A Web Service for Remote Model Reduction of Large-Scale Linear Systems on a Cluster

José M. Badía*, Peter Benner**, Maribel Castillo*, Rafael Mayo*,
Enrique S. Quintana-Ortí*, Gregorio Quintana-Ortí*

Resumen— **This paper describes a library for model reduction of large-scale, dense linear time-invariant systems on parallel distributed-memory computers. The library is enhanced with a Web service which allows remote submission of jobs via any standard Web browser, and serves as a demonstrator of the capabilities of the library.**

Our experimental results on a cluster composed of 32 Intel Pentium-II nodes connected via a Myrinet switch show the performance of the parallel kernels of the library.

Palabras clave— **Model reduction, linear control systems, Lyapunov equations, web interfaces, grid/web computing.**

I. Introduction

The development of the Internet and its popularization via the world wide web (www) is producing a tremendous impact in our society. The www is daily used to consult databases, retrieve documents, submit jobs to remote systems, and a large number of services which is continuously growing. The extension of these services has recently reached the area of scientific computing in projects like NetSolve [1], LAPK (`http://lapk3.hsc.usc.edu/lapk`), and the SLICOT Web Environment [10], among many.

In this paper we further explore the www as a means of offering an enhanced service for parallel model reduction to a wide range of the scientific community. The advantage of providing this service via the www is that the end-user remains completely isolated from the complexities of the software and the hardware system, and the maintenance of both. A secondary goal of this service is to serve as a demonstrator of the capabilities of the parallel model reduction library, which can be then downloaded and installed on any (parallel) architecture with MPI.

The problem considered here, model reduction, is of fundamental importance in many modeling and control applications involving linear time-invariant (LTI) systems. In state-space form, such systems are described by the following models:

Continuous LTI system:

$$\begin{array}{rcll} \dot{x}(t) & = & Ax(t)+Bu(t), & t>0, \quad x(0)=x^0, \\ y(t) & = & Cx(t)+Du(t), & t\geq 0. \end{array} \tag{1}$$

Discrete LTI system:

$$\begin{array}{rcll} x_{k+1} & = & Ax_k+Bu_k, & x_0=x^0, \\ y_k & = & Cx_k+Du_k, & k=0,1,2,\ldots. \end{array} \tag{2}$$

In both models, $A \in \mathbb{R}^{n\times n}$ is the state matrix, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$, and $x^0 \in \mathbb{R}^n$ is the initial state of the system. Also, n is known as the order of the system, and the associated transfer function matrix (TFM) is $G(s) = C(sI-A)^{-1}B+D$. Here we assume that the state matrix A is stable. In the continuous-time case, this implies that the spectrum of A is contained in the open left half plane while, in the discrete-time case, the spectrum of A must lie inside the unit circle. We do not require, however, minimality of the system.

Specifically, the model reduction problem consists in finding a reduced-order LTI system,

Continuous LTI system:

$$\begin{array}{rcll} \dot{\hat{x}}(t) & = & \hat{A}\hat{x}(t)+\hat{B}\hat{u}(t), & t>0 \quad \hat{x}(0)=\hat{x}^0, \\ \hat{y}(t) & = & \hat{C}\hat{x}(t)+\hat{D}\hat{u}(t), & t\geq 0, \end{array} \tag{3}$$

Discrete LTI system:

$$\begin{array}{rcll} \hat{x}_{k+1} & = & \hat{A}\hat{x}_k+\hat{B}\hat{u}_k, & \hat{x}_0=\hat{x}^0, \\ \hat{y}_k & = & \hat{C}\hat{x}_k+\hat{D}\hat{u}_k, & k=0,1,2,\ldots, \end{array} \tag{4}$$

of order r, $r \ll n$, and associated TFM $\hat{G}(s) = \hat{C}(sI-\hat{A})^{-1}\hat{B}+\hat{D}$ which approximates $G(s)$.

Model reduction of large-scale systems arises, among others, in control of large flexible mechanical structures or large power systems as well as in circuit simulation and VLSI design; see, e.g., [7], [8], [11], [17]. LTI systems with state-space dimension n of order 10^2 to 10^4 are common in these applications.

