Parallel Model Reduction of Large-Scale Unstable Systems

Peter Benner*
Institut für Mathematik, MA 4-5
Technische Universität Berlin
D-10623 Berlin, Germany
benner@math.tu-berlin.de

Maribel Castillo† Enrique S. Quintana-Ortí† Gregorio Quintana-Ortí†
Depto. de Ingeniería y Ciencia de Computadores
Universidad Jaume I
12.071-Castellón, Spain
{castillo,quintana,gquintan}@icc.uji.es

Abstract

We discuss an efficient algorithm for model reduction of large-scale unstable systems and an implementation which allows to reduce models of order up to $\mathcal{O}(10^4)$ using parallel computing techniques. The major computational step involves the additive decomposition of a transfer function via a block diagonalization based on the solution of a Sylvester equation. The actual model reduction is then achieved by reducing the stable part using techniques based on balanced truncation, singular perturbation approximation, or optimal Hankel norm approximation. We will see that all core computational steps can be based on the sign function method. Numerical experiments on a cluster of Intel Pentium II processors show the efficiency of our methods.

Topic area: Section 2. Algorithms
Classic presentation

1 Introduction

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

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

with $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$, and $D \in \mathbb{R}^{p \times m}$. For simplicity we assume that the spectrum of A is dichotomic with respect to the imaginary axis, i.e., $Re(\lambda) \neq 0$ for all eigenvalues λ of A. This implies that the system (1) has no poles on the imaginary axis. The case with poles on the imaginary axis could be treated as well with the method described in

*Supported by the DFG Research Center "Mathematics for key technologies" (FZT 86) in Berlin.
†Supported by the project CTDIA/2002/122 of the *Generalitat Valenciana* and the project PI-1B2001-14 of the *Fundació Caixa-Castelló/Bancaixa*.

this paper, but this would add some distracting technicalities. Throughout this paper we will denote the spectrum of A by $\Lambda(A)$. The number of state variables n is called the *order* of the system.

We are interested in finding a reduced-order LTI system,

$$\begin{array}{rcll} \dot{\hat{x}}(t) & = & \hat{A}\hat{x}(t) + \hat{B}\hat{u}(t), & t > 0, \\ \hat{y}(t) & = & \hat{C}\hat{x}(t) + \hat{D}\hat{u}(t), & t \geq 0, \end{array} \tag{2}$$

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

A well-known approach for model reduction is based on state-space truncation or projected dynamics, see, e.g., the recent monographs [1, 19]. Methods based on truncated state-space transformations usually differ in the measurement of the approximation error and the way they attempt to minimize this error. Balanced truncation (BT) methods [18, 21, 23, 25], singular perturbation approximation (SPA) methods [17], and optimal Hankel-norm approximation (HNA) methods [13] all belong to the family of absolute error methods, which try to minimize $\|\Delta_a\| = \|G - \hat{G}\|$ for some system norm $\|\,.\,\|$. For BT and SPA methods, the error measure is $\|G\|_\infty$ where $\|\ .\ \|_\infty$ denotes the $\mathcal{L}_\infty$- or $\mathcal{H}_\infty$-norm of a stable, rational matrix function defined by

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

Here $\jmath := \sqrt{-1}$ and $\sigma_{\max}(M)$ is the largest singular value of the matrix M. Note that computing the best r-th order approximation to a transfer function in the $\mathcal{H}_\infty$-norm is an open problem in approximation theory, even in the scalar ($m = p = 1$) case [2]. HNA methods find the optimal solution to the absolute error minimization problem using the *Hankel norm* $\|\,.\,\|_H$, defined as

$$\|G\|_H = \max_{j=1,\ldots,n} \sigma_j,$$

where the σ_j are the Hankel singular values of the system.

