State-Space Truncation Methods for Parallel Model Reduction of Large-Scale Systems

Peter Benner [a,1], Enrique S. Quintana-Ortí [b,2], Gregorio Quintana-Ortí [b,2]

[a] *Institut für Mathematik, MA 4-5, Technische Universität Berlin, 10623 Berlin, Germany; e-mail: benner@math.tu-berlin.de*

[b] *Depto. de Ingeniería y Ciencia de Computadores, Universidad Jaume I, 12.071-Castellón, Spain; e-mail: {quintana,gquintan}@icc.uji.es*

Abstract

We discuss a parallel library of efficient algorithms for model reduction of large-scale systems with state-space dimension up to $\mathcal{O}(10^4)$. We survey the numerical algorithms underlying the implementation of the chosen model reduction methods. The approach considered here is based on state-space truncation of the system matrices and includes absolute and relative error methods for both stable and unstable systems. In contrast to serial implementations of these methods, we employ Newton-type iterative algorithms for the solution of the major computational tasks. Experimental results report the numerical accuracy and the parallel performance of our approach on a cluster of Intel Pentium II processors.

Key words: Model reduction, state-space truncation, linear matrix equations, algebraic Riccati equations, sign function method, parallel linear algebra.

1 Introduction

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., [1,15]. LTI systems with state-space

[1] Supported by the DFG Research Center "Mathematics for key technologies" (FZT 86) in Berlin.

[2] Partially supported by the CICYT project No. TIC2002-004400-C03-01 and the *Generalitat Valenciana* project CTIDIA/2002/122.

dimension in the thousands are common in these applications. In particular, consider the continuous linear time-invariant (LTI) system in state-space form:

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

where $A \in \mathbb{R}^{n\times n}$ is the state matrix, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$, and $x^0 \in \mathbb{R}^n$ is the initial state of the system. Here, n is known as the order (or state-space dimension) of the system and the associated transfer function matrix (TFM) is $G(s) = C(sI - A)^{-1}B + D$. In the model reduction problem we are interested in finding a reduced-order LTI system,

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

of order r, $r \ll n$, and associated TFM $\hat{G}(s) = \hat{C}(sI - \hat{A})^{-1}\hat{B} + \hat{D}$ which approximates $G(s)$. Model reduction of discrete-time LTI systems can be formulated in an analogous manner. For brevity we will omit most of the details concerned with model reduction for discrete-time systems in this paper; see also [8].

A continuous-time LTI system is (c-)stable if all its poles are in the left half plane. Sufficient for this is that A is (c-)stable, i.e., the spectrum of A is contained in the open left half plane, denoted as $\Lambda(A) \subset \mathbb{C}^-$.

There is no general technique for model reduction that can be considered as optimal in an overall sense since the system characteristics play a vital role. In this paper we focus on the so-called state-space truncation approach [1,33]. That is, the reduced-order model is obtained from truncating a state-space transformation $(A, B, C, D) \to (TAT^{-1}, TB, CT^{-1}, D)$, where $T \in \mathbb{R}^{n\times n}$ is nonsingular, $T =: \left[T_l^T, L_l^T\right]^T$, and $T^{-1} =: [T_r, L_r]$, so that with $T_l \in \mathbb{R}^{r\times n}$, $T_r \in \mathbb{R}^{n\times r}$, the reduced-order model is

$$\hat{A} = T_l A T_r, \quad \hat{B} = T_l B, \quad \hat{C} = CT_r, \quad \hat{D} = D. \tag{3}$$

This corresponds to projecting the dynamics of the system onto an r-dimensional linear manifold via $\hat{x} = T_r T_l x$.

State-space truncation methods for model reduction differ in the measurement of the approximation error and the way they attempt to minimize this error. Balanced truncation (BT) methods [32,39,41,44], singular perturbation approximation (SPA) methods [31], and optimal Hankel-norm approximation (HNA) methods [19] 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 $\|\Delta_a\|_\infty$ where $\|\ .\ \|_\infty$ denotes the $\mathcal{L}_\infty$- or $\mathcal{H}_\infty$-

norm of a stable, rational matrix function defined by

$$\|G\|_\infty = \operatorname*{ess\,sup}_{\omega\in\mathbb{R}} \sigma_{\max}(G(\imath\omega)). \tag{4}$$

Here $\imath = \sqrt{-1}$ and $\sigma_{\max}(M)$ is the largest singular value of the matrix M. In contrast to BT and SPA, which compute sub-optimal approximations, HNA methods find the optimal solution of the absolute error minimization problem if the *Hankel norm* $\|\,.\,\|_H$ is used, where

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

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

Relative error methods attempt to minimize the relative error $\|\Delta_r\|_\infty$ defined implicitly by $G - \hat{G} = G\Delta_r$. Among these, the balanced stochastic truncation (BST) method [17,22,48] is particularly popular. In contrast to BT and SPA, BST reduced-order models approximate the original TFM uniformly over the whole frequency range and also provide a good approximation of the phase properties [38]. The latter property is particularly important in the context of inverse problems [13].

All state-space truncation methods mentioned so far can only be applied if the system is stable. However, if stabilization of the system is the computational task to solve, the system is obviously unstable. 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 can be employed. Unstable systems often occur in controller reduction: controllers are often themselves unstable systems and therefore the task of controller reduction leads to model reduction of unstable systems [46]. Model reduction of unstable systems is usually dealt with by first separating the stable and the unstable parts of the system, and then reducing the stable part using any of the state-space truncation methods.

In general, model reduction methods for LTI systems with dense state matrices have a computational cost of $\mathcal{O}(n^3)$ floating-point arithmetic operations (flops) and require storage for $\mathcal{O}(n^2)$ numbers. While current desktop computers provide enough computational power to reduce models of order n in the hundreds using libraries like SLICOT [3] or the MATLAB control-related toolboxes, large-scale applications clearly require the use of advanced computing techniques. One approach would be to exploit any special structure of the given system, e.g., sparsity of the state matrix A. Several approaches for this have been discussed recently, see, e.g., [1,18,23,30]. These methods are specialized for certain problem classes and often lack properties like error bounds or preservation of stability, passivity, or phase information. Though a lot of research is ongoing, these methods cannot be used as a black-box. Therefore, we

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

will focus here on the parallelization of state-space truncation methods which will allow to reduce large problems without going through the tedious process of developing a specialized code for the given problem. Note that we not only parallelize the underlying computational steps but often replace them by new methods that are better suited for parallel computations!

The rest of the paper is structured as follows. In Section 2 we review a procedure for model reduction of unstable systems. Absolute and relative error methods for model reduction of stable systems are described, respectively, in Sections 3 and 4. Efficient algorithms for the solution of the major computational problems arising in state-space truncation methods are discussed in Section 5. The integration of these algorithms in a parallel library for model reduction, PLiCMR, is outlined in Section 6. Finally, the performance on a cluster of Intel Pentium II processors is reported in Section 7, and some concluding remarks follow in Section 8.

