsolve, 1774
ltsolvec, 1774
ltsolverc, 1774
MARKED, 1773
MARK, 1773
newlu_clear, 1774
newlu_getL, 1774
newlu_getpermp, 1775
newlu_getpermq, 1775
newlu_getU, 1775
newlu_inverse, 1776
newlu_solve, 1777
newlu_solveh, 1778
newlu_solvem, 1779
newlu_solvemh, 1780
newlu_solvemt, 1781
newlu_solvet, 1782
perm_inv, 1783
UNFLIP, 1773
uhsolvec, 1783
usolve, 1783
usolvec, 1783
usolverc, 1783
utsolve, 1784
utsolvec, 1784
utsolverc, 1784
newlu_clear
newlu.c, 1774
newlu_data, 932
cols, 933
data_type, 933
L, 933
pinv, 933
q, 933
rows, 933
U, 933
newlu_getL
newlu.c, 1774
newlu_getpermp
newlu.c, 1775

newlu_getpermq
newlu.c, 1775
newlu_getU
newlu.c, 1775
newlu_inverse
newlu.c, 1776
newlu_solve
newlu.c, 1777
newlu_solveh
newlu.c, 1778
newlu_solvem
newlu.c, 1779
newlu_solvemh
newlu.c, 1780
newlu_solvemt
newlu.c, 1781
newlu_solvet
newlu.c, 1782
next
mess_threadpooljob_st, 913
nextid
mess_threadpool_st, 911
nlu
multi_umfpack, 925
multilu, 928
nm_gpStep
mess_options_st, 895
nm_linesearch
mess_options_st, 895
nm_maxit
mess_options_st, 895
nm_output
mess_options_st, 896
nm_res2_tol
mess_options_st, 896
nm_singleshifts
mess_options_st, 896
nm_stepfunction
mess_options_st, 896
nm_stepfunction_aux
mess_options_st, 896
nnz
CCSRFmt, 809
mess_matrix_file_st, 870
mess_matrix_st, 872
norm.c, 1784, 1786
norm2data, 933
C, 934
x1, 934
norm2mvp
matrix/norm.c, 1785
norminv2data, 934
sol, 935
x1, 935
norminv2mvp
matrix/norm.c, 1785
not_empty
mess_threadpool_st, 911
not_full
mess_threadpool_st, 912
np
tests/data/bips98_606/source.txt, 1855
tests/data/siso_ww_vref_6405/source.txt, 1856
nrhs
blas_defs.h, 1003
num_mvp
__mess_solver_status, 791
num_procs
mklpardiso_solver, 924
num_solvers
__glyap_dae2, 769
num_threads
mess_threadpool_st, 912
Numeric
umfpack_control_solver, 946
numeric
multi_umfpack, 926

OCTMESS_ERROR
MESS_error.h, 1645
OCTMESS_GENERATOR
cpp_interface/MESS_matrix/MESS_matrix.cc, 1647
cpp_interface/MESS_vector/MESS_vector.cc, 1689
MESS_enum.cc, 1634
MESS_enums.cc, 1639
MESS_enums.h, 1642
MESS_matrix.h, 1651
MESS_matrix_install.cc, 1665
MESS_matrix_ops_mm_generator.h, 1673
MESS_matrix_ops_mm_ocm.cc, 1679
MESS_matrix_ops_mm_om.cc, 1679
MESS_matrix_ops_mm_oscm.cc, 1681
MESS_matrix_ops_mm_osm.cc, 1682
MESS_vector.h, 1692
MESS_vector_install.cc, 1701
MESS_vector_ops_mv_generator.h, 1710
MESS_vector_ops_ocm_mv.cc, 1716
MESS_vector_ops_om_mv.cc, 1717
MESS_vector_ops_oscm_mv.cc, 1717
MESS_vector_ops_osm_mv.cc, 1717
OCTMESS_MESS_MATRIX_PRINT
octmess/src/MESS_matrix.cc, 1648
OCTMESS_MESS_VECTOR_PRINT
octmess/src/MESS_vector.cc, 1690
octmess/src/MESS_matrix.cc
DEFUN_DLD, 1648
OCTMESS_MESS_MATRIX_PRINT, 1648
octmess/src/MESS_options.cc
DEFUN_DLD, 1686
octmess/src/MESS_status.cc
DEFUN_DLD, 1688
octmess/src/MESS_vector.cc
DEFUN_DLD, 1690
OCTMESS_MESS_VECTOR_PRINT, 1690
octmess_matrix_conv

MESS_matrix.h, 1652
offsets.c, 1786
MESS_OFFSET, 1787
omega
__mess_solver_options, 790
ones.c, 1787, 1788
op_add
MESS_matrix.h, 1652
MESS_matrix_ops_mm_mm.cc, 1673
MESS_vector.h, 1692
MESS_vector_ops_mv_mv.cc, 1711
op_eq
MESS_enums.cc, 1641
MESS_enums.h, 1642, 1643
op_herm_ldiv
MESS_matrix.h, 1652, 1653
MESS_matrix_ops_mm_mm.cc, 1673
MESS_vector.h, 1692, 1693
MESS_vector_ops_mm_mv.cc, 1707
op_herm_mul
MESS_matrix.h, 1653
MESS_matrix_ops_mm_mm.cc, 1674
MESS_vector.h, 1693
MESS_vector_ops_mm_mv.cc, 1707
MESS_vector_ops_mv_mv.cc, 1712
op_hermitian
MESS_matrix.h, 1653
MESS_matrix_ops_mm_mm.cc, 1674
op_ldiv
MESS_matrix.h, 1653, 1654
MESS_matrix_ops_mm_mm.cc, 1674
MESS_vector.h, 1693, 1694
MESS_vector_ops_mm_mv.cc, 1708
op_mul
MESS_matrix.h, 1654–1657
MESS_matrix_ops_mm_mm.cc, 1675
MESS_matrix_ops_mm_os.cc, 1680
MESS_matrix_ops_mm_osc.cc, 1681
MESS_vector.h, 1694–1696
MESS_vector_ops_mm_mv.cc, 1708
MESS_vector_ops_mv_os.cc, 1714, 1715
MESS_vector_ops_mv_osc.cc, 1715, 1716
op_mul_herm
MESS_matrix.h, 1657
MESS_matrix_ops_mm_mm.cc, 1675
op_mul_trans
MESS_matrix.h, 1657, 1658
MESS_matrix_ops_mm_mm.cc, 1675
op_ne
MESS_enums.cc, 1641
MESS_enums.h, 1643
op_not
MESS_matrix.h, 1658
MESS_matrix_ops_mm_mm.cc, 1676
MESS_vector.h, 1696
MESS_vector_ops_mv_mv.cc, 1712
op_sub
MESS_matrix.h, 1658, 1659
MESS_matrix_ops_mm_mm.cc, 1676
MESS_vector.h, 1697
MESS_vector_ops_mv_mv.cc, 1713
op_to_int
equations.c, 1421
op_trans_ldiv
MESS_matrix.h, 1659
MESS_matrix_ops_mm_mm.cc, 1676
MESS_vector.h, 1697, 1698
MESS_vector_ops_mm_mv.cc, 1708
op_trans_mul
MESS_matrix.h, 1659, 1660
MESS_matrix_ops_mm_mm.cc, 1677
MESS_vector.h, 1698, 1699
MESS_vector_ops_mm_mv.cc, 1709
MESS_vector_ops_mv_mv.cc, 1713
op_transpose
MESS_matrix.h, 1660
MESS_matrix_ops_mm_mm.cc, 1677
op_uminus
MESS_matrix.h, 1660
MESS_matrix_ops_mm_mm.cc, 1677
MESS_vector.h, 1699
MESS_vector_ops_mv_mv.cc, 1714
op_uplus
MESS_matrix.h, 1661
MESS_matrix_ops_mm_mm.cc, 1678
MESS_vector.h, 1699
MESS_vector_ops_mv_mv.cc, 1714
op_xldiv
MESS_matrix.h, 1661, 1662
MESS_matrix_ops_mm_mm.cc, 1678
MESS_vector.h, 1699, 1700
MESS_vector_ops_mm_mv.cc, 1709
op_xmul
MESS_matrix.h, 1662
MESS_matrix_ops_mm_mm.cc, 1678
MESS_vector.h, 1700
MESS_vector_ops_mm_mv.cc, 1709
openmp.c, 1788
_GNU_SOURCE, 1788
operator bool
MESS_direct, 831
MESS_equation, 846
MESS_matrix, 868
MESS_multidirect, 878
MESS_options, 891
MESS_status, 904
MESS_vector, 921
operator mess_datatype_t
MESS_datatype_t, 828
operator mess_direct
MESS_direct, 831
operator mess_direct_cholpackage_t
MESS_direct_cholpackage_t, 833
operator mess_direct_lupackage_t
MESS_direct_lupackage_t, 836
operator mess_equation

MESS_equation, 846
operator mess_equation_t
MESS_equation_t, 851
operator mess_matrix
MESS_matrix, 868
operator mess_memusage_t
MESS_memusage_t, 874
operator mess_multidirect
MESS_multidirect, 878
operator mess_multidirect_t
MESS_multidirect_t, 883
operator mess_operation_t
MESS_operation_t, 887
operator mess_options
MESS_options, 891
operator mess_parameter_t
MESS_parameter_t, 898
operator mess_residual_t
MESS_residual_t, 901
operator mess_status
MESS_status, 904
operator mess_storage_t
MESS_storage_t, 909
operator mess_vector
MESS_vector, 921
operator!=
MESS_datatype_t, 828
MESS_direct_cholpackage_t, 833
MESS_direct_lupackage_t, 836
MESS_equation_t, 851
MESS_memusage_t, 874
MESS_multidirect_t, 883
MESS_operation_t, 887
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_storage_t, 909
operator==
MESS_datatype_t, 828
MESS_direct_cholpackage_t, 834
MESS_direct_lupackage_t, 836
MESS_equation_t, 851
MESS_memusage_t, 875
MESS_multidirect_t, 884
MESS_operation_t, 887
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_storage_t, 910
opt
__presordata, 793
bicgstab_solver, 808
parameter_mvp_call, 939
options
__matlab_equation, 780
options.c, 1788, 1789
opts
mess_freelist_st, 853
orth.c, 1790
output
mess_h2_options_st, 855
output.txt, 1791
outputs
mess_dynsys_st, 842

p
bicgstab_solver, 808
multilu, 928
tests/data/siso_ww_vref_6405/source.txt, 1856
PLATFORM_ID
CMakeCCompilerId.c, 1258
CMakeCXXCompilerId.cpp, 1259
PRE_POST_FUNCTION2
equations.c, 1419
PRE_POST_FUNCTION
equations.c, 1419
PY_ARRAY_UNIQUE_SYMBOL
Interface to Python, 623
matrix.c, 1620
python/c_interface/conv/options.c, 1790
vector.c, 1967
PY_GET_DOUBLE
Interface to Python, 623
PY_GET_LONG
Interface to Python, 623
PY_GET_MATRIX
Interface to Python, 623
PY_GET_VECTOR
Interface to Python, 623
PY_SET_BOOL
Interface to Python, 624
PY_SET_DOUBLE
Interface to Python, 624
PY_SET_LONG
Interface to Python, 624
PY_SET_VECTOR
Interface to Python, 624
PYCMESS_ERROR
Interface to Python, 624
PYMESS_MESS_HAVE_GENERATOR
mess_version.c, 1718
PYMESS_READ_MATRIX
Interface to Python, 625
pair
jobdata_st, 818
pair_mess_int_t2, 935
l1, 935
l2, 935
pair_mess_int_t2_complex, 935
d, 936
l1, 936
l2, 936
pair_mess_int_t2_double, 936
d, 937
l1, 937
l2, 937
pair_mess_int_t_complex, 937
d, 937
l, 937

