MAX-PLANCK-GESELLSCHAFT

Max Planck Institute Magdeburg
Preprints

Peter Benner Vasile Sima Matthias Voigt

FORTRAN 77 Subroutines for the Solution of Skew-Hamiltonian/Hamiltonian Eigenproblems – Part II: Implementation and Numerical Results

MAX–PLANCK–INSTITUT
FÜR DYNAMIK KOMPLEXER
TECHNISCHER SYSTEME
MAGDEBURG

MPIMD/13-11 July 29, 2013

Abstract

Skew-Hamiltonian/Hamiltonian matrix pencils $\lambda\mathcal{S} - \mathcal{H}$ appear in many applications, including linear quadratic optimal control problems, $\mathcal{H}_\infty$-optimization, certain multi-body systems and many other areas in applied mathematics, physics, and chemistry. In these applications it is necessary to compute certain eigenvalues and/or corresponding deflating subspaces of these matrix pencils. Recently developed methods exploit and preserve the skew-Hamiltonian/Hamiltonian structure and hence increase reliability, accuracy and performance of the computations. In this paper we describe the implementation of the algorithms in the style of subroutine included in the Subroutine Library in Control Theory (SLICOT) described in Part I of this work [7] and address various details. Furthermore, we perform numerical tests using real-world examples to demonstrate the superiority of the new algorithms compared to standard methods.

Impressum:

Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg

Publisher:
Max Planck Institute for
Dynamics of Complex Technical Systems

Address:
Max Planck Institute for
Dynamics of Complex Technical Systems
Sandtorstr. 1
39106 Magdeburg

www.mpi-magdeburg.mpg.de/preprints

1 Introduction

In this paper we discuss algorithms for the solution of generalized eigenvalue problems with skew-Hamiltonian/Hamiltonian structure. Usually, we are interested in a certain subset of the spectrum, e.g., the eigenvalues with negative real part, or the purely imaginary eigenvalues; or corresponding deflating subspaces. In Part I of this paper we summarize structure-preserving algorithms for the computation of the desired spectral information. All definitions, theoretical considerations and applications can be found there. In this part we address certain implementation details and give a detailed documentation of the subroutines. Finally, we perform a series of numerical tests to illustrate the efficiency and robustness of the algorithms and their implementation.

2 Implementation Details

In this section we focus on the description of our implementation of the algorithms presented in Part I of this paper. We describe inputs and outputs of each individual main subroutine and certain implementation details.

2.1 General Remarks

Our subroutines are written by employing the rigorous implementation and documentation standards of the Subroutine Library in Control Theory (SLICOT[1]), see [2, 1]. The parameters of each SLICOT routine can be classified as follows:

- mode parameters,
- input/output parameters,
- tolerances,
- workspace,
- error/warning indicator.

Mode parameters specify, e.g., what outputs we want to compute or what method we want to use for computations. Input/output parameters are usually the dimension of the involved matrices and the matrices themselves with their leading dimensions. In the sequel, `LDx` usually denotes the leading dimension of the array "`x`". The workspace consists of memory for different data types. Here, integer workspace is denoted by `IWORK` with size `LIWORK`, similarly for logical (boolean) workspace `BWORK` of size `LBWORK`, double precision workspace `DWORK` of size `LDWORK`, and double complex workspace `ZWORK` of size `LZWORK`. The error indicator `INFO` tells the user if an illegal value was used as input (`INFO` takes negative values) or if there occurred an error during program execution (`INFO` takes positive values). A warning indicator `IWARN` informs the user about possibly unreliable or inaccurate results or additional information about the

[1] http://www.slicot.org

Hamiltonian	skew-Hamiltonian
$\mathtt{DE} = \begin{bmatrix} \mathtt{e}_{11} & \mathtt{d}_{11} & \mathtt{d}_{12} & \mathtt{d}_{13} & \cdots \\ \mathtt{e}_{21} & \mathtt{e}_{22} & \mathtt{d}_{22} & \mathtt{d}_{23} & \cdots \\ \mathtt{e}_{31} & \mathtt{e}_{32} & \mathtt{e}_{33} & \mathtt{d}_{33} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$	$\mathtt{DE} = \begin{bmatrix} \star & \star & \mathtt{d}_{12} & \mathtt{d}_{13} & \cdots \\ \mathtt{e}_{21} & \star & \star & \mathtt{d}_{23} & \cdots \\ \mathtt{e}_{31} & \mathtt{e}_{32} & \star & \star & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$

Figure 1: Storage layout for the (skew-)symmetric submatrices D and E

results. We omit these parameters in the following interface description since they occur in every routine in a similar way. We refer to the comments within every individual subroutine for more details.

2.2 Storage Layout

Since Hamiltonian and skew-Hamiltonian matrices have certain block structures we use a packed storage layout proposed in [4] to avoid saving redundant data. More specifically, if a real $2n \times 2n$ Hamiltonian matrix $\mathcal{H} = \begin{bmatrix} A & D \\ E & -A^T \end{bmatrix}$ is given, we save the submatrix A in a conventional $n \times n$ array `A`, the symmetric submatrices D and E are stored in an $n \times (n+1)$ array `DE` such that the upper triangular part of D is stored in `DE(1:n,2:n+1)` and the lower triangular part of E is stored in `DE(1:n,1:n)`. The skew-symmetric parts of a skew-Hamiltonian matrix are similarly stored with the notable difference that the parts containing the diagonal and the first superdiagonal of the array `DE` are not referenced. See also Figure 1 for a visualization. Similarly, as every orthogonal or unitary symplectic $2n \times 2n$ matrix has the block structure $\mathcal{U} = \begin{bmatrix} U_1 & U_2 \\ -U_2 & U_1 \end{bmatrix}$, we only store the matrix U_1 is an $n \times n$ array `U1` and the matrix U_2 is an $n \times n$ array `U2`.

A similar storage format is also applied to complex skew-Hamiltonian or Hamiltonian matrices. In contrast to the real case, for skew-Hamiltonian matrices, also the parts containing the diagonal and the first superdiagonal of the array `DE` are referenced.

2.3 Panel Blocking for Larger Problems

The problems considered here are usually based on applying sequences of Givens rotations. When updating the involved matrices we successively have to transform the corresponding rows and columns in each step. However, for larger matrices this kind of transformations can become very inefficient due to FORTRAN's memory and cache management. FORTRAN uses a column-major memory layout, i.e., elements of a column are internally stored one after the other. On the other hand, the distance in

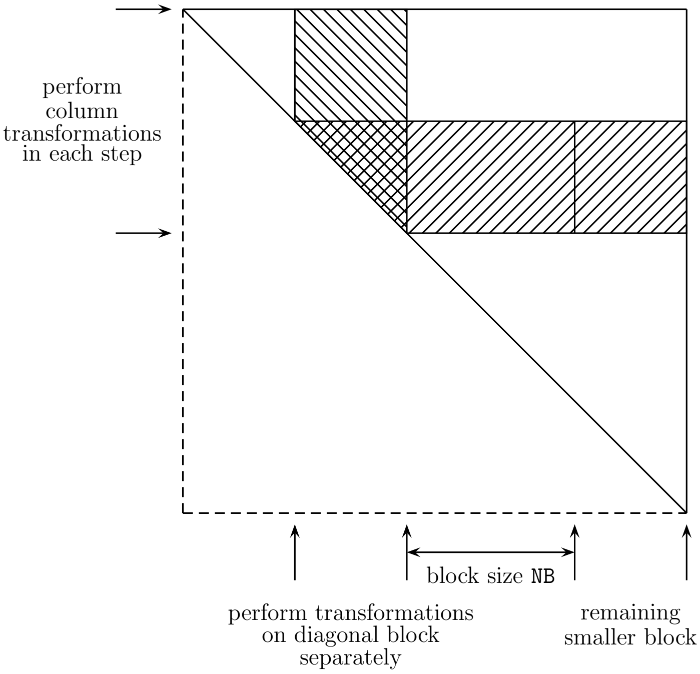

perform column transformations in each step
block size NB
perform transformations on diagonal block separately
remaining smaller block

Figure 2: Panel blocking technique for an upper triangular matrix

the internal memory between two successive elements in a row is exactly the leading dimension of that array. Therefore, rows can only be put into the cache memory by caching also the remaining parts of the columns that contain elements of the rows under consideration. For larger arrays, this easily leads to chunk sizes that do not fit into cache memory anymore. Therefore, our idea is to store the information of a certain number of Givens rotations and apply the row transformations only on panels of block size NB which fit into the cache.

An example for such a panel update is depicted in Figure 2. It illustrates the blocking technique for an update of a triangular matrix. Updates on columns are always directly applied after the generation of the Givens rotation, whereas rows are split into certain subpanels of maximum block size NB. Note that updates on the diagonal block are done separately as then the remaining parts of the rows have equal size and can therefore be easily decomposed into subblocks. We note that each part of the code has to be blocked in a different way. This is due to different matrix structures or dependencies of the updates and generation of the next Givens rotations. Therefore sometimes parts of rows have to be updated in each step. We have blocked versions for some of our codes and we will compare them with the unblocked versions in Section 4.

3 Interface Description

This section gives a brief overview over the main individual routines and their interfaces. For brevity we only describe the most important parameters and omit, e.g., leading dimensions and error or warning indicators. If we say that an array contains a matrix we mean that this matrix is stored in the leading part of this array. This is important because sometimes arrays can be larger than the matrices themselves. Table 1 gives an overview about the algorithms presented in [7] and the corresponding FORTRAN routines described in this paper. The structure of the calling graph for the unfactored case is similar and depicted in Figure 4.

Table 1: Overview of algorithms and FORTRAN routines

Algorithm # in [7]	FORTRAN routine
1	ZGHFDF
2	DGHFST
3	ZGHFXC
4	ZGHUDF
5	DGHUST
6	ZGHUXC
7	DGHFDF
8	DGHURV
9	DGHFYR
10	DGHFXC
11	DGHUDF
12	DGHUTR
13	DGHUYR
14	DGHUXC

3.1 The Complex Case

In this subsection we describe the interfaces of the subroutines needed for computing the eigenvalues and stable deflating subspaces of a complex skew-Hamiltonian/Hamiltonian matrix pencil. We begin with the factored case. In Figure 3, the corresponding calling graph with all needed subroutines is depicted. For brevity we only show the needed driver routines and elementary subroutines that deal with skew-Hamiltonian/Hamiltonian pencils of elementary size, i.e., up to size 4×4. Further called SLICOT subroutines are omitted.

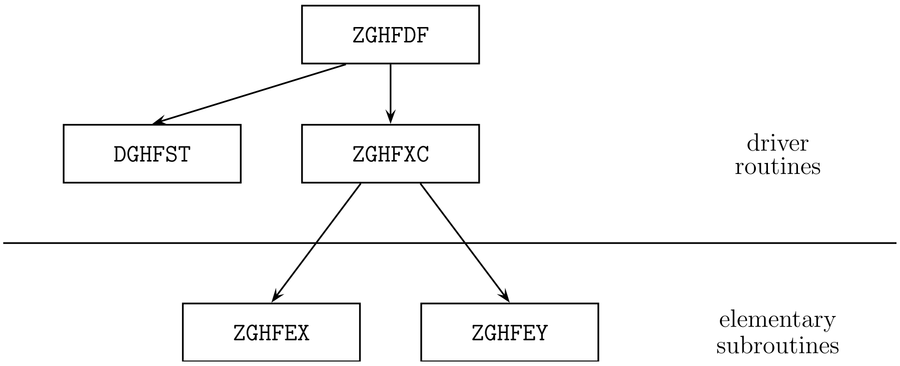


Figure 3: Calling graph for the the computation of the eigenvalues and stable deflating subspace of a complex skew-Hamiltonian/Hamiltonian matrix pencil in factored form

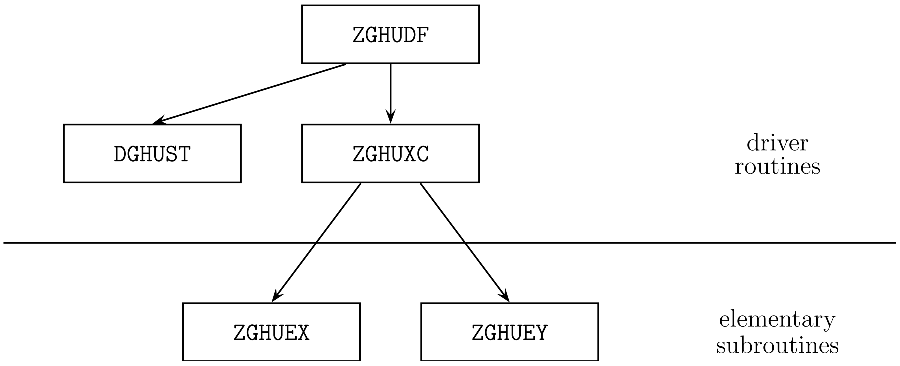


Figure 4: Calling graph for the the computation of the eigenvalues and stable deflating subspace of a complex skew-Hamiltonian/Hamiltonian matrix pencil in unfactored form

3.1.1 Subroutine ZGHFDF (implements Algorithm 1)

Specification:

```
      SUBROUTINE ZGHFDF( COMPQ, COMPU, ORTH, N, Z, LDZ, B, LDB, FG,
     $                   LDFG, NEIG, D, LDD, C, LDC, Q, LDQ, U, LDU,
     $                   ALPHAR, ALPHAI, BETA, IWORK, LIWORK, DWORK,
     $                   LDWORK, ZWORK, LZWORK, BWORK, INFO )
```

Purpose:

To compute the eigenvalues of a complex `N`-by-`N` skew-Hamiltonian/Hamiltonian pencil $\lambda\mathcal{S} - \mathcal{H}$, with

$$\mathcal{S} = \mathcal{J}\mathcal{Z}^H\mathcal{J}^T Z \quad \text{and} \quad \mathcal{H} = \begin{bmatrix} B & F \\ G & -B^H \end{bmatrix}, \quad \mathcal{Z} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}. \qquad (1)$$

