Accelerating BST Methods for Model Reduction with Graphics Processors

Peter Benner[1], Pablo Ezzatti[2], Enrique S. Quintana-Ortí[3], and Alfredo Remón[3]

[1] Max Planck Institute for Dynamics of Complex Technical Systems (Magdeburg, Germany)
benner@mpi-magdeburg.mpg.de
[2] Centro de Cálculo-Instituto de Computación, Universidad de la República (Montevideo, Uruguay)
pezzatti@fing.edu.uy
[3] Depto. de Ingeniería y Ciencia de Computadores, Universidad Jaume I (Castellón, Spain)
{quintana,remon}@icc.uji.es

Abstract Model order reduction of dynamical linear time-invariant system appears in many scientific and engineering applications. Numerically reliable SVD-based methods for this task require $\mathcal{O}(n^3)$ floating-point arithmetic operations, with n being in the range $10^3 - 10^5$ for many practical applications. In this paper we investigate the use of graphics processors (GPUs) to accelerate model reduction of large-scale linear systems via Balanced Stochastic Truncation, by off-loading the computationally intensive tasks to this device. Experiments on a hybrid platform consisting of state-of-the-art general-purpose multi-core processors and a GPU illustrate the potential of this approach.

Key words: Model reduction, linear dynamical systems, Lyapunov equations, SVD-based methods, GPUs.

1 Introduction

Model order reduction is an important numerical tool to reduce the time and cost required for the design of optimal controllers in many industrial processes where dynamics can be modeled by a linear time-invariant (LTI) system of the form:

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

Here, $x(t)$ contains the states of the system, with $x^0 \in \mathbb{R}^n$ the initial state, $u(t) \in \mathbb{R}^m$ and $y(t) \in \mathbb{R}^p$ contain the inputs and outputs, respectively, and $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$. The system in (1) can also be described by the associated transfer function matrix (TFM) $G(s) = C(sI_n - A)^{-1}B + D$. A particularly important property is that the number of states (also known as the

state-space dimension or the order) of the system, n, is in general much larger than m and p.

The goal of model reduction is to find a reduced-order LTI system,

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

of order r, with $r \ll n$, and associated TFM $\hat{G}(s) = \hat{C}(sI_n - \hat{A})^{-1}\hat{B} + \hat{D}$ which approximates the dynamics of the original system defined by $G(s)$. The reduced-order realization (2) can then replace the original model of order n in subsequent simulations or processes, thus simplifying these tasks considerably. Model order reduction of large-scale systems appears, e.g., in thermal, thermo-mechanical, electro-mechanical and acoustic finite element models [1]. We consider a system to be large-scale if $n \sim \mathcal{O}(1,000) - \mathcal{O}(100,000)$, while, often, $m, p \sim \mathcal{O}(10) - \mathcal{O}(100)$.

The numerical method for model order reduction considered in this paper is based on the so-called state-space truncation approach and requires, at an initial stage, the solution of a Lyapunov and a Riccati equation. The reduced-order system is then obtained using a variant of the balanced stochastic truncation (BST) method [2], which only requires dense linear algebra computations. Although there exist several other approaches for model order reduction (see, e.g., [1,3] and the references therein), those are specific for a certain subset of problems and often do not possess relevant properties such as error bounds, preservation of stability and passivity, or phase information. A comparison of the numerical properties of SVD-based methods (as Balanced Stochastic Truncation, BST) and Krylov subspace methods can be found in [1].

The Lyapunov and Riccati equations are solved in our algorithms via the matrix sign function, which yields a computational cost for the global model order reduction procedure of $\mathcal{O}(n^3)$ flops (floating-point arithmetic operations). This calls for the application of high performance computing in the reduction of models with n in the order of thousands or larger.

Recent work on the implementation of the BLAS specification and some relevant linear algebra operations included in LAPACK [4,5,6,7] has demonstrated the potential of graphics processors (GPUs) to yield high performance for the execution of dense linear algebra operations, specially if they can be cast in terms of matrix-matrix products. In [8] we built upon these works to deal with the solution of the standard Lyapunov equation on a GPU. Here, we extend this work by tackling the different stages in BST methods for model reduction of linear systems, namely, the solution of the Lyapunov and Riccati equations, the computation of the SVD, and other auxiliary computations. The target architecture is a hybrid platform consisting of a general-purpose multicore processor and a GPU. We exploit these two resources by designing a hybrid numerical algorithm for model order reduction that performs fine-grain computations on the CPU while off-loading computationally intensive operations to the GPU.