pair_mess_int_t_double, 938
d, 938
l, 938
para.c, 1791
mvp_parameter, 1791
mvp_parameterinv, 1792
parameter
callback_equation.c, 1009
equation_glyap.c, 1355
equation_glyap_dae1.c, 1364
equation_glyap_dae2.c, 1380
equation_glyap_so1.c, 1390
equation_glyap_so2.c, 1401
equation_lyap.c, 1412
equation_stable.c, 1417
equations.c, 1421
mess_equation_st, 849
parameter_mvp_call, 939
eqn, 939
have_mass, 939
opt, 939
parent
mess_equation_st, 849
perm
banded_lapack, 807
perm.c, 1792, 1796
perm_inv
csparse.c, 1296
csparse_chol.c, 1310
matrix/perm.c, 1796
newlu.c, 1783
permute
check_perm.c, 1203
check_vector_perm.c, 1249
phase
mklpardiso_solver, 924
pinv
multilu, 928
multilu_adavanced.c, 1754
newlu_data, 933
transfer.c, 1886
pinv.c, 1799
plot
__mess_plotter, 786
__mess_plotterScript, 789
x11init.c, 1979
Plot Export, 663
MESS_PLOT_LIN, 665
MESS_PLOT_LOG, 665
mess_plot_axis_scale_t, 665
mess_plotExport, 665
mess_plotExport_addData, 665
mess_plotExport_clear, 665
mess_plotExport_clearData, 666
mess_plotExport_createGnuScript, 666
mess_plotExport_createGnuScript_filename, 666
mess_plotExport_createTikzScript, 667
mess_plotExport_createTikzScript_filename, 667
mess_plotExport_init, 668
mess_plotExport_save, 668
mess_plotExport_setColor, 669
mess_plotExport_setLabel, 669
mess_plotExport_setLegendPos, 670
mess_plotExport_setType, 670
mess_plotExportData, 665
plot_len
__mess_plotter, 786
__mess_plotterScript, 789
plot_scriptExporter.c, 1799
mess_plotExport_createGnuScript_int, 1800
mess_plotExport_createTikzScript_int, 1800
plot_scriptExporter.h, 1801
plot_x11.h, 1802
Plotting Interface, 656
pp
tests/data/bips98_606/source.txt, 1855
tests/data/siso_ww_vref_6405/source.txt, 1856
pperm
jobdata_st, 818
pre
bicgstab_solver, 808
precond.c, 1803
precond/diag.c
__mess_precond_diag_clear, 1327
__mess_precond_diag_complexclear, 1328
__mess_precond_diag_solve, 1328
__mess_precond_diag_solvecomplex, 1329
precond/sor.c
__mess_precond_sor_clear, 1851
__mess_precond_sor_solve, 1851
__mess_precond_ssor_solve, 1852
Preconditioners, 344
mess_precond_clear, 344
mess_precond_diag, 345
mess_precond_ilu0, 345
mess_precond_iluk, 346
mess_precond_init, 347
mess_precond_sor, 347
mess_precond_ssor, 348
print
MESS_datatype_t, 828
MESS_direct, 831
MESS_direct_cholpackage_t, 834
MESS_direct_lupackage_t, 836
MESS_equation, 846
MESS_equation_t, 851
MESS_matrix, 868
MESS_memusage_t, 875
MESS_multidirect, 878
MESS_multidirect_t, 884
MESS_operation_t, 887
MESS_options, 891
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_status, 904, 905
MESS_storage_t, 910

MESS_vector, 921
print.c, 1803
print_format, 1803
print_format_mutex, 1803
print_as_scalar
MESS_datatype_t, 828
MESS_direct, 831
MESS_direct_cholpackage_t, 834
MESS_direct_lupackage_t, 836
MESS_equation, 846
MESS_equation_t, 851
MESS_matrix, 868
MESS_memusage_t, 875
MESS_multidirect, 878
MESS_multidirect_t, 884
MESS_operation_t, 887
MESS_options, 891
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_status, 905
MESS_storage_t, 910
MESS_vector, 921
print_error
misc/init.c, 1464
print_format
print.c, 1803
print_format_mutex
print.c, 1803
print_handle
interface_matlab.c, 1467
print_info
misc/init.c, 1464
print_name_tag
MESS_direct, 832
MESS_equation, 847
MESS_matrix, 868
MESS_multidirect, 879
MESS_options, 891
MESS_status, 905
MESS_vector, 921
print_print
misc/init.c, 1464
print_raw
MESS_datatype_t, 828
MESS_direct, 832
MESS_direct_cholpackage_t, 834
MESS_direct_lupackage_t, 836
MESS_equation, 847
MESS_equation_t, 851
MESS_matrix, 868
MESS_memusage_t, 875
MESS_multidirect, 879
MESS_multidirect_t, 884
MESS_operation_t, 887
MESS_options, 891
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_status, 905
MESS_storage_t, 910
MESS_vector, 921
print_warn
misc/init.c, 1464
profile.txt, 1803
proj_space
mess_options_st, 896
proj_sym.c, 1803
project.c, 1804
pt
mklpardiso_solver, 924
ptr
MESS_direct, 832
MESS_equation, 847
MESS_matrix, 869
MESS_multidirect, 879
MESS_options, 892
MESS_status, 905
MESS_vector, 921
ptrs
mess_freelist_st, 853
PyArray_CALL_DATA
matrix.c, 1620
python/c_interface/conv/options.c, 1790
vector.c, 1967
PyArray_CALL_ISSIGNED
matrix.c, 1620
python/c_interface/conv/options.c, 1790
vector.c, 1967
PyArray_CALL_ISUNSIGNED
matrix.c, 1620
python/c_interface/conv/options.c, 1790
vector.c, 1967
PyArray_ENABLEFLAGS
matrix.c, 1620
python/c_interface/conv/options.c, 1790
vector.c, 1967
pymess.c, 1804
pymess.h, 1805
pymess_care
Interface to Python, 638
pymess_dense_nm_gmpare
Interface to Python, 639
pymess_direct_chol_select
Interface to Python, 640
pymess_direct_select
Interface to Python, 641
pymess_eps
Interface to Python, 641
pymess_glyap
Interface to Python, 642
pymess_gstein
Interface to Python, 642
pymess_is_reimplemented
Interface to Python, 643
pymess_lradi
Interface to Python, 644
pymess_lrnm

Interface to Python, 644
pymess_lyap
Interface to Python, 645
pymess_mess_git_branch
Interface to Python, 647
pymess_mess_git_id
Interface to Python, 648
pymess_mess_have_amd
Interface to Python, 648
pymess_mess_have_arpack
Interface to Python, 648
pymess_mess_have_bzip2
Interface to Python, 648
pymess_mess_have_cholmod
Interface to Python, 648
pymess_mess_have_colamd
Interface to Python, 648
pymess_mess_have_csparse
Interface to Python, 648
pymess_mess_have_matio
Interface to Python, 648
pymess_mess_have_mess64
Interface to Python, 648
pymess_mess_have_mklpardiso
Interface to Python, 648
pymess_mess_have_openmp
Interface to Python, 648
pymess_mess_have_superlu
Interface to Python, 648
pymess_mess_have_umfpack
Interface to Python, 648
pymess_mess_have_zlib
Interface to Python, 649
pymess_mess_is_debug
Interface to Python, 649
pymess_mess_version
Interface to Python, 649
pymess_mess_version_major
Interface to Python, 649
pymess_mess_version_minor
Interface to Python, 649
pymess_mess_version_patch
Interface to Python, 649
pymess_mess_version_verbose
Interface to Python, 650
pymess_multidirect_select
Interface to Python, 650
pymess_set_errorlevel
Interface to Python, 651
pymess_sylvester_sparsedense
Interface to Python, 651
pymess_test_equation
Interface to Python, 652
pymess_test_matrix
Interface to Python, 653
pymess_test_status
Interface to Python, 653
pymess_test_vector
Interface to Python, 653
python/c_interface/conv/options.c
ConvStringtoC, 1790
NO_IMPORT_ARRAY, 1790
NPY_ARRAY_OWNDATA, 1790
NPY_NO_DEPRECATED_API, 1790
PY_ARRAY_UNIQUE_SYMBOL, 1790
PyArray_CALL_DATA, 1790
PyArray_CALL_ISSIGNED, 1790
PyArray_CALL_ISUNSIGNED, 1790
PyArray_ENABLEFLAGS, 1790

Q
_sylv_solver_sd_st, 801
blas_defs.h, 1003
q
bicgstab_solver, 808
multilu, 928
newlu_data, 933
QA
_glyap3_st, 796
_sylv_solver_d_st, 799
QE
_glyap3_st, 796
_sylv_solver_d_st, 799
QF
_sylv_solver_d_st, 799
QH
_sylv_solver_d_st, 799
qinv
multilu, 928
qperm
jobdata_st, 818

REFINE
twoside_mgs.c, 1951
RELRES_TOL
dense_nm.c, 1325
RES2_COUNT
lib/lrcf_adi/lradi.c, 1597
RES_ABORT
lib/lrcf_adi/lradi.c, 1597
RHS
mess_equation_st, 850
RWORK
blas_defs.h, 1003
rail_371_c60.C, 1807
rand.c, 1807, 1808
rand50x50condest
check_condest.c, 1072
rand50x50norm1
check_norm.c, 1197
rand50x50norm2
check_norm.c, 1197
rand50x50norm2inv
check_norm.c, 1197
rand50x50normf
check_norm.c, 1197
rand50x50norminf

check_norm.c, 1198
randc50x50norm1
check_norm.c, 1198
randc50x50norm2
check_norm.c, 1198
randc50x50norm2inv
check_norm.c, 1198
randc50x50normf
check_norm.c, 1198
randc50x50norminf
check_norm.c, 1198
rank.c, 1809
rc2data, 940
x1, 940
x2, 940
Z, 940
Zold, 940
rc2mvp
rc2norm.c, 1810
rc2norm.c, 1809
IT_MAX, 1810
rc2mvp, 1810
rcFnorm.c, 1811
rdim
mess_bt_options_st, 824
mess_bt_status_st, 826
mess_h2_options_st, 855
read.c, 1811
_ParseIfmt, 1817
_ParseRfmt, 1817
__hb_read, 1813
__hb_read_data, 1813
__hb_read_info, 1813
__mm_read_data, 1814
__mm_read_info, 1815
_lowercase, 1815
_substr, 1817
_uppercase, 1818
LINE_LENGTH, 1812
MM_BANNER, 1812
MM_STR_ARR, 1812
MM_STR_COORD, 1812
MM_STR_CPX, 1812
MM_STR_GE, 1812
MM_STR_HER, 1812
MM_STR_INT, 1812
MM_STR_MATRIX, 1812
MM_STR_PAT, 1812
MM_STR_REAL, 1812
MM_STR_SKSYM, 1812
MM_STR_SYM, 1812
SET_LAST, 1812
TOKEN_LENGTH, 1812
ready
multilu, 928
real_complex.c, 1818
mess_vector_isreal, 1819
real_matrix_data
check_colscale.c, 1063
realimag.c, 1819
realloc_call_t
malloc.c, 1610
reimplemented.c, 1820
rel_change
__mess_lrcfadi_step, 781
mess_status_st, 907
rel_changes
mess_status_st, 907
relres
__mess_solver_status, 792
res
__mess_solver_status, 792
res2
__mess_lrcfadi_step, 781
res2.c, 1820
res2_0
mess_status_st, 907
res2_change
__mess_lrcfadi_step, 781
mess_status_st, 907
res2_data
lrcf_adi/norms/res2.c, 1822
res2_template.c, 1831
res2_data_st, 941
B, 942
eqn, 942
F, 942
G, 942
K, 942
M, 942
rt1, 942
rt2, 942
rt3, 942
rt4, 942
rtb, 942
rtc, 942
rtz, 942
t, 942
t1, 942
t2, 942
t3, 942
t4, 942
Z, 942
res2_mvp
lrcf_adi/norms/res2.c, 1822
res2_template.c, 1833
res2_mvpBK
lrcf_adi/norms/res2.c, 1823
res2_mvpt
lrcf_adi/norms/res2.c, 1824
res2_norm
mess_status_st, 907
res2_norms
mess_status_st, 907
res2_template.c, 1830
ARNOLDI_ITERATIONS, 1831

ARPACK_NCV, 1831
mess_lrcfadi_residual_lyap, 1831
mess_lrcfadi_residual_nm, 1832
res2_data, 1831
res2_mvp, 1833
res2g_mvp, 1834
res2gt_mvp, 1835
res2nm_data, 1831
res2nm_mvp_none, 1836
res2nm_mvp_transposed, 1837
res2nmg_mvp_none, 1838
res2nmg_mvp_transposed, 1839
res2t_mvp, 1840
res2g_mvp
lrcf_adi/norms/res2.c, 1825
res2_template.c, 1834
res2g_mvpBK
lrcf_adi/norms/res2.c, 1826
res2gt_mvp
lrcf_adi/norms/res2.c, 1827
res2_template.c, 1835
res2nm_data
lrcf_adi/norms/res2.c, 1822
res2_template.c, 1831
res2nm_data_st, 943
A, 944
B, 944
C, 944
eqn, 944
M, 944
t1, 944
t2, 944
t3, 944
t4, 944
tb, 944
tc, 944
tz, 944
Z, 944
res2nm_mvp
lrcf_adi/norms/res2.c, 1828
res2nm_mvp_none
res2_template.c, 1836
res2nm_mvp_transposed
res2_template.c, 1837
res2nmg_mvp
lrcf_adi/norms/res2.c, 1829
res2nmg_mvp_none
res2_template.c, 1838
res2nmg_mvp_transposed
res2_template.c, 1839
res2t_mvp
res2_template.c, 1840
Residual Computations, 300
mess_direct_care_res2, 301
mess_direct_generalized_lyapunov_res2, 301
mess_direct_generalized_stein_res2, 303
mess_direct_generalized_sylvester_res2, 304
mess_direct_lyapunov_res2, 305
mess_direct_res2, 307
mess_direct_res2m, 308
mess_direct_stein_res2, 309
mess_direct_sylvester_res2, 310
mess_direct_sylvestersg_res2, 312
residual_method
mess_options_st, 896
restarts
__mess_solver_options, 790
__mess_solver_status, 792
ret
jobdata_st, 818
retdata
th_done_entry, 945
rowptr
CCSRFmt, 809
mess_matrix_st, 872
rows
jobdata_st, 818
lapackqr_solver, 821
mess_direct_st, 839
mess_matrix_file_st, 870
mess_matrix_st, 872
mess_multidirect_st, 881
multi_umfpack, 926
multilu, 928
newlu_data, 933
rowsX
_sylv_solver_d_st, 799
_sylv_solver_sd_st, 801
rpath
tests/direct/CMakeFiles/check_backslash.↩ dir/link.txt, 1504
tests/direct/CMakeFiles/check_det.dir/link.txt, 1504
tests/direct/CMakeFiles/check_dgeqp3.dir/link.txt, 1505
tests/direct/CMakeFiles/check_direct_care.↩ dir/link.txt, 1505
tests/direct/CMakeFiles/check_getLU.dir/link.txt, 1506
tests/direct/CMakeFiles/check_hessenberg.↩ dir/link.txt, 1506
tests/direct/CMakeFiles/check_lapackqr.dir/link.txt, 1507
tests/direct/CMakeFiles/check_pinv.dir/link.txt, 1507
tests/direct/CMakeFiles/check_trisolve.dir/link.txt, 1508
tests/direct/CMakeFiles/glyap_test.dir/link.txt, 1508
tests/direct/CMakeFiles/gstein_test.dir/link.txt, 1508
tests/direct/CMakeFiles/inverse.dir/link.txt, 1509
tests/direct/CMakeFiles/linsolve.dir/link.txt, 1509
tests/direct/CMakeFiles/linsolve_multi.dir/link.txt, 1509
tests/direct/CMakeFiles/linsolvem.dir/link.txt, 1510
tests/direct/CMakeFiles/linsolvem_multi.dir/link.txt, 1510

