Technische Universität Chemnitz

Sonderforschungsbereich 393

Numerische Simulation auf massiv parallelen Rechnern

Thilo Penzl

Algorithms for Model Reduction of Large Dynamical Systems

Preprint SFB393/99-40

Preprint-Reihe des Chemnitzer SFB 393

SFB393/99-40 **December 1999**

Contents

Author:

Thilo Penzl
Department of Mathematics and Statistics
University of Calgary
Calgary, Alberta, T2N1N4
Canada

Fakultät für Mathematik
Technische Universität Chemnitz
D-09107 Chemnitz
FRG

Algorithms for Model Reduction of Large Dynamical Systems

Thilo Penzl

Abstract

Three algorithms for the model reduction of large-scale, continuous-time, time-invariant, linear, dynamical systems with a sparse or structured transition matrix and a small number of inputs and outputs are described. They rely on low rank approximations to the controllability and observability Gramians, which can efficiently be computed by ADI based iterative low rank methods. The first two model reduction methods are closely related to the well-known square root method and Schur method, which are balanced truncation techniques. The third method is a heuristic, balancing-free technique. The performance of the model reduction algorithms is studied in numerical experiments.

Keywords: dynamical systems, Lyapunov equation, model reduction, balanced truncation, Schur method, square root method, numerical algorithms, sparse matrices.

AMS Subject Classification: 65F30, 65F50, 93A15, 93A25.

1 Introduction

In this paper we consider realizations

$$\begin{aligned} \dot{x}(\tau) &= Ax(\tau) + Bu(\tau) \\ y(\tau) &= Cx(\tau) + Du(\tau) \end{aligned} \tag{1}$$

of continuous-time, time-invariant, linear, dynamical systems, where $A \in \mathbb{R}^{n,n}$, $B \in \mathbb{R}^{n,m}$, $C \in \mathbb{R}^{q,n}$, $D \in \mathbb{R}^{q,m}$, $\tau \in \mathbb{R}$, and n is the *order* of the realization. $\mathbb{R}$ and $\mathbb{R}^{n,m}$ denote the sets of real numbers and real n-by-m matrices, respectively. Together with an initial condition $x(\tau_0) = x_0$, realizations (1) are uniquely described by the matrix tuple (A, B, C, D). When appropriate, we will also use the equivalent notation $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$, which is more common in control theory. The vector-valued functions u, x, and y are referred to as *input*, *state*, and *output* of the system, respectively. In particular, we focus on realizations with large state space dimension (say, $n > 1000$) and small input space and output space dimensions (say, $m, q < \frac{1}{100}n$). Moreover, we assume that the matrix A is sparse or structured. An important source for such dynamical systems are parabolic differential equations. Their semidiscretization w.r.t. the space component leads to systems of type (1). Usually, the dimension n depends on the fineness of the discretization and is very large, whereas m and q are small and independent of the discretization. Large dynamical systems also arise from circuit simulation; e.g., [1].

Often numerical methods for controller design or simulation cannot be applied to very large systems because of their extensive numerical costs. This motivates model reduction, which is the approximation of the original, large realization by a realization of smaller

This work was supported by the Deutscher Akademischer Austauschdienst, Bonn, Germany.

order. In this paper we describe three model reduction algorithms for large systems, which are numerically inexpensive with respect to both memory and computation.

The remainder of this paper is organized as follows. In § 2 we give an introduction to continuous-time, *algebraic Lyapunov equations* (ALEs) and describe an iterative solution method. The reason for this is that many important model reduction methods for small and medium systems as well as our methods for large systems are based on ALEs. § 3.1 contains a brief discussion of model reduction in general. §§ 3.2 and 3.3 deal with existing methods for systems of moderate size and large scale systems, respectively. Moreover, §§ 3.1 and 3.2 provide the foundation for the three model reduction methods for large systems described in § 4. The efficiency of these methods is demonstrated by numerical experiments in § 5. Finally, conclusions are provided in § 6.

2 Numerical solution of Lyapunov equations and the LR-Smith(l) iteration

Besides model reduction several topics in control theory, such as stability analysis [32], stabilization, optimal control [33, 46], solution of algebraic Riccati equations [28, 31], and balancing [35] involve ALEs. These linear matrix equations usually have the structure

$$FX + XF^T = -GG^T, \tag{2}$$

where $G \in \mathbb{R}^{n,t}$ is a rectangular matrix with $t \leq n$ and the matrix $F \in \mathbb{R}^{n,n}$ is *stable*, i.e., $\lambda(F) \subset \mathbb{C}_-$. Here, $\lambda(F)$ denotes the spectrum of F. $\mathbb{C}_-$ is the open left half of the complex plane, i.e., $\mathbb{C}_- = \{a \in \mathbb{C} \mid \operatorname{Re} a < 0\}$, where $\mathbb{C}$ is the set of the complex numbers and $\operatorname{Re} a$ is the real part of a. The stability of F is sufficient for the existence of a solution matrix $X \in \mathbb{R}^{n,n}$, which is unique, symmetric, and positive semidefinite. If the pair (F, G) is controllable, then X is even positive definite. Note that (2) is mathematically equivalent to a system of linear equations with $\mathcal{O}(n^2)$ unknowns. For this reason, ALEs of order $n > 1000$ are said to be of large scale.

The relation between (1) and (2) depends on the particular problem, but in context with model reduction mostly $F = A$ or A^T and $G = B$ or C^T. For this reason, we assume that F is sparse or structured, whereas G is a matrix with very few columns. If the latter is true, the nonnegative eigenvalues of the solution X tend to decay very fast [38]. In this case the solution matrix can be approximated very accurate by a positive semidefinite matrix of relatively low rank. This property is essential for our model reduction algorithms.

The Bartels-Stewart method [2] and the Hammarling method [20] are the direct standard methods for ALEs. Whereas the first is applicable to the more general Sylvester equation, the second tends to deliver more accurate results in the presence of round-off errors. Both methods require the computation of the Schur form of F. As a consequence, they generally cannot profit by sparsity or other structures in the equation. The squared Smith method [48] and the sign function method [40] are iterative methods, which cannot exploit sparsity or structures as well. However, they are of particular interest when dense ALEs should be solved on parallel computers [14, 3]. The alternating direction implicit iteration (ADI) [36, 52] is an iterative method which often delivers good results for sparse or structured ALEs. The solution methods mentioned so far have the computational complexity $\mathcal{O}(n^3)$, except for the ADI method. Its complexity strongly depends on the structure of F and is sometimes better. All methods have the memory complexity $\mathcal{O}(n^2)$ because they generate the dense solution matrix X explicitly. It should be stressed that often the memory complexity rather than the amount of computation is the limiting factor for the applicability of solution methods to large ALEs.

Low rank methods are the only existing methods which can solve very large ALEs. They require that G consists of very few columns and they exploit sparsity or structures in F. The solution X is not formed explicitly. Instead, low rank factorizations, which approximate X, are computed. Note that throughout this paper "low rank" stands for "(resulting in or having) a rank much smaller than n". (We will later call certain matrices, whose numbers of columns are much smaller than n, low rank factors although their column rank may be full.) Most low rank methods [21, 22, 25, 41] are Krylov subspace methods, which are based either on the Lanczos process or on the Arnoldi process; see, e.g., [17, 42]. Furthermore, there are low rank methods [41, 19] based on the explicit representation of the ALE solution in integral form, e.g., [30]. Two low rank methods related to the ADI iteration and the Smith method were proposed in [37]. Here, we describe the *cyclic low rank Smith method* (LR-Smith(l)), which is the more efficient of both methods. Note that in the following algorithm and the remainder of this paper I_n denotes the n-by-n identity matrix.

Algorithm 1 (LR-Smith(l) iteration [37])

INPUT: F, G, $\mathcal{P} = \{p_i\}_{i=1}^{l} \subset \mathbb{C}_-$

OUTPUT: $Z = Z_{i_{max}l}$, such that $ZZ^T \approx X$

(1. Find a suitable transformation matrix H and transform the equation: $F := HFH^{-1}$, $G := HG$.)

2. $Z_1 = \sqrt{-2p_1}(F + p_1 I_n)^{-1} G$

FOR $i = 2, \dots, l$

3. $Z_i = \left[\; (F - p_i I_n)(F + p_i I_n)^{-1} Z_{i-1} \quad \sqrt{-2p_i}(F + p_i I_n)^{-1} G \;\right]$

END

4. $Z^{(l)} = Z_l$

FOR $i = 1, 2, \dots, i_{max} - 1$

5. $Z^{((i+1)l)} = \left(\prod_{j=1}^{l} (F - p_j I_n)(F + p_j I_n)^{-1} \right) Z^{(il)}$

6. $Z_{(i+1)l} = \left[\; Z_{il} \quad Z^{((i+1)l)} \;\right]$

END

(7. Transform the factor of the approximate solution back: $Z_{i_{max}l} := H^{-1} Z_{i_{max}l}$.)

Numerical aspects of this method are discussed in detail in [37]. However, some remarks should be made here. The LR-Smith(l) iteration is mathematically equivalent to the ADI iteration, where $p_1, \dots, p_l$ are cyclically used as shift parameters. Usually, a small number of different parameters is sufficient to achieve an almost optimal rate of convergence (say, $l \in \{10, \dots, 25\}$). It is important to use a set $\mathcal{P}$ of pairwise distinct shift parameters, which is closed under complex conjugation (i.e., if $p \in \mathcal{P}$, then $\bar{p} \in \mathcal{P}$). This ensures that

Z is a real matrix. Such suboptimal shift parameters can be computed efficiently by a heuristic algorithm [37, Algorithm 1]. In practical implementations the iteration should be stopped when round-off errors start to dominate the difference between the exact solution and the numerically computed approximate solution ZZ^T, which is characterized by the stagnation of the normalized residual norms (see also (21)) on a level that is in the vicinity of the machine precision. Usually, this is the case for $i_{max}l \in \{30, \ldots, 100\}$ if ALEs with a symmetric matrix F are considered. For unsymmetric problems $i_{max}l$ tends to be larger. The resulting factor Z consists of $i_{max}lt$ columns.