The rest of the paper is structured as follows. In Section 2 we briefly review the BST method for model order reduction, including the Lyapunov solver, the

sign function-based Riccati solver and the remaining stages of the method. In Section 3 high performance implementations for a hybrid CPU-GPU platform are described. In Section 4 we present experimental results that expose the parallelism attained by the numerical algorithms on a platform consisting of two Intel QuadCore processors connected to an NVIDIA Tesla C1060 GPU. Finally, in Section 5 we provide a few concluding remarks and future lines of work.

2 Model Reduction methods based on SVD

Relative error methods attempt to minimize the relative error $\|\Delta_r\|_\infty$, defined implicitly by $G - \hat{G} = G\Delta_r$. Among these, BST and its variants are particularly popular [9,10,11]. Due to their high computational cost, BST methods have been used only for problems of moderate dimension, i.e., models of state-space dimension in the order of hundreds. The implementation included in the *Subroutine Library in Control Theory* – SLICOT[1] [12], available for MATLAB® and Fortran 77, made feasible to target systems with a few thousands of state-space variables on nowadays standard desktop computers, but larger problems remain un-affordable, unless a cluster of computers and a message-passing library as PLiCMR [13] is employed.

BST is a technique where the reduced order model is obtained truncating a balanced stochastic realization. Such a realization is obtained as follows. Define $\Phi(s) = G(s)G^T(-s)$, and let W be a *square minimum phase right spectral factor* of Φ, i.e., $\Phi(s) = W^T(-s)W(s)$. As D has full row rank, $E = DD^T$ is positive definite and a minimal state-space realization (A_W, B_W, C_W, D_W) of W is given by (see [14,15])

$$\begin{array}{ll} A_W = A, & B_W = BD^T + W_c C^T, \\ C_W = E^{-\frac{1}{2}}(C - B_W^T X_W), & D_W = E^{\frac{1}{2}}. \end{array} \tag{3}$$

Here, W_c is the controllability Gramian of $G(s)$ given by the solution of the Lyapunov equation

$$AW_c + W_c A^T + BB^T \ = \ 0 \tag{4}$$

while W_o is the observability Gramian of $W(s)$ obtained as the *stabilizing* solution of the algebraic Riccati equation (ARE)

$$\begin{array}{l} 0 = (A - B_W E^{-1} C)^T W_o + W_o(A - B_W E^{-1} C) + W_o B_W E^{-1} B_W^T W_o + \\ \quad C^T E^{-1} C. \end{array} \tag{5}$$

In the following subsections we revisit the sign function methods for the solution of Lyapunov and Riccati equations introduced in [16] and [17] respectively. For the solution of the Lyapunov equation, the algorithm introduced in [8] has demonstrated to be highly efficient on hybrid architectures equipped with a GPU. This Lyapunov solver provides a low-rank approximation to the full-rank factor of the solution matrix.

[1] Available from `http://www.slicot.org`.

2.1 Solution of the Lyapunov equation

The matrix sign function was introduced in [17] as an efficient tool to solve stable (standard) Lyapunov equations. The variant of the Newton iteration method for the matrix sign function in Algorithm `CECLNC` [16] can be employed for the solution of a Lyapunov equation (like that in (4)).

Algorithm `CECLNC`:

$A_0 \leftarrow A$, $\tilde{S}_0 \leftarrow B^T$
$k \leftarrow 0$
`repeat`
  Compute the rank-revealing QR (RRQR) decomposition
    $\frac{1}{\sqrt{2c_k}} \left[\tilde{S}_k, \;\; c_k \tilde{S}_k A_k^{-T} \right] = Q_s \begin{bmatrix} U_s \\ 0 \end{bmatrix} \Pi_s$
  $\tilde{S}_{k+1} \leftarrow U_s \Pi_s$
  $A_{k+1} \leftarrow \frac{1}{\sqrt{2}} \left(A_k/c_k + c_k A_k^{-1} \right)$
  $k \leftarrow k + 1$
`until` $\|A_k - I\|_1 < \tau_l \|A_k\|_1$