tests/direct/CMakeFiles/sylvester_dense.dir/link.↩ txt, 1511
tests/direct/CMakeFiles/sylvester_sparsedense.↩ dir/link.txt, 1511
tests/eigen/CMakeFiles/check_arnoldi_template.↩ dir/link.txt, 1512
tests/eigen/CMakeFiles/check_arnoldi_template↩ _nrm.dir/link.txt, 1512
tests/eigen/CMakeFiles/check_eig.dir/link.txt, 1513
tests/eigen/CMakeFiles/check_eig2.dir/link.txt, 1513
tests/eigen/CMakeFiles/check_eig_complex.↩ dir/link.txt, 1513
tests/eigen/CMakeFiles/check_eig_hessenberg_↩ schur.dir/link.txt, 1514
tests/eigen/CMakeFiles/check_eig_hessenberg_↩ schur_largest_ev.dir/link.txt, 1514
tests/eigen/CMakeFiles/check_gsign.dir/link.txt, 1515
tests/eigen/CMakeFiles/check_schur.dir/link.txt, 1515
tests/eigen/CMakeFiles/check_sign.dir/link.txt, 1516
tests/eigen/CMakeFiles/check_svd.dir/link.txt, 1516
tests/eigen/CMakeFiles/check_svd_complex.↩ dir/link.txt, 1516
tests/eigen/CMakeFiles/check_svd_econ.↩ dir/link.txt, 1517
tests/eigen/CMakeFiles/check_svd_econ_↩ complex.dir/link.txt, 1517
tests/formats/CMakeFiles/check_cholmod.↩ dir/link.txt, 1518
tests/formats/CMakeFiles/check_cimport.dir/link.↩ txt, 1518
tests/formats/CMakeFiles/check_cimport_c2d.↩ dir/link.txt, 1519
tests/formats/CMakeFiles/check_convert.dir/link.↩ txt, 1519
tests/formats/CMakeFiles/check_mvpcall_↩ matrix.dir/link.txt, 1520
tests/formats/CMakeFiles/check_perm.dir/link.txt, 1520
tests/formats/CMakeFiles/check_real_imag_↩ part.dir/link.txt, 1521
tests/io/CMakeFiles/check_fputs_block.dir/link.txt, 1521
tests/io/CMakeFiles/check_matrixio.dir/link.txt, 1522
tests/io/CMakeFiles/check_vectorio.dir/link.txt, 1522
tests/itsolver/CMakeFiles/check_itsolver.dir/link.txt, 1523
tests/itsolver/CMakeFiles/check_precond.↩ dir/link.txt, 1523
tests/lrcf_adi/CMakeFiles/check_ber_sign_fac.↩ dir/link.txt, 1524
tests/lrcf_adi/CMakeFiles/check_dae2_handles_↩ adi.dir/link.txt, 1524
tests/lrcf_adi/CMakeFiles/check_dae2_handles_↩ para.dir/link.txt, 1525
tests/lrcf_adi/CMakeFiles/check_dense_nm.↩ dir/link.txt, 1525
tests/lrcf_adi/CMakeFiles/check_dense_nm_↩ tridiag.dir/link.txt, 1526
tests/lrcf_adi/CMakeFiles/check_linesearch.↩ dir/link.txt, 1526
tests/lrcf_adi/CMakeFiles/check_lradi_dae1.↩ dir/link.txt, 1527
tests/lrcf_adi/CMakeFiles/check_lradi_dae2.↩ dir/link.txt, 1527
tests/lrcf_adi/CMakeFiles/check_lradi_glyap.↩ dir/link.txt, 1528
tests/lrcf_adi/CMakeFiles/check_lradi_so.dir/link.↩ txt, 1528
tests/lrcf_adi/CMakeFiles/check_lradi_unsym.↩ dir/link.txt, 1529
tests/lrcf_adi/CMakeFiles/check_lrnm_dae1.↩ dir/link.txt, 1529
tests/lrcf_adi/CMakeFiles/check_lrnm_dae2.↩ dir/link.txt, 1530
tests/lrcf_adi/CMakeFiles/check_lrnm_g.dir/link.txt, 1530
tests/lrcf_adi/CMakeFiles/check_lrnm_so.↩ dir/link.txt, 1531
tests/lrcf_adi/CMakeFiles/check_lrnm_unsym.↩ dir/link.txt, 1531
tests/lrcf_adi/CMakeFiles/check_lyap_sign_fac.↩ dir/link.txt, 1532
tests/lrcf_adi/CMakeFiles/check_so1_handles.↩ dir/link.txt, 1532
tests/lrcf_adi/CMakeFiles/check_so2_handles.↩ dir/link.txt, 1533
tests/matrix/CMakeFiles/check_add.dir/link.txt, 1533
tests/matrix/CMakeFiles/check_addcol2p.↩ dir/link.txt, 1534
tests/matrix/CMakeFiles/check_any.dir/link.txt, 1534
tests/matrix/CMakeFiles/check_bandwidth.↩ dir/link.txt, 1535
tests/matrix/CMakeFiles/check_biorth.dir/link.txt, 1535
tests/matrix/CMakeFiles/check_cat.dir/link.txt, 1536
tests/matrix/CMakeFiles/check_colaxpy2.dir/link.↩ txt, 1536
tests/matrix/CMakeFiles/check_coldot.dir/link.txt, 1537
tests/matrix/CMakeFiles/check_coldotE.dir/link.txt, 1538
tests/matrix/CMakeFiles/check_coldotc.dir/link.txt, 1537
tests/matrix/CMakeFiles/check_coldotcE.dir/link.↩ txt, 1538
tests/matrix/CMakeFiles/check_colnorm.dir/link.txt,

1539
tests/matrix/CMakeFiles/check_colscale.dir/link.txt, 1539
tests/matrix/CMakeFiles/check_colsub.dir/link.txt, 1540
tests/matrix/CMakeFiles/check_colvecaxpy.↩ dir/link.txt, 1540
tests/matrix/CMakeFiles/check_colvecdot.↩ dir/link.txt, 1541
tests/matrix/CMakeFiles/check_colvecdotc.↩ dir/link.txt, 1541
tests/matrix/CMakeFiles/check_condest.dir/link.txt, 1542
tests/matrix/CMakeFiles/check_decomp.dir/link.txt, 1542
tests/matrix/CMakeFiles/check_diag.dir/link.txt, 1543
tests/matrix/CMakeFiles/check_dynorm2.dir/link.↩ txt, 1543
tests/matrix/CMakeFiles/check_dynorm2t.↩ dir/link.txt, 1544
tests/matrix/CMakeFiles/check_dynorm_indefinite2.↩ dir/link.txt, 1544
tests/matrix/CMakeFiles/check_dynorm_indefinitef.↩ dir/link.txt, 1545
tests/matrix/CMakeFiles/check_dynormf.dir/link.txt, 1545
tests/matrix/CMakeFiles/check_dynormft.dir/link.↩ txt, 1546
tests/matrix/CMakeFiles/check_eliminate_zeros.↩ dir/link.txt, 1546
tests/matrix/CMakeFiles/check_eps.dir/link.txt, 1547
tests/matrix/CMakeFiles/check_fro_inn.dir/link.txt, 1547
tests/matrix/CMakeFiles/check_gaxpy.dir/link.txt, 1548
tests/matrix/CMakeFiles/check_getelement.↩ dir/link.txt, 1548
tests/matrix/CMakeFiles/check_matrix_biorth.↩ dir/link.txt, 1549
tests/matrix/CMakeFiles/check_matrix_colaxpy.↩ dir/link.txt, 1549
tests/matrix/CMakeFiles/check_matrix_gbiorth.↩ dir/link.txt, 1550
tests/matrix/CMakeFiles/check_matrix_null.↩ dir/link.txt, 1550
tests/matrix/CMakeFiles/check_matrix_sum.↩ dir/link.txt, 1551
tests/matrix/CMakeFiles/check_mgs.dir/link.txt, 1551
tests/matrix/CMakeFiles/check_mgs_inplace.↩ dir/link.txt, 1552
tests/matrix/CMakeFiles/check_mgsadd.dir/link.txt, 1552
tests/matrix/CMakeFiles/check_mul.dir/link.txt, 1553
tests/matrix/CMakeFiles/check_mul_small.↩ dir/link.txt, 1553
tests/matrix/CMakeFiles/check_mul_tall.dir/link.txt, 1554
tests/matrix/CMakeFiles/check_mulnormf2.↩ dir/link.txt, 1554
tests/matrix/CMakeFiles/check_mvp.dir/link.txt, 1555
tests/matrix/CMakeFiles/check_norm.dir/link.txt, 1555
tests/matrix/CMakeFiles/check_one_value.↩ dir/link.txt, 1555
tests/matrix/CMakeFiles/check_orth.dir/link.txt, 1556
tests/matrix/CMakeFiles/check_proj_sym.↩ dir/link.txt, 1556
tests/matrix/CMakeFiles/check_rank.dir/link.txt, 1556
tests/matrix/CMakeFiles/check_rowsub.dir/link.txt, 1557
tests/matrix/CMakeFiles/check_scalem.dir/link.txt, 1557
tests/matrix/CMakeFiles/check_set_get_col_↩ row.dir/link.txt, 1558
tests/matrix/CMakeFiles/check_setelement.↩ dir/link.txt, 1558
tests/matrix/CMakeFiles/check_sub.dir/link.txt, 1559
tests/matrix/CMakeFiles/check_trace.dir/link.txt, 1559
tests/matrix/CMakeFiles/check_transpose.↩ dir/link.txt, 1559
tests/matrix/CMakeFiles/check_tridiag.dir/link.txt, 1560
tests/misc/CMakeFiles/check_freelist.dir/link.txt, 1560
tutorials/bt/CMakeFiles/tutorial_bt.dir/link.txt, 1561
tutorials/bt/CMakeFiles/tutorial_bt_gglrsrm.↩ dir/link.txt, 1561
tutorials/bt/CMakeFiles/tutorial_bt_gslrsrm.↩ dir/link.txt, 1562
tutorials/bt/CMakeFiles/tutorial_bt_lrsrm.dir/link.txt, 1562
tutorials/bt/CMakeFiles/tutorial_bt_lrsrm_somor.↩ dir/link.txt, 1563
tutorials/bt/CMakeFiles/tutorial_bt_lrsrm_somor↩ _so.dir/link.txt, 1563
tutorials/direct/CMakeFiles/tutorial_backslash.↩ dir/link.txt, 1564
tutorials/direct/CMakeFiles/tutorial_direct_↩ solvers.dir/link.txt, 1564
tutorials/direct/CMakeFiles/tutorial_qr.dir/link.txt, 1565
tutorials/direct/CMakeFiles/tutorial_sylvester.↩ dir/link.txt, 1565
tutorials/eig/CMakeFiles/tutorial_eig.dir/link.txt, 1565
tutorials/eig/CMakeFiles/tutorial_eigs.dir/link.txt, 1566

tutorials/eig/CMakeFiles/tutorial_gschur.dir/link.txt, 1566
tutorials/eig/CMakeFiles/tutorial_schur.dir/link.txt, 1567
tutorials/eig/CMakeFiles/tutorial_sign.dir/link.txt, 1567
tutorials/h2/CMakeFiles/tutorial_h2norm.dir/link.txt, 1568
tutorials/h2/CMakeFiles/tutorial_irka_biorth.↩ dir/link.txt, 1568
tutorials/h2/CMakeFiles/tutorial_tsia.dir/link.txt, 1569
tutorials/io/CMakeFiles/tutorial_vector_read.↩ dir/link.txt, 1569
tutorials/itsolver/CMakeFiles/tutorial_it_unsym.↩ dir/link.txt, 1570
tutorials/lrcf_adi/CMakeFiles/tutorial_care.↩ dir/link.txt, 1570
tutorials/lrcf_adi/CMakeFiles/tutorial_lradi.↩ dir/link.txt, 1570
tutorials/lrcf_adi/CMakeFiles/tutorial_lrnm-g.↩ dir/link.txt, 1571
tutorials/lrcf_adi/CMakeFiles/tutorial_lrnm.↩ dir/link.txt, 1571
tutorials/lrcf_adi/CMakeFiles/tutorial_lyap.↩ dir/link.txt, 1571
tutorials/matgen/CMakeFiles/tutorial_gen_fdm_↩ cd.dir/link.txt, 1572
tutorials/matgen/CMakeFiles/tutorial_gen_↩ laplace.dir/link.txt, 1572
tutorials/matrix/CMakeFiles/tutorial_matrix_↩ operations.dir/link.txt, 1573
tutorials/matrix/CMakeFiles/tutorial_norms.↩ dir/link.txt, 1573
tutorials/plot/CMakeFiles/tutorial_plot.dir/link.txt, 1574
tutorials/plot/CMakeFiles/tutorial_spy.dir/link.txt, 1574

rt1
res2_data_st, 942
rt2
res2_data_st, 942
rt3
res2_data_st, 942
rt4
res2_data_st, 942
rtb
res2_data_st, 942
rtc
res2_data_st, 942
rtz
res2_data_st, 942
rwork
blas_defs.h, 1003