2 Model Reduction of Unstable Systems

Usually, unstable poles cannot be neglected when modeling the dynamics of a system, and therefore should be preserved in the reduced-order system in some sense. This is trivially satisfied using the following approach [40,47]: first, compute an *additive decomposition* of the TFM,

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

such that G_- is stable and G_+ is unstable. Then any of the absolute or relative error state-space truncation methods for model reduction can be applied to G_- in order to obtain a reduced-order transfer function $\hat{G}_-$, and the reduced-order system is synthesized by $\hat{G}(s) = \hat{G}_-(s) + G_+(s)$. Hence, the unstable part is preserved in the reduced-order system. This is an important property 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 many applications the number of unstable poles is very low compared to the number of stable poles.

We now describe how the decomposition (5) can be computed using the matrix sign function. A definition of this matrix function and an iterative algorithm for its computation are given in subsection 5.1.

Consider the realization (A, B, C, D) of a continuous-time LTI system, and let $\operatorname{sign}(A)$ denote the sign function of A. We start by computing a (rank-revealing) QR factorization

$$I_n - \operatorname{sign}(A) = QRP, \quad R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}$$

where $Q \in \mathbb{R}^{n\times n}$ is orthogonal, $R \in \mathbb{R}^{n\times n}$ is upper triangular, with $R_{11} \in \mathbb{R}^{k\times k}$, and $P \in \mathbb{R}^{n\times n}$ is a permutation matrix. Note that the zeros in the last $n-k$ rows of R are to be understood as "zero with respect to a given tolerance threshold". Then the first k columns of Q span the stable A-invariant subspace. Thus,

$$\tilde{A} \;:=\; Q^T A Q \;=\; \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \tag{6}$$

where $\Lambda(A_{11}) = \Lambda(A) \cap \mathbb{C}^-$, and $\Lambda(A_{22}) = \Lambda(A) \cap \mathbb{C}^+$.
In a second step, we compute a matrix $V \in \mathbb{R}^{n\times n}$ such that

$$\hat{A} \;:=\; V^{-1}\tilde{A}V \;=\; \begin{bmatrix} I_k & -Y \\ 0 & I_{n-k} \end{bmatrix} \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} \begin{bmatrix} I_k & Y \\ 0 & I_{n-k} \end{bmatrix}, \tag{7}$$

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

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

As $\Lambda(A_{11}) \cap \Lambda(A_{22}) = \emptyset$, equation (8) has a unique solution [27]. Sylvester equations with strictly stable/unstable coefficient matrices can be solved using the iterative algorithm described in subsection 5.2.
The desired additive decomposition of $G(s) = C(sI-A)^{-1}B + D$ is finally obtained by performing the state-space transformation

$$\begin{aligned}(\hat{A}, \hat{B}, \hat{C}, \hat{D}) &:= (V^{-1}Q^TAQV, V^{-1}Q^TB, CQV, D) \\ &= \left(\begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}, \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, [\, C_1 \;\; C_2], D \right),\end{aligned}$$

where $\hat{A}$, $\hat{B}$, and $\hat{C}$ are partitioned conformally with the partitioning in (6)–(7), so that

$$\begin{aligned} G(s) &= C(sI-A)^{-1}B + D = \hat{C}(sI-\hat{A})^{-1}\hat{B} + \hat{D} \\ &= \left\{\hat{C}_1(sI_k - A_{11})^{-1}B_1 + D\right\} + \left\{C_2(sI_{n-k} - A_{22})^{-1}B_2\right\} \\ &=: G_-(s) + G_+(s), \end{aligned}$$

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

3 Absolute Error Methods for Stable Systems

Absolute error methods are strongly related to the controllability Gramian W_c and the observability Gramian W_o of the system. In the continuous-time case,

the Gramians are given by the solutions of two coupled *Lyapunov equations*

$$AW_c + W_cA^T + BB^T = 0, \quad A^TW_o + W_oA + C^TC = 0. \tag{9}$$

(In the discrete-time case, the Gramians are the solutions of two coupled analogous *Stein equations*.) 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 (even if they are not Cholesky factors in a strict sense).

Note that the efficient algorithms for the solution of the coupled Lyapunov in (9) briefly reviewed in subsection 5.3 do not compute square Cholesky factors, but full-rank factors of W_c, W_o.

Consider now the singular value decomposition (SVD)

$$SR^T = U\Sigma V^T \, [U_1 \; U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}, \tag{10}$$

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

It should be emphasized that, though mathematically equivalent, our methods for solving (9) and (10) significantly differ from standard methods used in the MATLAB toolboxes or SLICOT [47]. As we are using full-rank factors rather than Cholesky factors, the solution of (9) is usually much more efficient if the Gramians have low (numerical) rank—which is typically the case in many large-scale models. The effect is even more drastic when looking at (10): instead of a cost of $\mathcal{O}(n^3)$ when using Cholesky factors, this new approach leads to an $\mathcal{O}(n_c \cdot n_o \cdot n)$ cost where n_c, n_o, the column dimensions of the full rank factors, often satisfy $n_c, n_o \ll n$; see [7].

3.1 Balanced truncation

The so-called *square-root* (SR) BT algorithms [29,41] determine the reduced-order model in (3) using the projection matrices

$$T_l = \Sigma_1^{-1/2} V_1^T R \quad \text{and} \quad T_r = S^T U_1 \Sigma_1^{-1/2}. \tag{11}$$

In case $\Sigma_1 > 0$ and $\Sigma_2 = 0$, this reduced-order model is a minimal balanced realization of the TFM $G(s)$.

If the original system is highly unbalanced (and hence, the state-space transformation matrix T is ill-conditioned), the *balancing-free square-root* (BFSR)

BT algorithms often provide more accurate reduced-order models in the presence of rounding errors [44]. These algorithms differ in the procedure to obtain T_l and T_r from the SVD factorization of SR^T and in that the reduced-order model is not balanced.
The absolute error of a realization of order r computed by the SR or BFSR BT algorithms satisfies the upper bound [19]

$$\|\Delta_a\|_\infty = \|G - \hat{G}\|_\infty \leq 2 \sum_{j=r+1}^{n} \sigma_j. \tag{12}$$

This allows an adaptive choice of the size of the reduced-order model if a given upper bound for the error is to be satisfied.

3.2 Singular perturbation approximation

Let the tuple $(\tilde{A}, \tilde{B}, \tilde{C}, D)$ denote a minimal realization of the system computed using either the SR or BFSR BT algorithms, and partition

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

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

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

where $\gamma = 0$ for continuous-time systems ($\gamma = 1$ for discrete-time systems) [31,43,44].
The realizations computed using the SR or BFSR SPA algorithms also satisfy the absolute error bound in (12).

3.3 Hankel-norm approximation

Using the *Hankel norm* of a stable rational TFM, $\|G\|_H$, it is possible to compute an approximation minimizing $\|\Delta_a\|_H$ for a given order r of the reduced-order system [19]. Here we only describe the basic computational steps of the HNA method in order to show which computational kernels (matrix products, QR factorizations, etc.) are needed to implement the HNA method. Further details are given in [1,19,33,49].
In the first step, a balanced minimal realization of G is computed using, e.g., the SR BT algorithm described in subsection 3.1.