In general, model reduction methods for LTI systems with dense state matrices have a computational cost of $\mathcal{O}(n^3)$ flops (floating-point arithmetic operations) and require storage for $\mathcal{O}(n^2)$ numbers. While current desktop computers provide enough computational power to deal with systems of order n in the hundreds (using libraries like SLICOT, available from `http://www.win.tue.nl/niconet/NIC2/slicot.html`.), large-scale applications clearly require the use of parallel computing techniques to obtain the reduced system. The usual approach a few decades ago was to solve these problems on very expensive parallel supercomputers, but the situation is changing in the last years. Clusters constructed from commodity systems (personal computers and local and system area switches) and "open" software have started to be widely used for

*Depto. de Ingeniería y Ciencia de Computadores, Universidad Jaume I, 12071–Castellón, Spain; {badia,castillo,mayo,quintana,gquintan}@icc.uji.es.
Supported by the Fundación Caixa-Castelló/Bancaixa PI-1B2001-14.
**Institut für Mathematik, Technische Universität Berlin, D-10623 Berlin, Germany; benner@math.tu-berlin.de.

parallel scientific computing. Notable improvements in the performance of commodity hardware have resulted in affordable tools for dealing with large-scale scientific problems. However, many end-users (control engineers, VLSI circuit designers, etc.) are not familiar with the basics of parallel algorithms and/or parallel architectures. Thus, if a parallel library of control algorithms is to be useful to a broader spectrum of the scientific community, it should be part of our work to provide, along with the library, the necessary tools to facilitate its use.

Our library provides parallel routines for model reduction using three absolute error methods. These approaches are briefly reviewed in Section II. The contents of the library are described in Section III. The library is completed with a Web service, presented in Section IV, which allows remote access to our parallel cluster and enables the user to apply any of the model reduction algorithms on a specific LTI system. Finally, the performance of the parallel library is reported in Section V, and some concluding remarks are given in Section VI.

II. Methods for Model Reduction

There is no general technique for model reduction that can be considered as optimal in an overall sense since the reliability, performance, and adequacy of the reduced system strongly depends on the system characteristics. The model reduction methods we study here rely on truncated state-space transformations where, given $Y = \left[T_l^T, L_l^T\right]^T \in \mathbb{R}^{n\times n}$ and $Y^{-1} = [T_r, L_r]$, with $T_l \in \mathbb{R}^{r\times n}$ and $T_r \in \mathbb{R}^{n\times r}$,

$$\begin{array}{rclcrcl} \hat{A} & = & T_l A T_r, & & \hat{B} & = & T_l B, \\ \hat{C} & = & C T_r, & \text{and} & \hat{D} & = & D. \end{array} \tag{5}$$

Model reduction methods based on truncated state-space transformations usually differ in the measure they attempt to minimize. Balanced truncation (BT) methods [16], [19], [21], [23], singular perturbation approximation (SPA) methods [15], and Hankel-norm approximation (HNA) methods [12] all belong to the family of absolute error methods, which try to minimize $\|\Delta_a\|_\infty = \|G - \hat{G}\|_\infty$. Here, $\|G\|_\infty$ denotes the $\mathcal{L}_\infty$- or $\mathcal{H}_\infty$-norm of a stable, rational matrix function defined as

$$\|G\|_\infty = \operatorname*{ess\,sup}_{\omega\in\mathbb{R}} \sigma_{\max}(G(\jmath\omega)), \tag{6}$$

where $\jmath := \sqrt{-1}$ and $\sigma_{\max}(M)$ is the largest singular value of the matrix M.

The family of relative error model reduction methods attempt to minimize, on the other hand, the relative error $\|\hat{\Delta}\|_\infty$, defined implicitly by $G - \hat{G} = \hat{\Delta} G$. Among these, the balanced stochastic truncation (BST) methods [9], [13], [24] are specially interesting. The current version of the model reduction library described here does not include BST methods.

We briefly review next the methods for absolute error model reduction. Serial implementations of these algorithms can be found in the *Subroutine Library in Control Theory* – SLICOT. Note that we use implementations that are different from the serial versions in SLICOT but result in the same reduced-order model. Our implementations are particularly efficient on parallel computing platforms.

