Methods

Robert Jendersie and Steffen W. R. Werner*

A comparison of numerical methods for model reduction of dense discrete-time systems

Ein Vergleich numerischer Methoden zur Modellreduktion von dichtbesetzten, zeitdiskreten Systemen

https://doi.org/10.1515/auto-2021-0035
Received February 2, 2021; accepted June 29, 2021

Abstract: Discrete-time systems are a common tool in the modeling of processes in many application areas such as digital signal processing and population dynamics. Model reduction is an essential remedy to handle high-fidelity systems in practice. To benefit from the performance gained by using reduced-order models, the computation of these models itself must be done with a reasonable use of resources. In this paper, we consider the case of medium-scale dense discrete-time systems and compare the performance of different numerical methods for the implementation of two basic model reduction techniques. Therefore, we give an overview of the considered model reduction methods and of the techniques used in underlying implementations. The outlined methods are then compared with established implementations in several numerical examples in terms of accuracy and performance.

Keywords: model order reduction, discrete-time systems, balanced truncation, LQG balanced truncation, matrix equations

Zusammenfassung: Zeitdiskrete Systeme sind ein typisches Werkzeug zur Modellierung von Prozessen in vielen Anwendungsbereichen wie z.B. in der digitalen Signalverarbeitung oder in Populationsdynamiken. Modellreduktion ist ein wesentliches Mittel zur Handhabung von hochgenauen Systemen in der praktischen Anwendung. Um aber von der zusätzlichen Performance durch die Nutzung von reduzierten Modellen zu profitieren, ist es nötig diese unter sinnvollem Einsatz von Ressourcen zu berechnen. In diesem Beitrag betrachten wir den Fall von mittelgroßen, dichtbesetzten, zeitdiskreten Systemen und vergleichen die Performance von verschiedenen, numerischen Methoden für die Implementierung zweier grundlegender Modellreduktionsverfahren. Dafür geben wir einen Überblick zu den betrachteten Modellreduktionsmethoden und den Techniken für die darunterliegenden Implementierungen. Die dargestellten Methoden werden dann mit etablierten Implementierungen in mehreren numerischen Beispielen in Bezug auf Genauigkeit und Performance verglichen.

Schlagwörter: Modellreduktion, zeitdiskrete Systeme, balanciertes Abschneiden, LQG balanciertes Abschneiden, Matrixgleichungen

1 Introduction

The modeling of real-world applications and processes commonly results in dynamical systems used for simulations and controller design. Discrete-time systems use difference equations to describe the dynamics at discrete points in time and are used in various application areas such as electromechanics, traffic control and general digital signal processing [23, 28], or population dynamics and medical processes [21]. Also, in the context of system identification and data-driven modeling, discrete-time systems are the natural tool of choice due to the use of discretized simulation and sensor data [29, 30, 37].

In this paper, we will consider standard discrete-time systems of the form

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k, \\ y_k &= Cx_k + Du_k, \end{aligned} \tag{1}$$

with the initial state $x_0 = 0$, and constant matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ and $D \in \mathbb{R}^{p\times m}$. In (1), the discrete control signal $u_k \in \mathbb{R}^m$ is used to influence the internal states $x_k \in \mathbb{R}^n$ to get the desired outputs $y_k \in \mathbb{R}^p$, with

***Corresponding author: Steffen W. R. Werner,** Computational Methods in Systems and Control Theory, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstraße 1, 39106 Magdeburg, Germany, e-mail: werner@mpi-magdeburg.mpg.de
Robert Jendersie, Faculty of Computer Science, Otto von Guericke University, Universitätsplatz 2, 39106 Magdeburg, Germany, e-mail: robert.jendersie@ovgu.de

the time steps $k \in \mathbb{N} \cup \{0\}$. Using the Z-transformation, the system (1) can be rewritten in the frequency domain for a direct input-to-output relation using its transfer function

$$H(z) = C(zI_n - A)^{-1}B + D,$$

with the complex variable $z \in \mathbb{C}$.

With the demand for increasing accuracy of models, the number n of difference equations describing (1) quickly increases, which consequently yields a high demand for computational resources in terms of time, memory and energy to apply (1) in simulations and controller design. A remedy can be seen in model order reduction, which aims for the approximation of the input-to-output behavior of (1) by an easy-to-evaluate surrogate system

$$\begin{aligned} \hat{x}_{k+1} &= \widehat{A}\hat{x}_k + \widehat{B}u_k, \\ \hat{y}_k &= \widehat{C}\hat{x}_k + \widehat{D}u_k, \end{aligned} \tag{2}$$

with reduced-order matrices $\widehat{A} \in \mathbb{R}^{r\times r}$, $\widehat{B} \in \mathbb{R}^{r\times m}$, $\widehat{C} \in \mathbb{R}^{p\times r}$ and $\widehat{D} \in \mathbb{R}^{p\times m}$, and $r \ll n$.

In practice, not only the evaluation of (2) needs to be fast, but also the construction of the reduced-order model (ROM) has to be done in a reasonable amount of time. Therefore, it is necessary to choose well performing numerical methods in the underlying implementation of the model reduction approaches. In this paper, we will compare different numerical methods for the implementation of balancing-related model reduction in terms of performance and accuracy, and outline in several numerical examples the possible performance gains in the computations of matrix equations and reduced-order models. For a concise presentation of the results, we will introduce the classical and LQG balanced truncation methods, and give a short survey on the considered matrix equation solvers. This paper is restricted to the case of medium-scale dense systems, arising, for example, in system identification and data-driven modeling [29, 30, 37]. In case of large-scale sparse systems, it is possible to first use a pre-reduction step [31], which yields a high-fidelity medium-scale dense approximation that can be then further reduced by the approaches presented in this paper.

In Section 2, we quickly recap two balancing-related model reduction methods for discrete-time systems, followed by a discussion of numerical methods in Section 3 that can be used to compute the reduced-order models. Section 4 then contains comparisons of the discussed matrix equation solvers and model reduction methods with reference implementations in MATLAB. The paper is concluded in Section 5.

2 Model reduction methods

In this section, we briefly describe two known balancing-related model reduction methods for discrete-time systems. There are further model reduction approaches known for discrete-time systems such as moment matching [1] and modal truncation [40]. These will not be treated in this paper, but implementations for dense systems can make use of the same numerical methods mentioned in Section 3.