Of course, the matrices $(F + p_i I_n)^{-1}$, which are involved in Steps 2, 3, and 5, are not formed explicitly. Instead, systems of type $(F + p_i I_n)x = y$ are solved. If F is sparse, this requires computing and storing l sparse LU factorizations, which are repeatedly used in backsubstitutions. Thus, the complexity of the method strongly depends on nonzero pattern and the bandwidth of F in this case. Therefore, it is often useful to improve this structure by a transformation of the ALE with the nonsingular matrix H (optional Steps 1 and 7). For example, H can be a permutation matrix for bandwidth reduction [5] or a matrix which transforms F into a tridiagonal matrix [15, 44]. However, the matrix H is never formed explicitly and multiplications with H and H^{-1} must be inexpensive. Alternatively, the shifted systems can be solved by iterative methods, such as (preconditioned) Krylov subspace methods; e.g., [42]. In this case Steps 1 and 7 can be omitted.

Basically, the model reduction algorithms proposed in § 4 can also be used in combination with the aforementioned alternative low rank methods for ALEs, which are mainly Krylov subspace methods. Our model reduction algorithms only require the availability of approximations to ALE solutions which have a high accuracy and a very low rank. In general, LR-Smith(l) delivers better results than Krylov subspace methods w.r.t. both criteria; see [37, § 6]. Moreover, the latter often fail for relatively easy problems. Therefore, we prefer LR-Smith(l) to Krylov subspace methods despite the fact that the computational cost of Krylov subspace methods is often lower.

3 Model reduction: preliminaries and some existing methods

3.1 Preliminaries

Assume that we are given a realization (A, B, C, D) of order n. The purpose of model reduction is to find a reduced realization $(\hat{A}, \hat{B}, \hat{C}, \hat{D})$ with $\hat{A} \in \mathbb{R}^{k,k}$, $\hat{B} \in \mathbb{R}^{k,m}$, $\hat{C} \in \mathbb{R}^{q,k}$, $\hat{D} \in \mathbb{R}^{q,m}$, such that the input-output behavior of the reduced system approximates that of the original system in some sense. Here, $k \in \{1, \ldots, n-1\}$ is the (arbitrary, but fixed) order of the reduced realization. Many practically important criteria for assessing the deviation of the reduced system from the original one are based on the difference of the *transfer matrices* G and $\hat{G}$ on the imaginary axis

$$\Delta G(\jmath\omega) = G(\jmath\omega) - \hat{G}(\jmath\omega),$$

where $\jmath = \sqrt{-1}$, $\omega \in \mathbb{R}$, $G(s) = C(sI_n - A)^{-1}B + D$, and $\hat{G}(s) = \hat{C}(sI_k - \hat{A})^{-1}\hat{B} + \hat{D}$. Measuring the difference $\Delta G(\jmath\omega)$ corresponds to comparing the frequency response of both systems, i.e., the response y to a sinusoidal input function u; see, e.g., [47].

There are typically two tasks which require the reduction of the order of a system.

- **Task 1** is the reduction of systems of small or moderate size (say, $n \in \{5, \ldots, 1000\}$ depending on the problem). Although numerical algorithms for controller design applied to such systems mostly require an affordable amount of computation and deliver satisfactory results, the reduction of the model or the controller is often desired because the practical implementation of the controller (in an electrical device, for example) is too complex or too expensive. Here, the main goal is to achieve a particular objective, for example, (sub)minimizing $\Delta G(s)$ w.r.t. the L_2 or L_∞ norm or a frequency weighted norm. The complexity of model reduction methods for such objectives usually prohibits their application to large scale problems.

- **Task 2** is the reduction of systems which are so large that numerical standard methods for controller design or simulation of the system cannot be applied due to their extensive numerical costs (say, $n > 1000$). Methods for controller design have in most cases at least the computational complexity $\mathcal{O}(n^3)$ and the memory complexity $\mathcal{O}(n^2)$. Moreover, they usually cannot benefit from structures in the system. The primary objectives of Task 2 are to replace the system by a smaller one, for which a controller can be designed or which can be simulated with reasonable numerical effort, and to lose only a very small amount of information by the reduction. Ideally, this loss should be of the same magnitude as the inaccuracies, that are caused by round-off or approximation errors in the subsequent controller design or simulation, and the inherent errors in the model of the underlying real process.

In practice, a model reduction process for a very large system can consist of two steps, where Task 2 and Task 1 are treated successively. The model reduction methods proposed in this paper are intended to solve problems related to Task 2.

Modal truncation [6], balanced truncation [34, 49, 43, 51], singular perturbational model reduction [10], frequency weighted balanced truncation [7], optimal Hankel norm approximation [16] are well-known model reduction methods for stable systems. All these methods mainly focus on Task 1. Each requires the solution of eigenvalue problems of order n, which make their standard implementations expensive when large systems should be reduced. However, they are very useful for systems of moderate size. Except for modal truncation each of the above methods is based either explicitly or implicitly on balanced realizations, the computation of which involves the solutions of a pair of ALEs

$$\begin{aligned} AX_B + X_B A^T &= -BB^T && (3) \\ A^T X_C + X_C A &= -C^T C. && (4) \end{aligned}$$

The solution matrices X_B and X_C are called *controllability* and *observability Gramians*. They play an important role in input-state and state-output energy considerations, which provide a motivation for some of the aforementioned model reduction methods as well as the methods proposed in § 4.

In the following theorem [16], $\|u\|_{\mathcal{L}_2(a,b)}$ denotes the $\mathcal{L}_2$ norm of a vector-valued function u, i.e., $\|u\|^2_{\mathcal{L}_2(a,b)} = \int_a^b u(\tau)^T u(\tau) d\tau$. A realization (1) is called *minimal* if both (A, B) and (A^T, C^T) are controllable.

Theorem 1 *Let (1) be a minimal realization with a stable matrix A. Then the solutions X_B and X_C of the ALEs (3) and (4), respectively, are positive definite and*

$$\min_{u \in \mathcal{L}_2(-\infty,0),\, x(0)=x_0} \|u\|^2_{\mathcal{L}_2(-\infty,0)} = x_0^T X_B^{-1} x_0. \qquad (5)$$

Furthermore, if $x(0) = x_0$ *and* $u(\tau) = 0$ *for* $\tau \geq 0$*, then*

$$\|y\|^2_{\mathcal{L}_2(0,\infty)} = x_0^T X_C x_0. \tag{6}$$

Loosely speaking, this means that a large input energy $\|u\|_{\mathcal{L}_2(-\infty,0)}$ is required to steer the system to a (normalized) final state x_0, which is contained in an invariant subspace of X_B related to the smallest eigenvalues of this matrix. Likewise, (normalized) initial states x_0 contained in an invariant subspace of X_C related to the smallest eigenvalues deliver little output energy $\|y\|_{\mathcal{L}_2(0,\infty)}$ and, thus, they hardly have an effect on the output.

Finally, it should be stressed that all model reduction methods in §§ 3.2 and 4 belong to the class of *state space projection methods*. That means the reduced realization of order k is obtained by a projection of the state space to a subspace of dimension k. Assume that $T \in \mathbb{R}^{n,n}$ is a nonsingular matrix such that the chosen subspace is spanned by the first k columns of T. Then the reduced system corresponds to the first k columns and rows of the transformed realization

$$\left[\begin{array}{c|c} T^{-1}AT & T^{-1}B \\ \hline CT & D \end{array}\right],$$

which is equivalent to (1). Using the MATLAB style colon notation we set

$$S_B = T_{(:,1:k)} \qquad \text{and} \qquad S_C = (T^{-T})_{(:,1:k)}. \tag{7}$$

The reduced order realization is given by

$$\left[\begin{array}{c|c} \hat{A} & \hat{B} \\ \hline \hat{C} & \hat{D} \end{array}\right] = \left[\begin{array}{c|c} S_C^T A S_B & S_C^T B \\ \hline C S_B & D \end{array}\right], \tag{8}$$

where

$$S_C^T S_B = I_k \tag{9}$$

holds. This means state space projection methods differ in the choice of matrices S_B, $S_C \in \mathbb{R}^{n,k}$ that fulfill (9). From the numerical point of view one is interested in attaining as small a condition number of S_B and S_C as possible.