The structured Schur form of the embedded real skew-Hamiltonian/skew-Hamiltonian pencil, $\lambda\mathcal{B}_\mathcal{S} - \mathcal{B}_\mathcal{T}$, with $\mathcal{B}_\mathcal{S} = \mathcal{J}\mathcal{B}_\mathcal{Z}^H\mathcal{J}^T\mathcal{B}_\mathcal{Z}$,

$$\mathcal{B}_\mathcal{Z} = \left[\begin{array}{cc|cc} \operatorname{Re}(Z_{11}) & -\operatorname{Im}(Z_{11}) & \operatorname{Re}(Z_{12}) & -\operatorname{Im}(Z_{12}) \\ \operatorname{Im}(Z_{11}) & \operatorname{Re}(Z_{11}) & \operatorname{Im}(Z_{12}) & \operatorname{Re}(Z_{12}) \\ \hline \operatorname{Re}(Z_{21}) & -\operatorname{Im}(Z_{21}) & \operatorname{Re}(Z_{22}) & -\operatorname{Im}(Z_{22}) \\ \operatorname{Im}(Z_{21}) & \operatorname{Re}(Z_{21}) & \operatorname{Im}(Z_{22}) & \operatorname{Re}(Z_{22}) \end{array}\right],$$

$$\mathcal{B}_\mathcal{T} = \left[\begin{array}{cc|cc} -\operatorname{Im}(B) & -\operatorname{Re}(B) & -\operatorname{Im}(F) & -\operatorname{Re}(F) \\ \operatorname{Re}(B) & -\operatorname{Im}(B) & \operatorname{Re}(F) & -\operatorname{Im}(F) \\ \hline -\operatorname{Im}(G) & -\operatorname{Re}(G) & -\operatorname{Im}(B^T) & \operatorname{Re}(B^T) \\ \operatorname{Re}(G) & -\operatorname{Im}(G) & -\operatorname{Re}(B^T) & -\operatorname{Im}(B^T) \end{array}\right], \quad \mathcal{T} = \mathrm{i}\mathcal{H},$$

is determined and used to compute the eigenvalues. Optionally, an orthonormal basis of the right deflating subspace, $\operatorname{Def}_-(\mathcal{S}, \mathcal{H})$, of the pencil $\lambda\mathcal{S} - \mathcal{H}$ in (1), corresponding to the eigenvalues with strictly negative real part, is computed. Namely, after transforming $\lambda\mathcal{B}_\mathcal{S} - \mathcal{B}_\mathcal{H}$, in the factored form, by unitary matrices, we have $\mathcal{B}_{\mathcal{S},\text{out}} = \mathcal{J}\mathcal{B}_{\mathcal{Z},\text{out}}^H\mathcal{J}^T\mathcal{B}_{\mathcal{Z},\text{out}}$,

$$\mathcal{B}_{\mathcal{Z},\text{out}} = \begin{bmatrix} \mathcal{B}_A & \mathcal{B}_D \\ 0 & \mathcal{B}_C \end{bmatrix} \quad \text{and} \quad \mathcal{B}_{\mathcal{H},\text{out}} = \begin{bmatrix} \mathcal{B}_B & \mathcal{B}_F \\ 0 & -\mathcal{B}_B^H \end{bmatrix}, \qquad (2)$$

and the eigenvalues with strictly negative real part of the complex pencil $\lambda\mathcal{B}_{\mathcal{S},\text{out}} - \mathcal{B}_{\mathcal{H},\text{out}}$ are moved to the top. Optionally, an orthonormal basis of the companion subspace, $\operatorname{range}(P_U)$ [5], which corresponds to the eigenvalues with negative real part, is computed. The embedding doubles the multiplicities of the eigenvalues of the pencil $\lambda\mathcal{S} - \mathcal{H}$.

Arguments:

Mode Parameters:

- `COMPQ (CHARACTER*1)`,
  `COMPU (CHARACTER*1)`: Specify whether to compute the right deflating subspace and the companion subspace corresponding to the eigenvalues of $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part, respectively.
  = `'N'`: do not compute the corresponding subspace;
  = `'C'`: compute the corresponding subspace.

- `ORTH (CHARACTER*1)`: Specifies the technique for computing the orthonormal bases of the deflating subspace and companion subspace (if needed).
  = `'P'`: QR factorization with column pivoting;
  = `'S'`: singular value decomposition.

Input/Output Parameters:

- `N (input INTEGER)`: Order of the pencil $\lambda\mathcal{S} - \mathcal{H}$. $\texttt{N} \geq 0$, even.
- `Z (input/output COMPLEX*16 array, dimension (LDZ, N))`: On entry, this array must contain the factor $\mathcal{Z}$ in the factorization $\mathcal{S} = \mathcal{J}\mathcal{Z}^H\mathcal{J}^T\mathcal{Z}$ of the skew-Hamiltonian matrix $\mathcal{S}$. Optionally, on exit, this array contains the matrix $\mathcal{B}_A$ in (2).
- `B (input/output COMPLEX*16 array, dimension (LDB, N))`: On entry, this array must contain the matrix B. Optionally, on exit, this array contains the matrix $\mathcal{B}_B$ in (2).
- `FG (input/output COMPLEX*16 array, dimension (LDFG, N))`: On entry, this array must contain the upper/lower triangular parts of the Hermitian matrices F and G, respectively. Optionally, on exit, this array contains the upper triangular matrix $\mathcal{B}_F$ in (2).
- `NEIG (output INTEGER)`: Optionally, the number of eigenvalues in $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.
- `D (output COMPLEX*16 array, dimension (LDD, N))`,
  `C (output COMPLEX*16 array, dimension (LDC, N))`: Optionally, these arrays contain the matrices $\mathcal{B}_D$ and $\mathcal{B}_C$ in (2), respectively.
- `Q (output COMPLEX*16 array, dimension (LDQ, 2*N))`,
  `U (output COMPLEX*16 array, dimension (LDU, 2*N))`: Optionally, these arrays contain orthonormal bases of the right deflating subspace and the companion subspace corresponding to the eigenvalues of the pencil $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.
- `ALPHAR (output DOUBLE PRECISION array, dimension (N))`,
  `ALPHAI (output DOUBLE PRECISION array, dimension (N))`,
  `BETA   (output DOUBLE PRECISION array, dimension (N))`: The scalars that define the eigenvalues of the pencil $\lambda\mathcal{S} - \mathcal{H}$. Together, the quantities $\alpha =$ `(ALPHAR(j),ALPHAI(j))` and $\beta =$ `BETA(j)` represent the j-th eigenvalue of the pencil $\lambda\mathcal{S} - \mathcal{H}$, in the form $\lambda = \alpha/\beta$. Since λ may overflow, the ratios should not, in general, be computed.

3.1.2 Subroutine DGHFST (implements Algorithm 2)

Specification:

```
      SUBROUTINE DGHFST( JOB, COMPQ, COMPU, N, Z, LDZ, B, LDB, FG, LDFG,
     $                   Q, LDQ, U1, LDU1, U2, LDU2, ALPHAR, ALPHAI,
     $                   BETA, IWORK, LIWORK, DWORK, LDWORK, INFO )
```

Purpose:

To compute the eigenvalues of a real `N`-by-`N` skew-Hamiltonian/skew-Hamiltonian pencil $\lambda\mathcal{S} - \mathcal{T}$ with

$$\mathcal{S} = \mathcal{J}\mathcal{Z}^T\mathcal{J}^T\mathcal{Z} \quad \text{and} \quad \mathcal{T} = \begin{bmatrix} B & F \\ G & B^T \end{bmatrix}.$$

Optionally, the pencil $\lambda\mathcal{S} - \mathcal{T}$ will be transformed to the structured Schur form: an orthogonal transformation matrix $\mathcal{Q}$ and an orthogonal symplectic transformation matrix $\mathcal{U} = \begin{bmatrix} U_1 & U_2 \\ -U_2 & U_1 \end{bmatrix}$ are computed, such that

$$\mathcal{U}^T\mathcal{Z}\mathcal{Q} = \begin{bmatrix} Z_{11} & Z_{12} \\ 0 & Z_{22} \end{bmatrix} = \mathcal{Z}_{\text{out}}, \quad \text{and} \quad \mathcal{J}\mathcal{Q}^T\mathcal{J}^T\mathcal{T}\mathcal{Q} = \begin{bmatrix} B_{\text{out}} & F_{\text{out}} \\ 0 & B_{\text{out}}^T \end{bmatrix}, \tag{3}$$

where Z_{11} and Z_{22}^T are upper triangular and B_{out} is upper quasi-triangular.

Arguments:

Mode Parameters:

- `JOB (CHARACTER*1)`: Specifies whether only the eigenvalues should be computed, or whether the matrices $\mathcal{Z}$ and $\mathcal{T}$ should be also transformed into the forms in (3).
  = `'E'`: compute the eigenvalues only;
  = `'T'`: put $\mathcal{Z}$ and $\mathcal{T}$ into the forms in (3), and return the eigenvalues.

- `COMPQ (CHARACTER*1)`,
  `COMPU (CHARACTER*1)`: Specify whether or not the orthogonal and orthogonal symplectic transformations should be accumulated in the arrays `Q`, `U1`, and `U2`, respectively.
  = `'N'`: the corresponding transformation matrix is not computed;
  = `'I'`: the corresponding transformation matrix is computed;
  = `'U'`: the corresponding transformation matrix is computed but multiplied by a given input matrix as described below.

Input/Output Parameters:

- `N (input INTEGER)`: Order of the pencil $\lambda\mathcal{S} - \mathcal{T}$. $\texttt{N} \geq 0$, even.

- `Z (input/output DOUBLE PRECISION array, dimension (LDZ, N))`,
  `B (input/output DOUBLE PRECISION array, dimension (LDB, N/2))`:
  On entry, these arrays must contain the matrices $\mathcal{Z}$ and B, respectively. Optionally, on exit, these arrays contain the matrices $\mathcal{Z}_{\text{out}}$ and B_{out}, respectively.

- `FG (input/output DOUBLE PRECISION array, dimension (LDFG, N/2+1))`:
  On entry, this array must contain the strictly lower triangular part of the skew-symmetric matrix G, and the strictly upper triangular part of the skew-symmetric matrix F. Optionally, on exit, this array contains the strictly upper triangular part of the skew-symmetric matrix F_{out}.

- `Q (input/output DOUBLE PRECISION array, dimension (LDQ, N))`: Optionally, on entry, this array must contain a given matrix $\mathcal{Q}_0$, and on exit, this array contains the product of the input matrix $\mathcal{Q}_0$ and the transformation matrix $\mathcal{Q}$ used to transform the matrices $\mathcal{Z}$ and $\mathcal{T}$. Optionally, on exit, this array contains only the orthogonal transformation matrix $\mathcal{Q}$.

- `U1 (input/output COMPLEX*16 array, dimension (LDU1, N/2))`,
`U2 (input/output COMPLEX*16 array, dimension (LDU2, N/2))`: Optionally, on entry, these arrays must contain the upper left and right blocks of a given matrix $\mathcal{U}_0$, and on exit, these arrays contain the updated upper left and right blocks U_1 and U_2 of the product of the input matrix $\mathcal{U}_0$ and the transformation matrix $\mathcal{U}$ used to transform the matrices $\mathcal{Z}$ and $\mathcal{T}$. Optionally, on exit, these arrays contain only the upper left and right blocks U_1 and U_2 of the orthogonal symplectic transformation matrix $\mathcal{U}$, respectively.

- `ALPHAR (output DOUBLE PRECISION array, dimension (N/2))`,
`ALPHAI (output DOUBLE PRECISION array, dimension (N/2))`,
`BETA   (output DOUBLE PRECISION array, dimension (N/2))`: The scalars that define the eigenvalues of the pencil $\lambda\mathcal{S} - \mathcal{T}$. Together, the quantities $\alpha =$ `(ALPHAR(j),ALPHAI(j))` and $\beta =$ `BETA(j)` represent the j-th eigenvalue of the pencil $\lambda\mathcal{S} - \mathcal{T}$, in the form $\lambda = \alpha/\beta$. Since λ may overflow, the ratios should not, in general, be computed. Due to the skew-Hamiltonian/skew-Hamiltonian structure of the pencil, every eigenvalue occurs twice and thus it has only to be saved once in `ALPHAR`, `ALPHAI` and `BETA`.

3.1.3 Subroutine `ZGHFXC` (implements Algorithm 3)

Specification:

```
      SUBROUTINE ZGHFXC( COMPQ, COMPU, N, A, LDA, C, LDC, D, LDD, B,
     $                   LDB, F, LDF, Q, LDQ, U1, LDU1, U2, LDU2, NEIG,
     $                   TOL, INFO )
```

Purpose:

To move the eigenvalues with strictly negative real parts of an `N`-by-`N` complex skew-Hamiltonian/Hamiltonian pencil $\lambda\mathcal{S} - \mathcal{H}$ in structured Schur form, with

$$\mathcal{S} = \mathcal{J}\mathcal{Z}^H\mathcal{J}^T\mathcal{Z}$$

to the leading principal subpencil, while keeping the triangular form. On entry, we have

$$\mathcal{Z} = \begin{bmatrix} A & D \\ 0 & C \end{bmatrix}, \quad \mathcal{H} = \begin{bmatrix} B & F \\ 0 & -B^H \end{bmatrix}$$

where A and B are upper triangular and C is lower triangular. $\mathcal{Z}$ and $\mathcal{H}$ are transformed by a unitary symplectic matrix $\mathcal{U}$ and a unitary matrix $\mathcal{Q}$ such that

$$\mathcal{Z}_{\text{out}} = \mathcal{U}^H\mathcal{Z}\mathcal{Q} = \begin{bmatrix} A_{\text{out}} & D_{\text{out}} \\ 0 & C_{\text{out}} \end{bmatrix}, \quad \text{and} \quad \mathcal{H}_{\text{out}} = \mathcal{J}\mathcal{Q}^H\mathcal{J}^T\mathcal{H}\mathcal{Q} = \begin{bmatrix} B_{\text{out}} & F_{\text{out}} \\ 0 & -B_{\text{out}}^H \end{bmatrix}, \tag{4}$$

where A_{out}, B_{out} and C_{out} remain in triangular form. Optionally, the unitary matrix $\mathcal{Q}$ and the unitary symplectic matrix $\mathcal{U} = \begin{bmatrix} U_1 & U_2 \\ -U_2 & U_1 \end{bmatrix}$ that fulfill (4) are computed.

Arguments:

Mode Parameters:

- `COMPQ (CHARACTER*1)`,
  `COMPU (CHARACTER*1)`: Specify whether or not the unitary and unitary symplectic transformations should be accumulated in the arrays `Q`, `U1`, and `U2`, respectively.
  = `'N'`: the corresponding transformation matrix is not computed;
  = `'I'`: the corresponding transformation matrix is computed;
  = `'U'`: the corresponding transformation matrix is computed but multiplied by a given input matrix as described below.

Input/Output Parameters:

- `N (input INTEGER)`: The order of the pencil $\lambda\mathcal{S} - \mathcal{H}$. $\texttt{N} \geq 0$, even.

- `A (input/output COMPLEX*16 array, dimension (LDA, N/2))`,
  `C (input/output COMPLEX*16 array, dimension (LDC, N/2))`,
  `D (input/output COMPLEX*16 array, dimension (LDD, N/2))`,
  `B (input/output COMPLEX*16 array, dimension (LDB, N/2))`: On entry, these arrays must contain the matrices A, C, D, and B, respectively. On exit, these arrays contain the transformed matrices A_{out}, C_{out}, D_{out}, and B_{out}, respectively.

