Peter Benner Martin Köhler Jens Saak

Sparse-Dense Sylvester Equations in $\mathcal{H}_2$-Model Order Reduction

Max Planck Institute Magdeburg
Preprints

MPIMD/11-11 December 16, 2011

Impressum:

Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg

Publisher:
Max Planck Institute for Dynamics of Complex Technical Systems

Address:
Max Planck Institute for Dynamics of Complex Technical Systems
Sandtorstr. 1
39106 Magdeburg

`www.mpi-magdeburg.mpg.de/preprints`

Max Planck Institute Magdeburg
Preprints

Peter Benner Martin Köhler Jens Saak

Sparse-Dense Sylvester Equations in $\mathcal{H}_2$-Model Order Reduction

MPIMD/11-11 December 16, 2011

Abstract

In this paper we study the pratical implementation of a new algorithm for $\mathcal{H}_2$-model order reduction (see, for instance [26, 4, 25, 12]), the so called two sided iteration algorithm (TSIA). It is based on the work of Wilson [26] from 1970 and the extensions done by Xu and Zeng in [27]. The main idea behind this algorithm is to fulfill a classical first order optimality condition. Other approaches for $\mathcal{H}_2$-model order reduction are for example the IRKA algorithm [12] which is based on the interpolation of the transfer function. The theoretical connection between both ideas is verified and the numerical behavior of both approaches is compared. An adaption for generalized state space systems is done, too. In order to implement the presented algorithm robustly and efficiently, it is crucial to overcome some numerical and technical problems. We present a new idea to compute the oblique projection and a fast solver for the Sylvester equation. The benefits of the algorithmic improvements presented in this paper are illustrated by several numerical examples.

Contents

Author's addresses:

Peter Benner, Martin Köhler and Jens Saak
Computational Methods in Systems and Control Theory, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstr. 1, 39106 Magdeburg Germany, ({benner,koehlerm,saak}@mpi-magdeburg.mpg.de)

1 Introduction

Dealing with large scale dynamical systems is important in many industrial applications. In design and optimization, it is often impossible to work with the original large scale systems, since the simulation of a newly designed integrated circuit or a new mechanical device usually takes more time than the design cycles of such objects allow. This is economically unacceptable in nearly every field of application. One possible way out of this problem is to use reduced order models with approximately the same properties as the original system. One of the most common technique for this task is balanced truncation introduced by Moore [18] which approximates with respect to the $\mathcal{H}_\infty$-approximation. In our case, we use the $\mathcal{H}_2$-norm for measuring the error. This is given by

$$||H||_{\mathcal{H}_2}^2 = \frac{1}{2\pi} \int\limits_{-\infty}^{+\infty} \operatorname{tr}\left(H(i\omega)^H H(i\omega)\right) \, \mathrm{d}\omega, \tag{1}$$

where $\operatorname{tr}(\cdot)$ denotes the trace of a matrix and $H(s) = C(sI - A)^{-1}B)$ (or $H(s) = C(sE - A)^{-1}B)$) is the transfer function of a *linear time invariant* (or standard state space) system Σ

$$\Sigma : \begin{cases} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{cases} \tag{2}$$

with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{q\times n}$ or a *generalized linear time invariant* (or generalized state space) system

$$\Sigma : \begin{cases} E\dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{cases} \tag{3}$$

with $E \in \mathbb{R}^{n\times n}$. We call $t \geq 0$ time variable, $x(t) \in \mathbb{R}^n$ the *state*, $u \in \mathbb{R}^p$ the *input* and $y \in \mathbb{R}^m$ the *output* of the system Σ. The dimension n of the state is called *order* of the system. Further we assume that A and E are invertible and the matrix pencil (A, E) is regular. The matrix E is assumed to be symmetric positive definite to define a proper inner product.

The biggest problem during the computation of of the $\mathcal{H}_2$-norm is the evaluation of the improper integral (1). It can be shown [1, 12] that this is equivalent to computing the controllability Gramian P solving

$$AP + PA^T + BB^T = 0, \tag{4a}$$

or for the generalized case

$$APE^T + EPA^T + BB^T = 0, \tag{4b}$$

respectively, and

$$||H||_{\mathcal{H}_2}^2 = \operatorname{tr}\left(C^TCP\right). \tag{4c}$$

Alternatively we can use the observability Gramian Q solving

$$A^TQ + QA + C^TC = 0, \tag{5a}$$

Then it holds

$$||H||_{\mathcal{H}_2}^2 = \operatorname{tr}\left(BQB^T\right). \tag{5b}$$

In the following Section 2 we will recall a well known result for $\mathcal{H}_2$-model order reduction and derive the TSIA algorithm [27] based on this. We only review those conditions and results necessary for the algorithm. There exist other characterizations and approaches to obtain $\mathcal{H}_2$ reduced order models. There are for example the Bernstein conditions from [4] or the interpolation based approach of Gugercin, Antoulas and Beattie [12].

In order to get a fast and robust algorithm it is necessary to analyze the individual steps of the algorithm. We will see that a special solver for Sylvester equations is needed which we will present in Section 3. The numerical results for the model order reduction using TSIA and the performance results of the Sylvester solvers are presented in Section 4.

2 First Order Conditions and the TSIA Algorithm

In this section we will review the $\mathcal{H}_2$-model order reduction problem and a first oder condition for an optimal reduced order model. For model order reduction we are interested in the $\mathcal{H}_2$-error between the original system (2) or (3) and a reduced order model

$$\Sigma_r : \begin{cases} \dot{x}_r(t) &= A_r x_r(t) + B_r u(t) \\ y_r(t) &= C x_r(t) \end{cases} \tag{6}$$

with $A_r \in \mathbb{R}^{r\times r}$, $B_r \in \mathbb{R}^{r\times p}$, $C_r \in \mathbb{R}^{q\times r}$ with $r \ll n$ and its transfer function $H_r(s) = C_r(sI - A_r)^{-1}B_r$. So we want to minimize the error

$$J(A_r, B_r, C_r) = ||H - H_r||^2_{\mathcal{H}_2}. \tag{7}$$

Our goal is to find a stable realization for a given dimension r which minimizes J, i.e., we want to find A_r, B_r and C_r which satisfy

$$\nabla J(A_r, B_r, C_r) = 0. \tag{8}$$

From the corresponding Lyapunov equations (4a) and (5a) of the error functional (7), we get

$$A_{err}P_{err} + P_{err}A_{err}^T + B_{err}B_{err}^T = 0, \tag{9a}$$

$$A_{err}^T Q_{err} + Q_{err}A_{err} + C_{err}^T C_{err} = 0, \tag{9b}$$

$$J(A_r, B_r, C_r) = \operatorname{tr}\left(P_{err}C_{err}C_{err}^T\right) = \operatorname{tr}\left(B_{err}^T Q_{err} B_{err}\right), \tag{9c}$$

where

$$A_{err} = \begin{bmatrix} A & 0 \\ 0 & A_r \end{bmatrix}, \qquad B_{err} = \begin{bmatrix} B \\ B_r \end{bmatrix}, \qquad C_{err} = \begin{bmatrix} C & -C_r \end{bmatrix},$$

and

$$P_{err} = \begin{bmatrix} P_{11} & P_{12} \\ P_{12}^T & P_{22} \end{bmatrix}, \qquad Q_{err} = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix}.$$

Already in 1970, Wilson obtained the first derivatives of (7) with respect to the reduced system matrices [26]:

$$\begin{aligned} \nabla J_{A_r} &= 2\left(Q_{22}P_{22} + Q_{12}^T P_{12}\right), \\ \nabla J_{B_r} &= 2\left(Q_{22}B_r + Q_{12}^T B\right), \\ \nabla J_{C_r} &= 2\left(C_r P_{22} - C P_{12}\right). \end{aligned}$$

For the generalized system it can easily be shown that the derivative with respect to B_r and A_r changes to

$$\nabla J_{B_r} = 2\left(Q_{22}B_r + \underbrace{Q_{12}^T E^{-1}}_{\tilde{Q}_{12}^T} B\right) = 2\left(Q_{22}B_r + \tilde{Q}_{12}^T B\right)$$

and

$$\nabla J_{A_r} = 2(Q_{12}^T P_{12} + Q_{22}P_{22}) = 2(\tilde{Q}_{12}^T E P_{12} + Q_{22}P_{22}).$$

