Parallel Algorithms for Balanced Truncation Model Reduction of Sparse Systems*

José M. Badía[1], Peter Benner[2], Rafael Mayo[1], and Enrique S. Quintana-Ortí[1]

[1] Depto. de Ingeniería y Ciencia de Computadores, Universidad Jaume I
12.071–Castellón, Spain
{badia,mayo,quintana}@icc.uji.es

[2] Fakultät für Mathematik, Technische Universität Chemnitz
D-09107 Chemnitz, Germany
benner@mathematik.tu-chemnitz.de

Abstract. We describe the parallelization of an efficient algorithm for balanced truncation that allows to reduce models with state-space dimension up to $\mathcal{O}(10^5)$. The major computational task in this approach is the solution of two large-scale sparse Lyapunov equations, performed via a coupled LR-ADI iteration with (super-)linear convergence. Experimental results on a cluster of Intel Xeon processors illustrate the efficacy of our parallel model reduction algorithm.

1 Introduction

Consider the continuous linear time-invariant (LTI) dynamical system in state-space form:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \quad t > 0, \quad x(0) = x^0, \\ y(t) &= Cx(t) + Du(t), \quad t \geq 0, \end{aligned} \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$, and $x^0 \in \mathbb{R}^n$ is the initial state of the system. Also, the number of states, n, is known as the state-space dimension (or the order) of the system, and the associated transfer function matrix (TFM) is defined by $G(s) = C(sI - A)^{-1}B + D$. Hereafter, we assume that the spectrum of the state matrix, A, is contained in the open left half plane, implying that the system is stable.

The model reduction problem requires finding a reduced-order LTI system,

$$\begin{aligned} \dot{\hat{x}}(t) &= \hat{A}\hat{x}(t) + \hat{B}\hat{u}(t), \quad t > 0, \quad \hat{x}(0) = \hat{x}^0, \\ \hat{y}(t) &= \hat{C}\hat{x}(t) + \hat{D}\hat{u}(t), \quad t \geq 0, \end{aligned} \tag{1.2}$$

of order r, $r \ll n$, and associated TFM $\hat{G}(s) = \hat{C}(sI - \hat{A})^{-1}\hat{B} + \hat{D}$ which approximates $G(s)$. The reduced-order model can then replace the original high-order system in subsequent calculations including, e.g., the design of controllers [19], thus saving valuable hardware resources.

* José M. Badía, Rafael Mayo, and E.S. Quintana-Ortí were supported by the CICYT project No. TIC2002-004400-C03-01 and FEDER, and project No. P1B-2004-6 of the *Fundación Caixa-Castellón/Bancaixa and UJI*. P. Benner was supported by the DFG Research Center "Mathematics for key technologies" (FZT 86) in Berlin.

Model reduction of moderately large linear systems (n in the hundreds) arises, e.g., in control of large flexible mechanical structures or large power systems [5,7,8,11,20]. Problems of such dimension can be reduced using libraries such as SLICOT[3] or the MATLAB control-related toolboxes on current desktop computers. Larger problems can still be reduced by using the methods in PLiCMR[4] on parallel computers [3,4]. Nevertheless, the applicability of these methods is ultimately limited as they do not exploit the usual sparse structure of the state matrix A and thus present a computational cost of $\mathcal{O}(n^3)$ floating-point arithmetic operations (flops) and require storage for $\mathcal{O}(n^2)$ real numbers.

Therefore, a different approach is necessary for very large sparse systems, with state-space dimension as high as $\mathcal{O}(10^5)$–$\mathcal{O}(10^6)$, such as those arising, among others, in weather forecast, circuit simulation and VLSI design, as well as air quality simulation (see, e.g., [2,6,13,14]). The model reduction method considered in this paper is based on the so-called state-space truncation and requires, at a first stage, the solution of two large sparse Lyapunov equations whose coefficient matrix is the state matrix of the system. A low-rank iteration [17,22] is employed for this purpose which only involves sparse numerical computations such as the solution of linear systems and matrix-vector products. The reduced-order system is then obtained using a slightly modified version of the balanced truncation (BT) method [4,21], and only requires dense linear algebra operations on much smaller data matrices.

Although there exist several other approaches for model reduction (see [2,12] and the references therein), those are specialized for certain problem classes and often lack properties such as error bounds or preservation of stability, passivity, or phase information. A complete comparison between the numerical properties of SVD-based methods (as BT) and Krylov subspace methods can be found in [2].