Next, a TFM $\tilde{G}(s) = \tilde{C}(sI - \tilde{A})^{-1}\tilde{B} + \tilde{D}$ is computed as follows: first, the order r of the reduced-order model is chosen such that the Hankel singular values of G satisfy

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

By applying the appropriate permutations, the balanced transformation of G is re-ordered such that the Gramians become diag $\left(\check{\Sigma}, \sigma_{r+1} I_k\right)$. The resulting balanced realization given by $(\check{A}, \check{B}, \check{C}, \check{D})$ is partitioned conformally with the partitioning of the Gramians, i.e.,

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

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

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

Here, $U := (C_2^T)^\dagger B_2$, where $M^\dagger$ denotes the pseudoinverse of M [21], and $\Gamma := \check{\Sigma}^2 - \sigma_{r+1}^2 I_{n-k}$. Now, following the procedure described in subsection 2, we can compute an additive decomposition of $\tilde{G}$ such that $\tilde{G}(s) = \hat{G}(s) + \tilde{G}_+(s)$ where $\hat{G}$ is stable and $\tilde{G}_+$ is antistable. Then $\hat{G}$ is an optimal r-th order Hankel-norm approximation of G.

The absolute error for a realization of order r computed using the HNA method satisfies [19]

$$\|\Delta_a\|_H = \|G - \hat{G}\|_H = \sigma_{r+1}. \tag{14}$$

This allows again an adaptive choice of r. Note that the TFM $\hat{G}$ computed using the HNA method also satisfies the $\mathcal{H}_\infty$-norm bound (12).

4 Relative Error Methods for Stable Systems

We assume here that $0 < p \leq m$, rank $(D) = p$, which implies that $G(s)$ must not be strictly proper. For strictly proper systems, the method can be applied introducing an ϵ-regularization by adding an artificial matrix $D = [\epsilon I_p \; 0]$ [20].

BST is a model reduction method based on truncating a balanced stochastic realization. Such a realization is obtained as follows; see [22] for details. Define $\Phi(s) = G(s)G^T(-s)$, and let W be a *square minimum phase right spectral factor* of Φ, satisfying $\Phi(s) = W^T(-s)W(s)$. As D has full row rank, $E := DD^T$

is positive definite, and a minimal state-space realization (A_W, B_W, C_W, D_W) of W is given by

$$A_W := A, B_W := BD^T + W_c C^T, C_W := E^{-\frac{1}{2}}(C - B_W^T X_W), D_W := E^{\frac{1}{2}},$$

where $W_c = S^T S$ is the controllability Gramian defined in (9), while X_W is the observability Gramian of $W(s)$ obtained as the stabilizing solution of the algebraic Riccati equation (ARE)

$$F^T X + XF + XB_W E^{-1} B_W^T X + C^T E^{-1} C = 0, \tag{15}$$

with $F := A - B_W E^{-1} C$. Here, X_W is symmetric positive (semi-)definite and thus admits a decomposition $X_W = R^T R$. In SR BST a transformation T yielding projection matrices T_l, T_r as in the BT method is obtained from the dominant left and right singular subspaces of SR^T such that the transformed system $(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}) = (T^{-1}AT, T^{-1}B, CT, D)$ is stochastically balanced. That is, the controllability Gramian $\tilde{W}_c$ satisfies

$$\tilde{W}_c := T^{-1} W_c T^{-T} = \operatorname{diag}(\sigma_1, \ldots, \sigma_n) = T^T X_W T =: \tilde{X}_W, \tag{16}$$

where $1 = \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$. A BST reduced-order model is then obtained by truncating the realization $(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D})$ to order r where $\sigma_r \gg \sigma_{r+1}$; This BSR satisfies the following relative error bound

$$\sigma_{r+1} \leq \|\Delta_r\|_\infty \leq \prod_{j=r+1}^{n} \frac{1+\sigma_j}{1-\sigma_j} - 1, \tag{17}$$

where $G\Delta_r = G - \hat{G}$. From that we obtain

$$\frac{\|G - \hat{G}\|_\infty}{\|G\|_\infty} \leq \prod_{j=r+1}^{n} \frac{1+\sigma_j}{1-\sigma_j} - 1. \tag{18}$$

Our algorithms differ in several ways from the ones considered in [38,48], though they are mathematically equivalent. Specifically, the Lyapunov equation for W_c is solved using a sign function iteration described in subsection 5.3, from which we obtain a full-rank factorization $W_c = S^T S$. The same approach is used to compute a full-rank factor R of X_W from a stabilizing approximation $\tilde{X}_W$ to X_W using the technique described in [46]: let $D = \begin{bmatrix} \hat{D}^T & 0 \end{bmatrix} U$ be an LQ decomposition of D. Note that $\hat{D} \in \mathbb{R}^{p \times p}$ is a square, nonsingular matrix as D has full row rank. Now set

$$H_W := \hat{D}^{-T} C, \quad \hat{B}_W := B_W \hat{D}^{-1}, \quad \hat{C} := (H_W - \hat{B}_W^T X).$$

Then the ARE (15) is equivalent to $A^T X + XA + \hat{C}^T \hat{C} = 0$. Using a computed approximation $\tilde{X}_W$ of X_W to form $\hat{C}$, the Cholesky or full-rank factor R of

X_W can be computed directly from the Lyapunov equation

$$A(R^T R) + (R^T R)A + \hat{C}^T \hat{C} = 0.$$

The approximation $\tilde{X}_W$ is obtained by solving (15) using Newton's method with exact line search as described in subsection 5.4 (see also [3]). The Lyapunov equation for R is solved using the sign function iteration from subsection 5.3.

5 Solving Linear and Quadratic Matrix Equations

The first step in all model reduction techniques discussed so far involves the numerical solution of linear and quadratic matrix equations, namely Sylvester, Lyapunov and Stein equations as well as AREs. In this section we will review how these equations can be solved by iterative methods that are particularly attractive for parallelization.

5.1 The matrix sign function

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. 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 stable and unstable parts of $\Lambda(Z)$. The *matrix sign function* of Z is defined as $\operatorname{sign}(Z) := S \begin{bmatrix} -I_k & 0 \\ 0 & I_{n-k} \end{bmatrix} S^{-1}$. 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 [25] for an overview.
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:

$$Z_0 \leftarrow Z, \quad Z_{j+1} \leftarrow \frac{1}{2}(Z_j + Z_j^{-1}), \quad j = 0, 1, 2, \ldots. \tag{19}$$

Under the given assumptions, the sequence $\{Z_j\}_{j=0}^{\infty}$ converges to $\operatorname{sign}(Z) = \lim_{j\to\infty} Z_j$ [37] with an ultimately quadratic convergence rate. As the initial convergence may be slow, the use of acceleration techniques is recommended; e.g., *determinantal scaling* [14] is given by

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

Note that the determinant $\det(Z_j)$ is a by-product of the computations required to implement (19).
Efficient parallelization of the matrix sign function has been reported, e.g., in [2,24].

