EFFICIENT BALANCING BASED MOR FOR SECOND ORDER SYSTEMS ARISING IN CONTROL OF MACHINE TOOLS

Peter Benner, Jens Saak
Mathematics in Industry and Technology, Faculty of Mathematics, Chemnitz University of Technology, Germany

Corresponding author: Jens Saak, Chemnitz University of Technology – Mathematics in Industry and Technology, Faculty of Mathematics,
Reichenhainer Straße 41, 09126 Chemnitz, Germany, jens.saak@mathematik.tu-chemnitz.de

Abstract. Behavioral simulation of machine tools is using discrete structural models. These models result from a finite element analysis being applied to their mechanical structure. Therefore they are in general sparse but very large since many details have to be resolved. This accounts for unacceptable computational and resource demands in simulation and especially control of these models. To reduce these demands and to be able to compute solutions and controls in acceptable, i.e. applicable, time frames model order reduction is applied. Classically modal truncation is used for this task. The reduced order models (ROMs) generated are normally relatively large and often need manual modification by addition of certain technically motivated modes. That means they are at least partially heuristic and cannot be generated fully automatic. Engineers are therefore searching for alternate reduction methods.

Here we will concentrate on the application of balancing based model order reduction techniques. A central topic is to provide a reduced order model for the construction and parameterization of a practicable controller for the application. Our main focus will be on presenting a way to efficiently compute the ROM exploiting the sparsity and second order structure of the FEM semi-discretization, rather than presenting a new reduction technique.

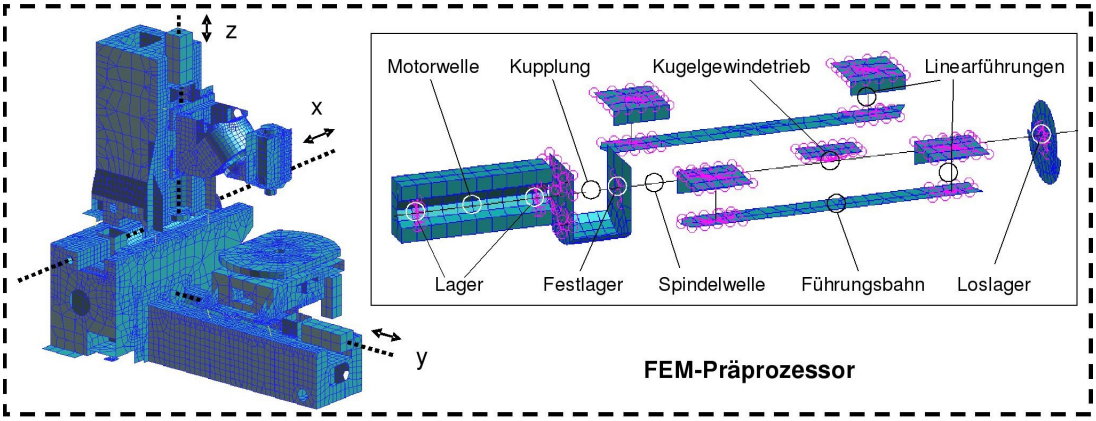


Figure 1: Finite element analysis structural model for an example machine tool

1 Introduction

Large parts of the results in this paper have been achieved within the WAZorFEM project. The joint project with the iwb (TU München) and the ICM (TU Braunschweig) is concerned with model reduction for FEM models of machine tools. These models include the control structure and a model of the stock removal process. Since the FEM models need to resolve many features (geometric, as well as structural, see, e.g., Figure 1) the modeling often leads to rather complex and very high dimensional systems of equations. In order to design practical controllers meeting reaction time restrictions given by the application the need to reduce the dimension of these plant models arises.

Another challenge in the special applications considered in the project is, that modeling the machine tool plants generally leads to second-order differential equations. Where the reduction of linear first order models is fairly well understood in the literature, the reduction of second order models is a field of current research. Especially the preservation of the second order structure in the reduced order model has been an active topic during the recent years.

The classical approach in the reduction of the plant models by engineers is modal truncation. Although leading to fairly good approximations the resulting models are at least partially heuristic. Many aspects in these models need fine-tuning by hand, i.e., special technically motivated modes need to be added by experience of the engineer. Also modal truncation reductions normally yield relatively large models that prevent modern controller design (as usually, order of controller $\geq$ order of plant model). Thus they are of no use in the application subject to this project. The engineers at iwb therefore look for alternative and more automatic reduction methods.

The open literature provides two large classes of methods for second-order structure models. These are based on balancing [23, 11, 12, 26] on the one hand, and Krylov subspace related, e.g., [2, 22, 1, 27, 19, 20, 14] on the other hand. Here we focus on balancing-related models as within the WAZorFEM project, Krylov subspace methods are explored in the the work package treated at ICM, see related paper [13].

1.1 The Simulation Model

The integrated simulation of the machine tool consists of two major parts: The structural model of the machine tool representing its behavior and reaction on certain control inputs, on the one hand, and the control loop generating those inputs, on the other hand. The prior consists of a set of FEM semi-discretized partial differential equations resulting in a large scale second order ordinary differential equation system of the form:

$$M\ddot{x}(t) + D\dot{x}(t) + Kx(t) = Bu(t), \qquad y(t) = C_v\dot{x}(t) + C_p x(t). \tag{1}$$

Here, $M, D, K \in \mathbb{R}^{n\times n}$ are the sparse system matrices reflecting mass, damping and stiffness of the structural model, $B \in \mathbb{R}^{n\times p}$ is the input matrix, $C_v, C_p \in \mathbb{R}^{m\times n}$ are the velocity and position output matrices, $x(t) \in \mathbb{R}^n$ is the state, $y(t) \in \mathbb{R}^m$ the measured output and $u(t) \in \mathbb{R}^p$ represents the input signals sent to the system. In the special case discussed here the damping is considered to follow Rayleigh damping modeling, i.e., the damping matrix is a proportional to mass and stiffness and given as

$$D = \alpha M + \beta K,$$

for certain $\alpha, \beta \in \mathbb{R}$.