2.1 Balanced truncation

A first model reduction approach is given by the balanced truncation method. Originated in [32] for the continuous-time case, the approach can be extended to discrete-time systems; see, e. g., [1, 13, 23]. The idea is to consider the discrete-time variants of the system's controllability and observability Gramians, to balance the system with respect to these and to truncate the states, which are hard to control and observe. For asymptotically stable discrete-time systems (spectral radius $\rho(A) < 1$), the discrete-time controllability and observability Gramians $P, Q \in \mathbb{R}^{n\times n}$ are given as the unique symmetric positive semi-definite solutions of the discrete-time Lyapunov equations

$$APA^\mathsf{T} - P + BB^\mathsf{T} = 0, \tag{3}$$

$$A^\mathsf{T}QA - Q + C^\mathsf{T}C = 0. \tag{4}$$

The positive square roots of the eigenvalues of PQ are the *Hankel singular values*, which quantify the influence of the corresponding states to the input-to-output behavior of the system, i. e., for a good approximation we truncate states that correspond to small Hankel singular values. In fact, the balanced truncation method provides an a priori error bound in the h_∞-norm, which only relies on the truncated Hankel singular values

$$\|H - \widehat{H}\|_{h_\infty} \leq 2 \sum_{k=r+1}^{n} \sigma_k.$$

Performing a state-space transformation of the system to balance the Gramians allows to assign states to Hankel singular values and, consequently, to truncate the states corresponding to the smallest Hankel singular values. This whole procedure is summarized in Algorithm 1 using the square root method for balancing and truncation in a single step. See [1] for a more detailed derivation of the balanced truncation method for discrete-time systems.

Algorithm 1: Balanced truncation square root method.

Input: System matrices A, B, C, D from (1).
Output: ROM matrices $\widehat{A}$, $\widehat{B}$, $\widehat{C}$, $\widehat{D}$ for (2).

1 Solve (3) and (4) for the Cholesky factorizations $P = RR^\mathsf{T}$ and $Q = LL^\mathsf{T}$.
2 Compute the singular value decomposition

$$L^\mathsf{T}R = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1^\mathsf{T} \\ V_2^\mathsf{T} \end{bmatrix},$$

with Σ_1 containing the r largest Hankel singular values and U_1, V_1 partitioned accordingly.
3 Compute the truncation matrices

$$T = RV_1\Sigma_1^{-\frac{1}{2}}, \quad W = LU_1\Sigma_1^{-\frac{1}{2}}.$$

4 Compute the reduced-order matrices as

$$\widehat{A} = W^\mathsf{T}AT, \quad \widehat{B} = W^\mathsf{T}B, \quad \widehat{C} = CT, \quad \widehat{D} = D.$$

Note that not all system are asymptotically stable in practice. Unstable systems can occur throughout the different application areas [23, 21, 30], or can even be modeled on purpose to improve the performance of constructed controllers. Due to their unstable behavior, these systems are more difficult to handle in simulations and controller design, as well as in model reduction. However, the balanced truncation method can also be applied to unstable discrete-time systems by first performing an additive decomposition of the system, to decouple the stable and unstable parts (see Section 3.1). Then, Algorithm 1 is only used on the stable system part and the results are coupled together with the unstable system part for the final reduced-order model.

The *inverse balanced truncation*, *Hankel-norm approximation* or the *singular perturbation approximation* are further model reduction methods for discrete-time systems known in the literature; see, e. g., [13, 23]. Those will be omitted here since they are essentially refinements of the balanced truncation method from this section.

2.2 LQG balanced truncation

The linear-quadratic Gaussian (LQG) balanced truncation method belongs to the class of balancing-related model reduction methods, i. e., it follows the theory of balanced truncation but replaces the system Gramians by other symmetric positive semi-definite matrices. The method was proposed in [27] for the continuous-time case as natural extension of balanced truncation to unstable systems. Note that LQG balanced truncation does not need an additive decomposition of the system to treat the unstable part. The discrete-time case was then considered in [26, 33]. In this approach, the Gramians in (3) and (4) are replaced by the unique stabilizing solutions P_{LQG} and Q_{LQG} of the discrete-time algebraic filter and regulator Riccati equations

$$\begin{aligned} &AP_{\text{LQG}}A^\mathsf{T} - P_{\text{LQG}} + BB^\mathsf{T} \\ &\quad - (AP_{\text{LQG}}C^\mathsf{T} + BD^\mathsf{T})(I_p + DD^\mathsf{T} + CP_{\text{LQG}}C^\mathsf{T})^{-1} \\ &\quad \times (AP_{\text{LQG}}C^\mathsf{T} + BD^\mathsf{T})^\mathsf{T} = 0, \end{aligned} \tag{5}$$

$$\begin{aligned} &A^\mathsf{T}Q_{\text{LQG}}A - Q_{\text{LQG}} + C^\mathsf{T}C \\ &\quad - (B^\mathsf{T}Q_{\text{LQG}}A + D^\mathsf{T}C)^\mathsf{T}(I_m + D^\mathsf{T}D + B^\mathsf{T}Q_{\text{LQG}}B)^{-1} \\ &\quad \times (B^\mathsf{T}Q_{\text{LQG}}A + D^\mathsf{T}C) = 0. \end{aligned} \tag{6}$$

The rest of the method then reads like the classical balanced truncation approach. The method is wrapped up in Algorithm 2.

Algorithm 2: LQG balanced truncation square root method.

Input: System matrices A, B, C, D from (1).
Output: ROM matrices $\widehat{A}$, $\widehat{B}$, $\widehat{C}$, $\widehat{D}$ for (2).

1 Solve (5) and (6) for the Cholesky factorizations $P_{\text{LQG}} = RR^\mathsf{T}$ and $Q_{\text{LQG}} = LL^\mathsf{T}$.
2 Follow Steps 2–4 of Algorithm 1.

3 Numerical methods

The main computational work for both introduced (and further) model reduction methods relies on the solution of matrix equations and the decomposition of transfer functions. In the following, we describe numerical methods to solve these problems in the dense system case. The restriction to the dense case enables the use of transformations of the describing matrices. This is not possible in the sparse system case as it would destroy the sparsity structure. While it is possible to compute a dense prereduction [31] and afterwards apply the methods described here, there are also computational approaches to directly handle sparse systems using different matrix operations; see, e. g., [15] for methods for sparse continuous-time systems.

3.1 Additive decomposition of transfer functions

We will first discuss the additive decomposition of transfer functions in the fashion of

$$H(z) = H_1(z) + H_2(z). \tag{7}$$

This can be used, for example, for a stable-unstable decomposition of (1) such that model reduction methods that can only be used on stable systems can still be applied (e. g., Section 2.1). Such a decomposition (7) can simply be achieved by block diagonalizing the system matrix A, i. e., we need to find a transformation $T \in \mathbb{R}^{n\times n}$ such that

$$T^{-1}AT = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}, \tag{8}$$

where $A_1 \in \mathbb{R}^{n_1\times n_1}$ contains a (desired) part of the eigenvalues of A and $A_2 \in \mathbb{R}^{(n-n_1)\times(n-n_1)}$ the rest. Also applying T^{-1} and T to the input and output matrices

$$T^{-1}B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \quad \text{and} \quad CT = \begin{bmatrix} C_1 & C_2 \end{bmatrix},$$

yields the additive decomposition (7).