All these methods are strongly related to the controllability Gramian W_c and the observability Gramian W_o of the LTI system. In the continuous-time case, the Gramians are given by the solutions of two coupled *Lyapunov equations*

$$\begin{array}{rcl} A W_c + W_c A^T + B B^T & = & 0, \\ A^T W_o + W_o A + C^T C & = & 0, \end{array} \tag{7}$$

while, in the discrete-time case, the Gramians are the solutions of two coupled *Stein equations*

$$\begin{array}{rcl} A W_c A^T - W_c + B B^T & = & 0, \\ A^T W_o A - W_o + C^T C & = & 0. \end{array} \tag{8}$$

As A is assumed to be stable, the Gramians W_c and W_o are positive semidefinite, and therefore there exist factorizations $W_c = S^T S$ and $W_o = R^T R$. Matrices S and R are usually called the *Cholesky factors* of the Gramians.

The Lyapunov equations are solved in our model reduction algorithms using the Newton iteration for the matrix sign function introduced by Roberts [18]. The Stein equations are solved using the Smith iteration proposed in [20]. Both iterations are specially adapted taking into account that, for large-scale non-minimal systems, the Cholesky factors are often of low (numerical) rank. By not allowing these factors to become rank-deficient, a large amount of workspace and computational cost can be saved. For a detailed description of this technique, see, e.g., [2], [3].

Consider now the singular value decomposition (SVD)

$$S R^T = [U_1\ U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}, \tag{9}$$

where the matrices are partitioned at a given dimension r such that $\Sigma_1 = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$, $\Sigma_2 = \operatorname{diag}(\sigma_{r+1}, \ldots, \sigma_n)$, $\sigma_j \geq 0$ for all j, and $\sigma_r > \sigma_{r+1}$. Here, $\sigma_1, \ldots, \sigma_n$ are known as the *Hankel singular values* of the system. If $\sigma_r > \sigma_{r+1} = 0$, then r is the *McMillan* degree of the system. That is, r is the state-space dimension of a minimal realization of the system.

The so-called *square-root* (SR) BT algorithms determine the reduced-order model in (5) using the projection matrices

$$T_l = \Sigma_1^{-1/2} V_1^T R \quad \text{and} \quad T_r = S^T U_1 \Sigma_1^{-1/2}. \tag{10}$$

In case $\Sigma_1 > 0$ and $\Sigma_2 = 0$, the reduced-order model computed by using T_l, T_r from (10) in (5) is a minimal realization of the TFM $G(s)$.

The *balancing-free square-root* (BFSR) BT algorithms often provide more accurate reduced-order models in the presence of rounding errors [23]. These

algorithms share the first two stages (solving the coupled equations and computing the SVD of SR^T) with the SR methods, but differ in the procedure to obtain T_l and T_r. Specifically, the following two QR factorizations are computed,

$$S^T U_1 = [P_1\ P_2] \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix}, \qquad R^T V_1 = [Q_1\ Q_2] \begin{bmatrix} \bar{R} \\ 0 \end{bmatrix}, \tag{11}$$

where $P_1,\ Q_1 \in \mathbb{R}^{n\times r}$ have orthonormal columns, and $\hat{R},\ \bar{R} \in \mathbb{R}^{r\times r}$ are upper triangular. The reduced system is then given by the projection matrices

$$T_l = (Q_1^T P_1)^{-1} Q_1^T, \qquad T_r = P_1, \tag{12}$$

and (5).

SR SPA or BFSR SPA reduced-order models can be obtained by first using the SR BT or BFSR BT algorithms, respectively, to obtain a minimal realization of the original system. Let the tuple $(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D})$ be a realization of this minimal system and partition

$$\tilde{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \quad \tilde{C} = [\,C_1\ \ C_2],$$

according to the desired size r of the reduced-order model, i.e., $A_{11} \in \mathbb{R}^{r\times r}$, $B_1 \in \mathbb{R}^{r\times m}$, $C_1 \in \mathbb{R}^{p\times r}$, etc. Then the SPA reduced-order model is obtained by applying the following formulae

$$\begin{array}{rcl} \hat{A} & := & A_{11} - A_{12}(\gamma I - A_{22})^{-1} A_{21}, \\ \hat{B} & := & B_1 - A_{12}(\gamma I - A_{22})^{-1} B_2, \\ \hat{C} & := & C_1 - C_2(\gamma I - A_{22})^{-1} A_{21}, \\ \hat{D} & := & \tilde{D} - C_2(\gamma I - A_{22})^{-1} B_2, \end{array} \tag{13}$$

where $\gamma = 0$ for continuous LTI systems and $\gamma = 1$ for discrete LTI systems [15], [22], [23].