1.2 Transformation to First Order

The task of model order reduction now is to find a ROM that captures the essential dynamics of the system and preserves its important properties. Since (1) is of second order we can essentially follow two paths during the computation of the reduced order model. We could decide to preserve the second order structure of the system and compute a second-order reduced model of the form

$$\hat{M}\ddot{\hat{x}}(t) + \hat{D}\dot{\hat{x}}(t) + \hat{K}\hat{x}(t) = \hat{B}u(t), \qquad \hat{y}(t) = \hat{C}_v\dot{\hat{x}}(t) + \hat{C}_p\hat{x}(t), \tag{2}$$

where $k \ll n$ and $\hat{M}, \hat{D}, \hat{K} \in \mathbb{R}^{k\times k}$, $\hat{B} \in \mathbb{R}^{k\times p}$, $\hat{C}_v, \hat{C}_p \in \mathbb{R}^{m\times k}$ and $\hat{x}(t) \in \mathbb{R}^k$. Unfortunately, the global balanced truncation error bound for the reduction is lost if the structure preserving balanced truncation is applied following [23, 11, 26]. (Note that recently it was shown in [30] that it can be reestablished in special cases under additional symmetry assumptions.)

On the other hand, still many simulation and controller design tools expect the system models to be of first order. This motivates the computation of a first-order ROM

$$\hat{E}\dot{\hat{x}}(t) = \hat{A}\hat{x}(t) + \hat{B}u(t), \qquad \hat{y}(t) = \hat{C}\hat{x}(t). \tag{3}$$

Here again $k \ll n$ and $\hat{E}, \hat{A} \in \mathbb{R}^{k\times k}$, $\hat{B} \in \mathbb{R}^{k\times p}$, $\hat{C} \in \mathbb{R}^{m\times k}$ and $\hat{x}(t) \in \mathbb{R}^k$.

The main idea behind both approaches is to rewrite (1) in first order representation and apply balanced truncation to the equivalent first-order model. This can be achieved, e.g., by phase space transformation. That means we introduce the new state variable $z^T(t) := [x^T(t), \dot{x}^T(t)]$, resulting in

$$\underbrace{\begin{bmatrix} I_n & 0 \\ 0 & M \end{bmatrix}}_{=:\mathscr{E}} \dot{z}(t) = \underbrace{\begin{bmatrix} 0 & I_n \\ -K & -D \end{bmatrix}}_{=:\mathscr{A}} z(t) + \underbrace{\begin{bmatrix} 0 \\ B \end{bmatrix}}_{=:\mathscr{B}} u(t), \qquad y(t) = \underbrace{\begin{bmatrix} C_p & C_v \end{bmatrix}}_{=:\mathscr{C}} z(t), \tag{4}$$

with $\mathscr{E}, \mathscr{A} \in \mathbb{R}^{2n\times 2n}$, $\mathscr{B} \in \mathbb{R}^{2n\times p}$, $\mathscr{C} \in \mathbb{R}^{m\times 2n}$ and $I_n \in \mathbb{R}^{n\times n}$ the identity matrix. So the system is formally blown up to double dimension.

1.3 Organization of this Paper

The main focus of our work is to show how the block structure of $\mathscr{E}$, $\mathscr{A}$, $\mathscr{B}$ and $\mathscr{C}$ in (4) can be exploited during computation, such that we can directly work on the original data, i.e., with the original finite element matrices, giving us the opportunity to exploit their sparsity and structure for efficient computation within an alternating directions implicit (ADI) framework [25] employed for implementing balanced truncation for large-scale, sparse systems [3]. This will be discussed in Section 2. Section 3 presents a method to reestablish the second order structure of the original system (1) also for the ROM. Section 4 describes two test examples and shows results of the corresponding numerical experiments. Finally in Section 5 we give some conclusions and an outlook on future tasks concerning this research project.

2 Computation of Reduced Order Models by Balanced Truncation

The efficient algorithms for the computation of the reduced order models resulting from the second order equation (1) can be derived from the aforementioned ADI framework in 2 stages. Starting with a revision of the ADI framework [25, 21, 6] for standard state space systems

$$\dot{x} = Ax + Bu, \qquad y = Cx, \tag{5}$$

in the next section, we will then discuss an extensions of these algorithms for the computation of reduced order models considering generalized state space systems

$$E\dot{x} = Ax + Bu, \qquad y = Cx \tag{6}$$

in Section 2.2. As the second stage we show how the structure of the system matrices in equation (4) can be exploited in the algorithms for generalized state space systems in Section 2.2. That enables us to work directly on the data given in the second order model (1). This will especially allow us to exploit special properties (e.g. symmetry) of the original data in the solvers involved.

2.1 Large Scale Sparse Standard State Space Systems

We are considering balancing based model order reduction[24] throughout this paper. There the main task is to solve the observability and controllability Lyapunov equations

$$AP + PA^T = -BB^T, \qquad A^TQ + QA = -C^TC. \tag{7}$$

From these we compute projection matrices T_l and T_r such that the ROM

$$\dot{\hat{x}}(t) = \hat{A}\hat{x}(t) + \hat{B}u(t), \qquad \hat{y}(t) = \hat{C}\hat{x}(t), \tag{8}$$

is computed as

$$\hat{A} = T_lAT_r, \qquad \hat{B} = T_lB \quad \text{and} \quad \hat{C} = CT_r. \tag{9}$$

As A is assumed to be stable and thus P and Q are positive semi-definite, there exist Cholesky factorizations $P = S^TS$ and $Q = R^TR$. In the so-called *square-root* balanced truncation (SRBT) algorithms [29, 18] these are used to define the projection matrices

$$T_l = \Sigma_1^{\frac{1}{2}}V_1^TR \quad \text{and} \quad T_r = S^TU_1\Sigma_1^{\frac{1}{2}} \tag{10}$$

determining the reduced order model. Here $\Sigma_1^{\frac{1}{2}}$, U_1 and V_1 are taken from the singular value decomposition

$$SR^T = U\Sigma V^T = \begin{bmatrix} U_1U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}, \tag{11}$$

where $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$ is assumed to be ordered such that $\sigma_j \geq \sigma_{j+1} \geq 0$ for all j and $\Sigma_1 = \operatorname{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{r\times r}$. If $\sigma_r > \sigma_{r+1} = 0$ then r is the McMillan degree of the system (i.e. minimal degree) and the resulting ROM is a minimal realization. A Laplace transformation of (5) defines the *transfer function matrix* (TFM)

$$G(s) := C(sI - A)^{-1}B, \tag{12}$$

which directly maps the inputs of (5) to its outputs. Analogously the TFM $\hat{G}(s)$ for (8) is defined. For the TFM error, the global bound

$$\|G - \hat{G}\|_\infty \leq 2\sum_{j=r+1}^{n} \sigma_j \tag{13}$$

can be proven [16].

For large scale sparse systems it is infeasible to compute either P, Q, or their Cholesky factors, since they are generally full matrices requiring $O(n^2)$ memory for storage. Thus, in the low rank ADI (LR-ADI) framework one exploits that both P and Q normally have very low rank compared to n. Therefore the Cholesky factors are replaced by low rank Cholesky factors (LRCFs) in the above defining the low rank square root balanced truncation method (LR-SRBT). The low rank factors can be computed directly by the low rank Cholesky factor ADI iteration (Algorithm 1, see also [6] for a strictly real version).