The most intuitive way to directly achieve (8) would be to do an eigenvalue decomposition of A and choose T as eigenvector basis. In that case, A_1 and A_2 are nearly diagonal (up to Jordan blocks). A disadvantage of this approach is that computing a full eigendecomposition can be fairly expensive and also the eigenvector basis T can easily become ill conditioned; see, e. g., [22]. This can be avoided by allowing A_1 and A_2 to be non-diagonal matrices.

A different way to compute (8) follows a two-step procedure. First, an orthogonal basis $V \in \mathbb{R}^{n\times n}$ is computed to transform A into block triangular shape, e. g., by the Schur decomposition or invariant subspaces [3, 7, 22, 36], such that

$$V^{\mathsf{T}}AV = \begin{bmatrix} A_1 & A_X \\ 0 & A_2 \end{bmatrix}. \tag{9}$$

In the second step, the matrix $A_X \in \mathbb{R}^{n_1\times(n-n_1)}$ needs to be eliminated by another transformation $W \in \mathbb{R}^{n\times n}$. This is generally done by solving the continuous-time Sylvester equation

$$A_1X - XA_2 + A_X = 0, \tag{10}$$

for $X \in \mathbb{R}^{n_1\times(n-n_1)}$, to set the desired transformation to be

$$W = \begin{bmatrix} I_{n_1} & X \\ 0 & I_{n-n_1} \end{bmatrix}, \quad \text{and} \quad W^{-1} = \begin{bmatrix} I_{n_1} & -X \\ 0 & I_{n-n_1} \end{bmatrix}.$$

Since in the discrete-time case the eigenvalues are considered with respect to the unit circle instead of the imaginary axis, the matrices A_1 and A_2 are in general not Hurwitz, which can lead to problems with iterative Sylvester equation solvers, which rely on that property, e. g., in the sign function iteration [8]. Instead, the scaled discrete-time Sylvester equation (Stein equation) can be solved. Choose an $\alpha > 0$ that separates the two spectra such that A_1 contains all the eigenvalues with absolute value larger than α and A_2 the eigenvalues with absolute value smaller than α. Then, instead of solving (10), X is given as the solution of

$$\left(\alpha A_1^{-1}\right) X \left(\frac{1}{\alpha}A_2\right) - X - A_1^{-1}A_X = 0. \tag{11}$$

With the scaling by α, the two coefficient matrices are stable in the discrete-time sense since $\rho\left(\alpha A_1^{-1}\right) < 1$ and $\rho\left(\frac{1}{\alpha}A_2\right) < 1$. By the separation of the spectra, A_1 has only eigenvalues that are larger in absolute value than α and therefore, A_1^{-1} is guaranteed to exist. A typical choice for the eigenvalue bound for a stable-anti-stable decomposition of the system is $\alpha = 1$ or, in case of eigenvalues on the unit circle, to make α a bit smaller than 1. The whole procedure is summarized in Algorithm 3.

In principle, the method in Algorithm 3 can be used to implement a *modal truncation method* in the discrete-time setting based on an eigenvalue decomposition similar to the method from the continuous-time case [19]. We tested this approach on the examples in Section 4 by selecting either the eigenvalues with smallest or largest absolute value, which did not result in any reasonable reduced-order models. An implementation of modal truncation for testing can be found in [16]. It is mentioned in [40] that a suitable selection of eigenvalues may yield better reduction results. However, this will not be part of this paper.

3.2 Lyapunov equation solvers

In both the additive decomposition and balanced truncation method, linear matrix equations need to be solved, namely discrete-time Lyapunov and Sylvester equations. For brevity, we will only mention basic ideas for the solution of Lyapunov equations of the form

$$AXA^{\mathsf{T}} - X + G = 0, \tag{12}$$

with $\rho(A) < 1$ and general symmetric right-hand side $G = G^{\mathsf{T}} \in \mathbb{R}^{n\times n}$, but note that similar ideas and algorithms exist for discrete-time Sylvester equations [38].

Algorithm 3: Additive decomposition of discrete-time transfer functions.

Input: System matrices A, B, C, D from (1), eigenvalue bound $\alpha > 0$.

Output: Decomposed system matrices A_1, B_1, C_1 and A_2, B_2, C_2, D_2 for (7).

1 Compute an orthogonal matrix $V = \begin{bmatrix} V_1 & V_2 \end{bmatrix}$ such that (9) holds, with accordingly partitioned V_1 and V_2, and with A_1 containing the eigenvalues with absolute value larger than α and A_2 the eigenvalues smaller than α.

2 Solve (11) for $X \in \mathbb{R}^{n_1 \times (n-n_1)}$.

3 Compute the subsystem matrices with eigenvalues larger than α

$$A_1 = V_1^\mathsf{T} A V_1, \quad B_1 = (V_1^\mathsf{T} - X V_2^\mathsf{T})B, \quad C_1 = C V_1,$$

and the subsystem matrices with eigenvalues smaller than α

$$A_2 = V_2^\mathsf{T} A V_2, \quad B_2 = V_2^\mathsf{T} B, \quad C_2 = C(V_1 X + V_2), \quad D_2 = D.$$

3.2.1 Smith iteration methods

First, we consider the *Smith iteration methods*. Those approaches are all based on the observation that the solution of (12) is given as matrix series

$$X = \sum_{k=0}^{\infty} A^k G (A^\mathsf{T})^k.$$

This directly yields the original Smith iteration [39], as the iterates are set to be

$$X_{k+1} = G + A X_k A^\mathsf{T}, \qquad \text{with } X_0 = G, \tag{13}$$

such that $X_k \to X$ for $k \to \infty$. The iteration can then be stopped if the residual of (12) becomes small enough or the iterates do not change anymore. A strong disadvantage of (13) is its usually very slow convergence, which was attempted to be solved in different extensions.

The *Smith(ℓ) method* (or *cyclic Smith method*) was introduced in [35]. This method takes inspiration from the cyclic approach of the alternating directions implicit (ADI) method [41] to speed up the convergence of the Smith iteration. Choose an $\ell \in \mathbb{N}$ and set

$$A_\ell = A^\ell \quad \text{and} \quad G_\ell = \sum_{j=0}^{\ell-1} A^j G (A^\mathsf{T})^j. \tag{14}$$

Then, the Smith(ℓ) iteration is given by

$$X_{k+1} = G_\ell + A_\ell X_k A_\ell^\mathsf{T}, \qquad \text{with } X_0 = G_\ell. \tag{15}$$

It can easily be seen that (15) resembles (13) for $\ell = 1$. The iteration in (15) is independent of the chosen ℓ and is known to improve in convergence behavior for increasing ℓ; see [42]. This leads to an improved runtime of the algorithm as long as the costs for the pre-computation of (14) do not outweigh the actual improvement of the convergence speed.

