Linear Algebra and its Applications 480 (2015) 44–71

Contents lists available at ScienceDirect

Linear Algebra and its Applications

www.elsevier.com/locate/laa

On the benefits of the LDL^T factorization for large-scale differential matrix equation solvers

Norman Lang [a,*], Hermann Mena [b], Jens Saak [a,c]

[a] *Technische Universität Chemnitz, Faculty of Mathematics, D-09126 Chemnitz, Germany*
[b] *University of Innsbruck, Department of Mathematics, A-6020 Innsbruck, Austria*
[c] *Max Planck Institute for Dynamics of Complex Technical Systems, D-39106 Magdeburg, Germany*

ARTICLE INFO

Article history:
Received 31 July 2014
Accepted 6 April 2015
Available online 26 April 2015
Submitted by D.B. Szyld

MSC:
15A23
65L06
93A15

Keywords:
Large-scale
Matrix differential equations
Low-rank
Riccati equations
Lyapunov equations

ABSTRACT

We propose efficient algorithms for solving large-scale matrix differential equations. In particular, we deal with the differential Riccati equations (DRE) and state the applicability to the differential Lyapunov equations (DLE). We focus on methods, based on standard versions of ordinary differential equations, in the matrix setting. The application of these methods yields algebraic Lyapunov equations (ALEs) with a certain structure to be solved in every step. The alternating direction implicit (ADI) algorithm and Krylov subspace based methods allow to exploit this special structure. However, a direct application of classic low-rank formulations requires the use of complex arithmetic. Using an LDL^T-type decomposition of both, the right hand side and the solution of the equation, we avoid this problem. Thus, the proposed methods are a more practical alternative for large-scale problems arising in applications. Also, they make the application of higher order methods feasible. The numerical results show the better performance of the proposed methods compared to earlier formulations.

* Corresponding author.
E-mail addresses: norman.lang@mathematik.tu-chemnitz.de (N. Lang), hermann.mena@uibk.ac.at (H. Mena), saak@mpi-magdeburg.mpg.de (J. Saak).

http://dx.doi.org/10.1016/j.laa.2015.04.006
0024-3795/

1. Introduction

Differential matrix equations arise in many fields like optimal control, model reduction of linear time-varying (LTV) systems, damping optimization in mechanical systems, control of shear flows and the numerical solution of stochastic differential equations [1–7]. We will mainly focus on solving the differential Riccati equation (DRE). However, all our methods and techniques naturally restrict to the differential Lyapunov equation (DLE) (see comments in Section 2.2). A more detailed explanation and extensive numerical experiments for the DLE will be presented elsewhere to keep the presentation within usual page limits.

The DRE is one of the most deeply studied nonlinear matrix differential equations arising in optimal control, optimal filtering, $\mathbf{H}_\infty$-control of linear-time varying systems, differential games, etc. (see, e.g., [8–11]). In the literature there is a large variety of approaches to compute the solution of small-scale DREs (see, e.g., [12–15]). In this article, we consider the numerical solution of large-scale DREs arising in optimal control problems for partial differential equations. In [16,17] efficient numerical methods capable of exploiting the structure based on matrix-valued versions of the backward differentiation formula (BDF), Midpoint and Trapezoidal rules and the Rosenbrock (Ros) methods are proposed. Moreover, the authors in [18] present an abstract framework based on operator splittings. In contrast to their work we will focus on the matrix setting.

The implementation in [16,19] uses a low-rank Alternating Directions Implicit (ADI) iteration feasible for solving the algebraic Lyapunov equations (ALE) in the inner iteration. Here, we also consider Krylov subspace based methods for the solution of the arising ALEs. When methods of order $p \geq 2$ are applied, complex arithmetic is required, which increases the computational cost. For the Rosenbrock methods an approach has been proposed to keep the computations in real arithmetic [16]. This yields a challenging implementation already for order 2. The ALE arises in many fields like optimal control and model order reduction [20,21]. Many methods for solving large-scale ALEs have been proposed [22–28]. However, there have been no attempts to solve large-scale *differential* Lyapunov equations, which, e.g., arise in Balanced Truncation model order reduction approaches for linear time-varying systems. Discretizing the DLE in time, also an ALE with special structure has to be solved in every step.

In this paper we present novel formulations of solution algorithms for differential matrix equations based on an LDL^T decomposition that keep the computations in real arithmetic. First, we describe how the LDL^T-type splitting can be applied to the BDF schemes and extend these ideas to the Rosenbrock methods. Moreover, the method can, in general, be used in combination with any implicit ODE solver which is applied in the matrix setting. The paper is organized as follows: in Section 2 we review matrix versions of standard methods for stiff problems and their application to DRE and DLE. Further, a column compression technique for complex data is provided. In Section 3, we present the LDL^T based algorithms. Then, in Section 4 we introduce some motivating examples

and test our methods. The numerical results show the performance of the new methods. Finally, some conclusions close the paper in Section 5.

2. Matrix versions of standard ODE integrators

In applications the DREs/DLEs are usually fairly stiff. This, in turn, demands for implicit methods to solve such equations numerically. Therefore, we will focus on matrix versions of standard ODE solvers for (vector valued) stiff problems, [12,14,17]. In order to efficiently exploit the problem structure, we are interested in methods, which, written in matrix form, yield an algebraic Riccati equation (ARE) or an ALE to be solved in each time step when they are applied to the DRE, or DLE. It turns out that there is a vast variety of methods that can be applied, e.g., the Backward Differentiation formulas, the Midpoint, the Trapezoidal rules and the Rosenbrock methods, [19].

2.1. Application to DREs

Let us first consider the time-varying symmetric DREs of the form

$$\begin{aligned} \dot{X}(t) &= -Q(t) - X(t)A(t) - A^T(t)X(t) + X(t)S(t)X(t), \\ X(t_f) &= X_{t_f} \end{aligned} \tag{1}$$

arising in the linear quadratic regulator (LQR) framework for time varying dynamical systems. Here $t \in [t_0, t_f]$ and $Q(t)$, $A(t)$, $S(t) \in \mathbb{R}^{n\times n}$ are assumed to be piecewise continuous locally bounded matrix-valued functions, which ensures the existence and uniqueness of the solution of (1), see [8]. Note that the DRE, originating from an LQR problem replaces the adjoint state from the optimization framework and thus has to be solved backwards in time. Defining $\tilde{X}(t_f; t) := X(t_f - t)$, we can easily reformulate (1) as an initial value problem of the form

$$\begin{aligned} \dot{\tilde{X}}(t) &= Q(t) + \tilde{X}(t)A(t) + A^T(t)\tilde{X}(t) - \tilde{X}(t)S(t)\tilde{X}(t), \\ \tilde{X}(t_0) &= \tilde{X}_0, \end{aligned}$$

since $\dot{\tilde{X}}(t_f; t) = -\dot{X}(t_f - t)$. In the remainder, we neglect the explicit use of the time dependency in the matrices defining the DRE. Furthermore, considering, e.g., finite element semi-discretized partial differential equation constrained optimal control problems one usually faces the generalized DRE

$$\begin{aligned} E^T\dot{X}E &= -Q - E^TXA - A^TXE + E^TXSXE, \\ E^T(t_f)X(t_f)E(t_f) &= E^T(t_f)X_{t_f}E(t_f). \end{aligned} \tag{2}$$

In order to simplify the expressions in the following sections we will focus on the standard case and only state the algorithms in terms of the generalized DRE. The latter can easily

Table 1
Coefficients of the BDF p-step methods up to order $p = 6$.

p	β	α_1	α_2	α_3	α_4	α_5	α_6
1	1	-1					
2	$\frac{2}{3}$	$-\frac{4}{3}$	$\frac{1}{3}$				
3	$\frac{6}{11}$	$-\frac{18}{11}$	$\frac{9}{11}$	$-\frac{2}{11}$			
4	$\frac{12}{25}$	$-\frac{48}{25}$	$\frac{36}{25}$	$-\frac{16}{25}$	$\frac{3}{25}$		
5	$\frac{60}{137}$	$-\frac{300}{137}$	$\frac{300}{137}$	$-\frac{200}{137}$	$\frac{75}{137}$	$-\frac{12}{137}$	
6	$\frac{60}{147}$	$-\frac{360}{147}$	$\frac{450}{147}$	$-\frac{400}{147}$	$\frac{225}{147}$	$-\frac{72}{147}$	$\frac{10}{147}$

be derived by applying the standard theory with $\tilde{A} := E^{-1}A$, $\tilde{B} = E^{-1}B$ and avoiding the inversion of E in the resulting algorithms.

Thus, we will consider

$$\begin{aligned} \dot{X} &= \mathcal{R}(t, X), \\ \mathcal{R}(t, X) &:= Q + XA + A^T X - XSX, \\ X(t_0) &= X_0. \end{aligned} \tag{3}$$

Backward differentiation formulas Applying the fixed-coefficients BDF method to the DRE (3), we obtain the matrix valued BDF scheme

$$X_{k+1} = \sum_{j=1}^{p} -\alpha_j X_{k+1-j} + \tau_k \beta \mathcal{R}(t_{k+1}, X_{k+1}),$$

where τ_k denotes the time step size, $t_{k+1} = t_k + \tau_k$, $X_{k+1} \approx X(t_{k+1})$. The expressions α_j, β denote the determining coefficients for the p-step BDF formula given in Table 1 (see, e.g., [29]). This leads to the Riccati-BDF difference equation

$$\begin{aligned} &-X_{k+1} + \tau_k \beta (Q_{k+1} + A_{k+1}^T X_{k+1} + X_{k+1} A_{k+1} - X_{k+1} S_{k+1} X_{k+1}) \\ &\quad - \sum_{j=1}^{p} \alpha_j X_{k+1-j} = 0 \end{aligned}$$

with $Q_{k+1} \equiv Q(t_{k+1})$, $A_{k+1} \equiv A(t_{k+1})$, $S_{k+1} \equiv S(t_{k+1})$, which can be written as the algebraic Riccati equation

$$\begin{aligned} &(\tau_k \beta Q_{k+1} - \sum_{j=1}^{p} \alpha_j X_{k+1-j}) + (\tau_k \beta A_{k+1} - \frac{1}{2} I)^T X_{k+1} + X_{k+1} (\tau_k \beta A_{k+1} - \frac{1}{2} I) \\ &\quad - X_{k+1} (\tau_k \beta S_{k+1}) X_{k+1} = 0, \end{aligned} \tag{4}$$

for X_{k+1}. For large-scale applications it is necessary to avoid forming the matrices X_k explicitly, because this in general leads to dense computations. In many applications the data is given in a low-rank form

$$\begin{aligned} Q_k &= C_k^T C_k, \qquad & C_k &\in \mathbb{R}^{q\times n}, \\ S_k &= B_k B_k^T, & B_k &\in \mathbb{R}^{n\times m}. \end{aligned} \tag{5}$$

Therefore, in practice one often observes the solution also to be of numerically low rank. That means, using low-rank representation based algorithms to solve (4), the solution can be well approximated by a product of the form $X_k \approx Z_k Z_k^T$ ($Z_k \in \mathbb{R}^{n\times z_k}$, $z_k \ll n$).