BT and SPA model reduction methods aim at minimizing the $\mathcal{H}_\infty$-norm norm of the error system $G - \hat{G}$. However, they usually do not succeed in finding an optimal approximation. Using the *Hankel norm* of a stable rational transfer function, defined by

$$\|G\|_H = \sigma_1(G), \tag{14}$$

where $\sigma_1(G)$ is the maximum Hankel singular value of G, it is possible to compute an approximation minimizing $\|G - \hat{G}\|_H$ for a given order r of the reduced-order system [12]. The derivation of a realization of $\hat{G}$ is quite involved. Due to space limitations, we refer the reader to [12], [25] and only describe the essential computational tools required in an implementation of the HNA method. In the first step, a balanced minimal realization of G is computed. This can be done using the SR version of the BT method described above. Then a transfer function $\tilde{G}(s) = \tilde{C}(sI - \tilde{A})^{-1}\tilde{B} + \tilde{D}$ is computed as follows: first, the order r of the reduced-order model is chosen such that the Hankel singular values of G satisfy

$$\begin{array}{l} \sigma_1 \geq \sigma_2 \geq \ldots \sigma_r > \sigma_{r+1} = \ldots \\ \qquad = \sigma_{r+k} > \sigma_{r+k+1} \geq \ldots \geq \sigma_n, \quad k \geq 1. \end{array}$$

Then, by applying appropriate permutations, the balanced transformation of G is re-ordered such that the Gramians become

$$\begin{bmatrix} \check{\Sigma} & \\ & \sigma_{r+1} I_k \end{bmatrix}.$$

The resulting balanced realization given by $\check{A}, \check{B}, \check{C}, \check{D}$ is partitioned conformally with the partitioning of the Gramians, i.e.,

$$\check{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad \check{B} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \quad \check{C} = [\,C_1\ \ C_2],$$

where $A_{11} \in \mathbb{R}^{n-k\times n-k}, B_1 \in \mathbb{R}^{n-k\times m}$, $C_1 \in \mathbb{R}^{p\times n-k}$, etc. Then the following formulae define a realization of $\tilde{G}$:

$$\begin{array}{rcl} \tilde{A} & = & \Gamma^{-1}(\sigma_{r+1}^2 A_{11}^T + \check{\Sigma} A_{11} \check{\Sigma} + \sigma_{r+1} C_1^T U B_1^T, \\ \tilde{B} & = & \Gamma^{-1}(\check{\Sigma} B_1 - \sigma_{r+1} C_1^T U), \\ \tilde{C} & = & C_1 \check{\Sigma} - \sigma_{r+1} U B_1^T, \\ \tilde{D} & = & D + \sigma_{r+1} U. \end{array}$$

Here, $U := (C_2^T)^\dagger B_2$, where $M^\dagger$ denotes the pseudoinverse of M, and $\Gamma := \check{\Sigma}^2 - \sigma_{r+1}^2 I_{n-k}$. This step only requires matrix scaling, matrix multiplication, and a QR decomposition. Now we can compute an additive decomposition of $\tilde{G}$ such that $\tilde{G}(s) = \hat{G}(s) + F(s)$ where $\hat{G}$ is stable and F is antistable. Then $\hat{G}$ is an optimal r-th order Hankel norm approximation of G. We have implemented the additive decomposition of $\tilde{G}$ via a block diagonalization of $\tilde{A}$, where we first compute a block Schur form using the sign function of $\tilde{A}$ and then annihilate the off-diagonal block by solving a Sylvester equation using again a sign function-based iterative solution procedure. For further details of the implementation of the HNA method and computational aspects see [5].

III. Parallelization and driver parallel routines in the Library

The absolute error methods basically perform matrix operations such as the solution of linear systems and linear matrix equations (Lyapunov and Sylvester), and the computation of matrix products and matrix decompositions (LU, QR, SVD, etc.). Linear matrix equations can be efficiently solved using iterative algorithms which only require operations like matrix products and matrix inversion. All these operations are basic dense matrix algebra kernels parallelized in ScaLAPACK and PBLAS. Thus, our parallel model reduction library, PLiCMR, heavily relies on the use of the available parallel infrastructure in ScaLAPACK, the serial computational libraries LAPACK and BLAS, and the communication library BLACS (see Figure 1).

Details of the contents and parallelization aspects of PLiCMR are given, e.g., in [3], [4].

