Parallel Algorithms for Model Reduction of Discrete–Time Systems

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

Report 00–18

Berichte aus der Technomathematik

Report 00–18 Dezember 2000

Parallel Algorithms for Model Reduction of Discrete-Time Systems*

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

January 24, 2001

Abstract

Computing reduced-order models of controlled dynamical systems is of fundamental importance in many analysis and synthesis methods in systems and control theory. Here we address algorithmic aspects of model reduction methods based on balanced truncation of linear discrete-time systems. We also consider singular perturbation approximation methods. In contrast to the often used approach of applying methods for continuous-time systems to discrete-time models employing a bilinear transformation, we devise special algorithms for discrete-time systems that are more efficient. All methods require in an initial stage the computation of the Gramians of the system. Using an accelerated fixed point iteration for computing the full-rank factors of the Gramians yields some favorable computational aspects, particularly for non-minimal systems. The computations only require efficient implementations of basic linear algebra operations readily available on modern computer architectures. We discuss aspects of the parallel implementation of these methods and show the performance and scalability on distributed memory computers. Our approach enables users to deal with very complex systems using relatively cheap infrastructure, as, e.g., a local PC or workstation network.

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 discrete, linear time-invariant (LTI) system,

$$x_{k+1} = Ax_k + Bu_k, \quad y_k = Cx_k + Du_k, \qquad k = 0, 1, 2, \ldots, \tag{1}$$

where $x_0 = \hat{x}$ is given and $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$. The number of state variables n is said to be the order of the system. We assume that the spectrum of A, denoted by $\Lambda(A)$, is contained in the open unit disk, i.e., A is *(Schur) stable* or *convergent*. This implies that all the poles of the TFM $G(s)$ are contained in the open unit disk and hence the stability of the system (1).

*Partially supported by the DAAD programme *Acciones Integradas Hispano-Alemanas*. Enrique S. Quintana-Ortí and Gregorio Quintana-Ortí's research is part of the project GV99-59-1-14 funded by the *Consellería de Cultura y Educación de la Generalidad Valenciana*.

†Zentrum für Technomathematik, Fachbereich 3/Mathematik und Informatik, Universität Bremen, 28334 Bremen, Germany. E-mail: `benner@math.uni-bremen.de`

‡Departamento de Informática, Universidad Jaime I, 12.080–Castellón, Spain. E-mails: `{quintana,gquintan}@inf.uji.es`

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

$$\tilde{x}_{k+1} \;=\; \tilde{A}\tilde{x}_k + \tilde{B}\tilde{u}_k, \quad \tilde{y}_k \;=\; \tilde{C}\tilde{x}_k + \tilde{D}\tilde{u}_k, \qquad k = 0, 1, 2, \dots, \tag{2}$$

of order ℓ, $\ell \ll n$, with $\tilde{x}_0 = \tilde{\bar{x}}$, such that $\tilde{G}(\lambda) = \tilde{C}(\lambda I - \tilde{A})^{-1}\tilde{B} + \tilde{D}$ approximates $G(\lambda)$. This problem arises whenever a dynamic system of high complexity is to be analyzed or a control system is to be synthesized. Often, models that are derived by interconnecting several sub-models contain redundancies, in particular if automatic tools for generating physical models such as DYMOLA or MODELICA [5, 6] are used. Usually, it is necessary to remove such redundancies: even if they would not influence the systems dynamics in theory, this may no longer be true if finite-precision arithmetic is used to simulate the behavior of the system. When simulating flexible mechanical structures, high-order finite-element models are necessary to obtain accurate models. Depending on the operation mode, high accuracy at low frequencies is often sufficient. Usually, this can be achieved using low-order models; see, e.g., [30]. In VLSI design, due to the ever increasing complexity, it is nowadays necessary to replace circuit or network models by low-order models in order to keep simulation times small enough; see, e.g., [7] and the references therein. In synthesis problems, often the complexity of a controller is of the same order as that of the model, e.g., when a dynamic compensator is used to control the system. As both the cost of hardware and the computation time grow steeply while the reliability of the controller decreases when its order is increased, usually the order has to be limited to $\mathcal{O}(10)$; see, e.g., [23] for some discussion of the competing demands in controller hardware.

A large class of model reduction methods rely on similarity transformations of the state-space model with a non-singular matrix Y. These lead to projection methods as follows: for $Y = \left[T_l^T, L_l^T\right]^T \in \mathbb{R}^{n\times n}$ 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 defined as $\tilde{A} = T_l A T_r$, $\tilde{B} = T_l B$, $\tilde{C} = C T_r$, and $\tilde{D} = D$. In this paper we will focus on these methods.

There is no general technique for model reduction that can be considered as optimal in a general sense since the reliability, performance, and adequacy of the reduced-order system strongly depend on the system characteristics. Model reduction methods usually differ in the measure they attempt to minimize. The methods considered here are based on balanced truncation (BT) methods [17, 25]. Additionally, we consider singular perturbation approximation (SPA) methods [16] based on a minimal realization of the system computed with a BT method. They all belong to the family of absolute error methods, which try to minimize $\|\Delta_a\|_\infty = \|G - \tilde{G}\|_\infty$. Here, $\|G\|_\infty$ denotes the $\mathcal{L}_\infty$- or $\mathcal{H}_\infty$-norm of a stable, rational matrix function which is defined as

$$\|G\|_\infty = \operatorname*{ess\ sup}_{\omega\in[-\pi,\pi)} \sigma_{\max}(G(e^{\jmath\omega})), \tag{3}$$

where $\jmath := \sqrt{-1}$ and $\sigma_{\max}(M)$ is the largest singular value of the matrix M [11].

BT model reduction methods are based on information retrieved from the controllability and observability Gramians W_c and W_o, respectively, of the system (1). These are given by the solutions of two "coupled" (as they share the same coefficient matrix A) *Stein equations* (or *discrete Lyapunov equations*)

$$A W_c A^T - W_c + B B^T = 0, \qquad A^T W_o A - W_o + C^T C = 0. \tag{4}$$

It should be noted here that the above linear matrix equations re-present systems of linear equations with $n(n+1)/2$ (exploiting symmetry) unknowns. For many applications where discrete or discretized LTI systems of the form (1) occur, n can be anywhere from a few hundreds to a couple of thousands. Problems of size larger than $n = 1000$ can often no longer be handled on desktop computers. One possibility to deal with such problems is to use advanced computer architectures like distributed memory parallel computers (e.g., workstation or PC clusters) in order to compute a reduced-order model that can be dealt with on a single processor machine. This will be the approach pursued in this paper. Note that such an approach is in principle feasible whenever several desktop computers are connected in a (local) network.