S
blas_defs.h, 1003
SCALAR_ERROR_MACRO
check_linesearch.c, 1139
SCALE
blas_defs.h, 1003
SDIM
blas_defs.h, 1003
SDA
blas_defs.h, 1003
SELECT
blas_defs.h, 1003
SET_COLOR
lib/lrcf_adi/lrnm.c, 1604
SET_IF_IMPLEMENTED
equations.c, 1419
SET_LAST
read.c, 1812
vector_read.c, 1972
SET_SOLVERNAME
solvername.h, 1850
SEP
blas_defs.h, 1003
SIZE
DCPX_T.c, 1321
MESS_HAVE_PTRDIFF_T.c, 1645
MESS_HAVE_SIZE_T.c, 1646
SLU
csparse_solver, 811
csparse_solver_complex, 812
SORT
blas_defs.h, 1003
STARTTIME
irka_biorth.c, 1474
STRINGIFY_HELPER
CMakeCCompilerId.c, 1258
CMakeCXXCompilerId.cpp, 1260
STRINGIFY
CMakeCCompilerId.c, 1258
CMakeCXXCompilerId.cpp, 1260
SUCCESS
cuthill.c, 1314
SYLV_SPARSE_DENSE_GENERALIZED
lib/direct/singlesolver/sylvester_sparsedense.↩ c, 1874
SYLV_SPARSE_DENSE_HALF_GENERALIZED
lib/direct/singlesolver/sylvester_sparsedense.↩ c, 1874
SYLV_SPARSE_DENSE_STANDARD
lib/direct/singlesolver/sylvester_sparsedense.↩ c, 1874
save_as_xpm_file
x11init.c, 1980
scal.c, 1841
scalars.txt, 1842
scale
blas_defs.h, 1003
scale.c, 1842
__ONE, 1842
ScalePoints
x11init.c, 1977
schur.c, 1842

applyUe_left, 1843
applyUe_right, 1843
COMPLEX_ZGEES_IF_REAL, 1843
schur_to_evd.c, 1843
screen_num
__mess_plotter, 787
seed
blas_defs.h, 1003
select.c, 1844
select_solver
check_getLU.c, 1121
inverse.c, 1470
linsolve.c, 1576
linsolve_multi.c, 1582
linsolvem.c, 1588
linsolvem_multi.c, 1595
selected_solver
multidirect.c, 1742
selected_solver_mutex
multidirect.c, 1742
semidefinite
_glyap3_st, 796
sep
blas_defs.h, 1003
set_status_properties
mess_status.c, 1687
SetColor
x11init.c, 1980
setelement.c, 1844
__cmp_mess_int_t_complex, 1844
__cmp_mess_int_t_double, 1845
shellsort.c, 1845
__shellsort, 1845
shifts
__glyap_so1, 771
__glyap_so2, 773
shiftsolver
__glyap_so1, 771
__glyap_so2, 773
shiftsolvers
__glyap_dae2, 769
sig
__mess_plotter, 787
sig_ready
multilu, 928
sigma
VWjob, 947
sigmadiff
mess_h2_status_st, 856
sign.c, 1846
F77_GLOBAL, 1846
sizemess_int_t
mess_matrix_file_st, 870
so
CMakeCache.txt, 1256
so1.c, 1847
so2.c, 1847
so_type
mess_bt_options_st, 824
Software Testing, 712
sol
norminv2data, 935
solve
mess_direct_st, 839
mess_multidirect_st, 881
mess_precond_st, 899
solveh
mess_direct_st, 839
mess_multidirect_st, 882
solvem
mess_direct_st, 839
mess_multidirect_st, 882
solvemh
mess_direct_st, 839
mess_multidirect_st, 882
solvemt
mess_direct_st, 839
mess_multidirect_st, 882
solver
mklpardiso_solver, 924
solver.h, 1848
solvername.h, 1850
SET_SOLVERNAME, 1850
solverpool
multilu, 928
Solvers, 334
mess_solver_bicgstab, 334
mess_solver_cg, 335
mess_solver_gaussseidel, 336
mess_solver_gmres, 337
mess_solver_gmres_restart, 338
mess_solver_jacobi, 339
mess_solver_jacobi_convergence, 340
mess_solver_sor, 341
mess_solver_ssor, 342
Solvers for Sets of Linear Systems, 317
solvet
mess_direct_st, 840
mess_multidirect_st, 882
soname
lib/CMakeFiles/mess.dir/link.txt, 1503
sor.c, 1850
sort.c, 1852
__cmp_mess_int_t2_complex, 1853
__cmp_mess_int_t2_double, 1853
__cmp_mess_int_t_complex, 1853
__cmp_mess_int_t_double, 1854
source.txt, 1854–1856
Sparse Eigenvalue Solvers, 442
MESS_EIGEN_ARPACK_BOTH_ENDS, 445
MESS_EIGEN_ARPACK_DEFAULT, 444
MESS_EIGEN_ARPACK_LARGE_ALGEBRAIC, 445
MESS_EIGEN_ARPACK_LARGE_IMAGPART, 445

MESS_EIGEN_ARPACK_LARGE_MAGNITUDE, 444
MESS_EIGEN_ARPACK_LARGE_REALPART, 444
MESS_EIGEN_ARPACK_SMALL_ALGEBRAIC, 445
MESS_EIGEN_ARPACK_SMALL_IMAGPART, 445
MESS_EIGEN_ARPACK_SMALL_MAGNITUDE, 444
MESS_EIGEN_ARPACK_SMALL_REALPART, 445
mess_eigen_arnoldi, 445
mess_eigen_arnoldi_inv, 446
mess_eigen_arnoldi_template, 447
mess_eigen_arnoldi_template_nrm, 449
mess_eigen_arnoldig, 450
mess_eigen_arnoldig_inv, 451
mess_eigen_arpack, 453
mess_eigen_arpack_lanczos, 455
mess_eigen_arpack_lanczos_template, 456
mess_eigen_arpack_template, 458
mess_eigen_arpack_which_t, 444
mess_eigen_eiggs, 459
mess_eigen_eigs, 460
mess_eigen_lanczos_template_nrm, 462
sparse_complex_matrix_value
MESS_matrix, 868
sparse_matrix_value
MESS_matrix, 868
splsolve.c, 1857
spy.c, 1857
stable.c, 1858
stat
__presordata, 793
bicgstab_solver, 808
statB
mess_bt_status_st, 826
statC
mess_bt_status_st, 826
stats
mess_freelist_st, 853
status.c, 1858, 1859
statusopt.c, 1860
std.c, 1860
stein_res2.c, 1861
stepdebug
__mess_solver_options, 790
mess_h2_options_st, 855
stepdebug_data
mess_h2_options_st, 855
stepfunction
mess_options_st, 896
stepfunction_aux
mess_options_st, 896
stop_rel
mess_status_st, 907
stop_res2
mess_status_st, 907
stop_res2c
mess_status_st, 907
stop_user
__mess_lrcfadi_step, 781
mess_status_st, 907
Storage Formats, 726
store_type
mess_matrix_file_st, 870
mess_matrix_st, 872
multilu, 928
sub.c, 1861
subsasgn
MESS_options, 891
subsref
MESS_options, 891
superlu.c, 1861
superlu_refinement_bug.c, 1862
main, 1862
superlu_refinement_bug_43.c, 1862
main, 1862
superlu_refinement_bug_50.c, 1862
main, 1862
suppress_cppcheck.txt, 1862
svd.c, 1862
swap
blockpart.c, 1006
sylv_res2.c, 1863
sylvester_d_clear
lib/direct/singlesolver/sylvester_dense.c, 1864
sylvester_d_solvem_generalized
lib/direct/singlesolver/sylvester_dense.c, 1865
sylvester_d_solvem_standard
lib/direct/singlesolver/sylvester_dense.c, 1865
sylvester_d_solvemh_generalized
lib/direct/singlesolver/sylvester_dense.c, 1866
sylvester_d_solvemh_standard
lib/direct/singlesolver/sylvester_dense.c, 1867
sylvester_d_solvemt_generalized
lib/direct/singlesolver/sylvester_dense.c, 1869
sylvester_d_solvemt_standard
lib/direct/singlesolver/sylvester_dense.c, 1870
sylvester_dense.c, 1863, 1871
sylvester_sd_clear
lib/direct/singlesolver/sylvester_sparsedense.↩c, 1874
sylvester_sd_solvem
lib/direct/singlesolver/sylvester_sparsedense.↩c, 1875
sylvester_sd_solvemh
lib/direct/singlesolver/sylvester_sparsedense.↩c, 1876
sylvester_sd_solvemt
lib/direct/singlesolver/sylvester_sparsedense.↩c, 1878
sylvester_sparsedense.c, 1873, 1880
symbolic_iluk
iluk.c, 1463

symbols/__my_module__my_sub.c
__my_module__my_sub, 948
symbols/__my_module_MOD_my_sub.c
__my_module_MOD_my_sub, 948
symbols/__my_module_NMOD_my_sub.c
__my_module_NMOD_my_sub, 948
symbols/__mymodule__mysub.c
__mymodule__mysub, 949
symbols/__mymodule_MOD_mysub.c
__mymodule_MOD_mysub, 949
symbols/__mymodule_NMOD_mysub.c
__mymodule_NMOD_mysub, 950
symbols/MY_MODULE_mp_MY_SUB-UPPER.c
MY_MODULE_mp_MY_SUB, 1766
symbols/MY_SUB-UPPER.c
MY_SUB, 1768
symbols/MYMODULE_mp_MYSUB-UPPER.c
MYMODULE_mp_MYSUB, 1769
symbols/MYSUB-UPPER.c
MYSUB, 1771
symbols/my_module_MP_my_sub.c
my_module_MP_my_sub, 1766
symbols/my_module_mp_my_sub_.c
my_module_mp_my_sub_, 1766
symbols/my_module_my_sub_.c
my_module_my_sub_, 1767
symbols/my_moduleSmy_sub.c
$my_sub, 1767
symbols/my_sub.c
my_sub, 1768
symbols/my_sub_.c
my_sub_, 1768
symbols/my_sub__.c
my_sub__, 1769
symbols/mymodule_MP_mysub.c
mymodule_MP_mysub, 1770
symbols/mymodule_mp_mysub_.c
mymodule_mp_mysub_, 1770
symbols/mymodule_mysub_.c
mymodule_mysub_, 1770
symbols/mymoduleSmysub.c
$mysub, 1771
symbols/mysub.c
mysub, 1772
symbols/mysub_.c
mysub_, 1772
symmetry
mess_matrix_file_st, 871
mess_matrix_st, 873