The paper is structured as follows. In Section 2 we briefly review the method for model reduction of sparse linear systems, based on the BT method and low-rank solvers for the Lyapunov equations arising in state-space truncation. In Section 3 we describe a few parallelization and implementation details of our model reduction algorithm. Finally, the efficacy of the algorithm is reported in Section 4, and some concluding remarks follow in Section 5.

2 Model Reduction of Sparse Linear Systems

2.1 The Square-Root BT Method

BT model reduction [18,23,24,25] belongs to the family of absolute error methods which aim at minimizing $\|\Delta_a\|_\infty = \|G - \hat{G}\|_\infty$. Here $\|G\|_\infty$ denotes the $\mathcal{L}_\infty$- or $\mathcal{H}_\infty$-norm of a stable, rational matrix function defined as

$$\|G\|_\infty = \sup_{\omega\in\mathbb{R}} \sigma_{\max}(G(\jmath\omega)), \tag{2.3}$$

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

[3] Available from `http://www.slicot.net`

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

BT methods are strongly related to the controllability Gramian W_c and the observability Gramian W_o of the system 1.1, given by the solutions of the two dual Lyapunov matrix equations:

$$AW_c + W_c A^T + BB^T = 0, \quad A^T W_o + W_o A + C^T C = 0. \tag{2.4}$$

As the state matrix A is assumed to be stable, W_c and W_o are positive semidefinite and therefore can be factored as $W_c = S^T S$ and $W_o = R^T R$, where S and R are the corresponding *Cholesky factors*.

Efficient Lyapunov solvers that exploit the sparsity of the coefficient matrix A and provide *full-rank factors* $\hat{S}$ and $\hat{R}$ that can replace S and R in subsequent computations are described in the next subsection.

Consider now the singular value decomposition (SVD) of the product

$$SR^T = U\Sigma V^T = [\,U_1\; U_2\,] \begin{bmatrix} \Sigma_1 & \\ & \Sigma_2 \end{bmatrix} [\,V_1\; V_2\,]^T, \tag{2.5}$$

where U and V are orthogonal matrices, and $\Sigma = \mathrm{diag}\,(\sigma_1, \sigma_2, \ldots, \sigma_n)$ is a diagonal matrix containing the singular values $\sigma_1, \ldots, \sigma_n$ of SR^T, which are called the *Hankel singular values* of the system.

Both common versions of BT, the *square-root* (SR) and *balancing-free square-root* (BFSR) BT algorithms, allow an adaptive choice of the state-space dimension r of the reduced-order model as they provide a realization $\hat{G}$ which satisfies

$$\|\Delta_a\|_\infty = \|G - \hat{G}\|_\infty \le 2 \sum_{j=r+1}^{n} \sigma_j. \tag{2.6}$$

Besides, if $\sigma_r > \sigma_{r+1} = 0$, then r is the state-space dimension of a minimal realization of the system.

SR BT methods determine the reduced-order model of order r as

$$\hat{A} = T_l A T_r, \quad \hat{B} = T_l B, \quad \hat{C} = C T_r, \quad \hat{D} = D, \tag{2.7}$$

with the projection matrices T_l and T_r given by

$$T_l = \Sigma_1^{-1/2} V_1^T R \quad \text{and} \quad T_r = S^T U_1 \Sigma_1^{-1/2}. \tag{2.8}$$

BFSR BT algorithms often provide more accurate reduced-order models in the presence of rounding errors.These algorithms obtain T_l and T_r from the following two QR factorizations,

$$S^T U_1 = [P_1\; P_2] \begin{bmatrix} R_s \\ 0 \end{bmatrix}, \qquad R^T V_1 = [Q_1\; Q_2] \begin{bmatrix} R_r \\ 0 \end{bmatrix}, \tag{2.9}$$

where the columns of P_1, $Q_1 \in \mathbb{R}^{n\times r}$ form orthonormal bases for $\mathrm{range}(T_r)$, $\mathrm{range}(T_l)$, respectively, and R_s, $R_r \in \mathbb{R}^{r\times r}$ are upper triangular. The reduced system is then given by 2.7 and the projection matrices

$$T_l = (Q_1^T P_1)^{-1} Q_1^T, \qquad T_r = P_1. \tag{2.10}$$

2.2 Low Rank Solution of Lyapunov Equations

In this subsection we revisit the Lyapunov solvers introduced in [17,22]. These iterative algorithms benefit from the frequently encountered low-rank property of the constant terms in 2.4 to provide low-rank approximations to a Cholesky or full-rank factor of the solution matrix. These approximations can reliably substitute S and R in the computation of 2.5, 2.8, and 2.9.