As A is assumed to be stable, W_c and W_o in (4) 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 often called the *Cholesky factors* of the Gramians, even if they are not (upper or lower) triangular. These factors are usually computed using Hammarling's method [13, 14, 27], yielding $S, R \in \mathbb{R}^{n\times n}$ upper triangular. Numerically reliable model reduction methods use these Cholesky factors rather than the Gramians themselves; see, e.g., [25, 28, 29]. The condition number of the solution matrix can be up to the square of that of its Cholesky factor. Hence, a significant increase in accuracy can often be observed working with the factor if the solution matrix is ill-conditioned.

However, if the system is not minimal, then W_c and/or W_o are singular and the Cholesky factors computed by Hammarling's method are not full-rank matrices. In particular, for large systems, it can often be observed that the *numerical rank* of the Cholesky factors is much less than n.

We therefore investigate in Section 2 a method based on an accelerated fixed point iteration that computes *full-rank factorizations* $W_c = \hat{S}^T \hat{S}$ and $\hat{W}_o = \hat{R}^T \hat{R}$, i.e., $\hat{S} \in \mathbb{R}^{\mathrm{rank}(W_c)\times n}$, $\hat{R} \in \mathbb{R}^{\mathrm{rank}(W_o)\times n}$. This can save a significant amount of computational cost and workspace solving the Stein equations and particularly in the subsequent computations for obtaining the reduced-order model. This approach also has some advantages regarding numerical robustness when determining the McMillan degree and a minimal realization of an LTI system.

We then describe in Section 3 model reduction methods that use these full-rank factors. The methods can be regarded as efficient variants of square-root and balancing-free square-root truncation methods. Using any of these methods to compute a minimal realization of the system, the singular perturbation approximation formulae can be applied to obtain a reduced-order model with zero steady-state error.

The parallel implementation of these model reduction methods is described in Section 4. Moreover, numerical examples reporting the accuracy and performance of the resulting routines on serial and parallel computers are provided. Using parallel computers with distributed memory such as Linux-PC or workstation clusters allows the application of our methods to systems of order up to $n = \mathcal{O}(10^5)$.

2 Computing the Gramians

Consider the Stein equations in (4). Both can be formulated in a fixed point form $X = FXF^T + G$, from which it is straightforward to derive the fixed point iteration

$$X_0 := G, \qquad X_{k+1} := FX_kF^T + G, \quad k = 0, 1, 2, \ldots.$$

This iteration converges to X if $\rho(F) < 1$, where $\rho(F)$ denotes the spectral radius of F. That is, convergence is guaranteed under the given assumptions. The convergence rate of this iteration is linear. A quadratically convergent version of the fixed point iteration is suggested in [24]. Setting $X_0 := G$, $F_0 := F$, this iteration can be written as

$$X_{k+1} := F_k X_k F_k^T + X_k, \quad F_{k+1} := F_k^2, \qquad k = 0, 1, 2, \ldots. \tag{5}$$

The above iteration is referred to as the *squared Smith iteration.* As the two equations in (4) share the same coefficient matrix A, we can derive a coupled iteration to solve both equations simultaneously,

$$\begin{array}{llllll} X_0 & := BB^T, & Y_0 & := C^T C, & A_0 & := A, \\ X_{k+1} & := A_k X_k A_k^T + X_k, & Y_{k+1} & := A_k^T Y_k A_k + Y_k, & A_{k+1} & := A_k^2, \qquad k = 0, 1, 2, \ldots. \end{array} \tag{6}$$

The most appealing feature of the squared Smith iteration regarding its implementation is that all the computational cost comes from matrix products. These can be efficiently implemented on modern serial and parallel computers [4].

The convergence theory of the Smith iteration (5) derived in [24] yields that for $\rho(F) < 1$ there exist real constants $0 < \mu$ and $0 < \rho < 1$ such that $\|X - X_k\|_2 \leq \mu \|G\|_2 (1-\rho)^{-1} \rho^{2^k}$. This shows that the method converges for all equations with Schur stable coefficient matrices F. Nevertheless, if the coefficient matrix F is highly non-normal such that $\|F\|_2 > 1$, then overflow may occur in the early stages of the iteration due to increasing $\|F_k\|_2$ although eventually, $\lim_{k\to\infty} F_k = 0$. Easily computable overflow bounds are derived in [2]. In case overflow occurs one can switch to a sign function based solution of the Stein equations that allows a similar efficient implementation, see [3, 2].

In the case considered here, the "right-hand sides" of the Stein equations are positive semidefinite and are given in factored form BB^T and $C^T C$. As A is stable, the Lyapunov stability theory (see, e.g., [15]) shows that the solution matrices are positive semidefinite and hence can be factored as $W_c = S^T S$, $W_o = R^T R$ with $S \in \mathbb{R}^{s\times n}$ and $R \in \mathbb{R}^{r\times n}$. If $s = r = n$ such that S, R are square, possibly singular, matrices, then these factors are called the Cholesky factors of the solutions. Here we consider *full-rank factors* of the solution, i.e., $\operatorname{rank}(S) = \operatorname{rank}(W_c) = s \leq n$, $\operatorname{rank}(R) = \operatorname{rank}(W_o) = r \leq n$. In particular, if $s, r \ll n$, we will show that significant savings in computational work are obtained by using the full-rank factors rather than the square Cholesky factors for subsequent computations.

The coupled squared Smith iteration (6) can be modified to compute the full-rank factors of W_c, W_o directly; see [2]. We focus here on the iteration for computing W_c; the iteration for W_o can be treated analogously. Setting $G = BB^T$, the X_k iteration in (6) can be re-written by setting $S_0 := B$ and

$$S_{k+1} S_{k+1}^T \quad \leftarrow \quad S_k S_k^T + A_k (S_k S_k^T) A_k^T = [\, S_k, \; A_k S_k \,] \begin{bmatrix} S_k^T \\ S_k^T A_k^T \end{bmatrix}, \quad \text{for } k = 0, 1, 2, \ldots. \tag{7}$$

In each step (7) the current iterate S_k is augmented by $A_k S_k$ such that $S_{k+1} := [S_k, \; S_k A_k]$. The computational cost for the k-th iteration step of such a procedure is $2(2^k m) n^2$ flops (floating-point arithmetic operations), where m is the number of columns of B. This compares to $3n^3$ flops for each iteration step for the X_k in (6).

The above approach requires to double in each iteration step the workspace needed for the iterates L_k. Two approaches are possible to limit the required workspace to a fixed size

[2]. We will focus here on one approach which is particularly appealing for the purpose of model reduction.