- `F (input/output COMPLEX*16 array, dimension (LDF, N/2))`: On entry, this array must contain the upper triangular part of the matrix F. On exit, this array contains the transformed matrix F_{out}.

- `Q (input/output COMPLEX*16 array, dimension (LDQ, N))`: Optionally, on entry, this array must contain a given matrix $\mathcal{Q}_0$, and on exit, this array contains the product of the input matrix $\mathcal{Q}_0$ and the transformation matrix $\mathcal{Q}$ used to transform the matrices $\mathcal{Z}$ and $\mathcal{H}$. Optionally, on exit, this array contains only the unitary transformation matrix $\mathcal{Q}$.

- `U1 (input/output COMPLEX*16 array, dimension (LDU1, N/2))`,
  `U2 (input/output COMPLEX*16 array, dimension (LDU2, N/2))`: Optionally, on entry, these arrays must contain the upper left and right blocks of a given matrix $\mathcal{U}_0$, and on exit, these arrays contain the updated upper left and right blocks U_1 and U_2 of the product of the input matrix $\mathcal{U}_0$ and the transformation matrix $\mathcal{U}$ used to transform the matrices $\mathcal{Z}$ and $\mathcal{H}$. Optionally, on exit, these arrays contain only the upper left and right blocks U_1 and U_2 of the unitary symplectic transformation matrix $\mathcal{U}$, respectively.

- `NEIG (output INTEGER)`: The number of eigenvalues in $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.

Tolerances:

- `TOL (DOUBLE PRECISION)`: The tolerance used to decide the sign of the eigenvalues. If the user sets `TOL` > 0, then the given value of `TOL` is used. If the user sets `TOL` ≤ 0, then an implicitly computed, default tolerance, defined by $\min\{\texttt{N}, 10\}\varepsilon$, is used instead, where ε is the machine precision. A larger value might be needed for pencils with multiple eigenvalues.

3.1.4 Subroutine `ZGHUDF` (implements Algorithm 4)

Specification:

```
      SUBROUTINE ZGHUDF( COMPQ, ORTH, N, A, LDA, DE, LDDE, B, LDB, FG,
     $                   LDFG, NEIG, Q, LDQ, ALPHAR, ALPHAI, BETA,
     $                   IWORK, DWORK, LDWORK, ZWORK, LZWORK, BWORK,
     $                   INFO )
```

Purpose:

To compute the eigenvalues of a complex `N`-by-`N` skew-Hamiltonian/Hamiltonian pencil $\lambda\mathcal{S} - \mathcal{H}$, with

$$\mathcal{S} = \begin{bmatrix} A & D \\ E & A^H \end{bmatrix} \quad \text{and} \quad \mathcal{H} = \begin{bmatrix} B & F \\ G & -B^H \end{bmatrix}.$$

The structured Schur form of the embedded real skew-Hamiltonian/skew-Hamiltonian pencil $\lambda\mathcal{B}_\mathcal{S} - \mathcal{B}_\mathcal{T}$, defined as

$$\mathcal{B}_\mathcal{S} = \left[\begin{array}{cc|cc} \mathrm{Re}(A) & -\mathrm{Im}(A) & \mathrm{Re}(D) & -\mathrm{Im}(D) \\ \mathrm{Im}(A) & \mathrm{Re}(A) & \mathrm{Im}(D) & \mathrm{Re}(D) \\ \hline \mathrm{Re}(E) & -\mathrm{Im}(E) & \mathrm{Re}(A^T) & \mathrm{Im}(A^T) \\ \mathrm{Im}(E) & \mathrm{Re}(E) & -\mathrm{Im}(A^T) & \mathrm{Re}(A^T) \end{array}\right],$$

$$\mathcal{B}_\mathcal{T} = \left[\begin{array}{cc|cc} -\mathrm{Im}(B) & -\mathrm{Re}(B) & -\mathrm{Im}(F) & -\mathrm{Re}(F) \\ \mathrm{Re}(B) & -\mathrm{Im}(B) & \mathrm{Re}(F) & -\mathrm{Im}(F) \\ \hline -\mathrm{Im}(G) & -\mathrm{Re}(G) & -\mathrm{Im}(B^T) & \mathrm{Re}(B^T) \\ \mathrm{Re}(G) & -\mathrm{Im}(G) & -\mathrm{Re}(B^T) & -\mathrm{Im}(B^T) \end{array}\right], \quad \mathcal{T} = \mathrm{i}\mathcal{H},$$

is determined and used to compute the eigenvalues. Optionally, an orthonormal basis of the right deflating subspace of the pencil $\lambda\mathcal{S} - \mathcal{H}$, corresponding to the eigenvalues with strictly negative real part, is computed. Namely, after transforming $\lambda\mathcal{B}_\mathcal{S} - \mathcal{B}_\mathcal{H}$ by unitary matrices, we have

$$\mathcal{B}_{\mathcal{S},\mathrm{out}} = \begin{bmatrix} \mathcal{B}_A & \mathcal{B}_D \\ 0 & \mathcal{B}_A^H \end{bmatrix} \quad \text{and} \quad \mathcal{B}_{\mathcal{H},\mathrm{out}} = \begin{bmatrix} \mathcal{B}_B & \mathcal{B}_F \\ 0 & -\mathcal{B}_B^H \end{bmatrix}, \tag{5}$$

and the eigenvalues with strictly negative real part of the complex pencil $\lambda\mathcal{B}_{\mathcal{S},\mathrm{out}} - \mathcal{B}_{\mathcal{H},\mathrm{out}}$ are moved to the top. The embedding doubles the multiplicities of the eigenvalues of the pencil $\lambda\mathcal{S} - \mathcal{H}$.

Arguments:

Mode Parameters:

- `COMPQ (CHARACTER*1):` Specifies whether to compute the deflating subspace corresponding to the eigenvalues of $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.
  = `'N'`: do not compute the corresponding subspace;
  = `'C'`: compute the corresponding subspace.

- `ORTH (CHARACTER*1):` Specifies the technique for computing an orthonormal basis of the deflating subspace (if needed).
  = `'P'`: QR factorization with column pivoting;
  = `'S'`: singular value decomposition.

Input/Output Parameters:

- `N (input INTEGER):` The order of the pencil $\lambda\mathcal{S} - \mathcal{H}$. `N` ≥ 0, even.

- `A (input/output COMPLEX*16 array, dimension (LDA, N)),`
  `B (input/output COMPLEX*16 array, dimension (LDB, N)):` On entry, these arrays must contain the matrices A and B. Optionally, on exit, these arrays contain the upper triangular matrices $\mathcal{B}_A$ and $\mathcal{B}_B$ in (5), respectively.

- `DE (input/output COMPLEX*16 array, dimension (LDDE, N)),`
  `FG (input/output COMPLEX*16 array, dimension (LDFG, N)):` On entry, these arrays must contain the (strictly) upper/lower triangular parts of the skew-Hermitian matrices D and E, and the Hermitian matrices F and G, respectively. Optionally, on exit, these arrays contain the upper triangular parts of the matrices $\mathcal{B}_D$ and $\mathcal{B}_F$ in (5), respectively.

- `NEIG (output INTEGER):` Optionally, the number of eigenvalues in $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.

- `Q (output COMPLEX*16 array, dimension (LDQ, 2*N)):` Optionally, on exit, this array contains an orthonormal basis of the right deflating subspace corresponding to the eigenvalues of the pencil $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.

- `ALPHAR (output DOUBLE PRECISION array, dimension (N)),`
  `ALPHAI (output DOUBLE PRECISION array, dimension (N)),`
  `BETA   (output DOUBLE PRECISION array, dimension (N)):` The scalars that define the eigenvalues of the pencil $\lambda\mathcal{S} - \mathcal{H}$. Together, the quantities $\alpha =$ `(ALPHAR(j),ALPHAI(j))` and $\beta =$ `BETA(j)` represent the j-th eigenvalue of the pencil $\lambda\mathcal{S} - \mathcal{H}$, in the form $\lambda = \alpha/\beta$. Since λ may overflow, the ratios should not, in general, be computed.

3.1.5 Subroutine DGHUST (implements Algorithm 5)

Specification:

```
      SUBROUTINE DGHUST( JOB, COMPQ, N, A, LDA, DE, LDDE, B, LDB,
     $                   FG, LDFG, Q, LDQ, ALPHAR, ALPHAI, BETA, DWORK,
     $                   LDWORK, INFO )
```

Purpose:

To compute the eigenvalues of a real N-by-N skew-Hamiltonian/skew-Hamiltonian pencil $\lambda\mathcal{S} - \mathcal{T}$ with

$$\mathcal{S} = \begin{bmatrix} A & D \\ E & A^T \end{bmatrix} \quad \text{and} \quad \mathcal{T} = \begin{bmatrix} B & F \\ G & B^T \end{bmatrix}.$$

Optionally, the pencil $\lambda\mathcal{S} - \mathcal{T}$ will be transformed to the structured Schur form: an orthogonal transformation matrix $\mathcal{Q}$ is computed such that

$$\mathcal{J}\mathcal{Q}^T\mathcal{J}^T\mathcal{S}\mathcal{Q} = \begin{bmatrix} A_{\text{out}} & D_{\text{out}} \\ 0 & A_{\text{out}}^T \end{bmatrix} \quad \text{and} \quad \mathcal{J}\mathcal{Q}^T\mathcal{J}^T\mathcal{T}\mathcal{Q} = \begin{bmatrix} B_{\text{out}} & F_{\text{out}} \\ 0 & B_{\text{out}}^T \end{bmatrix}, \tag{6}$$

A_{out} is upper triangular, and B_{out} is upper quasi-triangular.

Arguments:

Mode Parameters:

- JOB (CHARACTER*1): Specifies whether only the eigenvalues should be computed, or whether the matrices $\mathcal{S}$ and $\mathcal{T}$ should be also transformed into the forms in (6).
  = 'E': compute the eigenvalues only;
  = 'T': put $\mathcal{S}$ and $\mathcal{T}$ into the forms in (6), and return the eigenvalues.

- COMPQ (CHARACTER*1): Specifies whether or not the orthogonal transformations should be accumulated in the array Q.
  = 'N': the transformation matrix is not computed;
  = 'I': the transformation matrix is computed;
  = 'U': the transformation matrix is computed but multiplied by a given input matrix as described below.

Input/Output Parameters:

- N (input INTEGER): The order of the pencil $\lambda\mathcal{S} - \mathcal{T}$. N ≥ 0, even.

- A (input/output DOUBLE PRECISION array, dimension (LDA, N/2)),
  B (input/output DOUBLE PRECISION array, dimension (LDB, N/2)):
  On entry, these arrays must contain the matrices A and B. Optionally, on exit, these arrays contain the matrices A_{out} and B_{out}, respectively.

- `DE (input/output DOUBLE PRECISION array, dimension (LDDE, N/2+1))`, `FG (input/output DOUBLE PRECISION array, dimension (LDFG, N/2+1))`: On entry, these arrays must contain the strictly upper/lower triangular parts of the skew-symmetric matrices D and E, and F and G, respectively. Optionally, on exit, these arrays contain the strictly upper triangular part of the matrices D_{out} and F_{out}.
- `Q (input/output DOUBLE PRECISION array, dimension (LDQ, N))`: Optionally, on entry, this array must contain a given matrix $\mathcal{Q}_0$, and on exit, this array contains the product of the input matrix $\mathcal{Q}_0$ and the transformation matrix $\mathcal{Q}$ used to transform the matrices $\mathcal{S}$ and $\mathcal{H}$. Optionally, on exit, this array contains only the orthogonal transformation matrix $\mathcal{Q}$.
- `ALPHAR (output DOUBLE PRECISION array, dimension (N/2))`, `ALPHAI (output DOUBLE PRECISION array, dimension (N/2))`, `BETA   (output DOUBLE PRECISION array, dimension (N/2))`: The scalars that define the eigenvalues of the pencil $\lambda\mathcal{S} - \mathcal{T}$. Together, the quantities $\alpha =$ `(ALPHAR(j),ALPHAI(j))` and $\beta =$ `BETA(j)` represent the j-th eigenvalue of the pencil $\lambda\mathcal{S} - \mathcal{T}$, in the form $\lambda = \alpha/\beta$. Since λ may overflow, the ratios should not, in general, be computed. Due to the skew-Hamiltonian/skew-Hamiltonian structure of the pencil, every eigenvalue occurs twice and thus it has only to be saved once in `ALPHAR`, `ALPHAI` and `BETA`.

3.1.6 Subroutine ZGHUXC (implements Algorithm 6)

Specification:

```
      SUBROUTINE ZGHUXC( COMPQ, N, A, LDA, D, LDD, B, LDB, F, LDF, Q,
     $                   LDQ, NEIG, TOL, INFO )
```

Purpose:

To move the eigenvalues with strictly negative real parts of an `N`-by-`N` complex skew-Hamiltonian/Hamiltonian pencil $\lambda\mathcal{S} - \mathcal{H}$ in structured Schur form to the leading principal subpencil, while keeping the triangular form. On entry we have

$$\mathcal{S} = \begin{bmatrix} A & D \\ 0 & A^H \end{bmatrix} \quad \text{and} \quad \mathcal{H} = \begin{bmatrix} B & F \\ 0 & -B^H \end{bmatrix}.$$

with A and B upper triangular. $\mathcal{S}$ and $\mathcal{H}$ are transformed by a unitary matrix $\mathcal{Q}$ such that

$$\mathcal{S}_{\text{out}} = \mathcal{J}\mathcal{Q}^H\mathcal{J}^T\mathcal{S}\mathcal{Q} = \begin{bmatrix} A_{\text{out}} & D_{\text{out}} \\ 0 & A_{\text{out}}^H \end{bmatrix} \quad \text{and} \quad \mathcal{H}_{\text{out}} = \mathcal{J}\mathcal{Q}^H\mathcal{J}^T\mathcal{H}\mathcal{Q} = \begin{bmatrix} B_{\text{out}} & F_{\text{out}} \\ 0 & -B_{\text{out}}^H \end{bmatrix}, \tag{7}$$

where A_{out} and B_{out} are upper triangular. Optionally, the matrix $\mathcal{Q}$ that fulfills (7) is computed.

Arguments:

Mode Parameters:

- `COMPQ (CHARACTER*1)`: Specifies whether or not the unitary transformations should be accumulated in the array `Q`.
  = `'N'`: the transformation matrix is not computed;
  = `'I'`: the transformation matrix is computed;
  = `'U'`: the transformation matrix is computed but multiplied by a given input matrix as described below.

Input/Output Parameters:

- `N (input INTEGER)`: The order of the pencil $\lambda\mathcal{S} - \mathcal{H}$. $\mathtt{N} \geq 0$, even.