Note that we need to perform essentially three types of operations. These are: multiplications with the operator F and the solutions of linear systems with F, which are required to determine the shift parameters p_i [7]. Moreover, for the iteration in Algorithm 1, we need to be able to solve shifted linear systems with F, i.e., we must apply $(F + p_iI)^{-1}$.

Algorithm 1 (Low-rank Cholesky factor ADI iteration (LRCF-ADI))

Input: F,G defining $FX+XF^T=-GG^T$ and shift parameters $\{p_1,\dots,p_{i_{max}}\}$
Output: $Z=Z_{imax}\in\mathbb{C}^{n\times ti_{max}}$, such that $ZZ^H\approx X$
1: $V_1=\sqrt{-2\operatorname{Re}(p_1)}(F+p_1I)^{-1}G$
2: $Z_1=V_1$
3: **for** $i=2,3,\dots,i_{max}$ **do**
4: $\quad V_i=\sqrt{\operatorname{Re}(p_i)/\operatorname{Re}(p_{i-1})}(V_{i-1}-(p_i+\overline{p_{i-1}})(F+p_iI)^{-1}V_{i-1})$
5: $\quad Z_i=[Z_{i-1}\ V_i]$
6: **end for**

2.2 Large Scale Sparse Generalized State Space Systems

We will now concentrate on generalized state space systems of the form (6). The natural observability and controllability Lyapunov equations for this system are the generalized Lyapunov equations

$$APE^T+EPA^T=-BB^T,\qquad A^TQE+E^TQA=-C^TC. \tag{14}$$

Throughout this paper we assume, that $E\in\mathbb{R}^{n\times n}$ is invertible. If so, we can formally rewrite the system into standard state space representation by simply multiplying by its inverse from the left, resulting in

$$\dot{x}=\underbrace{E^{-1}A}_{=:\tilde{A}}x+\underbrace{E^{-1}B}_{=:\tilde{B}}u,\qquad y=Cx. \tag{15}$$

Following (5), (7) the observability and controllability Gramians for (15) solve the equations

$$\tilde{A}\tilde{P}+\tilde{P}\tilde{A}^T=-\tilde{B}\tilde{B}^T,\qquad \tilde{A}^T\tilde{Q}+\tilde{Q}\tilde{A}=-C^TC. \tag{16}$$

Inserting the definitions of $\tilde{A}$ and $\tilde{B}$ we observe that $\tilde{P}=P$, but $\tilde{Q}=E^TQE$. So when rewriting Algorithm 1, we need to keep track of all changes carefully to examine whether the final version is actually solving (7) or (16).

The final goal in modifying the algorithm however is to keep the increase in the per step computations as small as possible. Note especially, that transforming the solution of (7) to that of (16) or vice versa only requires one sparse matrix multiplication or one sparse linear system solve with E respectively due to the symmetry of the factorizations we are computing. Both can be computed with $O(n)$ complexity.

In the following we consider the Lyapunopv equation

$$FX+XF^T=-GG^T$$

and distinguish the two cases:

1. $F=\tilde{A}=E^{-1}A$, $G=\tilde{B}=E^{-1}B$, and $X=P$;
2. $F=\tilde{A}^T=A^TE^{-T}$, $G=C^T$, and $X=Q$.

The first of which is the easy case since we already observed, that the solutions of the two Lyapunov equations containing B are equal. Let us now consider the two critical steps in the algorithm. These are the initialization of the LRCF (line 1) and its incrementation (line 6). Starting with the initialization we find:

$$\begin{aligned}V_1&=\sqrt{-2\operatorname{Re}(p_1)}(F+p_1I)^{-1}G\\&=\sqrt{-2\operatorname{Re}(p_1)}(A+p_1E)^{-1}EG\\&=\sqrt{-2\operatorname{Re}(p_1)}(A+p_1E)^{-1}B.\end{aligned}$$

Analogously for the increment we observe:

$$\begin{aligned}V_i&=\sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}}(V_{i-1}-(p_i+\overline{p_{i-1}})(F+p_iI)^{-1}V_{i-1})\\&=\sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}}(V_{i-1}-(p_i+\overline{p_{i-1}})(E^{-1}A+p_iI)^{-1}V_{i-1})\\&=\sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}}(V_{i-1}-(p_i+\overline{p_{i-1}})(A+p_iE)^{-1}EV_{i-1}).\end{aligned}$$

Algorithm 2 (Generalized Low-rank Cholesky factor ADI iteration (G-LRCF-ADI))

Input: E, A and B, or C as in (14) and shift parameters $\{p_1, \ldots, p_{i_{max}}\}$.
Output: $Z = Z_{imax} \in \mathbb{C}^{n \times t i_{max}}$, such that $ZZ^H \approx P$, Q in (14), respectively.

1: **if** right hand side given is C **then**
2: $\quad A = A^T$, $E = E^T$, $G = C^T$
3: **else**
4: $\quad G = B$
5: **end if**
6: $V_1 = \sqrt{-2\operatorname{Re}(p_1)}(A + p_1 E)^{-1} G$
7: $Z_1 = V_1$
8: **for** $i = 2, 3, \ldots, i_{max}$ **do**
9: $\quad V_i = \sqrt{\operatorname{Re}(p_i)/\operatorname{Re}(p_{i-1})}(V_{i-1} - (p_i + \overline{p_{i-1}})(A + p_i E)^{-1}(EV_{i-1}))$
10: $\quad Z_i = [Z_{i-1} V_i]$
11: **end for**

Thus, in both steps we can shift with the mass matrix E instead of the identity at the cost of an additional sparse matrix vector product. The cost for the solution of the shifted linear system here does not change considerably. Surely, it will be slightly more expensive to compute the sparse matrix sum $A + p_i E$ in order to set up the coefficient matrix for sparse direct solvers, than just adding p_i to the diagonal of A when no mass matrix is present. On the other hand, since A and E are normally arising in the same context (e.g. from a finite element semi discretization), they will in general have the same sparsity pattern, such that the computational and memory complexity for the actual solve does not change. For iterative solvers the change comes at the cost of one additional sparse matrix vector product and one vector-vector addition per iteration step. Note especially that we do not even have to compute $\tilde{B} = E^{-1}B$. Instead we can directly initialize the computation with the original B.