The *squared Smith method*, also known as *Smith accelerative iteration*, yields a significantly reduced number of iteration steps by using updates on the system matrix A in every iteration step. The iteration reads as

$$\begin{aligned} X_{k+1} &= X_k + A_k X_k A_k^\mathsf{T}, && \text{with } X_0 = G, \\ A_{k+1} &= A_k A_k, && \text{with } A_0 = A. \end{aligned} \tag{16}$$

The iteration (16) can be stopped for $A_k X_k A_k^\mathsf{T}$ becoming small enough as this update would not change the solution anymore. Further extensions in the fashion of (16) are known as the *r-Smith iterations* from [42], which use r-order powers for the update of A_k. Those methods yield an even faster convergence than the squared Smith method but usually do not pay off due to the significantly increasing computational costs.

In the context of balanced truncation, further improvements of the squared Smith iteration are possible. Taking a closer look at (3) and (4), it turns out that the computations for updating the A_k matrix in (16) are in fact the same for both Lyapunov equations. This can be exploited in a dual-type squared Smith iteration that solves (3) and (4) at the same time but only computes the updates for A_k ones per iteration step. This approach saves one matrix-matrix product per iteration step.

In Algorithm 1, only the Cholesky factors of the solutions to the Lyapunov equations are needed. Also observing that those factors are usually of small rank due to the low-rank right-hand sides of the Lyapunov equations, it makes sense to directly compute those low-rank Cholesky factors instead of the full solution followed by a Cholesky-like decomposition. The basic idea is to enforce the same factorization in the solution as given by the right-hand side, e. g., $X_k = R_k R_k^\mathsf{T}$ for (3). While this idea can be

used in all the Smith-type methods from above, we only consider the squared Smith iteration (16) for brevity. Setting $X_0 = R_0 R_0^\mathsf{T} = BB^\mathsf{T}$ gives the iteration

$$R_{k+1}R_{k+1}^\mathsf{T} = R_k R_k^\mathsf{T} + A_k R_k R_k^\mathsf{T} A_k^\mathsf{T},$$

which yields the iteration on the solution factor to be

$$R_{k+1} = \begin{bmatrix} R_k & A_k R_k \end{bmatrix}. \tag{17}$$

The number of columns of the solution factor in (17) doubles in every step. Performing a column compression of R_{k+1}, e. g., via a rank-revealing QR decomposition or singular value decomposition, reduces the potential memory costs as well as the operations that need to be performed in the next iteration step [14]. Combining the factorized approach with the dual squared Smith iteration solver is the recommended strategy for an efficient implementation of the balanced truncation method in [13].

3.2.2 Sign function iteration

The sign function iteration method has been shown to be an efficient numerical method for solving stable continuous-time Lyapunov equations

$$\widetilde{A}X + X\widetilde{A}^\mathsf{T} + \widetilde{G} = 0; \tag{18}$$

see, e. g., [10]. To compute the solution of (12) with this method, the discrete-time Lyapunov equation needs to be transformed into (18), e. g., by setting

$$\widetilde{A} = (A + I_n)^{-1}(A - I_n),$$
$$\widetilde{G} = 2(A + I_n)^{-1} G (A + I_n)^{-\mathsf{T}};$$

see, e. g., [38]. Then, the solutions of (12) and (18) are the same. For discrete-time stable A, the inverse $(A + I_n)^{-1}$ always exists and the eigenvalues of $\widetilde{A}$ lie in the left open half-plane. Similarly to the Smith iteration, one can construct dual and factorized sign function solvers to directly compute low-rank factors of the solutions to the dual continuous-time Lyapunov equations [10]. Using these methods for the discrete-time Gramian factors in (3) and (4) also requires the transformation of the right-hand side factors, e. g., in case of (3) to use

$$\widetilde{B} = \sqrt{2}(A + I_n)^{-1} B.$$

3.2.3 Direct solvers

A well-known approach to solve Lyapunov equations directly is the Bartels-Steward algorithm [6], which utilizes the Schur decomposition of the system matrix A. The method has been adapted for the discrete-time case in [5] and has been modified to compute only the Cholesky factor of the solution in Hammarling's method [24]. Both variants for discrete-time Lyapunov equations are implemented in the SLICOT library [12] and are available in MATLAB by the Control System Toolbox™ as `dlyap` and `dlyapchol`.

3.3 Riccati equation solvers

The LQGBT method from Section 2.2 requires the solution of discrete-time algebraic Riccati equations of the form

$$A^\mathsf{T}XA - X + C^\mathsf{T}C - A^\mathsf{T}XB(I_m - B^\mathsf{T}XB)^{-1}B^\mathsf{T}XA = 0. \tag{19}$$

It is assumed that (19) has a stabilizing solution X. Here, we will briefly state the main ideas of different solution techniques for (19).

3.3.1 Structure-preserving doubling algorithm

An iterative method to solve (dual) discrete-time Riccati equations (19) is the *structure-preserving doubling algorithm (SDA)*, described in [18]. Setting $G_0 = BB^\mathsf{T}$, $Q_0 = C^\mathsf{T}C$ and $A_0 = A$, the method follows the iteration scheme

$$G_{k+1} = G_k + A_k(I_n + G_k Q_k)^{-1} G_k A_k^\mathsf{T},$$
$$Q_{k+1} = Q_k + A_k^\mathsf{T} Q_k (I_n + G_k Q_k)^{-1} A_k,$$
$$A_{k+1} = A_k (I_n + G_k Q_k)^{-1} A_k.$$

For $k \to \infty$, the matrix Q_k converges to the stabilizing solution of (19), while G_k converges to the stabilizing solution of the dual Riccati equation. A suitable stopping criteria for the SDA method, besides the relative change of the solutions, is to check if $\|A_k\|$ is small enough such that further updates will not change the solutions anymore.

3.3.2 Newton method

A different technique for solving (19) is given by Newton's method. First proposed in [25], the idea is to apply a Newton scheme for finding the zeros of (19) by refining a stabilizing starting guess. This mainly involves the solution of discrete-time Lyapunov equations

$$A_k^\mathsf{T} X_k A_k - X_k + K_k^\mathsf{T} K_k + C^\mathsf{T}C = 0,$$

for X_k in each iteration step, where

$$A_k = A - BK_k,$$
$$K_{k+1} = (I_m + B^\mathsf{T} X_k B)^{-1} B^\mathsf{T} X_k A.$$

Then for $k \to \infty$, X_k converges to the stabilizing solution X provided that the initial K_0 is chosen such that $A_0 = A - BK_0$ is a discrete-time stable matrix [25]. Such a stabilizing initial K_0 can, e. g., be computed using a partial stabilization technique [9]. In cases when A itself is already stable, a simple choice is $K_0 = 0$. However, the choice of the initial guess has in general a huge impact on the convergence behavior as the algorithm will likely need more steps for an initial guess that is far away from the final solution.