The number of columns of factor $\tilde{S}$ is doubled at each iteration and, in consequence, the computational and storage costs associated with its update grow at each iteration. To moderate this increase, and the number of columns in the factors, an RRQR factorization is computed at each step. This approach yields important gains when the number of iterations that are required for convergence is large. Note that Q_s is not accumulated as it is not needed in further computations. This reduces the cost of the RRQR significantly. For simplicity, we do not detail this compression procedure; see [18].

On convergence, after j iterations, $\tilde{S} = \frac{1}{\sqrt{2}} \tilde{S}_j$, of dimension $\tilde{k}_c \times n$, is the full (row-)rank approximation of S, so that $W_c = S^T S \approx \tilde{S}^T \tilde{S}$.

The Newton iteration for the sign function method usually presents a fast convergence rate, which is ultimately quadratic. Initial convergence can be accelerated using several techniques. In our case, we employ a scaling factor defined by the parameter

$$c_k = \sqrt{\|A_k\| / \|A_k^{-1}\|}.$$

In the convergence test, τ_l is a tolerance threshold for the iteration that is usually set as a function of the problem dimension and the machine precision ε. In particular, to avoid stagnation in the iteration, we set $\tau_l = n \cdot \sqrt{\varepsilon}$ and perform one or two additional iteration steps after the stopping criterion is satisfied. Due to the quadratic convergence of the Newton iteration, this is usually enough to reach the attainable accuracy. The RRQR decomposition can be obtained by means of the traditional QR factorization with column pivoting [18] plus a reliable rank estimator.

2.2 Solution of the Riccati equation

The solution of an Algebraic Riccati Equation (ARE) of the form

$$F^T X + XF - XGX + Q \;=\; 0, \tag{6}$$

can be obtained from the stable invariant subspace of the Hamiltonian matrix defined in [19]

$$H = \begin{bmatrix} F & G \\ -Q & -F^T \end{bmatrix}. \tag{7}$$

This solution can be obtained computing the matrix sign function of H [17]:

$$\mathrm{sign}(H) = Y = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}, \tag{8}$$

and then, solving the overdetermined system

$$\begin{bmatrix} Y_{11} \\ Y_{12} + I_n \end{bmatrix} X = \begin{bmatrix} I_n - Y_{21} \\ -Y_{11} \end{bmatrix} \tag{9}$$

(e.g., applying the least squares method).

Algorithm `GECRSG` summarizes the steps to solve an ARE with this method.

Algorithm `GECRSG`:

$H_0 \leftarrow \begin{bmatrix} F & G \\ -Q & -F^T \end{bmatrix}$
$k \leftarrow 0$
`repeat`
  $H_{k+1} \leftarrow \frac{1}{2}\left(H_k/d_k + d_k H_k^{-1}\right)$
`until` $\|H_{k+1} - H_k\|_1 < \tau_r \|H_k\|_1$
Solve

$$\begin{bmatrix} Y_{11} \\ Y_{12} + I_n \end{bmatrix} X = \begin{bmatrix} I_n - Y_{21} \\ -Y_{11} \end{bmatrix}$$

The scaling factor d_k and the tolerance threshold τ_r can be defined here following the ideas presented for c_k and τ_l, respectively, in the previous subsection.

3 High performance implementation on a hybrid architecture

In this section, we describe an efficient implementation of the BST model order reduction method, specially designed for hybrid platforms composed of a

multicore CPU connected to a single GPU. The objective of the hybrid implementation is to reduce the computational time of the BST method, executing each operation on the most convenient architecture and reducing the amount of data transfers between the two components. Specifically, the most expensive computations are executed on the GPU while the fine-grain operations are executed on the CPU.

3.1 Hybrid implementation of the Lyapunov solver

The most time consuming operation of Algorithm `CECLNC` is the update of A_{k+1}. This is due to the large dimension of A and the significant computational cost of the matrix inversion.