For the case where $F = \tilde{A}^T$, we first note that

$$I - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1} = (F + p_i I)^{-1}(F - \overline{p_{i-1}} I),$$

and therefore

$$V_i = \sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}}\left(V_{i-1} - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1} V_{i-1}\right) = \sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}}(F + p_i I)^{-1}(F - \overline{p_{i-1}} I)V_{i-1}.$$

Inserting this in Algorithm 1 we get

$$\begin{aligned}
V_i &= \sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}}(F + p_i I)^{-1}(F - \overline{p_{i-1}} I)V_{i-1} \\
&= \sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}}((A^T + p_i E^T)E^{-T})^{-1}((A^T - \overline{p_{i-1}} E^T)E^{-T})V_{i-1} \\
&= \sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}} E^T (A^T + p_i E^T)^{-1}(A^T - \overline{p_{i-1}} E^T)E^{-T} V_{i-1} \\
&= E^T \left(\sqrt{\frac{\operatorname{Re}(p_i)}{\operatorname{Re}(p_{i-1})}} \left(I - (A^T + p_i E^T)^{-1} E^T\right) \right) E^{-T} V_{i-1}.
\end{aligned}$$

Now observing that the multiplication with E^T in the i-th step is canceled by the E^{-T} in the $(i+1)$-st step, we see that in this case the actual iteration operator changes exactly the same way as above. For the initialization step (line 1) we also have

$$V_1 = E^T \sqrt{-2\operatorname{Re}(p_1)}(A^T + p_1 E^T)^{-1} C^T.$$

That means the above also holds for $i = 1$. The multiplication with E^T here determines whether we are actually computing the solution factor for Q from (14) or $\tilde{Q}$ from (16).

Avoiding E^T completely is obviously the cheapest choice. We then compute the solution of the generalized Lyapunov equation. This is desirable if we can work with generalized ROMs. There we solve the generalized Lyapunov equation and determine the projection matrices T_l and T_r in (10) with respect to these generalized Gramians. The ROM (3) is then computed as (see e.g. [9])

$$\hat{E} = T_l E T_r, \qquad \hat{A} = T_l A T_r, \qquad \hat{B} = T_l B \quad \text{and} \quad \hat{C} = C T_r. \tag{17}$$

If we necessarily need a standard state space form ROM, we can use Algorithm 2 to compute $R = \tilde{R}$, S with $RR^H = P = \tilde{P}$ and $SS^H = Q$ and transform S to $\tilde{S}$ (the factor of $\tilde{Q}$) in a post processing step.

2.3 Large Scale Sparse Second Order Systems

Following the technique presented in the introduction, we trace the reduction of the second order system back to the reduction of a generalized first order system of double dimension. That means the main task in this section will be to map the matrix operation $x = \mathscr{E}^{-1}\mathscr{A}f$, $\mathscr{E}^{-1}\mathscr{A}x = f$ and $(\mathscr{A}+p\mathscr{E})^{-1}\mathscr{E}$ as well as $x = (\mathscr{E}^{-1}\mathscr{A})^T f$, $(\mathscr{E}^{-1}\mathscr{A})^T x = f$ and $(\mathscr{A}^T + p\mathscr{E}^T)^{-1}\mathscr{E}^T$ to operations with the original system matrices M, D, K of (1). In the following we will always decompose $x, f \in \mathbb{R}^{2n}$ into $x = [x_1^T x_2^T]^T$ and $f = [f_1^T f_2^T]^T$ where $x_1, x_2, f_1, f_2 \in \mathbb{R}^n$ to have them fit the block sizes in $\mathscr{E}$ and $\mathscr{A}$.

For the linear algebra operations involved in the ADI iteration for computing the controllability and observability Gramians, we show in the following list how to perform these operations using original data from the second-order model only:

$$
\begin{aligned}
x = \mathscr{E}^{-1}\mathscr{A}f \quad &\Leftrightarrow \mathscr{E}x = \mathscr{A}f \quad && \Leftrightarrow x_1 = f_2,\\
&&& \quad Mx_2 = -Kf_1 - Df_2\\[1ex]
x = (\mathscr{E}^{-1}\mathscr{A})^T f \quad &\Leftrightarrow x = \mathscr{A}^T\mathscr{E}^{-T}f \quad && \Leftrightarrow M^T\tilde{f}_2 = f_2,\\
&&& \quad x_1 = -K^T\tilde{f}_2,\\
&&& \quad x_2 = f_1 - D^T\tilde{f}_2\\[1ex]
\mathscr{E}^{-1}\mathscr{A}x = f \quad &\Leftrightarrow \mathscr{A}x = \mathscr{E}f \quad && \Leftrightarrow x_2 = f_1,\\
&&& \quad Kx_1 = Mf_2 + Df_1\\[1ex]
(\mathscr{E}^{-1}\mathscr{A})^T x = f \quad &\Leftrightarrow \mathscr{A}^T\mathscr{E}^-Tx = f \quad && \Leftrightarrow K^T x_2 = -f_1,\\
&&& \quad x_1 = f_2 + D^T x_2\\[1ex]
x = (\mathscr{A}+p\mathscr{E})^{-1}\mathscr{E}f \quad &\Leftrightarrow (\mathscr{A}+p\mathscr{E})x = \mathscr{E}f \quad && \Leftrightarrow (p^2M - pD + K)x_1 = Df_1 - M(f_2 + pf_1),\\
&&& \quad x_2 = f_1 - px_1\\[1ex]
x = (\mathscr{A}^T + p\mathscr{E}^T)^{-1}\mathscr{E}^T f \quad &\Leftrightarrow (\mathscr{A}^T + p\mathscr{E}^T)x = \mathscr{E}^T f \quad && \Leftrightarrow \tilde{f}_2 = M^T f_2,\\
&&& \quad (p^2M^T + pD^T - K^T)x_2 = f_1 - p\tilde{f}_2,\\
&&& \quad x_1 = \tilde{f}_2 + D^T x_2 + pM^T x_2
\end{aligned}
$$

From the rightmost column we see that we can perform all matrix operations needed by Algorithm 2 and its preceding parameter computation directly using the original system matrices M, D, K, B, C_p, C_v. Computation of the two matrix polynomials and their usage in sparse direct solvers is *cheap* with the same arguments as in Section 2.2. The important message here is that exploiting the block structure of the $2n \times 2n$ matrices in the equivalent first order representation, we can reduce the computational and storage cost to essentially $O(n)$. That means all system matrices can be stored in $O(n)$.