The functionality and naming convention of the parallel routines closely follow analogous routines from SLICOT. Three parallel driver routines are provided as part of PLiCMR:

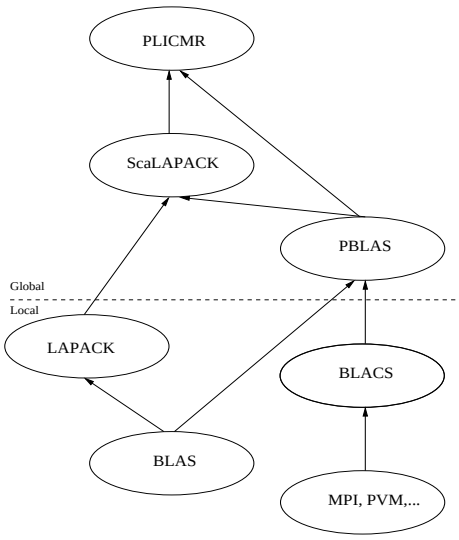

Fig. 1. PLiCMR and the underlying libraries.

- `pab09ax`: SR and BFSR BT method.
- `pab09bx`: SR and BFSR SPA method.
- `pab09cx`: HNA method.

The library also includes several other computational and auxiliary parallel routines.

A standarized version of the library is integrated into the subroutine library PSLICOT [6], with parallel implementations of a subset of SLICOT. It can be downloaded from `ftp://ftp.esat.kuleuven.ac.be/pub/WGS/SLICOT`. The version in `http://spine.act.uji.es/~plicmr` might, at some stages, contain more recent updates than the version integrated into PSLICOT. The library can be installed on any parallel architecture where the above-mentioned computational and communication libraries are available. The efficiency of the parallel routines will depend on the performance of the underlying libraries for matrix computation (BLAS) and communication (usually, MPI or PVM).

IV. Web Service

In this section we describe the access procedure and the structure of the service for parallel model reduction via the www. This service is provided on a parallel cluster composed of 32 Intel Pentium-II personal computers, running Linux operating system, and connected with a 1.2 Gbps Myrinet network. The cluster is owned by the Parallel Scientific Research Group, at the University Jaume I (`http://anna.act.uji.es`).

The user can submit a job through `http://spine.act.uji.es/~plicmr/pmrW3/prmW3s.html` using any standard web browser. This page contains a form that allows the user to specify parameters of the job like the model reduction algorithm to employ, the dimensions of the system, the desired order for the reduced system, the input matrices, etc. (see Figure 2).

When the user submits a job, the System Web Server receives the input data, creates a process, the

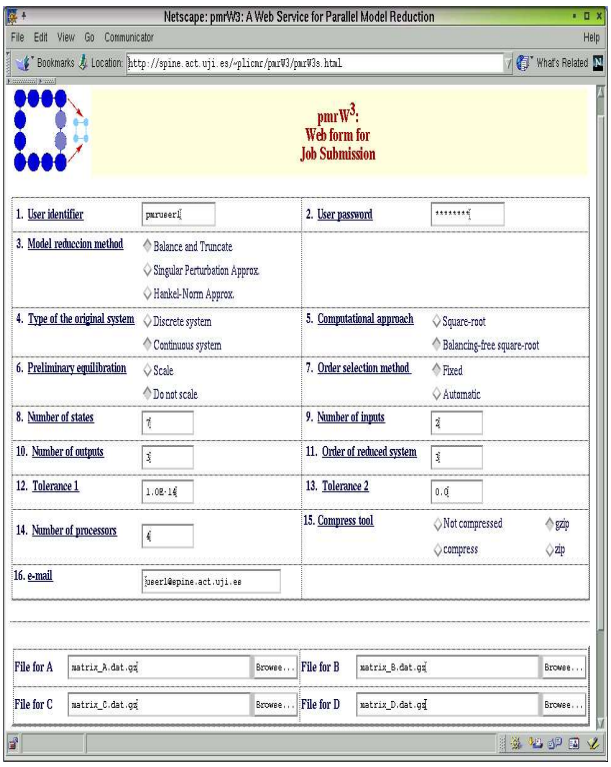

Fig. 2. Web interface.

Web Agent, and passes all data to this process. (In some systems, the Web Agent is really a thread of the System Web Server.)