3.2 Balanced truncation techniques

The perhaps most popular way to tackle model reduction problems corresponding to Task 1 is balanced truncation. The basic motivation for this technique is provided by Theorem 1 and its subsequent discussion. Unfortunately, the equations (5) and (6) for the input-state and state-output mapping, respectively, suggest different subspaces for a state space projection. However, both parts can be treated simultaneously after a transformation of the system (A, B, C, D) with a nonsingular matrix $T \in \mathbb{R}^{n,n}$ into a balanced system $(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D}) = (T^{-1}AT, T^{-1}B, CT, D)$; see [35]. A realization $(\tilde{A}, \tilde{B}, \tilde{C}, \tilde{D})$ with a stable matrix $\tilde{A}$ and the corresponding transformed Gramians $\tilde{X}_B = T^{-1}X_B T^{-T}$ and $\tilde{X}_C = T^T X_C T$ is called *balanced* if

$$\tilde{X}_B = \tilde{X}_C = \operatorname{diag}(\sigma_1, \ldots, \sigma_n), \tag{10}$$

where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n > 0$. The values σ_i are called *Hankel singular values.* A balanced realization exists if (A, B, C, D) is a stable, minimal realization; e.g., [16]. The result of the basic balanced truncation algorithm [34] is given by (8), where the balancing transformation matrix T is used to define the matrices S_B and S_C in (7). If $\sigma_k \neq \sigma_{k+1}$, the reduced order realization is minimal, stable, and balanced. Its Gramians are equal to $\mathrm{diag}(\sigma_1, \ldots, \sigma_k)$. One of the main attractions of balanced truncation is the availability of the following L_∞ error bound, which was independently derived in [7] and [16],

$$\|\Delta G\|_{L_\infty} = \sup_{\omega \in \mathbb{R}} \|G(\jmath\omega) - \hat{G}(\jmath\omega)\| \leq 2 \sum_{i=k+1}^{n} \sigma_i \,. \tag{11}$$

Here, $\|\cdot\|$ is the spectral norm of a matrix. Finally, it should be mentioned that there exist several implementations of balanced truncation model reduction algorithms [34, 49, 43, 51]. They deliver reduced order realizations with identical transfer matrices but differ in their numerical robustness.

3.3 Methods based on Padé approximation and Krylov subspaces

Motivated by applications in circuit simulation, quite a large number of publications on model reduction of large systems (Task 2) have appeared in the last few years. The majority of the proposed algorithms are based on Padé approximation and Krylov subspaces. For this reason, we briefly sketch the underlying principle of these algorithms despite the fact that the algorithms proposed in § 4 are based on a completely different approach. For simplicity, we only consider the special case, where $m = 1$, $q = 1$, and $D = 0$, in this section. See [11] for a detailed survey.

Let $\check{u}(s)$ and $\check{y}(s)$ be the Laplace transformed of $u(t)$ and $y(t)$, respectively. It is well-known that the frequency domain representation

$$\check{y}(s) = G(s)\check{u}(s) \qquad \text{with} \qquad G(s) = C(sI_n - A)^{-1}B,$$

which maps the transformed input to the transformed output without reference to the (transformed) state, is mathematically equivalent to the state space representation (1) of a dynamical system provided that $x(0) = 0$. The basic principle of the Padé approximation based model reduction algorithms (*Padé algorithms*) is an expansion of $G(s)$ about a point $s_0 \in \mathbb{C} \cup \{\infty\}$. A frequent choice is $s_0 = \infty$ although other or even multiple expansion points (e.g., [13]) can be used. For example, the expansion about infinity delivers

$$G(s) = \sum_{i=0}^{\infty} \frac{1}{s^{i+1}} M_i \qquad \text{with} \qquad M_i = CA^iB, \tag{12}$$

which can be interpreted as a Taylor series, that converges if $|s|$ is sufficiently large. The matrices M_i, which are called *Markov parameters*, are system invariants. The leading $2n$ parameters determine the system uniquely. Expansions of $G(s)$ about infinity as well as other points can be rewritten in terms of a rational function in s. More precisely,

$$G(s) = \frac{\psi_{n-1}(s)}{\varphi_n(s)},$$

where ψ_{n-1} and φ_n are polynomials of degree at most $n-1$ and n, respectively. The reduction of the system order is now achieved by computing a rational function of the type

$$\hat{G}(s) = \frac{\psi_{k-1}(s)}{\varphi_k(s)}, \tag{13}$$

which corresponds to a realization $(\hat{A}, \hat{B}, \hat{C}, 0)$ of reduced order provided that $k < n$. In view of the series (12) a sensible choice of the polynomials ψ_{k-1} and φ_k is that for which

$$G(s) - \hat{G}(s) = \sum_{i=2k}^{\infty} \frac{1}{s^{i+1}} M_i$$

holds. That means, the leading $2k-1$ Markov parameters of G and $\hat{G}$ coincide. This procedure is known as *moment matching*. The resulting function $\hat{G}$ is called *Padé approximant* of G.

In the last decade two basic approaches to compute the reduced system corresponding to the Padé approximant (13) have been pursued although model reduction based on Padé approximation was studied much earlier; e.g., [45]. The so-called *asymptotic waveform evaluation* (AWE) [39] computes the reduced system in a way that involves computing the leading Markov parameters of G and the coefficients of the polynomials ψ_{k-1} and φ_k explicitly, which tends to be numerically unstable. The second approach [8, 12, 9], which is called *Padé via Lanczos* (PVL) algorithm, exploits a connection between Padé approximation and Krylov subspace processes [18]. AWE and PVL are mathematically equivalent, but PVL turns out to be numerically more robust in general. For this reason, PVL is usually preferred to AWE in practical applications. There exist a number of variations of the PVL algorithm; see, e.g., [11] and references therein.

A few general aspects in context with Padé algorithms should be discussed here. First, it is not clear, whether they deliver good approximations for values s which lie far away from the expansion point. In contrast to balanced truncation, no L_∞ error bounds are known for Padé algorithms. Unlike the algorithms proposed in § 4, the implementation of such methods often becomes more complicated when $m > 1$ and $q > 1$. An essential advantage of Padé algorithms is that they are not restricted to systems of type (1). They can be applied to more general linear, time-invariant, differential-algebraic equations. Moreover, their numerical costs are quite low.

Finally, we want to stress that there exist a few algorithms [23, 24, 26, 27], which are not directly motivated by Padé approximation and moment matching, but involve Krylov subspace techniques. For example, in [26] a Galerkin technique is proposed, whereas in [24] an approach related to GMRES (e.g., [42]) is pursued to generate a reduced system. In both cases low rank approximations to the Gramians X_B and X_C are involved. These approximations are computed by Krylov subspace methods for ALEs described in [41, 22, 25]. A basic difference to the model reduction methods proposed in § 4 is that the latter can use arbitrary, symmetric, positive semidefinite low rank approximations to the Gramians. In particular, this includes approximations generated by Algorithm 1, which are often more accurate and of lower rank than those computed by Krylov subspace methods for ALEs.

4 Three model reduction methods for large systems

4.1 Low Rank Square Root Method

The original implementation of balanced truncation [34] involves the explicit balancing of the realization (1). This procedure is dangerous from the numerical point of view because the balancing transformation matrix T tends to be highly ill-conditioned. Moreover, the implementation in [34] is restricted to minimal realizations. The so-called *square root method* [49] (see also [43, 51]) is an attempt to cope with these problems. It is constructed in a way that avoids explicit balancing of the system. The method is based on the Cholesky factors of the Gramians instead of the Gramians themselves. In [49] the use of the Hammarling method was proposed to compute these factors, but, basically, any numerical method that delivers Cholesky factors of the Gramians can be applied. For example, a combination of a modified sign function iteration for ALEs and the square root method, which, in particular, allows the efficient model reduction of large dense systems on parallel computers, has been proposed in [4]. For large systems with a structured transition matrix A, the LR-Smith(l) method can be an attractive alternative because the Hammarling method or sign function based methods can generally not benefit from such structures. Algorithm 2, which we refer to as *low rank square root method* (LRSRM), is based on the algorithm proposed in [49] and differs formally from the implementation given there only in Step 1. In the original implementation this step is the computation of exact Cholesky factors, which may have full rank. We formally replace these (exact) factors by (approximating) low rank Cholesky factors to obtain the following algorithm.

Algorithm 2 (Low rank square root method (LRSRM))

INPUT: A, B, C, D, k

OUTPUT: $\hat{A}$, $\hat{B}$, $\hat{C}$, $\hat{D}$

1. Compute low rank factors Z_B and Z_C by Algorithm 1 (or alternative low rank methods), such that $Z_B Z_B^T$ and $Z_C Z_C^T$ are approximate solutions of (3) and (4), respectively.

2. $U_{C0}\Sigma_0 U_{B0}^T := Z_C^T Z_B \quad$ (SVD), $\quad U_C = U_{C0(:,1:k)}, \quad \Sigma = \Sigma_{0(1:k,1:k)}, \quad U_B = U_{B0(:,1:k)}$

3. $S_B = Z_B U_B \Sigma^{-1/2}, \quad S_C = Z_C U_C \Sigma^{-1/2}$

4. $\hat{A} = S_C^T A S_B, \quad \hat{B} = S_C^T B, \quad \hat{C} = C S_B, \quad \hat{D} = D$

In this algorithm we assume that $k \leq \operatorname{rank} Z_C^T Z_B$. Note further that throughout this paper singular value decompositions (SVDs) are arranged so that the diagonal matrix containing the singular values has the same dimensions as the factorized matrix and the singular values appear in nonincreasing order. The use of (approximated) low rank factors of the Gramians reduces the computational cost and the memory requirement of the square root method significantly. Note that we only have to compute an SVD of an r_C-by-r_B matrix, where r_B and r_C are the numbers of columns in Z_B and Z_C, respectively, and $r_B, r_C \ll n$. In contrast, if exact Gramians (of possibly full rank) were used, the implementation would involve an SVD of a square matrix of order n. The complexity of algorithm LRSRM except for Step 1 is $\mathcal{O}(n \max\{r_B^2, r_C^2\})$ w.r.t. computation and $\mathcal{O}(n \max\{r_B, r_C\})$ w.r.t. memory. However, the total complexity depends on the numerical costs for the LR-Smith(l) iterations in Step 1 of Algorithm 4, which in turn strongly depend on the structural and algebraic properties of the matrix A.