3 Regaining the Second Order Structure for the Reduced Order Model

Second order balanced truncation has been introduced in [23]. The general idea for reducing the second order system to a second order ROM is essentially as follows: the system (1) is equivalently rewritten to first order form (4). Then from the first order system the balancing matrices are obtained. To this end [23] defines the second order Gramians based on the equivalent first order system in standard state space form

$$
\begin{bmatrix} \dot{x} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & I_n \\ -M^{-1}K & -M^{-1}D \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}B \end{bmatrix} u, \qquad y = \begin{bmatrix} C_p & C_v \end{bmatrix} u.
$$

For this system the Gramians P and Q as in (7) are computed. These are compatibly partitioned as

$$
P = \begin{bmatrix} P_p & P_o \\ P_o^T & P_v \end{bmatrix}, \qquad Q = \begin{bmatrix} Q_p & Q_o \\ Q_o^T & Q_v \end{bmatrix}. \tag{18}
$$

The second order position Gramians are given as P_p and Q_p. Analogously P_v and Q_v define the velocity Gramians. Using pairs of these we can now define the position balanced (P_p, Q_p), velocity balanced (P_v, Q_v), position-velocity balanced (P_p, Q_v) and velocity position balanced (P_v, Q_p) ROMs following [26, Definition 2.2]. Now, e.g., the

position balancing Gramian pair (P_p, Q_p) takes the role of (P, Q) in the computation of the projectors T_l and T_r in (10) and the reduced order system (2) is obtained according to

$$\hat{M} = T_l M T_r, \qquad \hat{D} = T_l D T_r, \qquad \hat{K} = T_l K T_r, \qquad \hat{B} = T_l B \quad \text{and} \quad \hat{C}_v = C_v T_r \quad \hat{C}_p = C_p T_r.$$

To preserve the stability and symmetry of the original system the projection is performed by an orthogonal matrix T as in

$$\hat{M} = T^T M T, \qquad \hat{D} = T^T D T, \qquad \hat{K} = T^T K T, \qquad \hat{B} = T^T B \quad \text{and} \quad \hat{C}_v = C_v T \quad \hat{C}_p = C_p T.$$

In general for a non-symmetric system we will not have $T_l^T = T_r$ and thus the balancing of the Gramian product (11) is no longer ensured. Therefore also the global error bound (13) is lost. For systems where M, D, K are symmetric, $C_v = 0$ and $C_p = B^T$ [30] reestablishes the error bound. The key idea there is to use the equivalent first order model

$$\begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix} \dot{z}(t) = \begin{bmatrix} 0 & -K \\ -K & -D \end{bmatrix} z(t) + \begin{bmatrix} 0 \\ B \end{bmatrix} u(t), \qquad y(t) = \begin{bmatrix} B^T & 0 \end{bmatrix} z(t), \tag{19}$$

and thus regain the symmetry in the first order system. Although these conditions might seem rather academic, there is a large class of systems arising in electrical engineering when developing RLCK circuits, which have exactly these properties. Velocity balancing, position-velocity balancing and velocity-position balancing can be applied similarly (see [26] for details).

In the floowing we present a method that no longer requires to compute the full Gramians for the first order representation to obtain the second order Gramians. Instead we rather compute low rank Cholesky factors of the second order Gramians from low rank Cholesky factors of the first order Gramians, which again enables us to perform all computations in low rank fashion and thus reduces the expenses to those of the ADI framework.

Let S be a low rank Cholesky factor of the Gramians Q computed by the (G-)LRCF-ADI Algorithm for either of the two first order representations. Note that the block structure in (19) can be exploited analogously to the procedure presented in Section 2.3. Note further that we have $\tilde{S}_1 = S_1$ since the $(1,1)$-block in $\mathscr{E}$ is I_n in (4). For the transformation (19) we need to consider $-K$ for transforming the LRCFs S_1 and $\tilde{S}_1$. We can now compatibly partition $S^H = [S_1^H S_2^H]$ as above and

$$\begin{bmatrix} Q_p & Q_o \\ Q_o^T & Q_v \end{bmatrix} = Q = SS^H = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \begin{bmatrix} S_1^H & S_2^H \end{bmatrix} = \begin{bmatrix} S_1 S_1^H & S_1 S_2^H \\ S_2 S_1^H & S_2 S_2^H \end{bmatrix}$$

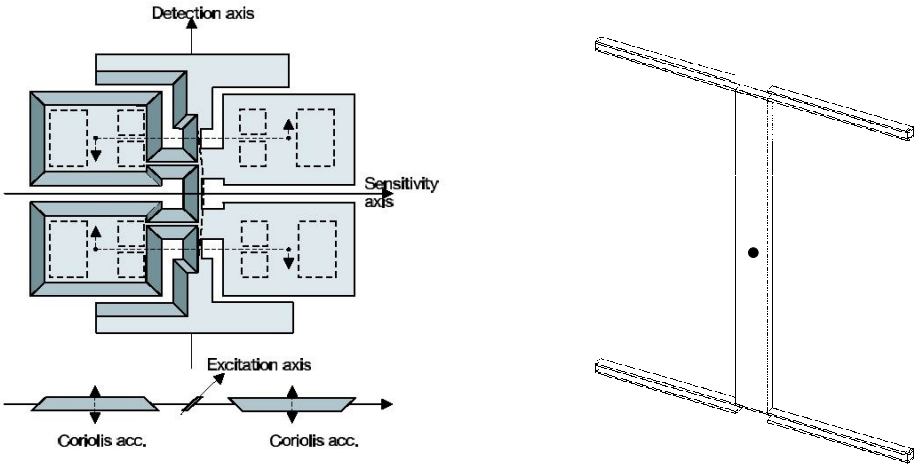
So $Q_p = S_1 S_1^H$, such that we can compute the second order low rank Cholesky factor is directly given as the upper n rows S_1 of the low rank Cholesky factor S. Analogously we can compute the LRCF R_1 of the second order position observability Gramian P_p from the LRCF R of the first order Gramian P. Also, using the lower n rows of the first order Gramian factors we can compute the second order velocity Gramians in case we want to apply any sort of velocity based balancing.

4 Numerical Experiments

In this section, we present some numerical results for balanced truncation applied to second-order models. The models resulting from machine tool models as described in the Introduction contain stiff springs and rigid body modes resulting in systems with zero and infinite poles. These cannot be handled with the methods described in this paper. Modifications to deal with these issues have been derived within the WAZorFEM project work, see [4], and are also known for first-order problems [15, 28], but are not yet available in the software used for performing the tests for large-scale problems. Current work includes incorporating these modifications and corresponding numerical experiments will be reported in the future. For these reasons we chose two second-order models with invertible mass matrix from a different application area in order to demonstrate the efficiency of the proposed algorithms.