The hybrid algorithm to accelerate the solution of this equation proceeds as follows. At the beginning of each iteration, the CPU transfers matrix A_k to the GPU. Then, the CPU and the GPU cooperate in the inversion of matrix A_k, which is returned to the CPU upon completion. The rest of the operations are performed on the CPU since they require a minor computational effort and can be efficiently executed on a multicore processor, e.g. invoking parallel implementations of `BLAS` and `LAPACK` to compute the linear algebra operations or using a simple OpenMP-based parallelization in other cases.

The inversion algorithm is based on the Gauss-Jordan elimination method [20], since this is a highly parallel algorithm. The implementation includes some performance-enhancing techniques, as the processing by blocks to exploit the hierarchical organization of the memory, hybrid and concurrent computing (CPU + GPU) to increment the resource utilization, look-ahead techniques [21] to avoid bottlenecks, padding to accelerate the memory accesses on the GPU, and multilevel blocks strategies to improve throughput of both devices.

3.2 Hybrid implementation of the Riccati solver

This stage can be divided into three steps:

- First, it is necessary to build matrix H performing some matrix-matrix multiplications (see equations (3)-(5)). As the dimensions of the matrices involved in these operations are moderate, the related computational cost is moderate as well. For this reason, and with the aim to reduce data transfers overheads, those operations are performed on the CPU.
- Second, the sign function for the extended matrix (7) is computed. The proposal is based on the efficient matrix inversion kernel described in subsection 3.1 and the utilization of OpenMP to accelerate the matrix addition and matrix norm evaluation performed on the CPU; see Algorithm `GECRSG`. Note that the solution of the Riccati equation via this solver involves matrices which are twice as big as those that appear in the sign function solver for the Lyapunov equation.
- Finally, the overdetermined system is solved. To do so, a multi-thread version of routine `GEQP3` (included in `LAPACK`) is employed. Other minor operations are also executed on the CPU and parallelized using OpenMP.

3.3 Remaining stages of the BST method

Once the low rank factor from the controllability gramian (S) and the observability gramian (W_o) have been computed from the solution of the Lyapunov and the Riccati equations respectively, only some minor operations with moderate computational effort are required to obtain the reduced order model.

The main computations in this step include some matrix-matrix products involving matrices of relatively small dimension. All these operations require a moderate number of arithmetic operations and, therefore, can be efficiently computed on the CPU using `BLAS`. Computing them on the CPU avoids data transfers and the associated overhead.

4 Numerical Experiments

In this section we evaluate the parallel performance of the BST model order reduction method. The target platform consists of two INTEL Xeon QuadCore E5410 processors (2.33GHz) with 8GB of RAM connected to an NVIDIA Tesla C1060 via a PCI-e bus. Multi-threaded implementation of BLAS, from the INTEL MKL library (version 10.2) for the general-purpose processor and from NVIDIA CUBLAS (version 3.2) for the GPU are used.

We compare three different implementations: a sequential one (BST_SCPU) that is executed on a single CPU core (used as the reference implementation), a parallel multi-thread routine (BST_MTCPU) that exploits all the cores from the CPU, and a hybrid CPU-GPU implementation (BST_HYB) that executes operations concurrently on the GPU and the CPU cores.

We employ double precision arithmetic for the solution of two instances of the `STEEL_I` model reduction problem [22], extracted from the Oberwolfach benchmark collection (University of Freiburg)[2]. This model arises in the manufacturing process of steel profiles. The goal is to design a control that yields moderate temperature gradients when the rail is cooled down. The mathematical model corresponds to the boundary control for a 2-D heat equation. A finite element discretization, followed by adaptive refinement of the mesh, results in the example in this benchmark. The problem dimensions depend of the discretization mesh; the two versions employed are $\texttt{STEEL\_I}_{1357}$ with $n = 1,357$, $m = 7$, $p = 6$; and $\texttt{STEEL\_I}_{5177}$ with $n = 5,177$, $m = 7$, $p = 6$.

Table 1 summarizes the results (in seconds) obtained with all the implementations evaluated. The execution time dedicated to solve the Lyapunov equation is shown in column 2; columns 3, 4 and 5 report the time required to initialize matrix H, compute sign(H) and solve the overdetermined system, respectively; column 6 displays the accumulated time. All the times given in Table 1 include the costs to perform all the necessary CPU-GPU data transfers.

