Zentrum für Technomathematik
Fachbereich 3 – Mathematik und Informatik

Balanced Truncation Model Reduction of Large-Scale Dense Systems on Parallel Computers

Peter Benner Enrique S. Quintana-Ortí
Gregorio Quintana-Ortí

Report 99–07

Berichte aus der Technomathematik

Report 99–07 September 1999

Balanced Truncation Model Reduction of Large-Scale Dense Systems on Parallel Computers[1]

Peter Benner
Zentrum für Technomathematik
Fachbereich 3 – Mathematik und Informatik
Universität Bremen
D–28334 Bremen
Germany
`benner@math.uni-bremen.de`

Enrique S. Quintana-Ortí Gregorio Quintana-Ortí
Departamento de Informática
Universidad Jaime I
12080 Castellón
Spain
`quintana@inf.uji.es` `gquintan@inf.uji.es`

[1]Partially supported by the DAAD programme *Acciones Integradas Hispano-Alemanas*. Enrique S. Quintana-Ortí and Gregorio Quintana-Ortí's research is funded by a Bancaixa research project.

Abstract

Model reduction is an area of fundamental importance in many modeling and control applications. In this paper we analyze the use of parallel computing in model reduction methods based on balanced truncation of large-scale dense systems. The methods require the computation of the Gramians of a linear-time invariant system. Using a sign function-based solver for computing the Cholesky factors of the Gramians yields some favorable computational aspects in the subsequent computation of the reduced-order model, particularly for non-minimal systems. As sign function-based computations only require efficient implementations of basic linear algebra operations readily available, e.g., in the BLAS, LAPACK, and ScaLAPACK, good performance of the resulting algorithms on parallel computers is to be expected. Our experimental results on a PC cluster show the performance and scalability of the parallel implementation.

1 Introduction

Consider the transfer function matrix (TFM) $G(\lambda) = C(\lambda I - A)^{-1}B + D$, and the associated stable, but not necessarily minimal, realization of a linear time-invariant (LTI) system,

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

with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, and $D \in \mathbb{R}^{p\times m}$. The number of state variables n is said to be the order of the system.

We are interested in finding a reduced order LTI system,

$$\begin{array}{rcl} \dot{x}_r(t) & = & A_r x_r(t) + B_r u_r(t), \quad t > 0, \quad x_r(0) = x_r^0, \\ y_r(t) & = & C_r x_r(t) + D_r u_r(t), \qquad t \geq 0 \end{array} \tag{2}$$

of order r, $r \ll n$, such that TFM $G_r(\lambda) = C_r(\lambda I - A_r)^{-1}B_r + D_r$, approximates (1). A large class of model reduction methods rely on similarity 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 $T_r \in \mathbb{R}^{n\times r}$, the reduced-order model is determined as $A_r = T_l A T_r$, $B_r = T_l B$, $C_r = C T_r$, and $D_r = D$. In this paper we will focus on these methods. Moreover, hereafter, we assume that A is a (Hurwitz) stable matrix, i.e., the spectrum of A as denoted by $\Lambda(A)$ is contained in the open left half plane. This implies that the system (1) is stable, that is, all the poles of $G(s)$ have strictly negative real parts.

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. Model reduction methods usually differ in the measure they attempt to minimize. The methods considered here are all based on balanced truncation (BT) methods [33, 38, 40, 41]. They belong to the family of absolute error methods, which try to minimize $\|\Delta_a\|_\infty = \|G - G_r\|_\infty$. Here, $\|G\|_\infty$ denotes the $\mathcal{L}_\infty$- or $\mathcal{H}_\infty$-norm of a stable, rational matrix function which is defined as

$$\|G\|_\infty = \operatorname*{ess\ sup}_{\omega\in\mathbb{R},\ \omega\geq 0} \sigma_{\max}(G(\jmath\omega)), \tag{3}$$

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

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., [15, 16, 20, 34]. LTI systems with state space dimension n of order 10^2 to 10^4 (and higher) are common in these applications. All balanced truncation model reduction methods for LTI systems with dense coefficient matrices A have a computational cost of $\mathcal{O}(n^3)$ floating-point operations (flops). Large-scale applications thus clearly benefit from using parallel computing techniques to obtain the reduced system.

A review of the most common approaches to compute reduced order models is given in Section 2.

BT model reduction methods are strongly related to the controllability Gramian W_c and the observability Gramian W_o of the system (1). The Gramians are given by the solutions of two "coupled" (as they share the same coefficient matrix A) *Lyapunov equations*

$$AW_c + W_c A^T + BB^T = 0, \tag{4}$$
$$A^T W_o + W_o A + C^T C = 0. \tag{5}$$

As A is assumed to be stable, W_c and W_o are positive semidefinite and therefore can be factored as $W_c = S^T S$ and $W_o = R^T R$. The factors S and R are called the *Cholesky factors* of the Gramians. The BT model reduction methods described in Section 2 use one of the products $W_c W_o$ or SR^T for computing the reduced-order model. Hence, all the methods considered have in common that in the first step, they compute the solutions to the Lyapunov equations (4) and (5). Traditional algorithms for solving these equations are the Bartels-Stewart method [4] which computes the solution of a Lyapunov equation explicitly and Hammarling's method [24] which computes the Cholesky factors in equations of the form (4) or (5) directly without computing the solution matrix. Both methods share an initial stage where the real Schur factorization of A has to be computed by means of the QR algorithm [23]. The parallelization of the QR algorithm on parallel distributed memory architectures has been reported several times in the literature as a difficult task [13, 21, 39]. The algorithm itself is not scalable [27], and the parallelization results [27, 28] are far from those of usual matrix factorizations [11, 17].

In Section 3 we describe a different technique, based on the matrix sign function, for solving Lyapunov equations as in (4) and (5). We will focus here only on BT model reduction methods using the product SR^T and hence describe algorithms for computing the Cholesky factors directly. This algorithm only requires efficient and scalable parallel kernels such as matrix factorizations, triangular linear system solvers, etc. Moreover, it has the advantage of returning full-rank factorizations of the Gramians. That is, if the system is not minimal, i.e., W_c and/or W_o are singular, then the Cholesky factors computed by this method are full-rank matrices having less rows than columns in contrast to Hammarling's method which returns in this case square but singular matrices. This saves some computational cost and workspace in the subsequent computations for computing the reduced-order model. This approach may also have some advantages regarding numerical robustness if the McMillan degree and a minimal realization of the system is to be determined using the Cholesky factors of the Gramians.

The model reduction techniques described here are based on a singular value decomposition (SVD) of the product SR^T. Hence, in Section 4 we describe several approaches for computing the SVD of the product of two matrices with enhanced accuracy.

The parallel implementations of the algorithms employ the kernels in ScaLAPACK [11]. This is a public-domain library for parallel computers which provides scalable parallel distributed subroutines for many matrix algebra kernels in LAPACK [1].

Our experimental results in Section 5 report the performance and scalability of the parallel implementations on a parallel distributed cluster based on Intel Pentium-II processors. These results can be extended to many similar model reduction algorithms proposed in the literature as they usually require the solution of the same types of computational problems. Finally, in Section 6 the conclusions of our work are outlined.

2 Balanced Truncation Model Reduction Methods

In this section we review three model reduction algorithms based on BT methods. Serial implementations of these algorithms can be found in the *Subroutine Library in Control Theory* – SLICOT[1] [8]. We also consider a modification of the algorithms by replacing the Cholesky factors of the Gramians by full-rank factors, resulting in a smaller arithmetic cost and workspace requirement in case of non-minimal LTI systems.

[1]Available from `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/SLICOT`.