Second order structure of the model also arises in many applications where micro mechanical systems are concerned. As two examples of such systems we are using the Butterfly Gyro example [10] from the Oberwohlfach Model Reduction Benchmark Collection [1] and a micro mechanic acceleration sensor investigated at FhG IIS/EAS Dresden in cooperation with Robert Bosch GmbH [17]. Sections 4.1 and 4.2 give brief introductions to the two models followed by a presentation of the reduction results. Also we compare with results we obtained with the parallel matrix sign function implementation in PLiCMR [8] on the Chemnitz Linux Cluster CHiC [5]. All computations have been carried out on an Intel Xeon Dual Core 3.0 GHz machine running 64 GB RAM. Note that this is a 64Bit architecture allowing us to address more than 2GB RAM in MATLAB, which is the main reason to use

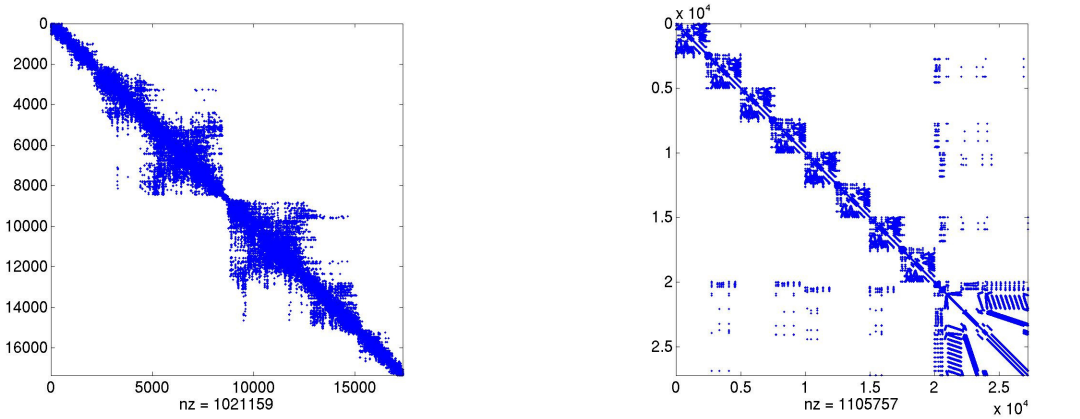
[1] `http://www.imtek.de/simulation/benchmark/`



(a) The *Butterfly Gyro*

(b) Base Configuration of an acceleration sensor

Figure 2: Example model schematics



(a) Stiffness matrix for the *Butterfly Gyro*

(b) Stiffness matrix for the acceleration sensor example

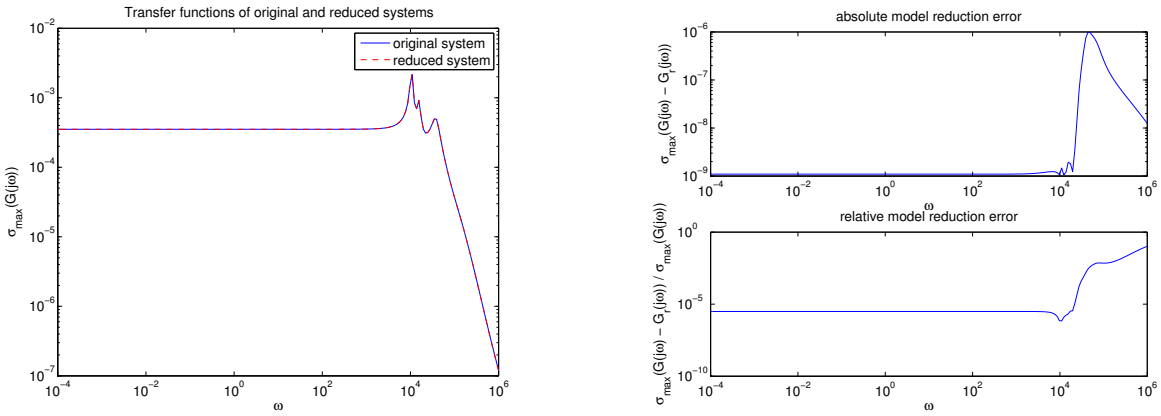
Figure 3: Sparsity patterns of the second order system matrices

it. Also the results shown here have been achieved using the reference implementation of Algorithm 2 found in the appendix applied directly to the equivalent first order models. (The primary reason for the need of more than 2GB memory.) The structure exploitation presented in Section 2.3 will go directly into the M.E.S.S. MATLAB-toolbox[2] and should drop the memory requirement below 2GB for both examples making them computable on standard 32Bit workstations.

4.1 The Butterfly Gyro

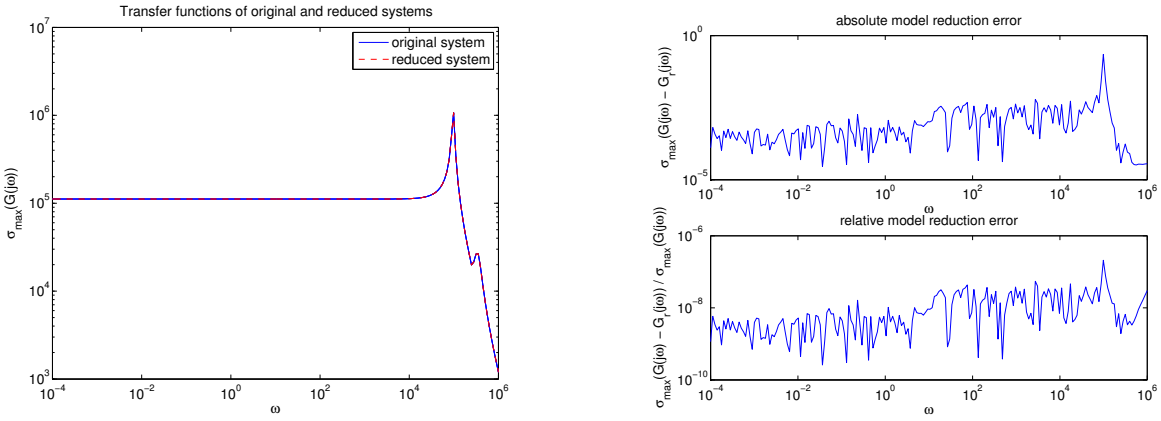
The *Butterfly Gyro* is a vibrating micro mechanical gyro for application in inertial navigation. The gyroscope is a three layered silicon wafer stack of which the actual sensor element is the middle layer. The name of the device is derived from the fact, that the sensor is set up as a pair of double wings connected to a common beam (See Figure 2 (a)). The input matrices have been obtained by an ANSYS model. The original model consists of 17,361 degrees of freedom resulting in an order 17,361 second order original model. Thus the equivalent first order original model is of dimension 34,722. Both systems have a single input and 12 outputs and the number of nonzero entries in the system matrices is of order 10^7 (see also Figure 3 (b)). The reduction results can be found in Figure 4. The ROM here is of order 18 and has been computed in roughly 1 hour including pre and post processing. Where pre processing means assembly of the first order original system and computation of the shift parameters. Here we used 20 parameters following the heuristic parameter choice as proposed by Penzl. Post processing is the computation of the original and reduced order Bode plots, as well as the absolute and relative approximation errors at 200 sampling points. In comparison the computation of an order 45 ROM on 256 nodes of the CHiC using PLICMR plus post processing on the same Xeon machine as above can be done in roughly half the time but results show slightly worse numerical properties, i.e., errors there are a bit larger.