5.2 Solution of Sylvester equations

Consider a Sylvester equation of the form

$$AX + XB + C = 0, \tag{20}$$

with $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{n\times n}$ stable matrices, and $C \in \mathbb{R}^{m\times n}$. This equation can be solved using a sign function-based iterative procedure, derived by Roberts [37], which can be formulated as follows

$$\begin{array}{rclcrcl} A_0 & \leftarrow & A, & & A_{j+1} & \leftarrow & \frac{1}{2}\left(A_j + A_j^{-1}\right), \\ B_0 & \leftarrow & B, & & B_{j+1} & \leftarrow & \frac{1}{2}\left(B_j + B_j^{-1}\right), \\ C_0 & \leftarrow & C, & & C_{j+1} & \leftarrow & \frac{1}{2}\left(C_j + A_j^{-1} C_j B_j^{-1}\right), \end{array} \qquad j = 0, 1, 2, \ldots. \tag{21}$$

It follows that $\lim_{j\to\infty} A_j = -I_m$, $\lim_{j\to\infty} B_j = -I_n$, and $X = \frac{1}{2}\lim_{j\to\infty} C_j$. For an efficient implementation of this iteration on modern computer architectures and numerical experiments reporting efficiency and accuracy, see [6].

5.3 Solution of Lyapunov and Stein equations

Exploiting that the Lyapunov equation $A^T X + XA + Q = 0$, with $A \in \mathbb{R}^{n\times n}$ stable and $Q \in \mathbb{R}^{n\times n}$ symmetric positive semidefinite, is a special case of the Sylvester equation (20), the iteration (21) boils down to

$$\begin{array}{rclcrcl} A_0 & \leftarrow & A, & & A_{j+1} & \leftarrow & \dfrac{1}{2}\left(A_j + A_j^{-1}\right), \\ Q_0 & \leftarrow & Q, & & Q_{j+1} & \leftarrow & \dfrac{1}{2}\left(Q_j + A_j^{-T} Q_j A_j^{-1}\right), \end{array} \qquad j = 0, 1, 2, \ldots, \tag{22}$$

so that $\lim_{j\to\infty} A_j = -I_n$ and $X = \frac{1}{2}\lim_{j\to\infty} Q_j$.
In [5,28] this iteration was modified to obtain the Cholesky factor rather than the solution itself of a Lyapunov equation of the form

$$A^T X + XA + L^T L = 0,$$

where $A \in \mathbb{R}^{n\times n}$ is stable and $L \in \mathbb{R}^{m\times n}$. The modified iteration can be formulated as follows:

$$\begin{array}{rclcrcl} A_0 & \leftarrow & A, & & A_{j+1} & \leftarrow & \dfrac{1}{2}\left(A_j + A_j^{-1}\right), \\ L_0 & \leftarrow & L, & & L_{j+1} & \leftarrow & \dfrac{1}{\sqrt{2}}\begin{bmatrix} L_j \\ L_j A_j^{-1} \end{bmatrix}, \end{array} \qquad j = 0, 1, 2, \ldots. \tag{23}$$

As in the applications considered here the Cholesky factors are often of low (numerical) rank, we can save some workspace and arithmetic work by a col-

umn compression of the iterates L_j. That is, in each step we compute a rank-revealing QR decomposition of the matrix L_{j+1} using the QR decomposition with column pivoting [21]. We then obtain $\lim_{j\to\infty} L_j = \hat{L}$, with $\hat{L}\hat{L}^T = X$; i.e., the iterates converge to the full-rank factors of the solution. Note that in all the absolute error methods discussed here, we need to solve both Lyapunov equations (9). We can couple the two iterations so that only one of the two sequences $\{A_j\}_{j=0}^{\infty}$ needs to be computed and the cost is further reduced; see, e.g., [7] for details.

In particular, in model reduction $m, p \ll n$ and the *numerical rank* of the Cholesky factors S, R of the system Gramians is also usually much smaller than n. Therefore, working with the full-rank factors quite often saves a large amount of workspace and computational cost. Details of the method and the implementation of BT model reduction using these factors instead of the Cholesky factors can be found in [7].

The same techniques described here can also be employed for the solution of the Stein equation arising in discrete-time systems if the sign function iteration is replaced by the squared Smith iteration; see [11] for details.

5.4 The Newton method for the ARE

In [26] Kleinman shows that, under suitable conditions, Newton's method applied to the classical ARE, as it appears in optimal control, converges to the desired stabilizing solution of the ARE.

All the convergence results for Newton's method applied to the classical ARE can be derived in a similar way for the case considered here; see [3,45]. In particular, we use these results to formulate Newton's method for an ARE of the form

$$\mathcal{R}(X) := F^T X + XF + XPX + Q = 0, \tag{24}$$

with $F \in \mathbb{R}^{n\times n}$ stable and $P, Q \in \mathbb{R}^{n\times n}$ symmetric positive semidefinite. This can then be applied to (15) with

$$F := A - B_W E^{-1} C, \quad P := B_W E^{-1} B_W^T, \quad Q := C^T E^{-1} C. \tag{25}$$

The Newton iteration for (24), starting from some initial stabilizing symmetric guess X_0, can be formulated as follows:

$$\begin{aligned} F_j &\leftarrow F + PX_j, \\ X_{j+1} &\leftarrow X_j + N_j, \end{aligned} \qquad j = 0, 1, 2, \ldots,$$

where N_j satisfies the Lyapunov equation $F_j^T N_j + N_j F_j + \mathcal{R}(X_j) = 0$.

In our implementation, we employ the sign function-based method (22) to solve the Lyapunov equations in each step of the Newton iteration [9].

Although Newton's method for the ARE (24) converges ultimately quadratically from any starting guess X_0 such that $F + PX_0$ is stable, the initial convergence may be slow. Even worse, sometimes the X_1 is an enormous leap away from X_0 and the exact solution, and the sequence $\{X_j\}_{j=0}^{\infty}$ only converges slowly afterwards. Therefore, in practice, we use a variant of Newton's method which includes an exact line search technique in order to accelerate convergence in the early stages of the iteration. This technique was proposed for (24) in [3].

Here we apply the suggested modified Newton's method to (15) using (25). As $A - B_W E^{-1} C$ is stable [22,48], we can start Newton's iteration with $X_0 = 0$ such that the problem of finding a stabilizing starting guess is circumvented in this case.

For further details on the exact line search and the parallelization of Newton's method, see [4].

6 A Parallel Library for Model Reduction

The numerical algorithms that we have described in the previous sections are all composed of basic matrix computations such as solving linear systems, matrix products, and QR factorizations (with and without column pivoting). Efficient implementations of these operations are available in parallel linear algebra libraries for distributed memory computers like PLAPACK and ScaLAPACK [12,42]. The use of these libraries enhances the reliability and improves portability of the model reduction routines. The performance will depend on the efficiencies of the underlying serial and parallel computational linear algebra libraries and the communication routines.

Using the kernels in ScaLAPACK, we have implemented a library for model reduction of LTI systems, PLiCMR [4], in Fortran 77. The library contains a few driver routines for model reduction and several computational routines for the solution of related equations in control. The functionality and naming convention of the parallel routines closely follow analogous routines from SLICOT. As part of PLiCMR, three parallel driver routines are provided for absolute error model reduction, one parallel driver routine for relative error model reduction, and an expert driver routine capable of performing any of the previous functions on stable and unstable systems:

- `pab09ax`: SR and BFSR BT algorithms.
- `pab09bx`: SR and BFSR SPA algorithms.
- `pab09cx`: HNA algorithm.
- `pab09hx`: SR and BFSR BST algorithms.
- `pab09mr`: Model reduction of stable/unstable systems employing any of the four previous methods.