The $\tilde{Q}_{12}$ matrix is introduced because we will see during the implementation section that we are able to obtain it directly. There is no need to solve with the E matrix in this way. Setting these three derivatives to zero will lead us to the so called *Wilson conditions* for $\mathcal{H}_2$-optimality [26]:

$$Q_{22}P_{22} + Q_{12}^T P_{12} = 0, \quad Q_{22}B_r + Q_{12}^T B = 0, \tag{10a}$$

or

$$\tilde{Q}_{12}^T E P_{12} + Q_{22}P_{22} = 0, \quad Q_{22}B_r + \tilde{Q}_{12}^T B = 0, \tag{10b}$$

and

$$C_r P_{22} - C P_{12} = 0. \tag{10c}$$

From the derivative we can get a left and a right projection matrix to compute an optimal reduced order system from the original matrices using

$$A_r = W^T A V, \qquad B_r = W^T B \qquad \text{and} \qquad C_r = CV$$

with $V = P_{12}P_{22}^{-1}$ and $W^T = -Q_{22}^{-1}Q_{12}^T$ or $W = -Q_{22}\tilde{Q}_{12}$ for the generalized case. In order to get an oblique projector we have to satisfy

$$W^T V = I \tag{11a}$$

or

$$W^T E V = I. \tag{11b}$$

This is satisfied by construction: From

$$0 = J_{A_r} = Q_{22}P_{22} + Q_{12}^T P_{12}$$

it follows that

$$-Q_{22}P_{22} = Q_{12}^T P_{12}$$

and thus (assuming Q_{22} to be invertible)

$$I = -Q_{22}^{-1}Q_{12}^T P_{12}P_{22}^{-1} = W^T V.$$

The same holds for the generalized case (11b). Another problem is that we can not guarantee that P_{22} and Q_{22} are invertible. To assure this, the reduced model needs to be completely controllable and observable ([1]) which is a strong restriction. In the case that they are invertible, the multiplication from the right is only a transformation of bases and does not change the subspace. The idea by Xu and Zeng [27] was to satisfy the Wilson conditions by setting

$$W = Q_{12} \quad \text{and} \quad V = P_{12},$$

and using the correction equation

$$\tilde{W}^T = (W^T V)^{-1} W^T. \tag{12}$$

In case of the generalized system we set

$$W = \tilde{Q}_{12} \quad \text{and} \quad V = P_{12}$$

and use

$$\tilde{W}^T = (W^T E V)^{-1} W^T E \tag{13}$$

as correction equation. It is evident that the oblique projection property is satisfied for $\tilde{W}^T V$ or $\tilde{W}^T E V$ by using these correction equations. Xu and Zeng [27] proved that the Wilson conditions are then fulfilled. On the other hand numerical experiments show that this becomes unstable already for relatively small original models. We present a solution to this problem in the next subsection.

Another important problem is that if we want to compute the optimal projection subspace we already need the optimal solution A_r, B_r and C_r. But if we knew them, we would already have solved our problem. A possible solution is to start with a reduced model, which emerged from an arbitrary projection of the original model, solve Lyapunov equations (9a) and (9b), compute the projectors, and restart the process with the newly obtained reduced model until we are satisfied. In this way we get a kind of a fixed point iteration. This procedure is called two sided iteration algorithm (TSIA) by Xu and Zeng in [27]. Another similar fixed point iteration which needs that Q_{22} and P_{22} are invertible was presented by van Dooren, Gallivan and Absil in [25]. The difference in their approach is the way to fulfill the oblique projection property. They use the classical idea by Wilson [26], which is too restrictive because they assume that P_{22} and Q_{22} are invertible.

The Wilson conditions do not give us any information whether we found a local or a global minimum. Using equivalence results from [12] it can be shown that we calculate a local minimizer for J.

2.1 Implementation of the TSIA Algorithm

Now that we have discussed the necessary basics of $\mathcal{H}_2$-model order reduction and mentioned the idea of the fixed point type iteration, we want to derive a practically usable algorithm. First we have seen that there is a correction equation which requires a numerically stable solution. A naive approach may lead to a loss of bi-orthonormality and accuracy. If we want to replace this by a more robust technique, we have to analyze what $W^T V = I$ means from a geometrical point of view. The columns of W and V form biorthonormal bases of a subspace in $\mathbb{R}^n$. Therefore we will use a biorthonormal Gram-Schmidt process to achieve this. Therefore we project every column of V or W onto

Algorithm 1 `[V,W]=biorth(V,W)` - biorthonormal Gram-Schmidt process

```
Input: $V \in \mathbb{R}^{n\times r}$, $W \in \mathbb{R}^{n\times r}$
Output: $V \in \mathbb{R}^{n\times r}$, $W \in \mathbb{R}^{n\times r}$ with $W^TV = I$
 1: for $i := 1, \dots, r$ do
 2:     $v := V(:, i)$
 3:     $v := P^i_{VW^T} v$
 4:     $v := \frac{v}{||v||_2}$
 5:     $w := W(:, i)$
 6:     $w := P^i_{WV^T} w$
 7:     $w := \frac{w}{||w||_2}$
 8:     $v := \frac{v}{w^Tv}$
 9:     $V(:, i) := v, \quad W(:, i) := w$
10: end for
```

the k already biorthonormalized columns. We can express this by two oblique projectors VW^T or WV^T,

$$P^k_{(VW^T)} := \prod_{i=1}^{k} \left(I - V(:,i)W(:,i)^T\right)$$

and

$$P^k_{(WV^T)} := \prod_{i=1}^{k} \left(I - W(:,i)V(:,i)^T\right).$$

When we sequentially apply $P^k_{VW^T}$ to the columns of V and $P^k_{WV^T}$ to the columns of W, we obtain Algorithm 1. The projections can be made more robust by using an iterative refinement if the norm of the projected vector changes too much. In the generalized case we have to use the E inner product in the projection. Therefore our E matrix has to be symmetric positive definite. The projections then change to

$$P^k_{(VW^T)} := \prod_{i=1}^{k} \left(I - V(:,i)EW(:,i)^T\right)$$

and

$$P^k_{(WV^T)} := \prod_{i=1}^{k} \left(I - W(:,i)EV(:,i)^T\right).$$

We also have to replace Step 8 in Algorithm 1 by

$$v := \frac{v}{w^TEv}.$$

and the norms in Step 4 and 7 by the energy norm.

In Section 2 we have seen that we only need the P_{12} and Q_{12} blocks of the two error Lyapunov equations (9a) and (9b). These two matrices contain all the subspace information for the projection of the original system Σ. All other information from P_{err} and Q_{err} can be neglected for the projection spaces. We can reduce the computational costs by extracting the two Sylvester equations which correspond to P_{12} and Q_{12}. This leads to

$$AP_{12} + P_{12}A_r^T + BB_r^T = 0 \tag{14}$$

and

$$A^TQ_{12} + Q_{12}A_r - C^TC_r = 0. \tag{15}$$

These Sylvester equations have a special structure. The coefficients are one large and sparse matrix and one small and dense matrix. A Sylvester equation solver which exploits these special properties is developed in Section 3.

Another efficiency improving observation is the evaluation of the error (7). From the properties of the trace and the block structure of the error Lyapunov equation (9a) and (9b), we can transform equation (9c) into

$$\begin{aligned} ||H - H_r||^2_{\mathcal{H}_2} &= \operatorname{tr}\left(B_{err}^T Q_{err} B_{err}\right) \\ &= \operatorname{tr}\left(B^TQ_{11}B\right) + \operatorname{tr}\left(B_r^TQ_{22}B_r\right) + 2\operatorname{tr}\left(B^TQ_{12}B_r\right) \\ &= ||H||^2_{\mathcal{H}_2} + ||H_r||^2_{\mathcal{H}_2} + 2\operatorname{tr}\left(B^TQ_{12}B_r\right). \end{aligned} \tag{16}$$

It is obvious that we need the $\mathcal{H}_2$-norm of the original system once. But this is not possible for large scale systems by solving the Lyapunov equation with a direct solver like the Bartels-Stewart method [2], Hammarling's method [13] or 2-Solve by Sorensen and Zhou [24]. In this case we have to use methods like the ADI ([21, 19]) or rational Arnoldi based [22, 14] algorithms which provide a good approximate solution of the underlying Lyapunov equation. The only additional computation in every step is the evaluation of the $\mathcal{H}_2$-norm of the reduced system. For this small and dense system it is no problem to use the above mentioned classical dense solvers. The information for the remaining term is available in every step. This seems to be a good and efficient way to evaluate the $\mathcal{H}_2$-error cheaply. But the accuracy of the $\mathcal{H}_2$-norm of the original system has a strong influence. The approximate solution of the Lyapunov equation causes a small error in the $\mathcal{H}_2$-norm. But if the reduced system is a quite good approximation, this perturbation will affect the evaluation of the error formula (16). For example, if we have $||H - H_r||_2^2 = 0$ in exact arithmetics,

$$||H||_2^2 + ||H_r||_2^2 = -2\operatorname{tr}\left(B^TQ_{12}B_r\right),$$

then it is obvious that $2\operatorname{tr}\left(B^TQ_{12}B_r\right) < 0$ holds. If we now have a perturbation in $||H||_2^2$ such that for the computed quantities, $||H||_2^2 < ||H_r||_2^2$, the whole expression (16) can be smaller than zero. This conflicts with properties of a norm. Unfortunately, another problem occurs when we use Equation (16). The computed error is the one of the previous reduced system from the last iteration step. That is why the algorithm will always iterate one step more than necessary if this is used as a stopping criterion. Notwithstanding this problem, we derive Algorithm 2 for the solution of the model order reduction problem.