In the remainder of this section we review the classical low-rank approximation based formulation. Using the low-rank factors (5), the ARE (4) can be written as

$$\begin{aligned} &\hat{C}_{k+1}^T \hat{C}_{k+1} + \hat{A}_{k+1}^T Z_{k+1} Z_{k+1}^T + Z_{k+1} Z_{k+1}^T \hat{A}_{k+1} \\ &\quad - Z_{k+1} Z_{k+1}^T \hat{B}_{k+1} \hat{B}_{k+1}^T Z_{k+1} Z_{k+1}^T = 0 \end{aligned} \tag{6}$$

with

$$\begin{aligned} \hat{A}_{k+1} &= \tau_k \beta A_{k+1} - \frac{1}{2} I, \\ \hat{B}_{k+1} &= \sqrt{\tau_k \beta} B_{k+1}, \\ \hat{C}_{k+1}^T &= [\, \sqrt{\tau_k \beta} C_{k+1}^T, \sqrt{-\alpha_1} Z_k, \ldots, \sqrt{-\alpha_p} Z_{k+1-p}\,]. \end{aligned}$$

Exploiting the sparsity of A_{k+1}, together with the low-rank representations of the constant and quadratic terms, Eq. (6) can be solved efficiently in terms of computational effort and storage costs, if the rank $z_k \ll n$ for all times. The described formulations above can serve as the basis of a DRE solver for large-scale problems. We note, the main idea here is to solve an ARE by, e.g., Krylov subspace methods, Newton's method or other methods, see e.g., the recent surveys [26,30], in every time step. Here we restrict our selves to these procedures. Applying Newton's method to the ARE (6) results in the solution of the ALE

$$\check{A}_{k+1}^{(\ell)\,T} X_{k+1}^{(\ell)} + X_{k+1}^{(\ell)} \check{A}_{k+1}^{(\ell)} = G_{k+1}^{(\ell)} G_{k+1}^{(\ell)\,T} \tag{7}$$

with $\check{A}_{k+1}^{(\ell)} = (\hat{A}_{k+1} - \tau_k \beta B_{k+1} B_{k+1}^T X_{k+1}^{(\ell-1)})$ and $G_{k+1}^{(\ell)} = [\hat{C}_{k+1}^T,\ \sqrt{\tau_k \beta} X_{k+1}^{(\ell-1)} B_{k+1}]$ for $X_{k+1}^{(\ell)}$ in the ℓ-th Newton step. For implementation details see [16] and the references therein. Note that for methods of order $p \geq 2$ some of the coefficients α_j, $j = 1, \ldots, p$ of the p-step BDF method are positive, see Table 1. This leads to algebraic Lyapunov equations which have indefinite right hand sides and thus the right hand side factor G of the ALE (7) becomes complex. The solution via Newton's method, in particular the application of the inner solver to the ALE (7), needs to deal with complex arithmetic and this, in turn, makes complex storage unavoidable.

Rosenbrock methods The application of the general p-stage Rosenbrock method, as a matrix-valued procedure, to the DRE (3) yields

$$\left(\frac{1}{\tau_k\gamma_{ii}}I - \frac{\partial\mathcal{R}}{\partial X}(t_k, X_k)\right) K_i = \mathcal{R}\left(t_{k,i}, X_k + \sum_{j=1}^{i-1} a_{ij}K_j\right) + \sum_{j=1}^{i-1}\frac{c_{ij}}{\tau_k}K_j + \gamma_i\tau_k\mathcal{R}_{t_k},$$

$$X_{k+1} = X_k + \sum_{j=1}^{p} m_j K_j, \tag{8}$$

where $t_{k,i} = t_k + \alpha_i\tau_k$, $i = 1, \ldots, p$, and γ_{ii}, a_{ij}, c_{ij}, γ_i, m_j and α_i are the method coefficients, that are available in text books as, e.g., [31]. We denote by K_i the $n \times n$ matrix representing the solution of the i-th-stage of the method and abbreviate $\mathcal{R}_{t_k} = \frac{\partial\mathcal{R}}{\partial t}(t_k, X(t_k))$. Note that for autonomous DREs $\mathcal{R}_{t_k} = 0$. Using the Frechét derivative

$$\frac{\partial\mathcal{R}}{\partial X}(t_k, X_k) : U \to (A_k - S_kX_k)^TU + U(A_k - S_kX_k), \tag{9}$$

of $\mathcal{R}$ at X_k with $X_k \approx X(t_k)$, $A_k \equiv A(t_k)$, $S_k \equiv S(t_k)$ and $U \in \mathbb{R}^{n\times n}$ and following the reformulations presented in [17], the general p-stage Rosenbrock scheme reads

$$\begin{aligned}\hat{A}_k^TK_i + K_i\hat{A}_k &= -\mathcal{R}\left(t_{k,i}, X_k + \sum_{j=1}^{i-1} a_{ij}K_j\right)\\ &\quad - \sum_{j=1}^{i-1}\frac{c_{ij}}{\tau_k}K_j - \gamma_i\tau_k\mathcal{R}_{t_k},\\ X_{k+1} &= X_k + \sum_{j=1}^{p} m_jK_j,\end{aligned} \tag{10}$$

with $\hat{A}_k := A_k - S_kX_k - \frac{1}{2\tau_k\gamma_{ii}}I$, $i = 1, \ldots, p$. Hence, in each stage of every time step of the integration method one algebraic Lyapunov equation has to be solved. In order to avoid explicitly forming the dense solutions K_i of the single stage equations in (10), as in the BDF-case, we assume the coefficient matrices to be given in low-rank form. The particular low-rank representation directly depends on the order of the Rosenbrock method. Therefore, as examples we review the first- and a second-order Rosenbrock scheme for an autonomous DRE discussed in [16]. Considering the autonomous case is not an inappropriate restriction, since the application of a low-rank factorization to $\mathcal{R}_{t_k}$ is straight forward, see [19, Section 4.4].

The 1-stage Rosenbrock scheme (Ros1) in low-rank representation is given as

$$\hat{A}_k^TX_{k+1} + X_{k+1}\hat{A}_k = -G_kG_k^T, \tag{11}$$

with $\gamma_{1,1} = 1$, $\hat{A}_k = A_k - S_kX_k - \frac{1}{2\tau_k}I$ and the right hand side factor

$$G_k = \left[C_k^T, \quad Z_kZ_k^TB_k, \quad \sqrt{\tfrac{1}{\tau_k}}Z_k\right].$$

The specific second-order Rosenbrock scheme (Ros2) proposed in [32] in classical low-rank representation and a number of reformulation steps following [16,19], reads

$$\begin{aligned} X_{k+1} &= X_k + \frac{3}{2}\tau_k K_1 + \frac{1}{2}\tau_k K_2, \\ \tilde{A}_k^T K_1 + K_1 \tilde{A}_k &= -\mathcal{R}(X_k), \\ \tilde{A}_k^T K_{21} + K_{21} \tilde{A}_k &= -\tau_k^2 K_1 B_k B_k^T K_1 - (2 - \frac{1}{\gamma}) K_1 \\ K_2 &= -K_{21} + (1 - \frac{1}{\gamma}) K_1 \end{aligned} \tag{12}$$

with $\tilde{A}_k := \gamma\tau_k(A_k - S_k X_k) - \frac{1}{2}I$. Again considering the low-rank splitting given in (5), the right hand side of the first stage in (12) becomes

$$-C_k^T C_k - A_k^T Z_k Z_k^T - Z_k Z_k^T A_k + Z_k Z_k^T B_k B_k^T Z_k Z_k^T.$$

As explained in [16,19], we consider the following two possible splittings of the form $-G_k G_k^T$. The partitioning

$$G_k = [\, C_k^T, \quad A_k^T Z_k + Z_k, \quad i Z_k Z_k^T B_k, \quad i A_k^T Z_k, \quad i Z_k \,] \tag{13}$$

of the right hand side ends up being complex. Avoiding complex data requires a superposition approach splitting the first stage equation into the two equations

$$\tilde{A}_k^T \hat{K}_1 + \hat{K}_1 \tilde{A}_k = -N_k N_k^T, \quad \tilde{A}_k^T \tilde{K}_1 + \tilde{K}_1 \tilde{A}_k = -U_k U_k^T \tag{14}$$

such that $K_1 := \hat{K}_1 - \tilde{K}_1$ and $-G_k G_k^T := -N_k N_k^T + U_k U_k^T$. Here,

$$N_k = [\, C_k^T, \quad A_k^T Z_k + Z_k \,], \quad U_k = [\, Z_k Z_k^T B_k, \quad A_k^T Z_k, \quad Z_k \,].$$

Several numerical experiments have shown that the formation of K_1 may suffer from cancellation problems in finite arithmetic. That is, constructing the solution $K_1 := \hat{K}_1 - \tilde{K}_1$ is affected by numerical inaccuracies and therefore breaks the entire second-order low-rank algorithm.

For completeness, the right hand side of the second stage equation of the Rosenbrock scheme (12) in standard low-rank representation with $K_1 = T_1 T_1^T$, $T_1 \in \mathbb{R}^{n \times t_k}$ reads

$$G_k = \left[\tau_k T_1 T_1^T B, \quad \sqrt{2 - \tfrac{1}{\gamma}} T_1 \right].$$

Other implicit methods As stated in [14] the application of any implicit method to the DRE yields an ARE to be solved in every step. For illustration, we will consider the Midpoint and Trapezoidal rules.

The Midpoint rule applied to the DRE (3) yields

$$X_{k+1} = X_k + \tau_k \mathcal{R}\left(t_k + \frac{\tau}{2}, \frac{1}{2}(X_{k+1} + X_k)\right).$$

Re-arranging terms, we see that this again leads to an ARE for X_{k+1}

$$\begin{aligned}&\left[\tau_k Q_{k'} + X_k + \frac{\tau_k}{2}\left(A_{k'}^T X_k + X_k A_{k'} - \frac{X_k S_{k'} X_k}{2}\right)\right] \\ &\quad + (\frac{\tau_k}{2}A_{k'} - \frac{\tau_k}{4}S_{k'}X_k - \frac{1}{2}I)^T X_{k+1} + X_{k+1}(\frac{\tau_k}{2}A_{k'} - \frac{\tau_k}{4}S_{k'}X_k - \frac{1}{2}I) \\ &\quad - X_{k+1}(\frac{\tau_k}{4}S_{k'})X_{k+1} = 0,\end{aligned} \tag{15}$$

where $X_k \approx X(t_k)$, $A_{k'} \equiv A(t_k + \frac{\tau_k}{2})$, $Q_{k'} \equiv Q(t_k + \frac{\tau_k}{2})$, $S_{k'} \equiv S(t_k + \frac{\tau_k}{2})$.