- `A (input/output COMPLEX*16 array, dimension (LDA, N/2))`,
  `B (input/output COMPLEX*16 array, dimension (LDB, N/2))`: On entry, these arrays must contain the matrices A and B. On exit, these arrays contain the transformed matrices A_{out} and B_{out}, respectively.

- `D (input/output COMPLEX*16 array, dimension (LDD, N/2))`,
  `F (input/output COMPLEX*16 array, dimension (LDF, N/2))`: On entry, these arrays must contain the upper triangular parts of the matrices D and F. On exit, these arrays contain the upper triangular parts of the transformed matrices D_{out} and F_{out}, respectively.

- `Q (input/output COMPLEX*16 array, dimension (LDQ, N))`: Optionally, on entry, this array must contain a given matrix $\mathcal{Q}_0$, and on exit, this array contains the product of the input matrix $\mathcal{Q}_0$ and the transformation matrix $\mathcal{Q}$ used to transform the matrices $\mathcal{S}$ and $\mathcal{H}$. Optionally, on exit, this array contains only the unitary transformation matrix $\mathcal{Q}$.

- `NEIG (output INTEGER)`: The number of eigenvalues in $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.

Tolerances:

- `TOL (DOUBLE PRECISION)`: The tolerance used to decide the sign of the eigenvalues. If the user sets $\mathtt{TOL} > 0$, then the given value of `TOL` is used. If the user sets $\mathtt{TOL} \leq 0$, then an implicitly computed, default tolerance, defined by $\min\{\mathtt{N}, 10\}\varepsilon$, is used instead, where ε is the machine precision. A larger value might be needed for pencils with multiple eigenvalues.

3.2 The Real Case

In this subsection we describe the interfaces of the subroutines needed for computing the eigenvalues and stable deflating subspaces of a real skew-Hamiltonian/Hamiltonian matrix pencil. The calling graphs for the factored and the unfactored case are depicted in Figures 5 and 6, respectively.

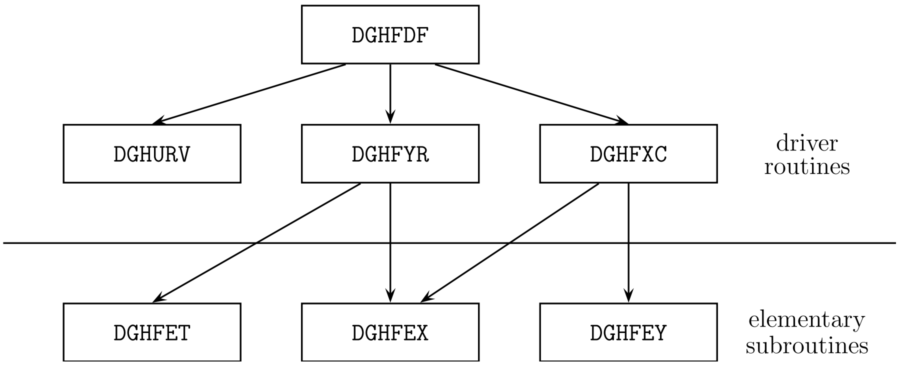


Figure 5: Calling graph for the the computation of the eigenvalues and stable deflating subspace of a real skew-Hamiltonian/Hamiltonian matrix pencil in factored form

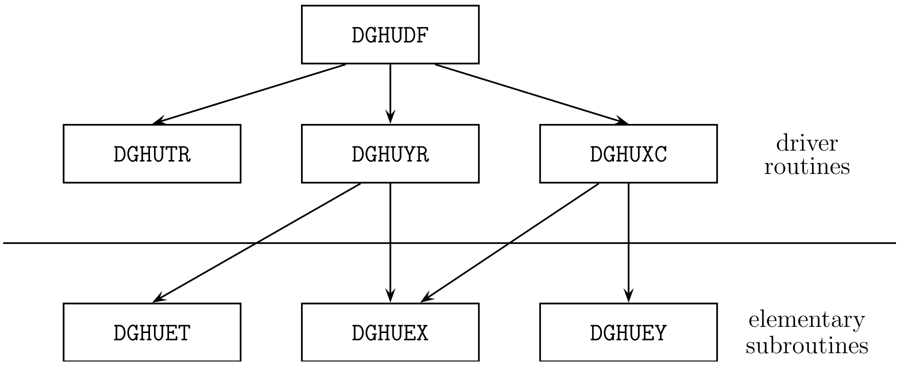


Figure 6: Calling graph for the the computation of the eigenvalues and stable deflating subspaces of a real skew-Hamiltonian/Hamiltonian matrix pencil in unfactored form

3.2.1 Subroutine DGHFDF (implements Algorithm 7)

Specification:

```
      SUBROUTINE DGHFDF( COMPQ, COMPU, ORTH, N, Z, LDZ, B, LDB, FG,
     $                   LDFG, NEIG, Q, LDQ, U, LDU, ALPHAR, ALPHAI,
     $                   BETA, IWORK, LIWORK, DWORK, LDWORK, BWORK,
     $                   IWARN, INFO )
```

Purpose:

To compute the relevant eigenvalues of a real `N`-by-`N` skew-Hamiltonian/Hamiltonian pencil $\lambda\mathcal{S} - \mathcal{H}$, with

$$\mathcal{S} = \mathcal{T}\mathcal{Z} = \mathcal{J}\mathcal{Z}^T\mathcal{J}^T\mathcal{Z} \quad \text{and} \quad \mathcal{H} = \begin{bmatrix} B & F \\ G & -B^T \end{bmatrix}.$$

Optionally, an orthonormal basis of the right deflating subspace of $\lambda\mathcal{S} - \mathcal{H}$ corresponding to the eigenvalues with strictly negative real part is computed. Optionally, an orthonormal basis of the companion subspace, range(P_U) [5], which corresponds to the eigenvalues with strictly negative real part, is computed.

Arguments:

Mode Parameters:

- `COMPQ (CHARACTER*1)`,
  `COMPU (CHARACTER*1)`: Specify whether to compute the right deflating subspace and companion subspace corresponding to the eigenvalues of $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part, respectively.
  = `'N'`: do not compute the corresponding subspace;
  = `'C'`: compute the corresponding subspace.

- `ORTH (CHARACTER*1)`: Specifies the technique for computing the orthogonal basis of the deflating subspace, and/or of the companion subspace (if needed).
  = `'P'`: QR factorization with column pivoting;
  = `'S'`: singular value decomposition.

Input/Output Parameters:

- `N (input INTEGER)`: The order of the pencil $\lambda\mathcal{S} - \mathcal{H}$. `N` ≥ 0, even.

- `Z (input/output DOUBLE PRECISION array, dimension (LDZ, N))`: On entry, this array must contain the non-trivial factor $\mathcal{Z}$ in the factorization $\mathcal{S} = \mathcal{J}\mathcal{Z}^T\mathcal{J}^T\mathcal{Z}$ of the skew-Hamiltonian matrix $\mathcal{S}$. On exit, this array is overwritten by some intermediate results, depending on the values of `COMPQ` and `COMPU`.

- `B (input DOUBLE PRECISION array, dimension (LDB, N/2))`: On entry, this array must contain the matrix B.

- `FG (input DOUBLE PRECISION array, dimension (LDFG, N/2+1))`: On entry, this array must contain the upper/lower triangular parts of the Hermitian matrices F and G, respectively.

- `NEIG (output INTEGER)`: Optionally, the number of eigenvalues in $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.

- `Q (output DOUBLE PRECISION array, dimension (LDQ, 2*N))`,
  `U (output DOUBLE PRECISION array, dimension (LDU, 2*N))`: Optionally, on exit, these arrays contain orthogonal bases of the right deflating subspace and the companion subspace corresponding to the eigenvalues of $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.

- `ALPHAR (output DOUBLE PRECISION array, dimension (N/2))`,
  `ALPHAI (output DOUBLE PRECISION array, dimension (N/2))`,
  `BETA   (output DOUBLE PRECISION array, dimension (N/2))`: The scalars that define the eigenvalues of the pencil $\lambda\mathcal{S}-\mathcal{H}$. If `INFO = 0`, the quantities $\alpha =$ `(ALPHAR(j),ALPHAI(j))`, and $\beta =$ `BETA(j)` represent together the j-th eigenvalue of the pencil $\lambda\mathcal{S}-\mathcal{H}$, in the form $\lambda = \alpha/\beta$. Since λ may overflow, the ratios should not, in general, be computed. Due to the skew-Hamiltonian/Hamiltonian structure of the pencil, only half of the spectrum is saved in `ALPHAR`, `ALPHAI` and `BETA`. Specifically, the eigenvalues with positive real parts or with non-negative imaginary parts, when real parts are zero, are returned. The remaining eigenvalues have opposite signs. If `IWARN = 1`, one or more `BETA(j)` is not representable. Therefore, the j-th eigenvalue is represented by the quantities $\alpha =$ `(ALPHAR(j),ALPHAI(j))`, $\beta =$ `BETA(j)`, and $\gamma =$ `IWORK(j)` in the form $\lambda = (\alpha/\beta)\cdot\mathbf{b}^{\gamma}$, where $\mathbf{b}$ is the machine base (often 2.0), returned in `DWORK(2)`.

3.2.2 Subroutine DGHURV (implements Algorithm 8)

Specification:

```
      SUBROUTINE DGHURV( JOB, COMPQ1, COMPQ2, COMPU1, COMPU2, N, Z, LDZ,
     $                   H, LDH, Q1, LDQ1, Q2, LDQ2, U11, LDU11, U12,
     $                   LDU12, U21, LDU21, U22, LDU22, T, LDT, ALPHAR,
     $                   ALPHAI, BETA, IWORK, LIWORK, DWORK, LDWORK,
     $                   INFO )
```

Purpose:

To compute the eigenvalues of a real `N`-by-`N` skew-Hamiltonian/Hamiltonian pencil $\lambda\mathcal{S}-\mathcal{H}$ with $\mathcal{S} = \mathcal{T}\mathcal{Z} = \mathcal{J}\mathcal{Z}^T\mathcal{J}^T\mathcal{Z}$ via generalized symplectic URV decomposition. That is, orthogonal matrices $\mathcal{Q}_1$ and $\mathcal{Q}_2$ and orthogonal symplectic matrices $\mathcal{U}_1$ and $\mathcal{U}_2$ are computed such that

$$\begin{aligned}
\mathcal{Q}_1^T\mathcal{T}\mathcal{U}_1 = \mathcal{Q}_1^T\mathcal{J}\mathcal{Z}^T\mathcal{J}^T\mathcal{U}_1 &= \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} = \mathcal{T}_{\text{out}}, \\
\mathcal{U}_2^T\mathcal{Z}\mathcal{Q}_2 &= \begin{bmatrix} Z_{11} & Z_{12} \\ 0 & Z_{22} \end{bmatrix} = \mathcal{Z}_{\text{out}}, \\
\mathcal{Q}_1^T\mathcal{H}\mathcal{Q}_2 &= \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix} = \mathcal{H}_{\text{out}},
\end{aligned} \tag{8}$$

where T_{11}, T_{22}^T, Z_{11}, Z_{22}^T, H_{11} are upper triangular and H_{22}^T is upper quasi-triangular. Optionally, the orthogonal transformation matrices $\mathcal{Q}_1$ and $\mathcal{Q}_2$, and the orthogonal symplectic transformation matrices $\mathcal{U}_1 = \begin{bmatrix} U_{11} & U_{12} \\ -U_{12} & U_{11} \end{bmatrix}$ and $\mathcal{U}_2 = \begin{bmatrix} U_{21} & U_{22} \\ -U_{22} & U_{21} \end{bmatrix}$ will be computed.

Arguments:

Mode Parameters:

- JOB (CHARACTER*1): Specifies whether only the eigenvalues should be computed, or whether the matrices $\mathcal{Z}$, $\mathcal{T}$, and $\mathcal{H}$ should be also transformed into the forms in (8).
  = 'E': compute the eigenvalues only;
  = 'T': put $\mathcal{Z}$, $\mathcal{T}$, and $\mathcal{J}$ into the forms in (8), and return the eigenvalues.

- COMPQ1 (CHARACTER*1),
  COMPQ2 (CHARACTER*1),
  COMPU1 (CHARACTER*1),
  COMPU2 (CHARACTER*1): Specify whether or not the orthogonal and orthogonal symplectic transformations should be accumulated in the arrays Q1, Q2, U11, U12, U21, and U22, respectively.
  = 'N': the corresponding transformation matrix is not computed;
  = 'I': the corresponding transformation matrix is computed;
  = 'U': the corresponding transformation matrix is computed but multiplied by a given input matrix as described below.

Input/Output Parameters:

- N (input INTEGER): The order of the pencil $\lambda\mathcal{S} - \mathcal{H}$. N ≥ 0, even.

- Z (input/output DOUBLE PRECISION array, dimension (LDZ, N)),
  H (input/output DOUBLE PRECISION array, dimension (LDH, N)): On entry, these arrays must contain the matrices $\mathcal{Z}$ and $\mathcal{H}$. Optionally, on exit, this arrays contain the matrices $\mathcal{Z}_{\mathrm{out}}$ and $\mathcal{H}_{\mathrm{out}}$.

- Q1 (input/output DOUBLE PRECISION array, dimension (LDQ1, N)),
  Q2 (input/output DOUBLE PRECISION array, dimension (LDQ2, N)): Optionally, on entry, these arrays must contain given matrices $\mathcal{Q}_{01}$ and $\mathcal{Q}_{02}$, and on exit, these arrays contain the product of the input matrices $\mathcal{Q}_{01}$ and $\mathcal{Q}_{02}$ and the transformation matrices $\mathcal{Q}_1$ and $\mathcal{Q}_2$, respectively, used to transform the matrices $\mathcal{Z}$, $\mathcal{T}$, and $\mathcal{H}$. Optionally, on exit, these arrays contain only the orthogonal transformation matrices $\mathcal{Q}_1$ and $\mathcal{Q}_2$.

- U11 (input/output DOUBLE PRECISION array, dimension (LDU11, N/2)),
  U12 (input/output DOUBLE PRECISION array, dimension (LDU12, N/2)),
  U21 (input/output DOUBLE PRECISION array, dimension (LDU21, N/2)),
  U22 (input/output DOUBLE PRECISION array, dimension (LDU22, N/2)): Optionally, on entry, these arrays must contain the upper left and right blocks of given matrices $\mathcal{U}_{01}$ and $\mathcal{U}_{02}$, and on exit, these arrays contain the updated upper left and right blocks U_{11}, U_{12}, U_{21}, and U_{22} of the product of the input matrices $\mathcal{U}_{01}$ and $\mathcal{U}_{02}$ and the transformation matrices $\mathcal{U}_1$ and $\mathcal{U}_2$, respectively, used to transform the matrices $\mathcal{Z}$ and $\mathcal{H}$. Optionally, on exit, these arrays contain only the upper left and right blocks U_{11}, U_{12}, U_{21}, and U_{22} of the orthogonal symplectic transformation matrices $\mathcal{U}_1$ and $\mathcal{U}_2$, respectively.