Equations (4) and (5) have a unique pair of solutions as assuming A to be stable guarantees that $\lambda_i + \lambda_j \neq 0$ for all λ_i, $\lambda_j \in \Lambda(A)$. Moreover, as BB^T and C^TC are semidefinite matrices, the Gramians are also semidefinite and can be decomposed as $W_c = S^TS$ and $W_o = R^TR$. The factors $S, R \in \mathbb{R}^{n\times n}$ can be chosen triangular and are called the Cholesky factors of the Gramians.

In [40] it is shown that BT model reduction can be achieved using SR^T instead of the product of the Gramians themselves. The resulting *square-root (SR) algorithm* avoids working with the Gramians as their condition number can be up to the square of the condition number of the Cholesky factors. In these algorithms equations (4) and (5) are initially solved for the Cholesky factors without ever forming the Gramians explicitly. This can be achieved, e.g., by the algorithms described in [24, 6]. Then the SVD of the product

$$SR^T = [U_1\ U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \tag{6}$$

is computed. Here, the matrices are partitioned at a given dimension r with $\Sigma_1 = \mathrm{diag}\,(\sigma_1, \ldots, \sigma_r)$ and $\Sigma_2 = \mathrm{diag}\,(\sigma_{r+1}, \ldots, \sigma_n)$ such that

$$\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r > \sigma_{r+1} \geq \sigma_{r+2} \geq \ldots \geq \sigma_n \geq 0. \tag{7}$$

If $\sigma_r > 0$ and $\sigma_{r+1} = 0$, i.e., $\Sigma_2 = 0$, then r is the *McMillan* degree of the given LTI system. That is, r is the state-space dimension of a minimal realization of the system.

For model reduction, r should be chosen in order to give a natural separation of the states, i.e., one should look in the Hankel singular values σ_k, $k = 1, \ldots, n$, for a large gap $\sigma_r \gg \sigma_{r+1}$ [40].

Notice that the SVD in (6) can be obtained without explicitly forming the product of the Cholesky factors using the techniques in [19, 25]. Finally, defining

$$T_l = \Sigma_1^{-1/2} V_1^T R \quad \text{and} \quad T_r = S^T U_1 \Sigma_1^{-1/2}, \tag{8}$$

the reduced system is given by

$$A_r = T_l A T_r, \quad B_r = T_l B, \quad C_r = C T_r, \quad \text{and} \quad D = D_r. \tag{9}$$

In case that $\Sigma_1 > 0$ and $\Sigma_2 = 0$, (9) is a *minimal realization* of the TFM $G(\lambda)$ [40], i.e., r is the minimum dimension of the state-space for which a realization of $G(\lambda)$ in the form of an LTI system (1) is possible. Hence, choosing r in (6) maximal such that $\sigma_r > 0$ and $\sigma_{r+1} = 0$, this procedure can be used to compute minimal realizations. Of course, the decision if $\sigma_{r+1} = 0$ has to be based on a numerically reliable criterion.

It can further be proved that for a stable LTI system, choosing any partitioning in (6) such that $\sigma_r > \sigma_{r+1}$ yields a stable, minimal, and balanced reduced model. The Gramians corresponding to the resulting TFM $G_r(\lambda)$ are both equal to Σ_1. See [40] for a proof.

As the reduced model in (9) is balanced, the projection matrices in (8) tend to be ill-conditioned if the original system is highly unbalanced, resulting in inaccurate reduced-order models. An alternative here are the *balancing-free (BF) algorithms* [38]. Here, the reduced-order model is not balanced. In these algorithms, after solving equations (4) and (5) for the Gramians, an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ is computed such that

$$QW_cW_oQ^T = S \tag{10}$$

is in upper real Schur form.Using an eigenvalue reordering procedure as described, e.g., in [23], a pair of orthogonal matrices Q_a, $Q_d \in \mathbb{R}^{n\times n}$ is computed such that the diagonal blocks in $Q_a^T W_c W_o Q_a$ and $Q_d^T W_c W_o Q_d$ are ordered, respectively, in ascending and descending order of the absolute magnitude of the eigenvalues. Let $Q_a = [Q_{a_1}, Q_{a_2}]$, and $Q_d = [Q_{d_1}, Q_{d_2}]$, with Q_{a_2}, $Q_{d_1} \in \mathbb{R}^{n\times r}$, then the SVD

$$Q_{a_2}^T Q_{d_1} = [U_1 \; U_2] \left[\begin{array}{cc} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{array} \right] \left[\begin{array}{c} V_1^T \\ V_2^T \end{array} \right], \tag{11}$$

provides the orthonormal matrices that allow the construction of the reduced system as in (8) and (9). The development of the periodic QR algorithm [12, 26] allows to compute the factorization in (10) without explicitly constructing the product $W_c W_o$. The method can also be adapted to work on the Cholesky factors thus providing a square-root algorithm.

Balancing-free square-root (BFSR) algorithms combine the best characteristics of the SR and BF approaches [41]. BFSR algorithms share the first two steps (solving (4) and (5) for the Cholesky factors and computing the SVD in (6)) with the SR algorithms. Then, two QR factorizations are computed,

$$S^T U_1 = [P_1 \; P_2] \left[\begin{array}{c} \hat{R} \\ 0 \end{array} \right], \qquad R^T V_1 = [Q_1 \; Q_2] \left[\begin{array}{c} \bar{R} \\ 0 \end{array} \right],$$

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. Note that P_2, Q_2 are not needed such that it is sufficient to compute the "skinny" QR decompositions $S^T U_1 = P_1 \hat{R}$ and $R^T V_1 = Q_1 \bar{R}$.

The reduced system is then given by the projection matrices

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

and (9).

For the implementations reported in this paper we chose the SR and BFSR algorithms as the BF algorithm usually shows no advantage over BFSR algorithms with respect to model reduction abilities. Moreover, the BF approach is potentially numerically unstable. For one, it uses the product $W_c W_o$ rather tan SR^T, leading to a squaring of the condition number of the matrix product. Second, the projection matrices T_l and T_r computed by the BFSR approach are often significantly better conditioned than those computed by the BF approach [41]. Furthermore, both SR and BFSR algorithms can be implemented efficiently on parallel computers while the BF method needs a parallel implementation of the QR algorithm (see Equation (10)) and re-ordering of eigenvalues which present severe difficulties regarding its parallel implementation as described in the introduction. For these reasons, we avoid the implementation of the BF algorithm.

Both, the SR and BFSR method can be used to compute minimal realizations of the given LTI system by choosing r in (7) as the McMillan degree of the system. Though the BFSR method is numerically more reliable than the SR approach (the computed projection matrices are usually better conditioned than those in (8)), the SR method is implemented here as the reduced-order model in (9) is balanced. A balanced realization is needed when computing reduced-order models based on optimal Hankel-norm approximation (HNA) [22]. The reduced-order models computed by either one of the SR or BFSR approaches can also be used to compute a singular perturbation approximation (SPA) [32] of the given LTI system.

Parallel versions of the SPA or HNA methods can be easily obtained based on the implementations of the SR and BFSR methods described here analogously to the corresponding SLICOT routines `AB09BD`[2] and `AB09CD`[2], respectively, [42]. Only matrix products and sums as well as the solution of linear systems are required to obtain SPA or HNA models from the models obtained by the SR or BFSR methods.

The main goal is here to show how parallel computing can be used to compute reduced-order models that can be easily handled, in a second stage, on a single-processor machine.