Applying the Trapezoidal rule to the DRE (3), we obtain

$$X_{k+1} = X_k + \frac{\tau_k}{2}(\mathcal{R}(t_k, X_k) + \mathcal{R}(t_{k+1}, X_{k+1})).$$

Collecting terms in the same way as for the previous method, we end up with an ARE for X_{k+1}

$$\begin{aligned}&\left[\frac{\tau_k}{2}Q_{k+1} + X_k + \frac{\tau_k}{2}\left(Q_k + A_k^T X_k + X_k A_k - X_k S_k X_k\right)\right] \\ &\quad + \left(\frac{\tau_k}{2}A_{k+1} - \frac{1}{2}I\right)^T X_{k+1} + X_{k+1}\left(\frac{\tau_k}{2}A_{k+1} - \frac{1}{2}I\right) \\ &\quad - X_{k+1}\left(\frac{\tau_k}{2}S_{k+1}\right)X_{k+1} = 0.\end{aligned} \tag{16}$$

In both cases an ARE has to be solved in every time step. Thus, as for the BDF methods, for the Midpoint and Trapezoidal rule and in general any implicit (Runge–Kutta) method the key ingredient for an efficient algorithm is a fast low-rank ARE solver.

2.2. Application to DLEs

As for the DREs the application of implicit ODE methods in the matrix setting for solving DLEs requires complex arithmetic. As an illustration, we will consider the BDF methods.

Let us consider the time-varying symmetric DLE of the form

$$\begin{aligned}\dot{X}(t) &= Q(t) + X(t)A(t) + A^T(t)X(t) \equiv \mathcal{L}(t, X(t)), \\ X(t_0) &= X_0,\end{aligned} \tag{17}$$

where $t \in [t_0, t_f]$ and $Q(t)$, $A(t) \in \mathbb{R}^{n\times n}$ are piecewise continuous locally bounded matrix-valued functions. Using the same notation as in the previous subsection the application of the BDF methods to the DLE yields an algebraic Lyapunov equation

$$(\tau_k \beta Q_{k+1} - \sum_{j=1}^{p} \alpha_j X_{k+1-j}) + \hat{A}_{k+1}^T X_{k+1} + X_{k+1}\hat{A}_{k+1} = 0, \tag{18}$$

where β, α_j are given in Table 1 and $\hat{A}_{k+1} := (\tau_k \beta A_{k+1} - \frac{1}{2} I)$. The algebraic equation (18) can be written in terms of low-rank factors similar to the Riccati case in (6). That is, the application of a BDF method of order $p \geq 2$ will also require complex arithmetic and storage.

The same happens when Rosenbrock methods are applied to the DLE. Using a p-stage Rosenbrock method, we also have to solve an ALE at each stage of the scheme, since the Frechét derivative of the Lyapunov operator $\mathcal{L}$ in (17) is again the Lyapunov operator of the form (9). Hence, the same numerical problems as for the DRE arise.

2.3. Classical column compression

For all kinds of time integration methods the solution factors of a certain number of previous time steps are a part of the right hand side factor G of the ALEs that have to be solved within the current time integration step. That is, the block size of the right hand side low-rank factor G will increase drastically over time. Therefore, the elimination of redundant information in terms of a column compression based on the numerical rank of the factor becomes necessary. As mentioned before, using higher order integration methods, the right hand sides become indefinite. Therefore, the right hand side factors in the classical low-rank setting become complex. This directly leads to the inadmissibility of the classic rank-revealing QR decomposition and SVD based column compression approaches. In the following we employ MATLAB notation to specify subblocks of a matrix. Note that the rank r in practice needs to be decided numerically or memory restrictions make a rank truncation necessary. Thus, we usually have $G_r G_r^T \approx GG^T$. Still, we present the results for exact computations here.

QR based column compression

i) Compute $G^T = QR\Pi^T$ with $G \in \mathbb{R}^{n\times k}$, $Q \in \mathbb{R}^{k\times k}$, $Q^TQ = I_k$, $R \in \mathbb{R}^{k\times n}$ and a permutation matrix $\Pi \in \mathbb{R}^{n\times n}$.
ii) Set $G_r = \Pi R_r^T \in \mathbb{R}^{n\times r}$, where $r := \operatorname{rank}(R)$ and $R_r := R(1:r,:) \in \mathbb{R}^{r\times n}$, $Q_r := Q(1:r, 1:r) \in \mathbb{R}^{r\times r}$, such that

$$G_r G_r^T = \Pi R_r^T R_r \Pi^T = \Pi R_r^T Q_r^T Q_r R_r \Pi^T = \Pi R^T Q^T Q R \Pi^T = GG^T.$$

SVD based column compression

i) Compute $G = U\Sigma V^T$ with $G \in \mathbb{R}^{n\times k}$, $U \in \mathbb{R}^{n\times k}$, $U^TU = I_k$, $\Sigma \in \mathbb{R}^{k\times k}$ and $V \in \mathbb{R}^{k\times k}$, $V^TV = I_k$.
ii) Set $G_r = U_r\Sigma_r \in \mathbb{R}^{n\times r}$, where $r = \operatorname{rank}(R)$, $U_r := U(:, 1:r) \in \mathbb{R}^{n\times r}$, $\Sigma_r := \Sigma(1:r, 1:r) \in \mathbb{R}^{r\times r}$, and $V_r := V(:, 1:r) \in \mathbb{R}^{k\times r}$, such that

$$G_rG_r^T = U_r\Sigma_r^2U_r^T = U_r\Sigma_rV_r^TV_r\Sigma_rU_r^T = U\Sigma V^TV\Sigma U^T = GG^T.$$

Column compression for complex data Let $G \in \mathbb{C}^{n\times k}$. Therefore, the QR decomposition similar to the real case reads

$$G^H = QR\Pi^T \text{ with } Q \in \mathbb{C}^{k\times k},\ Q^HQ = I_k,\ R \in \mathbb{C}^{k\times n} \text{ and } \Pi \in \mathbb{R}^{n\times n}.$$

Analogously, setting the compressed factor $G_r = \Pi R_r^H \in \mathbb{C}^{n\times r}$ fails, since we have given the right hand side product $G_rG_r^T$. This yields

$$G_rG_r^T = \Pi R_r^H\bar{R}_r\Pi^T \neq \Pi R_r^HQ_r^H\bar{Q}_r\bar{R}_r\Pi^T = GG^T,$$

since the QR decomposition is computed with respect to an unitary matrix Q, i.e., $Q^HQ = I_k$ and not $Q^H\bar{Q} = I_k$.

A similar problem occurs in the case of the SVD based approach. There, we compute

$$G = U\Sigma V^H \text{ with } U \in \mathbb{C}^{n\times k},\ U^HU = I_k,\ \Sigma \in \mathbb{C}^{k\times k},\ V \in \mathbb{C}^{k\times k},\ V^HV = I_k$$

and therefore obtain

$$G_rG_r^T = U_r\Sigma_rV^H\bar{V}U_r^T \neq U_r\Sigma_rV^HV\Sigma_rU_r^T = U_r\Sigma_r^2U_r^T.$$

Clearly, using the matrix $G \in \mathbb{C}^{n\times k}$ in the real symmetric and indefinite product $GG^T \in \mathbb{R}^{n\times n}$ requires us to properly adjust the compression to the outer product in use. We propose the following procedure:

i) Compute $G = QR\Pi^T$ with $Q \in \mathbb{C}^{n\times k}$, $Q^HQ = I_k$, $R \in \mathbb{C}^{k\times k}$, and the permutation matrix $\Pi \in \mathbb{R}^{k\times k}$.
ii) Compute a decomposition $R\Pi^T\Pi R^T = RR^T = V\Lambda V^T$ with $V \in \mathbb{C}^{k\times k}$, $V^TV = I_k$ and a diagonal matrix $\Lambda \in \mathbb{C}^{k\times k}$ with diagonal entries $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_k|$.
iii) Set the compressed factor $G_r := QV_r\Lambda_r^{\frac{1}{2}} \in \mathbb{C}^{n\times r}$ with $r \leq k$ and $|\lambda_{r+1}| \leq \varepsilon$.

Following the statements in [33, Theorem 4.4.13], the existence of a matrix V with $V^TV = I_k$ as in Step ii) is guaranteed, since RR^T is complex symmetric and therefore diagonalizable.

Computing the eigendecomposition for the complex symmetric matrix RR^T within Step ii) in general leads to

$$RR^T = \tilde{V}\Lambda\tilde{V}^{-1}$$

with eigenvectors (columns in $\tilde{V}$) $\tilde{v}_i \in \mathbb{C}^k$. In any software tool based on LAPACK [34], the $\tilde{v}_i$ are normalized but not necessarily orthogonal with respect to the complex inner product. That is, the $\tilde{v}_i$ satisfy the properties

$$\begin{aligned} &\tilde{v}_i^*\tilde{v}_i = 1 \quad \Rightarrow \quad \tilde{v}_i^T\tilde{v}_i \neq 1, \\ &\tilde{v}_i^*\tilde{v}_j \neq 0. \end{aligned} \tag{19}$$

From the principle of biorthogonality, see e.g., [33, Theorem 1.4.7 and Proof of Theorem 4.4.13], we know that the eigenvectors $\tilde{v}_i$, $\tilde{v}_j$ of the symmetric matrix RR^T are orthogonal with respect to the real inner product, i.e., $\tilde{v}_i^T\tilde{v}_j = 0$ for $i \neq j$. Since, the right hand side is constructed to be of the form $GG^T = QV\Lambda V^TQ^T$, we need to ensure $RR^T = V\Lambda V^T$. Using (19) and the orthogonality of $\tilde{v}_i$, $\tilde{v}_j$, we have

$$\tilde{V}^T\tilde{V} = \begin{bmatrix} \tilde{v}_1^T\tilde{v}_1 & & 0 \\ & \ddots & \\ 0 & & \tilde{v}_k^T\tilde{v}_k \end{bmatrix}.$$

That is, the eigenvectors $\tilde{v}_i$ need to be normalized with respect to the real inner product. Defining $V := \tilde{V}\tilde{D}$ with $\tilde{D} = \operatorname{diag}\left(\sqrt{\tilde{v}_1^T\tilde{v}_1}, \ldots, \sqrt{\tilde{v}_k^T\tilde{v}_k}\right)^{-1}$ yields,

$$\begin{aligned} & V^TV = \tilde{D}\tilde{V}^T\tilde{V}\tilde{D} = I_k \\ \Leftrightarrow \quad & \tilde{D}\tilde{V}^T = V^T = V^{-1} = \tilde{D}^{-1}\tilde{V}^{-1}, \end{aligned}$$

with $\tilde{D}^{-1} = \operatorname{diag}\left(\sqrt{\tilde{v}_1^T\tilde{v}_1}, \ldots, \sqrt{\tilde{v}_k^T\tilde{v}_k}\right)$. Therefore, we obtain

$$\begin{aligned} RR^T &= \tilde{V}\Lambda\tilde{V}^{-1} = \tilde{V}\Lambda\tilde{D}\tilde{D}^{-1}\tilde{V}^{-1} \\ &= \tilde{V}\tilde{D}\Lambda\tilde{D}^{-1}\tilde{V}^{-1} = V\Lambda V^{-1} = V\Lambda V^T. \end{aligned}$$

That means, scaling the eigenvectors $\tilde{v}_i$ with $(\tilde{v}_i^T\tilde{v}_i)^{-\frac{1}{2}}$, $i = 1, \ldots, k$ does not change the eigendecomposition of the complex symmetric matrix RR^T and we end up with the required representation

$$RR^T = V\Lambda V^T.$$

Again, note that using the BDF and Rosenbrock methods of order $p \geq 2$, the Midpoint or Trapezoidal rules will lead to indefinite right hand sides for the ALEs that have to be

solved in the innermost iteration. The associated complex splittings require complex data storage, complex arithmetic and as the above statements show the effort for the necessary column compression techniques increases as well. In the case of definite right hand sides and the corresponding real splittings the column compression is either performed by using the QR or an SVD decomposition. Given complex data the proposed approach computes a QR decomposition of the factor to be compressed and an eigendecomposition of the small complex symmetric matrix RR^T that additionally increases the over-all computational effort of the classical low-rank methods for the solution of the DRE.