Specifically, given an "l–cyclic" set of (complex) shift parameters $\{p_1, p_2, \ldots\}$, $p_k = \alpha_k + \beta_k\, \jmath$, such that $p_k = p_{k+l}$, the cyclic *low-rank alternating direction implicit* (LR-ADI) iteration proposed in [22] can be reformulated as follows:

$$\begin{array}{ll} V_0 = (A + p_1 I_n)^{-1} B, & \hat{S}_0 = \sqrt{-2\,\alpha_1}\, V_0, \\ V_{k+1} = V_k - \delta_k (A + p_{k+1} I_n)^{-1} V_k, & \hat{S}_{k+1} = \left[\hat{S}_k\ ,\ \gamma_k V_{k+1} \right], \end{array} \tag{2.11}$$

where $\gamma_k = \sqrt{\alpha_{k+1}/\alpha_k}$, $\delta_k = p_{k+1} + \overline{p_k}$, $\overline{p_k}$ being the conjugate of p_k, and I_n denotes the identity matrix of order n. On convergence, after $k_{\max}$ iterations, a low-rank matrix $\hat{S}_k$ of order $n \times k_{\max} m$ is computed such that $\hat{S}_k \hat{S}_k^T$ approximates $W_c = S^T S$. An analogous iteration involving A^T provides a low-rank approximation $\hat{R}_k$ of R.

The performance of iteration 2.11 strongly depends on the selection of the shift parameters. In practice, the set $\{p_1, p_2, \ldots, p_l\}$ should be chosen so that it is closed under complex conjugation. In case the eigenvalues of A are real, the optimal parameters can be computed explicitly [27]. Otherwise, an optimal solution is not known. A heuristic procedure is proposed in [22] to compute the parameters by approximating the eigenvalues of A and A^{-1} of largest magnitude. The procedure is based on an Arnoldi iteration involving A and A^{-1}. For further details on the convergence of the LR-ADI iteration and the properties of the heuristic selection procedure, see [22].

It should be emphasized that the methods just described for solving 2.4 and 2.5 significantly differ from standard methods used in the MATLAB toolboxes or SLICOT [26]. First, the proposed LR-ADI iteration for the solution of the dual Lyapunov equation exploits the sparsity in the coefficient matrix. Besides, as we are using low-rank approximations to the full-rank or Cholesky factors, the computation of the SVD in 2.5 is usually much more efficient: instead of a computational cost of $\mathcal{O}(n^3)$ when using the Cholesky factors, this approach leads to an $\mathcal{O}(k_{\max}^2 \cdot m \cdot p \cdot n)$ cost where, in model reduction, often $m, p \ll n$; see [4]. This latter advantage is shared by the routines in our dense parallel model reduction library PLiCMR [4]. However, PLiCMR routines do not exploit the sparsity of the coefficient matrix.

3 Parallel Implementation

3.1 Implementation Details

Iteration 2.11 can be stopped when the contribution of the columns added to $\hat{S}_{k+1}$ is negligible; thus, in practice, we stop the iteration when $||\gamma_k V_{k+1}||_1$ is "small".

The use of direct linear system solvers [10] (based, e.g., on the LU or Cholesky factorization) is appealing as the same coefficient matrix is involved in iterations k

and $k + l$. Therefore, the computed factorization can be re-used several times provided sufficient workspace is available to store the factors. Also, many direct sparse solvers that include an initial phase of symbolic factorization only need to perform this phase once, as all linear systems share the same sparsity pattern.

Notice that even in case all data matrices are real, the LR-ADI iteration will involve complex arithmetic in case any of the shifts is complex. Special care must be taken so that all shifts are real in case all eigenvalues of A are real as that reduces the computational cost of the iteration significantly.

3.2 Parallelization

The LR-ADI iteration and the computation of the acceleration shifts basically require linear algebra operations such as matrix-vector products (in the Arnoldi procedure for computation of the shifts for the iteration) and the solution of linear systems (also in the computation of the shifts, and in the LR-ADI iteration). Our approach for dealing with these matrix operations is based on the use of parallel linear algebra libraries. (For an extensive list of those, visit

`http://www.netlib.org/utk/people/JackDongarra/la-sw.html`.)