T
blas_defs.h, 1003
t
res2_data_st, 942
t1
res2_data_st, 942
res2nm_data_st, 944
t2
res2_data_st, 942
res2nm_data_st, 944
t3
res2_data_st, 942
res2nm_data_st, 944
t4
res2_data_st, 942
res2nm_data_st, 944
TAU
blas_defs.h, 1004
TOKEN_LENGTH
read.c, 1812
vector_read.c, 1972
TRANSA
blas_defs.h, 1004
TRANS
blas_defs.h, 1004
TRY_MALLOC
cuthill.c, 1314
TRY_REALLOC
cuthill.c, 1314
tail
mess_threadpool_st, 912
TargetDirectories.txt, 1882
tau
lapackqr_solver, 821
tau_cpx
lapackqr_solver, 821
tb
res2nm_data_st, 944
tc
res2nm_data_st, 944
test.c, 1882
test_cases
check_dynorm_indefinite2.c, 1093
check_dynorm_indefinitef.c, 1095
test_det
check_det.c, 1084
test_sylvester_info
tests/direct/sylvester_dense.c, 1872
tests/direct/sylvester_sparsedense.c, 1881
TestRestrict.c, 1883
foo, 1883
main, 1883
testkron
check_kron.c, 1131
testmat
check_getLU.c, 1122
inverse.c, 1472
linsolve.c, 1578
linsolve_multi.c, 1584
linsolvem.c, 1590
tests/data/TripleChain/source.txt
alpha, 1857
beta, 1857
n1, 1857
v, 1857
tests/data/bips98_606/source.txt
A11, 1855

A12, 1855
A21, 1855
A22, 1855
B, 1855
C, 1855
E11, 1855
FROM, 1855
mmwrite, 1855
np, 1855
pp, 1855
tests/data/siso_ww_vref_6405/source.txt
A11, 1856
A12, 1856
A21, 1856
A22, 1856
B, 1856
C, 1856
E11, 1856
FROM, 1856
mmwrite, 1856
np, 1856
p, 1856
pp, 1856
tests/direct/CMakeFiles/check_backslash.dir/link.txt
D_FILE_OFFSET_BITS, 1504
rpath, 1504
tests/direct/CMakeFiles/check_det.dir/link.txt
D_FILE_OFFSET_BITS, 1504
rpath, 1504
tests/direct/CMakeFiles/check_dgeqp3.dir/link.txt
D_FILE_OFFSET_BITS, 1505
rpath, 1505
tests/direct/CMakeFiles/check_direct_care.dir/link.txt
D_FILE_OFFSET_BITS, 1505
rpath, 1505
tests/direct/CMakeFiles/check_getLU.dir/link.txt
D_FILE_OFFSET_BITS, 1506
rpath, 1506
tests/direct/CMakeFiles/check_hessenberg.dir/link.txt
D_FILE_OFFSET_BITS, 1506
rpath, 1506
tests/direct/CMakeFiles/check_lapackqr.dir/link.txt
D_FILE_OFFSET_BITS, 1507
rpath, 1507
tests/direct/CMakeFiles/check_pinv.dir/link.txt
D_FILE_OFFSET_BITS, 1507
rpath, 1507
tests/direct/CMakeFiles/check_trisolve.dir/link.txt
D_FILE_OFFSET_BITS, 1508
rpath, 1508
tests/direct/CMakeFiles/glyap_test.dir/link.txt
D_FILE_OFFSET_BITS, 1508
rpath, 1508
tests/direct/CMakeFiles/gstein_test.dir/link.txt
D_FILE_OFFSET_BITS, 1508
rpath, 1508
tests/direct/CMakeFiles/inverse.dir/link.txt
D_FILE_OFFSET_BITS, 1509
rpath, 1509
tests/direct/CMakeFiles/linsolve.dir/link.txt
D_FILE_OFFSET_BITS, 1509
rpath, 1509
tests/direct/CMakeFiles/linsolve_multi.dir/link.txt
D_FILE_OFFSET_BITS, 1509
rpath, 1509
tests/direct/CMakeFiles/linsolvem.dir/link.txt
D_FILE_OFFSET_BITS, 1510
rpath, 1510
tests/direct/CMakeFiles/linsolvem_multi.dir/link.txt
D_FILE_OFFSET_BITS, 1510
rpath, 1510
tests/direct/CMakeFiles/sylvester_dense.dir/link.txt
D_FILE_OFFSET_BITS, 1511
rpath, 1511
tests/direct/CMakeFiles/sylvester_sparsedense.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1511
rpath, 1511
tests/direct/sylvester_dense.c
main, 1872
test_sylvester_info, 1872
tests/direct/sylvester_sparsedense.c
main, 1881
test_sylvester_info, 1881
tests/eigen/CMakeFiles/check_arnoldi_template.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1512
rpath, 1512
tests/eigen/CMakeFiles/check_arnoldi_template_↩
nrm.dir/link.txt
D_FILE_OFFSET_BITS, 1512
rpath, 1512
tests/eigen/CMakeFiles/check_eig.dir/link.txt
D_FILE_OFFSET_BITS, 1513
rpath, 1513
tests/eigen/CMakeFiles/check_eig2.dir/link.txt
D_FILE_OFFSET_BITS, 1513
rpath, 1513
tests/eigen/CMakeFiles/check_eig_complex.dir/link.txt
D_FILE_OFFSET_BITS, 1513
rpath, 1513
tests/eigen/CMakeFiles/check_eig_hessenberg_↩
schur.dir/link.txt
D_FILE_OFFSET_BITS, 1514
rpath, 1514
tests/eigen/CMakeFiles/check_eig_hessenberg_↩
schur_largest_ev.dir/link.txt
D_FILE_OFFSET_BITS, 1514
rpath, 1514
tests/eigen/CMakeFiles/check_gsign.dir/link.txt
D_FILE_OFFSET_BITS, 1515
rpath, 1515
tests/eigen/CMakeFiles/check_schur.dir/link.txt
D_FILE_OFFSET_BITS, 1515
rpath, 1515
tests/eigen/CMakeFiles/check_sign.dir/link.txt

D_FILE_OFFSET_BITS, 1516
rpath, 1516
tests/eigen/CMakeFiles/check_svd.dir/link.txt
D_FILE_OFFSET_BITS, 1516
rpath, 1516
tests/eigen/CMakeFiles/check_svd_complex.dir/link.txt
D_FILE_OFFSET_BITS, 1516
rpath, 1516
tests/eigen/CMakeFiles/check_svd_econ.dir/link.txt
D_FILE_OFFSET_BITS, 1517
rpath, 1517
tests/eigen/CMakeFiles/check_svd_econ_complex.↩ dir/link.txt
D_FILE_OFFSET_BITS, 1517
rpath, 1517
tests/formats/CMakeFiles/check_cholmod.dir/link.txt
D_FILE_OFFSET_BITS, 1518
rpath, 1518
tests/formats/CMakeFiles/check_cimport.dir/link.txt
D_FILE_OFFSET_BITS, 1518
rpath, 1518
tests/formats/CMakeFiles/check_cimport_c2d.dir/link.txt
D_FILE_OFFSET_BITS, 1519
rpath, 1519
tests/formats/CMakeFiles/check_convert.dir/link.txt
D_FILE_OFFSET_BITS, 1519
rpath, 1519
tests/formats/CMakeFiles/check_mvpcall_matrix.↩ dir/link.txt
D_FILE_OFFSET_BITS, 1520
rpath, 1520
tests/formats/CMakeFiles/check_perm.dir/link.txt
D_FILE_OFFSET_BITS, 1520
rpath, 1520
tests/formats/CMakeFiles/check_real_imag_part.↩ dir/link.txt
D_FILE_OFFSET_BITS, 1521
rpath, 1521
tests/io/CMakeFiles/check_fputs_block.dir/link.txt
D_FILE_OFFSET_BITS, 1521
rpath, 1521
tests/io/CMakeFiles/check_matrixio.dir/link.txt
D_FILE_OFFSET_BITS, 1522
rpath, 1522
tests/io/CMakeFiles/check_vectorio.dir/link.txt
D_FILE_OFFSET_BITS, 1522
rpath, 1522
tests/itsolver/CMakeFiles/check_itsolver.dir/link.txt
D_FILE_OFFSET_BITS, 1523
rpath, 1523
tests/itsolver/CMakeFiles/check_precond.dir/link.txt
D_FILE_OFFSET_BITS, 1523
rpath, 1523
tests/lrcf_adi/CMakeFiles/check_ber_sign_fac.dir/link.↩ txt
D_FILE_OFFSET_BITS, 1524
rpath, 1524
tests/lrcf_adi/CMakeFiles/check_dae2_handles_adi.↩ dir/link.txt
D_FILE_OFFSET_BITS, 1524
rpath, 1524
tests/lrcf_adi/CMakeFiles/check_dae2_handles_para.↩ dir/link.txt
D_FILE_OFFSET_BITS, 1525
rpath, 1525
tests/lrcf_adi/CMakeFiles/check_dense_nm.dir/link.txt
D_FILE_OFFSET_BITS, 1525
rpath, 1525
tests/lrcf_adi/CMakeFiles/check_dense_nm_tridiag.↩ dir/link.txt
D_FILE_OFFSET_BITS, 1526
rpath, 1526
tests/lrcf_adi/CMakeFiles/check_linesearch.dir/link.txt
D_FILE_OFFSET_BITS, 1526
rpath, 1526
tests/lrcf_adi/CMakeFiles/check_lradi_dae1.dir/link.txt
D_FILE_OFFSET_BITS, 1527
rpath, 1527
tests/lrcf_adi/CMakeFiles/check_lradi_dae2.dir/link.txt
D_FILE_OFFSET_BITS, 1527
rpath, 1527
tests/lrcf_adi/CMakeFiles/check_lradi_glyap.dir/link.txt
D_FILE_OFFSET_BITS, 1528
rpath, 1528
tests/lrcf_adi/CMakeFiles/check_lradi_so.dir/link.txt
D_FILE_OFFSET_BITS, 1528
rpath, 1528
tests/lrcf_adi/CMakeFiles/check_lradi_unsym.dir/link.txt
D_FILE_OFFSET_BITS, 1529
rpath, 1529
tests/lrcf_adi/CMakeFiles/check_lrnm_dae1.dir/link.txt
D_FILE_OFFSET_BITS, 1529
rpath, 1529
tests/lrcf_adi/CMakeFiles/check_lrnm_dae2.dir/link.txt
D_FILE_OFFSET_BITS, 1530
rpath, 1530
tests/lrcf_adi/CMakeFiles/check_lrnm_g.dir/link.txt
D_FILE_OFFSET_BITS, 1530
rpath, 1530
tests/lrcf_adi/CMakeFiles/check_lrnm_so.dir/link.txt
D_FILE_OFFSET_BITS, 1531
rpath, 1531
tests/lrcf_adi/CMakeFiles/check_lrnm_unsym.dir/link.txt
D_FILE_OFFSET_BITS, 1531
rpath, 1531
tests/lrcf_adi/CMakeFiles/check_lyap_sign_fac.↩ dir/link.txt
D_FILE_OFFSET_BITS, 1532
rpath, 1532
tests/lrcf_adi/CMakeFiles/check_so1_handles.dir/link.txt
D_FILE_OFFSET_BITS, 1532
rpath, 1532
tests/lrcf_adi/CMakeFiles/check_so2_handles.dir/link.txt
D_FILE_OFFSET_BITS, 1533
rpath, 1533