So far we have assumed that the Cholesky factors S and R of the Gramians are square $n \times n$ matrices. For non-minimal systems, we have $\operatorname{rank}(S) < n$ and/or $\operatorname{rank}(R) < n$. Hence, rather than working with the Cholesky factors, we may use *full-rank factors* of W_c, W_o. As W_c and W_o are positive semidefinite, there exist matrices $\hat{S} \in \mathbb{R}^{n_c \times n}$, $\hat{R} \in \mathbb{R}^{n_o \times n}$, such that $W_c = \hat{S}^T\hat{S}$, $W_o = \hat{R}^T\hat{R}$, and

$$\begin{aligned} n_c &:= \operatorname{rank}\left(\hat{S}\right) = \operatorname{rank}(S) = \operatorname{rank}(W_c), \\ n_o &:= \operatorname{rank}\left(\hat{R}\right) = \operatorname{rank}(R) = \operatorname{rank}(W_o). \end{aligned}$$

The full-rank factors $\hat{S}$, $\hat{R}$, hereafter sometimes also referred to as Cholesky factors, can be obtained from S and R by omitting the trailing rows of the factors that must be zero due to the triangular form of S and R. (For non-triangular factorizations, it is sufficient to compute QR factorizations of S and R.) A more efficient way is to compute $\hat{S}$ and $\hat{R}$ directly, e.g., with the algorithm described in Section 3. In the latter case, S and R can be defined by $S := \begin{bmatrix} \hat{S} & 0 \end{bmatrix}$, $R := \begin{bmatrix} \hat{R} & 0 \end{bmatrix}$. The SVD in (6) can then be obtained from that of $\hat{S}\hat{R}^T$ as follows. Here we assume $n_c \geq n_o$, the case $n_c < n_o$ can be treated analogously. Then we can compute the SVD

$$\hat{S}\hat{R}^T = \hat{U}\begin{bmatrix} \hat{\Sigma} \\ 0 \end{bmatrix}\hat{V}^T, \qquad \hat{\Sigma} = \operatorname{diag}(\sigma_1, \ldots, \sigma_{n_o}), \tag{12}$$

where $\hat{U} \in \mathbb{R}^{n_c \times n_c}$, $\hat{V} \in \mathbb{R}^{n_o \times n_o}$. Partitioning $\hat{U} = \begin{bmatrix} \hat{U}_1 & \hat{U}_2 \end{bmatrix}$ such that $\hat{U}_1 \in \mathbb{R}^{n_c \times n_o}$, the SVD of SR^T is given by

$$SR^T = \left[\begin{array}{c|cc} \hat{U}_1 & \hat{U}_2 & 0 \\ \hline 0 & 0 & I_{n-n_c} \end{array}\right]\left[\begin{array}{c|c} \hat{\Sigma}_1 & 0 \\ \hline 0 & 0 \end{array}\right]\left[\begin{array}{c|c} \hat{V}^T & 0 \\ \hline 0 & I_{n-n_o} \end{array}\right]. \tag{13}$$

Then the decision on the index r yielding the McMillan degree of the system or the size of the reduced order model can be based on the singular values of $\hat{\Sigma}_1$. Note that the subsequent computations can also be performed working with $\hat{U}_1$, $\hat{\Sigma}_1$, and $\hat{V}$ rather than using the data from the full-size SVD in (13). This amounts in a significant savings of workspace and computational cost. For example, using the Golub–Reinsch SVD (see, e.g., [23]), (6) requires $22n^3$ flops and workspace for $2n^2$ real numbers if U, V are to be formed explicitly while (12) only requires $14n_cn_o^2 + 8n_o^3$ flops and workspace for $n_c^2 + n_o^2$ real numbers. In particular, for large-scale dynamical systems, the *numerical rank* of W_c, W_o and $\hat{S}$, $\hat{R}$ is often much less than n; see [35, 36] and Remarks 3.1, 3.3 below. Suppose that (numerically) $n_c = n_o = n/10$ (which can quite frequently be observed when the system comes from the spatial discretization of parabolic or hyperbolic partial differential equations), then the computation of (12) is 1000

[2]Available from `ftp://wgs.esat.kuleuven.ac.be/pub/WGS/SLICOT/libindex.html#A`

times less expensive than that of (6) and only 1% of the workspace is required for (12) as compared to (6). Some more savings are obtained from the cheaper computation of the projection matrices yielding the reduced-order models.

In the next section we show how the full-rank factors $\hat{S}$ and $\hat{R}$ can be computed directly without having to compute S, R or even W_c, W_o first.

3 Computing the Cholesky Factors of the System Gramians

All the model reduction methods in the previous section require, as a first stage, the solutions (or their Cholesky factors) of two Lyapunov equations. In this section we describe Lyapunov equation solvers based on the matrix sign function. Details of the algorithms can be found in [9]. These are specially appropriate for parallel distributed memory computers. Some minor modifications of the algorithms in order to address some properties of the Lyapunov equations corresponding to model reduction are described here.

Consider a matrix $Z \in \mathbb{R}^{n\times n}$, with no eigenvalues on the imaginary axis and let $Z = S\begin{bmatrix} J^- & 0 \\ 0 & J^+ \end{bmatrix}S^{-1}$ be its Jordan decomposition [23]. Here, the Jordan blocks in $J^- \in \mathbb{R}^{k\times k}$ and $J^+ \in \mathbb{R}^{(n-k)\times(n-k)}$ contain, respectively, the eigenvalues of Z in the open left and right complex planes. The *matrix sign function* of Z is defined as $\operatorname{sign}(Z) := S\begin{bmatrix} -I_k & 0 \\ 0 & I_{n-k} \end{bmatrix}S^{-1}$, where I_k denotes the identity matrix of order k. Note that $\operatorname{sign}(Z)$ is unique and independent of the order of the eigenvalues in the Jordan decomposition of Z. Many other definitions of the sign function can be given; see [29] for an overview.

The matrix sign function has proved useful in many problems involving spectral decomposition as $(I_n - \operatorname{sign}(Z))/2$ defines the skew projector onto the stable Z–invariant subspace parallel to the unstable subspace. (By the *stable* invariant subspace of Z we denote the Z–invariant subspace corresponding to the eigenvalues of Z in the open left half plane.)

Applying Newton's root-finding iteration to $Z^2 = I_n$, where the starting point is chosen as Z, we obtain the Newton iteration for the matrix sign function:

$$Z_0 \leftarrow Z, \quad Z_{k+1} \leftarrow \frac{1}{2}(Z_k + Z_k^{-1}), \qquad k = 0, 1, 2, \ldots, \tag{14}$$

Under the given assumptions, the sequence $\{Z_k\}_{k=0}^{\infty}$ converges to $\operatorname{sign}(Z) = \lim_{k\to\infty} Z_k$ [37].

Although the convergence of the Newton iteration is globally quadratic, the initial convergence may be slow. Acceleration is possible, e.g., via *determinantal scaling* [14],

$$Z_k \leftarrow c_k Z_k, \quad c_k = |\det(Z_k)|^{-\frac{1}{n}},$$

where $\det(Z_k)$ denotes the determinant of Z_k. Other acceleration schemes can be employed; see [2] for a comparison of these schemes.

Roberts [37] was the first to use the matrix sign function for solving Lyapunov (and Riccati) equations. In the proposed method, the solution of the stable Lyapunov equation

$$A^T X + XA + Q = 0, \tag{15}$$

is computed by applying the Newton iteration (14) to the Hamiltonian matrix $H = \begin{bmatrix} A & 0 \\ Q & -A^T \end{bmatrix}$ corresponding to (15). The solution matrix X^* can then be determined from the stable Z–invariant subspace given by the range of the projector $(I_n - \operatorname{sign}(H))/2$. Roberts also shows in [37] that, when applied to H, the Newton iteration (14) can be simplified to

$$\begin{aligned} A_0 &\leftarrow A, & A_{k+1} &\leftarrow \frac{1}{2}\left(A_k + A_k^{-1}\right), \\ Q_0 &\leftarrow Q, & Q_{k+1} &\leftarrow \frac{1}{2}\left(Q_k + (A_k^{-1})^T Q_k A_k^{-1}\right), \end{aligned} \qquad k = 0,1,2,\ldots \tag{16}$$

and that $X = \frac{1}{2}\lim_{k\to\infty} Q_k$. The sequences for A_k and Q_k require $6n^3$ flops per iteration so that 5–6 iterations are as expensive as the Bartels–Stewart method [4].