In case of a generalized system we have to change the biorthonormalization step, as shown above. The modifications for the Sylvester equations are obtained

Algorithm 2 Two sided iteration algorithm (TSIA) with $\mathcal{H}_2$-error computation

```
Input: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $C \in \mathbb{R}^{m\times n}$ and initial reduced model $A_r \in \mathbb{R}^{r\times r}$, $B_r\mathbb{R}^{r\times p}$ and $C_r \in \mathbb{R}^{m\times r}$
Output: $A_r \in \mathbb{R}^{r\times r}$, $B_r \in \mathbb{R}^{r\times p}$, $C_r \in \mathbb{R}^{m\times r}$ satisfying the Wilson conditions (10) approximately.
 1: $h := ||H||_2^2$ with $H(s) = C(sI - A)^{-1}B$
 2: $i := 1$
 3: while not converged do
 4:     Solve $AP_{12} + P_{12}A_r^T + BB_r^T = 0$.
 5:     Solve $A^TQ_{12} + Q_{12}A_r - C^TC_r = 0$.
 6:     Solve $A_r^TQ_{22} + Q_{22}A_r + C_r^TC_r = 0$.
 7:     $e_{i-1} := ||H - H_r||_2^2 = h + \operatorname{tr}\left(B_r^TQ_{22}B_r\right) + 2\operatorname{tr}\left(B^TQ_{12}B_r\right)$
 8:     $[V_i, W_i] := \texttt{biorth}(P_{12}, Q_{12})$
 9:     $A_r := W_i^TAV_i$, $B_r := W_i^TB$ and $C_r := CV_i$
10:     $i := i + 1$
11: end while
12: Solve $A^TQ_{12} + Q_{12}A_r - C^TC_r = 0$.
13: Solve $A_r^TQ_{22} + Q_{22}A_r + C_r^TC_r = 0$.
14: $e_{maxit} := ||H - H_r||_2^2 = h + \operatorname{tr}\left(B_r^TQ_{22}B_r\right) + 2\operatorname{tr}\left(B^TQ_{12}B_r\right)$
```

from the corresponding equivalent standard state space representation of our generalized system (3):

$$\dot{x}(t) = \underbrace{E^{-1}A}_{\tilde{A}}\,x(t) + \underbrace{E^{-1}B}_{\tilde{B}}\,u.$$

By inserting $\tilde{A}$ and $\tilde{B}$ in Equation (14) we end up with

$$\begin{aligned} \tilde{A}P_{12} + P_{12}A_r^T + \tilde{B}B_r^T &= 0 \\ \Leftrightarrow \quad E^{-1}AP_{12} + P_{12}A_r^T + E^{-1}BB_r^T &= 0 \\ \Leftrightarrow \quad AP_{12} + EP_{12}A_r^T + BB_r^T &= 0, \end{aligned} \tag{17}$$

which we call a *semi generalized* Sylvester equation. For the second equation (15) we get

$$\begin{aligned} \tilde{A}^TQ_{12} + Q_{12}A_r - C^TC_r &= 0 \\ \Leftrightarrow \quad A^TE^{-T}Q_{12} + Q_{12}A_r - C^TC_r &= 0 \\ \Leftrightarrow \quad A^T\tilde{Q}_{12} + E^T\tilde{Q}_{12}A_r - C^TC_r &= 0, \end{aligned} \tag{18}$$

from which we obtain $\tilde{Q}_{12}$ directly without an extra solve with E.

As we can easily observe, we have to solve two special Sylvester equations in every step. In order to get a fast algorithm we have to solve these efficiently. An exact solution is necessary if we want to use the $\mathcal{H}_2$-error formula and obtain the correct subspace information. Therefore we present what we call a semi-direct Sylvester solver in Section 3 exploiting these special structures.

Convergence Criteria. So far we did not care about the convergence or convergence criteria. The convergence can not be guaranteed, like for the

IRKA algorithm [12]. But Xu and Zeng showed that if the algorithm converges the optimality conditions are satisfied. The proof is based on plugging $W^T = (Q_{12}^T P_{12})^{-1} Q_{12}^T$ and $V = P_{12}$ instead of $W = -Q_{12}Q_{22}^{-1}$ and $V = P_{12}P_{22}^{-1}$ into the first derivatives of J and checking the Wilson conditions. It is obvious that this proof works for the generalized case as well. In order to retrieve stopping criteria we will mention a set of different ideas. The first self-evident idea is to use the $\mathcal{H}_2$-error which can be computed easily in every step. Because we want a minimizing solution the change of the $\mathcal{H}_2$-error should be checked for convergence. Therefore we can use for example

$$\frac{|e_{i-1} - e_{i-2}|}{e_0} < tol. \tag{19}$$

If we specify an absolute error bound, the computed solution may not fulfill the Wilson conditions because it is not converged at the time the criterion is satisfied. Another criterion is that the reduced system be stable. Numerical instabilities can cause an unstable reduced system matrix in one or more iteration steps. If this happens we have to do at least one step more even if another criterion tells us that the algorithm converged. The check if the system is stable can be done by computing all eigenvalues of the reduced system matrix. Because our reduced system is small they can be computed with standard dense methods like the Francis-QR algorithm [10]. Another idea is to check the subspace information which is contained in W and V. It is clear that if the algorithm has converged; the subspaces will not change anymore. This criterion can be expressed as

$$\operatorname{rank}([W_i \; W_{i-1}]) \equiv \operatorname{rank}([W_i]) \tag{20}$$

and

$$\operatorname{rank}([V_i \; V_{i-1}]) \equiv \operatorname{rank}([V_i]). \tag{21}$$

Due to roundoff errors and other numerical instabilities these conditions seldomly hold in our test cases.

Another criterion could be the evaluation of the three gradients ∇J_{A_r} ∇J_{B_r} and ∇J_{C_r} of J. Nearly all information for these are available in every iteration step or can be computed cheaply. As we will see in Section 4, this is not a good criterion because it does not behave as expected in exact arithmetics.

We will find another stopping criterion by analyzing the relation between the IRKA algorithm [12] and the TSIA in the next subsection.

2.2 Equivalence of IRKA and TSIA

The IRKA algorithm was presented by Gugercin, Antoulas and Beattie in [12] for single-input single-output (SISO) systems, with a possible extension to the MIMO case briefly discussed. Their main idea is to interpolate the transfer function of a standard state space system (2) at a set of interpolation points. They characterize optimality conditions for this set of interpolation points and prove that these conditions are equal to the Wilson conditions or the Bernstein-Hyland conditions [4]. For the projection they construct two subspaces

$$\mathcal{V} = \operatorname{span}(V) = \operatorname{span}\left((\sigma_1 I - A)^{-1}B, \ldots, (\sigma_r I - A)^{-1}B\right) \tag{22}$$

and

$$\mathcal{W} = \operatorname{span}(W) = \operatorname{span}\left((\bar{\sigma_1} I - A)^{-T} C^T, \ldots, (\bar{\sigma_r} I - A)^{-T} C^T\right) \tag{23}$$

with a set of interpolation points $\sigma = \{\sigma_i\}_{i=1}^r$. The matrices V and W have to satisfy the condition $W^T V = I$, like in the TSIA algorithm and can be realized in the same way with a correction equation (12) or a biorthonormalization like we presented it in Subsection 2.1. It tries to construct an optimal set of interpolation points by projecting the original system and computing new interpolation points from the reduced order model. The set for the next step is computed as the mirror image of the poles, i.e., the eigenvalues, of the reduced system matrix A_r. So the points for the next step will be

$$\sigma_i = -\lambda_i(A_r),$$

where $\lambda_i(A_r)$ denotes the i-th eigenvalue of A_r. The convergence check is easily done by checking that the set of interpolation points does not change anymore. So this is independent of any expensive $\mathcal{H}_2$-norm computation, which makes this an easy to evaluate criterion.