Table 1 shows a list of the computational routines included in PLiCMR.

[4] Available from `http://spine.act.uji.es/~plicmr.html`.

Purpose	Routine	
Spectral division	`pmb05rd`	
	Continuous-time	Discrete-time
ARE solver	`pdgecrnz`	–
Sylvester solver	`psb04md`	–
Lyapunov solver	`pdgeclne`	–
Coupled Lyapunov/Stein solver	`psb03odc`	`psb03odd`

Table 1
Computational routines in PLiCMR.

6.1 Implementation details

The efficiency of our model reduction routines strongly depends on the efficiency of two numerical kernels: the QR factorization with column pivoting, employed in iteration (23) for the coupled Lyapunov equations in (9), and the matrix inversion routines necessary, e.g., in the Newton iteration for the matrix sign function and related iterations for Sylvester and Lyapunov equations. Highly efficient parallel routines are adopted for both computations in our library. First, we employ a BLAS-3 version of the QR factorization with column pivoting [36] which outperforms the traditional BLAS-2 implementation both in serial and parallel architectures. This new version has been included in LAPACK (version 3.0) as routine `DGEQP3`. We have developed a ScaLAPACK-like parallel implementation of the routine. Secondly, we propose to use an inversion procedure based on Gauss-Jordan elimination. This approach presents a better balance of the computational load for parallel distributed-memory architectures, see [35].

The numerical rank of a matrix is estimated in our routines by using the QR factorization with column pivoting and an incremental estimator. Setting a tolerance threshold for the numerical rank is a delicate problem, specially if the matrix has no large gap in its singular value distribution. As a general solution, in order to determine the numerical rank of a square matrix of order n, we set the rank tolerance threshold, τ_{rank}, to $\tau_{\text{rank}} = 10 \cdot \sqrt{n} \cdot \varepsilon$, where ε is the machine precision. We found this threshold to serve our purposes in practice.

Most of the computational routines in the library are based on iterative methods with quadratic convergence, e.g., the Newton iteration is used to solve AREs, the iteration (19) for computing the matrix sign function is also an implementation of Newton's method, etc. In all these cases we use an iteration tolerance threshold, τ_{iter}, defined as $\tau_{\text{iter}} = 10 \cdot n \cdot \sqrt{\varepsilon}$. Table 2 lists the specific convergence criteria employed by the computational routines. As all the iterative algorithms in the library present an ultimately quadratic convergence, once the corresponding threshold is satisfied two more iterations are carried

out to guarantee the maximum attainable accuracy.

pmb05rd	$\|Z_{j+1} - Z_j\| < \tau_{\text{iter}} \cdot \|Z_j\|$		
Continuous-time		Discrete-time	
pdgecrnz	$\|\mathcal{R}(X_j)\|_F < \tau_{\text{iter}} \cdot \|X_j\|_F$	–	–
psb04md –	$\|A_j + I_n\|_F < \tau_{\text{iter}} \cdot \|A_j\|_F$ and $\|B_j - I_n\|_F < \tau_{\text{iter}} \cdot \|B_j\|_F$	–	–
pdgeclne	$\|A_j + I_n\|_F < \tau_{\text{iter}} \cdot \|A_j\|_F$	–	–
psb03odc	$\|A_j + I_n\|_F < \tau_{\text{iter}} \cdot \|A_j\|_F$	psb03odd	$\|A_j\|_F < \tau_{\text{iter}} \cdot \|A\|_F$

Table 2
Convergence criteria for the iterations in the computational routines. The Frobenious norm was employed in all cases.

In ScaLAPACK [12] the computations are performed by a logical grid of $n_p = p_r \times p_c$ processes which are mapped onto the physical processors, depending on the available number of these. All data matrices are partitioned into $mb \times nb$ blocks, and these blocks are then distributed among the processes in column-major order. Our current implementations of the routines for model reduction in PLiCMR require all data matrices passed as arguments to the driver and computational routines of the libraries to start at entry (1,1) which has to be stored by process (0,0).

7 Experimental Results

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. 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 via a *Myrinet* multistage network; the communication library BLACS is based on an implementation of the MPI communication library 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.
We compare the accuracy and performance of the parallel routines in PLiCMR and the corresponding serial routines in SLICOT:

- ab09ad: SR and BFSR BT algorithms.
- ab09bd: SR and BFSR SPA algorithms.
- ab09cd: HNA algorithm.
- ab09hd: SR and BFSR BST algorithms.

As we did not find any significant difference between the SR and BFSR algorithms, in the experiments we only report results for the latter.

7.1 Accuracy of the reduced-order models

We evaluate the numerical performance of our model reduction using eight moderate-scale examples coming from very different application areas ranging from meteorology over servomechanism design to structural mechanics. For a detailed description of the applications, see [15] and the references therein.

Table 3 shows the parameters of the systems used in the evaluation. In order to fix the order of the reduced-order system automatically, the SLICOT and PLiCMR routines select r so that $\sigma_r > \max(\tau_1, n \cdot \varepsilon \cdot \sigma_1) > \sigma_{r+1}$, where τ_1 is a user-specified tolerance threshold. In our case, we set $\tau_1 = \eta \cdot \sigma_1$, where the value η is adjusted for each particular case as shown in the table. The SPA and HNA methods also employ a second tolerance threshold equal to $\max(\tau_2, n \cdot \varepsilon \cdot \sigma_1)$ in order to determine a minimal realization of the system. In our experiments we set $\tau_2 = 0$.

Example	n	m	p	σ_1	η	r
Eady	598	1	1	$9.93e{+2}$	$1.0e{-3}$	9
CDplayer	120	2	2	$1.17e{+6}$	$1.0e{-8}$	42
FOM	1006	1	1	$5.00e{+1}$	$1.0e{-3}$	10
PDE	84	1	1	$5.34e{+0}$	$1.0e{-3}$	2
Heat-c	200	1	1	$3.25e{-2}$	$1.0e{-3}$	4
ISS	270	3	3	$5.79e{-2}$	$1.0e{-3}$	36
Build	48	1	1	$2.50e{-3}$	$1.0e{-3}$	30
Beam	348	1	1	$2.38e{+3}$	$1.0e{-3}$	12

Table 3
Parameters of the examples employed in the numerical evaluation of the parallel model reduction routines.

Table 4 shows the absolute error, $\|\Delta_a\|_\infty = \|G - \hat{G}\|_\infty$, of the reduced-order systems computed with the absolute error model reduction routines and the theoretical bound (12) for the different examples. We used the function `linorm` in order to compute the H_∞ norms of the errors using MATLAB (except in the FOM example, where we had to use gridding on the frequency response plot). This function is based on the SLICOT subroutine `ab13dd` which computes the $\mathcal{L}_\infty$-norm of a continuous- or discrete LTI system. Note that the MATLAB function `normhinf` from the Robust Control Toolbox [16] was not able to compute any of the error norms in the CD player example correctly. The error bound was obtained from the Hankel singular values provided in the data files from [15]. The table shows that the models computed by SLICOT and

the parallel routines are equally good approximations to the original system. Mostly, both models satisfy the theoretical absolute error bound. Only in one case (SPA applied to "Heat-c") the error bound is slightly missed. This can be due to rounding errors which can effect computation of the bound, the absolute error norm, or the reduced-order models. A detailed investigation of this discrepancy between theory and practice is needed here. Further experimental results are reported in [10].