tests/matrix/CMakeFiles/check_add.dir/link.txt
D_FILE_OFFSET_BITS, 1533
rpath, 1533
tests/matrix/CMakeFiles/check_addcol2p.dir/link.txt
D_FILE_OFFSET_BITS, 1534
rpath, 1534
tests/matrix/CMakeFiles/check_any.dir/link.txt
D_FILE_OFFSET_BITS, 1534
rpath, 1534
tests/matrix/CMakeFiles/check_bandwidth.dir/link.txt
D_FILE_OFFSET_BITS, 1535
rpath, 1535
tests/matrix/CMakeFiles/check_biorth.dir/link.txt
D_FILE_OFFSET_BITS, 1535
rpath, 1535
tests/matrix/CMakeFiles/check_cat.dir/link.txt
D_FILE_OFFSET_BITS, 1536
rpath, 1536
tests/matrix/CMakeFiles/check_colaxpy2.dir/link.txt
D_FILE_OFFSET_BITS, 1536
rpath, 1536
tests/matrix/CMakeFiles/check_coldot.dir/link.txt
D_FILE_OFFSET_BITS, 1537
rpath, 1537
tests/matrix/CMakeFiles/check_coldotE.dir/link.txt
D_FILE_OFFSET_BITS, 1538
rpath, 1538
tests/matrix/CMakeFiles/check_coldotc.dir/link.txt
D_FILE_OFFSET_BITS, 1537
rpath, 1537
tests/matrix/CMakeFiles/check_coldotcE.dir/link.txt
D_FILE_OFFSET_BITS, 1538
rpath, 1538
tests/matrix/CMakeFiles/check_colnorm.dir/link.txt
D_FILE_OFFSET_BITS, 1539
rpath, 1539
tests/matrix/CMakeFiles/check_colscale.dir/link.txt
D_FILE_OFFSET_BITS, 1539
rpath, 1539
tests/matrix/CMakeFiles/check_colsub.dir/link.txt
D_FILE_OFFSET_BITS, 1540
rpath, 1540
tests/matrix/CMakeFiles/check_colvecaxpy.dir/link.txt
D_FILE_OFFSET_BITS, 1540
rpath, 1540
tests/matrix/CMakeFiles/check_colvecdot.dir/link.txt
D_FILE_OFFSET_BITS, 1541
rpath, 1541
tests/matrix/CMakeFiles/check_colvecdotc.dir/link.txt
D_FILE_OFFSET_BITS, 1541
rpath, 1541
tests/matrix/CMakeFiles/check_condest.dir/link.txt
D_FILE_OFFSET_BITS, 1542
rpath, 1542
tests/matrix/CMakeFiles/check_decomp.dir/link.txt
D_FILE_OFFSET_BITS, 1542
rpath, 1542
tests/matrix/CMakeFiles/check_diag.dir/link.txt
D_FILE_OFFSET_BITS, 1543
rpath, 1543
tests/matrix/CMakeFiles/check_dynorm2.dir/link.txt
D_FILE_OFFSET_BITS, 1543
rpath, 1543
tests/matrix/CMakeFiles/check_dynorm2t.dir/link.txt
D_FILE_OFFSET_BITS, 1544
rpath, 1544
tests/matrix/CMakeFiles/check_dynorm_indefinite2.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1544
rpath, 1544
tests/matrix/CMakeFiles/check_dynorm_indefinitef.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1545
rpath, 1545
tests/matrix/CMakeFiles/check_dynormf.dir/link.txt
D_FILE_OFFSET_BITS, 1545
rpath, 1545
tests/matrix/CMakeFiles/check_dynormft.dir/link.txt
D_FILE_OFFSET_BITS, 1546
rpath, 1546
tests/matrix/CMakeFiles/check_eliminate_zeros.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1546
rpath, 1546
tests/matrix/CMakeFiles/check_eps.dir/link.txt
D_FILE_OFFSET_BITS, 1547
rpath, 1547
tests/matrix/CMakeFiles/check_fro_inn.dir/link.txt
D_FILE_OFFSET_BITS, 1547
rpath, 1547
tests/matrix/CMakeFiles/check_gaxpy.dir/link.txt
D_FILE_OFFSET_BITS, 1548
rpath, 1548
tests/matrix/CMakeFiles/check_getelement.dir/link.txt
D_FILE_OFFSET_BITS, 1548
rpath, 1548
tests/matrix/CMakeFiles/check_matrix_biorth.dir/link.txt
D_FILE_OFFSET_BITS, 1549
rpath, 1549
tests/matrix/CMakeFiles/check_matrix_colaxpy.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1549
rpath, 1549
tests/matrix/CMakeFiles/check_matrix_gbiorth.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1550
rpath, 1550
tests/matrix/CMakeFiles/check_matrix_null.dir/link.txt
D_FILE_OFFSET_BITS, 1550
rpath, 1550
tests/matrix/CMakeFiles/check_matrix_sum.dir/link.txt
D_FILE_OFFSET_BITS, 1551
rpath, 1551
tests/matrix/CMakeFiles/check_mgs.dir/link.txt
D_FILE_OFFSET_BITS, 1551
rpath, 1551

tests/matrix/CMakeFiles/check_mgs_inplace.dir/link.txt
D_FILE_OFFSET_BITS, 1552
rpath, 1552
tests/matrix/CMakeFiles/check_mgsadd.dir/link.txt
D_FILE_OFFSET_BITS, 1552
rpath, 1552
tests/matrix/CMakeFiles/check_mul.dir/link.txt
D_FILE_OFFSET_BITS, 1553
rpath, 1553
tests/matrix/CMakeFiles/check_mul_small.dir/link.txt
D_FILE_OFFSET_BITS, 1553
rpath, 1553
tests/matrix/CMakeFiles/check_mul_tall.dir/link.txt
D_FILE_OFFSET_BITS, 1554
rpath, 1554
tests/matrix/CMakeFiles/check_mulnormf2.dir/link.txt
D_FILE_OFFSET_BITS, 1554
rpath, 1554
tests/matrix/CMakeFiles/check_mvp.dir/link.txt
D_FILE_OFFSET_BITS, 1555
rpath, 1555
tests/matrix/CMakeFiles/check_norm.dir/link.txt
D_FILE_OFFSET_BITS, 1555
rpath, 1555
tests/matrix/CMakeFiles/check_one_value.dir/link.txt
D_FILE_OFFSET_BITS, 1555
rpath, 1555
tests/matrix/CMakeFiles/check_orth.dir/link.txt
D_FILE_OFFSET_BITS, 1556
rpath, 1556
tests/matrix/CMakeFiles/check_proj_sym.dir/link.txt
D_FILE_OFFSET_BITS, 1556
rpath, 1556
tests/matrix/CMakeFiles/check_rank.dir/link.txt
D_FILE_OFFSET_BITS, 1556
rpath, 1556
tests/matrix/CMakeFiles/check_rowsub.dir/link.txt
D_FILE_OFFSET_BITS, 1557
rpath, 1557
tests/matrix/CMakeFiles/check_scalem.dir/link.txt
D_FILE_OFFSET_BITS, 1557
rpath, 1557
tests/matrix/CMakeFiles/check_set_get_col_row.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1558
rpath, 1558
tests/matrix/CMakeFiles/check_setelement.dir/link.txt
D_FILE_OFFSET_BITS, 1558
rpath, 1558
tests/matrix/CMakeFiles/check_sub.dir/link.txt
D_FILE_OFFSET_BITS, 1559
rpath, 1559
tests/matrix/CMakeFiles/check_trace.dir/link.txt
D_FILE_OFFSET_BITS, 1559
rpath, 1559
tests/matrix/CMakeFiles/check_transpose.dir/link.txt
D_FILE_OFFSET_BITS, 1559
rpath, 1559
tests/matrix/CMakeFiles/check_tridiag.dir/link.txt
D_FILE_OFFSET_BITS, 1560
rpath, 1560
tests/misc/CMakeFiles/check_freelist.dir/link.txt
D_FILE_OFFSET_BITS, 1560
rpath, 1560
th_done_entry, 944
done, 945
id, 945
retdata, 945
th_ht_free
threadpool.c, 1884
th_ht_getname
threadpool.c, 1884
The ADI and the ADI-NM Algorithm, 483
mess_lradi, 484
mess_lrcfadi_adi, 486
mess_lrcfadi_galerkin, 488
mess_lrcfadi_getritzvalues, 489
mess_lrcfadi_nm, 490
mess_lrcfadi_nm_linesearch, 492
mess_lrcfadi_nm_linesearch_lambda, 493
mess_lrcfadi_nm_linesearch_poly, 494
mess_lrcfadi_parameter, 495
mess_lrcfadi_residual, 497
mess_lrnm, 499
mess_parameter, 500
thread
__mess_plotter, 787
threadpool.c, 1883
hash, 1884
th_ht_free, 1884
th_ht_getname, 1884
threadpool.h, 1884
threads
mess_threadpool_st, 912
time
mess_bt_status_st, 826
mess_h2_status_st, 856
Time Measurement, 696
mess_ctime, 697
mess_timer, 697
mess_timer_clear, 697
mess_timer_diff, 697
mess_timer_get, 697
mess_timer_getandreset, 698
mess_timer_init, 698
mess_wtime, 699
time.c, 1885
__USE_POSIX199309, 1885
MESS_ERROR, 1885
time_VW
mess_bt_status_st, 826
time_adi
mess_status_st, 908
time_all
mess_status_st, 908
time_lyap

mess_bt_status_st, 826
timer
mess_timer_st, 914
title
__mess_plotter, 787
__mess_plotterScript, 789
to_clear
mess_equation_st::AINV_st, 802
mess_equation_st::AX_st, 806
mess_equation_st::ApEINV_st, 803
mess_equation_st::ApEX_st, 805
mess_equation_st::EINV_st, 813
mess_equation_st::EX_st, 814
tol
__mess_solver_options, 790
mess_bt_options_st, 824
mess_eigen_arpack_options_t, 843
mess_h2_options_st, 855
trace.c, 1885
Trans
blas_defs.h, 1004
trans
blas_defs.h, 1004
transA
blas_defs.h, 1004
transB
blas_defs.h, 1004
transfer.c, 1886
pinv, 1886
transpose.c, 1886
trcpx
check_trace.c, 1239
Triangular Solvers, 265
mess_solver_lhsolve, 266
mess_solver_lhsolve_kernelcsr_complex, 267
mess_solver_lhsolvem, 268
mess_solver_lsolve, 269
mess_solver_lsolve_kernelcsr_complex, 269
mess_solver_lsolve_kernelcsr_real, 270
mess_solver_lsolve_kernelcsr_real_complex, 270
mess_solver_lsolvem, 271
mess_solver_ltsolve, 272
mess_solver_ltsolve_kernelcsr_complex, 273
mess_solver_ltsolve_kernelcsr_real, 274
mess_solver_ltsolve_kernelcsr_real_complex, 274
mess_solver_ltsolvem, 275
mess_solver_lusolvem_kernelcsr_complex, 276
mess_solver_uhsolve, 277
mess_solver_uhsolve_kernelcsr_complex, 278
mess_solver_uhsolvem, 279
mess_solver_usolve, 279
mess_solver_usolve_kernelcsr_complex, 280
mess_solver_usolve_kernelcsr_real, 281
mess_solver_usolve_kernelcsr_real_complex, 281
mess_solver_usolvem, 282
mess_solver_utsolve, 283
mess_solver_utsolve_kernelcsr_complex, 284
mess_solver_utsolve_kernelcsr_real, 285
mess_solver_utsolve_kernelcsr_real_complex, 285
mess_solver_utsolvem, 286
tridiag.c, 1887
trisolve.c, 1887
mess_solver_lcsolve_kernelcsr_complex, 1889
mess_solver_ucsolve_kernelcsr_complex, 1889
triul.c, 1890
trreal
check_trace.c, 1239
tsia.c, 1890
MAT_INIT, 1891
tsiag.c, 1891
MAT_INIT, 1891
tutorial_backslash.c, 1891
tutorial_bt.c, 1892
tutorial_bt_gglrsrm.c, 1896
tutorial_bt_gslrsrm.c, 1898
tutorial_bt_lrsrm.c, 1900
tutorial_bt_lrsrm_somor.c, 1902
tutorial_bt_lrsrm_somor_so.c, 1904
tutorial_care.c, 1906
tutorial_direct_solvers.c, 1908
tutorial_eig.c, 1912
tutorial_eigs.c, 1914
tutorial_gen_fdm_cd.c, 1915
tutorial_gen_laplace.c, 1916
tutorial_gschur.c, 1917
tutorial_h2norm.c, 1918
tutorial_irka_biorth.c, 1919
tutorial_it_unsym.c, 1921
tutorial_lradi.c, 1925
tutorial_lrnm-g.c, 1928
tutorial_lrnm.c, 1930
tutorial_lyap.c, 1933
tutorial_matrix_operations.c, 1935
tutorial_norms.c, 1938
tutorial_plot.c, 1939
tutorial_qr.c, 1941
tutorial_schur.c, 1943
tutorial_sign.c, 1944
tutorial_spy.c, 1945
tutorial_sylvester.c, 1947
tutorial_tsia.c, 1950
tutorial_vector_read.c, 1950
tutorials.dox, 1951
tutorials/bt/CMakeFiles/tutorial_bt.dir/link.txt
D_FILE_OFFSET_BITS, 1561
rpath, 1561
tutorials/bt/CMakeFiles/tutorial_bt_gglrsrm.dir/link.txt
D_FILE_OFFSET_BITS, 1561
rpath, 1561
tutorials/bt/CMakeFiles/tutorial_bt_gslrsrm.dir/link.txt
D_FILE_OFFSET_BITS, 1562
rpath, 1562
tutorials/bt/CMakeFiles/tutorial_bt_lrsrm.dir/link.txt
D_FILE_OFFSET_BITS, 1562
rpath, 1562