3.3.3 Sign function iteration

As in case of Lyapunov equations, there are efficient iterative solution methods for continuous-time Riccati equations. In particular with the structure given in model order reduction problems, namely that B and C^T have a much smaller number of columns than rows, the algorithm in [11] is promising. Therefore, the discrete-time Riccati equation (19) needs to be transformed into continuous-time form. Such a transformation is suggested in [4]. Assume that A has no eigenvalues equal to -1 and set $G = BB^\mathsf{T}$, $Q = C^\mathsf{T}C$, and $F = I_n + A^\mathsf{T} + Q(I_n + A)^{-1}G$, then with the following matrices

$$\begin{aligned}\widetilde{A} &= I_n - 2F^{-\mathsf{T}}\\ \widetilde{G} &= 2(I_n + A)^{-1}GF^{-1},\\ \widetilde{Q} &= 2F^{-1}Q(I_n + A)^{-1},\end{aligned}$$

the stabilizing solution of (19) is also the stabilizing solution of the continuous-time algebraic Riccati equation

$$\widetilde{A}^\mathsf{T}X + X\widetilde{A} - X\widetilde{G}X + \widetilde{Q} = 0. \tag{20}$$

Note that $\widetilde{G}$ and $\widetilde{Q}$ are still symmetric positive semi-definite and can be rewritten into low-rank factors $\widetilde{G} = \widetilde{B}\widetilde{B}^\mathsf{T}$ and $\widetilde{Q} = \widetilde{C}^\mathsf{T}\widetilde{C}$, e. g., by using an eigenvalue decomposition.

3.3.4 Direct solvers

A direct solver for discrete-time algebraic Riccati equations based on generalized eigenvalue problems was developed in [34] and described in [2]. It is available in MATLAB's Control System Toolbox as `dare`.

4 Numerical experiments

The experiments reported here have been executed on a machine with 2 Intel(R) Xeon(R) Silver 4110 CPU processors running at 2.10 GHz and equipped with 192 GB total main memory. The computer is run on CentOS Linux release 7.5.1804 (Core) with MATLAB 9.4.0.813654 (R2018a). For the comparisons, implementations from the Control System Toolbox and Robust Control Toolbox™ are used. The sign function-based matrix equation solvers were taken from the MORLAB toolbox [16]. There, also the squared Smith iteration, SDA and Newton methods from this comparison were implemented afterwards. Note that all non-built-in MATLAB functions used in the comparisons are directly written in plain MATLAB. The timings reported in this paper were obtained by using the `tic`, `toc` commands. The functions are first run up to 4 times to make use of the just-in-time compiler of MATLAB and then the average execution time of up to 16 further runs is taken. In the following sections, relative execution times are given to illustrate the differences in the performance of the methods. However, the actual computation times of the experiments are available at:

doi:10.5281/zenodo.4745518

In the following, the solvers for matrix equations and model reduction methods are compared in several benchmark examples, listed below:

adre The A matrix was obtained from [21] modeling a black bear population in the USA. A single unstable eigenvalue was extracted beforehand using Algorithm 3 and the input and output matrices were generated randomly with a uniform distribution in $[0, 1]$.

heateq From the SLICOT benchmark collection [17], this example models the heat transfer in a 1D rod.

pde The A matrix was taken from the PDE900 example in the Harwell-Boing Collection [20] and additionally scaled by $\frac{1}{\|A\|_1}$. Input and output matrices were generated randomly with a uniform distribution in $[0, 1]$.

rand$*$ These are several random models with state-space dimension $*$ generated with the `drss` function from the Control System Toolbox. The system matrix A was afterwards scaled by 0.99 to ensure discrete-time stability.

In summary, all examples are asymptotically stable, and the dimensions are given in Table 1.

4.1 Lyapunov equation solvers

As a first insight on the different iterative methods for discrete-time Lyapunov equations discussed in Section 3.2, we consider solving (only) the discrete-time Lyapunov equation (3) by the standard Smith method

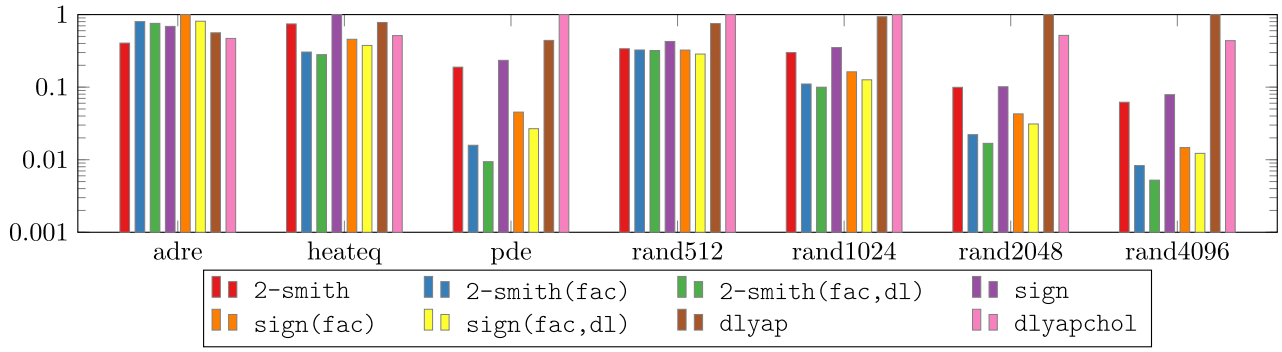

Figure 1: Normalized execution times of different solvers for the dual Lyapunov equations.

Table 1: System dimensions of the benchmark examples.

Model	State-space (n)	Input (m)	Output (p)
adre	68	1	1
heateq	200	1	1
pde	900	1	1
rand512	512	5	4
rand1024	1024	2	4
rand2048	2048	3	3
rand4096	4096	1	2

(`smith`)), the Smith(ℓ) method with $\ell = 4, 8$ (`smith(4)`, `smith(8)`), the squared Smith iteration (`2-smith`) and, for comparison, by the sign function iteration method (`sign`) applied to the transformed continuous-time Lyapunov equation (18). The results in terms of iteration steps that were needed to reach a reasonable residual can be found in Table 2. We can see the nice breakdown of the iteration numbers comparing `smith` with `smith(4)` and `smith(8)`. But it turns out that except in the `pde` example, where the Smith iteration itself converges rapidly, they are not comparable with the squared Smith and sign function iterations. Also, note the slightly worse residuals for `sign` due to the additional transformation that is performed in the beginning.

In the context of model reduction, the solutions to both Lyapunov equations (3) and (4) are needed in factorized form. Considering the results in Table 2, we decided to take only `2-smith` and `sign` into the next comparison. Both methods can be re-formulated as factorized and dual solvers, which we will denote by `2-smith(fac)`, `2-smith(fac,dl)` for the factorized and dual factorized Smith iteration, and `sign(fac)`, `sign(fac,dl)` for the factorized and dual factorized sign function iteration. Those methods are compared with the Bartels-Steward algorithm in `dlyap` and the Hammarling method in `dlyapchol`. In Figure 1, the normalized execution times of the different methods are shown. We see that for small systems, the equation solvers perform all very similarly. For increasing system sizes, the factorized versions of sign function and Smith iteration outperform their unfactorized versions as well as the direct solvers. The values plotted in Figure 1 can also be seen as the inverses of the resulting speed-up factors, e. g., looking at rand4096, the `smith(fac,dl)` performs fastest with the normalized execution time of 0.0052, which is around 190 times faster than `dlyap`.