Example	Bound in (12)	`ab09ad`	`pab09ax`	`ab09bd`	`pab09bx`	`ab09cd`	`pab09cx`
Eady	$1.1e{+}0$	$4.6e{-}1$	$4.6e{-}1$	$4.1e{-}1$	$4.1e{-}1$	$2.6e{-}1$	$2.6e{-}1$
CDplayer	$2.4e{-}1$	$2.0e{-}2$	$2.0e{-}2$	$2.2e{-}2$	$2.2e{-}2$	$6.5e{-}2$	$3.6e{-}2$
FOM	$1.0e{-}1$	$1.0e{-}1$	$1.0e{-}1$	$1.0e{-}1$	$1.0e{-}1$	$3.6e{-}2$	$3.6e{-}2$
PDE	$1.0e{-}2$	$4.6e{-}3$	$4.6e{-}3$	$7.4e{-}3$	$7.4e{-}3$	$3.8e{-}3$	$3.8e{-}3$
Heat-c	$3.4e{-}5$	$2.6e{-}5$	$2.6e{-}5$	$4.9e{-}5$	$4.9e{-}5$	$2.9e{-}5$	$2.9e{-}5$
ISS	$1.8e{-}3$	$1.1e{-}4$	$1.1e{-}4$	$1.1e{-}4$	$1.1e{-}4$	$1.5e{-}4$	$1.5e{-}4$
Build	$2.7e{-}5$	$4.9e{-}6$	$4.9e{-}6$	$4.8e{-}6$	$4.8e{-}6$	$6.7e{-}6$	$6.7e{-}6$
Beam	$1.2e{+}1$	$2.4e{+}0$	$2.4e{+}0$	$1.7e{+}0$	$1.7e{+}0$	$1.7e{+}0$	$1.7e{+}0$

Table 4
Absolute error $\|\Delta_a\|_\infty$ of the reduced-order models computed for the examples employed in the numerical evaluation of the absolute error model reduction routines.

Table 5 shows the relative error $\|G-\hat{G}\|_\infty/\|G\|_\infty$ of the reduced-order systems computed with the serial and the parallel BST routines and the theoretical bound (18) for the different examples. In those cases where a regularization is necessary we used $D = [\, I_p \; 0\,]$. (Again the H_∞ norm in the FOM example could not be computed using function `linorm` and gridding had to be used.)

7.2 Parallel performance of the computational routines

We first report the performance of the major computational routines in PLiC-MR. Specifically, we report results for a single iteration of the ARE solver, the spectral division routine based on the sign function, and the linear matrix equation solvers. The Newton method for the ARE requires the solution of a Lyapunov equation at each iteration step. We fix the number of iterations for this LME solver to 10. This value is determined from our experience when evaluating the collection of benchmark examples in [10].

As in practice $m, p \ll n$, in the experiments in this subsection we employ systems with $n/10 = m = p$ and random entries in U$[0, 1]$. The systems are generated with $n/1.1$ stable and $0.1n/1.1$ unstable poles in an attempt to mimic a real case. The dimensions of the numerical problems to be solved

Example	Bound in (18)	`ab09hd`	`pab09hx`
Eady	$4.0e{-3}$	$1.1e{-3}$	$1.1e{-3}$
CDplayer	$7.2e{-2}$	$7.4e{-5}$	$3.2e{-5}$
FOM	$2.1e{-2}$	—	$6.2e{-3}$
PDE	$1.6e{-3}$	$5.5e{-4}$	$5.5e{-4}$
Heat-c	$3.4e{-5}$	$2.6e{-5}$	$2.6e{-5}$
ISS	$1.8e{-3}$	$9.6e{-5}$	$9.6e{-5}$
Build	$2.7e{-5}$	$4.9e{-6}$	$4.9e{-6}$
Beam	$9.1e{+1}$	$4.8e{-1}$	$2.1e{-1}$

Table 5
Relative error $\|G - \hat{G}\|_\infty / \|G\|_\infty$ of the reduced-order models computed for the examples employed in the numerical evaluation of the relative error model reduction routines. (The entry with a dash denotes a problem that could not be solved due to memory restrictions.)

are therefore those that would arise when model reduction is applied to such systems; e.g., spectral division via the matrix sign function is applied on a square matrix of order n; the Sylvester equation involves square coefficient matrices A and B of order $n/1.1$ and $0.1n/1.1$, respectively; once additive decomposition is performed, all other linear and quadratic matrix solvers work on problems of dimension given by $n/1.1$, $m/1.1$, and $p/1.1$.

Our first experiment reports the execution time of a single iteration of the computational routines on a system of order $n = 880$; see Figure 1. This is about the largest size we could evaluate 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 execution of a parallel algorithm on a single node is likely to require a higher time than that of a serial implementation of the algorithm (implemented 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 $n_p = 4$, speed-ups of 2.63, 2.04, and 2.61 are obtained for routines `pdgecrnz`, `pmb05rd`, and `pdgeclne`, respectively. In all 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. In our test, when $n_p = 10$, speed-ups of only 3.88, 2.93, 4.17 are obtained by routines `pdgecrnz`, `pmb05rd` and `pdgeclne`, respectively.

We next evaluate the performance of a single iteration of the computational routines when the problem size per node is constant. For that purpose, we fix