Iteration (16) can be applied to (5) directly by setting $Q_0 := C^TC$ in order to obtain the observability Gramian as $W_o = Q_\infty/2$. For (4), we just have to replace A by A^T while setting $Q_0 := BB^T$. Both iterations can be combined so that W_c and W_o are computed simultaneously as follows

```
$A_0 := A$, $P_0 := BB^T$, $Q_0 := C^TC$.
FOR $k = 0,1,2,\ldots$ *until convergence*
    $c_k \leftarrow |\det(A_k)|^{-\frac{1}{n}}$,
    $A_{k+1} \leftarrow \frac{1}{2c_k}\left(A_k + c_k^2 A_k^{-1}\right)$,
    $P_{k+1} \leftarrow \frac{1}{2c_k}\left(P_k + c_k^2 A_k^{-1} P_k (A_k^{-1})^T\right)$,
    $Q_{k+1} \leftarrow \frac{1}{2c_k}\left(Q_k + c_k^2 (A_k^{-1})^T Q_k A_k^{-1}\right)$,        (17)
```

At convergence, $W_c = P_\infty/2$, and $W_o = Q_\infty/2$.

Efficient convergence criteria for these iterations have been proposed in [9, 6]. As A is a stable matrix, $A_\infty = \lim_{k\to\infty} A_k = -I_n$, and a suitable convergence criterion for the iterations is to stop when the relative error in the A-iterates drops below a tolerance threshold, i.e., if

$$\|A_k + I_n\|_\infty \le \tau \|A_k\|_\infty$$

for a user-defined tolerance τ. In our implementations we employ $\tau = n\sqrt{\varepsilon}$, and perform two additional iterations once the convergence criterion is satisfied. Due to the quadratic convergence of the Newton iteration, this is usually enough to reach the attainable accuracy.

Iteration (17) has been used in [30] as the first step in a balanced model reduction algorithm and was extended for generalized Lyapunov equations in [5].

In [31, 9] iteration (16) was modified to obtain the Cholesky factors rather than the solutions themselves. The basic idea is that if $Q = C^TC$ and $C_0 = C$, the iterations for the symmetric matrices Q_k can be written in factored form as

$$Q_{k+1} = C_{k+1}^T C_{k+1} = \frac{1}{2c_k}\begin{bmatrix} C_k \\ c_k C_k A_k^{-1} \end{bmatrix}^T \begin{bmatrix} C_k \\ c_k C_k A_k^{-1} \end{bmatrix},$$

yielding

$$C_{k+1} \leftarrow \frac{1}{\sqrt{2c_k}}\begin{bmatrix} C_k \\ c_k C_k A_k^{-1} \end{bmatrix}. \tag{18}$$

Similarly, for iteration (16) with $B_0 := B$ we obtain

$$B_{k+1} \leftarrow \frac{1}{\sqrt{2c_k}}[\,B_k,\; c_k A_k^{-1} B_k\,]. \tag{19}$$

Using (18) and (19), the workspace required to store the B_k's, C_k's is doubled in each iteration step. This can be avoided by computing in each iteration step an LQ factorization of B_{k+1} and a QR factorization of C_{k+1} such that

$$B_{k+1} = [S_{k+1}\, 0]U_{k+1}, \quad C_{k+1} = V_{k+1} \begin{bmatrix} R_{k+1} \\ 0 \end{bmatrix}.$$

As

$$B_k B_k^T = S_k S_k^T, \quad C_k^T C_k = R_k^T R_k,$$

it is sufficient to store the triangular factors in B_k, C_k. The orthogonal factors need not be accumulated, but still the amount of work required in each iteration step is increased by these factorizations. Therefore, in [9] a compromise is proposed: first, fix the available workspace for the C_k's (here also for the B_k's). A reasonable amount is a $2n \times n$ array (or $n \times 2n$ for the B_k's) as the rank of the required Cholesky factor R (and S) cannot exceed n. Therefore, we may use (18) and (19) as long as $C_k \in \mathbb{R}^{p_k \times n}$ for $p_k \leq n$ and $B_k \in \mathbb{R}^{n \times m_k}$ for $m_k \leq n$, and then switch to working with the triangular factors obtained by the factorizations. Hence, we perform $k \leq \log_2 \frac{n}{m}$ iterations with (19) and $l \leq \log_2 \frac{n}{p}$ iterations with (18) before starting to compute factorizations in each step and work only with the triangular factors. If convergence is achieved before the switch, we have to compute final factorizations to obtain the Cholesky factors.

Here, we propose a slightly different strategy taking into account that for large-scale non-minimal systems, the Cholesky factors are often of low (numerical) rank such that we can save some workspace and arithmetic work by not allowing the iterates for the Cholesky factors to become rank-deficient. That is, in each step we compute a rank-revealing QR decomposition of the matrix in (18), using a QR decomposition with column pivoting [23]. The rank-revealing LQ decomposition in (19) is obtained as a QR decomposition with column pivoting of B_{k+1}^T. The rank of the iterates is then determined using an incremental condition estimator [10] resulting in the following decompositions:

$$\begin{aligned} \frac{1}{\sqrt{2c_k}} \begin{bmatrix} C_k \\ c_k C_k A_k^{-1} \end{bmatrix} &= V_{k+1} \begin{bmatrix} (R_{k+1})_{11} & (R_{k+1})_{12} \\ 0 & (R_{k+1})_{22} \end{bmatrix} \Pi_{k+1}^C, \\ \frac{1}{\sqrt{2c_k}} [\, B_k,\ c_k A_k^{-1} B_k \,] &= \Pi_{k+1}^B \begin{bmatrix} (S_{k+1})_{11} & 0 \\ (S_{k+1})_{21} & (S_{k+1})_{22} \end{bmatrix} U_{k+1}. \end{aligned}$$

Here, V_{k+1}, U_{k+1} are orthogonal matrices, Π_{k+1}^C and Π_{k+1}^B are permutation matrices, $(R_{k+1})_{11} \in \mathbb{R}^{r_{k+1} \times r_{k+1}}$ and $(S_{k+1})_{11}^T \in \mathbb{R}^{s_{k+1} \times s_{k+1}}$ are upper triangular matrices where r_{k+1}, s_{k+1} are the estimated ranks of the iterates, and $(R_{k+1})_{22} \approx 0$, $(S_{k+1})_{22} \approx 0$. Setting $(R_{k+1})_{22} = 0$, $(S_{k+1})_{22} = 0$, we can proceed with the new iterates

$$\begin{aligned} C_{k+1} &\leftarrow [\,(R_{k+1})_{11}\ (R_{k+1})_{12}\,]\Pi_{k+1}^C \in \mathbb{R}^{r_{k+1} \times n}, \\ B_{k+1} &\leftarrow \Pi_{k+1}^B \begin{bmatrix} (S_{k+1})_{11} \\ (S_{k+1})_{21} \end{bmatrix} \in \mathbb{R}^{n \times s_{k+1}}. \end{aligned}$$

Using the notation in Section 2, we have that $\lim_{k\to\infty} B_k = \hat{S}^T$ and $\lim_{k\to\infty} C_k = \hat{R}$, i.e., the iterates converge to the full-rank factors of the Gramians. As in most applications, $m, p \ll n$ and the *numerical rank* (see, e.g., [23]) of the Cholesky factors S, R of the system Gramians is also usually much smaller than n, this technique quite often saves a large amount of workspace and computational cost compared to using the technique described above where the workspace for the iterates is increased up to size $n \times n$. As outlined in Section 2, the product $\hat{S}\hat{R}^T$ is a small size, in general rectangular, matrix for which the subsequent computations needed for model reduction are much cheaper than for the full or triangular $n \times n$ Cholesky factors as obtained by Hammarling's method which is used in the implementation of the balanced truncation model reduction algorithms described in [42].