3. LDL^T-type Lyapunov solvers

As we have shown in the previous sections, the main ingredient to determine the solution of a DRE (3) is to solve an ALE of the form

$$F^T X + XF = -W. \tag{20}$$

For higher order methods ($p \geq 2$) the matrix W appears to be indefinite in every step of either the Rosenbrock method or the Newton method within the BDF schemes, the Midpoint or Trapezoidal rule. In this section we present a new approach, which avoids the problem of complex arithmetic and storage arising when the right hand side is decomposed as $W = GG^T$. We propose to split the right hand side W in the form GSG^T with $G \in \mathbb{R}^{n\times k}$, $k \ll n$ and a small, compared to the dimension n, but indefinite matrix $S = S^T \in \mathbb{R}^{k\times k}$.

3.1. LDL^T-type ADI

Following [35], the ADI iteration is adapted to the splitting $X \approx LDL^T$ of the solution of a Lyapunov equation of the form (20). The solution factor L will be of low rank and D is a symmetric and block-diagonal matrix, as we will easily see below. The one step iteration [24] at step $j = 1, 2, \ldots$ of the ADI method becomes

$$\begin{aligned} L_j D_j L_j^T = &-2\mathrm{Re}(\mu_j)(F^T + \mu_j I)^{-1} GSG^T (F + \bar{\mu}_j I)^{-1} \\ &+ (F^T + \mu_j I)^{-1}(F^T - \mu_j I) L_{j-1} D_{j-1} L_{j-1}^T (F - \mu_j I)(F + \mu_j I)^{-1}, \end{aligned} \tag{21}$$

with $L_0, D_0 = [\,]$ and ADI shift parameters $\mu_j \in \mathbb{C}$. Using the inherent structure of (21) the factors L_j, D_j can be computed as follows:

$$\begin{aligned} L_j &:= [\,(F^T + \mu_j I)^{-1}\mathcal{G}, \quad (F^T + \mu_j I)^{-1}(F^T - \mu_j I) L_{j-1}\,], \\ D_j &:= \begin{bmatrix} -2\mathrm{Re}(\mu_j) S & \\ & D_{j-1} \end{bmatrix}. \end{aligned}$$

For the sake of easier reading we define $R_j := (F^T + \mu_j I)^{-1}$ and $T_j := (F^T - \mu_j I)$. Plugging in the factors L_j and D_j recursively yields

$$L_j = [\, R_j G,\, R_j T_j R_{j-1} G,\, \ldots,\, R_j T_j \ldots R_2 T_2 R_1 G \,],$$
$$D_j = \begin{bmatrix} -2\mathrm{Re}(\mu_j) S & & & \\ & -2\mathrm{Re}(\mu_{j-1}) S & & \\ & & \ddots & \\ & & & -2\mathrm{Re}(\mu_1) S \end{bmatrix}. \tag{22}$$

Since the ordering of the ADI shifts μ_j does not affect the solutions quality, the indices can be reversed. Additionally, using the commutativity of the R_j's and T_j's the reordered sequence leads to

$$L_j = [\, R_1 G,\, R_2 T_1 (R_1 G),\, \ldots,\, R_{j+1} T_j (R_j T_{j-1} \ldots R_2 T_1 R_1 G) \,],$$
$$\begin{aligned} D_j &= \begin{bmatrix} -2\mathrm{Re}(\mu_1) S & & & \\ & -2\mathrm{Re}(\mu_2) S & & \\ & & \ddots & \\ & & & -2\mathrm{Re}(\mu_j) S \end{bmatrix} \\ &= -2\mathrm{diag}\,(\mathrm{Re}(\mu_1), \ldots, \mathrm{Re}(\mu_j)) \otimes S \end{aligned} \tag{23}$$

in complete analogy to the procedure first employed in [23] for the ZZ^T case. Thus, the LDL^T-based factorization does not differ to much from the low-rank factored ADI as proposed in [36–38].

The introduction of the potentially indefinite matrix S in the decomposition of the right hand side immediately avoids the necessity of complex storage and arithmetic. Moreover, the introduction of the diagonal block D_j in every step allows to remove the multiplication of the shifts μ_j from the low-rank factor L_j and for the computation of the block diagonal matrix D_j one only needs to store the given symmetric matrix S and the shift sequence which is done during the ADI anyway. Considering the same right hand side factor G, the classical low-rank factor Z and the LDL^T factor L are computed by the same iteration sequence. Thus, they will be of the same size z_k and quality. A sketch of the LDL^T-type procedure is given in Algorithm 3.1.

Remark Let $\varrho(M)$ denote the spectral radius of a matrix M. Note that the matrices $W_{j-1} S W_{j-1}^T \in \mathbb{R}^{n \times n}$ and $W_{j-1}^T W_{j-1} S \in \mathbb{R}^{k \times k}$ share the same non-zero spectrum. Therefore, to avoid the computation of the norm of the large and usually dense matrix products in Step 2 of Algorithm 3.1, we exploit

$$\|W_{j-1} S W_{j-1}^T\|_2 = \varrho(W_{j-1} S W_{j-1}^T) = \varrho(W_{j-1}^T W_{j-1} S).$$

3.2. LDL^T-type Krylov subspace method

Following the statements in [25,39] the rational Krylov subspace method (RKSM) and the extended Krylov subspace method (EKSM, also called KPIK for Krylov plus

Algorithm 3.1 LDL^T-factorization based ADI method.

INPUT: ADI shifts $\mu_1, \ldots, \mu_\ell \in \mathbb{C}$, G, S, tolerance ε
OUTPUT: $L = L_{n_{ADI}}$, $D = D_{n_{ADI}}$

$W_0 = G$, $j = 1$
while $\|W_{j-1} S W_{j-1}^T\|_2 \geq \varepsilon \|GSG^T\|_2$ **do**
  Solve $(F + \mu_j E)V_j = W_{j-1}$ for V_j.
  if μ_j is real **then**
    $W_j = W_{j-1} - 2\mu_j V_j$, $L_j = [L_{j-1},\ V_j]$
  else
    $\eta_j = \sqrt{2}$, $\delta_j = \operatorname{Re}(\mu_j)/\operatorname{Im}(\mu_j)$
    $W_{j+1} = W_{j-1} - 4\operatorname{Re}(\mu_j)(\operatorname{Re}(V_j) + \delta_j \operatorname{Im}(V_j))$
    $L_{j+1} = [L_{j-1},\ \eta_j(\operatorname{Re}(V_j) + \delta_j \operatorname{Im}(V_j)),\ \eta_j\sqrt{\delta_j^2 + 1}\operatorname{Im}(V_j)]$
    $j = j + 1$
  end if
  $j = j + 1$
end while
$D_j = -2\operatorname{diag}(\operatorname{Re}(\mu_1), \ldots, \operatorname{Re}(\mu_j)) \otimes S$

inverted Krylov) compute a solution

$$X_s = V_s Y_s V_s^T \tag{24}$$

of the ALE (20) with a given right hand side of the form $W := \hat{G}\hat{G}^T$. Here, V_s denotes an orthonormal basis of the Krylov subspace

$$\mathcal{K}_s(F, \hat{G}, p) = \{\hat{G}, (F^T - \mu_1 I)^{-1}\hat{G}, \ldots, \prod_{j=1}^{s}(F^T - \mu_j I)^{-1}\hat{G}\} \subset \mathbb{R}^{n\times(s+1)k} \text{ or}$$

$$\mathcal{K}_{2s}(F, F^{-s}\hat{G}) = \{F^{-s}\hat{G}, \ldots, F^{-1}\hat{G}, \hat{G}, F\hat{G}, \ldots, F^s\hat{G}\} \subset \mathbb{R}^{n\times(2s+1)k},$$

respectively, where k is the number of columns of $\hat{G}$ and Y_s is the solution of the projected small-scale ALE

$$V_s^T F^T V_s Y_s + Y_s V_s^T F V_s = -V_s^T \hat{G}\hat{G}^T V_s. \tag{25}$$

That is, the RKSM and EKSM Lyapunov solvers directly compute the solution of (20) in the required LDL^T-type format. Exploiting the inherent structure of the solution $X_s = V_s Y_s V_s^T$ given by the Krylov subspace methods, the LDL^T based methods avoid the additional computation of a ZZ^T decomposition of the solution X_s as it is done in the classical low-rank algorithms. Note that a splitting GSG^T of the right hand side W of (20) does not affect the procedure. Since S is symmetric, and therefore diagonal up to an orthogonal similarity transformation, the Krylov subspace spanned by the columns of V_s does not change. That is, the only change in the above procedure is the solution of

$$V_s^T F^T V_s Y_s + Y_s V_s^T F V_s = -V_s^T G S G^T V_s$$

instead of (25).

Algorithm 3.2 LDL^T factored BDF method of order p.

Require: $E(t)$, $A(t)$, $S(t)$, $Q(t)$, $\in \mathbb{R}^{n\times n}$ smooth matrix-valued functions satisfying (5), $t \in [a,b]$, and step size τ.
Ensure: $(L_{k+1}, D_{k+1}, t_{k+1})$ such that $X_{k+1} \approx L_{k+1}D_{k+1}L_{k+1}^T$.