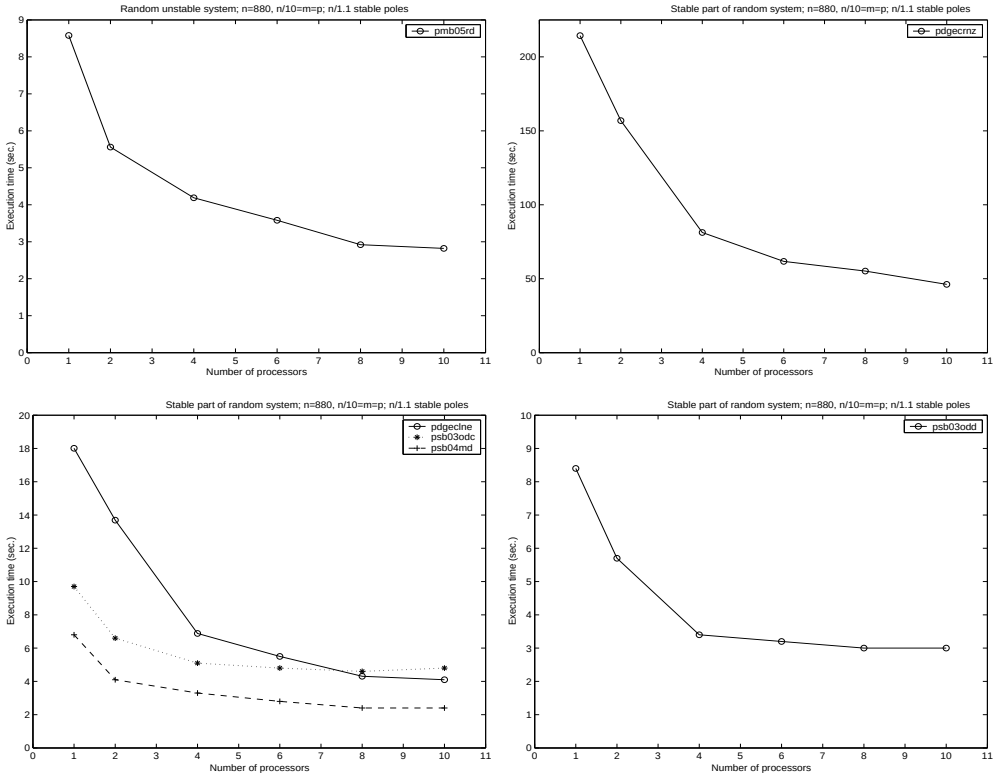

Fig. 1. Execution time of a single iteration of the computational routines.

the order of the system to $n/\sqrt{n_p} = 880$, and we report the Mflops per node in Figure 2.

The figure demonstrates the scalability of our parallel 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. As the major computations in our driver routines are performed in these routines, we can also conclude the scalability of the model reduction parallel algorithms.

7.3 Parallel performance of the driver routines

In this subsection we evaluate the performance of the driver routines for model reduction using five large-scale examples. Three of these examples correspond to continuous-time systems, while the remaining two are discrete-time models. In the experiments we only report results for the BT and BST methods. The performances of the SPA and HNA methods were closely similar to those of the BT method.

Example 1: This continuous LTI system comes from a finite element discretization of a mathematical model for optimal cooling of steel profiles. The process is modeled by a boundary control problem for a linearized 2-dimensional heat equation. The system has 6 inputs and outputs. Different meshes are employed resulting in realizations of order $n = 821$, 1357, 3113, and 5177. As

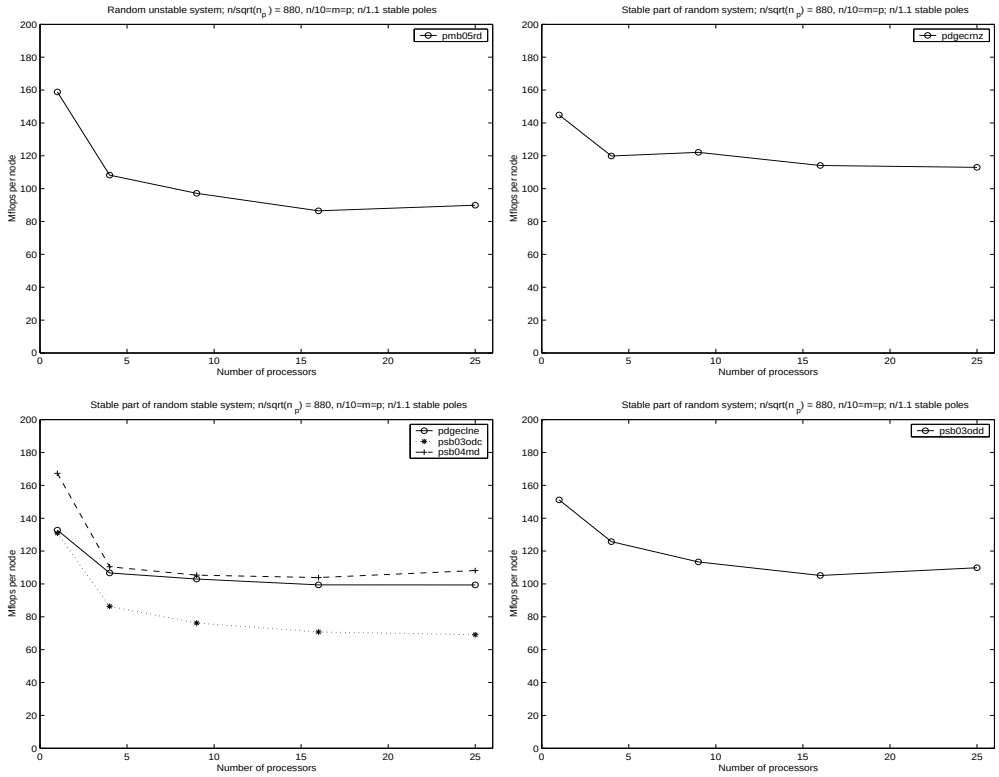

Fig. 2. Mflop rate of a single iteration of the computational routines.

there is no significant gap in the Hankel singular values of the system, we compute in this experiment reduced-order systems of fixed order $r = 40$.

Example 2: (see [34]): This model comes from the simulation of a catalytic tubular reactor used in a *gPROMS* training course. 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 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. A reduced-order system with r=9 states is computed in this example.

Example 3: (see [34]): The model in the previous example is here discretized using zero-order hold with sampling time $\Delta t = 0.1$ sec. such that $x_k = x(k\Delta t)$, $k = 0, 1, 2, \ldots$. Again, a reduced system of order r=9 is obtained for this example.

Example 4: (see [15]): The heat equation in this case is an example of a semidiscretized point control problem for a parabolic PDE. The given equation models the heat diffusion in a 1-dimensional thin rod with a single heat source. The equation is parameterized by a scalar α that we set to $\alpha = 0.01$. The spatial domain is discretized in segments of length $h = \frac{1}{N+1}$ and centered differences are used to approximate the diffusion operator. A heat source is

assumed to be located at 1/3 of the length and the temperature is measured at 2/3 of the length.
This example, of order $n = N$ and with a single input and a single output ($m = p = 1$), can be scaled to obtain very large systems. We thus employ this case to report results for a problem of dimensions that are close to the maximum that can be solved in a single processor, $n = 800$, and a much larger problem, of order $n = 3000$. A reduced system of order $r = 10$ was computed in both cases.
Example 5: (see [15]): The same equation in the last example is used here. This time, however, a full discretization of the control problem for the heat equation is obtained using the Crank–Nicholson scheme. This results in a discrete LTI system. In this case, the Hankel singular values decay slightly slower than in the continuous case, leading to a higher dimension of the reduced-order system if the same approximation error is to be achieved. We again compute in this case reduced systems of order $r = 10$ for models with $n = 800$ and $n = 3000$ states.

Table 6 reports the execution time of the serial and the parallel BT and BST model reduction routines. The execution times of the BST approach are up to 20 times of those of the BT method. This is easily explained by its much higher computational cost (an ARE needs to be solved in BST). Notice that our driver routines are not a direct parallelization of the serial SLICOT routines, but employ numerical solvers for the LME and the AREs which are different from those used in the serial codes. Therefore, concluding the degree of parallelism of the driver routines from the results in the table is misdirected. Our driver routines are intended to help a control engineer to reduce large-scale models and/or obtain the reduced-order models in a shorter period of time. Both goals are achieved in PLiCMR: The order of the largest model that could be reduced using SLICOT serial routines `ab09ad` and `ab09hd` was in the hundreds. Using the parallel routines `pab09ax` and `pab09hx` in PLiCMR we could reduce models of order around 5000 and 3000, respectively. The use of the parallel algorithms reduced the execution times of the serial routines in all cases. The reduction however is quite different depending on the case. Thus, execution times are obtained for the parallel algorithms that range from 7.51% to 60% of those of the corresponding serial algorithms. Notice here that we are interested here in reducing the execution time as much as possible, not obtaining the best possible speed-up (which would surely be higher had we employed a smaller number of processors in this experiment!).