In each iteration step, we can compute a rank-revealing LQ factorization (see, e.g., [10, Chapter 5]) $\tilde{S}_{k+1} := [\, S_k, \;\; S_k A_k \,] = Q \hat{S}_{k+1} \Pi_{k+1}^T$. In that case, the next iterate $S_{k+1} \in \mathbb{R}^{n \times \mathrm{rank}(\tilde{S}_{k+1})}$ is obtained as the left $n \times \mathrm{rank}(\tilde{S}_{k+1})$ part of the product of the permutation matrix Π_{k+1} and the lower triangular matrix $\hat{S}_{k+1}$, i.e.,

$$\left[\begin{array}{cc} (\Pi_{k+1})_{11} & (\Pi_{k+1})_{12} \\ (\Pi_{k+1})_{21} & (\Pi_{k+1})_{22} \end{array} \right] \left[\begin{array}{cc} (\hat{S}_{k+1})_{11} & 0 \\ (\hat{S}_{k+1})_{21} & 0 \end{array} \right] =: [\, S_{k+1}, \; 0 \,],$$

starting from S_0 obtained by a rank-revealing LQ factorization of B. It follows that $\tilde{S}_{k+1}\tilde{S}_{k+1}^T = S_{k+1}S_{k+1}^T$ and $\sqrt{2}S = (\lim_{k\to\infty} S_k)^T$.

As $\mathrm{rank}(W_c)$ may be up to n, this requires a work space of size up to $2n \times n$. On the other hand, the (numerical) rank of the full-rank factors is often much less than n [19, 20]. Hence, the computational cost of performing LQ factorizations is bounded by keeping the number of columns of S_{k+1} small. If $m \ll n$ and the numerical rank of $\mathrm{rank}(W_c)$ is small, then the computational cost of this approach is usually less than the cost of Hammarling's method [14]. This approach can also be used to compute low-rank approximations to the full-rank factor by either increasing the tolerance threshold for determining the numerical rank or by fixing the allowed number of columns in S_k.

In the next section we show how the full rank-factors computed in the way described above can be used in model reduction algorithms for discrete-time systems.

3 Model Reduction using Full-Rank Factors

3.1 Model reduction methods based on balanced truncation

In [25] it is shown that BT model reduction can be achieved using SR^T instead of the product of the Gramians themselves. Here, S and R denote the square, possibly singular Cholesky factors of W_c and W_o, respectively. The resulting *square-root (SR) method* avoids working with the Gramians as their condition number can be up to the square of the condition number of the Cholesky factors. The first step in the SR method is to compute the SVD

$$SR^T = [U_1 \; U_2] \left[\begin{array}{cc} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{array} \right] \left[\begin{array}{c} V_1^T \\ V_2^T \end{array} \right]. \tag{8}$$

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

$$\sigma_1 \geq \sigma_2 \geq \ldots \sigma_\ell > \sigma_{\ell+1} \geq \sigma_{\ell+2} \geq \ldots \geq \sigma_n \geq 0. \tag{9}$$

If $\sigma_\ell > 0$ and $\sigma_{\ell+1} = 0$, i.e., $\Sigma_2 = 0$, then ℓ is the *McMillan* degree of the given discrete-time LTI system. That is, ℓ is the state-space dimension of a *minimal realization* of the system, i.e., a realization of $G(\lambda)$ in the form of an LTI system (1) of minimal order.

For model reduction, ℓ 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 $\sigma_\ell \gg \sigma_{\ell+1}$ [25].

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 $\mathrm{rank}(S) < n$ and/or $\mathrm{rank}(R) < n$. Hence, rather than working with the Cholesky factors, we may use the full-rank factors $\hat{S}$, $\hat{R}$ of W_c, W_o that are computed when using the method described in the last section. The SVD in (8) can then be obtained from that of $\hat{S}\hat{R}^T$ as follows. Here we assume $\mathrm{rank}(\hat{S}) =: s \geq r := \mathrm{rank}(\hat{R})$; the case $s < r$ can be treated analogously. Then we can compute the SVD

$$\hat{S}\hat{R}^T \;=\; \hat{U} \begin{bmatrix} \hat{\Sigma} \\ 0 \end{bmatrix} \hat{V}^T, \qquad \hat{\Sigma} = \mathrm{diag}\,(\sigma_1, \ldots, \sigma_r)\,, \tag{10}$$

where $\hat{U} \in \mathbb{R}^{s \times s}$, $\hat{V} \in \mathbb{R}^{r \times r}$. Now let $\ell \leq r$ be the order of the reduced-order (or minimal) system. Partitioning $\hat{U} = \begin{bmatrix} \hat{U}_1 & \hat{U}_2 \end{bmatrix}$ such that $\hat{U}_1 \in \mathbb{R}^{s \times \ell}$, $\hat{U}_2 \in \mathbb{R}^{s \times s-\ell}$, $\hat{V}_1 \in \mathbb{R}^{r \times \ell}$, $\hat{V}_2 \in \mathbb{R}^{r \times r-\ell}$, $\hat{\Sigma}_1 = \mathrm{diag}\,(\sigma_1, \ldots, \sigma_\ell)$, and $\hat{\Sigma}_2 = \mathrm{diag}\,(\sigma_{\ell+1}, \ldots, \sigma_r)$, the SVD of SR^T is given by

$$SR^T \;=\; \left[\begin{array}{c|cc} \hat{U}_1 & \hat{U}_2 & 0 \\ \hline 0 & 0 & I_{n-s} \end{array}\right] \left[\begin{array}{c|cc} \hat{\Sigma}_1 & 0 & 0 \\ \hline 0 & \hat{\Sigma}_2 & 0 \\ 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c|cc} \hat{V}_1 & \hat{V}_2 & 0 \\ \hline 0 & 0 & I_{n-s} \end{array}\right]^T . \tag{11}$$

We will see that all the subsequent computations can also be performed just working with $\hat{U}_1$, $\hat{\Sigma}_1$, and $\hat{V}_1$ rather than using the data from the full-size SVD in (11). This amounts in significant savings of workspace and computational cost. For example, using the Golub–Reinsch SVD (see, e.g., [10]), (8) requires $22n^3$ flops and workspace for $2n^2$ real numbers if U, V are to be formed explicitly while (10) only requires $14sr^2 + 8r^3$ flops and workspace for $s^2 + r^2$ 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. Suppose that (numerically) $s = r = n/10$, then the computation of (10) is 1000 times less expensive than that of (8) and only 1% of the workspace is required for (10) as compared to (8).