4.2 Riccati equation solvers

We will do a similar comparison as in the previous section for the solution of the dual Riccati equations (5) and (6). We compare the SDA approach (`sda`), the Newton method (`newton`) with the squared Smith iteration as inner Lyapunov equation solver, the sign function iteration (`sign`) with the transformed Riccati equations (20) and the direct approach using the `dare` function from MATLAB. As the resulting residuals behave similarly for all methods, they are not reported here for brevity. Figure 2 shows the normalized execution times of the tested Riccati equation solvers. In general, the direct approach is the slowest for all examples. Except for the two smallest examples, `newton` is slower than the SDA and sign function iteration, but still gives an immense speed-up for increasing system dimensions compared to the direct approach. Also, the sign function method keeps improving for increasing dimensions and is even slightly faster than `sda` in the largest example, where both methods yield a speed-up factor of around 420 compared to the direct approach.

4.3 Model reduction methods

Lastly, we will compare different implementations of the two model reduction approaches from Section 2. For the

Table 2: Iteration steps and residuals (in brackets) of Smith iteration methods and sign function iteration.

	smith	smith(4)	smith(8)	2-smith	sign
adre	106 (1.49e-13)	25 (1.49e-13)	12 (6.16e-14)	6 (3.95e-14)	8 (1.08e-12)
heateq	2737 (3.13e-13)	683 (3.12e-13)	341 (3.06e-13)	11 (2.02e-16)	9 (4.31e-16)
pde	45 (3.62e-13)	10 (1.87e-13)	4 (1.88e-13)	5 (3.25e-14)	6 (2.57e-13)
rand512	1475 (1.63e-12)	385 (9.57e-13)	196 (8.87e-13)	10 (8.90e-13)	11 (1.13e-10)
rand1024	1443 (2.41e-12)	377 (1.49e-12)	192 (1.35e-12)	10 (1.40e-12)	12 (2.73e-10)
rand2048	1450 (6.14e-12)	378 (3.58e-12)	193 (3.21e-12)	10 (3.36e-12)	12 (7.04e-10)
rand4096	1443 (8.01e-12)	377 (4.76e-12)	192 (4.36e-12)	10 (4.52e-12)	12 (1.75e-09)

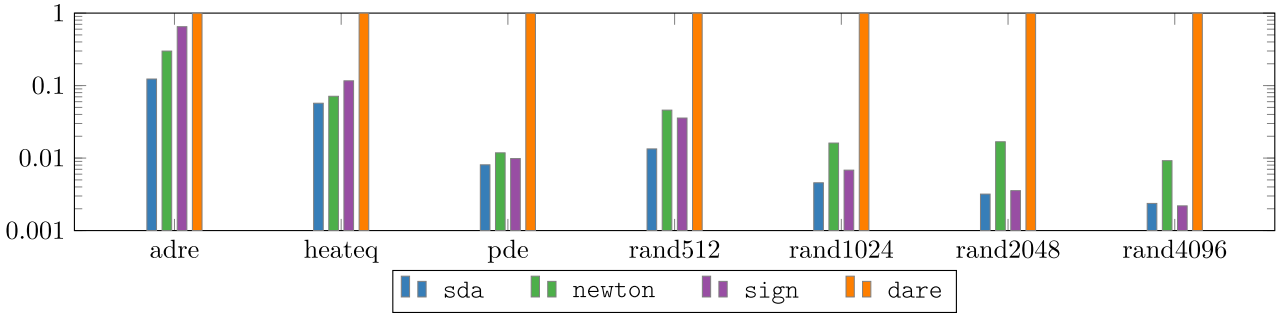

Figure 2: Normalized execution times of different solvers for the dual Riccati equations.

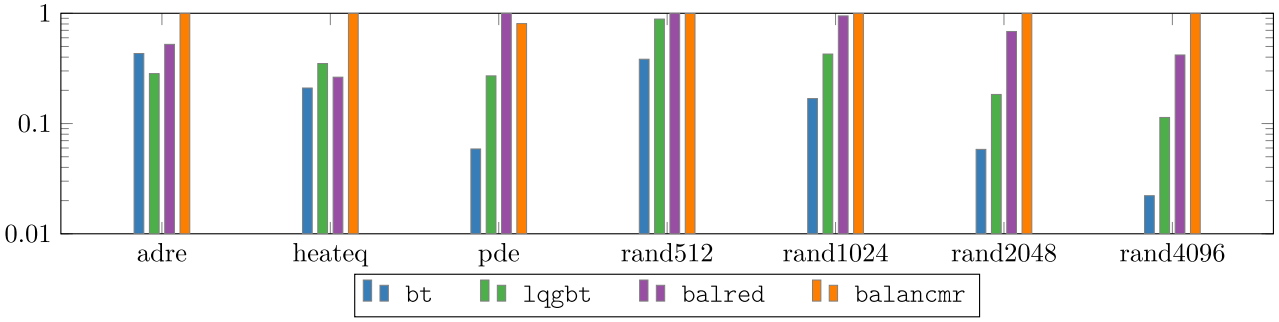

Figure 3: Normalized execution times of different model reduction methods / implementations.

balanced truncation method (`bt`), we will use the squared Smith iteration in dual factorized form and for the LQG balanced truncation (`lqgbt`), we will use the SDA approach. For comparison, we use the `balancmr` and `balred` functions from the Robust Control and Control System Toolboxes. Both functions provide the balanced truncation approach for the discrete-time case and use implementations of the direct matrix equation solvers from Section 3.2.3. In all three implementations of the balanced truncation method, a stable-unstable additive decomposition is performed (see Section 3.1) but has no further effect since all example systems are asymptotically stable. In Figure 3, the normalized execution times of all different test implementations can be seen. Overall, the plain MATLAB implementations of `bt` and `lqgbt` outperform the MATLAB built-in functions. Of particular interest is the performance of `lqgbt`, as Riccati equations need to be solved here. Riccati equations are in general computationally more expensive to solve than Lyapunov equations, since the nonlinearity of the equation needs to be handled, which either results in iterative solutions of Lyapunov equations or the computation of linear spectral problems of double size; cf. Section 3.