4.2 Low Rank Schur Method

An alternative to the basic balanced truncation algorithm described in § 3.2 and the square root method is provided by the so-called *Schur method* [43], which is (in exact arithmetics) mathematically equivalent to the first two methods in the sense that the transfer matrices of the reduced realizations are identical. It has become quite popular because it generates projection matrices S_B and S_C, which have generally much smaller condition numbers compared to those by the square root method.

Algorithm 3 (Schur method for balanced truncation model reduction [43])

INPUT: A, B, C, D, k

OUTPUT: $\hat{A}$, $\hat{B}$, $\hat{C}$, $\hat{D}$

1. Solve (3) and (4).

2. Determine the k largest eigenvalues of X_BX_C and compute orthonormal bases $V_B, V_C \in \mathbb{R}^{n,k}$ for the corresponding right and left, invariant subspaces, respectively, by means of ordered Schur factorizations.

3. $U_C\Sigma U_B^T := V_C^T V_B$ (SVD)

4. $S_B = V_B U_B \Sigma^{-1/2}$, $S_C = V_C U_C \Sigma^{-1/2}$

5. $\hat{A} = S_C^T A S_B$, $\hat{B} = S_C^T B$, $\hat{C} = C S_B$, $\hat{D} = D$

Even if the ALEs (3) and (4) in Step 1 can be solved in an inexpensive way, Algorithm 3 cannot be applied to large systems because a dense eigenvalue problem of order n needs to be solved in Step 2. For this reason we propose the following modification we refer to as *low rank Schur method* (LRSM). We solve the ALEs (3) and (4) approximately by applying Algorithm 1 twice. Assuming that we obtain matrices $Z_B \in \mathbb{R}^{n,r_B}$ and $Z_C \in \mathbb{R}^{n,r_C}$, such that $X_B \approx Z_B Z_B^T =: \check{X}_B$, $X_C \approx Z_C Z_C^T =: \check{X}_C$, and $\max\{r_B, r_C\} \ll n$, we then formally replace X_BX_C by the approximation $\check{X}_B\check{X}_C$ in Step 2 of Algorithm 3. The basic idea of our approach is now to avoid forming $\check{X}_B\check{X}_C$ explicitly in this step. Instead, we generate a low rank factorization of this matrix product, which enables us to compute V_B and V_C in a more efficient way. Note that $r = \operatorname{rank} \check{X}_B\check{X}_C \leq \min\{r_B, r_C\} \ll n$. Steps 3–5 of Algorithm 3 remain the same. Our modification of Step 2 should be described in detail now.

First, we determine an “economy size” SVD (that means the version of the SVD where the diagonal matrix containing the singular values is square and has full rank) of the product $\check{X}_B\check{X}_C$, which reveals its low rank structure. For this purpose, we compute “economy size” QR factorizations $Z_B = Q_{B1}R_B$ and $Z_C = Q_{C1}R_C$ with $Q_{B1} \in \mathbb{R}^{n,r_B}$ and $Q_{C1} \in \mathbb{R}^{n,r_C}$. After that, an “economy size” SVD

$$R_B Z_B^T Z_C R_C^T =: Q_{B2} D Q_{C2}^T$$

with the nonsingular diagonal matrix $D \in \mathbb{R}^{r,r}$ is computed. Defining $Q_B = Q_{B1}Q_{B2}$ and $Q_C = Q_{C1}Q_{C2}$, we finally get the desired SVD of $\check{X}_B\check{X}_C$ by

$$\check{X}_B\check{X}_C = Z_B Z_B^T Z_C Z_C^T = Q_{B1}R_B Z_B^T Z_C R_C^T Q_{C1}^T = Q_B D Q_C^T. \tag{14}$$

By means of this equation we now compute an orthonormal basis for the right, dominant, invariant subspace of $\check{X}_B\check{X}_C$. Obviously, the right, invariant subspace related to the nonzero eigenvalues of $\check{X}_B\check{X}_C$ coincides with the range of Q_B. Because of

$$\check{X}_B\check{X}_C Q_B = Z_B Z_B^T Z_C Z_C^T Q_B = Q_B D Q_C^T Q_B \tag{15}$$

all nonzero eigenvalues of $\check{X}_B\check{X}_C$ are eigenvalues of the matrix $DQ_C^TQ_B$ as well. Assuming that $r \ll n$, the merit of our approach is that we have to determine the dominant eigenvalues of the r-by-r matrix $DQ_C^TQ_B$ instead of those of the n-by-n matrix $\check{X}_B\check{X}_C$ itself. More precisely, we compute an ordered Schur factorization

$$DQ_C^TQ_B =: P_B T_B P_B^T = \left[\begin{array}{cc} P_{B1} & P_{B2} \end{array}\right] \left[\begin{array}{cc} T_{B11} & T_{B12} \\ 0 & T_{B22} \end{array}\right] \left[\begin{array}{cc} P_{B1} & P_{B2} \end{array}\right]^T, \tag{16}$$

where the block $T_{B11} \in \mathbb{R}^{k,k}$ ($k \le r$) corresponds to the k largest eigenvalues of T_B. The desired orthonormal basis in the right, dominant, invariant subspace is formed by the columns of the matrix $V_B = Q_B P_{B1}$ because

$$\check{X}_B\check{X}_C V_B = Z_B Z_B^T Z_C Z_C^T Q_B P_B = Q_B D Q_C^T Q_B P_B = Q_B P_{B1} T_{B11} = V_B T_{B11},$$

which in turn is a consequence of (15) and (16). An orthonormal basis in the left, dominant, invariant subspace of $\check{X}_B\check{X}_C$ is obtained by an analogous procedure.

Piecing the single steps together we obtain the following algorithm.

Algorithm 4 (Low rank Schur method (LRSM))

INPUT: A, B, C, D, k

OUTPUT: $\hat{A}$, $\hat{B}$, $\hat{C}$, $\hat{D}$

1. Compute low rank factors Z_B and Z_C by Algorithm 1 (or alternative low rank methods), such that $Z_B Z_B^T$ and $Z_C Z_C^T$ are approximate solutions of (3) and (4), respectively.
2. $Q_{B1}R_B := Z_B, \quad Q_{C1}R_C := Z_C$ ("economy size" QR factorizations)
3. $Q_{B2}DQ_{C2}^T := R_B Z_B^T Z_C R_C^T$ ("economy size" SVD)
4. $Q_B = Q_{B1}Q_{B2}, \quad Q_C = Q_{C1}Q_{C2}$
5. $P_B T_B P_B^T := DQ_C^T Q_B, \quad P_C T_C P_C^T := D^T Q_B^T Q_C$ (Schur factorizations with nonincreasing ordered eigenvalues on the main diagonals of T_B and T_C)
6. $V_B = Q_B P_{B(:,1:k)}, \quad V_C = Q_C P_{C(:,1:k)}$
7. $U_C \Sigma U_B^T := V_C^T V_B$ (SVD)
8. $S_B = V_B U_B \Sigma^{-1/2}, \quad S_C = V_C U_C \Sigma^{-1/2}$
9. $\hat{A} = S_C^T A S_B, \quad \hat{B} = S_C^T B, \quad \hat{C} = C S_B, \quad \hat{D} = D$

The execution of Steps 2–9 has the complexity $\mathcal{O}(n \max\{r_B^2, r_C^2\})$ w.r.t. computation and

$\mathcal{O}(n \max\{r_B, r_C\})$ w.r.t. memory, whereas the original Algorithm 3 has the computational complexity $\mathcal{O}(n^3)$ and the memory complexity $\mathcal{O}(n^2)$.

The square root method and the Schur method based on exact Gramians are known to be mathematically equivalent in the sense that they deliver reduced realizations with identical transfer matrices. The analog statement holds also for LRSRM and LRSM. In the following lemma we assume that in LRSRM and LRSM the same low rank factors Z_B and Z_C are used und that both algorithms generate reduced realizations of order k.

Lemma 1 *Let $k < \operatorname{rank} Z_C^T Z_B$ and $\sigma_1 \geq \sigma_2 \geq \ldots$ be the singular values of $Z_C^T Z_B$. If $\sigma_{k+1} \neq \sigma_k$, LRSRM and LRSM deliver reduced realizations, which have identical transfer matrices.*

Proof: Throughout this proof we provide the matrices in Algorithm 2 with a tilde (e.g., $\tilde{\Sigma}$) to distinguish them from the variables that correspond to Algorithm 4. In Steps 2–6 of Algorithm 4 we compute orthonormal bases V_B and V_C in the right and left, k-dimensional, dominant, invariant subspaces of $Z_B Z_B^T Z_C Z_C^T$. Observe that

$$\lambda(Z_B Z_B^T Z_C Z_C^T)\backslash\{0\} = \lambda(Z_C^T Z_B Z_B^T Z_C)\backslash\{0\} = \lambda(\tilde{\Sigma}_0 \tilde{\Sigma}_0^T)\backslash\{0\},$$

where eigenvalue multiplicities are retained by the equalities. Thus, there exists a nonsingular matrix $W_{B1} \in \mathbb{R}^{k,k}$, such that V_B fulfills

$$Z_B Z_B^T Z_C Z_C^T V_B W_{B1} = V_B W_{B1} \tilde{\Sigma}^2.$$

Note that $\tilde{\Sigma}$ and the dominant invariant subspaces are uniquely defined because $\sigma_{k+1} \neq \sigma_k$. Furthermore, it follows from Step 2 in Algorithm 2 that

$$Z_B Z_B^T Z_C Z_C^T \tilde{V}_B = \tilde{V}_B \tilde{\Sigma}^2.$$

As a consequence, $\tilde{V}_B = V_B W_{B2}$ holds for a certain nonsingular matrix $W_{B2} \in \mathbb{R}^{k,k}$. This and a comparison of Step 4 in Algorithm 2 and Step 8 in Algorithm 4 reveal that a nonsingular matrix $W_{B3} \in \mathbb{R}^{k,k}$ exists, such that $\tilde{S}_B = S_B W_{B3}$. Analogously, it can be shown that $\tilde{S}_C = S_C W_{C3}$ holds for a certain nonsingular matrix $W_{C3} \in \mathbb{R}^{k,k}$. It is easy to prove that $S_C^T S_B = \tilde{S}_C^T \tilde{S}_B = I_k$, which leads to $W_{C3} = W_{B3}^{-T}$. Finally, we obtain $\tilde{S}_C^T A \tilde{S}_B = W_{B3}^{-1} S_C^T A S_B W_{B3}$, $\tilde{S}_C^T B = W_{B3}^{-1} S_C^T B$, and $C\tilde{S}_B = C S_B W_{B3}$, from which the statement of the lemma follows. $\square$