[2] `http://www.tu-chemnitz.de/mathematik/industrie_technik/software/mess.php?lang=en`



(a) Bode Plot (b) Error Plots

Figure 4: Results for the Gyro example



(a) Bode Plot (b) Error Plots

Figure 5: Results for the acceleration sensor example

4.2 Micro mechanic Acceleration Sensor

The basic structure of the micro mechanic acceleration sensor consists of a seismic mass coupled to two beam configurations at both its ends (See Figure 2 (b)). The beam configurations as well as the seismic mass have been modeled by beam elements and connected by coupling elements. The original simulations [17] have been performed using SABER and ANSYS. The model has 4 inputs an 3 outputs. The order of the second order system is 27,225 resulting in an equivalent first order system of order 54,450. Although the system is considerably larger, the number of nonzero elements is about the same as in the gyro example case (see Figure 3 (a)). The single computation steps here are the same as for the gyro example. Here we used 25 shift parameters and the computation took about 1 hour as in the gyro example. The larger dimension is compensated by the fact, that here especially the controllability Gramian factor computation converged in only 23 steps to the required accuracy. The ROM, for which the results can be found in Figure 5, is of order 25. Here we cannot provide a comparison with the CHiC experiments, since PLICMR crashed for this model due to memory allocation errors in the required SCALAPACK routines.

5 Conclusions

We have shown that the G-LRCF-ADI delivers an efficient framework for the computation of low rank factorized solutions of (generalized) Lyapunov equations arising in balanced truncation contexts for generalized first order systems, as well as second order systems. Although generally MOR methods for second order systems require to reformulate the problem for an equivalent double sized first order system by phase space transformation, we showed that the special structure of the block matrices in the first order forms can be exploited to work directly with the sparse $n \times n$-dimensional matrices instead of the generally full $2n \times 2n$ matrices of the standard state space first order form. Also we presented an extension of the Meyer/Srinivasan approach that allows us to compute the factors of the second order Gramians directly from the low rank Cholesky factors for the equivalent first order form, without having to compute the full Gramians. Numerical experiments have shown the efficiency of our method and proof that our single processor MATLAB codes are even competitive with the parallel computer implementation of

the matrix sign function method in PLICMR.

As mentioned earlier the integration of the methods for the descriptor case, i.e., the case where the mass matrix is no longer invertible, is subject to current research and implementation work. The results and especially numerical experiments for the machine tool models will be presented in upcoming joint WAZorFEM publications.

6 Acknowledgements

This research has been supported by the research project WAZorFEm: *Integrierte Simulation des Systems "Werkzeugmaschine-Antrieb-Zerspanprozess" auf der Grundlage ordnungsreduzierter FEM-Strukturmodelle* funded by the German Science Foundation (DFG) under grant BE 2174/9-1.

The authors whish to thank A. Köhler at FhG IIS/EAS for providing the acceleration sensor test example.

7 Appendix

```
function [Z,res,niter]=lrcf_g_adi(F,G,E,p,maxiter,rtol,type)
% function [Z,res,niter]=lrcf_g_adi_r(F,G,E,p,maxiter,rtol)
%
% Generates a low rank matrix Z such that X=Z*Z.' solves the
% Lyapunov equation:
%
% F*X*E' + E*X*F' = G*G'                                (1)
%
% Inputs:
%
% F,G,E    The matrices F,G and E in the above euqation (1).
%  p       a vector of shift parameters
% maxiter  maximum iteration number.
% rtol     normalized residual stopping criterion tolerance
% type     type of the Lyapunov equation:
%           'B' -> A*X*E' + E*X*A' = B*B'
%           'C' -> A'*X*E + E'*X*A = C'*C
%
% Outputs:
%
% Z     The solution factor Z such that Z*Z'=X solves (1).
% res   the vector of residuals
% niter the number of iteration steps taken
%

% input parameters not checked!

n=size(F,1);
l=length(p);

if type=='C'
  F=F';G=G';E=E';
else
  if type~='B'
    error('type has to be one of ''B'', ''C'' ');
  end
end
gnorm=normest(G*G');
pc=p(1);
V1=sqrt(-2*real(pc))*((F+pc*E)\G);
Z=V1;
res=zeros(1,maxiter);
res(1)=res2g(F,G,E,Z)/gnorm;
fprintf(1,'step: %4d  normalized residual: %d\n',1,res(1));
for i=2:maxiter
  ip=mod(i+l-1,l)+1;
  pp=pc;
  pc=p(ip);
```

```
  V1=sqrt(real(pc)/real(pp))*(V1-(pc+conj(pp))*((F+pc*E)\(E*V1)));
  Z=[Z V1];
  res(i)=res2g(F,G,E,Z)/gnorm;
  fprintf(1,'step: %4d  normalized residual: %d\n',i,res(i));
  if (res(i)<rtol)||(norm(V1,2)/normest(Z)<n*eps)
    break;
  end
end
niter=i;

function nor = res2g(F,G,E,Z)
% function norm = res2(F,G,E,Z)
%
% Computes the 2 Norm of the Lyapunov residual of Z exploiting the
% symmetry of the residual operator
%
% R := F*Z*Z'*E' + E*Z*Z'*F' + G*G'
%
% That means it computes the spectral radius of this operator by a
% power iteration, exploitig the sparsity of F and rectangular
% structure of Z and G. Thus it can be computed in O(n) effort and
% is therefore much cheaper than the computation of e.g. the
% Fobenius norm.

n=size(F,1);
tol=sqrt(eps);
% Initialization of nomalized iteration vector
q0=ones(n,1);
q0=q0/norm(q0);

% The Power Iteration
s=1;sold=0;i=0;
FZ=F*Z;
while(((abs(s-sold)/s)>tol)&&(i<20))
  sold=s;
  %perform the matrix multiplication
  q0=FZ*(Z'*(E'*q0))+E*((Z*(FZ'*q0))+(G*(G'*q0)));
  q0=q0/norm(q0);
  %compute the Rayleigh quotient
  y1=Z'*(E'*q0);
  y2=FZ'*q0;
  y3=G'*q0;
  s=real(2*(y1'*y2)+y3'*y3); % should be redundant but cuts of numerical
                             % erroneous imaginary parts.
  i=i+1;
end
nor=abs(s);
```

8 References

[1] Z. BAI, K. MEERBERGEN, AND Y. SU, *Arnoldi methods for structure-preserving dimension reduction of second-order dynamical systems*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D. Sorensen, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 173–189.