All methods mentioned so far assume that the system is stable and therewith cannot be applied directly to unstable systems which occur quite often, in particular if stabilization of the system is the computational task to solve. If a stabilization strategy for a large-scale unstable system is to be designed, but the model is too large to be treated by the stabilization procedure, model reduction of the unstable plant model is needed. Unstable systems can also occur in controller reduction: controllers are often themselves unstable systems and therefore the task of controller reduction leads to model reduction of unstable systems [27]. Usually, unstable poles cannot be neglected when modeling the dynamics of the system, and therefore should be preserved in the reduced-order system in some sense. This is trivially satisfied using the following approach [22, 26]: first, compute an *additive decomposition* of the transfer function,

$$G(s) = G_-(s) + G_+(s) = \left(C_-(sI - A_-)^{-1}B_- + D_-\right) + \left(C_+(sI - A_+)^{-1}B_+ + D_+\right), \tag{4}$$

such that G_- is stable (has all its poles in the open left half plane) and G_+ is unstable (has all its poles in the open right half plane). Then any of the BT, SPA, or HNA model reduction methods can be applied to G_- in order to obtain a reduced-order system $\hat{G}_-$ and the reduced-order system is synthesized by

$$\hat{G}(s) = \hat{G}_-(s) + G_+(s). \tag{5}$$

Hence, the unstable part is preserved in the reduced-order system. This is advantageous in controller reduction where it is needed to guarantee the stabilization property of the controller. Of course, if the number of unstable poles is dominating, the potential for reducing the model is limited. But in most applications, in particular those coming from semi-discretization of parabolic or hyperbolic partial differential equations, the numbers of unstable poles is very low compared to the number of stable poles.

The outline of the paper is as follows: in the next section, we will show how an additive decomposition of a transfer matrix can be computed using sign function-based algorithms for block diagonalization of a general square matrix. In Section 3 we review the methods used to reduce the stable part of the system. These methods are also based on sign function computations and use low-rank factorizations of the system Gramians. Sign function-based methods allow very efficient and scalable implementations of the proposed algorithms. We will briefly discuss the implementation of the suggested procedure for model reduction of unstable systems in Section 4. The numerical examples reported in Section 5 on a cluster of Intel Pentium II processors reveal the performance of the parallel algorithms.

2 Additive decomposition of a transfer function

As the procedure described in this section can be used for an additive decomposition of any transfer function having no poles on the imaginary axis, and therefore has many other applications apart from model reduction of unstable systems, we present the method in terms of a generic realization (A, B, C, D) of a transfer function $G(s)$.

First, we describe the sign function method as it will serve as the major tool in the computations. Consider a matrix $Z \in \mathbb{R}^{n\times n}$ with $\Lambda(Z) \cap \imath\mathbb{R} = \emptyset$ and let $Z = S \begin{bmatrix} J^- & 0 \\ 0 & J^+ \end{bmatrix} S^{-1}$ be its Jordan decomposition. 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 half of the complex plane. 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 [15] 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:

$$Z_0 \leftarrow Z, \quad Z_{j+1} \leftarrow \frac{1}{2}(Z_j + Z_j^{-1}), \qquad j = 0, 1, 2, \ldots. \tag{6}$$

Under the given assumptions, the sequence $\{Z_j\}_{j=0}^{\infty}$ converges to $\operatorname{sign}(Z) = \lim_{j\to\infty} Z_j$ [20] with a locally quadratic convergence rate. As initial convergence may be slow, acceleration techniques are used, e.g., *determinantal scaling* [12]:

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

where $\det(Z_j)$ denotes the determinant of Z_j. For a survey of acceleration schemes, see [3].

Once we have computed sign (Z), we can obtain an orthogonal basis for the stable or anti-stable right invariant subspace of Z by computing a (rank-revealing) QR factorization of $I_n - \text{sign}\,(Z)$ or $I_n + \text{sign}\,(Z)$, respectively; e.g., let

$$I_n - \text{sign}\,(Z) = QRP, \quad R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \searrow & \square \\ & \end{bmatrix}, \quad R_{11} \in \mathbb{R}^{k \times k},$$

where P is a permutation matrix. Then

$$\tilde{Z} \;:=\; Q^T Z Q \;=\; \begin{bmatrix} Z_{11} & Z_{12} \\ 0 & Z_{22} \end{bmatrix} \tag{7}$$

where $\Lambda\,(Z_{11}) = \Lambda\,(Z) \cap\ \mathbb{C}^-$, $Z_{11} \in \mathbb{R}^{k \times k}$, $\Lambda\,(Z_{22}) = \Lambda\,(Z) \cap\ \mathbb{C}^-$, $Z_{22} \in \mathbb{R}^{n-k \times n-k}$; i.e., the similarity transformation defined by Q separates the stable and the unstable parts of the eigenspectrum of Z.