If we want to use this criterion for the TSIA algorithm we have to prove that both methods produce comparable subspaces and that TSIA uses the interpolation idea implicitly. The following theorem provides the result for SISO systems. A similar result is given by Kubalińska in [17] for $\mathcal{H}_{2,\alpha}$ model order reduction.

Theorem 1. *Let $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 1}$ and $C \in \mathbb{R}^{1\times n}$ define a SISO linear time invariant system (2) and $A_r \in \mathbb{R}^{r\times r}$, $B_r \in \mathbb{R}^{r\times 1}$ and $C_r \in \mathbb{R}^{1\times r}$ defining the corresponding reduced order system (6). Furthermore A and A_r are assumed to be stable and A_r to be diagonalizeable. Let (A_r, B_r) be controllable and (A_r^T, C_r^T) be observable. If the interpolation points σ_i for the IRKA algorithm are given by the mirror images of the eigenvalues λ_i of A_r,*

$$\sigma_i = -\lambda_i(A_r),$$

then the following holds:

$$\mathcal{V} \equiv \operatorname{span}(P_{12}) \quad \textit{and} \quad \mathcal{W} \equiv \operatorname{span}(Q_{12})\,. \tag{24}$$

Proof. First we prove equality for the $\mathcal{V}$ space in (24). Therefore we need to insert the eigenvalue decomposition of $USU^{-1} = A_r$ into Equation (14) and to multiply from the right by U:

$$A\underbrace{P_{12}U}_{\tilde{P}} + \underbrace{P_{12}U}_{\tilde{P}} S + BB_r^T U = 0.$$

Transforming this equation to

$$-A\tilde{P} - \tilde{P}S = B\underbrace{B_r^T U}_{\tilde{B}}$$

will lead us to a formula for each column of $\tilde{P}$. Since S is a diagonal matrix, every column of $\tilde{P}$ can be written as

$$(-A - S_{jj}I)\tilde{P} = \tilde{B}_j B.$$

Using $\sigma_j = -\lambda_j(A_r) = -S_{jj}$ we obtain

$$(\sigma_j I - A)\tilde{P} = \tilde{B}_j B. \tag{25}$$

Because the pair (A_r, B_r) is controllable, $\tilde{B}_j \neq 0$ holds for all $1 \leq j \leq r$. So the columns of $\tilde{P}$ span the same subspace as $(\sigma_j I - A)^{-1}B$. Because U is invertible the subspace does not change for $P_{12} = \tilde{P}U$. The proof for $\mathcal{W}$ is analogously. □

The theorem shows that both algorithms will produce the same subspaces, so we can use the interpolation point criterion from the IRKA algorithm in the TSIA algorithm, too, or we can interpret the TSIA method as an alternative formulation of IRKA. Thus the convergence theory for IRKA can be used for TSIA as well.

3 Solving Sparse-Dense Sylvester Equations

As we know from the previous section, the efficient solution of specially structured Sylvester equations is a key ingredient for our $\mathcal{H}_2$-model order reduction algorithm. Therefore we will discuss a strategy which exploits the special structure of these equations. We call a Sylvester equation

$$AX + XH + M = 0, \tag{26}$$

with $A \in \mathbb{R}^{n\times n}$, $H \in \mathbb{R}^{r\times r}$ and $M \in \mathbb{R}^{n\times r}$, *sparse-dense* if A is a large and sparse matrix and H is a small and dense matrix. This definition can be used similarly for the *semi generalized* Sylvester Equation

$$AX + EXH + M = 0, \tag{27}$$

with $A \in \mathbb{R}^{n\times n}$, $E \in \mathbb{R}^{n\times n}$, $H \in \mathbb{R}^{r\times r}$ and $M \in \mathbb{R}^{n\times r}$, and the *generalized* Sylvester equation

$$AXF + EXH + M = 0, \tag{28}$$

with $A \in \mathbb{R}^{n\times n}$, $E \in \mathbb{R}^{n\times n}$, $F \in \mathbb{R}^{r\times r}$, $H \in \mathbb{R}^{r\times r}$ and $M \in \mathbb{R}^{n\times r}$. In the next subsection we assume that all conditions for the unique solvability of Equations (26), (27) and (28) are fulfilled [9, 1]. At the end of this section we discuss how we can solve the transposed equation reusing most information from the standard solve. The generalized equation is considered for the sake of completeness.

3.1 Standard Sylvester Equation

After we have seen that an efficient solver for sparse-dense Sylvester equations is necessary, we develop an algorithm which exploits this special structure. We only want to use two admissible operations on the matrix A, the solution of shifted linear systems $(A + pI)$ and the matrix vector product, i.e., an element wise access is forbidden. The eigenvalues and the eigenvectors are not known. Details about the direct solution of $(A + pI)x = b$ are presented in [16]. The main idea of this algorithm was presented by Sorensen and Antoulas in [23] but rejected because the authors considering the (direct) solution of large scale linear systems and the complex arithmetics infeasible. Another problem we are confronted with is that we are not able to compute a Schur decomposition of A because of the runtime complexity and the storage requirements of the QR algorithm [10, Chap. 7]. This prohibits the usage of the classical Bartels-Stewart method [2] or the Hessenberg-Schur method [11]. On the small and dense matrix H we will allow arbitrary operations. To derive an algorithm we have to modify the Sylvester equation (26) in a way that we do not breach our requirements on the matrix A.

We replace H in equation (26) by its complex Schur decomposition, i.e., $H = USU^H$ with $U^H U = UU^H = I$ and get

$$AX + XUSU^H + M = 0.$$

Multiplying this by U from the right we can rearrange it to a different Sylvester equation

$$A\underbrace{XU}_{\tilde{X}} + \underbrace{XU}_{\tilde{X}} S + \underbrace{MU}_{\tilde{M}} = 0. \tag{29}$$

In contrast to equation (26) we can use the special structure of equation (29) to formulate a column-by-column solver. The most important observation is that S is an upper triangular matrix. Using this information we can write down a formula for the first column $\tilde{X}_1$ of $\tilde{X}$:

$$\begin{aligned} A\tilde{X}_1 + \tilde{X}_1 S_{11} + \tilde{M}_1 &= 0 \\ (A + S_{ii} I)\tilde{X}_1 &= -\tilde{M}_1. \end{aligned} \tag{30}$$

The second column of $\tilde{X}$ can be written similarly

$$\begin{aligned} A\tilde{X}_2 + S_{12}\tilde{X}_1 + S_{22}\tilde{X}_2 + \tilde{M}_2 &= 0 \\ (A + S_{22} I)\tilde{X}_2 &= -\tilde{M}_2 - S_{12}\tilde{X}_1. \end{aligned} \tag{31}$$

As we can see easily by induction we obtain a scheme for each column j of $\tilde{X}$:

$$(A + S_{jj} I)\tilde{X}_j = -\tilde{M}_j - \sum_{i=1}^{j-1} S_{ij}\tilde{X}_i. \tag{32}$$

To get the solution of the original equation (26) from the now known solution of Equation (29) we have to multiply $\tilde{X}$ from the right by U^T. The whole procedure is shown in Algorithm 3. Step 4 can be removed if we use an eigenvalue decomposition $H = USU^{-1}$ instead of a Schur decomposition because then S is diagonal. But in that case we have to solve with U in the end. This can cause stability problems and we will not save computing time because reducing the cost for the linear combination is compensated by a higher complexity of the matrix decomposition. A problem of the algorithm is that we need complex arithmetics. Caused by the inexact arithmetics the solution X will not become completely real again. Avoiding this we can truncate the imaginary part of X. In Section 4 we will show that this only causes an error of the order of the machine precision. The runtime complexity of the overall Algorithm 3 is estimated as follows:

Algorithm 3 Solution of the Sparse-Dense Sylvester Equation

```
Input: $A \in \mathbb{R}^{n\times n}$, $H \in \mathbb{R}^{r\times r}$, $M \in \mathbb{R}^{n\times r}$ defining (26)
Output: $X \in \mathbb{R}^{n\times r}$ solving (26)
1: Compute Schur decomposition $USU^T = H$.
2: $\tilde{M} := MU$
3: for $j := 1, \dots, r$ do
4:     $\hat{M} := -\tilde{M}_j - \sum_{i=1}^{j-1} S_{ij}\tilde{X}_i$
5:     Solve $(A + S_{jj}I)\tilde{X}_j = \hat{M}$.
6: end for
7: $X := \tilde{X}U$
```