- T (output DOUBLE PRECISION array, dimension (LDT, N)): Optionally, on exit, this array contains the matrix $\mathcal{T}_{\mathrm{out}}$.

- `ALPHAR (output DOUBLE PRECISION array, dimension (N/2))`,
  `ALPHAI (output DOUBLE PRECISION array, dimension (N/2))`,
  `BETA   (output DOUBLE PRECISION array, dimension (N/2))`: The scalars that define the eigenvalues of the pencil $\lambda\mathcal{S}-\mathcal{H}$. If `INFO = 0`, the quantities $\alpha =$ `(ALPHAR(j),ALPHAI(j))`, and $\beta =$ `BETA(j)` represent together the j-th eigenvalue of the pencil $\lambda\mathcal{S}-\mathcal{H}$, in the form $\lambda = \alpha/\beta$. Since λ may overflow, the ratios should not, in general, be computed. Due to the skew-Hamiltonian/Hamiltonian structure of the pencil, only half of the spectrum is saved in `ALPHAR`, `ALPHAI` and `BETA`. Specifically, the eigenvalues with positive real parts or with non-negative imaginary parts, when real parts are zero, are returned. The remaining eigenvalues have opposite signs. If `INFO = 3`, one or more `BETA(j)` is not representable. Therefore, the j-th eigenvalue is represented by the quantities $\alpha =$ `(ALPHAR(j),ALPHAI(j))`, $\beta =$ `BETA(j)`, and $\gamma =$ `IWORK(j)` in the form $\lambda = (\alpha/\beta)\cdot\mathbf{b}^{\gamma}$, where $\mathbf{b}$ is the machine base (often 2.0). This is not an error.

3.2.3 Subroutine `DGHFYR` (implements Algorithm 9)

Specification:

```
      SUBROUTINE DGHFYR( COMPQ1, COMPQ2, COMPQ3, N, A, LDA, B, LDB, D,
     $                   LDD, Q1, LDQ1, Q2, LDQ2, Q3, LDQ3, IWORK,
     $                   LIWORK, DWORK, LDWORK, BWORK, INFO )
```

Purpose:

To compute the transformed matrices $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{D}$, using orthogonal matrices $\mathcal{Q}_1$, $\mathcal{Q}_2$ and $\mathcal{Q}_3$ for a real `N`-by-`N` regular pencil

$$\lambda\mathcal{A}\mathcal{B}-\mathcal{D} = \lambda\begin{bmatrix} A_{11} & 0 \\ 0 & A_{22}\end{bmatrix}\begin{bmatrix} B_{11} & 0 \\ 0 & B_{22}\end{bmatrix} - \begin{bmatrix} 0 & D_{12} \\ D_{21} & 0\end{bmatrix}, \qquad (9)$$

where A_{11}, A_{22}, B_{11}, B_{22}, and D_{12} are upper triangular, D_{21} is upper quasi-triangular and the generalized matrix product $A_{11}^{-1}D_{12}B_{22}^{-1}A_{22}^{-1}D_{21}B_{11}^{-1}$ is in periodic Schur form, such that $\mathcal{Q}_3^T\mathcal{A}\mathcal{Q}_2$, $\mathcal{Q}_2^T\mathcal{B}\mathcal{Q}_1$ are upper triangular, $\mathcal{Q}_3^T\mathcal{D}\mathcal{Q}_1$ is upper quasi-triangular, and the pencil $\lambda\mathcal{Q}_3^T\mathcal{A}\mathcal{B}\mathcal{Q}_1-\mathcal{Q}_3^T\mathcal{D}\mathcal{Q}_1$ is in generalized Schur form.

Arguments:

Mode Parameters:

- `COMPQ1 (CHARACTER*1)`,
  `COMPQ2 (CHARACTER*1)`,
  `COMPQ3 (CHARACTER*1)`: Specify whether or not the orthogonal transformations should be accumulated in the arrays `Q1`, `Q2`, `Q3`.
  = `'N'`: the corresponding transformation matrix is not computed;
  = `'I'`: the corresponding transformation matrix is computed;
  = `'U'`: the corresponding transformation matrix is computed but multiplied by a given input matrix as described below.

Input/Output Parameters:

- `N (input INTEGER):` Order of the pencil $\lambda\mathcal{A}\mathcal{B} - \mathcal{D}$. $\mathtt{N} \geq 0$, even.
- `A (input/output DOUBLE PRECISION array, dimension (LDA, N)),`
  `B (input/output DOUBLE PRECISION array, dimension (LDB, N)),`
  `D (input/output DOUBLE PRECISION array, dimension (LDD, N)):` On entry, these arrays must contain the matrices $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{D}$ in (9). The zero (off-)diagonal blocks need not be set to zero. On exit, these arrays contain the transformed upper (quasi-)triangular matrices.
- `Q1 (input/output DOUBLE PRECISION array, dimension (LDQ1, N)),`
  `Q2 (input/output DOUBLE PRECISION array, dimension (LDQ2, N)),`
  `Q3 (input/output DOUBLE PRECISION array, dimension (LDQ3, N)):`
  Optionally, on entry, these arrays must contain given matrices $\mathcal{Q}_{01}$, $\mathcal{Q}_{02}$, and $\mathcal{Q}_{03}$ and on exit, these arrays contain the product of the input matrices $\mathcal{Q}_{01}$, $\mathcal{Q}_{02}$, and $\mathcal{Q}_{03}$ and the transformation matrices $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{Q}_3$, respectively, used to transform the matrices $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{D}$. Optionally, on exit, these arrays contain only the orthogonal transformation matrices $\mathcal{Q}_1$, $\mathcal{Q}_2$, and $\mathcal{Q}_3$, respectively.

3.2.4 Subroutine DGHFXC (implements Algorithm 10)

Specification:

```
      SUBROUTINE DGHFXC( COMPQ, COMPU, N, A, LDA, C, LDC, D, LDD, B,
     $                   LDB, F, LDF, Q, LDQ, U1, LDU1, U2, LDU2, NEIG,
     $                   IWORK, LIWORK, DWORK, LDWORK, INFO )
```

Purpose:

To move the eigenvalues with strictly negative real parts of an N-by-N real skew-Hamiltonian/Hamiltonian pencil $\lambda\mathcal{S} - \mathcal{H}$ in structured Schur form, with $\mathcal{S} = \mathcal{J}\mathcal{Z}^T\mathcal{J}^T\mathcal{Z}$,

$$\mathcal{Z} = \begin{bmatrix} A & D \\ 0 & C \end{bmatrix}, \quad \text{and} \quad \mathcal{H} = \begin{bmatrix} B & F \\ 0 & -B^T \end{bmatrix}$$

to the leading principal subpencil, while keeping the triangular form. Above, A is upper triangular, B is upper quasi-triangular, and C is lower triangular. The matrices $\mathcal{Z}$ and $\mathcal{H}$ are transformed by an orthogonal symplectic matrix $\mathcal{U}$ and an orthogonal matrix $\mathcal{Q}$ such that

$$\mathcal{Z}_{\text{out}} = \mathcal{U}^T\mathcal{Z}\mathcal{Q} = \begin{bmatrix} A_{\text{out}} & D_{\text{out}} \\ 0 & C_{\text{out}} \end{bmatrix} \quad \text{and} \quad \mathcal{H}_{\text{out}} = \mathcal{J}\mathcal{Q}^T\mathcal{J}^T\mathcal{H}\mathcal{Q} = \begin{bmatrix} B_{\text{out}} & F_{\text{out}} \\ 0 & -B_{\text{out}}^T \end{bmatrix}, \tag{10}$$

where A_{out}, B_{out}, and C_{out} remain in triangular form. Optionally, the orthogonal matrix $\mathcal{Q}$ and the orthogonal symplectic matrix $\mathcal{U} = \begin{bmatrix} U_1 & U_2 \\ -U_2 & U_1 \end{bmatrix}$ that fulfill (10) are computed.

Arguments:

Mode Parameters:

- `COMPQ (CHARACTER*1)`,
  `COMPU (CHARACTER*1):` Specify whether or not the orthogonal and orthogonal symplectic transformations should be accumulated in the arrays `Q`, `U1`, and `U2`.
  = `'N'`: the corresponding transformation matrix is not computed;
  = `'I'`: the corresponding transformation matrix is computed;
  = `'U'`: the corresponding transformation matrix is computed but multiplied by a given input matrix as described below.

Input/Output Parameters:

- `N (input INTEGER):` The order of the pencil $\lambda\mathcal{S} - \mathcal{H}$. $\mathtt{N} \geq 0$, even.

- `A (input/output DOUBLE PRECISION array, dimension (LDA, N/2))`,
  `C (input/output DOUBLE PRECISION array, dimension (LDC, N/2))`,
  `D (input/output DOUBLE PRECISION array, dimension (LDD, N/2))`,
  `B (input/output DOUBLE PRECISION array, dimension (LDB, N/2)):`
  On entry, these arrays must contain the matrices A, C, D, and B, respectively. On exit, these arrays contain the transformed matrices A_{out}, C_{out}, D_{out}, and B_{out}, respectivly.

- `F (input/output DOUBLE PRECISION array, dimension (LDF, N/2)):`
  On entry, this array must contain the upper triangular part of the matrix F. On exit, this array contains the transformed upper triangular part of the matrix F_{out}.

- `Q (input/output DOUBLE PRECISION array, dimension (LDQ, N)):` Optionally, on entry, this array must contain a given matrix $\mathcal{Q}_0$, and on exit, this array contains the product of the input matrix $\mathcal{Q}_0$ and the transformation matrix $\mathcal{Q}$ used to transform the matrices $\mathcal{Z}$ and $\mathcal{H}$. Optionally, on exit, this array contains only the orthogonal transformation matrix $\mathcal{Q}$.

- `U1 (input/output DOUBLE PRECISION array, dimension (LDU1, N/2))`,
  `U2 (input/output DOUBLE PRECISION array, dimension (LDU2, N/2)):`
  Optionally, on entry, these arrays must contain the upper left and right blocks of a given matrix $\mathcal{U}_0$, and on exit, these arrays contain the updated upper left and right blocks U_1 and U_2 of the product of the input matrix $\mathcal{U}_0$ and the transformation matrix $\mathcal{U}$ used to transform the matrices $\mathcal{Z}$ and $\mathcal{H}$. Optionally, on exit, these arrays contain only the upper left and right blocks U_1 and U_2 of the orthogonal symplectic transformation matrix $\mathcal{U}$, respectively.

- `NEIG (output INTEGER):` The number of eigenvalues in $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.

3.2.5 Subroutine DGHUDF (implements Algorithm 11)

Specification:

```
      SUBROUTINE DGHUDF( COMPQ, ORTH, N, A, LDA, DE, LDDE, B, LDB, FG,
     $                   LDFG, NEIG, Q, LDQ, ALPHAR, ALPHAI, BETA,
     $                   IWORK, LIWORK, DWORK, LDWORK, BWORK, INFO )
```

Purpose:

To compute the relevant eigenvalues of a real N-by-N skew-Hamiltonian/Hamiltonian pencil $\lambda\mathcal{S} - \mathcal{H}$, with

$$\mathcal{S} = \begin{bmatrix} A & D \\ E & A^T \end{bmatrix} \quad \text{and} \quad \mathcal{H} = \begin{bmatrix} B & F \\ G & -B^T \end{bmatrix}.$$

Optionally, an orthogonal basis of the right deflating subspace of $\lambda\mathcal{S} - \mathcal{H}$ corresponding to the eigenvalues with strictly negative real part is computed.

Arguments:

Mode Parameters:

- COMPQ (CHARACTER*1): Specifies whether to compute the right deflating subspace corresponding to the eigenvalues of $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.
  = 'N': do not compute the corresponding subspace;
  = 'C': compute the corresponding subspace.

- ORTH (CHARACTER*1): Specifies the technique for computing the orthogonal basis of the deflating subspace (if needed).
  = 'P': QR factorization with column pivoting;
  = 'S': singular value decomposition.

Input/Output Parameters:

- N (input INTEGER): The order of the pencil $\lambda\mathcal{S} - \mathcal{H}$. N ≥ 0, even.

- A (input/output DOUBLE PRECISION array, dimension (LDA, N/2)),
  B (input/output DOUBLE PRECISION array, dimension (LDB, N/2)):
  On entry, these arrays must contain the matrices A and B. On exit, these arrays are overwritten by some intermediate results, depending on the value of COMPQ.

- DE (input/output DOUBLE PRECISION array, dimension (LDDE, N/2+1)),
  FG (input/output DOUBLE PRECISION array, dimension (LDFG, N/2+1)):
  On entry, these arrays must contain the (strictly) upper/lower triangular parts of the skew-symmetric matrices D and E, and the symmetric F and G. On exit, these arrays are overwritten by some intermediate results, depending on the value of COMPQ.

- NEIG (output INTEGER): Optionally, the number of eigenvalues in $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.

- Q (output DOUBLE PRECISION array, dimension (LDQ, 2*N)): Optionally, on exit, this array contains an orthogonal basis of the right deflating subspace corresponding to the eigenvalues of $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.

- ALPHAR (output DOUBLE PRECISION array, dimension (N/2)),
  ALPHAI (output DOUBLE PRECISION array, dimension (N/2)),
  BETA (output DOUBLE PRECISION array, dimension (N/2)): The scalars that define the eigenvalues of the pencil $\lambda\mathcal{S} - \mathcal{H}$. Together, the quantities α = (ALPHAR(j),ALPHAI(j)), and β = BETA(j) represent the j-th eigenvalue of the pencil $\lambda\mathcal{S} - \mathcal{H}$, in the form $\lambda = \alpha/\beta$. Since λ may overflow, the ratios should not, in general, be computed. Due to the skew-Hamiltonian/Hamiltonian structure of the pencil, only half of the spectrum is saved in ALPHAR, ALPHAI and BETA. Specifically, the eigenvalues with positive real parts or with non-negative imaginary parts, when real parts are zero, are returned. The remaining eigenvalues have opposite signs.