For additional insight, we show in Figures 4 and 5 the relative frequency response errors of the computed

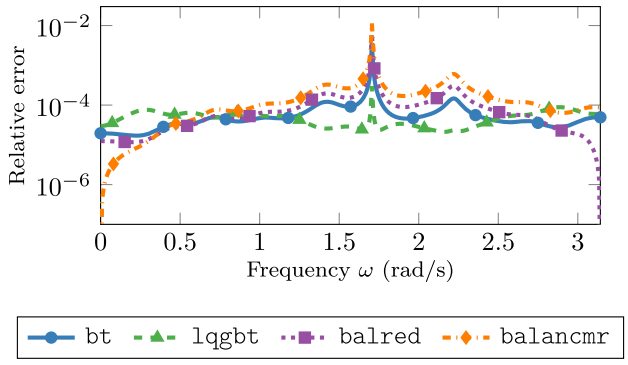

Figure 4: Relative frequency response error for the adre example with reduced order $r = 24$.

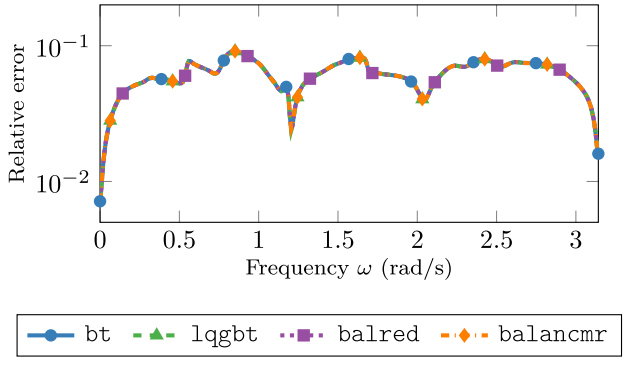

Figure 5: Relative frequency response error for the rand1024 example with reduced order $r = 100$.

reduced-order models for two selected examples from the benchmark collection. The relative error therein was computed by

$$\frac{\|H(e^{i\omega}) - \widehat{H}(e^{i\omega})\|_2}{\|H(e^{i\omega})\|_2},$$

with $\omega \in [0, \pi]$ rad/s. The two plots illustrate that we can produce good and comparable reduced-order approximations while achieving an immense speed-up in computations by using appropriate numerical methods in the underlying implementations. While in Figure 5 the balanced truncation methods with Lyapunov equation look identical, the results look quite different in Figure 4. We assume that this deviation arises from the bad conditioning of the system matrix A in this example. The different underlying implementations result in different numerical errors during the computations, which become visible in this example due to the bad conditioning such that `balancmr` is more accurate in the beginning of the frequency interval, `bt` with our implementation in the middle and `balred` at the end of the interval.

5 Conclusions

In this paper, we presented a numerical comparison of different implementations of two balancing-related model reduction approaches and different matrix equation solvers for medium-scale dense discrete-time systems. We gave an overview of the theory of Lyapunov and LQG balanced truncation and summarized ideas from the literature for the solution of dense Lyapunov and Riccati equations needed in the model reduction methods. We have shown in numerical examples that the iterative matrix equation solvers are comparably fast to the direct methods for small-scale systems ($\lessapprox 500$), but yield a significant performance improvement for increasing system order. In case of the model reduction methods, the iteration-based implementations turned out to be faster than the MATLAB built-in functions even for smaller system sizes, and also showed tremendous speed-ups for the systems of larger order.

We considered only the standard model reduction case of systems with significantly less inputs and outputs than internal states. While the iterative methods can still be efficiently used for systems with many inputs and outputs, it is recommended to use non-factorized versions of the approaches to avoid the performance loss by repeated truncations of the right-hand side iteration matrices.

Funding: The work of Steffen W. R. Werner was supported by the German Research Foundation (DFG) Research Training Group 2297 “MathCoRe”, Magdeburg (Grant No. 314838170).

References

1. A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*, volume 6 of *Adv. Des. Control*. SIAM Publications, Philadelphia, PA, 2005. doi:10.1137/1.9780898718713.
2. W. F. Arnold and A. J. Laub. Generalized eigenproblem algorithms and software for algebraic Riccati equations. *Proc. IEEE*, 72(12):1746–1754, 1984. doi:10.1109/PROC.1984.13083.
3. Z. Bai, J. Demmel and M. Gu. An inverse free parallel spectral divide and conquer algorithm for nonsymmetric eigenproblems. *Numer. Math.*, 76(3):279–308, 1997. doi:10.1007/s002110050264.
4. Y. Bar-Ness and A. Halbersberg. Solution of the singular discrete regulator problem using eigenvector methods. *Internat. J. Control*, 31(4):615–625, 1980. doi:10.1080/00207178008961069.
5. A. Y. Barraud. A numerical algorithm to solve $A^TXA - X = Q$. In *IEEE Conference on Decision and Control including the 16th Symposium on Adaptive Processes and A Special Symposium on Fuzzy Set Theory and Applications*, pages 420–423, 1977. doi:10.1109/CDC.1977.271607.