Defining

$$T_l = \Sigma_1^{-1/2} V_1^T R = \hat{\Sigma}_1^{-1/2} \hat{V}_1^T \hat{R} \qquad \text{and} \qquad T_r = S^T U_1 \Sigma_1^{-1/2} = \hat{S}^T \hat{U}_1 \hat{\Sigma}_1^{-1/2}, \tag{12}$$

the reduced-order system (2) is given by

$$\tilde{A} = T_l A T_r, \quad \tilde{B} = T_l B, \quad \tilde{C} = C T_r, \quad \text{and} \quad \tilde{D} = D. \tag{13}$$

In case that $\Sigma_1 > 0$ and $\Sigma_2 = 0$, (13) is a minimal realization of the TFM $G(\lambda)$ [25, 31]. Hence, choosing ℓ in (8) maximal such that $\sigma_\ell > 0$ and $\sigma_{\ell+1} = 0$, this procedure can be used to compute minimal realizations if the decision "$\sigma_{\ell+1} = 0$" is based on a numerically reliable criterion. It can further be proved that for a stable LTI system, choosing any partitioning in (8) such that $\sigma_\ell > \sigma_{\ell+1}$ yields a stable, minimal, and balanced reduced-order model. The Gramians corresponding to the resulting TFM $\tilde{G}(\lambda)$ are both equal to Σ_1. An important feature of this model reduction technique is that it is endowed with a computable error bound. It is shown in [9] that

$$\|G - \tilde{G}\|_\infty \leq 2 \sum_{k=\ell+1}^{n} \sigma_k =: \delta. \tag{14}$$

The reduced-order model in (13) computed by the SR method is balanced. Hence the projection matrices in (12) tend to be ill-conditioned if the original system is highly unbalanced, resulting in inaccurate reduced-order models. An alternative here are *balancing-free (BF) methods* [22] for which the reduced-order model is not balanced. The *balancing-free square-root (BFSR) method* combines the best characteristics of the SR and BF approaches [28, 29]. It shares the first two steps (solving the equations in (4) for the Cholesky factors and computing the SVD in (8)) with the SR method described above. Then, two "skinny" QR factorizations are computed,

$$S^T U_1 = \hat{S}^T \hat{U}_1 = P T_S, \qquad R^T V_1 = \hat{R}^T \hat{V}_1 = Q T_R,$$

where $P,\ Q \in \mathbb{R}^{n\times\ell}$ have orthonormal columns and $T_S,\ T_R \in \mathbb{R}^{\ell\times\ell}$ are upper triangular. Taking into account that $T_l T_r = I_\ell$, the reduced-order system is then given as in (13) with the projection matrices defined by $T_l = (Q^T P)^{-1} Q^T$ and $T_r = P$. The reduced-order model computed by the BFSR method shares the properties with the model obtained from the SR method except that it is not balanced. In particular, the error bound (14) also holds here.

We have implemented only the SR and BFSR algorithms as the BF algorithm described in [22] 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 than 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 [28, 29]. Furthermore, both SR and BFSR algorithms can be efficiently parallelized while the BF method needs a parallelized version of the QR algorithm with re-ordering of eigenvalues. This presents severe implementation difficulties; see, e.g., [2] for a discussion of this topic. Implementation details as well as accuracy and performance details are reported in the next sections.

3.2 Singular perturbation approximation

If for the reduced-order model, small steady-state error is required, then this can be achieved by a *singular perturbation approximation* (SPA) of the original system. That is, with SPA it is possible to compute a reduced-order system such that its frequency response at $\omega = 0$ equals that of the original system. In other words,

$$\tilde{G}(e^{\jmath\cdot 0}) = \tilde{G}(1) = G(1) = G(e^{\jmath\cdot 0}). \tag{15}$$

The SPA reduced-order model can be computed as follows [16, 28]. For any nonsingular matrix T, let

$$\begin{aligned} T^{-1}AT &=: \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \\ T^{-1}B &=: \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \quad CT =: \begin{bmatrix} C_1 & C_2 \end{bmatrix}, \end{aligned} \tag{16}$$

with $A_{11} \in \mathbb{R}^{\ell\times\ell}$, $B_1 \in \mathbb{R}^{\ell\times m}$, and $C_1 \in \mathbb{R}^{p\times\ell}$. In case the original system is minimal, a singular perturbation approximation (SPA) is then given by

$$\begin{aligned} \tilde{A} &:= A_{11} + A_{12}(I_{n-\ell} - A_{22})^{-1}A_{21}, \\ \tilde{B} &:= B_1 + A_{12}(I_{n-\ell} - A_{22})^{-1}B_2, \\ \tilde{C} &:= C_1 + C_2(I_{n-\ell} - A_{22})^{-1}A_{21}, \\ \tilde{D} &:= D + C_2(I_{n-\ell} - A_{22})^{-1}B_2. \end{aligned} \tag{17}$$

If the original system is not minimal, we first have to compute a minimal realization and then apply (17) to this realization. This can be achieved, e.g., by either the SR or BFSR method choosing ℓ such that $\sigma_\ell > \sigma_{\ell+1} = 0$ in (9). The reduced-order model is then obtained from applying (17) to the minimal realization. Note that if equal order of the reduced-order systems is assumed, the computable upper bound (14) on $\|\Delta_a\|_\infty$ also holds for singular perturbation approximation. The SPA model yields a good approximation at low frequencies, i.e., for $z = e^{\jmath\omega}$ with ω small, from (15) we expect a good match of $G(z)$ and $\tilde{G}(z)$.

We have implemented the SPA method starting from a minimal realization computed by either the SR or BFSR method using full-rank factors as described above. Suppose that the McMillan degree of the system is t and a minimal realization is given by $(\hat{A}_{11}, \hat{B}_1, \hat{C}_1, \hat{D})$. For this purpose, the minimal realization is partitioned according to the desired size ℓ of the reduced-order model, i.e.,

$$\begin{array}{rclrcl} \hat{A}_{11} & =: & \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, & \hat{B}_1 & =: & \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \\ \hat{C}_1 & =: & \begin{bmatrix} C_1 & C_2 \end{bmatrix}, & \hat{D} & =: & D. \end{array}$$

Then the SPA formulae (17) are applied to the system as partitioned above.

There are several options for the actual implementation of the SPA formulae. One way is to compute an LU decomposition with partial pivoting of $I - A_{22}$ and then solve the linear systems

$$(I - A_{22})Z_A = A_{21}, \quad (I - A_{22})Z_B = B_2$$

for Z_A and Z_B, respectively, by forward and backward substitution. Then the reduced-order system is obtained as

$$\begin{array}{rclrcl} A_r & := & A_{11} + A_{12}Z_A, & B_r & := & B_1 + A_{12}Z_B, \\ C_r & := & C_1 + C_2Z_A, & D_r & := & D + C_2Z_B. \end{array}$$

This implementation is matrix-multiplication rich and therefore good performance on parallel computers can be expected.

4 Parallel Implementation and Numerical Examples

4.1 Implementation details

The numerical algorithms that we have described in the previous two sections are all composed of basic matrix computations such as linear systems, matrix products, QR factorizations (with column pivoting), etc. All these operations can be efficiently implemented on modern serial and parallel computers. Although one could develop its own parallel routines for this purpose, nowadays there exist libraries of parallel kernels for distributed memory computers [4, 26]. The use of these libraries enhances the reliability and improves portability of the model reduction routines. The performance will depend on the efficiency of the underlying serial and parallel libraries and the communication routines.