3.2.6 Subroutine DGHUTR (implements Algorithm 12)

Specification:

```
      SUBROUTINE DGHUTR( JOB, COMPQ1, COMPQ2, N, A, LDA, DE, LDDE, C1,
     $                   LDC1, VW, LDVW, Q1, LDQ1, Q2, LDQ2, B, LDB, F,
     $                   LDF, C2, LDC2, ALPHAR, ALPHAI, BETA, IWORK,
     $                   LIWORK, DWORK, LDWORK, INFO )
```

Purpose:

To compute the eigenvalues of a real N-by-N skew-Hamiltonian/Hamiltonian pencil $\lambda\mathcal{S} - \mathcal{H}$ with

$$\mathcal{S} = \begin{bmatrix} A & D \\ E & A^T \end{bmatrix} \quad \text{and} \quad \mathcal{H} = \begin{bmatrix} C & V \\ W & -C^T \end{bmatrix}.$$

Optionally, decompositions of $\mathcal{S}$ and $\mathcal{H}$ will be computed via orthogonal transformations $\mathcal{Q}_1$ and $\mathcal{Q}_2$ such that

$$\begin{aligned} \mathcal{Q}_1^T \mathcal{S} \mathcal{J} \mathcal{Q}_1 \mathcal{J}^T &= \begin{bmatrix} A_{\text{out}} & D_{\text{out}} \\ 0 & A_{\text{out}}^T \end{bmatrix}, \\ \mathcal{J} \mathcal{Q}_2^T \mathcal{J}^T \mathcal{S} \mathcal{Q}_2 &= \begin{bmatrix} B_{\text{out}} & F_{\text{out}} \\ 0 & B_{\text{out}}^T \end{bmatrix} = \mathcal{T}, \\ \mathcal{Q}_1^T \mathcal{H} \mathcal{Q}_2 &= \begin{bmatrix} C_{1,\text{out}} & V_{\text{out}} \\ 0 & C_{2,\text{out}}^T \end{bmatrix}, \end{aligned} \tag{11}$$

and A_{out}, B_{out}, $C_{1,\text{out}}$ are upper triangular, $C_{2,\text{out}}$ is upper quasi-triangular and D_{out} and F_{out} are skew-symmetric. Optionally, the orthogonal transformation matrices $\mathcal{Q}_1$ and $\mathcal{Q}_2$ will be computed.

Arguments:

Mode Parameters:

- `JOB (CHARACTER*1)`: Specifies whether only the eigenvalues should be computed, or whether the matrices $\mathcal{S}$ and $\mathcal{H}$ should be also transformed into the forms in (11).
  = `'E'`: compute the eigenvalues only;
  = `'T'`: put $\mathcal{S}$ and $\mathcal{H}$ into the forms in (11), and return the eigenvalues.

- `COMPQ1 (CHARACTER*1)`:

- `COMPQ2 (CHARACTER*1)`: Specify whether or not the orthogonal transformations should be accumulated in the arrays `Q1`, `Q2`.
  = `'N'`: the corresponding transformation matrix is not computed;
  = `'I'`: the corresponding transformation matrix is computed;
  = `'U'`: the corresponding transformation matrix is computed but multiplied by a given input matrix as described below.

Input/Output Parameters:

- `N (input INTEGER)`: The order of the pencil $\lambda\mathcal{S} - \mathcal{H}$. $\texttt{N} \geq 0$, even.

- `A  (input/output DOUBLE PRECISION array, dimension (LDA, N/2))`,
  `C1 (input/output DOUBLE PRECISION array, dimension (LDC1, N/2))`:
  On entry, these arrays must contain the matrices A and C. Optionally, on exit, these arrays contain the matrices A_{out} and $C_{1,\text{out}}$, respectively.

- `DE (input/output DOUBLE PRECISION array, dimension (LDDE, N/2+1))`,
  `VW (input/output DOUBLE PRECISION array, dimension (LDVW, N/2+1))`:
  On entry, these arrays must contain the upper/lower triangular parts of the skew-symmetric matrices D and E, and the symmetric matrices V and W, respectively. Optionally, on exit, these arrays contain the matrices D_{out} and V_{out}, respectively.

- `Q1 (input/output DOUBLE PRECISION array, dimension (LDQ1, N))`:
  Optionally, on entry, this array must contain a given matrix $\mathcal{Q}$, and on exit, this array contains the product of the input matrix $\mathcal{Q}$ and the transformation matrix $\mathcal{Q}_1$ used to transform the matrices $\mathcal{S}$ and $\mathcal{H}$. Optionally, on exit, this array contains only the orthogonal transformation matrix $\mathcal{Q}_1$.

- `Q2 (output DOUBLE PRECISION array, dimension (LDQ2, N))`: Optionally, on exit, this array contains the product of the matrix $\mathcal{J}\mathcal{Q}\mathcal{J}^T$ and the transformation matrix $\mathcal{Q}_2$ used to transform the matrices $\mathcal{S}$ and $\mathcal{H}$. Optionally, on exit, this array contains only the orthogonal transformation matrix $\mathcal{Q}_2$.

- `B  (output DOUBLE PRECISION array, dimension (LDB, N/2))`,
  `C2 (output DOUBLE PRECISION array, dimension (LDC2, N/2))`: Optionally, on exit, these arrays contain the matrices B_{out} and $C_{2,\text{out}}$, respectively.

- `F (output DOUBLE PRECISION array, dimension (LDF, N/2))`: Optionally, on exit, this array contains the strictly upper triangular part of the matrix F_{out}.

- `ALPHAR (output DOUBLE PRECISION array, dimension (N/2))`,
  `ALPHAI (output DOUBLE PRECISION array, dimension (N/2))`,
  `BETA   (output DOUBLE PRECISION array, dimension (N/2))`: The scalars that define the eigenvalues of the pencil $\lambda\mathcal{S} - \mathcal{H}$. Together, the quantities α = `(ALPHAR(j),ALPHAI(j))`, and β = `BETA(j)` represent the j-th eigenvalue of the pencil $\lambda\mathcal{S} - \mathcal{H}$, in the form $\lambda = \alpha/\beta$. Since λ may overflow, the ratios should not, in general, be computed. Due to the skew-Hamiltonian/Hamiltonian structure of the pencil, only half of the spectrum is saved in `ALPHAR`, `ALPHAI` and `BETA`. Specifically, the eigenvalues with positive real parts or with non-negative imaginary parts, when real parts are zero, are returned. The remaining eigenvalues have opposite signs.

3.2.7 Subroutine `DGHUYR` (implements Algorithm 13)

Specification:

```
      SUBROUTINE DGHUYR( COMPQ1, COMPQ2, N, A, LDA, B, LDB, Q1, LDQ1,
     $                   Q2, LDQ2, IWORK, LIWORK, DWORK, LDWORK, BWORK,
     $                   INFO )
```

Purpose:

To compute the transformed matrices $\mathcal{A}$ and $\mathcal{B}$, using orthogonal matrices $\mathcal{Q}_1$ and $\mathcal{Q}_2$ for a real N-by-N regular pencil

$$\lambda\mathcal{A} - \mathcal{B} = \lambda \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} - \begin{bmatrix} 0 & B_{12} \\ B_{21} & 0 \end{bmatrix}, \tag{12}$$

where A_{11}, A_{22} and B_{12} are upper triangular, B_{21} is upper quasi-triangular and the generalized matrix product $A_{11}^{-1}B_{12}A_{22}^{-1}B_{21}$ is in periodic Schur form, such that $\mathcal{Q}_2^T\mathcal{A}\mathcal{Q}_1$ is upper triangular, $\mathcal{Q}_2^T\mathcal{B}\mathcal{Q}_1$ is upper quasi-triangular, and the matrix pencil $\lambda\mathcal{Q}_2^T\mathcal{A}\mathcal{Q}_1 - \mathcal{Q}_2^T\mathcal{B}\mathcal{Q}_1$ is in generalized Schur form.

Arguments:

Mode Parameters:

- `COMPQ1 (CHARACTER*1)`,
  `COMPQ2 (CHARACTER*1)`: Specify whether or not the orthogonal transformations should be accumulated in the arrays `Q1` and `Q2`, respectively.
  = `'N'`: the corresponding transformation matrix is not computed;
  = `'I'`: the corresponding transformation matrix is computed;
  = `'U'`: the corresponding transformation matrix is computed but multiplied by a given input matrix as described below.

Input/Output Parameters:

- `N (input INTEGER)`: Order of the pencil $\lambda\mathcal{A} - \mathcal{B}$, $\mathtt{N} \geq 0$, even.

- `A (input/output DOUBLE PRECISION array, dimension (LDA, N))`, `B (input/output DOUBLE PRECISION array, dimension (LDB, N))`: On entry, these arrays must contain the matrices $\mathcal{A}$ and $\mathcal{B}$ in (12). The zero (off-)diagonal blocks need not be set to zero. On exit, these arrays contain the transformed upper (quasi-)triangular matrices.

- `Q1 (input/output DOUBLE PRECISION array, dimension (LDQ1, N))`, `Q2 (input/output DOUBLE PRECISION array, dimension (LDQ2, N))`: Optionally, on entry, these arrays must contain given matrices $\mathcal{Q}_{01}$ and $\mathcal{Q}_{02}$, and on exit, these arrays contain the product of the input matrices $\mathcal{Q}_{01}$ and $\mathcal{Q}_{02}$, and the transformation matrices $\mathcal{Q}_1$ and $\mathcal{Q}_2$, respectively, used to transform the matrices $\mathcal{A}$ and $\mathcal{B}$. Optionally, on exit, these arrays contain only the orthogonal transformation matrices $\mathcal{Q}_1$ and $\mathcal{Q}_2$.

3.2.8 Subroutine `DGHUXC` (implements Algorithm 14)

Specification:

```
      SUBROUTINE DGHUXC( COMPQ, N, A, LDA, D, LDD, B, LDB, F, LDF, Q,
     $                   LDQ, NEIG, IWORK, LIWORK, DWORK, LDWORK, INFO )
```

Purpose:

To move the eigenvalues with strictly negative real parts of an `N`-by-`N` real skew-Hamiltonian/Hamiltonian pencil $\lambda\mathcal{S} - \mathcal{H}$ in structured Schur form with

$$\mathcal{S} = \begin{bmatrix} A & D \\ 0 & A^T \end{bmatrix} \quad \text{and} \quad \mathcal{H} = \begin{bmatrix} B & F \\ 0 & -B^T \end{bmatrix}$$

to the leading principal subpencil while keeping the triangular form. Above, A is upper triangular and B upper quasi-triangular. The matrices $\mathcal{S}$ and $\mathcal{H}$ are transformed by an orthogonal matrix $\mathcal{Q}$ such that

$$\begin{aligned} \mathcal{S}_{\text{out}} &= \mathcal{J}\mathcal{Q}^T\mathcal{J}^T\mathcal{S}\mathcal{Q} = \begin{bmatrix} A_{\text{out}} & D_{\text{out}} \\ 0 & A_{\text{out}}^T \end{bmatrix}, \quad \text{and} \\ \mathcal{H}_{\text{out}} &= \mathcal{J}\mathcal{Q}^T\mathcal{J}^T\mathcal{H}\mathcal{Q} = \begin{bmatrix} B_{\text{out}} & F_{\text{out}} \\ 0 & -B_{\text{out}}^T \end{bmatrix}, \end{aligned} \tag{13}$$

where A_{out} is upper triangular and B_{out} is upper quasi-triangular. Optionally, the matrix $\mathcal{Q}$ that fulfills (13) is computed.

Arguments:

Mode Parameters:

- `COMPQ (CHARACTER*1)`: Specifies whether or not the orthogonal transformations should be accumulated in the array `Q`.
  = `'N'`: the transformation matrix is not computed;

= `'I'`: the transformation matrix is computed;
= `'U'`: the transformation matrix is computed but multiplied by a given input matrix as described below.

Input/Output Parameters:

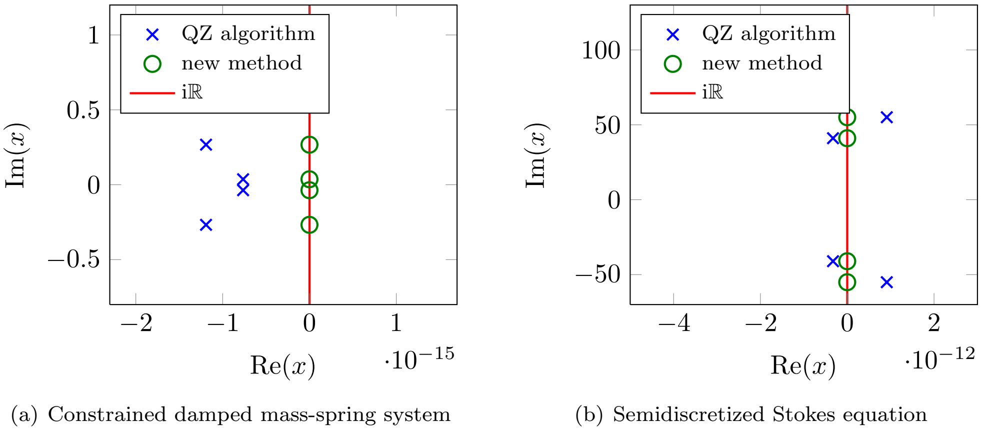
- `N (input INTEGER)`: The order of the pencil $\lambda\mathcal{S} - \mathcal{H}$. $\texttt{N} \geq 0$, even.
- `A (input/output DOUBLE PRECISION array, dimension (LDA, N/2))`, `B (input/output DOUBLE PRECISION array, dimension (LDB, N/2))`: On entry, these arrays must contain the matrices A and B. On exit, these arrays contain the transformed matrices A_{out} and B_{out}, respectively.
- `D (input/output DOUBLE PRECISION array, dimension (LDD, N/2))`, `F (input/output DOUBLE PRECISION array, dimension (LDF, N/2))`: On entry, these arrays must contain the (strictly) upper triangular parts of the matrices D and F. On exit, these arrays contain the transformed (strictly) upper triangular parts of the matrices D_{out} and F_{out}, respectively.
- `Q (input/output DOUBLE PRECISION array, dimension (LDQ, N))`: Optionally, on entry, this array must contain a given matrix $\mathcal{Q}_0$, and on exit, this array contains the product of the input matrix $\mathcal{Q}_0$ and the transformation matrix $\mathcal{Q}$ used to transform the matrices $\mathcal{S}$ and $\mathcal{H}$. Optionally, on exit, this array contains only the orthogonal transformation matrix $\mathcal{Q}$.
- `NEIG (output INTEGER)`: The number of eigenvalues in $\lambda\mathcal{S} - \mathcal{H}$ with strictly negative real part.

4 Numerical Results

In this section we present some numerical results of our implementations. The tests have been performed on a 2.6.32-31-generic Ubuntu machine with Intel®Core$^{\text{TM}}$2 Quad CPU Q9550 with 2.83GHz per core and 8GB RAM. All codes have been compiled using `gfortran` with the optimization level `-O2` (safe optimizations). For better handling the codes, MEX gateway functions have been written for calling the routines from MATLAB 7.14.0.739 (R2012a). For this purpose we also use MATLAB's optimized LAPACK and BLAS libraries.