In a second step, we compute a matrix V such that

$$\hat{Z} \;:=\; V^{-1} \tilde{Z} V \;=\; \begin{bmatrix} Z_{11} & 0 \\ 0 & Z_{22} \end{bmatrix}.$$

Note that, under the given assumption,

$$V \;=\; \begin{bmatrix} I_k & Y \\ 0 & I_{n-k} \end{bmatrix},$$

where $Y \in \mathbb{R}^{k \times n-k}$ satisfies the *Sylvester equation*

$$Z_{11} Y - Y Z_{22} + Z_{12} = 0. \tag{8}$$

As $\Lambda\,(Z_{11}) \cap \Lambda\,(Z_{22}) = \emptyset$, equation (8) has a unique solution [16].

The Sylvester equation (8) can again be solved using a sign function-based procedure. For an equation

$$EY - YF + W = 0$$

with E and $-F$ stable matrices, this procedure, already derived in [20], can be formulated as follows

$$\begin{array}{lllll} E_0 \;:=\; E, & E_{j+1} \;:=\; \frac{1}{2}\left(E_j + E_j^{-1}\right), & & \\ F_0 \;:=\; \pm F, & F_{j+1} \;:=\; \frac{1}{2}\left(F_j + F_j^{-1}\right), & & j = 0, 1, 2, \ldots. \\ W_0 \;:=\; W, & W_{j+1} \;:=\; \frac{1}{2}\left(W_j + E_j^{-1} W_j F_j^{-1}\right), & & \end{array} \tag{9}$$

It follows that $\lim_{j\to\infty} E_j = -I_k$, $\lim_{j\to\infty} F_j = I_{n-k}$, and

$$Y \;=\; \frac{1}{2} \lim_{j\to\infty} W_j.$$

For an efficient implementation of this iteration on modern computer architectures and numerical experiments reporting efficiency and accuracy, see [6]. Having solved (8) with the scheme in (9) we obtain V. Note that $V^{-1} = \begin{bmatrix} I_k & -Y \\ 0 & I_{n-k} \end{bmatrix}$.

Applying the above steps to the matrix A from the realization of $G(s)$ we now obtain the desired additive decomposition of $G(s) = C(sI-A)^{-1}B+D$ as follows: perform a state space transformation

$$(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}) \;:=\; (V^{-1}Q^TAQV, V^{-1}Q^TB, CQV, D)$$

and partition

$$\tilde{A} = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \quad \tilde{C} = [\, C_1 \;\; C_2], \tag{10}$$

according to the partitioning in (7). Then

$$\begin{aligned} G(s) &= C(sI-A)^{-1}B + D = \tilde{C}(sI-\tilde{A})^{-1}\tilde{B} + \tilde{D} \\ &= \begin{bmatrix} C_1 & C_2 \end{bmatrix} \begin{bmatrix} (sI_k - A_{11})^{-1} & \\ & (sI_{n-k} - A_{22})^{-1} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} + D \\ &= \left\{C_1(sI_k - A_{11})^{-1}B_1 + D\right\} + \left\{C_2(sI_{n-k} - A_{22})^{-1}B_2\right\} \\ &=: G_-(s) + G_+(s), \end{aligned}$$

where $G_-(s)$ is a stable TFM and $G_+(s)$ is an unstable TFM.

3 Model reduction for stable systems

Here we briefly review the methods for absolute error model reduction of stable systems. If the original system is unstable, the procedure in the previous section allows to separate the stable and unstable parts, so that we consider hereafter $(A, B, C, D) := (A_{11}, B_1, C_1, D)$, the realization associated with the stable TFM $G_-(s)$. Serial implementations of the model reduction algorithms discussed here can be found in the *Subroutine Library in Control Theory* – SLICOT[1]. Our implementations are particularly efficient on parallel computing platforms and, although the methods are different from those employed in SLICOT, closely similar results are obtained by the reduced-order models computed using both approaches [5].

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*

$$\begin{aligned} AW_c + W_cA^T + BB^T &= 0, \\ A^TW_o + W_oA + C^TC &= 0. \end{aligned} \tag{11}$$

As A is assumed to be stable, the Gramians W_c and W_o are positive semidefinite, and therefore there exist factorizations $W_c = S^TS$ and $W_o = R^TR$. Matrices S and R are often 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 [20] and closely related to the iteration for Sylvester equations discussed in the previous section. This iteration is 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., [4, 8].

[1] Available from `http://www.win.tue.nl/niconet/NIC2/slicot.html`.