Here we will employ the ScaLAPACK parallel library [4]. This is a public domain library that implements parallel versions of many of the kernels in LAPACK [1], using the message-passing paradigm. The ScaLAPACK is based on PB-BLAS (the parallel version of the serial BLAS) for computation and BLACS for communication. The BLACS can be ported to any (serial and) parallel architecture with an implementation of the MPI (our case) or the PVM communication libraries [12, 8].

In ScaLAPACK the computations are performed on a logical grid of $n_p = p_r \times p_c$ processes. The processes are mapped onto the physical processors, depending on the available number of these. All data (matrices) have to be distributed among the process grid prior to the invocation of a ScaLAPACK routine. It is the user's responsibility to perform this data distribution. Specifically, in ScaLAPACK the matrices are partitioned into $nb \times nb$ square blocks and these blocks are distributed (and stored) among the processes in column-major order. See [4] for details.

Employing ScaLAPACK, we have implemented the following model reduction algorithms for discrete-time LTI systems as Fortran 77 subroutines:

- `PDGEBTSR`: the BT SR method for model reduction;
- `PDGEBTBS`: the BT BFSR method for model reduction;
- `PDGESPSR`: the SPA formulae based on the SR method;
- `PDGESPBS`: the SPA formulae based on the BFSR method.

We compare these parallel routines with the analogous serial algorithms in SLICOT[1]. This library includes the following Fortran 77 routines:

- `AB09AD`: the BT SR or BFSR method for model reduction;
- `AB09BD`: the SPA formulae based on the SR or BFSR methods.

All the experimental results described in the following two subsections were obtained on Intel Pentium-II machines using IEEE double-precision floating-point arithmetic (i.e., the machine epsilon was $\varepsilon \approx 2.2204 \times 10^{-16}$).

4.2 Numerical accuracy

We evaluate the accuracy obtained by our model reduction algorithms using an LTI system that comes from the simulation of a catalytic tubular reactor used in a *gPROMS* training course [21]. A gas phase reaction (oxidation of *o*-xylene to phthalic anhydride) takes place inside the reactor which is packed with catalyst particles. The reactor is cooled externally. The mathematical model consists of a boundary control problem for a system of coupled partial differential equations including conservation laws for mass and energy. A continuous-time LTI system is obtained from a semi-discretization of the PDE system. The order of the system is $n = 1171$, and the numbers of inputs and outputs are $m = 6$ and $p = 4$, respectively. The continuous-time system was discretized using zero-order hold (i.e., $x_k = x(kT_s)$, $k = 0, 1, 2, \ldots$) with a sampling time $T_s = 0.1$ sec.

The ranks of the factors of the Gramians as computed using the algorithms described in Section 2 are 92 and 81 for W_c and W_o, respectively. Note that the SR and BFSR approach do not differ in this stage of the algorithm.

The accuracy of the reduced-order models can be judged on the basis of the magnitude for the frequency response of the ith output to the jth input, i.e., $|G_{ij}(z)|$ for $z = e^{\jmath\omega T_s}$ and

[1]Available via anonymous ftp at `ftp://www.esat.kuleuven.ac.be/pub/WGS/SLICOT`.

frequencies $\omega \in [\,0\,, \pi/T_s]$ (which is one part of the so-called discrete *Bode plots*; see, e.g., [18]). For BT methods, a good match of the reduced-order models with the original models at high frequencies is expected while for SPA models, good approximation at low frequencies should be observed. Note that this neither excludes good matching at low frequencies for BT models nor good approximation at high frequencies for SPA models.

In Figure 1 we show the magnitude for the frequency response for several of the 24 input-output (I/O) channels of the system and the reduced-order models of order $r = 40$. The plots show the expected behavior from which it can be concluded that decreasing the system order by a factor of roughly 30 does not change the frequency response significantly as visible deviations only occur when the magnitude is already very low.

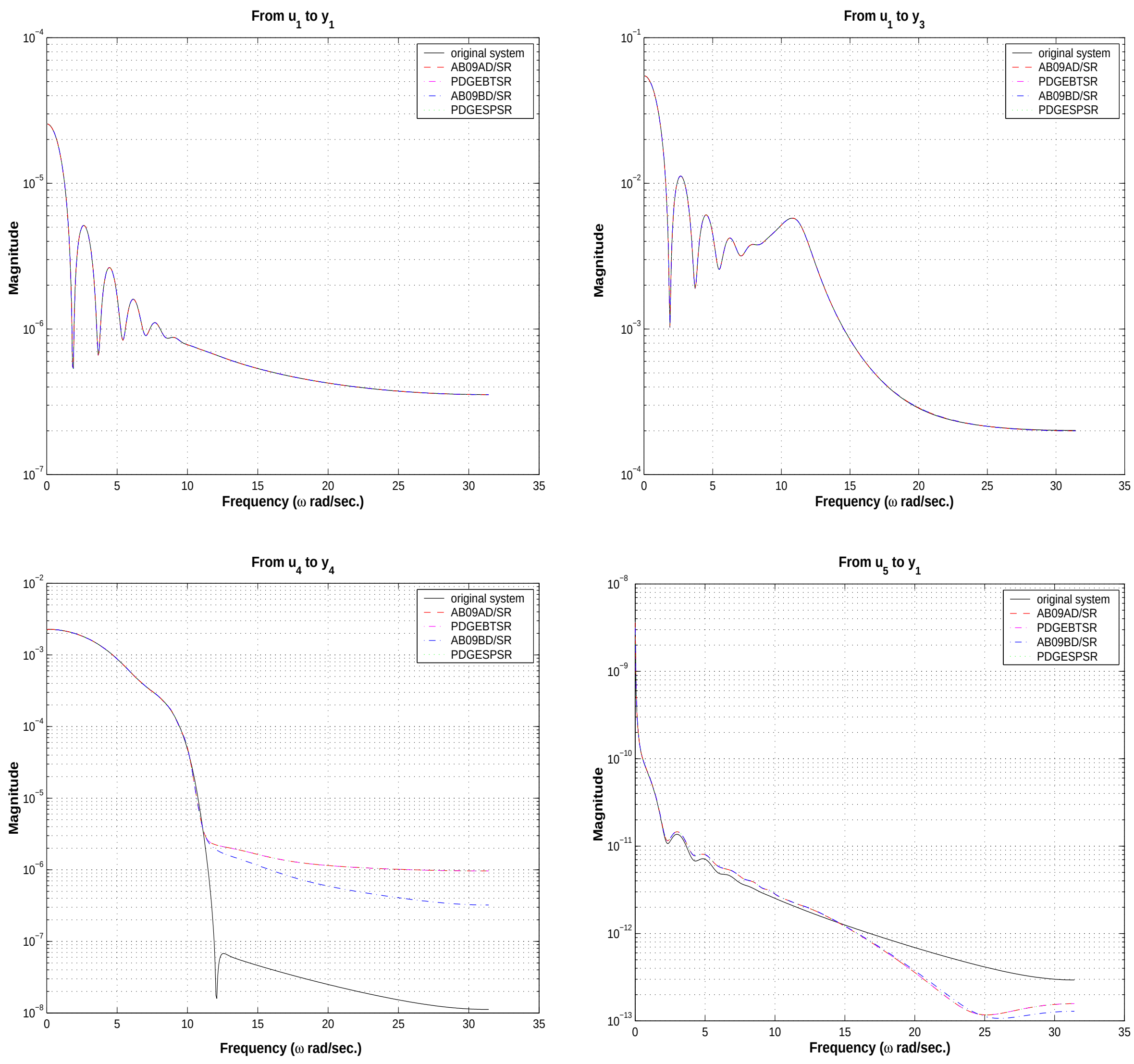

Figure 1: Frequency response (magnitude) of original and reduced-order models

Figure 2 reports the absolute errors of the transfer functions for the original system and the reduced-order models computed by the parallel algorithms. The absolute error is computed as the maximum singular value of the error system at the frequency ω, i.e.,