4.3 Dominant Subspaces Projection Model Reduction

The *dominant subspaces projection model reduction* (DSPMR) is motivated by Theorem 1. As a consequence of (5) and (6), the invariant subspaces of the Gramians X_B and X_C w.r.t. the maximal eigenvalues are the state subspaces which dominate the input-state and state-output behavior of the system (1). The subspaces $\operatorname{range} Z_B$ and $\operatorname{range} Z_C$ can be considered as approximations to these dominant subspaces because $Z_B Z_B^T \approx X_B$ and $Z_C Z_C^T \approx X_C$. The straightforward idea is now to use the sum of both subspaces for a state space projection. That means the reduced realization is given by (8), where S_B is a matrix, such that $\operatorname{range} S_B = \operatorname{range} Z_B + \operatorname{range} Z_C$. More precisely, we choose S_B as a matrix with orthonormal columns and set $S_B = S_C =: S$ because this results in an orthoprojection, which is advantageous in view of numerical robustness. The basic version of our algorithm is given as follows.

Algorithm 5 (Dominant subspaces projection model reduction - basic version (DSPMR-B))

INPUT: A, B, C, D

OUTPUT: $\hat{A}$, $\hat{B}$, $\hat{C}$, $\hat{D}$, k

1. Compute low rank factors Z_B and Z_C by Algorithm 1 (or alternative low rank methods), such that $Z_B Z_B^T$ and $Z_C Z_C^T$ are approximate solutions of (3) and (4), respectively.

2. Compute an orthonormal basis S in $\operatorname{range} Z_B + \operatorname{range} Z_C$, e.g., by an "economy size" (rank-revealing) QR decomposition or an "economy size" SVD of the matrix $\left[\begin{array}{cc} Z_B & Z_C \end{array}\right]$ and set $k = \operatorname{rank} S$.

3. $\hat{A} = S^T A S, \quad \hat{B} = S^T B, \quad \hat{C} = C S, \quad \hat{D} = D$

There exists the following connection between LRSRM, LRSM, and DSPMR-B.

Lemma 2 *Let $k_1 \leq \operatorname{rank} Z_B^T Z_C$ be the order of the reduced system generated by LRSRM and LRSM. Assume that $\sigma_{k_1+1} \neq \sigma_{k_1}$, where $\sigma_1, \sigma_2, \dots$ are the nonincreasingly ordered singular values of $Z_B^T Z_C$. Denote the left (right) n-by-k_1 matrices used for a state space projection (8) in these two algorithms by S_C^{LRSRM} (S_B^{LRSRM}) and S_C^{LRSM} (S_B^{LRSM}), respectively. $S^{DSPMR} \in \mathbb{R}^{n,k_2}$ is the matrix S generated in Step 2 of Algorithm 5, where $k_2 = \operatorname{rank} \left[\begin{array}{cc} Z_B & Z_C \end{array}\right] \geq k_1$. Then,*

$$\operatorname{range} S_C^{LRSRM} = \operatorname{range} S_C^{LRSM} \subseteq \operatorname{range} Z_C \subseteq \operatorname{range} S^{DSPMR}, \tag{17}$$

$$\operatorname{range} S_B^{LRSRM} = \operatorname{range} S_B^{LRSM} \subseteq \operatorname{range} Z_B \subseteq \operatorname{range} S^{DSPMR}. \tag{18}$$

Proof: The equalities follow from the proof of Lemma 1. The inclusions are easy to derive from Algorithms 2, 4, and 5. □

In this sense, the state space projections of LRSRM and LRSM are contained in that of DSPMR, which, however, generally delivers a reduced realization of larger order.

There is only a case for Algorithm 5 when the rank of S is much smaller than n. However, the rank k of this matrix can still be larger than the desired order of the reduced realization. There are at least two ways to cope with this problem. If k is sufficiently small (say, $k < 500$), standard implementations of model reduction methods for moderately sized systems (Task 1; see §§ 3.1 and 3.2) can be used to reduce the system delivered by Algorithm 5 further. Alternatively, a realization of arbitrary small order can be obtained by a modification of Algorithm 5. This modification is a heuristic choice of a sufficiently small subspace of $\operatorname{range} Z_B + \operatorname{range} Z_C$. We propose to choose the columns of S as the k dominant, left singular vectors of the matrix

$$Z = \left[\begin{array}{cc} \frac{1}{\|Z_B\|_F} Z_B & \frac{1}{\|Z_C\|_F} Z_C \end{array}\right]. \tag{19}$$

The scalar factors $1/\left\|Z_B\right\|_F$ and $1/\left\|Z_C\right\|_F$ are weighting factors with which we try to attain an equilibrium of the input-state and state-output relations. In particular, a scaling of the matrices B and C results in a scaling of $\hat{B}$ and $\hat{C}$ with the same factors, but it does not affect the choice of S.

Algorithm 6 (Dominant subspaces projection model reduction - refined version (DSPMR-R))

INPUT: A, B, C, D, k

OUTPUT: $\hat{A}$, $\hat{B}$, $\hat{C}$, $\hat{D}$

1. Compute low rank factors Z_B and Z_C by Algorithm 1 (or alternative low rank methods), such that $Z_B Z_B^T$ and $Z_C Z_C^T$ are approximate solutions of (3) and (4), respectively.

2. $Z = \left[\begin{array}{cc} \frac{1}{\|Z_B\|_F} Z_B & \frac{1}{\|Z_C\|_F} Z_C \end{array} \right]$

3. $UEV^T := Z$ (“economy size” SVD), $S = U_{(:,1:k)}$

4. $\hat{A} = S^T A S$, $\hat{B} = S^T B$, $\hat{C} = CS$, $\hat{D} = D$

The reduced system $(\hat{A}, \hat{B}, \hat{C}, \hat{D})$ is stable if $A + A^T < 0$. Although instability of the reduced system has not been encountered in our numerical experiments, stability is not guaranteed in general.

5 Numerical experiments

We demonstrate the performance of LRSRM, LRSM, and DSPMR-R in numerical experiments with three large test examples of dynamical systems (1). These experiments were carried out on a SUN Ultra 450 workstation at the Department of Mathematics and Statistics of the University of Calgary. The computations were performed with MATLAB 5.2 using IEEE double precision arithmetic (machine precision $\epsilon_{mach} = 2^{-52} \approx 2.2 \cdot 10^{-16}$). Our implementation makes use of the data structure for sparse matrices offered by MATLAB whenever this is profitable.

Example 1 This example is a simplified linear model of a nonlinear problem arising from a cooling process, which is part of the manufacturing method for steel rails [50]. The temperature of the rail is lowered by water sprayed through several nozzles on its surface. Since the problem is “frozen” w.r.t. one space dimension and symmetric w.r.t. another, it is sufficient to consider the problem related to half the cross-section Ω of the rail, where homogeneous Neumann boundary conditions are imposed on the artificial boundary segment Γ_7 (see Figure 1). The pressure of the nozzles can be steered independently for different sections $\Gamma_1, \ldots, \Gamma_6$ of the surface. This corresponds to the boundary control of a two-dimensional instationary heat equation in $\mathbf{x} = \mathbf{x}(\tau, \xi_1, \xi_2)$. The nozzle pressures provide the input signals $u_i = u_i(\tau)$, which form the right hand side of the third type boundary conditions (20). The output signals of this model are given by the temperature in several interior observation points marked by small circles in Figure 1. After a proper scaling of the physical quantities we get the parabolic differential equation

$$\begin{aligned}
\frac{\partial}{\partial \tau}\mathbf{x} &= \frac{\partial^2}{\partial \xi_1^2}\mathbf{x} + \frac{\partial^2}{\partial \xi_2^2}\mathbf{x} && (\xi_1, \xi_2) \in \Omega \\
\mathbf{x} + \frac{\partial}{\partial \vec{n}}\mathbf{x} &= u_i && (\xi_1, \xi_2) \in \Gamma_i,\ i = 1, \ldots, 6 \\
\frac{\partial}{\partial \vec{n}}\mathbf{x} &= 0 && (\xi_1, \xi_2) \in \Gamma_7.
\end{aligned} \tag{20}$$

We utilized the MATLAB PDE toolbox to obtain a finite element discretization of the problem. Figure 1 shows the initial triangularization. The actual triangularization is the result of two steps of regular mesh refinement, i.e., in each refinement step all triangles are split into four congruent triangles. The final result of this procedure is a generalized dynamical system of the type $M\dot{\tilde{x}} = -N\tilde{x} + \tilde{B}u$, $y = \tilde{C}\tilde{x}$ with dimensions $n = 3113$, $m = 6$, and $q = 6$, where M is the mass matrix and N is the stiffness matrix of the discretization. We compute a Cholesky factorization $U_M U_M^T = M$ of the sparse, symmetric, positive definite, well-conditioned mass matrix. Defining $A = -U_M^{-1} N U_M^{-T}$, $B = U_M^{-1}\tilde{B}$, and $C = \tilde{C} U_M^{-T}$ leads to a mathematically equivalent standard system (1). Note that the matrix A is never formed explicitly because the result would be a dense matrix. Instead, we exploit the product structure.