Consider now the singular value decomposition (SVD)

$$SR^T = [U_1\ U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}, \tag{12}$$

where the matrices are partitioned at a given dimension r such that $\Sigma_1 = \mathrm{diag}\,(\sigma_1, \ldots, \sigma_r)$, $\Sigma_2 = \mathrm{diag}\,(\sigma_{r+1}, \ldots, \sigma_n)$, $\sigma_j \geq \sigma_{j+1} \geq 0$ for all j, and $\sigma_r > \sigma_{r+1}$. Here, $\sigma_1, \ldots, \sigma_n$ are the *Hankel singular values* of the system.

The so-called *square-root* (SR) BT algorithms determine the reduced-order model as

$$\begin{array}{rclcrcl} \hat{A} & = & T_l A T_r, & & \hat{B} & = & T_l B, \\ \hat{C} & = & C T_r, & & \hat{D} & = & D, \end{array} \tag{13}$$

using the projection matrices

$$T_l = \Sigma_1^{-1/2} V_1^T R \quad \text{and} \quad T_r = S^T U_1 \Sigma_1^{-1/2}. \tag{14}$$

The method described above can also be used to compute a minimal realization of the given LTI system by partitioning according to $\Sigma_1 > 0$ and $\Sigma_2 = 0$. (Computing a minimal realization is the first step in SPA model reduction as described below.)

The *balancing-free square-root* (BFSR) BT algorithms often provide more accurate reduced-order models in the presence of rounding errors [25]. These algorithms share the first two stages (solving the coupled Lyapunov 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,

$$S^T U_1 = [P_1\ P_2] \begin{bmatrix} \hat{R} \\ 0 \end{bmatrix}, \qquad R^T V_1 = [Q_1\ Q_2] \begin{bmatrix} \bar{R} \\ 0 \end{bmatrix}, \tag{15}$$

where the columns of P_1, $Q_1 \in \mathbb{R}^{n \times r}$ form orthonormal bases for range(T_r), range(T_l), respectively, and $\hat{R}$, $\bar{R} \in \mathbb{R}^{r \times r}$ are upper triangular. The reduced system is then given by the projection matrices

$$T_l = (Q_1^T P_1)^{-1} Q_1^T, \qquad T_r = P_1, \tag{16}$$

and (13).

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 $(\bar{A}, \bar{B}, \bar{C}, \bar{D})$ be a realization of this minimal system and partition

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

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

$$\begin{array}{rcl} \hat{A} & := & A_{11} - A_{12}(\gamma I - A_{22})^{-1} A_{21}, \\ \hat{B} & := & B_1 - A_{12}(\gamma I - A_{22})^{-1} B_2, \\ \hat{C} & := & C_1 - C_2(\gamma I - A_{22})^{-1} A_{21}, \\ \hat{D} & := & \bar{D} - C_2(\gamma I - A_{22})^{-1} B_2, \end{array} \tag{17}$$

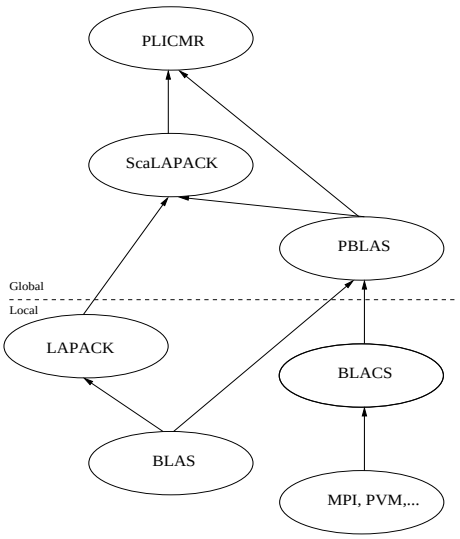

Figure 1: PLiCMR and the underlying libraries.

where $\gamma = 0$ for continuous LTI systems [17, 24, 25]. Versions of (17) suitable for distributed memory computing are discussed in [9].

The derivation of a reduced realization using the HNA method is quite involved. One key step in the method is an additive decomposition of a TFM as discussed in the previous section. Due to space limitations, we refer the reader to [13, 19, 29] for detailed derivations of the method.