The Web Agent then executes a PHP program to validate the user/password and the input data provided in the form. On success, the files with the system matrices are uploaded from the user's local file system, and the Web Agent prepares the job submission. This stage basically includes preprocessing and uncompressing (if necessary) the files with the system matrices, preparing a file with the parameters of the system, and creating a shell-script specifying the parameters of the submission (name of paths to the data and parallel model reduction code, number of processors, etc.). This shell-script also includes the instructions that, on termination of the job, will return an e-mail to the user with the matrices of the reduced model (processed with the same compressor the user employed). The Web Agent submits the job using PBS (portable batch system), a standard batch queuing and workload management system. Figure 3 shows the Web service system and the iteractions between its components.

V. Experimental Results

All the experiments presented in this section were performed on Intel Pentium-II processors using IEEE double-precision floating-point arithmetic ($\varepsilon \approx 2.2204 \times 10^{-16}$). The parallel algorithms were evaluated on a parallel distributed cluster of 32 nodes. Each node consists of an Intel Pentium-II processor at 300 MHz, and 128 MBytes of RAM. We employ a BLAS library, specially tuned for the Pentium-II processor as part of the ITXGEMM project [14],

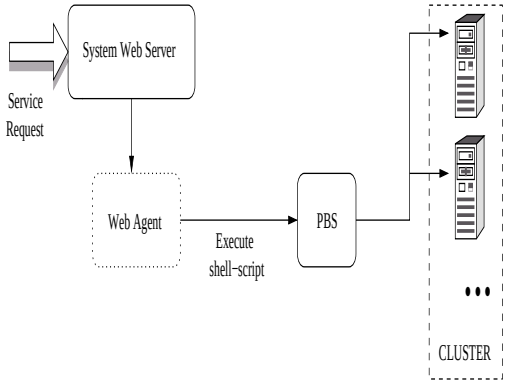

Fig. 3. Components of the Web service system.

that achieves around 180 Mflops (millions of flops per second) for the matrix product (routine `DGEMM`). The nodes are connected via a *Myrinet* crossbar network; the communication library BLACS is based on an implementation of the communication library MPI specially developed and tuned for this network. The performance of the interconnection network was measured by a simple loop-back message transfer resulting in a latency of 33 μsec and a bandwidth of 200 Mbit/sec. We made use of the LAPACK, PBLAS, and ScaLAPACK libraries wherever possible.

We only report results using the BFSR BT algorithms for a continuous LTI system. Similar results were obtained for the SPA and HNA algorithms and for discrete LTI systems.

We analyze the performance of our parallel model reduction algorithms using a random continuous LTI system constructed as follows. First, we generate a random positive semidefinite diagonal Gramian $W_c = \mathrm{diag}\,(\Sigma_{q_1}, \Sigma_{q_2}, 0_{q_3}, 0_{q_4})$, where $\Sigma_{q_1} \in \mathbb{R}^{q_1 \times q_1}$ contains the desired Hankel singular values for the system and $\Sigma_{q_2} \in \mathbb{R}^{q_2 \times q_2}$. Then, we construct a random positive semidefinite diagonal Gramian $W_o = \mathrm{diag}\,(\Sigma_{q_1}, 0_{q_2}, \Sigma_{q_3}, 0_{q_4})$, with $\Sigma_{q_3} \in \mathbb{R}^{q_3 \times q_3}$. Next, we set A to a random stable diagonal matrix and compute $F = -(AW_c + W_cA^T)$ and $G = -(A^TW_o + W_oA)$. Thus,

$$\begin{aligned} F &= \mathrm{diag}\,(f_1, f_2, \ldots, f_{q_1+q_2}, 0_{q_3+q_4}), \\ G &= \mathrm{diag}(g_1, g_2, \ldots, g_{q_1}, 0, \ldots, \\ &\qquad 0, g_{q_1+q_2+1}, \ldots, g_{q_1+q_2+q_3}, 0_{q_4}). \end{aligned}$$

A matrix $B \in \mathbb{R}^{n \times (q_1+q_2)}$ such that $F = BB^T$ is then obtained as

$$B = \mathrm{diag}\left(\sqrt{f_1}, \sqrt{f_2}, \ldots, \sqrt{f_{q_1+q_2}}\right).$$

The procedure for obtaining C is analogous. The LTI system is finally transformed into $A := U^TAU$, $B := U^TB$, and $C := CU$ using a random orthogonal transformation $U \in \mathbb{R}^{n \times n}$. The system thus defined has a minimal realization of order $r = q_1$. The Cholesky factors satisfy $\mathrm{rank}\,(S) = q_1 + q_2$ and $\mathrm{rank}\,(R) = q_1 + q_3$.