Step	number of calls	Complexity per call
1	1	$\mathcal{O}(25r^3)$
2	1	$\mathcal{O}(r^2n)$
4	r	$\mathcal{O}(rn)$
5	r	$\mathcal{O}(S)$
7	1	$\mathcal{O}(r^2n)$
	Overall:	$r \cdot \mathcal{O}(S) + \mathcal{O}(r^2n) + \mathcal{O}(25r^3)$

where $O(S)$ denotes the runtime complexity of the solver for equation in Step 5. Because we assume that $r \ll n$, we can neglect the dependencies on r. We will get $r \cdot O(S) + r \cdot O(n)$. The complexity for the solution of a linear system $O(S)$ is at least $O(n)$ (In the case that the matrix is diagonal.) As upper bound for $O(S)$ we have $O(n^3)$ if our large matrix is dense. With modern sparse direct solvers, like UMFPACK [6] or SuperLU [7, 8], the complexity will be much smaller than $O(n^3)$ for typical sparse matrices that appear in practice. Alternatively we can use a good iterative solver. That is why the linear solve in Step 5 will dominate the whole algorithm. We end up with a complexity $O(r \cdot S)$. In contrast to this the classical solvers for the Sylvester equation [11, 2] are in $O(n^3)$ with a relatively large constant.

3.2 The Generalized Cases

Unfortunately we can not handle the two generalized cases in one step because the semi generalized equation (27) is nearly the same as the standard equation (26), but the (complete) generalized equation (28) needs some more investigation.

The semi generalized equation can be rearranged to

$$A\underbrace{XU}_{\tilde{X}} + E\underbrace{XU}_{\tilde{X}}S + \underbrace{MU}_{\tilde{M}} = 0$$

by inserting the Schur decomposition $H = USU^T$ and multiplying from the right by U. For the first column of $\tilde{X}$, like in equation (30), we get

$$(A + S_{ii}E)\tilde{X}_1 = -\tilde{M}_1. \tag{33}$$

For all other columns we have to take care of the E matrix. If we consider the

Algorithm 4 Solution of the Sparse-Dense Generalized Sylvester Equation

```
Input: $A \in \mathbb{R}^{n\times n}$, $E \in \mathbb{R}^{n\times n}$, $F \in \mathbb{R}^{r\times r}$, $H \in \mathbb{R}^{r\times r}$, $M \in \mathbb{R}^{n\times r}$ defining (28)
Output: $X \in \mathbb{R}^{n\times r}$ solving (28)
 1: Compute generalized Schur decomposition $(F, H) = (Q\hat{F}Z^H, Q\hat{H}Z^H)$.
 2: $\tilde{M} := MZ$
 3: for $j := 1, \ldots, r$ do
 4:   $\hat{M} := -\tilde{M}_j - \sum_{i=1}^{j-1} \left(\hat{F}_{ij}\hat{X}_i^A + \hat{H}_{ij}\hat{X}_i^E\right)$
 5:   Solve $(\hat{F}_{jj}A + \hat{H}_{jj}E)\tilde{X}_j = \hat{M}$.
 6:   if $j < p$ then
 7:     $\hat{X}_j^A := A\tilde{X}_j$
 8:     $\hat{X}_j^E := E\tilde{X}_j$
 9:   end if
10: end for
11: $X := \tilde{X}Q^H$
```

second column of the solution

$$\begin{aligned} A\tilde{X}_2 + E\tilde{X}_1S_{12} + E\tilde{X}_2S_{22} + \tilde{M}_2 = 0 \\ \Leftrightarrow \quad (A + S_{22}E)\tilde{X}_2 = -\tilde{M}_2 - S_{12}E\tilde{X}_1, \end{aligned} \tag{34}$$

we can again obtain that formula for all columns as by an induction argument

$$(A + S_{jj}E)\tilde{X}_j = -\tilde{M}_j - E\sum_{i=1}^{j-1} S_{ij}\tilde{X}_i. \tag{35}$$

Algorithm 3 can easily be adapted to solve this equation.

The generalized case does not work directly in this way because we have the F matrix to the right of the first X in equation (28). To overcome this problem, we replace F and H by their generalized Schur decomposition $(Q\hat{F}Z^H, Q\hat{H}Z^H)$. This can be done via the QZ algorithm [10, Chap. 7.7]. Inserting this in (28) we get

$$AXQ\hat{F}Z^H + EQ\hat{H}Z^H + M = 0.$$

By multiplying this from the right with Z we get a similar form like for the standard equation in (29)

$$A\underbrace{XQ}_{\tilde{X}}\hat{F} + E\underbrace{XQ}_{\tilde{X}}\hat{H} + \underbrace{MZ}_{\tilde{M}} = 0. \tag{36}$$

Observing that $\hat{F}$ and $\hat{H}$ are upper triangular matrices leads to a formula for the first column of $\tilde{X}$:

$$\begin{aligned} A\tilde{X}_1\hat{F}_{11} + E\tilde{X}_1\hat{H}_{11} + \tilde{M}_1 = 0 \\ \Leftrightarrow \quad (\hat{F}_{11}A + \hat{H}_{11}E)\tilde{X}_1 = -\tilde{M}_1. \end{aligned} \tag{37}$$

All other columns can be obtained in a substitution scheme like in the standard case, but we have to take care of the linear combinations with A and F. So for an arbitrary column j of $\tilde{X}$ we find

$$A\left(\tilde{X}\hat{F}_{jj}+\sum_{i=1}^{j-1}\hat{F}_{ij}\tilde{X}_i\right)+E\left(\tilde{X}\hat{H}_{jj}+\sum_{i=1}^{j-1}\hat{H}_{ij}\tilde{X}_i\right)+\tilde{M}_j=0,$$

which can be rearranged to

$$(\hat{F}_{jj}A+\hat{H}_{jj}E)\tilde{X}_j=-\tilde{M}_j-\sum_{i=1}^{j-1}\left(\hat{F}_{ij}\underbrace{A\tilde{X}_i}_{\hat{X}_i^A}+\hat{H}_{ij}\underbrace{E\tilde{X}_i}_{\hat{X}_i^E}\right). \tag{38}$$

To obtain the solution of the initial system, we have to multiply $\tilde{X}$ with Q^H from the right. We can evaluate the linear combination in the right hand side in a more efficient way if we precompute the matrix-vector products $\hat{X}_i^A$ and $\hat{X}_i^E$ as soon as we have computed column i of $\tilde{X}$. This will save r^2-3r+2 matrix-vector products, but we consume $\mathcal{O}(2n(r-1))$ more memory. The whole procedure is presented in Algorithm 4. The problem with complex arithmetics is the same as in Algorithm 3 for the standard and the semi generalized equation and can be solved in the same way. The complexity estimation is the same as in the standard case. Although we need more operations in general, the solution of the linear system dominates Algorithm 4.

3.3 Transposed Case

As we have seen in Section 2 we need the solution of the transposed equations (26) or (27), too. If we have already solved the standard equation, we have computed the linear solver (LU-decomposition, preconditioner,...) for $(A+S_{jj}I)$ or $(A+S_{jj}E)$ and the Schur decomposition of A_r, respectively (A_r, E_r). The reuse of this solver will reduce the computational costs for the transposed equation dramatically. We now check in which steps of the algorithm we have to change something. First we take a look at the transpose of equation (26)

$$A^HX+XH^H+M=0. \tag{39}$$

Inserting the complex Schur decomposition $USU^H=H$ yields

$$A^HX+XUS^HU^H+M=0. \tag{40}$$

This is nearly the same as the original problem only now S^H is a lower triangular matrix and we have to start with the $r-$th column of $\tilde{X}$ instead of the first. In the formula (32) for the right hand side of each step we have to modify the sum and we will get

$$\hat{M}=-\tilde{M}_j-\sum_{i=j+1}^{r}\overline{S_{ji}}\tilde{X}_i \tag{41}$$

For the inner linear system we will get $(A^T+\overline{S_{jj}}I)=(A+S_{jj})^H$. This strategy can easily be modified for the solution of the generalized or the semi generalized equation.

problem size	r	$\text{cond}_2\,(W^TV)$	corr. equation $\Vert I - \tilde{W}^TV\Vert_2$	biorth-correction. $\Vert I - \tilde{W}^T\tilde{V}\Vert_2$
$n = 625$	5	6.86576 e+06	1.5851 e−10	1.7719 e−15
	10	9.11312 e+12	1.0988 e−03	4.2150 e−15
$n = 2\,500$	10	1.40444 e+13	5.8070 e−04	1.6743 e−15
	15	5.04833 e+16	1.4482 e+00	8.6232 e−15
$n = 40\,000$	10	3.54464 e+14	1.0580 e−01	9.5054 e−15
	15	1.69087 e+16	2.0499 e+00	8.6848 e−15

Table 1: Error of the correction equation after one step of TSIA

4 Numerical Results

In this section we want to present some numerical experiments with the TSIA algorithm and the underlying sparse-dense Sylvester solvers. The algorithms are implemented in MATLAB® and our upcoming M.E.S.S. library which is written in C and Fortran. Most computations are done with the help of our C library. The hardware consists of two six-core Intel® Xeon®X5650 CPUs with 2.66 GHz and 48GB DDR3 RAM. The software used is the following: CentOS 5.5[1] 64bit, GNU C and Fortran compiler 4.5.1[2], ATLAS 3.8.4[3], Lapack 3.3.1[4], SuiteSparse 3.5[5] and MATLAB 2010b. We also tried the Sylvester equation solver from SLICOT[6], but this change had no influence on the performance results.