The Lyapunov solvers described above are iterative procedures composed of LU and QR factorizations, triangular linear systems, and matrix inversions (see, e.g., (17) and (18)). These matrix operations are of wide appeal for computer architectures that can take advantage of block-partitioned algorithms, and specially for parallel distributed architectures [11, 17].

In our experiments, we base our rank decisions on a relative tolerance threshold $10n\varepsilon$, where n is the size of the matrix and ε is the machine precision.

Remark 3.1 *By allowing $\|(R_{k+1})_{22}\|$ and $\|(S_{k+1})_{22}\|$ to become larger or by fixing the number of allowed columns in R_k and S_k, the above procedure can also be used to obtain low-rank approximations of the Cholesky factors. In particular for LTI systems having Gramians with fast decaying eigenvalues, such low-rank approximations often yield all relevant information needed for model reduction. This is also used in [35] for the model reduction algorithms developed there for LTI systems with sparse coefficient matrix A.*

Remark 3.2 *The full-rank factors of the system Gramians can also be used directly to compute a reduced-order model using the* dominant subspace *approach proposed in [35].*

Remark 3.3 *The above algorithm can be used to compute a* numerically minimal realization *and the corresponding state-space dimension $\widetilde{n}_{\min}$, i.e., the* numerical McMillan degree*, of an LTI system in the presence of rounding errors. Here, we denote quantities computed with finite precision arithmetic with a "tilde". Having computed the Cholesky factors $\widetilde{S}$ and $\widetilde{R}$ of the Gramians using the algorithm described above, and computing the SVD of $\widetilde{S}\widetilde{R}^T$ as in (6), we obtain the Hankel singular values $\{\widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_n\}$. In analogy to the numerical rank of a matrix (see, e.g., [23, Chapter 2]), we define $\widetilde{n}_{\min}$ by*

$$\widetilde{\sigma}_1 \geq \ldots \widetilde{\sigma}_{\widetilde{n}_{\min}} > \varepsilon\widetilde{\sigma}_1 \geq \widetilde{\sigma}_{\widetilde{n}_{\min}+1} \geq \ldots \geq \widetilde{\sigma}_n.$$

A numerically minimal realization can then be computed by any of the methods described in Section 2 by splitting the SVD in (6) such that $\Sigma_1 = \operatorname{diag}(\widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_{\widetilde{n}_{\min}})$.

As with any rank decision, the accuracy of the computed singular values is crucial to the successful determination of the numerical McMillan degree. Therefore we review in the next section some methods that can be used for computing an SVD of SR^T with enhanced accuracy.

4 Computing the SVD of SR^T with Enhanced Accuracy

The SR and BFSR algorithms require the computation of the SVD of the matrix product SR^T (see equation (6)). The most direct method is to construct the matrix product and then compute the SVD.

The accuracy of the computed SVD can be improved if the SVD is computed without forming the explicit product [19, 25]. The main drawback of this approach is the lack of efficient parallel routines for its computation.

A recent method [18] allows an accurate computation of the SVD in case S and R are full-column rank matrices. The algorithm, adapted for the matrix product SR^T, works as follows:

Input : $S \in \mathbb{R}^{l\times n}$, $R \in \mathbb{R}^{k\times n}$, $\operatorname{rank}(S) = \operatorname{rank}(R) = n$.

Step 1. Compute $\Delta_S = \operatorname{diag}(\|Se_i\|_2)$, with e_i the i-th column of I_n.
Set $S = S\Delta_S^{-1}$ and $R = R\Delta_S$.

Step 2. Compute a QR decomposition with column pivoting of R

$$R\Pi = Q\begin{bmatrix} \bar{R} \\ 0_{k-n,n} \end{bmatrix}.$$

Step 3. Compute the matrix product $F = S\Pi\bar{R}^T$.

Step 4. Compute the SVD of F by means of the Jacobi method

$$U^TFV = \begin{bmatrix} \Sigma \\ 0_{l-n,n} \end{bmatrix}.$$

Output: The SVD of SR^T is given by

$$U^T(SR^T)Q\begin{bmatrix} V & 0_{n,k-n} \\ 0_{k-n,n} & I_{k-n,k-n} \end{bmatrix} = \begin{bmatrix} \Sigma & 0_{n,k-n} \\ 0_{k-n,n} & 0_{l-n,k-n} \end{bmatrix}.$$

Unfortunately, the above algorithm requires both S and R to be full-column rank matrices and hence can only be applied in our situation if both Gramians are nonsingular, that is, if the LTI system represents a minimal realization of the TFM $G(\lambda)$. This is rarely the case in practice; in particular in model reduction one frequently starts with just any realization. Moreover, as mentioned before, the computed numerical rank of the Cholesky factors may be smaller than expected.

ScaLAPACK [11] provides a parallel routine for computing the SVD of a general matrix. As no parallel routine is available in ScaLAPACK (version 1.6) for computing the SVD of a matrix product without forming the product explicitly, this will be our method of choice in case S or R are column rank-deficient matrices, that is, if we want to work with the full-rank factors $\hat{S}$, $\hat{R}$ and compute the SVD in (12).

Otherwise, we will use an adapted version of the above algorithm. The implementation of the algorithm is direct on parallel architectures if we replace in Step 4 the Jacobi SVD with the implementation provided in ScaLAPACK. Note that this implementation is based on the Golub–Kahan method (see, e.g., [23]) and hence the favorable error bounds in [18] may not hold. Still, the modified algorithm will yield better numerical results than just computing the SVD of the explicit matrix product SR^T. See [3] for details.

5 Experimental Results

In this section we evaluate the accuracy and performance of our parallel model reduction algorithms for continuous-time LTI systems:

- PDGECMSR: The SR method for model reduction.

- PDGECMBS: The BFSR method for model reduction.

For this purpose, we compare these parallel routines with the analogous serial algorithms in SLICOT. This library includes the Fortran 77 routine AB09AD which implements, among others, the SR and BFSR methods.

All the experimental results in this section were obtained on Intel Pentium-II machines, using Fortran 77 algorithms, and IEEE double-precision floating-point arithmetic ($\varepsilon \approx 2.2204 \times 10^{-16}$).

5.1 Numerical accuracy

We first illustrate the reliability of the parallel model reduction algorithms by means of two examples of moderate order.

We employ the difference in frequency response between the TFMs of the original system and the reduced system to measure the reliability of the model reduction algorithms. An upper bound for this difference is given by [40]

$$\|G - G_r\|_\infty \leq 2 \sum_{i=r+1}^{n} \sigma_i =: \delta,$$

where σ_k, $k = 1, \ldots, n$, are the Hankel singular values of the system, r is the order of the reduced model, and $\| \, . \, \|_\infty$ here denotes the $\mathcal{L}_\infty$-norm of a TFM. In routine AB09AD, r is determined such that $\sigma_r < \tau$, with τ a user-supplied tolerance threshold. The recommended value for model reduction is $\tau = c\sigma_1$ with $c \in \left[10^{-5}, 10^{-3}\right]$. In case a minimal realization is required it is suggested to use a tolerance threshold $\tau = n\varepsilon\sigma_1$. We employ the same thresholds in our parallel algorithms.