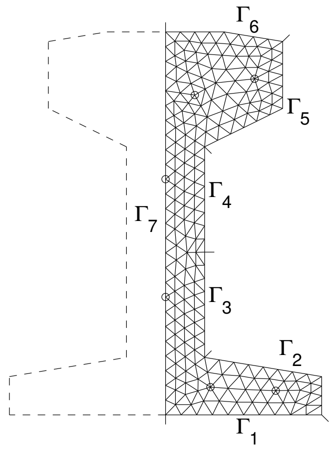

Figure 1: Example 1. Cross-section of the steel rail and initial triangularization of Ω.

Example 2 This example is a dynamical system with dimensions $n = 3600$, $m = 4$, and $q = 2$. It was also used as a test example in [37, Example 5]. The example arises from the control of a process in chromatography. See [29] for background information. The matrix A is sparse and unsymmetric. It has relatively bad algebraic properties. For example, its symmetric part is indefinite and there are eigenvalues of A with dominant imaginary parts. Such properties have usually a negative effect on the convergence of iterative methods for ALEs.

The Bode plots of Examples 1 and 2 are quite smooth. For this reason and because these examples are MIMO systems (i.e., systems with $m, q > 1$), we omit printing such plots for the first two examples. In order to demonstrate that our algorithms are also applicable to systems with Bode plots which are not smooth, we include a third example. The Bode magnitude plot of the following Example 3, that is a purely theoretical test example, shows three spikes; see Figure 2.

Example 3 The system matrices are given as follows, where $e_i \in \mathbb{R}^{i,1}$ is the vector with each entry equal to 1.

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & A_3 & \\ & & & A_4 \end{bmatrix}, \quad A_1 = \begin{bmatrix} -1 & 100 \\ -100 & -1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -1 & 200 \\ -200 & -1 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} -1 & 400 \\ -400 & -1 \end{bmatrix}, \quad A_4 = -\operatorname{diag}(1, 2, \ldots, 1000), \quad B = \begin{bmatrix} 10e_6 \\ e_{1000} \end{bmatrix}, \quad C = B^T.$$

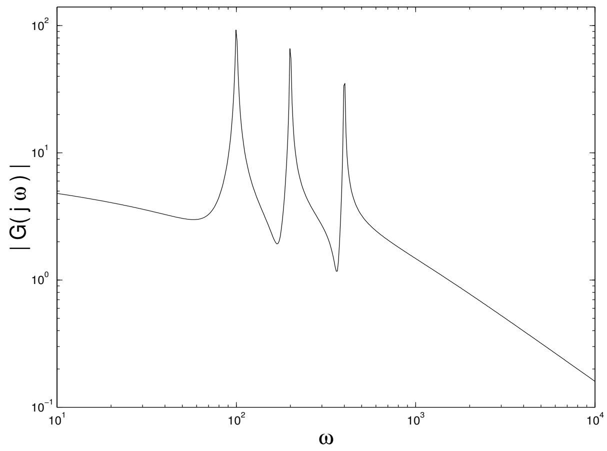

Figure 2: Example 3. Bode magnitude plot.

In our tests we apply each model reduction method to each test example three times (three "runs").

- In **Run 1** we compute the low rank approximations to the Gramians very accurately. That means we do not terminate the LR-Smith(l) iteration in Step 1 of each method before a stagnation of the iteration caused by round-off errors is observed. Moreover, we allow a relatively large order of the reduced model. In the spirit of Task 2 (see § 3.1) our goal is to attain as small an approximation error as possible. Ideally, the magnitude of this error should be in the vicinity of the machine precision.

- In **Run 2** we use the same quite accurate low rank factors Z_B and Z_C as in Run 1, but we limit the maximal order of the reduced model to a smaller value. This can be considered as an attempt of Task 2 and Task 1 in a single sweep.

- The number of LR-Smith(l) iteration steps is restricted to a small value in **Run 3**, which generally leads to relatively inaccurate approximations to the Gramians. Indeed, in a practical implementation r_B and r_C, the numbers of columns in the Cholesky factors Z_B and Z_C, respectively, which are proportional to the number of iteration steps, may be restricted by memory limits. Given such relative inaccurate approximations, we try to generate as good a reduced order model as possible without fixing the reduced order k a priori. Instead, k is chosen as the numerical rank of $Z_C^T Z_B$ in LRSRM and LRSM, whereas k is the numerical rank of Z given by (19) for DSPMR. This means that the reduced orders of the realizations delivered by DSPMR are generally larger than those of the realizations by LRSRM and LRSM.

Each test run of our numerical experiments can be subdivided into two stages. In the first stage we run the LR-Smith(l) iteration twice to compute the matrices Z_B and Z_C. Within this iteration we solve sparse or structured systems of linear equations directly although iterative solvers (see [42], for example) could be used instead. To reduce the numerical costs, the bandwidth of the involved sparse matrices (M and N in Example 1, A in Example 2) is reduced by a suitable simultaneous column-row reordering, which is done by means of the MATLAB function `SYMRCM`. This corresponds to Step 1 in Algorithm 1. We use l-cyclic shift parameters p_i computed by the algorithm proposed in [37, Algorithm 1]. The accuracy of the approximated ALE solutions is measured by the normalized residual norm (NRN), which is defined as

$$\mathrm{NRN}(Z) = \frac{\left\| FZZ^T + ZZ^TF^T + GG^T \right\|_F}{\|GG^T\|_F} \tag{21}$$

with $(F, G, Z) = (A, B, Z_B)$ or (A^T, C^T, Z_C). The parameter l and the values of r_B, r_C, $\mathrm{NRN}(Z_B)$, and $\mathrm{NRN}(Z_C)$ are shown in Table 1.

		Example 1	Example 2	Example 3
system dimensions (n, m, q)		(3113, 6, 6)	(3600, 4, 2)	(1006,1,1)
l		10	20	12
Run 1, 2	r_B	360	480	72
	r_C	420	240	72
	$\mathrm{NRN}(Z_B)$	$3.4 \cdot 10^{-11}$	$1.2 \cdot 10^{-11}$	$9.7 \cdot 10^{-13}$
	$\mathrm{NRN}(Z_C)$	$1.2 \cdot 10^{-12}$	$8.4 \cdot 10^{-13}$	$1.2 \cdot 10^{-12}$
Run 3	r_B	60	80	12
	r_C	60	40	12
	$\mathrm{NRN}(Z_B)$	$2.2 \cdot 10^{-3}$	$2.2 \cdot 10^{-3}$	$9.0 \cdot 10^{-4}$
	$\mathrm{NRN}(Z_C)$	$3.0 \cdot 10^{-3}$	$2.2 \cdot 10^{-3}$	$9.0 \cdot 10^{-4}$

Table 1: System dimensions and parameters describing the LR-Smith(l) iterations in Step 1 of LRSRM, LRSM, and DSPMR.

The second stage consists of the computation of the reduced order models themselves by LRSRM, LRSM, and DSPMR. It should be noted that the first two methods often deliver reduced models with an unstable matrix $\hat{A}$. We believe that this phenomenon is mainly caused by round-off errors in Run 1 (where high accuracy reduced realizations are computed) and by the use of quite inaccurate Gramians in Run 3. However, there are usually only few slightly unstable modes (i.e., eigenvalues of $\hat{A}$ with a nonnegative real part of very small magnitude). Mostly, these unstable modes are hardly controllable and observable. If unstable modes are encountered, we remove them by modal truncation [6]. That means the order of the reduced system is further decreased by the number of unstable modes in this kind of optional postprocessing. Table 2 displays the order k of the reduced realizations after postprocessing. Furthermore, it is shown whether the reduced realization (before postprocessing) is stable or unstable. Note that in our experiments DSPMR always delivered stable reduced realizations.

		Example 1	Example 2	Example 3
Run 1	LRSRM	194 (u)	173 (u)	46 (u)
	LRSM	197 (u)	196 (u)	50 (s)
	DSPMR	200 (s)	200 (s)	50 (s)
Run 2	LRSRM	40 (s)	40 (s)	10 (s)
	LRSM	40 (s)	40 (s)	10 (s)
	DSPMR	40 (s)	40 (s)	10 (s)
Run 3	LRSRM	54 (u)	38 (u)	12 (s)
	LRSM	54 (u)	38 (u)	12 (s)
	DSPMR	120 (s)	120 (s)	18 (s)

Table 2: Orders of reduced realizations delivered by LRSRM, LRSM, and DSPMR. DSPMR-R is applied in Runs 1 and 2, whereas DSPMR-B is used in Run 3. It is also shown whether the reduced realization is stable (s) or unstable (u).

Next, we study the numerical costs of the algorithms. Table 3 shows the total number of floating point operations ("flops", see [17, § 1.2.4]) required for each test run. These values include the computational cost for both computing of the low rank Cholesky factors and performing the model reduction itself.