$$\|G(e^{\jmath\omega T_s}) - \tilde{G}(e^{\jmath\omega T_s})\|_2 = \sigma_{\max}\left(G(e^{\jmath\omega T_s}) - \tilde{G}(e^{\jmath\omega T_s}\right),$$

where $\| \,.\, \|_2$ denotes the matrix 2-norm, and $G(z)$, $\tilde{G}(z)$ are the TFMs of the original and the reduced-order model, respectively. From (3) we know that

$$\|G - \tilde{G}\|_\infty = \underset{\omega\in[-\pi,\pi)}{\text{ess sup}} \|G(e^{\jmath\omega T_s}) - \tilde{G}(e^{\jmath\omega T_s}\|_2$$

such that $\|G(e^{\jmath\omega T_s}) - \tilde{G}(e^{\jmath\omega T_s})\|_2 \leq \delta$ has to be satisfied where δ is the computable upper bound from (14). For the reduced-order model of order $r = 40$ computed here, we obtain $\delta = 1.5387 \times 10^{-5}$.

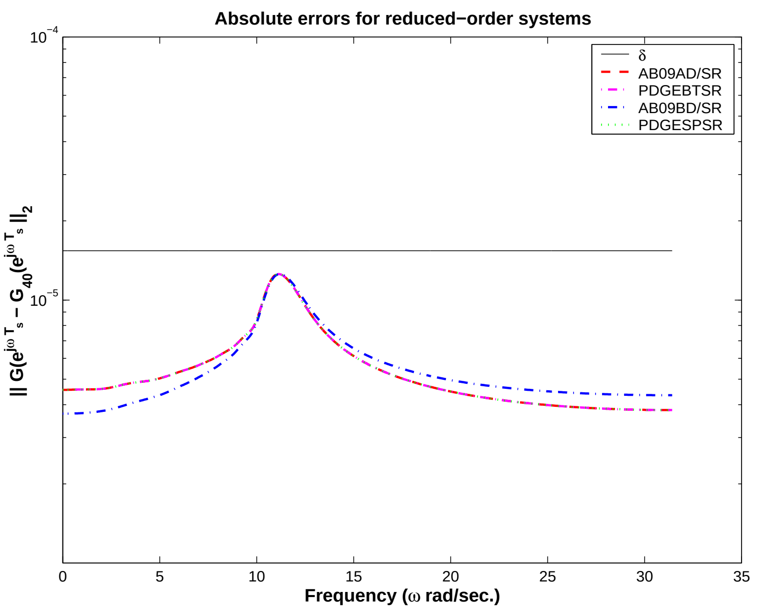

Figure 2: Absolute errors for BT and SPA reduced-order models

Figure 2 shows that there is no visible difference between the SPA models computed by the SLICOT routine AB09BD and our routine PDGESPSR. The BT models differ in that the model computed by PDGEBTSR almost matches the SPA reduced-order models while the SLICOT AB09AD model shows the expected behaviour of larger errors at $\omega = 0$ and smaller errors for increasing frequency—the differences to the other models are marginal, though. As to be expected, all reduced-order systems satisfy the error bound (14). The slightly unexpected behavior of the absolute errors can be explained as in this example, $\|A_{12}\|$ and $\|A_{21}\|$ are small such that SPA computes almost the same model as BT.

4.3 Parallel performance

In this subsection we analyze the performance of the parallel algorithms on a parallel distributed Beowulf cluster. 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 around 180 Mflops (millions of flops per second) for the matrix product (routine DGEMM). The nodes are connected by a *Myrinet* switch and the communication library BLACS employs a tuned implementation of MPI. The performance of the interconnection network for MPI was measured by a simple loopback message transfer an offered a latency of 33 μsec and a bandwidth around 200 Mbit/sec. We made extensive use of the LAPACK, PB-BLAS, and ScaLAPACK libraries.

In order to measure the performance, we generate random LTI systems for the parallel numerical experiments as follows. First, we generate a positive semidefinite diagonal Gramian matrix, $W_c = \mathrm{diag}\,(\Sigma_{q_1}, \Sigma_{q_2}, 0_{q_3}, 0_{q_4})$, where $\Sigma_{q_1} \in \mathbb{R}^{q_1 \times q_1}$ contains the desired Hankel singular values for the system and $\Sigma_{q_2} \in \mathbb{R}^{q_2 \times q_2}$ is a random diagonal matrix. We next construct a positive semidefinite diagonal Gramian matrix, $W_o = \mathrm{diag}\,(\Sigma_{q_1}, 0_{q_2}, \Sigma_{q_3}, 0_{q_4})$, with $\Sigma_{q_3} \in \mathbb{R}^{q_3 \times q_3}$. Then, A is set to a random (Schur) stable diagonal matrix and we compute $F = -(AW_cA^T - W_c)$ and $G = -(A^TW_oA - W_o)$. Thus,

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

Matrices $B \in \mathbb{R}^{n\times(q_1+q_2)}$ and $C \in \mathbb{R}^{(q_1+q_3)\times n}$ such that $F = BB^T$ and $G = C^C$ are then generated as

$$B = \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \\ 0 & 0 \end{bmatrix}, \qquad C = \begin{bmatrix} C_1 & 0 & 0 & 0 \\ 0 & 0 & C_3 & 0 \end{bmatrix},$$

where

$$\begin{aligned} B_1 &= \mathrm{diag}\left(\sqrt{f_1}, \ldots, \sqrt{f_{q_1}}\right), & B_2 &= \mathrm{diag}\left(\sqrt{f_{q_1+1}}, \ldots, \sqrt{f_{q_1+q_2}}\right), \\ C_1 &= \mathrm{diag}\left(\sqrt{g_1}, \ldots, \sqrt{g_{q_1}}\right), & C_3 &= \mathrm{diag}\left(\sqrt{g_{q_1+q_2+1}}, \ldots, \sqrt{g_{q_1+q_2+q_3}}\right). \end{aligned}$$

The LTI system is finally transformed as $A := U^TAU$, $B := U^TB$, 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 $\ell = q_1$ and the Cholesky factors satisfy $\mathrm{rank}(S) = q_1 + q_2$ and $\mathrm{rank}(R) = q_1 + q_3$.