8 Conclusions

Over the last years we have developed the library PLiCMR for model reduction of large-scale LTI systems on parallel architectures. Using the kernels in this library, efficient model reduction of systems with thousands of states is possible

Example (n, m, p)	r	ab09ad	pab09ax (n_p)	ab09hd	pab09hx (n_p)
Ex. 1 $(821, 6, 6)$	40	227	79 (16)	1819	1060 (16)
Ex. 1 $(1357, 6, 6)$	40	–	203 (16)	–	1053 (16)
Ex. 1 $(3113, 6, 6)$	40	–	701 (25)	–	14730 (25)
Ex. 1 $(5177, 6, 6)$	40	–	2314 (32)	–	–
Ex. 2 $(1171, 6, 4)$	9	678	144 (16)	–	1819 (16)
Ex. 3 $(1171, 6, 4)$	9	218	98 (16)	–	1050 (16)
Ex. 4 $(800, 1, 1)$	10	218	65 (16)	1713	331 (16)
Ex. 4 $(3000, 1, 1)$	10	–	679 (25)	–	6310 (25)
Ex. 5 $(800, 1, 1)$	10	400	30 (16)	1959	331 (16)
Ex. 5 $(3000, 1, 1)$	10	–	370 (25)	–	6266 (25)

Table 6
Execution time (in sec.) of the model reduction routines in SLICOT and PLiCMR. (Entries with a dash denote problems that could not be solved due to memory restrictions.)

on a cluster of moderate dimensions. Iterative algorithms are employed for the solution of the major numerical problems that arise in the model reduction methods resulting in highly parallel algorithms with coarse granularity.
A collection of benchmark examples shows the numerical accuracy and the parallel performance of our approach on a cluster of Intel Pentium II processors.

Acknowledgment

We thank Jens Saak for providing the data sets used in Example 1 of Subsection 7.3.

References

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

[2] Z. Bai, J. Demmel, J. Dongarra, A. Petitet, H. Robinson, and K. Stanley. The spectral decomposition of nonsymmetric matrices on distributed memory parallel computers. *SIAM J. Sci. Comput.*, 18:1446–1461, 1997.

[3] P. Benner. Numerical solution of special algebraic Riccati equations via an exact line search method. In *Proc. European Control Conf. ECC 97* (CD-ROM), Paper 786. BELWARE Information Technology, Waterloo, Belgium, 1997.

[4] P. Benner, R. Byers, E.S. Quintana-Ortí, and G. Quintana-Ortí. Solving algebraic Riccati equations on parallel computers using Newton's method with exact line search. *Parallel Comput.*, 26(10):1345–1368, 2000.

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

[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. Balanced truncation model reduction of large-scale dense systems on parallel computers. *Math. Comput. Model. Dyn. Syst.*, 6(4):383–405, 2000.

[8] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Parallel algorithms for model reduction of discrete-time systems. Berichte aus der Technomathematik, Report 00–18, FB3 – Mathematik und Informatik, Universität Bremen, 28334 Bremen (Germany), December 2000. Available from `http://www.math.uni-bremen.de/zetem/berichte.html`.

[9] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Efficient numerical algorithms for balanced stochastic truncation. *Int. J. Appl. Math. Comp. Sci.*, 11(5):1123–1150, 2001.

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

[11] P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. Numerical solution of discrete stable linear matrix equations on multicomputers. *Parallel Algorithms and Appl.*, 17(1):127–146, 2002.

[12] 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.

[13] M. Buchholz, T. Lüttich, H. Auracher, and W. Marquardt. Pool boiling at high heat fluxes (part I): Local temperature & heat flux fluctuations at the heater surface. In D. Gorenflo and A. Luke, editors, *Proc. Int. Refrig. Conf. Comm. B1, Paderborn*, Paderborn, Germany, October 2001.

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

[15] Y. Chahlaoui and P. Van Dooren. A collection of benchmark examples for model reduction of linear time invariant dynamical systems. SLICOT Working Note 2002–2, February 2002. Available from `http://www.win.tue.nl/niconet/NIC2/reports.html`.

[16] R.Y. Chiang and M.G. Safonov, *Robust Control Toolbox. For Use with* MATLAB. *User's Guide, Version 2*, The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, Mass, 01760, 2000.

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

[18] 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.

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

[20] K. Glover. Multiplicative approximation of linear multivariable systems with L_∞ error bounds. In *Proc. American Control Conf.*, pages 1705–1709, 1986.

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

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

[23] E.J. Grimme, D.C. Sorensen, and P. Van Dooren. Model reduction of state space systems via an implicitly restarted Lanczos method. *Numer. Algorithms*, 12:1–31, 1996.

[24] S. Huss, E.S. Quintana-Ortí, X. Sun, and J. Wu. Parallel spectral division using the matrix sign function for the generalized eigenproblem. *Int. J. of High Speed Computing*, 11(1):1–14, 2000.

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

[26] D.L. Kleinman. On an iterative technique for Riccati equation computations. *IEEE Trans. Automat. Control*, AC-13:114–115, 1968.

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

[28] 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.

[29] A.J. Laub, M.T. Heath, C.C. Paige, and R.C. Ward. Computation of system balancing transformations and other application of simultaneous diagonalization algorithms. *IEEE Trans. Automat. Control*, 34:115–122, 1987.

[30] J.-R. Li and J. White. Low rank solution of Lyapunov equations. *SIAM J. Matrix Anal. Appl.*, 24(1):260–280, 2002.

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

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

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

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

[35] E.S. Quintana-Ortí, G. Quintana-Ortí, X. Sun, and R.A. van de Geijn. A note on parallel matrix inversion. *SIAM J. Sci. Comput.*, 22:1762–1771, 2001.

[36] G. Quintana-Orti, X. Sun, and C.H. Bischof. A BLAS-3 version of the QR factorization with column pivoting. *SIAM J. Sci. Comput.*, 19:1486–1494, 1998.

[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. Model reduction for robust control: A Schur relative error method. *Int. J. Adapt. Cont. and Sign. Proc.*, 2:259–272, 1988.

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

[40] 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.

[41] 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.

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

[43] 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.

[44] 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.

[45] A. Varga. On computing high accuracy solutions of a class of Riccati equations. *Control–Theory and Advanced Technology*, 10(4):2005–2016, 1995.

[46] 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`.

[47] A. Varga. Model reduction software in the SLICOT library. In B.N. Datta, editor, *Applied and Computational Control, Signals, and Circuits*, volume 629 of *The Kluwer International Series in Engineering and Computer Science*, pages 239–282. Kluwer Academic Publishers, Boston, MA, 2001.

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

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