We first evaluate the execution time of the parallel model reduction algorithms. In the example, we set $n = 1000$, $m = p = 100$, and $q_1 = q_2 = q_3 = 50$. As there is no noticeable gap in the Hankel singular value distribution of the system, we obtain a reduced-order model of order $r = 40$.

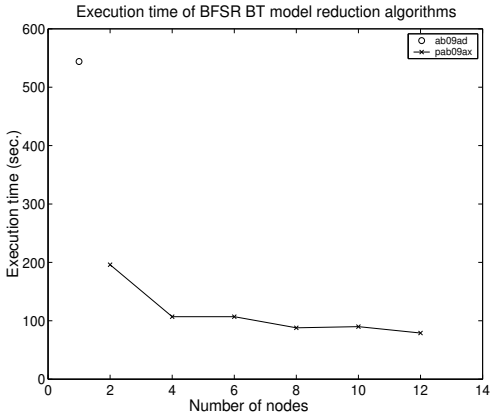

Fig. 4. Performance of the parallel model reduction algorithms.

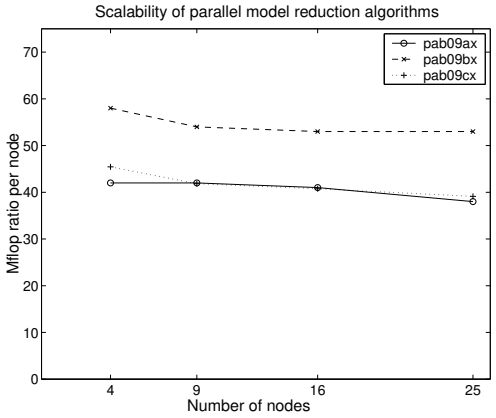

Fig. 5. Scalability of the parallel model reduction algorithms.

Figure 4 reports the execution time of the BFSR BT serial routine for model reduction in SLICOT (`ab09ad`) and the corresponding parallel algorithm (`pab09ax`) as the number of nodes, n_p, is increased. The results show a considerable acceleration achieved by the parallel algorithm (with even super speed-ups). This is partially due to the efficiency of the Lyapunov solvers used in our algorithms which compute factors of the solutions in compact (full-rank) form instead of square matrices, thus requiring less computations. Comparison of the results on 2 and 4 nodes roughly shows the efficiency of the parallel algorithm. The execution time is reduced by a factor of almost 2 (the number of resources, that is nodes, is doubled). Using a larger number of nodes does not achieve a significant reduction of the execution time due to the small ratio $n/\sqrt{n_p}$.

We next evaluate the scalability of our parallel algorithms. As the memory of the system does not allow to test the serial algorithms on larger problems, in the experiment we fix the problem size per node using $n/\sqrt{n_p} = 800$, $m/\sqrt{n_p} = 400$, $p/\sqrt{n_p} = 400$,

and $q/\sqrt{n_p} = 200$, with $q_1 = q_2 = q_3 = q$. In Figure 4 we report the Mflop ratio per node for the parallel model reduction algorithms. Figure 5 shows a high scalability of all three algorithms, as there is only a minor decrease in the Mflop ratio per node as the number of nodes is increased up to 25 (a problem of size $n = 4000$). The scalability confirms that a larger problem can be solved by increasing proportionally the number of nodes employed.

VI. Concluding Remarks

In this paper we have described a parallel library for model reduction of large-scale, dense LTI systems using absolute error methods, and a web service which allows interaction between remote users and a cluster environment.

Our experiments report a notable reduction in the execution time when the parallel algorithms are used. An important part of this reduction is due to the use of compact factors of the Gramian matrices. Future extensions of the library will include parallel routines for relative error model reduction.

Referencias

[1] D. Arnold, S. Agrawal, L.S. Blackford, J. Dongarra, M. Miller, S. Vahdiyar, K. Sagi, and Z. Shi. Users's guide to NetSolve. Technical Report UT-CS-01-467, University of Tennessee at Knoxville, Computer Science Dept., 2001. Available from `http://icl.cs.utk.edu/netsolve`.