Specifically, in case sparse matrices are involved, the matrix-vector products are computed as a series of "usaxpy" ($y = y + \alpha \cdot x$) operations, with each process in the parallel system performing, approximately, an equal amount of these operations. Although beneficial, no attempt to balance the computational load is done here as the contribution of the matrix-vector products to the cost of the model reduction algorithm is a minor one: in practice, the number of matrix-vector products is proportional to the number of different shifts, and much smaller than the number of linear systems that need to be solved (one per iteration). Besides, the cost of a matrix-vector product is in general much lower than that of solving a linear system.

Sparse linear systems can be solved in our parallel codes by using appropriate kernels from MUMPS or SuperLU [1,9]. In both cases, the analysis phase (corresponding to a symbolic factorization) is performed only once for the iteration, and the factorization phase is only performed during the first l iterations.

In case the coefficient matrix is banded we use the linear system solvers in ScaLAPACK. We also implemented our own parallel routine for the banded matrix-vector product as this is not available in the library.

Our parallel codes can also deal with dense linear systems using the kernels in ScaLAPACK. However, in such case the methods in PLiCMR (see [4] for an overview) are preferred as the Lyapunov equation solvers used there have quadratic convergence rate as opposed to the (super-)linear convergence of the ADI method at a comparable cost of the remaining computational stages.

After the LR-ADI iteration is completed, the model reduction algorithm proceeds to compute the product of $\hat{S}_k \hat{R}_k^T$ as in 2.5 and the reduced-order model using 2.8 or 2.9–2.10. As both $\hat{S}_k$ and $\hat{R}_k$ are full dense factors, their product and the SVD are computed using the kernels in ScaLAPACK. The SVD of a matrix with complex entries is not available in ScaLAPACK. This difficulty is circumvented by collecting the product

$\hat{S}_k \hat{R}_k^T$ into a single node and computing its SVD using the serial routine from LAPACK. We exploit here that the size of the resulting matrix is order of $k_{\max}m \times k_{\max}p$ which, in general, is much smaller than $n \times n$.

Once the projection matrices T_l and T_r have been computed, it only remains to apply these matrices to form the reduced-order system (T_lAT_r, T_lB, CT_r, D). These matrix products are performed by taking into account the characteristics of matrices A, B, and C which can be sparse, banded, or dense. In the former two cases, the product is computed as a series of matrix-vector products, while in the latter case we rely again on the corresponding ScaLAPACK routine.

4 Experimental Results

All the experiments presented in this section were performed on a cluster of $n_p = 32$ nodes using IEEE double-precision floating-point arithmetic ($\varepsilon \approx 2.2204 \times 10^{-16}$). Each node consists of an Intel Xeon processor at 2.4 GHz with 1 GByte of RAM. We employ a BLAS library specially tuned for this processor that achieves around 3800 Mflops (millions of flops per second) for the matrix product (routine DGEMM) [15]. The nodes are connected via a *Myrinet* multistage network and the MPI communication library is 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 18 μsec. and a bandwidth of 1.4 Gbit/sec.

Our benchmark for model reduction of sparse systems consists of three scalable models and serves to illustrate the benefits gained from applying model reduction via BT combined with parallelism to large-scale systems. This is by no means a comparison of the efficacy of the direct linear system solvers in MUMPS, SuperLU, and ScaLAPACK. In order to avoid it, we report results using a different solver and problem size for each example.

As in all our experiments both SR and BFSR BT algorithms delivered closely similar results, we only report data for the first algorithm.

Example 1. This example models the heat diffusion in a (1-dimensional) thin rod with a single heat source [5]. The system is parameterized by a scalar α that we set to $\alpha = 0.1$ in our experiments. The spatial domain is discretized into segments of length $h = \frac{1}{n+1}$ and centered differences are used to approximate the diffusion operator. A heat point is assumed to be located at 1/3 of the length of the rod and the temperature is recorded at 2/3 of the length.

In this case, a model of state-space dimension $n = 10,000$ was reduced to order $r = 85$ using the sparse direct solvers in SuperLU 2.0 (distr.) and $l = 50$ different shifts for the LR-ADI iteration. Only 16 nodes of the parallel system were used resulting in the following execution times (expressed in minutes and seconds) for each one of the stages of the algorithm:

Comp. shifts	LR-ADI iter.	Comp. SVD+SR	Total
11.74" (11.3%)	1' 28" (86.1%)	2.49" (2.4%)	1' 42"

Thus, the computation of the reduced-order model required slightly more than 1.5 minutes. This is clearly a low temporal cost that we may be willing to pay for the great benefits obtained from using the reduced-order model in the subsequent computations needed for simulation, optimization, or control purposes.