As test problem for the standard state space system we use a well known FDM discretization of the convection-diffusion equation from LyaPack [19, 20], because it is an easily scalable problem. The test examples for the generalized case are a semi-discretized heat transfer problem for optimal cooling of steel profiles [3] and a tunable optical filter [15]. As an artificial example we use a system from Penzl [19] with a prescribed spectra.

First, as we already mentioned in Section 2, we compare the correction formula (12) for the skew-projection with our biorthonormalization approach. We use the standard state space example and a random initial projection to obtain a first reduced model. Using this we perform one step of the TSIA Algorithm 2 and check the different ideas. Table 1 shows the results for varying problem dimensions. As we can easily see the correction equation works only for one small case. In all other cases the condition number of W^TV is getting too large to solve the correction equation with a satisfactory error. Our biothornomalization approach on the other hand gives us consistent results even where the problem size is getting larger. So this approach is preferred for large scale problems.

After we have seen how to fulfill the oblique projector condition in a robust way, we will present some convergence results of the TSIA algorithm. The Figures 1 and 2 show the different information available during the iteration for the reduction from order 10 000 or 1 668 to order 10. We can see that the

[1] http://www.centos.org
[2] http://gcc.gnu.org
[3] http://math-atlas.sf.net
[4] http://www.netlib.org/lapack
[5] http://www.cise.ufl.edu/research/sparse/SuiteSparse/
[6] http://www.slicot.org

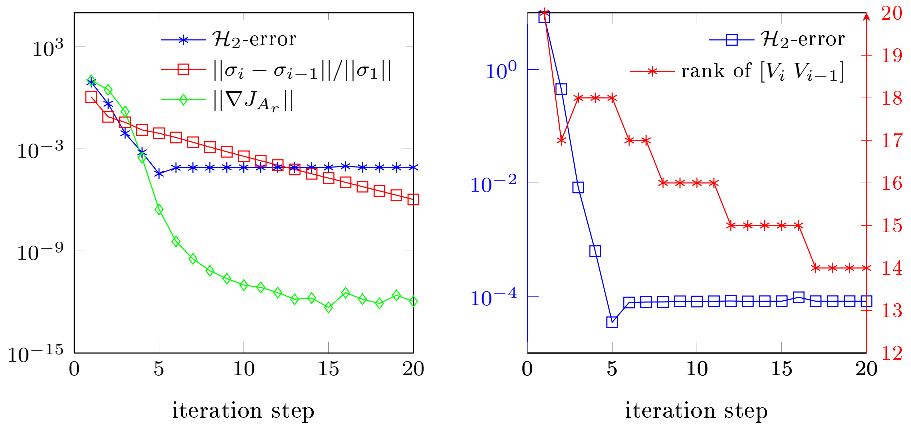


Figure 1: convergence of the TSIA algorithm for the FDM heat equation of order 10 000 to 10

n	$r = 5$	$r = 10$	$r = 15$	$r = 20$	$r = 25$
625	5.141 e−04	1.798 e−06	1.111 e−09	7.185 e−14	9.993 e−15
2 500	1.594 e−02	4.494 e−05	2.642 e−08	4.722 e−11	1.345 e−13
10 000	5.977 e−02	2.646 e−04	1.248 e−06	2.293 e−09	2.088 e−11
40 000	2.573 e−01	8.126 e−03	9.437 e−05	4.147 e−07	2.687 e−09
90 000	6.083 e−01	1.293 e−02	1.702 e−04	1.280 e−06	3.409 e−08
160 000	1.107 e+00	3.183 e−02	2.037 e−04	1.097 e−05	4.877 e−06
250 000	1.751 e+00	4.982 e−02	6.924 e−04	1.162 e−05	2.990 e−07
762 500	3.993 e+00	1.131 e−01	9.493 e−03	2.721 e−05	9.238 e−07
1 000 000	7.130 e+00	2.038 e−01	1.530 e−02	2.469 e−04	9.302 e−06

Table 2: $\mathcal{H}_2$-error of the TSIA algorithm for the standard state space example after 15 steps.

algorithm converges after 5 to 10 steps. It is obvious that all stopping criteria, which are not based on the evaluation of the $\mathcal{H}_2$-error are very slow or not reliable like the norm of the gradient ∇J_{A_r} in the generalized case. For the rank based criterion it can not be guaranteed that we get $\operatorname{rank}([V_i\ V_{i-1}]) = \operatorname{rank}([W_i\ W_{i-1}]) = 10$ at the end. The relative change in the interpolation points can be used as a good criterion, but we are not able to provide problem independent limits. As a good choice we can use

$$||\sigma_i - \sigma_{i-1}|| < \sqrt{\mathbf{u}} \cdot n \cdot ||\sigma_1||$$

where $\mathbf{u}$ denotes the machine precision. Experiments show that this is mostly too strong. Another problem of this criterion is that the convergence of this criterion is relatively slow in comparison with the stagnation of the $\mathcal{H}_2$-error. The small jump-up in Figure 1 is caused by rounding errors. The Tables 2 and 3 show the results after 15 steps of TSIA for different problem sizes. We can see that we are able to reduce problems up to a full order of 1 000 000. This takes from about 4 700 seconds for $r = 5$ to about 54 000 seconds for $r = 25$. So the technical modifications we have done in the TSIA algorithm enable us to solve

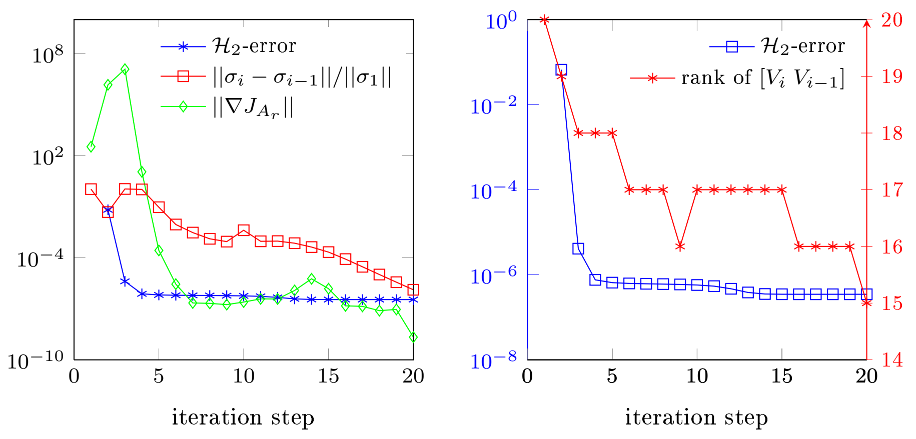


Figure 2: convergence of the TSIA algorithm for the 2D optical filter order 1 668 to 10

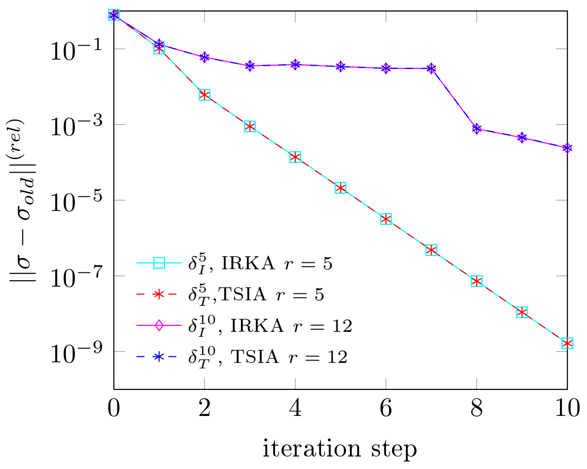
problem		$r = 5$	$r = 10$	$r = 15$	$r = 20$	$r = 25$
rail	1 357	2.223 e−03	1.479 e−03	5.825 e−04	3.003 e−04	1.408 e−04
rail	5 177	2.250 e−03	1.558 e−03	5.930 e−04	2.704 e−04	1.975 e−04
rail	20 209	5.478 e−03	5.278 e−03	1.457 e−03	1.445 e−03	8.005 e−04
rail	79 841	4.211 e−03	1.945 e−03	1.555 e−03	1.494 e−03	1.039 e−03
F-2D	1 688	4.575 e−01	9.859 e−03	4.916 e−04	1.669 e−05	3.648 e−06
F-3D	106 437	6.908 e+02	3.546 e+00	7.271 e−01	3.832 e−02	1.611 e−02

Table 3: $\mathcal{H}_2$-error of the TSIA algorithm for the generalized state space examples after 15 steps.

such problem in less than one day. Without special solvers for the Sylvester equation this will take much longer or is impracticable on current hardware.