[2] P. Benner and E.S. Quintana-Ortí. Solving stable generalized Lyapunov equations with the matrix sign function. *Numer. Algorithms*, 20(1):75–100, 1999.

[3] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Balanced truncation model reduction of large-scale dense systems on parallel computers. *Mathematical and Computer Modeling of Dynamical Systems*, 6(4):383–405, 2000.

[4] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. PSLICOT routines for model reduction of stable large-scale systems. In *Proc. 3rd NICONET Workshop on Numerical Software in Control Engineering, Louvain-la-Neuve, Belgium, January 19, 2001*, pages 39–44. 2001.

[5] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Computing optimal Hankel norm approximations of large-scale systems. In preparation.

[6] I. Blanquer, D. Guerrero, V. Hernández, E.S. Quintana-Ortí, and P. Ruíz. Parallel-SLICOT implementation and documentation standards. SLICOT Working Note 1998–1, `http://www.win.tue.nl/niconet/`, September 1998.

[7] J. Cheng, G. Ianculescu, C.S. Kenney, A.J. Laub, and P. M. Papadopoulos. Control-structure interaction for space station solar dynamic power module. *IEEE Control Systems*, pages 4–13, 1992.

[8] P. Y. Chu, B. Wie, B. Gretz, and C. Plescia. Approach to large space structure control system design using traditional tools. *AIAA J. Guidance, Control, and Dynamics*, 13:874–880, 1990.

[9] U.B. Desai and D. Pal. A transformation approach to stochastic model reduction. *IEEE Trans. Automat. Control*, AC–29:1097–1100, 1984.

[10] E. Elmroth, P. Johansson, B. Kågström, and D. Kressner. A Web computing environment for the SLICOT library. SLICOT Working Note 2001–2, `http://www.win.tue.nl/niconet/`, June 2001.

[11] L. Fortuna, G. Nummari, and A. Gallo. *Model Order Reduction Techniques with Applications in Electrical Engineering*. Springer-Verlag, 1992.

[12] K. Glover. All optimal Hankel-norm approximations of linear multivariable systems and their L^∞ norms. *Internat. J. Control*, 39:1115–1193, 1984.

[13] M. Green. Balanced stochastic realization. *Linear Algebra Appl.*, 98:211–247, 1988.

[14] J. Gunnels, G. Henry, and R.A. van de Geijn. A family of high-performance matrix multiplication algorithms. In V.N. Alexander, J. Dongarra, B.A. Julianno, R.S. Renner, and C.J. Kenneth Tan, editors, *Computational Science - ICCS 2001, Part I, Lecture Notes in Computer Science 2073*, pages 51–60, 2001.

[15] Y. Liu and B.D.O. Anderson. Controller reduction via stable factorization and balancing. *Internat. J. Control*, 44:507–531, 1986.

[16] B. C. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Trans. Automat. Control*, AC-26:17–32, 1981.

[17] C.R. Paul. *Analysis of Multiconductor Transmission Lines*. Wiley–Interscience, Singapur, 1994.

[18] J.D. Roberts. Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *Internat. J. Control*, 32:677–687, 1980. (Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971).

[19] M.G. Safonov and R.Y. Chiang. A Schur method for balanced-truncation model reduction. *IEEE Trans. Automat. Control*, AC–34:729–733, 1989.

[20] R.A. Smith. Matrix equation $XA + BX = C$. *SIAM J. Appl. Math.*, 16(1):198–201, 1968.

[21] M.S. Tombs and I. Postlethwaite. Truncated balanced realization of a stable non-minimal state-space system. *Internat. J. Control*, 46(4):1319–1330, 1987.

[22] A. Varga. Balancing-free square-root algorithm for computing singular perturbation approximations. In *Proc. of the 30th IEEE Conf. Dec. Control*, pages 1062–1065, 1991.

[23] A. Varga. Efficient minimal realization procedure based on balancing. In *Prepr. of the IMACS Symp. on Modeling and Control of Technological Systems*, volume 2, pages 42–47, 1991.

[24] A. Varga and K. H. Fasol. A new square–root balancing–free stochastic truncation model reduction algorithm. In *Prepr. 12th IFAC World Congress*, volume 7, pages 153–156, Sydney, Australia, 1993.

[25] K. Zhou, J.C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice-Hall, Upper Saddle River, NJ, 1996.