4.1 Structure-Preserving Computations

The most important feature of our algorithms is structure-preservation. This means that only reductions that keep the skew-Hamiltonian/Hamiltonian structure are performed. Therefore, only skew-Hamiltonian/Hamiltonian perturbations of the eigenvalues are possible. In particular, simple, finite, purely imaginary eigenvalues stay on the imaginary axis as long as their pairwise distance is large enough. In such a situation the perturbation off the imaginary axis would not lead to the formation of



(a) Constrained damped mass-spring system

(b) Semidiscretized Stokes equation

Figure 7: Computed purely imaginary eigenvalues of two skew-Hamiltonian/Hamiltonian example matrix pencils

a quadruple of eigenvalues which is necessary by the Hamiltonian eigensymmetry. In Figure 7, some of the computed eigenvalues by the QZ algorithm ([8]) and our new method are depicted. For the tests we used extended skew-Hamiltonian/Hamiltonian pencils for the $\mathcal{L}_\infty$-norm computation of descriptor systems ([12]). The pencils are related to models for constrained mass-spring systems or semidiscretized Stokes equations (see [10] and references therein). The figure nicely shows that the eigenvalues computed by the standard QZ algorithm are perturbed off the imaginary axis whereas the new method preserves the eigenvalue symmetry. In particular, the new approach allows a reliable determination of the stable eigenvalues. If we furthermore want to compute the stable deflating subspaces we have to know these in advance. For the first presented examples (Figure 7(a)), the QZ algorithm computes more stable than unstable eigenvalues which is impossible by theory. Therefore, also the stable deflating subspace computed by this method will have a too high dimension. This undesired behavior is avoided by our method.

A second example that illustrates the superiority of our method arises in the context of gyroscopic systems of the form

$$M\ddot{x}(t) + G\dot{x}(t) + Kx(t) = 0 \tag{14}$$

with $M = M^T > 0$, $G = -G^T$, and $K = K^T$. To analyse stability of such a system we have to consider the quadratic eigenvalue problem

$$\left(M\lambda^2 + G\lambda + K\right) y = 0. \tag{15}$$

It can be shown that a necessary condition for (14) to be stable is that all eigenvalues of (15) are purely imaginary [9]. A linearization of (15) to second companion form [11]

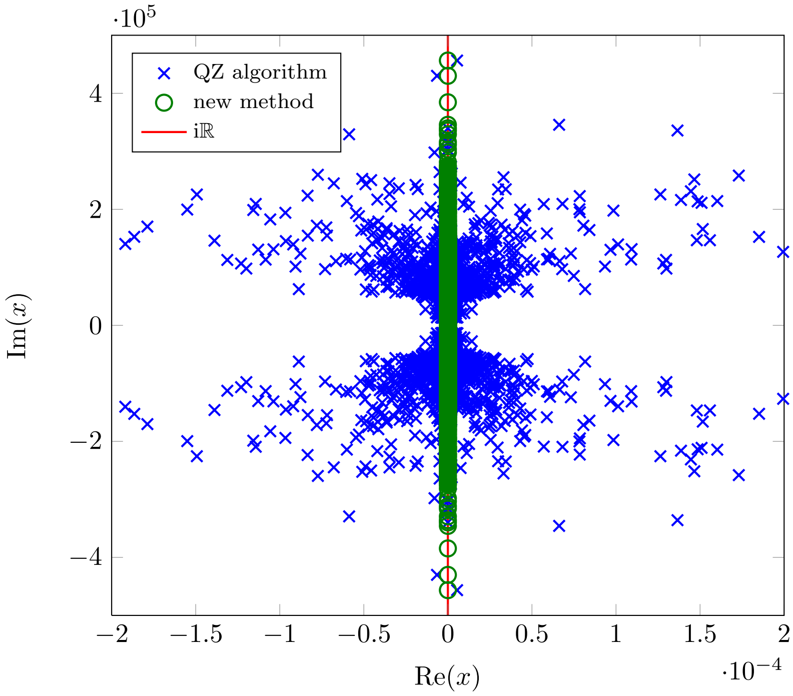


Figure 8: Computed eigenvalues from a skew-Hamiltonian/Hamiltonian matrix pencil resulting from a linearized gyroscopic system

leads to an eigenvalue problem for the skew-Hamiltonian/Hamiltonian matrix pencil

$$\lambda \begin{bmatrix} M & G \\ 0 & M \end{bmatrix} - \begin{bmatrix} 0 & -K \\ M & 0 \end{bmatrix}.$$

The example we use here is the "Rolling Tires" system from [6] with a system dimension of $n = 2697$. The computed eigenvalues for both the QZ algorithm and our method are depicted in Figure 8. For our algorithm, all eigenvalues are determined to be on the imaginary axis which means that the necessary stability criterion for the gyroscopic system is fulfilled. However, for the QZ algorithm this is not the case. Since the QZ algorithm does not respect the skew-Hamiltonion/Hamiltonian structure, all eigenvalues are perturbed off the imaginary axis. Some of them are also very far away from the imaginary axis (the maximum absolute value of the real parts is 1.4836e-03). So in contrast to the structure-preserving approach, one could think that the necessary stability criterion is not fulfilled.

4.2 Solving Algebraic Riccati Equations

In this subsection we use our algorithms for computing the solution of algebraic Riccati equations and compare with the results of the MATLAB function `care`. We consider continuous-time algebraic Riccati equations of the form

$$0 = Q + A^T X + XA - XGX, \tag{16}$$

where A, G, Q, $X \in \mathbb{R}^{n\times n}$. In many problems the matrices $Q = Q^T$ and $G = G^T$ are given in factored form $Q = C^T \tilde{Q} C$, $G = BR^{-1}B^T$ with $C \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{n\times m}$, $\tilde{Q} = \tilde{Q}^T \in \mathbb{R}^{p\times p}$, and $R = R^T \in \mathbb{R}^{m\times m}$. If $\tilde{Q} \geq 0$, $R > 0$, (A, B) is stabilizable, and (A, C) is detectable, then (16) has a unique, positive semidefinite symmetric, stabilizing solution X_*.

A popular method for determining X_* is to compute the stable invariant subspace spanned by $\begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$ of the Hamiltonian matrix

$$\mathcal{H} = \begin{bmatrix} A & -G \\ -Q & -A^T \end{bmatrix} = \begin{bmatrix} A & -BR^{-1}B^T \\ -C^T\tilde{Q}C & -A^T \end{bmatrix} \in \mathbb{R}^{2n\times 2n}.$$

If U_1 is invertible, then $X_* = U_2 U_1^{-1}$ (see [3] and references therein). Here, we use a slightly more general approach, namely we compute the right stable deflating subspace of the skew-Hamiltonian/Hamiltonian matrix pencil

$$\lambda\mathcal{S} - \mathcal{H} = \lambda \begin{bmatrix} I_n & 0 \\ 0 & I_n \end{bmatrix} - \begin{bmatrix} A & -G \\ -Q & -A^T \end{bmatrix} \in \mathbb{R}^{2n\times 2n}$$

which is equal to the stable invariant subspace of $\mathcal{H}$.

For benchmarking we use the examples collected in [3] which are often difficult to solve due to ill-conditioning of the problem or the solution. In Table 2 the relative residuals for each individual problem are presented. We compared the skew-Hamiltonian/Hamiltonian pencil approach with orthogonalization via pivoted QR factorization (QRP), singular value decomposition (SVD) and the MATLAB solver (`care`). To ensure comparability we use the same scaling technique for the ARE for both our codes and `care` (by calling `arescale` in MATLAB). Except for one example (which also `care` could not solve), our codes could compute X_* in all tests. The relative residuals are most often of the same order of magnitude. For five problems, our codes obtained better results for at least one orthogonalization option (for tests # 5, 13, 14, 31, 32 the relative residual is at least one order of magnitude lower than the one of `care`). On the other hand, `care` performed better for 6 examples (# 8, 12, 15, 17, 24, 34). In particular, for example 17, the difference is about 10 orders of magnitude, for the other five examples the difference is about one order of magnitude. Similar results are achieved when having a look at the relative errors compared to the analytic solution if it is known. We omit it since it does not give significantly more information. In conclusion we can say hat both approaches give results of similar quality, even though our codes are not specifically designed for solving algebraic Riccati equations.

Table 2: Relative residuals of the solution of algebraic Riccati equations: Comparison of the new algorithm with orthogonalization via pivoted QR factorization (QRP), singular value decomposition (SVD), and the MATLAB solver `care`

test #	ex. #	n	m	p	parameters	QRP	SVD	`care`
1	1.1	2	1	2		3.0044e-15	2.5749e-15	3.0062e-15
2	1.2	2	1	2		7.3931e-16	3.4594e-15	6.5338e-16
3	1.3	4	2	4		2.4751e-15	2.4167e-15	3.9430e-15
4	1.4	8	2	8		2.5514e-15	1.5739e-15	1.1924e-15
5	1.5	9	3	9		**8.7957e-15**	2.3342e-13	9.3663e-14
6	1.6	30	3	5		8.8269e-12	4.4861e-12	1.4481e-12
7	2.1	2	1	1	$\varepsilon = 1$	9.0528e-16	9.5989e-16	7.5037e-16
8	2.1	2	1	1	$\varepsilon = 10^{-6}$	1.7361e-10	3.2218e-10	**0**
9	2.2	2	2	1	$\varepsilon = 1$	5.5948e-16	3.7261e-16	1.1068e-15
10	2.2	2	2	1	$\varepsilon = 10^{-8}$	1.5895e-09	7.7370e-10	2.3218e-09
11	2.3	2	1	2	$\varepsilon = 1$	7.3951e-16	1.4259e-15	1.1378e-15
12	2.3	2	1	2	$\varepsilon = 10^{6}$	2.0448e-10	3.7537e-11	**6.5854e-13**
13	2.3	2	1	2	$\varepsilon = 10^{-6}$	**1.6745e-21**	4.6784e-18	6.8373e-20
14	2.4	2	2	2	$\varepsilon = 1$	**0**	1.2684e-14	1.1531e-15
15	2.4	2	2	2	$\varepsilon = 10^{-7}$	2.9441e-15	1.1608e-14	**1.6454e-16**
16	2.5	2	1	2	$\varepsilon = 1$	1.4121e-15	1.3570e-15	1.9343e-15
17	2.5	2	1	2	$\varepsilon = 0$	3.6694e-05	1.2326e-06	**1.2232e-15**
18	2.6	3	3	3	$\varepsilon = 1$	5.8902e-15	3.8570e-15	5.7262e-15
19	2.6	3	3	3	$\varepsilon = 10^{6}$	4.7596e+02	4.4341e+02	6.3670e+02
20	2.7	4	1	2	$\varepsilon = 1$	2.4085e-16	1.6736e-16	1.4054e-15
21	2.7	4	1	2	$\varepsilon = 10^{-6}$	1.9697e-08	3.2989e-11	1.3429e-11
22	2.8	4	1	1	$\varepsilon = 1$	7.4186e-16	4.0395e-15	5.6954e-15
23	2.8	4	1	1	$\varepsilon = 10^{-6}$	3.8032e-15	1.0134e-15	4.6214e-15
24	2.9	55	2	10	#1	1.0737e-11	5.7755e-12	**2.4757e-13**
25	3.1	9	5	4		3.8305e-15	2.6481e-15	3.2909e-15
26	3.1	39	20	19		3.4076e-15	4.6692e-15	8.0452e-15
27	3.2	8	8	8		2.9567e-15	2.2579e-15	3.7270e-15
28	3.2	64	64	64		9.8352e-15	8.8604e-15	1.2277e-14
29	4.1	21	1	1	$q = r = 1.0$	1.0359e-06	4.4380e-07	6.8088e-07
30	4.1	21	1	1	$q = r = 100.0$	2.1010e-05	2.1627e-05	6.3995e-05
31	4.2	20	1	1	$a = 0.05,\ b = c = 0.1,$ $[\beta_1, \beta_2] = [0.1, 0.5],$ $[\gamma_1, \gamma_2] = [0.1, 0.5]$	**1.4274e-17**	1.1291e-13	1.8773e-13
32	4.2	100	1	1	$a = 0.01,\ b = c = 1.0,$ $[\beta_1, \beta_2] = [0.2, 0.3],$ $[\gamma_1, \gamma_2] = [0.2, 0.3]$	**1.3742e-15**	1.2528e-12	3.5524e-12
33	4.3	60	2	60	$\ell = 30,\ \mu = 4.0,$ $\delta = 4.0,\ \kappa = 1.0$	7.8279e-15	7.9629e-15	2.6545e-14
34	4.4	421	211	211		5.1450e-03	1.0845e-05	**7.9411e-07**

Table 3: Comparison of runtimes for the real case (measured in secs.)

Problem size	eigenvalues only		eigenvalues and deflating subspaces	
	DGGEV	DGHUTR	DGGES	DGHUDF
2	3.2480e-06	2.2000e-06	7.7860e-06	2.1138e-05
4	1.1510e-05	1.3725e-05	4.3221e-05	1.1633e-04
8	3.7886e-05	7.3677e-05	1.4393e-04	3.6971e-04
16	1.2640e-04	1.9300e-04	3.6360e-04	1.3859e-03
32	7.1310e-04	7.3620e-04	1.7058e-03	5.0400e-03
64	3.0412e-03	3.0708e-03	8.3355e-03	2.5425e-02
128	1.8980e-02	1.6620e-02	4.3790e-02	1.1256e-01
256	1.4190e-01	1.0272e-01	2.8654e-01	5.8121e-01
512	1.4790e+00	8.9793e-01	2.5960e+00	3.9449e+00
1024	2.2127e+01	1.2964e+01	4.8888e+01	4.5998e+01
2048	4.2508e+02	2.6144e+02	5.6186e+02	6.3338e+02
4096	2.9650e+03	2.8367e+03	4.2058e+03	5.5788e+03

4.3 Comparison of Runtimes

In this subsection we discuss the runtimes of our codes and compare them with standard implementations included in LAPACK. The results are listed in Tables 3 and 4, respectively. In Figure 4.3 the speedup factors of the new codes compared to MATLAB's LAPACK implementations are depicted to summarize these results. In general, pure eigenvalue computations are much faster than the computation of both eigenvalues and deflating subspaces. The reason is that for the subspace computation the transformation matrices for the embedded pencils (*of double size*) are accumulated in the final step. However, during our tests we often observe that LAPACK routines, even though they are faster, are not able to solve (random) examples. Especially, for larger problems `INFO = N+2` is returned which indicates that the desired reordering of the eigenvalue could not be successfully performed. Note that LAPACK routines can much better exploit blocked codes of Level 3 BLAS which is not the case for our codes since they are algorithmically based on Givens rotations. Even though the panel blocking technique we present here gives some improvements for larger examples there is still the question whether one can find better ways of blocking our codes.