$t_0 = a$.
for $k = 0$ to $\lceil \frac{b-a}{\tau} \rceil$ **do**
  $t_{k+1} = t_k + h$.
  $\hat{A}_{k+1} = \tau\beta A_{k+1} - \frac{1}{2}E$.
  $\hat{C}_{k+1}^T = [C_{k+1}^T, E^T L_k, \ldots, E^T L_{k+1-p}]$.
  for $\ell = 1$ to ℓ_{max} **do**
    $G^{(\ell)} = [\hat{C}_{k+1}^T,\ K^{(\ell-1)}]$.
    $S^{(\ell)} = \begin{bmatrix} \tau\beta I_q & & & & \\ & -\alpha_1 D_k & & & \\ & & \ddots & & \\ & & & -\alpha_p D_{k+1-p} & \\ & & & & \tau\beta I_m \end{bmatrix}$.
    Compute $L^{(\ell)}$, $D^{(\ell)}$ by an LDL^T-factorization based Algorithm such that $X^{(\ell)} \approx L^{(\ell)}D^{(\ell)}L^{(\ell)\,T}$ is the solution of

$$F^{(\ell)\,T}X^{(\ell)}E_{k+1} + E_{k+1}^T X^{(\ell)}F^{(\ell)} = -G^{(\ell)}S^{(\ell)}G^{(\ell)\,T}.$$

    $K^{(\ell)} = E_{k+1}^T(L^{(\ell)}(D^{(\ell)}(L^{(\ell)\,T}B_{k+1})))$.
  end for
  $L_{k+1} = L^{(\ell_{max})}$, $D_{k+1} = D^{(\ell_{max})}$.
end for

3.3. Application to matrix-valued ODE solvers

Applying the LDL^T-type splitting to the arising ALEs within the previously described matrix-valued ODE solvers allows us to avoid complex arithmetic arising from the standard low-rank splitting of the right hand sides of the ALEs which need to be solved in the innermost iteration of the BDF, the Midpoint and Trapezoidal rules and Rosenbrock methods. In addition, the number of system solves within the ADI iteration can be reduced by an a priori elimination of redundant column blocks in the right hand sides. Again, for simplicity we restrict ourselves to the autonomous case as in Section 2.1. In particular, we will demonstrate the advantages of the LDL^T-type splitting in the example of the aforementioned general p-step BDF method, as well as for the first- and second-order Rosenbrock schemes.

Backward differentiation formulas Using the LDL^T-type factorization $X_{k+1} := L_{k+1}D_{k+1}L_{k+1}^T$ instead of the standard low-rank representation of the solution of the DRE, Algorithm 3.1 in [16] changes to Algorithm 3.2. That is, the application of the LDL^T factorization and the associated splitting GSG^T of the right hand side of (7) allows us to put the coefficients α_j, $j = 1,\ldots,p$ into the diagonal blocks of S. This avoids taking the square root of the non-positive coefficients (see Table 1) and in turn removes complex data and arithmetic.

As mentioned in Section 2 the Midpoint or Trapezoidal rule also leads to the solution of an ARE in every time integration step. Having a closer look at the corresponding

Eqs. (15) and (16), we note that again the constant terms of the AREs are indefinite. Therefore, the application of the above steps also avoids complex computations. Furthermore, the problem of indefinite right hand sides also appears for the application of Rosenbrock methods.

First-order Rosenbrock method (linear implicit Euler) As given in Eq. (11) the first-order Rosenbrock scheme in standard low-rank formulation deals with the right hand side

$$G_k = \left[C_k^T, \quad Z_k Z_k^T B_k, \quad \sqrt{\tfrac{1}{\tau_k}} Z_k \right] \in \mathbb{R}^{n\times(q+m+z_k)}$$

where G_k is of size $n \times (q + m + z_k)$. Here, the right hand side is definite and therefore can be split into real factors G_k. Still, the application of the LDL^T-type factorization with the associated right hand side $\tilde{G}_k \tilde{S}_k \tilde{G}_k^T$

$$\begin{aligned} \tilde{G}_k &= [\, C_k^T, \quad L_k, \quad L_k \,] && \in \mathbb{R}^{n\times(q+2\ell_k)} \\ \tilde{S}_k &= \begin{bmatrix} I & & \\ & D_k L_k^T B_k B_k L_k D_k & \\ & & \frac{1}{\tau_k} D_k \end{bmatrix} && \in \mathbb{R}^{(q+2\ell_k)\times(q+2\ell_k)} \end{aligned}$$

for the solution factorization $X_k \approx L_k D_k L_k^T$ can be exploited to improve the numerical computations. Re-arranging the blocks in the form

$$\begin{aligned} \tilde{G}_k &= [\, C_k^T, \quad L_k \,] && \in \mathbb{R}^{n\times(q+\ell_k)}, \\ \tilde{S}_k &= \begin{bmatrix} I & \\ & D_k L_k^T B_k B_k L_k D_k + \frac{1}{\tau_k} D_k \end{bmatrix} && \in \mathbb{R}^{(q+\ell_k)\times(q+\ell_k)} \end{aligned} \tag{26}$$

leads to a factor $\tilde{G}_k$ of size $n \times (q + \ell_k)$ representing the same product. The number of columns of G_k, $\tilde{G}_k$ equals the number of solves within the first step of the Lyapunov solver and the number of columns which are added to the right hand side at every subsequent iteration step. This at least saves m system solves in every step of the Lyapunov solver within every time integration step, since it can be shown that the block sizes ℓ_k, z_k satisfy $\ell_k \leq z_k$. Both block sizes linearly depend on the size of the right hand side put into the ADI or Krylov based Lyapunov solvers. Therefore, ℓ_k cannot exceed z_k. In total this means that assuming a constant number n_{lyap} of Lyapunov solver steps per time step, the LDL^T-type factorization for the linear implicit Euler integration method requires at least $m \cdot n_{lyap} \cdot n_{ODE}$ less linear system solves during the solution of the DRE (3) compared to the standard low-rank factorization. Here, n_{ODE} is the number of time steps taken in the linear implicit Euler scheme. Note that the products $D_k L_k^T B_k$ are of size $\ell_k \times m$ and therefore do not require a significant amount of computation time as long as ℓ_k, $m \ll n$, which is a required assumption for low-rank computations anyway.

Second-order Rosenbrock method As introduced in Eqs. (13) and (14) for the first stage equation of the second order method we either have to deal with the complex right hand side

$$G_k = [\, C_k^T, \quad A_k^T Z_k + Z_k, \quad iZ_k Z_k^T B_k, \quad iA_k^T Z_k, \quad iZ_k \,], \quad \in \mathbb{R}^{n\times(q+m+3z_k)}$$

or the split Lyapunov equation and the corresponding right hand sides $N_k N_k^T$, $U_k U_k^T$ with

$$\begin{aligned} N_k &= [\, C_k^T, \quad A_k^T Z_k + Z_k \,], && \in \mathbb{R}^{n\times(q+z_k)}, \\ U_k &= [\, Z_k Z_k^T B_k, \quad A_k^T Z_k, \quad Z_k \,], && \in \mathbb{R}^{n\times(m+2z_k)}. \end{aligned}$$

In order to avoid the complex blocks, the splitting of the first stage Lyapunov equation into two separate ALEs, and the additionally introduced terms using (14), again, we consider the LDL^T-type splitting. Hence, the right hand side of the first stage equation becomes

$$-C_k^T C_k - A_k^T L_k D_k L_k^T - L_k D_k L_k^T A_k + L_k D_k L_k^T B_k B_k^T L_k D_k L_k^T$$

and we obtain the splitting $-\tilde{G}_k \tilde{S}_k \tilde{G}_k^T$ with

$$\begin{aligned} \tilde{G}_k &= [\, C_k^T, \quad A_k^T L, \quad L_k, \quad L_k \,], \\ \tilde{S}_k &= \begin{bmatrix} I_q & & & \\ & & D_k & \\ & D_k & & \\ & & & -D_k L_k^T B_k B_k^T L_k D_k \end{bmatrix}. \end{aligned} \tag{27}$$

Re-arranging blocks, similar to (26), leads to

$$\begin{aligned} \tilde{G}_k &= [\, C_k^T, \quad A_k^T L_k, \quad L_k \,] && \in \mathbb{R}^{n\times(q+2\ell_k)}, \\ \tilde{S}_k &= \begin{bmatrix} I_q & & \\ & & D_k \\ & D_k & -D_k L_k^T B_k B_k^T L_k D_k \end{bmatrix} && \in \mathbb{R}^{(q+2\ell_k)\times(q+2\ell_k)}. \end{aligned} \tag{28}$$

Hence, the number of system solves within the Lyapunov solver for the first stage equation is reduced from $q + m + 3z_k$ for the classical low-rank representation to $q + 2\ell_k$ for the LDL^T-type factorization. That is, we are able to save at least $m + z_k$ linear system solves for the solution of stage 1.

As mentioned in Section 2.1, the right hand side of the second stage equation reads

$$G_k = \left[\tau_k T_1 T_1^T B, \quad \sqrt{2 - \tfrac{1}{\gamma}} T_1 \right] \in \mathbb{R}^{n\times(m+t_k)}.$$

Now, using the LDL^T-type splitting with $K_1 = \tilde{T}_1 D_1 \tilde{T}_1^T$ we obtain

$$\tilde{G}_k = T_1 \in \mathbb{R}^{n \times \tilde{t}_k},$$
$$\tilde{S}_k = +\tau_k^2 D_1 T_1^T B B^T T_1 D_1 + (2 - \frac{1}{\gamma}) D_1 \in \mathbb{R}^{\tilde{t}_k \times \tilde{t}_k}.$$

Since $\tilde{t}_k \leq t_k$, we save at least another m linear system solves for the solution of the second stage equation.

In total this leads to savings of a minimum of $2m + z_k$ solves in each step of the ALE solver within each time integration step. Again, note that for an increasing order of the Rosenbrock scheme the number of ALEs, which have to be solved via the classical low-rank or the LDL^T based scheme increases. That means the total number of system solves within the Lyapunov solvers to be performed will increase as well. Therefore, using the LDL^T approach with analogous block re-arrangements as above will lead to similar savings for each of these stages. That means, the higher the order of the integration method one uses, the better the accuracy of the solution will be, while at the same time the speedup caused by the LDL^T-type factorization will increasingly pay off.

3.4. LDL^T column compression

As for the classical low-rank methods the right hand side low-rank factors will increase within each time integration step. That is, we also need to perform a column compression in order to reduce the number of columns of the LDL^T-type right hand sides or DRE solutions. Consider the matrix GSG^T, where $G \in \mathbb{R}^{n \times k}$, $S = S^T \in \mathbb{R}^{k \times k}$. Following the statements in Section 6.3.3 in [40] the factors G, S can be compressed as follows:

i) Compute $G = QR\Pi^T$ with $Q \in \mathbb{R}^{n \times k}$, $R \in \mathbb{R}^{k \times k}$ and $\Pi \in \mathbb{R}^{n \times n}$.
ii) Compute a decomposition $R\Pi^T S \Pi R^T = V \Lambda V^T$ with $V \in \mathbb{R}^{k \times k}$ and a diagonal matrix $\Lambda \in \mathbb{R}^{k \times k}$ with diagonal entries $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_k|$.
iii) Set the compressed factors $G_r := QV_r \in \mathbb{R}^{n \times r}$, $S_r := \Lambda_r$ with $r \leq k$ and $|\lambda_{r+1}| \leq \varepsilon$.