Example 1 This is Example 18 from [7]. The system matrices in this example come from a linear-quadratic optimal control problem of one-dimensional heat flow. The system is single-input/single-output ($m = p = 1$) and the order n is of arbitrary size. Increasing n results in a finer grid for the underlying finite-element (FE) approximation scheme of the space variable. Here, we report results for $n \in \{100, 500, 1000\}$. The system is parameterized by the tuple $(a, b, c, \beta_1, \beta_2, \gamma_1, \gamma_2)$ that we set in our example to $(0.001, 1, 1, 0.2, 0.3, 0.2, 0.3)$. Note that transforming the discretized system to a LTI system as in (1) results in a system with dense coefficient matrix A although the FE matrices are sparse.

As there is no significant gap between any two consecutive Hankel singular values of the system we obtain a reduced model of fixed order $r = 6$ and $n = 100, 500, 1000$.

Algorithm AB09AD relies on Hammarling's method to solve the Lyapunov equations for the Cholesky factors. The factors S and R computed with this method are square of order 100. The explicit product SR^T (a $n \times n$ matrix) is then constructed and the SVD is obtained by (a slightly modified version of) the Golub–Kahan method. Our parallel algorithms instead solve the Lyapunov equations for the Cholesky factors using the Newton iteration for the matrix sign function. The full-rank factors $\hat{S}$ and $\hat{R}$ thus obtained are of size $n_c \times n$ and $n_o \times n$, respectively, with $n_c = 34, 32, 32$ and $n_o = 36, 38, 37$ for $n = 100, 500, 1000$. The product $\hat{S}\hat{R}^T \in \mathbb{R}^{n_c \times n_o}$ is explicitly constructed and the SVD is computed using the Golub–Kahan method. Table 1 shows the Hankel norm and the difference δ obtained with the serial and

n	subroutine	$\sigma_1 = \|G(s)\|_H$	δ
100	AB09AD/SR	7.22939244096230e-1	9.69240669317034e-5
	PDGECMSR	7.22939244096153e-1	9.69240669316399e-5
500	AB09AD/SR	7.23152354428920e-1	1.01290748783967e-4
	PDGECMSR	7.23152354400824e-1	1.01290748735695e-4
1000	AB09AD/SR	7.23159123599860e-1	1.01548816539096e-4
	PDGECMSR	7.23159123524023e-1	1.01548811421665e-4

Table 1: Hankel norm and difference of the reduced models for Example 1.

parallel algorithms. We only report the results for the SR algorithms as they are identical to those obtained for the BFSR algorithms.

Table 2 reports the norms of the matrices defining the original system and of those defining the reduced-order models obtained with the serial and the parallel algorithms for $n = 100$. The results for $n = 500, 1000$ are omitted here as they show no significant difference.

	$\|A\|_\infty$ or $\|A_r\|_\infty$	$\|B\|_\infty$ or $\|B_r\|_\infty$	$\|C\|_\infty$ or $\|C_r\|_\infty$
system	1.224e+2	1.183	9.901e-3
AB09AD/SR	1.2924697511577e+1	2.0142237527450e-1	2.0142237527450e-1
PDGECMSR	1.2924697511571e+1	2.0142237527451e-1	2.0142237527451e-1
AB09AD/BFSR	1.2998109448472e+1	2.0271109120962e+0	2.0037682847113e-2
PDGECMBS	1.2998109448466e+1	2.0271109120963e+0	2.0037682847114e-2

Table 2: Matrix norms of the system and the reduced models (r=6) for Example 1.

Figure 1 reports the absolute errors in the frequency response on the imaginary axis between the original system and the reduced models computed by the serial and parallel algorithms. The absolute error is computed as the maximum singular value of the error system at $\jmath\omega$, i.e.,

$$\|G(\jmath\omega) - G_r(\jmath\omega)\|_2 = \sigma_{\max}(G(\jmath\omega) - G_r(\jmath\omega)),$$

where $\| \, . \, \|_2$ denotes the matrix 2-norm, and $G(\lambda)$, $G_r(\lambda)$ are the TFMs of the original and the reduced model, respectively. Note that here, $\sigma_{\max}(G(\jmath\omega) - G_r(\jmath\omega)) = |\, G(\jmath\omega) - G_r(\jmath\omega)\,|$ as the example is a single-input, single-output system. Recall that, corresponding to (3),

$$\|G - G_r\|_\infty = \operatorname*{ess\,sup}_{\omega\in\mathbb{R},\, \omega\geq 0} \|G(\jmath\omega) - G_r(\jmath\omega)\|_2.$$

Hence, from the plots in Figure 1 one can estimate $\|G - G_r\|_\infty \approx \delta$ for all sizes of n. The upper limit δ is attained for small frequencies whereas the absolute error decreases for larger frequencies.

All the results we have reported show that there is almost no difference in the obtained accuracy for the serial and the parallel model reduction routines. Also note that there is no visible difference in the reduced-order models obtained by either the SR or BFSR methods in this example. Moreover, reducing the meshsize of the FE discretization which usually increases the accuracy of the computed solution of the underlying partial differential equation does not change the system dynamics here — the $r = 6$ model is almost identical for all sizes of n and approximates the system behavior satisfactorily.