The connection between IRKA and TSIA is shown in Figure 3. We do one step of IRKA and use this projection to get the first reduced model for TSIA. Then both methods should converge in the same way as shown in Theorem 1. We see that the difference between the interpolation points is basically the same for IRKA and TSIA. If we take the overall distance $||\delta_{IRKA} - \delta_{TSIA}||_2$, where $\delta = \left\{||\sigma^{(i)} - \sigma^{(i-1)}||_2\right\}_{i=1}^{maxit}$, between the changes of the interpolation points, we can observe that it is not exactly the same and it is growing for larger models and bigger reduced orders. This is caused by rounding errors, stability problems and many other influences.

In order to show an advantage of the TSIA we regard a special model problem presented by Penzl in [20]. It prescribes a certain spectrum such that the Bode plot shows some peaks. The dimension of the model is 408. We approximate this model using IRKA with start parameters computed as some Ritz values of the system matrix, TSIA based on the first IRKA step and TSIA with a special pair of initial projection matrices V and W. The problem setup for the first two cases is the exactly the same as in the previous example. For the third case we



r	$\|\|\delta_I^r - \delta_T^r\|\|_2$
5	1.375 e−10
10	6.196 e−05

Figure 3: Difference of the interpolation points in IRKA and TSIA for the standard state space example, $n = 2\,500$.

choose the projection matrices V and W as

$$V = \begin{pmatrix} I \\ 0 \end{pmatrix} \quad \text{and} \quad W = \begin{pmatrix} I \\ 0 \end{pmatrix}. \tag{42}$$

We want to regard the connection between the dimension of the reduced and the corresponding $\mathcal{H}_2$ error. Therefore we compute reduced order models of dimension 4 up to 24, see Figure 4. Theoretically IRKA and the IRKA started TSIA should produce the same reduced order model as we have already seen in Theorem 1. But for certain reduced dimensions, IRKA exhibits some instabilities and does not deliver the desired $\mathcal{H}_2$ optimal reduced order model, as we see in the figure. The interesting fact is that the $\mathcal{H}_2$-error of the IRKA algorithm jumps up for dimension 20 and 24. Although the equivalent TSIA model should compute the same reduced model it produces the results we except without showing the peak. Even the TSIA with the special initial reduced model produces the same $\mathcal{H}_2$-error beginning at dimension 14. This behavior can be reproduced with different initial approximation parameters. This indicates a better stability of the TSIA when compared to IRKA and may be explained by the fact that IRKA requires an eigendecomposition of possibly non-normal matrices A_r which is avoided by TSIA.

The underlying Sylvester Equation Solver The key ingredient of the fast implementation of the TSIA algorithm are the special Sylvester equation solvers. We solve the systems

$$(A + S_{jj} I)\tilde{X}_j = \hat{M}$$

using sparse LU decomposition. The factors are precomputed and used for Equations (14) and (15). Because the factors are data independent, we can easily compute them in parallel. The parallel parts of the Sylvester solver (and of cause of the TSIA algorithm) are realized by a mixture of OpenMP and Pthreads. Another efficiency improving technique is the so called pattern reuse or single pattern multi value LU decomposition. This special solver technique

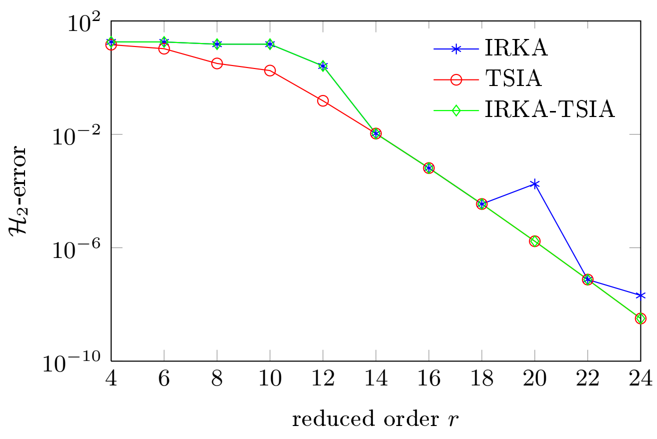


Figure 4: $\mathcal{H}_2$-error depending on the reduced dimension r for a articial problem.

problem	r	Lapack	algorithm 3	speedup
FDM conv-diff $n = 625$	5	1.19s	0.046s	25.870
	10	1.121s	0.052s	21.558
	15	1.127s	0.066s	17.076
FDM conv-diff $n = 10\,000$	5	990.254s	0.210s	4 715.495
	10	996.163s	0.249s	4 000.655
	15	1 005.483s	0.339s	2 966.027
FDM conv-diff $n = 40\,000$	5	54 858.001s	0.996s	55 078.314
	10	54 604.927s	1.279s	42 693.453
	15	54 644.927s	1.992s	27 432.192
FDM conv-diff $n = 1\,000\,000$	5	out of memory	118.392s	-
	10	out of memory	151.984s	-
	15	out of memory	253.312s	-

Table 4: Performence algorithm 3 vs. Lapack DTRSYL

for families $(A + p_iI)$ of linear systems is developed in [16]. To evaluate the performance of Algorithms 3 and 4 we use our standard state space example and the generalized state space example as sparse input data. The dense matrices and the right hand sides are random matrices generated with the MATLAB `rand` function. The classical dense solving was done via the Lapack routines DTRSYL for the standard equation and DTGSYL for the generalized one. Because the TSIA algorithm requires the solution of the standard equation and the transposed one, we solve both using the modifications presented in Subsection 3.3. We can estimate the runtime of TSIA by the runtime of the Sylvester solver. DTRSYL is able to handle the transposed equation directly, but DTGSYL is not. This is due to the generalized Sylvester equation

$$\begin{aligned} AR - LB &= C, \\ DR - LE &= F, \end{aligned}$$

behind this solver. We can not reuse the Schur decomposition for the transposed case like it is done in the standard case, which is one argument why this solver

problem	r	Lapack	Algorithm 4	speedup
rail profile $n = 1\,357$	5	$151.915s$	$0.066s$	2 301.742
	10	$169.927s$	$0.073s$	2 327.767
	15	$170.596s$	$0.087s$	1 960.874
rail profile $n = 5\,177$	5	$9\,718.017s$	$0.128s$	75 922.008
	10	$9\,714.235s$	$0.170s$	57 142.559
	15	$9\,284.022s$	$0.228s$	40 719.395
rail profile $n = 20\,209$	5	$693\,563.279s$ = 8.03 days	$0.571s$	1 214 646.723
	10	-	$0.924s$	-
	15	-	$1.335s$	-
rail profile $n = 79\,841$	5	out of memory	$4.498s$	-
	10	out of memory	$7.287s$	-
	15	out of memory	$10.647s$	-
filter 2D $n = 1\,668$	5	$226.058s$	$0.063s$	3 588.222
	10	$225.560s$	$0.105s$	2 148.190
	15	$226.031s$	$0.101s$	2 237.930
filter 3D $n = 106\,437$	5	out of memory	$377.761s$	-
	10	out of memory	$552.555s$	-
	15	out of memory	$761.060s$	-

Table 5: Performance results of Algorithm 4 vs. Lapack DTGSYL

FDM conv-diff	r		
n	5	15	25
625	3.338 e−17	3.227 e−17	2.123 e−17
10 000	1.319 e−16	5.477 e−17	6.685 e−17
40 000	7.976 e−17	1.289 e−16	1.093 e−16
250 000	9.495 e−17	2.511 e−16	2.561 e−16

Table 6: relative difference between X in $\mathrm{Re}\,(X)$ in Algorithm 3, $\frac{||X-\mathrm{Re}\,(X)||_2}{||\,\mathrm{Re}\,(X)||_2}$

is so slow. Tables 4 and 5 show us the runtime of the Sylvester solver. As we easily can see, even for relatively small systems of order 1 357 or 1 668 the computation time of the classical solver is too large to use it in an iterative outer algorithm. For systems larger than 10 000 it is nearly impossible to use the classical solver. If we want to reduce the heat equation example of order 40 000 to order 10 with the classical solver and we need 15 steps of TSIA, then we need $15 \cdot 54\,604s \approx 9.5$ days to solve the Sylvester equations, which is not acceptable in practice. The most time consuming part in DTRSYL or DTGSYL are the Schur decompositions of the large matrices that have to be performed. Algorithms 3 and 4 avoid such operations on the large matrices.