Since $R\Pi^T S \Pi R^T \in \mathbb{R}^{k \times k}$ is symmetric, a decomposition $V \Lambda V^T$ always exists and can e.g., be computed via an eigendecomposition. Comparing the computational cost, the above procedure is equal to the classical low-rank column compression for complex data if the sizes of the thin rectangular matrices coincide.

4. Numerical results

All the following examples are executed on a 64 bit CentOS 5.5 system with two Intel® Xeon® X5650@2.67 GHz with a total of 12 cores and 48 GB main memory.

As an illustrating problem in this section we consider the linear quadratic regulator (LQR) problem

$$\begin{aligned} \min_u \ J(t;\, y, u) &= \int_{t_0}^{t_f} y^T(t) Q y(t) + u^T(t) R u(t)\ dt + y(t_f)^T M y(t_f), \\ \text{s.t.} \quad E\dot{x}(t) &= Ax(t) + Bu(t), \\ y &= Cx(t) \end{aligned} \tag{29}$$

on the finite time horizon $t \in [t_0, t_f]$ with symmetric weighting matrices Q, R. The state space systems, we consider in the remainder, are all linear time invariant (LTI). Considering LTV systems, it is still a crucial question to find an efficient storage strategy for the given data $E(t)$, $A(t)$, $B(t)$, $C(t)$ and the resulting solution factors of $X(t)$ of the DRE. The optimal solution to (29) is given by the feedback law (see, e.g., [41])

$$u = -R^{-1} B^T X(t) E x(t) = -K(t) x(t). \tag{30}$$

This means, in order to compute the optimal solution u of (29), we need to find a matrix valued function $X(t)$, which is given as the solution of the generalized DRE

$$\begin{aligned} E^T \dot{X} E &= -Q - A^T X E - E^T X A + E^T X B B^T X E, \\ E^T X(t_f) E &= 0. \end{aligned} \tag{31}$$

Note that the DRE arising from an LQR problem has to be solved backwards in time. Therefore, the following results, in particular the convergence behavior of the DRE to the ARE, need to be interpreted starting from the end point of the corresponding time interval. For the examples below we depict the evolution of one component $K_{i,j}(t)$ of the feedback matrix $K(t)$ in Eq. (30) with $i = 1, \ldots, m$, $j = 1, \ldots, n$. The selected components $K_{i,j}$ are chosen with a relatively large amplitude such that differences are well visible. In contrast to the depiction of the convergence behavior, all relative errors in the remainder are given in the Frobenius norm $\|.\|_F$ for the full feedback $K(t)$ over the entire time interval. That is, the errors of the low-rank schemes compared to a reference solution or between both low-rank representations are computed in the form

$$\frac{\|K_{ref}(t) - K_{LR/LDL}(t)\|_F}{\|K_{ref}(t)\|_F} \quad \text{or} \quad \frac{\|K_{LR}(t) - K_{LDL}(t)\|_F}{\|K_{LR}(t)\|_F},$$

respectively. Further, for all examples machine precision is used as the accuracy tolerance for both, the Lyapunov solvers and the column compression techniques inside the DRE solvers. These tolerances are chosen in order to compare the most accurate results available for the different solution strategies. That is, the problem is avoided that both schemes introduce additional and in particular different numerical errors which may lead to relatively large errors in the direct comparison of the classical ZZ^T and the LDL^T computations. Finally, the choice of the error tolerances for, e.g., the compression techniques, the iterative Lyapunov solvers, and Newton's method is up to the user and of course depends on the demands of the application to be considered.

4.1. ADI based Lyapunov solvers

The examples in the section below present the numerical results achieved for an ADI iteration based Lyapunov solver inside the time integration schemes.

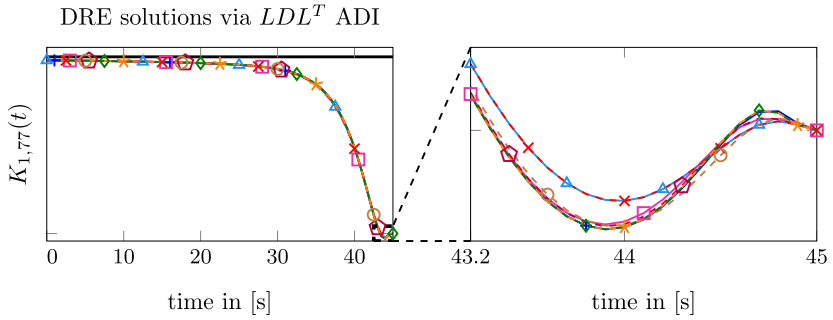
4.1.1. Example 1: steel profile

We consider the semi-discretized heat transfer model described in [42]. The model is given with $m = 7$ inputs and $q = 6$ outputs. The solution is computed on the time interval $[0, 45]$ s. Note that the time line for the simulation is scaled by $1e2$. That is, we consider a model time interval of 4500 seconds. In the remainder we state the real time instances as a hundredth of the model time quantities.

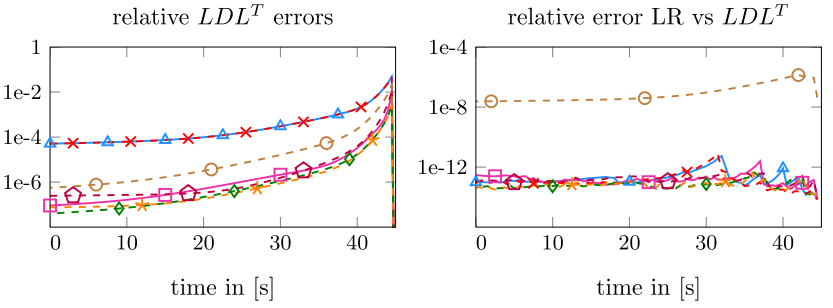
In order to be able to compare the results of the different low-rank DRE solvers to a classical dense 4th order Rosenbrock scheme (Ros4) [43], we start with the smallest state space dimension available, $n = 371$. Fig. 1a shows component $K_{1,77}(t)$ of the reference solution computed via the Ros4 with the fixed time step size $\tau =$ 1e-4 compared to the LDL^T based solutions of the BDF methods of order $p = 1, 2, 3$, the Midpoint and Trapezoidal rules, and the Rosenbrock methods of order one and two performed with a fixed time step size $\tau =$ 1e-1. In addition, the constant solution of the corresponding ARE is depicted in order to show the convergence of the several methods. Fig. 1b presents the relative errors of the entire solution $K(t)$ for the different LDL^T methods compared to the reference solution in the Frobenius norm. Further, the relative errors of the solutions of the classical low-rank and the LDL^T representation are depicted in Fig. 1c.

Table 2 presents the computation times for the different methods the relative error between the classical low-rank methods and the LDL^T based algorithms, as well as the relative errors of the LDL^T procedures compared to the reference solution. We see that the LDL^T based DRE solvers achieve a speed-up up to a factor of around 2 for the majority of the methods. The rather small time savings in the case of the 1-stage Rosenbrock scheme is due to the definiteness of the right hand side of the ALE (11). Here, the benefits of avoiding complex data and the splitting of the ALE do not come into effect. That is, the decrease in time originates solely from saving the m system solves within each ADI step at each time integration step. Table 2 further shows that the classical low-rank solvers and the LDL^T based schemes achieve the same results except for acceptable round off errors. Still, there seems to be a problem with the Ros2 method. This has to be further investigated in the future.

Fig. 2 presents some accuracy results with respect to the chosen time integration methods and the time step size. In Fig. 2a the comparison of the computation times and the achieved accuracy is given for the BDF methods of order $p = 1, 2, 3$, the first- and second-order Rosenbrock methods, the Midpoint and Trapezoidal rules for both, the classical low-rank and the LDL^T based integration schemes, computed with the time step size $\tau =$ 1e-1. In accordance with Table 2, we observe the superiority of the LDL^T based methods with respect to the computation times. Fig. 2b shows the increasing accuracy for decreasing time step sizes τ using the LDL^T based algorithms. Here, the accuracy



(a) Convergence behavior of solution component $K_{1,77}$ on the time interval $[0, 45]$ s (left) and zoom into the time interval $t \in [43.2, 45]$ s (right).



(b) Relative error of the LDL^T based solution compared to the reference solution for $t \in [0, 45]$ s.

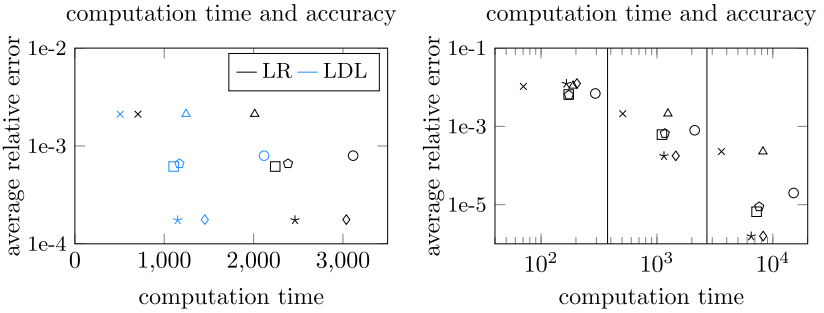
(c) Relative error between the solutions of the classical ZZ^T low-rank and the LDL^T based methods for $t \in [0, 45]$ s.

Fig. 1. Comparison of the dense 4th order Rosenbrock reference solution computed with step size $\tau =$ 1e-4 and the LDL^T BDF methods of order $p = 1, 2, 3$, the Midpoint and Trapezoidal rules and the Rosenbrock methods of order $p = 1, 2$ for Example 4.1.1 computed with a fixed step size $\tau =$ 1e-1.

Table 2
Timings, avg. rel. Frobenius errors for Example 4.1.2 with $n = 371$ on the time interval $[0, 45]$ s, $\tau =$ 1e-1.

	Time in s		Speedup	Avg. rel. err.	
	LR	LDL		LRvsLDL	LDLvsRef
BDF1	2012.55	1243.56	1.62	1.10e-12	2.42e-03
BDF2	2242.24	1105.14	2.03	3.28e-13	1.15e-03
BDF3	2385.34	1170.08	2.04	1.71e-13	1.25e-03
Ros1	705.62	507.12	1.39	1.29e-12	2.41e-03
Ros2	3113.80	2117.19	1.47	3.60e-07	1.15e-03
Midpoint	3037.82	1453.94	2.09	1.06e-13	3.56e-04
Trapezoidal	2463.25	1150.39	2.14	9.02e-14	3.54e-04

of the BDF3 method is slightly worse compared to the BDF2 scheme. This is due to the fact that for the time step sizes $\tau =$ 1e-1 and $\tau =$ 1e-2 the BDF2 method already reaches the maximum possible accuracy and the additional summand of the previous time step solutions in the constant term of the ARE (4) may introduce additional numerical errors.