		Example 1	Example 2	Example 3
Run 1	LRSRM	$1.2 \cdot 10^{10}$	$2.5 \cdot 10^{10}$	$1.8 \cdot 10^{8}$
	LRSM	$2.6 \cdot 10^{10}$	$3.5 \cdot 10^{10}$	$3.3 \cdot 10^{8}$
	DSPMR	$2.7 \cdot 10^{10}$	$4.0 \cdot 10^{10}$	$3.3 \cdot 10^{8}$
Run 2	LRSRM	$9.7 \cdot 10^{9}$	$2.2 \cdot 10^{10}$	$1.3 \cdot 10^{8}$
	LRSM	$2.3 \cdot 10^{10}$	$3.2 \cdot 10^{10}$	$2.8 \cdot 10^{8}$
	DSPMR	$2.7 \cdot 10^{10}$	$3.9 \cdot 10^{10}$	$3.3 \cdot 10^{8}$
Run 3	LRSRM	$1.1 \cdot 10^{9}$	$2.6 \cdot 10^{9}$	$4.5 \cdot 10^{7}$
	LRSM	$2.0 \cdot 10^{9}$	$3.3 \cdot 10^{9}$	$4.9 \cdot 10^{7}$
	DSPMR	$1.5 \cdot 10^{9}$	$3.2 \cdot 10^{9}$	$4.9 \cdot 10^{7}$

Table 3: Total numbers of flops required.

The computational costs of LRSM and DSPMR are slightly larger than that of LRSRM. However, each is much smaller than the cost of standard implementations of the balanced truncation method, which involve the computation of Schur factorizations or SVDs of dense n-by-n matrices. A rough estimation of their cost gives $50n^3$ flops, which are $1.5 \cdot 10^{12}$ flops for Example 1 and $2.3 \cdot 10^{12}$ flops for Example 2. Because of the block diagonal structure of A in Example 3, the Gramians could be directly computed within $\mathcal{O}(n^2)$ operations. However, standard balanced truncation algorithms would still require $\mathcal{O}(n^3)$ flops.

The second complexity aspect, which should briefly be discussed here, is the memory requirement of our methods. It is dominated by the amount of memory needed for storing the low rank factors Z_B and Z_C and the LU factors arising from the l LU factorizations of the matrices in the shifted systems of linear equations, which need to be solved in the course of the LR-Smith(l) method. Of course, these quantities strongly depend on the particular problem. However, taking into account that suitably reordered sparse matrices often have a relative small bandwidth (115 for M and N in Example 1, 57 for A in Example 2) and considering the number of columns in the low rank factors given in Table 1 reveal that our methods demand considerably less memory than standard implementations, which usually require storing a few dense n-by-n matrices. Of course, this demand can be reduced even further by solving the shifted linear systems iteratively.

Finally, we show how accurate the reduced order models approximate the original ones. To this end we compare the frequency response of the original system with those of the reduced systems in Figures 3, 4, and 5. There we display the function

$$\|\Delta G(\jmath\omega)\|/c = \|G(\jmath\omega) - \hat{G}(\jmath\omega)\|/c$$

for a certain frequency range $\omega \in [\omega_{min}, \omega_{max}]$. For Examples 1 and 2 we choose $[\omega_{min}, \omega_{max}]$ $= [10^{-10}, 10^{10}]$. For Example 3 we consider the frequency range $[\omega_{min}, \omega_{max}] = [10^1, 10^4]$, which contains the three spikes.

The scalar parameter c, which we define as

$$c = \max_{\omega \in [\omega_{min}, \omega_{max}]} \|G(\jmath\omega)\|,$$

is used for a normalization and can be considered as an approximation to the L_∞ norm of G. That means, our plots show relative error curves in this particular sense. It should be mentioned that, in contrast to the majority of publications on model reduction, no simultaneous Bode plots of the original and reduced systems are used because it would be impossible to distinguish the single curves in that type of plot.

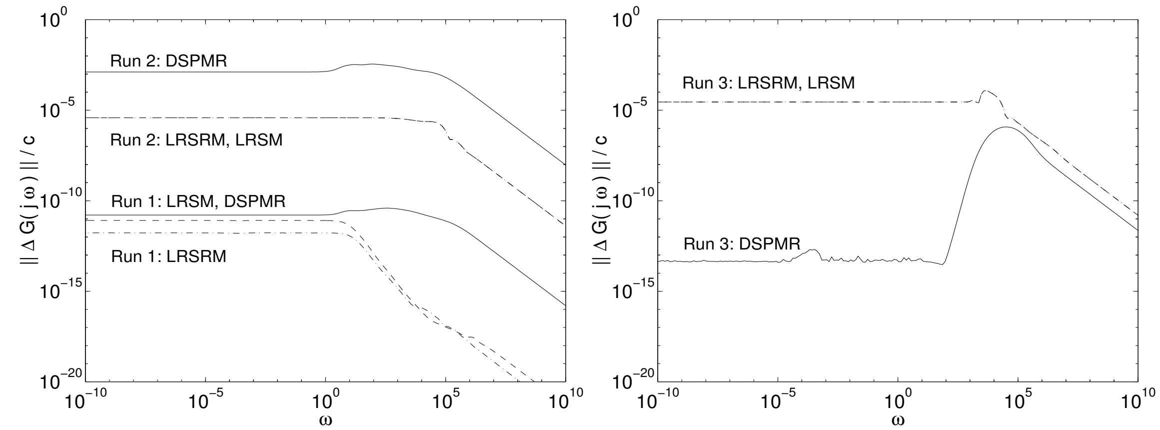

Figure 3: Example 1. Error plots for LRSRM (dashdotted line), LRSM (dashed line), and DSPMR (solid line).

Except for DSPMR in Run 2 for Example 2, our algorithms generate reduced systems whose approximation properties are quite satisfactory. In particular, in Run 1 we attain error norms which are in the vicinity of the given machine precision. Note that the methods mentioned in § 3.3 typically deliver considerably less accurate reduced systems. The

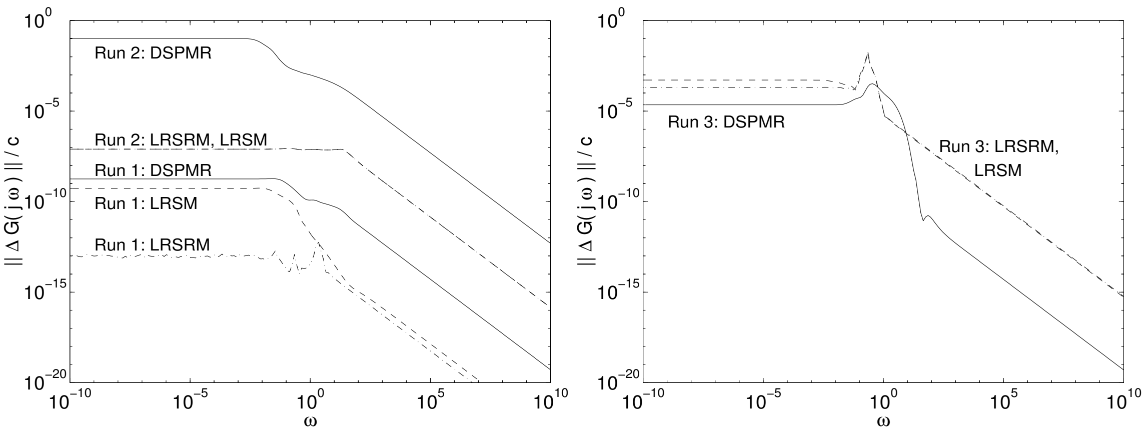

Figure 4: Example 2. Error plots for LRSRM (dashdotted line), LRSM (dashed line), and DSPMR (solid line).

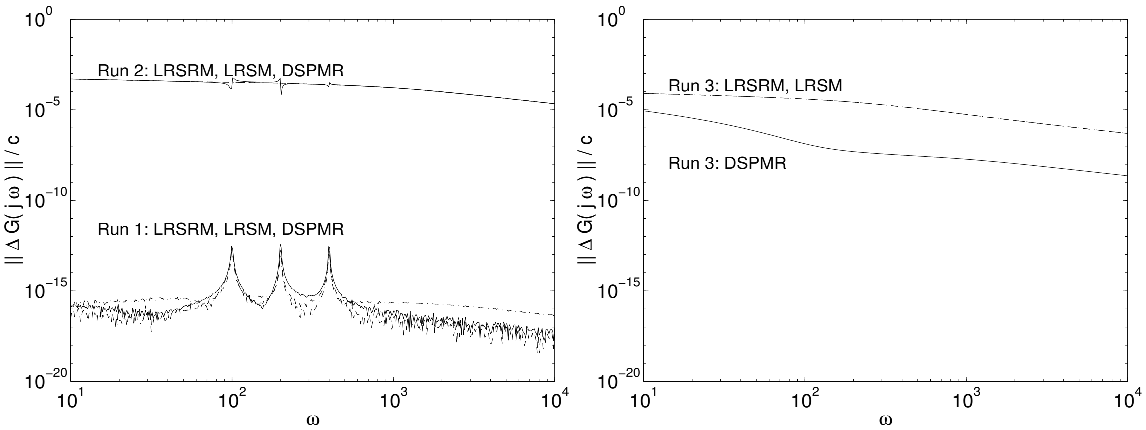

Figure 5: Example 3. Error plots for LRSRM (dashdotted line), LRSM (dashed line), and DSPMR (solid line).

error curves for the algorithms LRSRM and LRSM, which are mathematically equivalent in exact arithmetics, are almost identical. We observed that the condition numbers of the projection matrices S_B and S_C are considerable higher for LRSRM than for LRSM. Moreover, the number of unstable modes in the reduced realization tends to be higher for LRSRM compared to LRSM. However, both aspects seem to have no negative effect on the approximation error of LRSRM. For Examples 1 and 2 the error curves of both methods are slightly better for Run 1 and considerably better for Run 2 compared to those of DSPMR. In Runs 1 and 2 for Example 3 all methods deliver almost identical results. DSPMR performs generally better in Run 3, which can be explained by (17) and (18). Note the superiority of DSPMR in the low-frequency range for Example 1. However, the reduced order of the realizations delivered by DSPMR is larger than those of the LRSRM and LRSM realizations in this run.