tutorials/bt/CMakeFiles/tutorial_bt_lrsrm_somor.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1563
rpath, 1563
tutorials/bt/CMakeFiles/tutorial_bt_lrsrm_somor_so.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1563
rpath, 1563
tutorials/direct/CMakeFiles/tutorial_backslash.dir/link.txt
D_FILE_OFFSET_BITS, 1564
rpath, 1564
tutorials/direct/CMakeFiles/tutorial_direct_solvers.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1564
rpath, 1564
tutorials/direct/CMakeFiles/tutorial_qr.dir/link.txt
D_FILE_OFFSET_BITS, 1565
rpath, 1565
tutorials/direct/CMakeFiles/tutorial_sylvester.dir/link.txt
D_FILE_OFFSET_BITS, 1565
rpath, 1565
tutorials/eig/CMakeFiles/tutorial_eig.dir/link.txt
D_FILE_OFFSET_BITS, 1565
rpath, 1565
tutorials/eig/CMakeFiles/tutorial_eigs.dir/link.txt
D_FILE_OFFSET_BITS, 1566
rpath, 1566
tutorials/eig/CMakeFiles/tutorial_gschur.dir/link.txt
D_FILE_OFFSET_BITS, 1566
rpath, 1566
tutorials/eig/CMakeFiles/tutorial_schur.dir/link.txt
D_FILE_OFFSET_BITS, 1567
rpath, 1567
tutorials/eig/CMakeFiles/tutorial_sign.dir/link.txt
D_FILE_OFFSET_BITS, 1567
rpath, 1567
tutorials/h2/CMakeFiles/tutorial_h2norm.dir/link.txt
D_FILE_OFFSET_BITS, 1568
rpath, 1568
tutorials/h2/CMakeFiles/tutorial_irka_biorth.dir/link.txt
D_FILE_OFFSET_BITS, 1568
rpath, 1568
tutorials/h2/CMakeFiles/tutorial_tsia.dir/link.txt
D_FILE_OFFSET_BITS, 1569
rpath, 1569
tutorials/io/CMakeFiles/tutorial_vector_read.dir/link.txt
D_FILE_OFFSET_BITS, 1569
rpath, 1569
tutorials/itsolver/CMakeFiles/tutorial_it_unsym.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1570
rpath, 1570
tutorials/lrcf_adi/CMakeFiles/tutorial_care.dir/link.txt
D_FILE_OFFSET_BITS, 1570
rpath, 1570
tutorials/lrcf_adi/CMakeFiles/tutorial_lradi.dir/link.txt
D_FILE_OFFSET_BITS, 1570
rpath, 1570
tutorials/lrcf_adi/CMakeFiles/tutorial_lrnm-g.dir/link.txt
D_FILE_OFFSET_BITS, 1571
rpath, 1571
tutorials/lrcf_adi/CMakeFiles/tutorial_lrnm.dir/link.txt
D_FILE_OFFSET_BITS, 1571
rpath, 1571
tutorials/lrcf_adi/CMakeFiles/tutorial_lyap.dir/link.txt
D_FILE_OFFSET_BITS, 1571
rpath, 1571
tutorials/matgen/CMakeFiles/tutorial_gen_fdm_cd.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1572
rpath, 1572
tutorials/matgen/CMakeFiles/tutorial_gen_laplace.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1572
rpath, 1572
tutorials/matrix/CMakeFiles/tutorial_matrix_operations.↩
dir/link.txt
D_FILE_OFFSET_BITS, 1573
rpath, 1573
tutorials/matrix/CMakeFiles/tutorial_norms.dir/link.txt
D_FILE_OFFSET_BITS, 1573
rpath, 1573
tutorials/plot/CMakeFiles/tutorial_plot.dir/link.txt
D_FILE_OFFSET_BITS, 1574
rpath, 1574
tutorials/plot/CMakeFiles/tutorial_spy.dir/link.txt
D_FILE_OFFSET_BITS, 1574
rpath, 1574
twoside_mgs.c, 1951
KAPPA, 1951
REFINE, 1951
type
__mess_plotExportData, 784
mess_dynsys_st, 842
mess_options_st, 896
mess_precond_st, 899
type_installed
MESS_datatype_t, 828
MESS_direct, 832
MESS_direct_cholpackage_t, 834
MESS_direct_lupackage_t, 837
MESS_equation, 847
MESS_equation_t, 852
MESS_matrix, 869
MESS_memusage_t, 875
MESS_multidirect, 879
MESS_multidirect_t, 884
MESS_operation_t, 887
MESS_options, 892
MESS_parameter_t, 898
MESS_residual_t, 901
MESS_status, 905
MESS_storage_t, 910
MESS_vector, 921
Types and Datastructures, 330
MESS_PRECOND_DIAG, 331

MESS_PRECOND_ILU0, 331
MESS_PRECOND_ILUK, 331
mess_precond, 331
mess_solver_options, 331
mess_solver_options_clear, 331
mess_solver_options_init, 332
mess_solver_options_print, 332
mess_solver_status, 331
mess_solver_status_clean, 332
mess_solver_status_clear, 332
mess_solver_status_init, 333
mess_solver_status_print, 333
tz
res2nm_data_st, 944

U
__iluk, 776
blas_defs.h, 1004
mvpdata, 930
mvpdatag, 931
newlu_data, 933
u
mvpdatag, 931
UMFPACK_CONTROL_LOG2_10
umfpack_control.c, 1954
UNFLIP
dfs.c, 1326
newlu.c, 1773
UNIT
blas_defs.h, 1004
UNSET_COLOR
lib/lrcf_adi/lrnm.c, 1604
UPLO
blas_defs.h, 1004
USE_FORTRAN_KERNELS
lu.c, 1608
uhsolvec
newlu.c, 1783
uind
jobdata_st, 818
multilu, 928
umfpack.c, 1951
umfpack_dl_defaults, 1952
umfpack_dl_free_numeric, 1952
umfpack_dl_free_symbolic, 1952
umfpack_dl_get_lunz, 1952
umfpack_dl_get_numeric, 1952
umfpack_dl_load_numeric, 1952
umfpack_dl_numeric, 1952
umfpack_dl_report_info, 1952
umfpack_dl_save_numeric, 1952
umfpack_dl_solve, 1952
umfpack_dl_symbolic, 1952
umfpack_zl_defaults, 1952
umfpack_zl_free_numeric, 1952
umfpack_zl_free_symbolic, 1952
umfpack_zl_get_lunz, 1952
umfpack_zl_get_numeric, 1952
umfpack_zl_load_numeric, 1952
umfpack_zl_numeric, 1952
umfpack_zl_report_info, 1952
umfpack_zl_save_numeric, 1952
umfpack_zl_solve, 1952
umfpack_zl_symbolic, 1952
umfpack_control.c, 1952
UMFPACK_CONTROL_LOG2_10, 1954
umfpack_control_clear, 1955
umfpack_control_det, 1956
umfpack_control_detc, 1956
umfpack_control_getL, 1956
umfpack_control_getpermp, 1957
umfpack_control_getpermq, 1957
umfpack_control_getscalerow, 1958
umfpack_control_getU, 1958
umfpack_control_inverse, 1959
umfpack_control_solve, 1960
umfpack_control_solveh, 1961
umfpack_control_solvem, 1962
umfpack_control_solvemh, 1963
umfpack_control_solvemt, 1964
umfpack_control_solver, 1955
umfpack_control_solvet, 1965
umfpack_dl_defaults, 1954
umfpack_dl_free_numeric, 1954
umfpack_dl_free_symbolic, 1954
umfpack_dl_get_determinant, 1954
umfpack_dl_get_lunz, 1954
umfpack_dl_get_numeric, 1954
umfpack_dl_load_numeric, 1954
umfpack_dl_numeric, 1954
umfpack_dl_report_info, 1954
umfpack_dl_save_numeric, 1954
umfpack_dl_solve, 1954
umfpack_dl_symbolic, 1954
umfpack_zl_defaults, 1954
umfpack_zl_free_numeric, 1955
umfpack_zl_free_symbolic, 1955
umfpack_zl_get_determinant, 1955
umfpack_zl_get_lunz, 1955
umfpack_zl_get_numeric, 1955
umfpack_zl_load_numeric, 1955
umfpack_zl_numeric, 1955
umfpack_zl_report_info, 1955
umfpack_zl_save_numeric, 1955
umfpack_zl_solve, 1955
umfpack_zl_symbolic, 1955
umfpack_control_clear
umfpack_control.c, 1955
umfpack_control_det
umfpack_control.c, 1956
umfpack_control_detc
umfpack_control.c, 1956
umfpack_control_getL
umfpack_control.c, 1956
umfpack_control_getpermp
umfpack_control.c, 1957
umfpack_control_getpermq

umfpack_control.c, 1957
umfpack_control_getscalerow
umfpack_control.c, 1958
umfpack_control_getU
umfpack_control.c, 1958
umfpack_control_inverse
umfpack_control.c, 1959
umfpack_control_solve
umfpack_control.c, 1960
umfpack_control_solveh
umfpack_control.c, 1961
umfpack_control_solvem
umfpack_control.c, 1962
umfpack_control_solvemh
umfpack_control.c, 1963
umfpack_control_solvemt
umfpack_control.c, 1964
umfpack_control_solver, 945
Control, 946
cpx, 946
dim, 946
Info, 946
Numeric, 946
umfpack_control.c, 1955
work, 946
umfpack_control_solvet
umfpack_control.c, 1965
umfpack_dl_defaults
banded.c, 980
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1954
umfpack_dl_free_numeric
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1954
umfpack_dl_free_symbolic
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1954
umfpack_dl_get_determinant
umfpack_control.c, 1954
umfpack_dl_get_lunz
umfpack.c, 1952
umfpack_control.c, 1954
umfpack_dl_get_numeric
umfpack.c, 1952
umfpack_control.c, 1954
umfpack_dl_load_numeric
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1954
umfpack_dl_numeric
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1954
umfpack_dl_report_info
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1954
umfpack_dl_report_numeric
multilu_umfpack.c, 1756
umfpack_dl_save_numeric
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1954
umfpack_dl_solve
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1954
umfpack_dl_symbolic
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1954
umfpack_zl_defaults
banded.c, 980
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1954
umfpack_zl_free_numeric
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1955
umfpack_zl_free_symbolic
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1955
umfpack_zl_get_determinant
umfpack_control.c, 1955
umfpack_zl_get_lunz
umfpack.c, 1952
umfpack_control.c, 1955
umfpack_zl_get_numeric
umfpack.c, 1952
umfpack_control.c, 1955
umfpack_zl_load_numeric
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1955
umfpack_zl_numeric
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1955
umfpack_zl_report_info
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1955
umfpack_zl_report_numeric
multilu_umfpack.c, 1756
umfpack_zl_save_numeric
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1955
umfpack_zl_solve
multilu_umfpack.c, 1756
umfpack.c, 1952

umfpack_control.c, 1955
umfpack_zl_symbolic
multilu_umfpack.c, 1756
umfpack.c, 1952
umfpack_control.c, 1955
unnz
multilu, 928
unstable
mess_status_st, 908
up
jobdata_st, 818
multilu, 928
uplo
blas_defs.h, 1004
upperbound
__glyap_so1, 771
__glyap_so2, 773
uptr
__ilu0_data, 775
usolve
newlu.c, 1783
usolvec
newlu.c, 1783
usolverc
newlu.c, 1783
ustore_type
multilu, 928
Utilde
mvpdatag, 931
utsolve
newlu.c, 1784
utsolvec
newlu.c, 1784
utsolverc
newlu.c, 1784
uvalues
jobdata_st, 818
multilu, 928
uvalues_cpx
multilu, 928
uvalues_next
jobdata_st, 818

V
mess_krylov_arnoldi_st, 858
VWjob, 947
v
mvpdatag, 931
tests/data/TripleChain/source.txt, 1857
v_map_cpx
check_set_get_col_row.c, 1221
v_map_real
check_set_get_col_row.c, 1221
VAL
check_one_value.c, 1198
VECTOR_AXPY
lrcf_adi/norms/res2.c, 1821
VECTOR_SCALE
lrcf_adi/norms/res2.c, 1822
VSL
blas_defs.h, 1004
VSR
blas_defs.h, 1004
VW_worker
H2 Model Order Reduction, 422
VWjob, 946
Asolver, 947
b, 947
bc, 947
c, 947
cc, 947
datatype, 947
i, 947
msolverid, 947
n, 947
sigma, 947
V, 947
W, 947
val
_scalepoints, 797
cholesky_solver, 810
hessenberg_solver, 815
lapack_solver, 820
lapackqr_solver, 821
val_cpx
cholesky_solver, 810
hessenberg_solver, 815
lapack_solver, 820
lapackqr_solver, 821
values
blas_defs.h, 1004
CCSRFmt, 809
mess_matrix_st, 873
mess_vector_st, 922
values_cpx
CCSRFmt, 809
mess_matrix_st, 873
mess_vector_st, 922
Various Macros, 700
MESS_CALL_IF_EXIST, 703
MESS_CLEAR_DIRECTS, 706
MESS_CLEAR_EQUATIONS, 706
MESS_CLEAR_MATRICES, 706
MESS_CLEAR_POINTERS, 707
MESS_CLEAR_VECTORS, 707
MESS_INIT_DIRECTS, 708
MESS_INIT_EQUATIONS, 708
MESS_INIT_MATRICES, 708
MESS_INIT_VECTORS, 709
MESS_MAX, 709
MESS_MIN, 709
MESS_RESET_MATRICES, 709
mess_check_complex, 703
mess_check_coord, 703
mess_check_csc, 703
mess_check_csr, 703
mess_check_datatype, 703