There are also significant differences in the behavior of the real and complex codes. The real codes have relatively constant speedup factors for small and medium-size problems up to orders of about 128. Then, the speedup factors increase up to order 2048 and then decrease again. However, for the complex codes, the speedups are constant for problems up to order 256 and get significantly slower for larger problems. Fortunately, for larger problems, we have developed blocked codes which are able to avoid this slow-down, see also Subsection 4.5.

Table 4: Comparison of runtimes for the complex case (measured in secs.)

Problem size	eigenvalues only		eigenvalues and deflating subspaces	
	ZGGEV	DGHUST	ZGGES	ZGHUDF
2	7.7400e-06	4.7300e-06	2.7047e-05	4.1847e-05
4	2.3252e-05	2.2831e-05	5.2271e-05	9.3827e-05
8	8.2346e-05	7.5673e-05	1.2291e-04	2.2438e-04
16	3.1020e-04	3.1190e-04	4.6900e-04	7.7210e-04
32	1.4953e-03	1.4844e-03	2.5219e-03	3.4171e-03
64	8.7930e-03	9.0812e-03	1.4392e-02	1.9041e-02
128	5.8440e-02	5.6700e-02	9.2550e-02	1.1988e-01
256	4.5301e-01	4.5600e-01	6.2856e-01	9.6518e-01
512	3.4875e+00	7.6826e+00	4.6978e+00	1.4286e+01
1024	3.8185e+01	1.4554e+02	5.6904e+01	2.6081e+02
2048	4.9624e+02	1.2935e+03	8.2872e+02	2.1489e+03
4096	4.8410e+03	1.0849e+04	7.7507e+03	1.7189e+04

4.4 Factored Versus Unfactored Matrix Pencils

In this subsection we compare the results of the previous subsection with the factored versions of the algorithms with respect to accuracy, memory requirements and speed.

4.4.1 Accuracy

We begin with an analysis of the obtained accuracy. We performed tests on random skew-Hamiltonian/Hamiltonian pencils of order 40. For the factored algorithms we

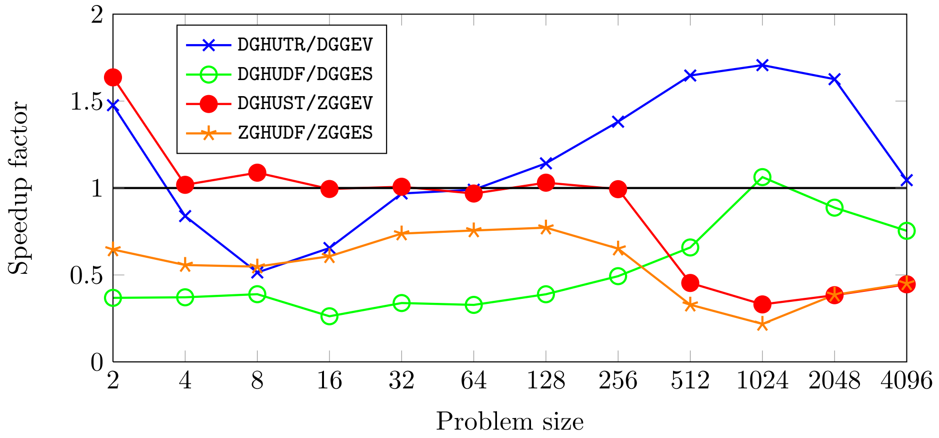


Figure 9: Speedup factors of the new routines compared to LAPACK software

Table 5: Comparison of the errors of the eigenvalues

	real case		complex case	
	unfactored	factored	unfactored	factored
$10^{-17} \leq \kappa_{\max} < 10^{-16}$	0	0	29	44
$10^{-18} \leq \kappa_{\max} < 10^{-17}$	825	805	932	926
$10^{-19} \leq \kappa_{\max} < 10^{-18}$	155	162	39	30
$10^{-20} \leq \kappa_{\max} < 10^{-19}$	6	17	0	0
$\kappa_{\max} < 10^{-20}$	14	16	0	0

Table 6: Comparison of the errors of the deflating subspaces

	real case		complex case	
	unfactored	factored	unfactored	factored
$10^{-11} \leq \alpha < 10^{-10}$	1	0	0	0
$10^{-12} \leq \alpha < 10^{-11}$	9	11	0	3
$10^{-13} \leq \alpha < 10^{-12}$	82	96	38	62
$10^{-14} \leq \alpha < 10^{-13}$	900	888	962	935
$10^{-15} \leq \alpha < 10^{-14}$	8	5	0	0

choose $\mathcal{Z} = \begin{bmatrix} A & 0 \\ 0 & I_{20} \end{bmatrix}$ with a random matrix A. Then we can easily form

$$\mathcal{S} = \mathcal{J}\mathcal{Z}^H\mathcal{J}^T\mathcal{Z} = \begin{bmatrix} A & 0 \\ 0 & A^H \end{bmatrix}$$

without any rounding error. This allows a fair comparison between the codes for factored and unfactored problems. First we analyze the accuracy of the computed eigenvalues. Therefore, we performed 1000 tests and compute the maximum of the reciprocal condition numbers $\kappa_{\max}$ of the matrices $\lambda_j\mathcal{S} - \mathcal{H}$, $j = 1, \ldots, 20$ for each problem. We divide the computed results into different classes and list the number of elements in each class in Table 5. Furthermore, we observe that in the real case, the unfactored codes were more accurate for 500 examples. For the complex case this was the case for 516 examples. We can conclude that the computed eigenvalues are similarly accurate for both types of codes.

We also have a look at the accuracy of the deflating subspaces. Let colspan $\mathcal{Q}$ be the computed stable deflating subspace. To measure the error we determine the angle α between the subspaces colspan$(\mathcal{S}\mathcal{Q})$ and colspan$(\mathcal{H}\mathcal{Q})$. Again, we perform 1000 tests and divide the results into classes listed in Table 6. Now, the unfactored version is more accurate for 615 examples in the real and for 592 examples in the complex case, respectively. Therefore, we can conclude that the subspace computation is slightly more accurate in the unfactored case.

Table 7: Comparison of runtimes for factored and unfactored versions (measured in secs.)

Problem size	real case		complex case	
	unfactored	factored	unfactored	factored
2	2.1149e-05	3.7359e-05	4.1219e-05	7.4963e-05
4	5.3765e-05	9.4412e-05	8.9305e-05	1.8112e-04
8	3.6373e-04	5.1282e-04	2.3065e-04	4.2514e-04
16	1.4868e-03	1.8846e-03	7.5680e-04	1.1702e-03
32	5.9223e-03	7.8657e-03	3.2732e-03	5.6365e-03
64	2.3258e-02	3.1986e-02	1.8261e-02	2.8526e-02
128	1.0901e-01	1.4402e-01	1.1473e-01	1.9216e-01
256	5.7424e-01	7.9756e-01	9.2289e-01	1.6150e+00
512	3.8463e+00	6.1073e+00	1.4380e+01	3.3246e+01
1024	4.6299e+01	1.0119e+02	2.5326e+02	4.1394e+02
2048	6.0400e+02	9.5667e+02	2.0491e+03	3.5848e+03
4096	5.4444e+03	7.9957e+03	1.6688e+04	2.8164e+04

4.4.2 Speed and Memory Requirements

We briefly compare the timing results of the factored and unfactored codes which are listed in Table 7. A run of the factored versions needs approximately 1.5 – 2 times as long as one of the unfactored versions. This is simply due to fact that also more matrices (usually ≈ 50% more) have to be updated within the factored codes. Also this higher amount of matrices has to be stored which leads to an approximately 50% higher memory usage.

4.4.3 Conclusion

In conclusion, we can say that one should always use the unfactored version of the code whenever the matrix $\mathcal{S}$ is explicitly given or can be formed without any rounding errors. This is due to the lower accuracy, larger runtimes and higher memory usage of the factored versions. However, one might think of situations where only the factor $\mathcal{Z}$ is known and it is not possible to appropriately form $\mathcal{S}$ due to numerical errors. Then we still recommend to use the factored versions even if there are all the disadvantages mentioned above.

4.5 Blocked Versus Unblocked Code

As already mentioned above, the routines get relatively slow if the problem gets too large. This is due to the unoptimized cache usage. Therefore, we have implemented the unfactorized algorithms using the panel blocking technique from Subsection 2.3. For illustration we generated a random example of order 2048 and compared the runtimes of the unblocked code with those of the blocked code for different block sizes

Table 8: Comparison of runtimes for blocked and unblocked code (measured in secs.)

block size NB	eigenvalues only		eigenvalues and deflating subspaces	
	real case	complex case	real case	complex case
unblocked	224.23	1332.72	582.65	2173.03
1	200.91	922.11	546.95	1644.15
2	147.87	738.48	517.41	1458.51
4	133.90	662.78	483.16	1365.91
8	**132.31**	**657.39**	466.02	**1345.67**
16	146.86	680.06	470.36	1383.82
32	139.47	680.46	469.58	1363.19
64	138.72	672.18	469.72	1376.00
128	139.44	700.46	**459.48**	1400.46
256	139.62	1024.32	468.70	1775.13
512	151.18	1116.98	482.61	1844.09
1024	216.94	1165.43	548.82	1955.30
2048	210.05	1244.88	555.28	2025.61

NB. The results can be found in Table 8. The optimal values are marked in boldface font. The time savings can be significant. For computing the eigenvalues of a complex pencil the reduction can be to less than 50% of the time needed for the unblocked code. Note that there is only a slight speedup for the subspace computation in the real case since the time-consuming routine DGHUYR cannot be blocked. Mostly, the optimal timings are attained for $\texttt{NB} = 8$, however almost optimal timings are observed for all $\texttt{NB} = 4, \ldots, 128$, so the choice of NB is flexible. An important point is that the problem must be sufficiently large in order to benefit from the panel blocking, otherwise one would even loose performance, especially for small block sizes. Finally, we also compare the performance of our blocked codes for $\texttt{NB} = 8$ with LAPACK for problem sizes of 512 to 4096. The speedup factors are depicted in Figure 4.5. For real problems we can achieve good speedups compared to LAPACK. When computing only the eigenvalues we can achieve a speedup factor of about 3.5. When we also compute the deflating subspaces we still get a factor of 1.2, so we are still faster than LAPACK. However, this is not the case anymore when we consider complex problems. In this case we can only achieve speedups of about 0.4 to 0.9, but the blocked codes are still faster than the unblocked ones.

5 Summary

In this paper we have presented implementation details and interface descriptions for structure-preserving algorithms for the computation of the eigenvalues and stable deflating subspaces of skew-Hamiltonian/Hamiltonian matrix pencils in FORTRAN 77. The advantages of our method are the increased reliability since critical purely imagi-

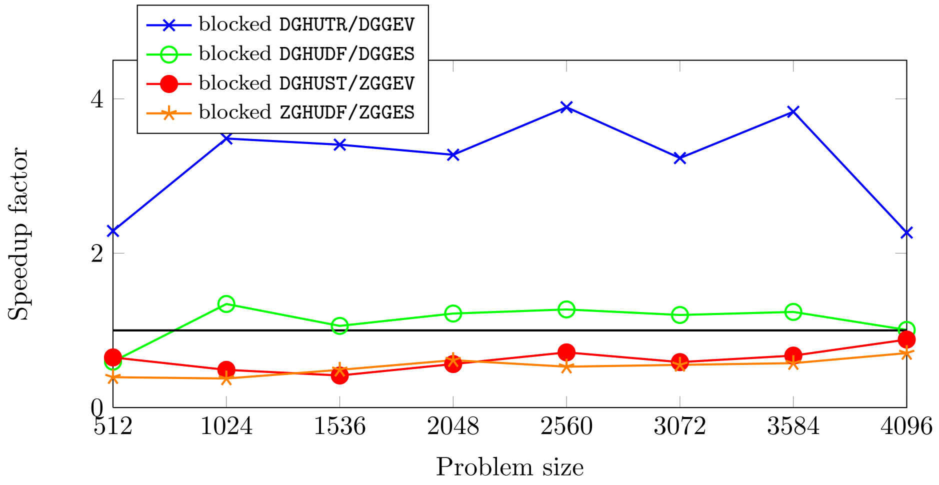


Figure 10: Speedup factors of the blocked codes compared to LAPACK software for larger problem sizes

nary eigenvalues are not perturbed from the imaginary axis (as long as their pairwise distance is large enough). This also allows the safe computation of the associated stable deflating subspaces of the pencil since the perturbation of eigenvalues from the left into the right half-plane (or vice versa) is avoided. Numerical examples have shown that the runtimes are often higher compared to LAPACK routines. However, a panel blocking technique has significantly improved performance for larger problems.

References

[1] *SLICOT Contributors' Kit 2.1*, August 1996. WGS Report 96-2.

[2] *SLICOT Implementation and Documentation Standards*, August 1996. WGS Report 96-1.

[3] J. Abels and P. Benner. CAREX - A collection of benchmark examples for continuous-time algebraic Riccati equations (version 2.0), 1999.

[4] P. Benner, R. Byers, and E. Barth. Algorithm 800: Fortran 77 subroutines for computing the eigenvalues of Hamiltonian matrices I: the square-reduced method. *ACM Trans. Math. Softw.*, 26(1):49–77, 2000.

[5] P. Benner, R. Byers, V. Mehrmann, and H. Xu. Numerical computation of deflating subspaces of skew-Hamiltonian/Hamiltonian pencils. *SIAM J. Matrix Anal. Appl.*, 24(1):165–190, 2001.

[6] P. Benner, H. Fassbender, and M. Stoll. Solving large-scale quadratic eigenvalue problems with Hamiltonian eigenstructure using a structure-preserving Krylov subspace method. *Electron. Trans. Numer. Anal.*, 29:212–229, 2008.

[7] P. Benner, V. Sima, and M. Voigt. FORTRAN 77 subroutines for the solution of skew-Hamiltonian/Hamiltonian eigenproblems – Part I: Algorithms and applications. Preprint MPIMD/13-11, Max Planck Institute Magdeburg, 2013.

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The John Hopkins University Press, Baltimore/London, third edition, 1996.

[9] P. Lancaster. Stability of linear gyroscopic systems: A review. *Linear Algebra Appl.*, 439(3):686–706, August 2013. Special Issue in Honor of Harm Bart.

[10] V. Mehrmann and T. Stykel. Balanced truncation model reduction for large-scale systems in descriptor form. In P. Benner, V. Mehrmann, and D. Sorensen, editors, *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lect. Notes Comput. Sci. Eng.*, chapter 3, pages 89–116. Springer-Verlag, Berlin, Heidelberg, New York, 2005.

[11] F. Tisseur and K. Meerbergen. The quadratic eigenvalue problem. *SIAM Rev.*, 43(2):235–286, 2001.

[12] M. Voigt. $\mathcal{L}_\infty$-norm computation for descriptor systems. Diploma thesis, Chemnitz University of Technology, Faculty of Mathematics, Germany, July 2010.

Max Planck Institute Magdeburg Preprints