(a) Elapsed time and average relative error of the low-rank methods with τ=1e-1. (for detailed values see Table 2)

(b) Timings and avg. rel. Frobenius errors of the LDL^T methods for $\tau \in \{1e0,1e-1,1e-2\}$.

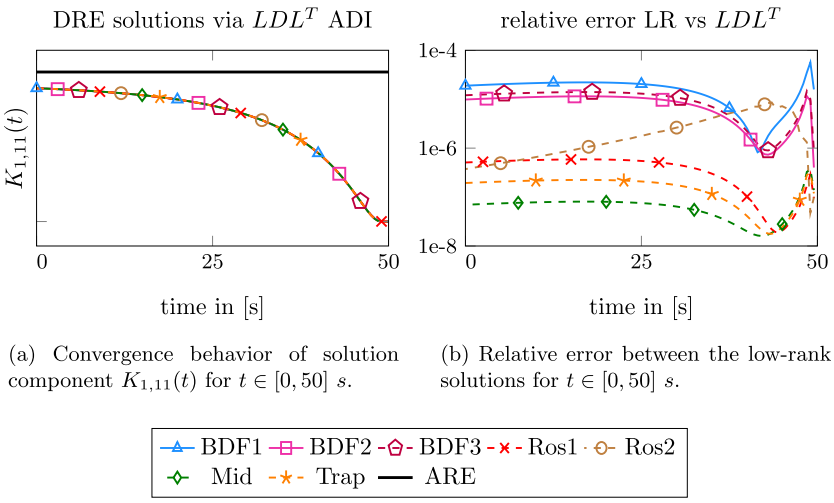
Fig. 2. Efficiency investigations for Example 4.1.1 w.r.t. the low-rank schemes (Fig. 2a) and decreasing time steps τ for the LDL^T methods (Fig. 2b). (Markers denote different time integration methods as above, see Fig. 1.)

Table 3
Timings, avg. rel. Frobenius errors between the low-rank methods for Example 4.1.2 with $n = 1357$ on the time interval $[0, 45]$ s, $\tau = 1$.

	Time in s		Speedup	Avg. rel. err.
	LR	LDL		LRvsLDL
BDF1	1716.73	1233.68	1.39	3.69e-13
BDF2	2771.89	1224.95	2.26	1.92e-14
BDF3	2833.40	1246.70	2.27	4.02e-14
Ros1	616.93	659.25	0.94	5.46e-13
Ros2	4825.75	2516.20	1.92	1.37e-06
Midpoint	3509.17	1456.12	2.41	2.58e-13
Trapezoidal	3052.79	1290.91	2.37	4.34e-13

The convergence behavior of the Ros2 depicted in Fig. 2b, which seems to be neither first- nor second-order results from the fact that the convergence order in general is approached asymptotically. That is, the region of second order convergence is not yet reached for the rather large timesteps chosen in order to keep the simulation time and storage consumption within appropriate limits.

In Table 3 we present the results of the steel profile model with $n = 1357$ degrees of freedom computed with a fixed time step size $\tau = 1$. Given are the timings of the standard low-rank codes compared to the LDL^T implementations and the average of the relative errors between both of them. For the Ros1 we observe that the timings for the classical low-rank version and the LDL^T method are basically the same with slight advantages for the classical low-rank splitting. The savings of the system solves within the LDL^T based scheme (see Section 3.3) cannot entirely compensate the additional effort of the LDL^T compression technique for the real definite right hand sides arising in the Ros1. Using higher order methods, as e.g., the Midpoint and Trapezoidal rules, the LDL^T routines benefit from all their advantages, i.e., avoiding complex data and the removal of redundant information by re-arranging the S block of the right hand sides. Therefore, the LDL^T version achieves a significant time saving.



(a) Convergence behavior of solution component $K_{1,11}(t)$ for $t \in [0, 50]$ s.

(b) Relative error between the low-rank solutions for $t \in [0, 50]$ s.

Fig. 3. Comparison of the LDL^T BDF methods of order $p = 1, 2, 3$, the Midpoint and Trapezoidal rules and the Rosenbrock methods of order $p = 1, 2$ for Example 4.1.2 computed with a fixed step size $\tau =$ 1e-1.

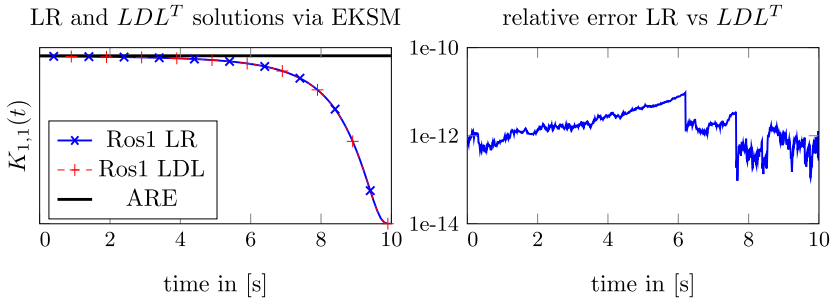
Table 4
Timings, avg. rel. Frobenius errors between the low-rank methods for Example 4.1.2 on the time interval $[0, 50]$ s, $\tau =$ 1e-1.

	Time in s		Speedup	Avg. rel. err.
	LR	LDL		LRvsLDL
BDF1	12 719.51	5666.01	2.23	1.95e-05
BDF2	12 285.17	5444.87	2.26	1.01e-05
BDF3	12 704.44	5530.13	2.30	1.24e-05
Ros1	3043.56	2926.64	1.04	5.23e-07
Ros2	19 967.16	9667.77	2.07	2.20e-06
Midpoint	15 219.88	6836.42	2.23	7.20e-08
Trapezoidal	13 042.51	5666.42	2.30	1.99e-07

4.1.2. Example 2: diffusion on the unit square

The second example describes a diffusion model acting on the unit square with $n =$ 1089 degrees of freedom. The system matrices E, A, B, C are given from a finite element discretization. Here, $E \in \mathbb{R}^{n\times n}$ is a FEM mass matrix, $A \in \mathbb{R}^{n\times n}$ denotes the 2D Laplacian on the unitsquare, $B \in \mathbb{R}^{n\times m}$ with $m = 1$ realizes a single input at the entire left boundary and we observe $q = 9$ degrees of freedom of the FE grid at the remaining edges of the unit square via the output matrix $C \in \mathbb{R}^{q\times n}$. The output matrix C consists of $q = 9$ unit vectors encoding the output locations with respect to the chosen degrees of freedom. Similar to the above example Fig. 3a shows the convergence behavior of the solutions of the different time integration methods, computed on the time interval $[0, 50]$ s with the time step size $\tau =$ 1e-1, to the solution of the ARE. Furthermore, Fig. 3b depicts the relative errors of the solutions of the low-rank methods compared to the LDL^T results.

Table 4 shows the timings of the standard low-rank algorithms and the LDL^T based schemes, as well as the average relative errors between both. In contrast to the above



(a) Convergence behavior of solution component $K_{1,1}(t)$ for $t \in [0, 10]$ s.

(b) Relative error between the low-rank solutions for $t \in [0, 10]$ s.

Fig. 4. Comparison of the standard low-rank and LDL^T Rosenbrock method of order 1 for Example 4.2.1 computed with a fixed step size τ = 1e-2.

Table 5
Timings, avg. rel. Frobenius errors between the low-rank methods for Example 4.2.1 on the time interval $[0, 10]$ s, τ = 1e-2.

	Time in s		Speedup	Avg. rel. err.
	LR	LDL		LRvsLDL
Ros1	421.98	254.57	1.66	2.04e-12

example, we observe that in the case of the first-order Rosenbrock method the saving of $m = 1$ system solves, related to the single input, within every ADI step at each time integration step approximately counterbalances the additional computational effort of the column compression for the LDL^T factorization with slight advantages for the LDL^T methods.

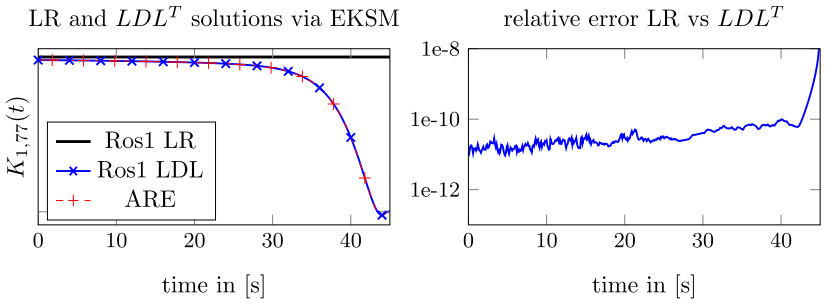
4.2. Krylov based Lyapunov solvers

The following example presents the numerical results achieved for an EKSM based Lyapunov solver. The basic EKSM code is available at the webpage of V. Simoncini,[1] see [25]. Here, we adapted the code in order to apply the EKSM to the closed loop operators $\check{A}$, $\hat{A}$ arising in the ALEs (7) and (10).

4.2.1. Example 3: carex model

The third example originates from the CAREX benchmark collection for continuous-time algebraic Riccati equations [44, Example 4.2]. The model is a single-input-single-output (SISO) state-space system with $E, A \in \mathbb{R}^{n\times n}$, $B = C^T \in \mathbb{R}^n$ and $n = 1000$. Fig. 4 shows component $K_{1,1}(t)$ computed via the classical low-rank Ros1 scheme compared to the LDL^T based version. The computation times and the average relative errors are presented in Table 5.

[1] http://www.dm.unibo.it/~simoncin/software.html.



(a) Convergence behavior of solution component $K_{1,77}(t)$ for $t \in [0, 45]$ s.

(b) Relative error between the low-rank solutions for $t \in [0, 45]$ s.

Fig. 5. Comparison of the standard low-rank and LDL^T Rosenbrock method of order 1 for Example 4.2.2 computed with $\tau =$ 1e-2.

Table 6
Timings, avg. rel. Frobenius errors between the low-rank methods for Example 4.2.2 on the time interval $[0, 45]$ s, $\tau =$ 1e-1.

	Time in s		Speedup	Avg. rel. err.
	LR	LDL		LRvsLDL
Ros1	197.73	118.58	1.67	1.32e-10

4.2.2. Example 4: steel profile

Again, we consider the semi-discretized heat transfer model from Example 1 in Section 4.1.1 with $n = 371$, $\tau =$ 1e-1 on the time interval $[0, 45]$ s. Similar to Example 4.2.1, Fig. 5 presents solution component $K_{1,77}$ for both, the classical low-rank and LDL^T based EKSM Lyapunov solvers inside the Ros1. Further, the relative error between both representations is given and shows the equality of the algorithms except for numerical deviations. Table 6, in addition shows the computation times and the average relative error between the solution approaches.