4 Implementation details

The additive decomposition and the model reduction methods basically require 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.). The iterative algorithms for efficiently solving linear matrix equations derived from the matrix sign function (see the previous sections and [6, 4, 7]) only require operations like matrix products and matrix inversion. All these operations are basic dense matrix algebra kernels parallelized in ScaLAPACK and PBLAS. Thus, the parallel model reduction routines, integrated into the PLiCMR library[2], heavily rely 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 the model reduction routines for stable systems are given, e.g., in [8, 10].

A standardized version of the library is integrated into the subroutine library PSLICOT [11], with parallel implementations of a subset of SLICOT. It can be downloaded from the URI `ftp://ftp.esat.kuleuven.ac.be/pub/WGS/SLICOT`. However, it is recommended to obtain the version from `http://spine.act.uji.es/~plicmr` as it 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

[2]See `http://spine.act.uji.es/~plicmr`

performance of the underlying libraries for matrix computation (BLAS) and communication (usually, MPI or PVM).

5 Numerical experiments

All the experiments presented in this section were performed on a cluster of 32 nodes using IEEE double-precision floating-point arithmetic ($\varepsilon \approx 2.2204 \times 10^{-16}$). Each node consists of an Intel Pentium-II processor at 300 MHz with 128 MBytes of RAM running the Linux (SuSE 7.3) operating system. We employ a BLAS library, specially tuned for the Pentium-II processor as part of the ITXGEMM and ATLAS projects [14, 28], that achieves around 180 Mflops (millions of flops per second) for the matrix product (routine `DGEMM`). The nodes are connected via a *Myrinet* multistage 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 whenever possible.

In this section we report the performance of the major parallel routines involved in the computation of the additive decomposition of a TFM. In particular, we report results for one iteration of the following two algorithms:

- `pdgesine`: The Newton iteration for the matrix sign function with determinantal scaling as in (6).
- `pdgecsne`: Solution of a Sylvester equation using the modified Newton iteration for the matrix sign function in (9).

In the experiments we employ systems of order n with 10% of unstable poles and random entries in $\mathrm{U}[0, 1]$. As we only report results for one iteration in this subsection, system matrices generated in a different manner should obtain similar performance results.

Our first experiment reports the execution time of one iteration of the parallel routines on a system of dimension $n = 1500$. This is about the largest size we could solve on a single node of our cluster, considering the number of data matrices involved, the amount of workspace necessary for computations, and the reduced size of the RAM per node. The left-hand plot in Figure 2 reports the execution time of one iteration of the parallel algorithms using n_p=1, 2, 4, 6, 8, and 10 nodes. The execution of the parallel algorithm on a single node is likely to require a higher time than that of a serial implementation of the algorithm (using, e.g., LAPACK and BLAS); however, at least for such large scale problems, we expect this overhead to be negligible compared to the overall execution time.

The figure shows reasonable speed-ups when a reduced number of processors is employed. Thus, e.g., when n_p=2, speed-ups of 1.51 and 1.62 are obtained for routines `pdgesine` and `pdgecsne`. In both cases the efficiency decreases as n_p gets larger (as the system dimension is fixed, the problem size per node is reduced) so that using more than a few processors does not achieve a significant reduction in the execution time for such a small problem.

We next evaluate the scalability of the parallel algorithms when the problem size per node is constant. For that purpose, we fix the problem dimensions to $n/\sqrt{n_p} = 1500$, and report the Mflops per node. The right-hand plot in Figure 2 shows the Mflop rate per node of one iteration of the parallel routines. These results demonstrate the scalability of our parallel