mess_check_dense, 703
mess_check_nocoord, 703
mess_check_nonnegative, 704
mess_check_notnullpointer, 704
mess_check_nullpointer, 704
mess_check_operation_type, 704
mess_check_positive, 704
mess_check_real, 704
mess_check_real_or_complex, 704
mess_check_same_cols, 704
mess_check_same_colsdim, 704
mess_check_same_colsrows, 705
mess_check_same_datatype, 705
mess_check_same_dim, 705
mess_check_same_rows, 705
mess_check_same_rowsdim, 705
mess_check_same_size, 705
mess_check_same_storetype, 705
mess_check_sparse, 705
mess_check_square, 705
mess_check_storage_type, 706
mess_check_true, 706
mess_free, 707
mess_try_alloc, 709
mess_try_alloc2, 710
mess_try_alloc_omp, 710
mess_try_calloc, 710
mess_try_calloc2, 710
mess_try_realloc, 711
mess_try_realloc2, 711
Various Other Functions related to Direct Solvers., 314
mess_decomp_lureuse_kernelcsr, 315
mess_decomp_lureuse_kernelcsr_complex, 315
mess_decomp_lureuse_kernelcsr_flopcount, 315
mess_direct_levelset, 315
mess_direct_splsolve, 315
mess_direct_splsolvec, 316
Various Other Operations on Matrices, 163
MESS_REORDER_AMD, 164
MESS_REORDER_COLAMD, 164
MESS_REORDER_NONE, 164
MESS_REORDER_RCM, 164
mess_matrix_colperm, 164
mess_matrix_colpermcopy, 165
mess_matrix_decomp, 166
mess_matrix_equal, 166
mess_matrix_equal_verbose, 167
mess_matrix_equalpattern, 167
mess_matrix_joinpatterns, 168
mess_matrix_memsize, 169
mess_matrix_memsize_nnz, 169
mess_matrix_perm, 170
mess_matrix_permcopy, 170
mess_matrix_reorder, 171
mess_matrix_reorder_amd, 172
mess_matrix_reorder_colamd, 173
mess_matrix_reorder_rcm, 173
mess_matrix_spy, 174
mess_reorder_t, 164
mess_vector_frommatrix, 174
mess_vector_tomatrix, 175
Various Other Operations with Vectors, 206
mess_vector_any, 208
mess_vector_cat, 209
mess_vector_convert_if_real, 210
mess_vector_filter, 210
mess_vector_filter_stable, 211
mess_vector_get, 211
mess_vector_iperm, 212
mess_vector_iperm_combine, 212
mess_vector_iperm_inplace, 213
mess_vector_iperm_split, 214
mess_vector_kron, 214
mess_vector_lift, 215
mess_vector_linspace, 216
mess_vector_logspace10, 217
mess_vector_logspace2, 217
mess_vector_logspacee, 218
mess_vector_map, 219
mess_vector_map_abs, 220
mess_vector_map_acos, 220
mess_vector_map_acosh, 220
mess_vector_map_arg, 220
mess_vector_map_asin, 220
mess_vector_map_asinh, 220
mess_vector_map_atan, 220
mess_vector_map_atanh, 220
mess_vector_map_ceil, 220
mess_vector_map_conj, 220
mess_vector_map_cos, 220
mess_vector_map_cosh, 220
mess_vector_map_exp, 220
mess_vector_map_expm1, 220
mess_vector_map_floor, 220
mess_vector_map_isfinite, 220
mess_vector_map_isinf, 220
mess_vector_map_isnan, 220
mess_vector_map_log, 220
mess_vector_map_not, 221
mess_vector_map_round, 221
mess_vector_map_sin, 221
mess_vector_map_sinh, 221
mess_vector_map_sqrt, 221
mess_vector_map_tan, 221
mess_vector_map_tanh, 221
mess_vector_memsize, 221
mess_vector_ones, 221
mess_vector_perm, 222
mess_vector_perm_combine, 222
mess_vector_perm_inplace, 223
mess_vector_perm_split, 224
mess_vector_print, 225
mess_vector_printinfo, 225
mess_vector_printshort, 226
mess_vector_rand, 226
mess_vector_rand_init, 227

mess_vector_read, 227
mess_vector_resize, 228
mess_vector_sort, 229
mess_vector_sort_imagpart, 229
mess_vector_sort_realpart, 230
mess_vector_split, 230
mess_vector_valuecomplex, 231
mess_vector_valuereal, 231
mess_vector_write, 232
mess_vector_zeros, 233
Vector Operations, 187
vector.c, 1966
ConvStringtoC, 1967
NO_IMPORT_ARRAY, 1967
NPY_ARRAY_OWNDATA, 1967
NPY_NO_DEPRECATED_API, 1967
PY_ARRAY_UNIQUE_SYMBOL, 1967
PyArray_CALL_DATA, 1967
PyArray_CALL_ISSIGNED, 1967
PyArray_CALL_ISUNSIGNED, 1967
PyArray_ENABLEFLAGS, 1967
vector.h, 1967
vector/cat.c
__ONE, 1020
vector/map.c
MESS_MAP_STATIC_FUNC2, 1616
MESS_MAP_STATIC_FUNC3, 1617
MESS_MAP_STATIC_FUNC, 1616
map_abs_cpx, 1617
map_abs_real, 1617
map_acos_cpx, 1617
map_acos_real, 1617
map_acosh_cpx, 1617
map_acosh_real, 1617
map_arg_cpx, 1617
map_arg_real, 1617
map_asin_cpx, 1617
map_asin_real, 1617
map_asinh_cpx, 1617
map_asinh_real, 1617
map_atan_cpx, 1617
map_atan_real, 1618
map_atanh_cpx, 1618
map_atanh_real, 1618
map_ceil_cpx, 1618
map_ceil_real, 1618
map_conj_cpx, 1618
map_conj_real, 1618
map_cos_cpx, 1618
map_cos_real, 1618
map_cosh_cpx, 1618
map_cosh_real, 1618
map_exp_cpx, 1618
map_exp_real, 1618
map_expm1_cpx, 1618
map_expm1_real, 1618
map_floor_cpx, 1618
map_floor_real, 1618
map_isfinite_cpx, 1618
map_isfinite_real, 1618
map_isinf_cpx, 1618
map_isinf_real, 1618
map_isnan_cpx, 1618
map_isnan_real, 1618
map_log_cpx, 1618
map_log_real, 1618
map_not_cpx, 1619
map_not_real, 1619
map_round_cpx, 1619
map_round_real, 1619
map_sin_cpx, 1619
map_sin_real, 1619
map_sinh_cpx, 1619
map_sinh_real, 1619
map_sqrt_cpx, 1619
map_sqrt_real, 1619
map_tan_cpx, 1619
map_tan_real, 1619
map_tanh_cpx, 1619
map_tanh_real, 1619
vector/norm.c
__ONE, 1786
vector/perm.c
__compare_complex, 1797
__compare_imagpart, 1797
__compare_real, 1798
__compare_realpart, 1798
vector/rand.c
__drand, 1809
vector_print.c, 1971
vector_read.c, 1971
__mm_read_vector_info, 1972
_lowercase, 1973
LINE_LENGTH, 1972
MM_BANNER, 1972
MM_STR_CPX, 1972
MM_STR_INT, 1972
MM_STR_REAL, 1972
MM_STR_VECTOR, 1972
mess_vector_mwrite, 1973
SET_LAST, 1972
TOKEN_LENGTH, 1972
vector_to_c
Interface to Python, 653
vector_to_python
Interface to Python, 654
vectors
mess_freelist_st, 853
VerifyC/CMakeFiles/FortranCInterface/symbols/__my↩_module__my_sub.c
__my_module__my_sub, 948
VerifyC/CMakeFiles/FortranCInterface/symbols/__my↩_module_MOD_my_sub.c
__my_module_MOD_my_sub, 948
VerifyC/CMakeFiles/FortranCInterface/symbols/__my↩_module_NMOD_my_sub.c

__my_module_NMOD_my_sub, 949
VerifyC/CMakeFiles/FortranCInterface/symbols/__↩ mymodule__mysub.c
__mymodule__mysub, 949
VerifyC/CMakeFiles/FortranCInterface/symbols/__↩ mymodule_MOD_mysub.c
__mymodule_MOD_mysub, 949
VerifyC/CMakeFiles/FortranCInterface/symbols/__↩ mymodule_NMOD_mysub.c
__mymodule_NMOD_mysub, 950
VerifyC/CMakeFiles/FortranCInterface/symbols/MY_↩ MODULE_mp_MY_SUB-UPPER.c
MY_MODULE_mp_MY_SUB, 1766
VerifyC/CMakeFiles/FortranCInterface/symbols/MY_↩ SUB-UPPER.c
MY_SUB, 1768
VerifyC/CMakeFiles/FortranCInterface/symbols/MYM↩ ODULE_mp_MYSUB-UPPER.c
MYMODULE_mp_MYSUB, 1769
VerifyC/CMakeFiles/FortranCInterface/symbols/MYS↩ UB-UPPER.c
MYSUB, 1771
VerifyC/CMakeFiles/FortranCInterface/symbols/my_↩ module_MP_my_sub.c
my_module_MP_my_sub, 1766
VerifyC/CMakeFiles/FortranCInterface/symbols/my_↩ module_mp_my_sub_.c
my_module_mp_my_sub_, 1767
VerifyC/CMakeFiles/FortranCInterface/symbols/my_↩ module_my_sub_.c
my_module_my_sub_, 1767
VerifyC/CMakeFiles/FortranCInterface/symbols/my_↩ moduleSmy_sub.c
$my_sub, 1767
VerifyC/CMakeFiles/FortranCInterface/symbols/my_↩ sub.c
my_sub, 1768
VerifyC/CMakeFiles/FortranCInterface/symbols/my_↩ sub_.c
my_sub_, 1769
VerifyC/CMakeFiles/FortranCInterface/symbols/my_↩ sub__.c
my_sub__, 1769
VerifyC/CMakeFiles/FortranCInterface/symbols/mymodule↩ _MP_mysub.c
mymodule_MP_mysub, 1770
VerifyC/CMakeFiles/FortranCInterface/symbols/mymodule↩ _mp_mysub_.c
mymodule_mp_mysub_, 1770
VerifyC/CMakeFiles/FortranCInterface/symbols/mymodule↩ _mysub_.c
mymodule_mysub_, 1771
VerifyC/CMakeFiles/FortranCInterface/symbols/mymodule↩ Smysub.c
$mysub, 1771
VerifyC/CMakeFiles/FortranCInterface/symbols/mysub.↩ c
mysub, 1772
VerifyC/CMakeFiles/FortranCInterface/symbols/mysub↩ _.c
mysub_, 1772
VerifyFortran
VerifyFortran.h, 1974
VerifyFortran.h, 1974
FortranCInterface_GLOBAL_, 1974
FortranCInterface_GLOBAL, 1974
FortranCInterface_MODULE_, 1974
FortranCInterface_MODULE, 1974
VerifyFortran, 1974
version
mess_matrix_file_st, 871
vj
mvpdatag, 931
VL
blas_defs.h, 1004
VR
blas_defs.h, 1004
VS
blas_defs.h, 1004
VT
blas_defs.h, 1004
vwork
jobdata_st, 819

W
blas_defs.h, 1004
mess_options_st, 896
mvpdata, 930
mvpdatag, 931
VWjob, 947
WORK_BACKGROUND
multilu_adavanced.c, 1744
WORK
blas_defs.h, 1005
wachspress.c, 1974
dn, 1975
ellipK, 1975
F, 1975
min, 1975
warn_handle
interface_matlab.c, 1467
which
mess_eigen_arpack_options_t, 843
WI
blas_defs.h, 1004
width
__mess_plotter, 787
win
__mess_plotter, 787
work
blas_defs.h, 1005
mklpardiso_solver, 924
umfpack_control_solver, 946
workcolptr
jobdata_st, 819
multilu, 928
worker

mess_threadpooljob_st, 913
workrowptr
jobdata_st, 819
multilu, 928
workvalues
jobdata_st, 819
WR
blas_defs.h, 1005
write.c, 1976
CHECK_NULL, 1976

X
blas_defs.h, 1005
x
__mess_plotExportData, 784
__mess_plotdata, 782
blas_defs.h, 1005
x1
norm2data, 934
norminv2data, 935
rc2data, 940
X11 Function Plotter, 657
mess_plot, 658
mess_plot_addData, 658
mess_plot_clearData, 659
mess_plot_close, 659
mess_plot_create, 660
mess_plot_save, 661
mess_plot_setColor, 661
mess_plot_setLabel, 661
mess_plot_update, 662
mess_plotdata, 658
x11init.c, 1976
__plotter, 1977
drawaxis, 1978
map_to_win, 1979
plot, 1979
save_as_xpm_file, 1980
ScalePoints, 1977
SetColor, 1980
x2
__stable_lyap, 794
rc2data, 940
xLabel
__mess_plotterScript, 789
xerbla.c, 1981
xerbla_, 1981
xerbla_
xerbla.c, 1981
xscale
__mess_plotter, 787
__mess_plotterScript, 789

y
__mess_plotExportData, 784
__mess_plotdata, 783
blas_defs.h, 1005
mvpdatag, 932
yLabel
__mess_plotterScript, 789
yscale
__mess_plotter, 787
__mess_plotterScript, 789

Z
__mess_lrcfadi_step, 781
_sylv_solver_sd_st, 801
blas_defs.h, 1005
rc2data, 940
res2_data_st, 942
res2nm_data_st, 944
z
mvpdatag, 932
z2
__stable_lyap, 794
zero_func
fdm_matrix.c, 1426
Zold
rc2data, 940
ZX
blas_defs.h, 1005
ZY
blas_defs.h, 1005