[2] Z. BAI AND Y. SU, *Dimension reduction of large-scale second-order dynamical systems via a second-order Arnoldi method.*, SIAM J. Sci. Comput., 26 (2005), pp. 1692–1709.

[3] P. BENNER, *Solving large-scale control problems*, IEEE Control Systems Magazine, 14 (2004), pp. 44–59.

[4] ——, *Integrierte Simulation des Systems Werkzeugmaschine - Antriebe - Zerspanprozess auf der Grundlage ordnungsreduzierter FEM-Strukturmodelle - Arbeitsbericht (work report)*, May 2008.

[5] P. BENNER, M. DÖHLER, M. PESTER, AND J. SAAK, *PLiCMR - usage on CHiC*, Chemnitz Scientific Computing Preprint 08-01, TU Chemnitz, 2008.

[6] P. BENNER, J. R. LI, AND T. PENZL, *Numerical solution of large lyapunov equations, riccati equations,*

and linear-quadratic control problems, Numerical Linear Algebra with Applications, 15 (2008), pp. 755–777.

[7] P. BENNER, H. MENA, AND J. SAAK, *On the parameter selection problem in the Newton-ADI iteration for large-scale Riccati equations*, Electronic Transitions on Numerical Analysis, 29 (2008), pp. 136–149.

[8] P. BENNER, E. QUINTANA-ORTÍ, AND G. QUINTANA-ORTÍ, *State-space truncation methods for parallel model reduction of large-scale systems*, Parallel Comput., 29 (2003), pp. 1701–1722.

[9] ——, *Parallel model reduction of large-scale linear descriptor systems via Balanced Truncation*, in High Performance Computing for Computational Science. Proc. 6th Intl. Meeting VECPAR'04, June 28–30, 2004, Valencia, Spain, 2004, pp. 65–78.

[10] D. BILLGER, *The butterfly gyro*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D. Sorensen, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 349–352.

[11] V. CHAHLAOUI, K. A. GALLIVAN, A. VANDENDORPE, AND P. VAN DOOREN, *Model reduction of second-order system*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D. Sorensen, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 149–172.

[12] Y. CHAHLAOUI, D. LEMONNIER, A. VANDENDORPE, AND P. V. DOOREN, *Second-order balanced truncation*, Linear Algebra Appl., 415 (2006), pp. 373–384.

[13] H. FASSBENDER AND A. SOPPA, *Machine tool simulation based on reduced order fe models*. this issue.

[14] J. FEHR, P. EBERHARD, AND M. LEHNER, *Improving the reduction process in flexible multibody dynamics by the use of 2nd order position gramian matrices*, in ENOC, St. Petersburg, Russland, 2008.

[15] F. D. FREITAS, J. ROMMES, AND N. MARTINS, *Gramian-Based Reduction Method Applied to Large Sparse Power System Descriptor Models*, IEEE Transactions on Power Systems, 23 (2008), pp. 1258–1270.

[16] K. GLOVER, *All optimal Hankel-norm approximations of linear multivariable systems and their L^∞-error bounds.*, Int. J. Control, 39 (1984), pp. 1115–1193.

[17] J. HAASE, S. REITZ, S. WÜNSCHE, P. SCHWARZ, U. BECKER, G. LORENZ, AND R. NEUL, *Netzwerk- und Verhaltensmodellierung eines mikromechanischen Beschleunigungssensors*, in 6. Workshop "Methoden und Werkzeuge zum Entwurf von Microsystemen", December 1997, pp. 23–30.

[18] A. J. LAUB, M. T. HEATH, C. C. PAIGE, AND R. C. WARD, *Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms.*, IEEE Trans. Autom. Control, 32 (1987), pp. 115–122.

[19] M. LEHNER AND P. EBERHARD, *Modellreduktion in elastischen Mehrkörpersystemen (Model Reduction in Flexible Multibody Systems)*, at-Automatisierungstechnik, 54 (2006), pp. 170–177.

[20] ——, *A two-step approach for model reduction in flexible multibody dynamics*, Multibody Syst. Dyn., 17 (2007), pp. 157–176.

[21] J.-R. LI AND J. WHITE, *Low rank solution of Lyapunov equations*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 260–280.

[22] R.-C. LI AND Z. BAI, *Structure-preserving model reduction using a Krylov subspace projection formulation.*, Commun. Math. Sci., 3 (2005), pp. 179–199.

[23] D. G. MEYER AND S. SRINIVASAN, *Balancing and model reduction for second-order form linear systems.*, IEEE Trans. Autom. Control, 41 (1996), pp. 1632–1644.

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

[25] T. PENZL, *Algorithms for model reduction of large dynamical systems*, Linear Algebra Appl., 415 (2006), pp. 322–343. (Reprint of Technical Report SFB393/99-40, TU Chemnitz, 1999.).

[26] T. REIS AND T. STYKEL, *Balanced truncation model reduction of second-order systems*, Math. Comput. Model. Dyn. Syst., 14 (2008), pp. 391–406.

[27] B. SALIMBAHRAMI AND B. LOHMANN, *Order reduction of large scale second-order systems using Krylov subspace methods*, Linear Algebra Appl., 415 (2006), pp. 385–405.

[28] T. STYKEL, *Low-rank iterative methods for projected generalized Lyapunov equations*, Electr. Trans. Num. Anal., 30 (2008), pp. 187–202.

[29] M. S. TOMBS AND I. POSTLETHWAITE, *Truncated balanced realization of a stable nonminimal state-space system*, Internat. J. Control, 46 (1987), pp. 1319–1330.

[30] B. YAN, S. X.-D. TAN, AND B. MCGAUGHY, *Second-order balanced truncation for passive-order reduction of RLCK circuits*, IEEE Trans. Circuits Syst. II, 55 (2008), pp. 942–946.