6. R. H. Bartels and G. W. Stewart. Solution of the matrix equation $AX + XB = C$: Algorithm 432. *Comm. ACM*, 15:820–826, 1972. doi:10.1145/361573.361582.
7. P. Benner. *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*. Dissertation, Fakultät für Mathematik, TU Chemnitz–Zwickau, Chemnitz, Germany, 1997.
8. P. Benner. Factorized solution of Sylvester equations with applications in control. In *Proc. Intl. Symp. Math. Theory Networks and Syst. MTNS 2004*, 2004.
9. P. Benner. Partial stabilization of descriptor systems using spectral projectors. In P. Van Dooren, S. P. Bhattacharyya, R. H. Chan, V. Olshevsky and A. Routray, editors, *Numerical Linear Algebra in Signals, Systems and Control*, volume 80 of *Lect. Notes Electr. Eng.*, pages 55–76. Springer Netherlands, 2011. doi:10.1007/978-94-007-0602-6_3.
10. P. Benner, J. M. Claver and E. S. Quintana-Ortí. Efficient solution of coupled Lyapunov equations via matrix sign function iteration. In *Proc. 3rd Portuguese Conf. on Automatic Control CONTROLO'98*, Coimbra, pages 205–210, 1998.
11. P. Benner, P. Ezzatti, Quintana-Ortí E. S. and A. Remón. A factored variant of the Newton iteration for the solution of algebraic Riccati equations via the matrix sign function. *Numer. Algorithms*, 66(2):363–377, 2014. doi:10.1007/s11075-013-9739-2.
12. P. Benner, V. Mehrmann, V. Sima, S. Van Huffel and A. Varga. SLICOT — A subroutine library in systems and control theory. In B. N. Datta, editor, *Applied and Computational Control, Signals, and Circuits*, volume 1, chapter 10, pages 499–539. Birkhäuser, Boston, MA, 1999. doi:10.1007/978-1-4612-0571-5_10.
13. P. Benner, E. S. Quintana-Ortí and G. Quintana-Ortí. Parallel algorithms for model reduction of discrete-time systems. *Internat. J. Systems Sci.*, 34(5):319–333, 2003. doi:10.1080/0020772031000158564.
14. P. Benner, E. S. Quintana-Ortí and G. Quintana-Ortí. Numerical solution of discrete stable linear matrix equations on multicomputers. *Parallel Algorithms and Appl.*, 17(1):127–146, 2002. doi:10.1080/10637190208941436.
15. P. Benner and J. Saak. Numerical solution of large and sparse continuous time algebraic matrix Riccati and Lyapunov equations: a state of the art survey. *GAMM Mitteilungen*, 36(1):32–52, 2013. doi:10.1002/gamm.201310003.
16. P. Benner and S. W. R. Werner. MORLAB – Model Order Reduction LABoratory (version 5.0), 2019. See also: http://www.mpi-magdeburg.mpg.de/projects/morlab. doi:10.5281/zenodo.3332716.
17. Y. Chahlaoui and P. Van Dooren. A collection of benchmark examples for model reduction of linear time invariant dynamical systems. Technical Report 2002–2, SLICOT Working Note, 2002. Available from www.slicot.org.
18. E. K.-W. Chu, H.-Y. Fan, W.-W. Lin and C.-S. Wang. Structure-preserving algorithms for periodic discrete-time algebraic Riccati equations. *Internat. J. Control*, 77(8):767–788, 2004. doi:10.1080/00207170410001714988.
19. E. J. Davison. A method for simplifying linear dynamic systems. *IEEE Trans. Autom. Control*, AC–11:93–101, 1966. doi:10.1109/TAC.1966.1098264.
20. I. S. Duff, R. G. Grimes and J. G. Lewis. Users' guide for the Harwell-Boeing sparse matrix collection (Release I). Technical Report TR/PA/92/86, CERFACS, Toulouse Cedex, France, 1992.
21. A. H. Freedman, K. M. Portier and M. E. Sunquist. Life history analysis for black bears (Ursus americanus)in a changing demographic landscape. *Ecological Modelling*, 167(1–2):47–64, 2003. doi:10.1016/S0304-3800(03)00171-6.
22. G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, fourth edition, 2013.
23. G. Gu. *Discrete-Time Linear Systems*. Springer, New York Dordrecht Heidelberg London, 2012. doi:10.1007/978-1-4614-2281-5.
24. S. J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA J. Numer. Anal.*, 2:303–323, 1982. doi:10.1093/imanum/2.3.303.
25. G. A. Hewer. An iterative technique for the computation of steady state gains for the discrete optimal regulator. *IEEE Trans. Autom. Control*, 16:382–384, 1971. doi:10.1109/TAC.1971.1099755.
26. J. Hoffmann, D. Prätzel-Wolters and E. Zerz. A balanced canonical form for discrete-time minimal systems using characteristic maps. *Linear Algebra Appl.*, 277(1–3):63–81, 1998. doi:10.1016/S0024-3795(97)10048-9.
27. E. A. Jonckheere and L. M. Silverman. A new set of invariants for linear systems – application to reduced order compensator design. *IEEE Trans. Autom. Control*, 28(10):953–964, 1983. doi:10.1109/TAC.1983.1103159.
28. W. G. Kelley and A. C. Peterson. *Difference Equations: An Introduction with Applications*. Academic Press, San Diego, London, Burlington MA, second edition, 2001.
29. S. Y. Kung. A new identification and model reduction algorithm via singular value decompositions. In *Proc. Twelfth Asilomar Conf. on Circuits, Systems and Computers*, November 6–8, pages 705–714, 1978.
30. J. N. Kutz, S. L. Brunton, B. W. Brunton and J. L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Society of Industrial and Applied Mathematics, Philadelphia, USA, 2016. doi:10.1137/1.9781611974508.
31. M. Lehner and P. Eberhard. A two-step approach for model reduction in flexible multibody dynamics. *Multibody Syst. Dyn.*, 17(2-3):157–176, 2007. doi:10.1007/s11044-007-9039-5.
32. B. C. Moore. Principal component analysis in linear systems: controllability, observability, and model reduction. *IEEE Trans. Autom. Control*, AC–26(1):17–32, 1981. doi:10.1109/TAC.1981.1102568.
33. M. R. Opmeer and F. Curtain. Linear quadratic gaussian balancing for discrete-time infinite-dimensional linear systems. *SIAM J. Control Optim.*, 43(4):1196–1221, 2004. doi:10.1137/S0363012903431189.
34. T. Pappas, A. J. Laub and N. R. Sandell. On the numerical solution of the discrete-time algebraic Riccati equation. *IEEE Trans. Autom. Control*, 25(4):631–641, 1980. doi:10.1109/TAC.1980.1102434.
35. T. Penzl. A cyclic low rank Smith method for large, sparse Lyapunov equations with applications in model reduction and optimal control. Technical Report SFB393/98-6, Fakultät für Mathematik, TU Chemnitz, Chemnitz, Germany, 1998.
36. J. D. Roberts. Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *Internat. J. Control*, 32(4):677–687, 1980. (Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971). doi:10.1080/00207178008922881.
37. P. J. Schmid. Dynamic mode decomposition of numerical

and experimental data. *J. Fluid Mech.*, 656:5–28, 2010. doi:10.1017/S0022112010001217.
38. V. Simoncini. Computational methods for linear matrix equations. *SIAM Rev.*, 38(3):377–441, 2016. doi:10.1137/130912839.
39. R. A. Smith. Matrix equation $XA + BX = C$. *SIAM J. Appl. Math.*, 16(1):198–201, 1968. doi:10.1137/0116017.
40. A. Varga. Enhanced modal approach for model reduction. *Math. Model. Syst.*, 1(2):91–105, 1995. doi:10.1080/13873959508837010.
41. E. L. Wachspress. Iterative solution of the Lyapunov matrix equation. *Appl. Math. Letters*, 107:87–90, 1988. doi:10.1016/0893-9659(88)90183-8.
42. B. Zhou, J. Lam and G.-R. Duan. On Smith-type iterative algorithms for the Stein matrix equation. *Appl. Math. Letters*, 22(7):1038–1044, 2009. doi:10.1016/j.aml.2009.01.012.

Bionotes

Robert Jendersie
Faculty of Computer Science, Otto von Guericke University, Universitätsplatz 2, 39106 Magdeburg, Germany
robert.jendersie@ovgu.de

Robert Jendersie is a student of computer science at the Otto von Guericke University Magdeburg. He finished his bachelor degree in 2019 working on model order reduction during an internship at the Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg. Currently, he is exploring the stability of neural networks in the context of physical simulations.

Steffen W. R. Werner
Computational Methods in Systems and Control Theory, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstraße 1, 39106 Magdeburg, Germany
werner@mpi-magdeburg.mpg.de

Steffen W. R. Werner finished his master studies in mathematics at the Otto von Guericke University Magdeburg in 2016. Thereafter, he joined the Max Planck Institute for Dynamics of Complex Technical Systems, where he is working on the application of structure-preserving model order reduction methods for mechanical systems.