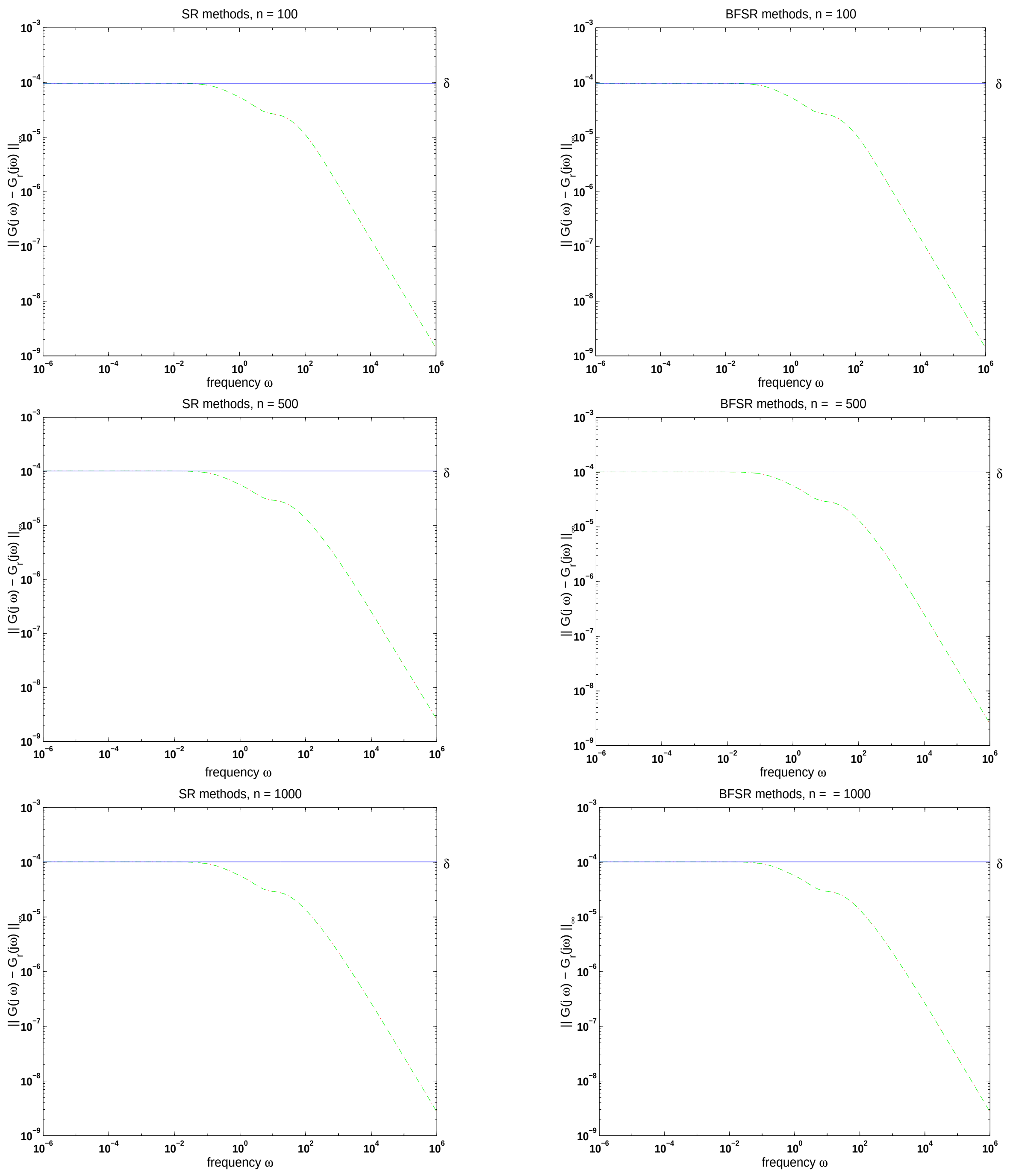

Figure 1: Absolute frequency response errors of the reduced models. Legend: "—" = upper bound given by δ, "– –" = PDGECMSR/PDGECMBS, "$\cdots$" = AB09AD.

5.2 Parallel performance

In this subsection we analyze the performance of the parallel algorithms on a parallel distributed cluster of 25 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, that achieves 200 Mflops (millions of floating-point operations per second) for the matrix product (routine DGEMM). The nodes are connected via a *Myrinet* crossbar network and the communication library BLACS is based on an implementation of the communication library MPI specially developed for this switch. 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 whenever possible.

To evaluate the performance of our parallel model reduction algorithm we generate random LTI systems as follows. First, we generate a random positive semidefinite diagonal Gramian $W_c = \text{diag}\,(\Sigma_{q_1}, \Sigma_{q_2}, 0_{q_3}, 0_{q_4})$, 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 $W_o = \text{diag}\,(\Sigma_{q_1}, 0_{q_2}, \Sigma_{q_3}, 0_{q_4})$, 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,

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

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

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

The procedure for obtaining C is analogous. The LTI system is finally transformed into $A := U^TAU$, $B := U^TB$, and $C := CU$ by a random orthogonal state 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 $\text{rank}\,(S) = q_1 + q_2$ and $\text{rank}\,(R) = q_1 + q_3$.

We first evaluate the reduction in the execution time achieved by the serial and parallel SR algorithms, AB09AD and PDGECMSR, respectively. We omit the results for the BFSR algorithms as they were closely similar. In the example, we set $n = 1000$, and choose several different values for m, p, q_1, q_2 and q_3. Figure 2 compares the execution times of the serial and the parallel algorithms as the number of nodes, n_p, is increased.

The figures show a considerable acceleration achieved by the parallel algorithm PDGECMSR when compared to the serial algorithm AB09AD (see results on 1 node). This is partially due to the efficiency of the Lyapunov solvers used in our algorithms which only need to compute low rank approximations of the Cholesky factors. Comparison of the results on 2 and 4 nodes shows the parallelism of the PDGECMSR algorithm. The execution time is reduced by a factor of almost 2 (the number of resources, that is nodes, is doubled). Using 10–12 nodes does not achieve a significant reduction of the execution time in this case due to the small ratio $n/\sqrt{n_p}$.

As the memory of our system does not allow to test the serial algorithm on larger problems, next we evaluate the scalability of the parallel algorithms. 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 $q/\sqrt{n_p} = 200$ for $q = q_1$, q_2 and q_3. In Figure 3 we report the Mflop ratio per node for the parallel algorithms PDGECMSR and PDGECMBS.

The figure shows a high scalability of the 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

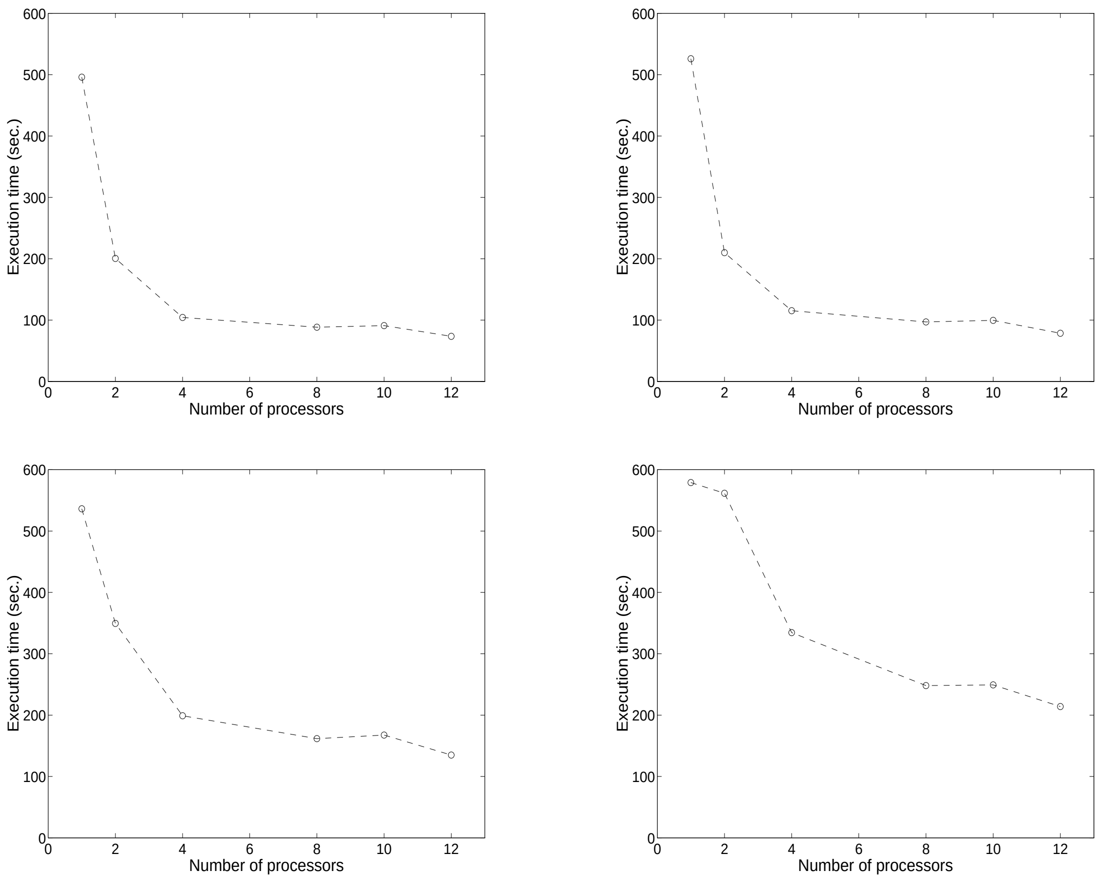

Figure 2: Execution time vs. number of nodes of the serial and parallel SR algorithms. Top left: $m = p = 100$, $q_1 = q_2 = q_3 = 50$; top right: $m = p = 100$, $q_1 = q_2 = q_3 = 98$; bottom left: $m = p = 200$, $q_1 = q_2 = q_3 = 100$; bottom right: $m = p = 400$, $q_1 = q_2 = q_3 = 200$.

size n=4000). The scalability confirms that a larger problem can be solved by increasing proportionally the number of nodes employed.