The only problem is that we need complex arithmetics if the small matrices have complex eigenvalues and Schur vectors. However this does not seem to be a problem, because in exact arithmetics the final solution will be real again and the truncation of the imaginary part should not cause a notable error, since from Table 6 we can observe that the relative error between the eventually complex solution and the real part of the solution is in the range of machine precision.

Another trick to save computation time is, that for real matrices all complex eigenvalues exist in pairs. Computing $LU = (A + S_{ii}I)$ we get the factorization of $(A + S_{i+1,i+1}I) = (A + \overline{S_{ii}}I) = \overline{LU}$ for free. A complete real variant is not possible at the moment because we require that there are no blocks on the diagonal of the Schur matrix. Using 2×2 blocks on the diagonal of S (or T) will destroy our linear solvers and conflicts with our initial conditions.

5 Conclusions

We have seen in the previous sections that it is possible to apply $\mathcal{H}_2$-model order reduction to large scale systems. Even problems of order larger than 100 000 can be solved in less than a day. The TSIA algorithm works for MIMO systems and is theoretically the same as the IRKA algorithm [12] in the SISO case. We have seen that there exists a model problem which makes the IRKA algorithm instable which is not the case for the corresponding TSIA algorithm shown in Section 4. In this way we see the TSIA algorithm as a more robust variant of the SISO-IRKA algorithm. The connection to the MIMO variant of the IRKA algorithm, called MIRIAM [17], is similar.

A new interpretation of the oblique projector condition $W^TV = I$ for the projection matrices lead us to a new robust technique to fulfill this.

We have extended the TSIA algorithm to generalized state space systems. Although our method is restricted to problems with a symmetric and positive definite matrix E, this will in most situations be no practical limitation as E often is a mass matrix. As the solution of the Sylvester equation is the most expensive operation in every iteration step, we have developed a family of Sylvester solvers that can handle the special sparse-dense structure of the matrices extending the idea of Sorensen and Antoulas for the standard Sylvester equation [23].

The $\mathcal{H}_2$-model order reduction is now at a point where it can be used for large scale problems. The special features of modern computers can be exploited to accelerate such algorithms. Special solvers for sparse matrices enable us to solve such problems without using supercomputers.

References

[1] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, SIAM Publications, Philadelphia, PA, 2005. 1, 4, 10

[2] R. H. Bartels and G. W. Stewart, *Solution of the matrix equation $AX + XB = C$*, Communications of the ACM, 15 (1972), pp. 820–826. 6, 11, 12

[3] P. Benner and J. Saak, *A semi-discretized heat transfer model for optimal cooling of steel profiles*, 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. 353–356. 15

[4] D. Bernstein and D. Hyland, *Numerical solution of the optimal model reduction equations*, in Proc. AIAA Dynamics Specialists Conf., 1984. 0, 2, 8

[5] A. Bunse-Gerstner, D. Kubalińska, G. Vossen, and D. Wilczek, *h_2-norm optimal model reduction for large scale discrete dynamical MIMO systems*, J. Comput. Appl. Math., 233 (2010), pp. 1202–1216.

[6] T. A. Davis, *Algorithm 832: UMFPACK V4.3—An unsymmetric-pattern multifrontal method*, ACM Trans. Math. Softw., 30 (2004), pp. 196–199. 12

[7] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, *A supernodal approach to sparse partial pivoting*, SIAM J. Matrix Analysis and Applications, 20 (1999), pp. 720–755. 12

[8] J. W. Demmel, J. R. Gilbert, and X. S. Li, *An Asynchronous Parallel Supernodal Algorithm for Sparse Gaussian Elimination*, SIAM J. Matrix Analysis and Applications, 20 (1999), pp. 915–952. 12

[9] J. Gardiner, A. Laub, J. Amato, and C. Moler, *Solution of the Sylvester matrix equation $AXB+CXD=E$*, ACM Trans. Math. Software, 18 (1992), pp. 223–231. 10

[10] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, third ed., 1996. 8, 11, 13

[11] G. H. Golub, S. Nash, and C. F. Van Loan, *A Hessenberg–Schur method for the problem $AX+XB=C$*, IEEE Trans. Automat. Control, AC-24 (1979), pp. 909–913. 11, 12

[12] S. Gugercin, A. C. Antoulas, and C. Beattie, *$\mathcal{H}_2$ Model Reduction for large-scale dynamical systems*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 609–638. 0, 1, 2, 4, 8, 21

[13] S. Hammarling, *Numerical solution of the stable, non-negative definite Lyapunov equation*, IMA J. Numer. Anal., 2 (1982), pp. 303–323. 6

[14] M. Heyouni, *Extended Arnoldi methods for large low-rank Sylvester matrix equations*, Applied Numerical Mathematics, 60 (2010), pp. 1171 – 1182. Special Issue: 9th IMACS International Symposium on Iterative Methods in Scientific Computing (IISIMSC 2008). 6

[15] D. Hohlfeld and H. Zappe, *An all-dielectric tunable optical filter based on the thermo-optic effect*, Journal of Optics A: Pure and Applied Optics, 6 (2004), pp. 504–511. 15

[16] M. Köhler and J. Saak, *Efficiency improving implementation techniques for large scale matrix equation solvers*, Chemnitz Scientific Computing Prep. 09-10, TU Chemnitz, 2009. Available from `http://nbn-resolving.de/urn:nbn:de:bsz:ch1-201000843`. 10, 19

[17] D. Kubalinska, *Optimal interpolation-based model reduction*, PhD thesis, Universität Bremen, 2008. 9, 21

[18] B. C. Moore, *Principal component analysis in linear systems: controllability, observability, and model reduction*, IEEE Trans. Automat. Control, AC-26 (1981), pp. 17–32. 1

[19] T. Penzl, *Numerische Lösung großer Lyapunov-Gleichungen*, Logos–Verlag, Berlin, Germany, 1998. Dissertation, Fakultät für Mathematik, TU Chemnitz, 1998. 6, 15

[20] ———, Lyapack *Users Guide*, Tech. Rep. SFB393/00-33, Sonderforschungsbereich 393 *Numerische Simulation auf massiv parallelen Rechnern*, TU Chemnitz, 09107 Chemnitz, Germany, 2000. Available from `http://www.tu-chemnitz.de/sfb393/sfb00pr.html`. 15, 17

[21] J. Saak, *Efficient Numerical Solution of Large Scale Algebraic Matrix Equations in PDE Control and Model Order Reduction*, PhD thesis, TU Chemnitz, July 2009. Available from `http://nbn-resolving.de/urn:nbn:de:bsz:ch1-200901642`. 6

[22] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288. 6

[23] D. C. Sorensen and A. C. Antoulas, *The Sylvester equation and approximate balanced reduction*, Linear Algebra and its Applications, 351-352 (2002), pp. 671 – 700. 10, 21

[24] D. C. Sorensen and Y. Zhou, *Direct methods for matrix Sylvester and Lyapunov equations*, J. Appl. Math, 2003 (2002), p. 2003. 6

[25] P. Van Dooren, K. A. Gallivan, and P. Absil, *$\mathcal{H}_2$-optimal approximation of MIMO linear dynamical systems*, Applied Mathematics Letters, 21 (2008), p. 1276—1273. 0, 4

[26] D. Wilson, *Optimum solution of model-reduction problem*, Electrical Engineers, Proceedings of the Institution of, 117 (1970), pp. 1161 –1165. 0, 2, 3, 4

[27] Y. Xu and T. Zeng, *Optimal $\mathcal{H}_2$ model reduction for large scale MIMO systems via tangential interpolation*, International Journal of Numerical Analysis and Modeling, 8 (2011), pp. 174–188. 0, 2, 4