Our first experiment evaluates the reduction in the execution time achieved by the parallel (BT and SPA) SR algorithms. For this purpose we compare the execution time of SLICOT routines AB09AD/SR and AB09BD/SR with those of our parallel routines PDGEBTSR and PDGESPSR. (The results for the BFSR algorithms showed no significant difference with respect to those reported here.) In the experiment, n is set to 1000, and we choose several different values for m, p, q_1, q_2 and q_3. Figure 3 shows the execution times of the serial algorithm, and those of the parallel algorithm for several number of nodes, n_p. The figure reports a remarkable reduction of the execution time achieved by the parallel algorithms on 2 nodes. This reduction is basically due to a significant decrease of the computational cost as only low rank approximations of the Cholesky factors need to be computed in our Stein solvers. This is particularly evident for small m, p, and q_1+q_2, q_1+q_3. For larger q_1, q_2, q_3, this advantage becomes less dramatic. But it should be noted that in most applications, $m, p \ll n$ holds. Comparison of the results on 2 and 4 nodes shows the parallel efficiency of our algorithms. E.g., in the $m = p = 200$, $q_1 = q_2 = q_3 = 100$ case, the execution time for routine PDGEBTSR is reduced from 61.8 to 34.4 seconds. This is a speed-up of about 1.8, close to the maximum that could be expected when the number of nodes is doubled. However, when further increasing the number of processors to 6 and 8 the speed-up factor decreases as in this case the ratio

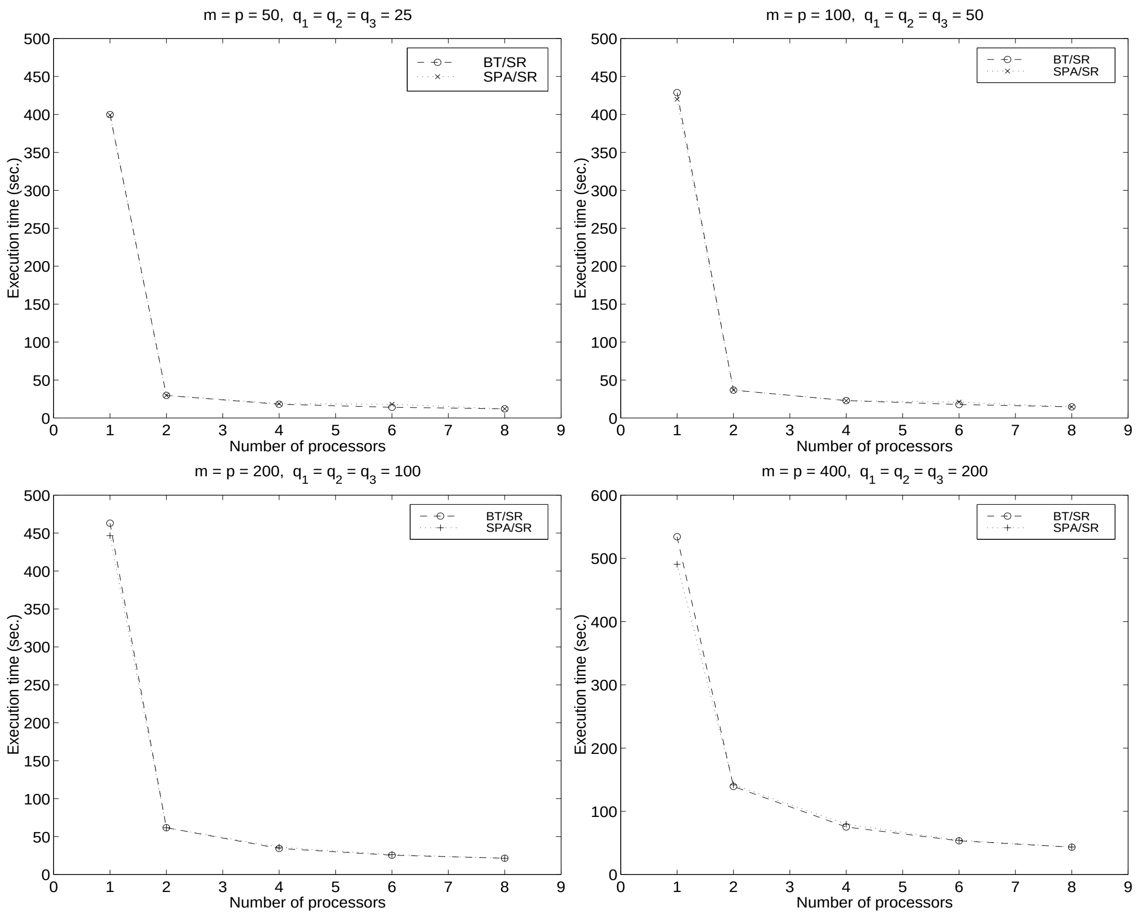

Figure 3: Execution time vs. number of nodes of the serial and parallel BT/SR and SPA/SR algorithms.

$n/\sqrt{n_p}$ is too small for obtaining good parallel performance. There is no big difference in the execution times of the BT and SPA algorithms as the main computational cost comes from solving the Stein equations for the Gramians and the subsequent SVD.

We next evaluate the scalability of the parallel algorithms. The scalability evaluates whether a larger problem can be solved by increasing proportionally the number of nodes of the parallel system. In the experiment we fix the problem size per node at $n/\sqrt{n_p} = 800$, $m/\sqrt{n_p} = 400$, $p/\sqrt{n_p} = 400$, and $q/\sqrt{n_p} = 100$ for $q = q_1$, q_2 and q_3. In Figure 4 we report the Mflops per node for the parallel algorithms `PDGEBTSR` and `PDGESPSR`. (Similar results were obtained for the corresponding BFSR routines.)

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 size $n = 4000$).

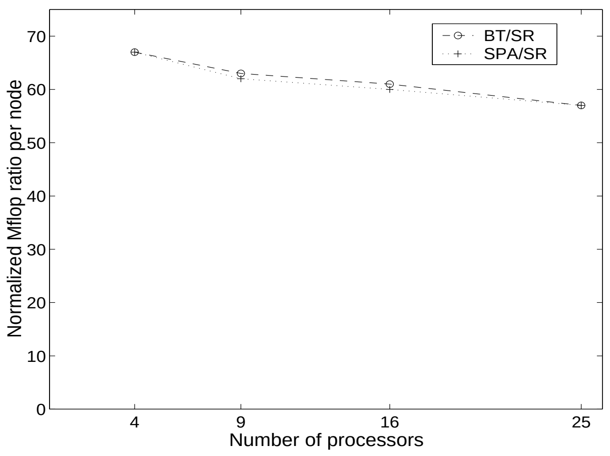

Figure 4: Scalability of the parallel algorithms with $n/\sqrt{n_p} = 800$, $m/\sqrt{n_p} = 400$, $p/\sqrt{n_p} = 400$, and $q/\sqrt{n_p} = 100$ for $q = q_1$, q_2 and q_3.