6 Concluding Remarks

In this paper we have described the design and use of parallel algorithms for model reduction of large-scale systems. The methods are based on balanced truncation methods and benefit from low numerical rank of the Cholesky factors of the Gramian matrices. Using these algorithms it is possible to obtain a low-order approximation of large dense systems. Employing these reduced-order models, more advanced model reduction techniques can be applied to further reduce the system order.

Our experiments report similar numerical results for reliable serial model reduction algorithms from the SLICOT library and our model reduction approach, based on the matrix sign function. The results on a cluster of Intel Pentium-II nodes show the performance of our model reduction approach and the scalability of the parallel algorithms.

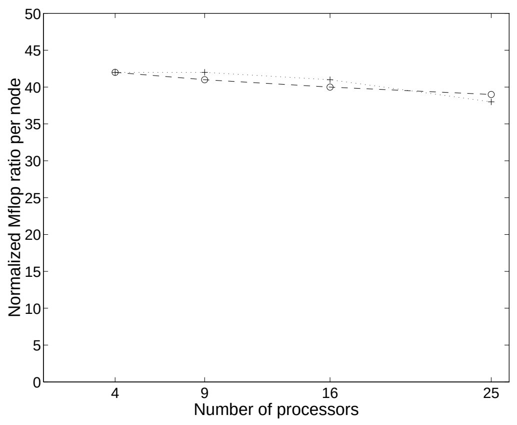

Figure 3: Scalability of the parallel algorithms with $n/\sqrt{n_p} = 800$, $m/\sqrt{n_p} = 400$, $p/\sqrt{n_p} = 400$, and $q/\sqrt{n_p} = 200$ for $q = q_1$, q_2 and q_3. Legend: Symbols "$--\text{o}--$" for PDGECMSR and "$\cdots + \cdots$" for PDGECMBS.

Acknowledgments

We would like to thank Thilo Penzl for many helpful discussions. Observations on the decay rate of Hankel singular values and their use in model reduction algorithms for systems with sparse coefficient matrices are reported in [35, 36].

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide.* SIAM, Philadelphia, PA, second edition, 1995.

[2] Z. Bai and J. Demmel. Design of a parallel nonsymmetric eigenroutine toolbox, Part I. In R.F. Sincovec et al, editor, *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, 1993. *See also:* Tech. Report CSD-92-718, Computer Science Division, University of California, Berkeley, CA 94720.

[3] J. Barlow. More accurate bidiagonal reduction for computing the singular value decompositon. Technical report, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802, USA, 1998. Available from `http://trantor.cse.psu.edu/~barlow/papers.html`.

[4] R.H. Bartels and G.W. Stewart. Solution of the matrix equation $AX + XB = C$: Algorithm 432. *Comm. ACM*, 15:820–826, 1972.

[5] P. Benner, J.M. Claver, and E.S. Quintana-Ortí. Efficient solution of coupled Lyapunov equations via matrix sign function iteration. In A. Dourado et al., editor, *Proc.* 3rd *Portuguese Conf. on Automatic Control CONTROLO'98,* Coimbra, pages 205–210, 1998.

[6] P. Benner, J.M. Claver, and E.S. Quintana-Ortí. Parallel distributed solvers for large stable generalized Lyapunov equations. *Parallel Processing Letters*, to appear.

[7] P. Benner, A.J. Laub, and V. Mehrmann. A collection of benchmark examples for the numerical solution of algebraic Riccati equations I: Continuous-time case. Technical Report SPC 95_22, Fakultät für Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1995. Available from `http://www.tu-chemnitz.de/sfb393/spc95pr.html`.

[8] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga. SLICOT - a subroutine library in systems and control theory. *Applied and Computational Control, Signals, and Circuits*, 1:505–546, 1999.

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

[10] C.H. Bischof. Incremental condition estimation. *SIAM J. Matrix Anal. Appl.*, 11(2):312–322, 1990.

[11] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R.C. Whaley. *ScaLAPACK Users' Guide.* SIAM, Philadelphia, PA, 1997.

[12] A. Bojanczyk, G.H. Golub, and P. Van Dooren. The periodic Schur decomposition; algorithms and applications. In *Proc. SPIE Conference, vol. 1770*, pages 31–42, 1992.

[13] D. Boley and R. Maier. A parallel QR algorithm for the unsymmetric eigenvalue problem. Technical Report TR-88-12, University of Minnesota at Minneapolis, Department of Computer Science, Minneapolis, MN, 1988.

[14] R. Byers. Solving the algebraic Riccati equation with the matrix sign function. *Linear Algebra Appl.*, 85:267–279, 1987.

[15] 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.

[16] 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.

[17] J.J. Dongarra, A. Sameh, and D. Sorensen. Implementation of some concurrent algorithms for matrix factorization. *Parallel Comput.*, 3:25–34, 1986.

[18] Z. Drmač. Accurate computation of the product-induced singular value decomposition with applications. *SIAM J. Numer. Anal.*, 35(5):1969–1994, 1998.

[19] K.V. Fernando and S.J. Hammarling. A product induced singular value decmoposition for two matrices and balanced realization. In B.N. Datta et al., editor, *Linear Algebra in Signals, Systems and Control*, pages 128–140. SIAM, 1988.

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

[21] G.A. Geist, R.C. Ward, G.J. Davis, and R.E. Funderlic. Finding eigenvalues and eigenvectors of unsymmetric matrices using a hypercube multiprocessor. In G. Fox, editor, *Proc. 3rd Conference on Hypercube Concurrent Computers and Appl.*, pages 1577–1582, 1988.

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

[23] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, third edition, 1996.

[24] S.J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA J. Numer. Anal.*, 2:303–323, 1982.

[25] M.T. Heath, A.J. Laub, C.C. Paige, and R.C. Ward. Computing the SVD of a product of two matrices. *SIAM J. Sci. Statist. Comput.*, 7:1147–1159, 1987.

[26] J.J. Hench and A.J. Laub. Numerical solution of the discrete-time periodic Riccati equation. *IEEE Trans. Automat. Control*, 39:1197–1210, 1994.

[27] G. Henry and R. van de Geijn. Parallelizing the *QR* algorithm for the unsymmetric algebraic eigenvalue problem: myths and reality. *SIAM J. Sci. Comput.*, 17:870–883, 1997.

[28] G. Henry, D.S. Watkins, and J.J. Dongarra. A parallel implementation of the nonsymmetric QR algorithm for distributed memory architectures. LAPACK Working Note 121, University of Tennessee at Knoxville, 1997.

[29] C. Kenney and A.J. Laub. The matrix sign function. *IEEE Trans. Automat. Control*, 40(8):1330–1348, 1995.

[30] W. Lang and U. Lezius. Numerical realization of the balanced reduction of a control problem. In H. Neunzert, editor, *Progress in Industrial Mathematics at ECMI94*, pages 504–512. John Wiley & Sons Ltd and B.G. Teubner, New York and Leipzig, 1996.

[31] V.B. Larin and F.A. Aliev. Construction of square root factor for solution of the Lyapunov matrix equation. *Sys. Control Lett.*, 20:109–112, 1993.

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

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

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

[35] T. Penzl. Algorithms for model reduction of large dynamical systems. Submitted for publication, 1999.

[36] T. Penzl. Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. Submitted for publication, 1999.

[37] 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).

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

[39] G.W. Stewart. A parallel implementation of the *QR* algorithm. *Parallel Computing*, 5:187–196, 1987.

[40] 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.

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

[42] A. Varga. Model reduction routines for SLICOT. NICONET Report 1999–8, The Working Group on Software (WGS), June 1999. Available from `http://www.win.tue.nl/niconet/NIC2/reports.html`.

99–07. Peter Benner, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí:
Balanced Truncation Model Reduction of Large-Scale Dense Systems on Parallel Computers, September 1999.