Note that most of the time is dedicated to compute the solution of the Riccati equation, in particular the computation of sign(H) (column 4). The rest of the time is basically spent in the Lyapunov equation solver. A careful study of these

[2] `http://www.imtek.de/simulation/benchmark/`.

Implementation	Lyapunov solver	H init.	sign(H)	System solver	Total time(s)
STEEL_I$_{1357}$					
BST_SCPU	7.74	0.10	118.15	3.23	129.22
BST_MTCPU	1.68	0.05	22.34	0.57	24.64
BST_HYB	9.46	0.05	10.93	0.57	21.01
STEEL_I$_{5177}$					
BST_SCPU	334.16	1.52	6404.65	325.34	7065.67
BST_MTCPU	63.75	0.86	1127.87	25.05	1217.53
BST_HYB	26.82	0.78	292.93	24.92	345.48

Table 1. Execution time (in secs.) of the different steps of the BST method for the STEEL_I problem.

two operations demonstrates that the computational effort is concentrated in the calculation of matrix inverses. This operation is accelerated in the BST_MTCPU implementation using multi-thread codes. The BST_HYB variant improves the parallelization of the matrix inversion procedure using the Gauss-Jordan elimination method, which is more suitable for its execution on parallel architectures, and off-loading part of the computations to the GPU.

The times reported for the STEEL_I$_{1357}$ instance show a notable benefit from the usage of the multicore (BST_MTCPU) and the hybrid (BST_HYB) implementation, which are respectively 5 and 6 times faster than the sequential implementation. From the results obtained for STEEL_I$_{5177}$ we can conclude that these differences are even higher for larger problems. In this case, BST_MTCPU is nearly 6 times faster than BST_SCPU, while BST_HYB is more than 20 times faster. The reason is that larger problems present a higher inherent parallelism which can be leveraged by the massively parallel architecture of the GPU.

5 Concluding Remarks

We have presented two high performance parallel implementations for the BST method for model reduction. Variant BST_MTCPU is optimized for its execution on a multicore CPU, while BST_HYB targets hybrid platforms composed of a CPU and a GPU. BST_HYB exploits the capabilities of both architectures, the multicore CPU and the many-core GPU, yielding a high performance implementation of the BST model reduction technique. Two levels of parallelism are exploited in this implementation: at the inner level, multithread lineal algebra kernels included in the BLAS library (MKL and CUBLAS) are employed to compute the most time-consuming linear algebra operations; at the outer level, operations proceed concurrently in both architectures, overlapping computations on the CPU and the GPU.

Experimental results on a platform consisting of a state-of-the-art general-purpose multi-core processor and a pre-Fermi GPU show that model order reduction of large-scale linear systems can be significantly accelerated using this kind of platforms.

The promising results obtained encourage us to further improve the developed implementations. On-going and future work include:

- Exploit the use of multiple GPUs to further reduce the computational time and increase the dimension of the affordable problems.
- Evaluate the use of mixed precision techniques that allow to perform most of the computations in single precision arithmetic.