These results also show that the most expensive part of the model reduction procedure was the LR-ADI iteration, which required about 86% of the total time. Notice that, as $l = 50$, this phase requires the computation of 50 matrix (LU) factorizations, while the computation of the shifts only requires one factorization. Convergence was achieved for this example after 128 iterations producing full-rank approximations of the Cholesky factors, $\hat{S}_k$ and $\hat{R}_k$, of order $128 \times n$. Thus, the SVD computation only involved a square matrix of order 128 requiring around 2.5 sec.

The reduced-order system obtained with our method satisfies the absolute error bound $\|G - \hat{G}\|_\infty \approx 1.91 \times 10^{-16}$, showing that it is possible to reduce the order of the system from $n = 10,000$ to 85 without a significative difference between the behavior of the original and the reduced-order models. Actually, the reduced-order model can be seen as a numerically minimal realization of the LTI system.

Example 2. The matrices in this case model the temperature distribution in a 2-D surface. The state matrix of the system presents a block tridiagonal sparse structure of order N^2 and is obtained from a discretization of the Poisson's equation with the 5-point operator on an $N \times N$ mesh. The input and output matrices were constructed as $B = \left[e_1^T,\ 0_{1\times N(N-1)}\right]^T$ (with $e_1 \in \mathbb{R}^N$ the first column of the identity matrix) and $C = \left[e_1^T,\ 1,\ 0_{1\times N-2},\ 1\right]$ (with $e_1 \in \mathbb{R}^{N(N-1)}$). A large system of order $n = 202,500$, with a single input and a single output, was reduced in this case to order $r = 30$. The parallel algorithm employed MUMPS 4.3 as the direct linear system solver, $l = 15$ shifts, and 16 nodes, resulting in the following execution times:

Comp. shifts	LR-ADI iter.	Comp. SVD+SR	Total
9.62" (0.7%)	19' 54" (97.6%)	19.9" (1.6%)	20' 22"

Here, obtaining the reduced-order model required about 20 minutes. This time could be further reduced by employing more nodes of the system. In this example the most expensive part was again the LR-ADI iteration, which required the major part of the execution time of the procedure. Convergence was achieved after 69 iterations, producing a reduced-order model which satisfies $\|G - \hat{G}\|_\infty \approx 3.2 \times 10^{-17}$. From this absolute error bound we again conclude that, for this example, it is possible to reduce the order of the system from $n = 202,500$ to 30 without a significative difference between the behavior of the original and the reduced-order models.

Example 3. The matrices in this example model a RLC circuit of n_0 sections interconnected in cascade resulting in a system with $n = 2n_0$ states and a single input/output [16]. The system is parameterized by scalars $R = 0.1$, $\bar{R} = 1.0$, $C = 0.1$, and $L = 0.1$. The state matrix in this example is tridiagonal.

In this third example we reduced a system of order $n = 200,000$ and a single input and output to order $r = 50$. Using the solver in ScaLAPACK 1.6 for banded linear systems, $l = 15$ shifts for the LR-ADI iteration, and 16 nodes of the parallel system, the model reduction algorithm required the following execution times:

Comp. shifts	LR-ADI iter.	Comp. SVD+SR	Total
2.26" (7.4%)	5.14" (16.9%)	22.87" (75.4%)	30.33"

The results show that scarcely more than half a minute was sufficient to reduce a large scale system of order $n = 200,000$. Convergence of the LR-ADI iteration was achieved for this example after 57 iterations, resulting in a reduced-order system that, for $r = 50$, satisfied $\|G - \hat{G}\|_\infty < 4.9 \times 10^{-23}$.

There are two important differences between the results for Examples 2 and 3. First, although the order of the models are of the same magnitude, the execution times for Example 3 are much lower. This is due to the ability of the banded solvers in ScaLAPACK to exploit the structure of the state matrix for Example 3. Second, the largest part of the computation time lies for Example 3 in the computation of the SVD and the reduced-order system (SVD+SR stage) while, for Example 2, it is spent in the LR-ADI iteration. However, the execution times of the SVD+SR stage are comparable for both examples. Thus, it is really the important reduction of the execution time of the LR-ADI iteration which, for Example 3, produced the shift of the bulk of the computation time to the SVD+SR stage.

5 Concluding Remarks

We have presented parallel algorithms for BT model reduction of large sparse (and dense) linear systems of order as large as $\mathcal{O}(10^5)$. Our model reduction algorithms employ kernels from parallel linear algebra libraries such as SuperLU, MUMPS, and ScaLAPACK. Three large-scale models are used in the experimental results to illustrate the benefits gained from applying model reduction via BT combined with parallel execution and to report the performance of the approach on a cluster of Intel Xeon processors.