5 Concluding Remarks

We have described efficient and reliable numerical algorithms for the realization of model reduction methods based on the square-root version of balanced truncation and singular perturbation approximation. Using the full-rank factors of the Gramians often enhances the efficiency and accuracy of these methods significantly.

Implementations of the discussed methods are based on highly optimized software packages for numerical linear algebra on serial and parallel computers. Our experiments report similar numerical results for reliable serial model reduction algorithms from the SLICOT library and our model reduction approach, based on the Smith iteration. 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. Parallel computing thus allows to use these methods for systems of state-space dimension up to order $\mathcal{O}(10^5)$ or even higher, depending on the available memory. It should be noted that our method can be employed using very cheap infrastructure. That is, for any (local) network of PCs or workstations, the methods can be used as soon as a communication library as MPI is installed in the computer network.

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] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Numerical solution of Schur stable linear matrix equations on multicomputers. Berichte aus der Technomathematik, Report 99–13, FB3 – Mathematik und Informatik, Universität Bremen, 28334 Bremen (Germany), November 1999. Available from `http://www.math.uni-bremen.de/zetem/berichte.html`.

[3] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Balanced truncation model reduction of large-scale dense systems on parallel computers. *Math. Comp. Model. Dyn. Syst.*, 6, 2000, to appear.

[4] 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.

[5] H. Elmqvist, D. Brück, and M. Otter. *Dymola — User's Manual.* Dynasim AB, Research Park Ideon, Lund, Sweden, 1996.

[6] H. Elmqvist, S.E. Mattsson, and M. Otter. Modelica — a language for physical system modeling, visualization and interaction. In O. Gonzalez, editor, *Proc. IEEE Intl. Symp. Computer Aided Control System Design, Island of Hawaii (USA), August 22-27, 1999*, pages 630–639. Omnipress, Madison, WI, 1999.

[7] R. Freund. Reduced-order modeling techniques based on Krylov subspaces and their use in circuit simulation. In B.N. Datta, editor, *Applied and Computational Control, Signals, and Circuits*, volume 1, chapter 9, pages 435–498. Birkhäuser, Boston, MA, 1999.

[8] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, B. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine – A Users Guide and Tutorial for Network Parallel Computing.* MIT Press, Cambridge, MA, 1994.

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

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

[11] M. Green and D.J.N Limebeer. *Linear Robust Control.* Prentice-Hall, Englewood Cliffs, NJ, 1995.

[12] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface.* MIT Press, Cambridge, MA, 1994.

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

[14] S.J. Hammarling. Numerical solution of the discrete-time, convergent, non-negative definite Lyapunov equation. *Sys. Control Lett.*, 17:137–139, 1991.

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

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

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

[18] A.G.O. Mutambara. *Design and Analysis of Control Systems.* CRC Press, Boca Raton, FL, 1999.

[19] T. Penzl. Algorithms for model reduction of large dynamical systems. Technical Report SFB393/99-40, Sonderforschungsbereich 393 *Numerische Simulation auf massiv parallelen Rechnern*, TU Chemnitz, 09107 Chemnitz, FRG, 1999. Available from `http://www.tu-chemnitz.de/sfb393/sfb99pr.html`.

[20] T. Penzl. Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. *Sys. Control Lett.*, 40:139–144, 2000.

[21] Process Systems Enterprise Ltd. *gPROMS Training Course I. Hands-On Session: Modelling and Simulation of a Catalytic Tubular Reactor*, 1998, 1999.

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

[23] G. Schelfhout. *Model Reduction for Control Design.* PhD thesis, Dept. Electrical Engineering, KU Leuven, 3001 Leuven–Heverlee, Belgium, 1996.

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

[25] 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.

[26] R.A. van de Geijn. *Using PLAPACK: Parallel Linear Algebra Package.* MIT Press, Cambridge, MA, 1997.

[27] A. Varga. A note on Hammarling's algorithm for the discrete Lyapunov equation. *Sys. Control Lett.*, 15(3):273–275, 1990.

[28] 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.

[29] 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`.

[30] P.M.R. Wortelboer. *Frequency-weighted Balanced Reduction of Closed-loop Mechanical Servo-systems: Theory and Tools.* PhD thesis, Delft University of Technology, Delft, NL, 1994.

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

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.

99–08. Ronald Stöver:
Collocation methods for solving linear differential-algebraic boundary value problems, September 1999.

99–09. Huseyin Akcay:
Modelling with Orthonormal Basis Functions, September 1999.

99–10. Heike Faßbender, D. Steven Mackey, Niloufer Mackey:
Hamilton and Jacobi come full circle: Jacobi algorithms for structured Hamiltonian eigenproblems, Oktober 1999.

99–11. Peter Benner, Vincente Hernández, Antonio Pastor:
On the Kleinman Iteration for Nonstabilizable System, Oktober 1999.

99–12. Peter Benner, Heike Faßbender:
A Hybrid Method for the Numerical Solution of Discrete-Time Algebraic Riccati Equations, November 1999.

99–13. Peter Benner, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí:
Numerical Solution of Schur Stable Linear Matrix Equations on Multicomputers, November 1999.

99–14. Eberhard Bänsch, Karol Mikula:
Adaptivity in 3D Image Processing, Dezember 1999.

00–01. Peter Benner, Volker Mehrmann, Hongguo Xu:
Perturbation Analysis for the Eigenvalue Problem of a Formal Product of Matrices, Januar 2000.

00–02. Ziping Huang:
Finite Element Method for Mixed Problems with Penalty, Januar 2000.

00–03. Gianfrancesco Martinico:
Recursive mesh refinement in 3D, Februar 2000.

00–04. Eberhard Bänsch, Christoph Egbers, Oliver Meincke, Nicoleta Scurtu:
Taylor-Couette System with Asymmetric Boundary Conditions, Februar 2000.

00–05. Peter Benner:
Symplectic Balancing of Hamiltonian Matrices, Februar 2000.

00–06. Fabio Camilli, Lars Grüne, Fabian Wirth:
A regularization of Zubov's equation for robust domains of attraction, März 2000.

00–07. Michael Wolff, Eberhard Bänsch, Michael Böhm, Dominic Davis:
Modellierung der Abkühlung von Stahlbrammen, März 2000.

00–08. Stephan Dahlke, Peter Maaß, Gerd Teschke:
Interpolating Scaling Functions with Duals, April 2000.

00–09. Jochen Behrens, Fabian Wirth:
A globalization procedure for locally stabilizing controllers, Mai 2000.