5. Conclusion

We have investigated the p-step BDF, the p-stage Rosenbrock methods and the Midpoint and Trapezoidal rules applied to matrix valued differential equations. In particular we have seen the application of those time integration schemes to the Riccati differential equation.

A review of an efficient solution strategy in terms of the standard low-rank techniques was given. We revealed several problems of the classical methods regarding complex data and cancellation effects arising in a superposition approach for the solution of the algebraic Lyapunov equations with indefinite right hand sides that need to be solved in the innermost loops of the DRE solvers. We have shown that these problems show up for higher order integration methods, that are recommended to use due to the high stiffness of, e.g., the DRE. Our main contribution is the presentation of an LDL^T based

decomposition of the solution of the DRE and the right hand side of the arising ALEs. This special type of factorization naturally avoids all of the aforementioned problems and allows us to a-priori reduce the number of system solves to be performed during the ADI iteration based Lyapunov solver. Also the additional computation of an artificial low-rank splitting for Krylov subspace Lyapunov solvers can be removed. Further, we have presented a column compression technique dealing with complex data later applied in a real inner product. In addition, we propose a compression technique for the LDL^T factors.

The theoretically stated advantages have been numerically validated for a number of examples. Here, we compared the accuracy, as well as the computation times of the classical low-rank and the LDL^T based methods for ADI and Krylov subspace based solvers for Lyapunov equations. Using an ADI based Lyapunov solver the given examples show that the LDL^T formulation significantly reduces the computation time for higher order methods. In case of, e.g., a first order Rosenbrock method the classical low-rank representation will achieve faster results as long as the savings of the linear system solves in the LDL^T based method cannot compensate the extra cost of the column compression for the LDL^T decompositions. In Section 3.3 we have shown that this directly depends on the number of inputs and outputs, which mainly determine the size of the right hand side, which we also observe in the examples in Sections 4.1.1 and 4.1.2. For the Krylov subspace based solvers we have shown that the time savings are generated by the avoidance of an additional and artificial recreation of a classical low-rank factorization of the form ZZ^T. We also observe that the extended Krylov subspace methods can compute the solution of the DRE in less time compared to the ADI solvers. One reason for this is the necessary computation of the ADI shift parameter. Another reason is that the ADI can be interpreted as a rational Krylov method, where in each step a different coefficient matrix is used compared to the repeated use of the same matrix in the case of EKSM. Also the shifts are in general complex. Therefore the linear systems in the ADI method (but also an RKSM based method) require complex arithmetic and cannot reuse LU decompositions in contrast to EKSM. Regarding the shift parameters at the moment we use the heuristic by Penzl, see e.g., [45], which is expensive with respect to the computational cost. In order to avoid the rather large computation times for the shift parameter computation, as a next step, we want to incorporate the ideas from [37]. A more direct comparison of the ADI and EKSM/RKSM based Lyapunov solvers inside the time integration methods is postponed and will be reported somewhere else. It is, however, expected that ADI and RKSM based solvers will be computationally slower due to the changing coefficient matrices, but can produce faster convergence in terms of required iteration numbers. In turn they will produce smaller factors and thus reduce the storage requirements in general.

References

[1] P. Benner, Z. Tomljanović, N. Truhar, Optimal damping of selected eigenfrequencies using dimension reduction, Numer. Linear Algebra Appl. 20 (2013) 1–17.

[2] F. Blanchini, D. Casagrende, P. Gardonio, S. Miani, Constant and switching gains in semi-active damping of vibrating structures, Internat. J. Control 85 (2012) 1886–1897.
[3] J. Hoepffner, Stability and control of shear flows subject to stochastic excitations, Ph.D. thesis, KTH Royal Institute of Technology, Sweden, 2006.
[4] C. Penland, M. Fluegel, P. Chang, The role of stochastic forcing in modulating ENSO predictability, J. Climate 17 (2004) 3125–3140.
[5] C. Penland, P.D. Sardeshmukh, The optimal growth of tropical sea surface temperature anomalies, J. Climate 8 (1995) 1999–2024.
[6] H. Sandberg, Model reduction for linear time-varying systems, Ph.D. thesis, Lund Institute of Technology, Sweden, 2004.
[7] H. Sandberg, A case study in model reduction of linear time-varying systems, Automatica 43 (3) (2006) 467–472.
[8] H. Abou-Kandil, G. Freiling, V. Ionescu, G. Jank, Matrix Riccati Equations in Control and Systems Theory, Birkhäuser, Basel, Switzerland, 2003.
[9] A. Ichikawa, H. Katayama, Remarks on the time-varying H_∞ Riccati equations, Systems Control Lett. 37 (5) (1999) 335–345.
[10] O.L.R. Jacobs, Introduction to Control Theory, 2nd edition, Oxford Science Publication, Oxford, 1993.
[11] I.R. Petersen, V.A. Ugrinovskii, A.V. Savkin, Robust Control Design Using H^∞ Methods, Springer-Verlag, London, UK, 2000.
[12] C. Choi, A.J. Laub, Efficient matrix-valued algorithms for solving stiff Riccati differential equations, IEEE Trans. Automat. Control 35 (1990) 770–776.
[13] E.J. Davison, M.C. Maki, The numerical solution of the matrix Riccati differential equation, IEEE Trans. Automat. Control 18 (1973) 71–73.
[14] L. Dieci, Numerical integration of the differential Riccati equation and some related issues, SIAM J. Numer. Anal. 29 (3) (1992) 781–815.
[15] C. Kenney, R.B. Leipnik, Numerical integration of the differential matrix Riccati equation, IEEE Trans. Automat. Control 30 (1985) 962–970.
[16] P. Benner, H. Mena, Numerical solution of the infinite-dimensional LQR-problem and the associated differential Riccati equations, preprint MPIMD/12-13, Max Planck Iinstitute Magdeburg, 2012, available from http://www.mpi-magdeburg.mpg.de/preprints/.
[17] P. Benner, H. Mena, Rosenbrock methods for solving differential Riccati equations, IEEE Trans. Automat. Control 58 (11) (2013) 2950–2957.
[18] E. Hansen, T. Stillfjord, Convergence analysis for splitting of the abstract differential Riccati equation, SIAM J. Numer. Anal. 52 (6) (2014) 3128–3139, http://dx.doi.org/10.1137/130935501.
[19] H. Mena, Numerical solution of differential Riccati equations arising in optimal control problems for parabolic partial differential equations, Ph.D. thesis, Escuela Politecnica Nacional, 2007.
[20] A. Antoulas, Approximation of Large-Scale Dynamical Systems, Adv. Des. Control, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005.
[21] B. Datta, Numerical Methods for Linear Control Systems Design and Analysis, Elsevier Academic Press, 2003.
[22] C. Kjelgaard-Mikkelsen, Numerical methods for large Lyapunov equations, Ph.D. thesis, Purdue University, 2009.
[23] J.R. Li, J. White, Low rank solution of Lyapunov equations, SIAM J. Matrix Anal. Appl. 24 (1) (2002) 260–280.
[24] T. Penzl, A cyclic low rank Smith method for large sparse Lyapunov equations, SIAM J. Sci. Comput. 21 (4) (2000) 1401–1418.
[25] V. Simoncini, A new iterative method for solving large-scale Lyapunov matrix equations, SIAM J. Sci. Comput. 29 (3) (2007) 1268–1288.
[26] V. Simoncini, Computational methods for linear matrix equations, Mar. 2013, preprint, Universita di Bologna.
[27] V. Simoncini, V. Druskin, L. Knizhnerman, Analysis of the rational Krylov subspace and the ADI methods for solving the Lyapunov equation, SIAM J. Numer. Anal. 49 (5) (2011) 1875–1898.
[28] B. Vandereycken, S. Vandewalle, A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations, SIAM J. Matrix Anal. Appl. 31 (5) (2010) 2553–2579.
[29] U.M. Ascher, L.R. Petzold, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, SIAM, Philadelphia, PA, 1998.
[30] P. Benner, J. Saak, Numerical solution of large and sparse continuous time algebraic matrix Riccati and Lyapunov equations: a state of the art survey, GAMM-Mitt. 36 (1) (2013) 32–52, http://dx.doi.org/10.1002/gamm.201310003.

[31] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II-Stiff and Differential Algebraic Problems, Springer Ser. Comput. Math., Springer-Verlag, New York, 2000.
[32] J.G. Blom, W. Hundsdorfer, E.J. Spee, J.G. Verwer, A second order Rosenbrock method applied to photochemical dispersion problems, SIAM J. Sci. Comput. 20 (4) (1999) 1456–1480.
[33] R.A. Horn, C.R. Johnson, Matrix Analysis, Cambridge University Press, Cambridge, 1985.
[34] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide, 3rd edition, SIAM, Philadelphia, PA, 1999.
[35] P. Benner, R.-C. Li, N. Truhar, On the ADI method for Sylvester equations, J. Comput. Appl. Math. 233 (4) (2009) 1035–1045.
[36] P. Benner, P. Kürschner, J. Saak, Efficient handling of complex shift parameters in the low-rank Cholesky factor ADI method, Numer. Algorithms 62 (2) (2013) 225–251, http://dx.doi.org/10.1007/s11075-012-9569-7.
[37] P. Benner, P. Kürschner, J. Saak, Self-generating and efficient shift parameters in ADI methods for large Lyapunov and Sylvester equations, Electron. Trans. Numer. Anal. 43 (2014) 142–162.
[38] P. Benner, P. Kürschner, J. Saak, An improved numerical method for balanced truncation for symmetric second order systems, Math. Comput. Model. Dyn. Syst. 19 (6) (2013) 593–615, http://dx.doi.org/10.1080/13873954.2013.794363.
[39] V. Druskin, V. Simoncini, Adaptive rational Krylov subspaces for large-scale dynamical systems, Systems Control Lett. 60 (8) (2011) 546–560.
[40] M. Bollhöfer, A. Eppler, Low-rank Cholesky factor Krylov subspace methods for generalized projected Lyapunov equations, in: P. Benner (Ed.), System Reduction for Nanoscale IC Design, in: Math. Ind., vol. 20, Springer International Publishing, 2016, in press.
[41] A. Locatelli, Optimal Control, Birkhäuser, Basel, Boston, Berlin, 2001.
[42] P. Benner, J. Saak, A semi-discretized heat transfer model for optimal cooling of steel profiles, in: P. Benner, V. Mehrmann, D. Sorensen (Eds.), Dimension Reduction of Large-Scale Systems, in: Lect. Notes Comput. Sci. Eng., Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 353–356.
[43] L.F. Shampine, Implementation of Rosenbrock methods, ACM Trans. Math. Software 8 (2) (1982) 93–103.
[44] J. Abels, P. Benner, CAREX – a collection of benchmark examples for continuous-time algebraic Riccati equations (version 2.0), Working Note 1999-14, SLICOT, available from www.slicot.org, Nov. 1999.
[45] T. Penzl, 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.