References

1. A. Antoulas, *Approximation of Large-Scale Dynamical Systems*. Philadelphia, PA: SIAM Publications, 2005.
2. P. Benner, E. S. Quintana-Ortí, and G. Quintana-Ortí, "Efficient numerical algorithms for balanced stochastic truncation," *Internat. J. in Applied Mathematics and Computer Science*, vol. 1, no. 1, pp. 15–21, 2005.
3. R. Freund, "Reduced-order modeling techniques based on Krylov subspaces and their use in circuit simulation," in *Applied and Computational Control, Signals, and Circuits*, B. Datta, Ed. Boston, MA: Birkhäuser, 1999, vol. 1, ch. 9, pp. 435–498.
4. V. Volkov and J. Demmel, "LU, QR and Cholesky factorizations using vector capabilities of GPUs," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-49, May 2008. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-49.html
5. P. Bientinesi, F. D. Igual, D. Kressner, and E. S. Quintana-Ortí, "Reduction to condensed forms for symmetric eigenvalue problems on multi-core architectures," in *Proceedings of the 8th International Conference on Parallel Processing and Applied Mathematics – PPAM'09*, ser. Lecture Notes in Computer Science, vol. 6067. Springer, 2010, pp. 387–395.
6. S. Barrachina, M. Castillo, F. D. Igual, R. Mayo, E. S. Quintana-Ortí, and G. Quintana-Ortí, "Exploiting the capabilities of modern GPUs for dense matrix computations," *Concurrency and Computation: Practice and Experience*, vol. 21, pp. 2457–2477, 2009.
7. H. Ltaif, S. Tomov, R. Nath, P. Du, and J. Dongarra, "A scalable high performance cholesky factorization for multicore with gpu accelerators," University of Tennessee, LAPACK Working Note 223, 2009.
8. P. Benner, P. Ezzatti, E. S. Quintana-Ortí, and A. Remón, "Using hybrid CPU-GPU platforms to accelerate the computation of the matrix sign function," in *Euro-Par 2009, Parallel Processing - Workshops*, ser. Lecture Notes in Computer Science, H.-X. Lin, M. Alexander, M. Forsell, A. Knüpfer, R. Prodan, L. Sousa, and A. Streit, Eds. Springer-Verlag, 2009, no. 6043, pp. 132–139.
9. U. Desai and D. Pal, "A transformation approach to stochastic model reduction," *IEEE Trans. Automat. Control*, vol. AC–29, pp. 1097–1100, 1984.
10. M. Green, "Balanced stochastic realization," *Linear Algebra Appl.*, vol. 98, pp. 211–247, 1988.

11. A. Varga and K. H. Fasol, "A new square–root balancing–free stochastic truncation model reduction algorithm," in *Prepr. 12th IFAC World Congress*, vol. 7, Sydney, Australia, 1993, pp. 153–156.
12. A. Varga, "Task II.B.1 – selection of software for controller reduction," The Working Group on Software (WGS), SLICOT Working Note 1999–18, December 1999, available from `http://www.slicot.org`.
13. P. Benner, E. Quintana-Ortí, and G. Quintana-Ortí, "State-space truncation methods for parallel model reduction of large-scale systems," *Parallel Comput.*, vol. 29, pp. 1701–1722, 2003.
14. B. Anderson, "An algebraic solution to the spectral factorization problem," *IEEE Trans. Automat. Control*, vol. AC-12, pp. 410–414, 1967.
15. B. Anderson, "A system theory criterion for positive real matrices," *SIAM J. Cont.*, vol. 5, pp. 171–182, 1967.
16. P. Benner, E. Quintana-Ortí, and G. Quintana-Ortí, "Solving linear-quadratic optimal control problems on parallel computers," *Optimization Methods Software*, vol. 23, no. 6, pp. 879–909, 2008.
17. J. Roberts, "Linear model reduction and solution of the algebraic Riccati equation by use of the sign function," *Internat. J. Control*, vol. 32, pp. 677–687, 1980, (Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971).
18. G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed. Baltimore: Johns Hopkins University Press, 1996.
19. P. Benner, R. Byers, E. Quintana-Ortí, and G. Quintana-Ortí, "Solving algebraic Riccati equations on parallel computers using Newton's method with exact line search," *Parallel Comput.*, vol. 26, no. 10, pp. 1345–1368, 2000.
20. A. V. Gerbessiotis, "Algorithmic and Practical Considerations for Dense Matrix Computations on the BSP Model," Oxford University Computing Laboratory, PRG-TR 32, 1997. [Online]. Available: http://web.njit.edu/~alexg/pubs/papers/PRG3297.ps
21. P. Strazdins, "A comparison of lookahead and algorithmic blocking techniques for parallel matrix factorization," Department of Computer Science, The Australian National University, Canberra 0200 ACT, Australia, Tech. Rep. TR-CS-98-07, 1998.
22. P. Benner and J. Saak, "A semi-discretized heat transfer model for optimal cooling of steel profiles," in *Dimension Reduction of Large-Scale Systems*, ser. Lecture Notes in Computational Science and Engineering, P. Benner, V. Mehrmann, and D. Sorensen, Eds. Springer-Verlag, Berlin/Heidelberg, Germany, pp. 353–356, 2005.