The parallelism of our approach depends on the problem structure and the parallelism of the underlying parallel linear algebra library (SuperLU, MUMPS, or the banded codes in ScaLAPACK).

References

1. P.R. Amestoy, I.S. Duff, J. Koster, and J.-Y. L'Excellent. MUMPS: a general purpose distributed memory sparse solver. In *Proc. PARA2000, 5th International Workshop on Applied Parallel Computing*, pages 122–131, 2000.
2. A.C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM Publications, Philadelphia, PA, 2005.
3. 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.
4. P. Benner, E.S. Quintana-Ortí, and G. Quintana-Ortí. State-space truncation methods for parallel model reduction of large-scale systems. *Parallel Comput.*, 29:1701–1722, 2003.
5. 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`.
6. C.-K. Cheng, J. Lillis, S. Lin, and N.H. Chang. *Interconnect Analysis and Synthesis*. John Wiley & Sons, Inc., New York, NY, 2000.
7. J. Cheng, G. Ianculescu, C.S. Kenney, A.J. Laub, and P. M. Papadopoulos. Control-structure interaction for space station solar dynamic power module. *IEEE Control Systems*, pages 4–13, 1992.

8. P.Y. Chu, B. Wie, B. Gretz, and C. Plescia. Approach to large space structure control system design using traditional tools. *AIAA J. Guidance, Control, and Dynamics*, 13:874–880, 1990.
9. J.W. Demmel, J.R. Gilbert, and X.S. Li. *SuperLU User's Guide*.
10. I.S. Duff, A.M. Erisman, and J.K. Reid. *Direct methods for sparse matrices*. Oxford Science Publications, Oxford, UK, 1993.
11. L. Fortuna, G. Nummari, and A. Gallo. *Model Order Reduction Techniques with Applications in Electrical Engineering*. Springer-Verlag, 1992.
12. 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.
13. R. Freund. Model reduction methods based on Krylov subspaces. *Acta Numerica*, 12:267–319, 2003.
14. R. W. Freund and P. Feldmann. Reduced-order modeling of large passive linear circuits by means of the SyPVL algorithm. In *Technical Digest of the 1996 IEEE/ACM International Conference on Computer-Aided Design*, pages 280–287. IEEE Computer Society Press, 1996.
15. K. Goto and R. van de Geijn. On reducing TLB misses in matrix multiplication. FLAME Working Note 9, Department of Computer Sciences, The University of Texas at Austin, `http://www.cs.utexas.edu/users/flame`, 2002.
16. S. Gugercin and A.C. Antoulas. A survey of balancing methods for model reduction. In *Proc. European Control Conf. ECC 03* (CD-ROM), Cambridge, 2003.
17. J.-R. Li and J. White. Low rank solution of Lyapunov equations. *SIAM J. Matrix Anal. Appl.*, 24(1):260–280, 2002.
18. B.C. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Trans. Automat. Control*, AC-26:17–32, 1981.
19. G. Obinata and B.D.O. Anderson. *Model Reduction for Control System Design*. Communications and Control Engineering Series. Springer-Verlag, London, UK, 2001.
20. C.R. Paul. *Analysis of Multiconductor Transmission Lines*. Wiley–Interscience, Singapur, 1994.
21. T. Penzl. Algorithms for model reduction of large dynamical systems. Technical Report SFB393/99-40, Sonderforschungsbereich 393 *Numerische Simulation auf massiv parallelen Rechnern*, TU Chemnitz, 09107 Chemnitz, Germany, 1999. Available from `http://www.tu-chemnitz.de/sfb393/sfb99pr.html`.
22. T. Penzl. A cyclic low rank Smith method for large sparse Lyapunov equations. *SIAM J. Sci. Comput.*, 21(4):1401–1418, 2000.
23. M.G. Safonov and R.Y. Chiang. A Schur method for balanced-truncation model reduction. *IEEE Trans. Automat. Control*, AC–34:729–733, 1989.
24. 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.
25. A. Varga. Efficient minimal realization procedure based on balancing. In *Prepr. of the IMACS Symp. on Modelling and Control of Technological Systems*, volume 2, pages 42–47, 1991.
26. A. Varga. 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.
27. E.L. Wachspress. ADI iteration parameters for the Sylvester equation, 2000. Available from the author.