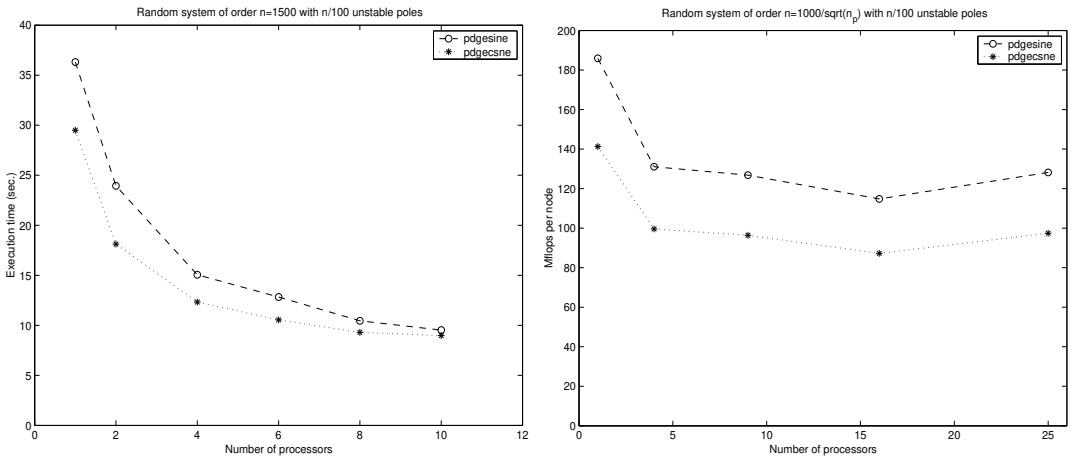

Figure 2: Performance (of one iteration) of the basic parallel routines for the computation of the additive decomposition of a TFM.

kernels, as there is only a minor decrease in the performance of the algorithms when n_p is increased while the problem dimension per node remains fixed.

6 Conclusions

We have presented an efficient approach for model reduction of large-scale unstable systems on parallel computers. All computational steps in the approach are solved using iterative algorithms derived from the matrix sign function, which has been shown to offer a high degree of parallelism. Numerical experiments on a cluster of Intel Pentium II processors confirm the efficiency and scalability of our methods. Analogous routines for model reduction of unstable large-scale discrete-time systems are under current development.

References

[1] A.C. Antoulas. *Lectures on the Approximation of Large-Scale Dynamical Systems*. SIAM Publications, Philadelphia, PA, to appear.

[2] A.C. Antoulas and A. Astolfi. H_∞-norm approximation. In V.D. Blondel and A. Megretski, editors, *2002 MTNS Problem Book, Open Problems on the Mathematical Theory of Networks and Systems*, pages 73–76. 2002. Available online from `http://www.nd.edu/~mtns/OPMTNS.pdf`.

[3] 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*, pages 391–398. SIAM, Philadelphia, PA, 1993. *See also:* Tech. Report CSD-92-718, Computer Science Division, University of California, Berkeley, CA 94720.

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

[5] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí and. Experimental evaluation of the parallel model reduction routines in PSLICOT. SLICOT Working Note 2002-7, July 2002. Available from `http://www.win.tue.nl/niconet/NIC2/reports.html`.

[6] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Solving linear matrix equations via rational iterative schemes. In preparation.

[7] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Solving linear and quadratic matrix equations on distributed memory parallel computers. In O. Gonzalez, editor, *Proc. 1999 IEEE Intl. Symp. CACSD, Kohala Coast-Island of Hawai'i, Hawai'i, USA, August 22–27, 1999* (CD-Rom), pages 64–69, 1999.

[8] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Balanced truncation model reduction of large-scale dense systems on parallel computers. *Math. Comput. Model. Dyn. Syst.*, 6(4):383–405, 2000.

[9] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Singular perturbation approximation of large, dense linear systems. In *Proc. 2000 IEEE Intl. Symp. CACSD, Anchorage, Alaska, USA, September 25–27, 2000*, pages 255–260. IEEE Press, Piscataway, NJ,, 2000.

[10] 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.

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

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

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

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

[16] P. Lancaster and M. Tismenetsky. *The Theory of Matrices*. Academic Press, Orlando, 2nd edition, 1985.

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

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

[19] G. Obinata and B.D.O. Anderson. *Model Reduction for Control System Design*. Communications and Control Engineering Series. Springer-Verlag, London, UK, 2001.

[20] 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).

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

[22] M.G. Safonov, E.A. Jonckheere, M. Verma, and D.J.N. Limebeer. Synthesis of positive real multivariable feedback systems. *Internat. J. Control*, 45(3):817–842, 1987.

[23] 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.

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

[25] 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.

[26] A. Varga. Task II.A.1 – selection of model reduction routines. SLICOT Working Note 1998–2, The Working Group on Software (WGS), June 1998. Available from `http://www.win.tue.nl/niconet/NIC2/reports.html`.

[27] A. Varga. Task II.B.1 – selection of software for controller reduction. SLICOT Working Note 1999–18, The Working Group on Software (WGS), December 1999. Available from `http://www.win.tue.nl/niconet/NIC2/reports.html`.

[28] R. C. Whaley and J. Dongarrra. Automatically tuned linear algebra software. In *Proc. of SC98*, 1998.

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