6 Conclusions

In this paper we have studied three model reduction algorithms for large dynamical systems. The first two methods, LRSRM and LRSM, are modifications of the well-known square root method and Schur method, which are balanced truncation techniques for systems of moderate order. These modifications are based on a substitution of the controllability and observability Gramians by low rank approximations. DSPMR, the third method, is not directly related to balanced truncation and more heuristic in nature. It is motivated by input and output energy considerations (Theorem 1) and related to the other two methods by certain inclusions that hold for the ranges of the corresponding projection matrices. The availability of relatively accurate low rank approximations to the system Gramians is of vital importance for each model reduction method. We compute these approximations by the LR-Smith(l) iteration, which is a low rank version of the well-known ADI iteration. However, alternative methods could be used.

The performance of the three model reduction algorithms has been studied in numerical experiments. The results of LRSRM and LRSM are fairly similar and mostly better than those for DSPMR. Because of this and its simplicity, LRSRM should be considered as the method of choice in general. On the other hand, in situations when the low rank approximations to the Gramians are not very accurate, DSPMR turns out to be an interesting alternative to LRSRM. Furthermore, DSPMR delivered stable reduced systems in each of our test runs, whereas the reduced systems generated by LRSRM and LRSM often contain a few unstable modes, which must be removed in a postprocessing step.

In our opinion the test results of LRSRM, LRSM, and DSPMR are quite promising in view of the attainable accuracy of the reduced systems and the numerical costs, although we expect that these costs are in many cases higher than those of model reduction methods based on Padé approximation and Krylov subspaces. Nevertheless, our methods can be applied to very large model reduction problems that do not allow the use of standard techniques, in which the Gramians are computed by the Bartels-Stewart method or the Hammarling method.

Acknowledgments

This paper was completed when the author was with the Department of Mathematics and Statistics of the University of Calgary. He would like to thank Prof. Peter Lancaster for his hospitality and Dr. Peter Benner and Dr. Andras Varga for interesting discussions.

References

[1] B. Anderson and S. Vongpanitlerd. *Network Analysis and Synthesis*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

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

[3] P. Benner, J. Claver, and E. Quintana-Orti. Parallel distributed solvers for large stable generalized Lyapunov equations. *Parallel Processing Letters*, 9:147–158, 1999.

[4] P. Benner, E. Quintana-Orti, and G. Quintana-Orti. Balanced truncation model reduction of large-scale dense systems on parallel computers. Submitted for publication.

[5] E. Cuthill. Several strategies for reducing the bandwidth of matrices. In D.J. Rose and R.A. Willoughby, editors, *Sparse Matrices and Their Applications.* Plenum Press, New York, 1972.

[6] E. Davison. A method for simplifying linear dynamic systems. *IEEE Trans. Automat. Control*, 11:93–101, 1966.

[7] D. Enns. Model reduction with balanced realizations: An error bound and a frequency weighted generalization. In *Proc. of the 23rd IEEE Conference on Decision and Control*, pages 127–132, Las Vegas, 1984.

[8] P. Feldmann and R. Freund. Efficient linear circuit analysis by Padé approximation via the Lanczos process. In *Proceedings of EURO-DAC '94 with EURO-VHDL '94*, pages 170–175, Los Alamitos, California, 1994. IEEE Computer Society Press.

[9] P. Feldmann and R. Freund. Efficient linear circuit analysis by Padé approximation via the Lanczos process. *IEEE Trans. Computer-Aided Design*, 14:639–649, 1995.

[10] K.V. Fernando and H. Nicholson. Singular perturbational model reduction of balanced systems. *IEEE Trans. Automat. Control*, 27:466–468, 1982.

[11] R. Freund. Reduced-order modeling techniques based on Krylov subspaces and their use in circuit simulation. Numerical Analysis Manuscript No. 98-3-02, Bell Laboratories, Murray Hill, New Jersey, 1998.

[12] K. Gallivan, E. Grimme, and P. Van Dooren. Asymptotic waveform evaluation via a Lanczos method. *Appl. Math. Lett.*, 7:75–80, 1994.

[13] K. Gallivan, E. Grimme, and P. Van Dooren. A rational Lanczos algorithm for model reduction. *Numer. Alg.*, 12:33–63, 1996.

[14] J.D. Gardiner and A.J. Laub. Parallel algorithms for the algebraic Riccati equations. *Internat. J. Control*, 54:1317–1333, 1991.

[15] G.A. Geist. Reduction of a general matrix to tridiagonal form. *SIAM J. Matrix Anal. Appl.*, 12:362–373, 1991.

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

[17] G.H. Golub and C.F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, 3rd edition, 1996.

[18] W. Gragg. Matrix interpretations and applications of the continued fraction algorithm. *Rocky Mountain J. Math.*, 4:213–225, 1974.

[19] T. Gudmundsson and A.J. Laub. Approximate solution of large sparse Lyapunov equations. *IEEE Trans. Automat. Control*, 39:1110–1114, 1994.

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

[21] A.S. Hodel, K.P. Poolla, and B. Tenison. Numerical solution of the Lyapunov equation by approximate power iteration. *Linear Algebra Appl.*, 236:205–230, 1996.

[22] D.Y. Hu and L. Reichel. Krylov-subspace methods for the Sylvester equation. *Linear Algebra Appl.*, 172:283–313, 1992.

[23] M. Huhtanen. Iterative model reduction of large state-space systems. *Int. J. Appl. Math. Comput. Sci.*, 9:245–263, 1999.

[24] I.M. Jaimoukha. A general minimal residual Krylov subspace method for large scale model reduction. *IEEE Trans. Automat. Control*, 42:1422–1427, 1997.

[25] I.M. Jaimoukha and E.M. Kasenally. Krylov subspace methods for solving large Lyapunov equations. *SIAM J. Numer. Anal.*, 31:227–251, 1994.

[26] I.M. Jaimoukha and E.M. Kasenally. Oblique projection methods for large scale model reduction. *SIAM J. Matrix Anal. Appl.*, 16:602–627, 1995.

[27] I.M. Jaimoukha and E.M. Kasenally. Implicitly restarted Krylov subspace methods for stable partial realizations. *SIAM J. Matrix Anal. Appl.*, 18:633–652, 1997.

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

[29] E. Kloppenburg and E. D. Gilles. Automatic control of the simulated moving bed process for C_8 aromatics separation using asymptotically exact input/output-linearization. *Journal of Process Control*, 9:41–50, 1999.

[30] P. Lancaster. Explicit solutions of linear matrix equations. *SIAM Rev.*, 12:544–566, 1970.

[31] P. Lancaster and L. Rodman. *Algebraic Riccati Equations.* Clarendon Press, Oxford, 1995.

[32] J. LaSalle and S. Lefschetz. *Stability of Liapunov's Direct Method.* Academic Press, New York, NY, 1961.

[33] V. Mehrmann. *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution*, volume 163 of *Lecture Notes in Control and Information Sciences.* Springer-Verlag, Heidelberg, 1991.

[34] B. C. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Trans. Automat. Control*, 26:17–31, 1981.

[35] C. Mullis and R. Roberts. Roundoff noise in digital filters: frequency transformations and invariants. *IEEE Trans. Acoustics, Speech, and Sign. Proc.*, 24:538–550, 1976.

[36] D. Peaceman and H. Rachford. The numerical solution of elliptic and parabolic differential equations. *J. Soc. Indust. Appl. Math.*, 3:28–41, 1955.

[37] T. Penzl. A cyclic low rank Smith method for large sparse Lyapunov equations. To appear in SIAM J. Sci. Comput.

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

[39] L. Pillage and R. Rohrer. Asymptotic waveform evaluation for timing analysis. *IEEE Trans. Computer-Aided Design*, 9:352–366, 1990.

[40] 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.

[41] Y. Saad. Numerical solution of large Lyapunov equations. In M.A. Kaashoek, J.H. Van Schuppen, and A.C.M. Ran, editors, *Signal Processing, Scattering, Operator Theory and Numerical Methods*, pages 503–511. Birkhäuser, Boston, MA, 1990.

[42] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, 1996.

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

[44] H.R. Schwarz. Tridiagonalization of a symmetric band matrix. *Numer. Math.*, 12:231–241, 1968.

[45] Y. Shamash. Model reduction using the Routh stability criterion and the Padé approximation technique. *Internat. J. Control*, 21:475–484, 1975.

[46] V. Sima. *Algorithms for Linear-Quadratic Optimization.* Pure and Applied Mathematics. Marcel Dekker, New York, NY, 1996.

[47] S. Skogestad and I. Postlethwaite. *Multivariable feedback control.* John Wiley and Sons Inc., New York, 1996.

[48] R.A. Smith. Matrix equation $XA + BX = C$. *SIAM J. Appl. Math.*, 16, 1968.

[49] M.S. Tombs and I. Postlethwaite. Truncated balanced realization of stable, non-minimal state-space systems. *Internat. J. Control*, 46:1319–1330, 1987.

[50] F. Tröltzsch and A. Unger. Fast solution of optimal control problems in the selective cooling of steel. Preprint SFB99-7, Technische Universität Chemnitz, Chemnitz, Germany, 1999.

[51] A. Varga. Efficient minimal realization procedure based on balancing. In A. El Moudni, P. Borne, and S. Tzafestas, editors, *Prepr. of IMACS Symp. on Modelling and Control of Technological Systems*, volume 2, pages 42–49, 1991.

[52] E.L. Wachspress. Iterative solution of the Lyapunov matrix equation. *Appl. Math. Lett.*, 1:87–90, 1988.