Max Planck Institute Magdeburg
Preprints

Norman Lang Hermann Mena Jens Saak

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

MPIMD/14-14 **July 30, 2014**

Authors addresses:

Norman Lang
Technische Universität Chemnitz,
Faculty of Mathematics,
D-09126 Chemnitz, Germany

norman.lang@mathematik.tu-chemnitz.de

Hermann Mena
University of Innsbruck,
Department of Mathematics,
A-6020 Innsbruck, Austria,

hermann.mena@uibk.ac.at

Jens Saak
Max Planck Institute for Dynamics of Complex Technical Systems,
Computational Methods in Systems and Control Theory,
D-39106 Magdeburg, Germany,
and
Technische Universität Chemnitz,
Faculty of Mathematics,
D-09126 Chemnitz, Germany

saak@mpi-magdeburg.mpg.de

Impressum:

Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg

Publisher:
Max Planck Institute for
Dynamics of Complex Technical Systems

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

http://www.mpi-magdeburg.mpg.de/preprints/

Abstract

We propose efficient algorithms for solving large-scale matrix differential equations. In particular, we deal with Riccati differential equations (RDEs) and Lyapunov differential equations (LDEs). 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 feasible the application of higher order methods. The numerical results show the better performance of the proposed methods compared to earlier formulations.

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 [14, 15, 27, 38, 39, 42, 43]. We will focus on solving Riccati differential equations (RDEs) and differential Lyapunov equations (LDE).

The RDE 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. [2, 29, 30, 41]). In the literature there is a large variety of approaches to compute the solution of small-scale RDEs (see e.g. [18, 20, 21, 33]). In this article we consider the numerical solution of large-scale RDEs arising in optimal control problems for partial differential equations. In [10] efficient numerical methods capable of exploiting this structure based on matrix-valued versions of the backward differentiation formula (BDF), Midpoint and Trapezoidal rules and the Rosenbrock (Ros) methods are proposed. Moreover, Hansen and Stillfjord [25] present an abstract framework based on operator splittings. In contrast to their work we will focus on the matrix setting.

The implementation in [10] uses a low-rank Alternating Directions Implicit (ADI) iteration 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 elaborate method has been proposed to keep the computations in real arithmetic, [37]. This yields a challenging implementation already for order 2. The ALE arises in many fields like optimal control and model order reduction [4, 19]. Many methods for solving large-scale ALEs have been proposed [34, 35, 40, 44, 45, 46, 48]. However, there have been

no attempts to solve large-scale *differential* Lyapunov equations. After discretization of the LDE in time, an ALE with special structure has to be solved in every step. If the structure of the matrix coefficients is not exploited, then it is not feasible to solve LDEs of high dimension arising in applications due to the fact that memory requirements and computational costs are quite high. So far, there is no solver in the literature that can perform this task efficiently. Numerical methods capable of exploiting the structure of the LDE based on matrix-valued version of standard ODE methods can be applied, e.g., methods based on a low-rank approximation of the solution as in [10]. Once again, if we want to apply higher order methods ($p \geq 2$) complex arithmetic is required. In this paper we restrict the presentation and experiments to the RDEs. However all our methods and techniques naturally restrict to the LDE (see comments in Section 2.2). A more detailed explanation and extensive numerical experiments will be presented elsewhere to keep the presentation within usual page limits.

In this paper we propose novel formulations of the algorithms 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. We focus on implicit methods since RDEs and LDEs, arising in applications, are often fairly stiff. We point out that the proposed methods are not restricted to solve RDE and LDE and can be applied, in principle, to any matrix differential equation. The paper is organized as follows: in Section 2 we review matrix versions of standard methods for stiff problems and their application to RDE and LDE. Further, a column compression technique for the treatment of complex data arising in the right hand side of the ALEs in the innermost iteration is proposed. In Section 3, we present the LDL^T based algorithms which are the main contribution of this paper. Then, in Section 4 we introduce some motivating examples and test our methods. 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 RDEs/LDEs 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, [11, 18, 21]. In order to optimally 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 RDE, or LDE. 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 Rosenbrock methods. In particular the BDF and Rosenbrock methods allow an efficient implementation for large-scale problems [10]. We will focus on these methods and see how the proposed ideas can be extended to other implicit methods.

2.1 Application to RDEs

Let us first consider the time-varying symmetric RDEs 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 [3]. Note that the RDE, 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 $\dot{\tilde{X}}(t_f; t) = -\dot{X}(t_f - t)$. Furthermore, considering e.g., finite element semi-discretized partial differential equation constrained optimal control problems one usually faces the generalized RDE

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

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 RDE. The latter can easily 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.

Therefore, in the remainder we will consider

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

Backward differentiation formulas: Applying the fixed-coefficients BDF method to the RDE (1) we obtain the matrix valued BDF scheme

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

where τ denotes the time step size, $t_{k+1} = t_k + \tau$, $X_{k+1} \approx X(t_{k+1})$. The expressions α_j, β denote the determining coefficients for the p-step BDF formula given in Table 1 (see,

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	$\frac{60}{147}$	$-\frac{360}{147}$	$\frac{450}{147}$	$-\frac{400}{147}$	$\frac{225}{147}$	$-\frac{72}{147}$	$\frac{10}{147}$

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

e.g. [5]). Note that the coefficients are chosen in a way such that α_0 corresponding to the current solution X_{k+1} equals 1, $\forall p = 1, \ldots, 6$. This leads to the Riccati-BDF difference equation

$$-X_{k+1} + \tau\beta(Q_{k+1} + A_{k+1}^T X_{k+1} + X_{k+1}A_{k+1} - X_{k+1}S_{k+1}X_{k+1}) - \sum_{j=1}^{p} \alpha_j X_{k+1-j} = 0$$

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

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

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 practical applications the data often is given in an low-rank representation of the form

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

In these situations one observes that the solution also is 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. In Section 3 we present a novel reformulation based on an $X_k \approx L_k D_k L_k^T$ ($L_k \in \mathbb{R}^{n\times \ell_k}, D_k \in \mathbb{R}^{\ell_k\times \ell_k}, \ell_k \ll n$) approximation and its benefits for the fast numerical computation. Using the low-rank factors (5), the ARE (4) can be written

as

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

with

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

Exploiting the sparsity of A_{k+1}, together with the low-rank representations of the constant and quadratic terms, equation (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 an RDE solver for large-scale problems. We note, the main idea here is to solve an ARE by e.g., Krylov subspace methods [26, 31, 32], using Newton's method or other methods [47] in every time step. Due to the availability of Newton based codes to the authors, here we restrict our selves to these procedures. This results in solving one algebraic Lyapunov equation in each Newton step. The coefficient matrix in this equation has the form sparse + low-rank perturbation. The special structure of the ALE can be efficiently exploited, e.g., by a low-rank version of the ADI iteration or Krylov subspace based methods, see e.g., the recent surveys [13, 45].

The implementation of the BDF methods for RDEs is sketched in Algorithm 2.1. For a detailed explanation see [10] and the references therein. For methods of order $p \geq 2$ some of the coefficients of the BDF method are negative, see Table 1. This leads to algebraic Lyapunov equations which have indefinite right hand sides and thus the right hand side factor G in Algorithm 2.1 becomes complex. This, in turn, makes complex arithmetic and storage unavoidable.

Rosenbrock methods: The application of the general p-stage Rosenbrock method, as a matrix-valued algorithm, to the RDE (1) yields

$$\begin{aligned}(\frac{1}{\tau\gamma_{ii}}I - \frac{\partial\mathcal{R}}{\partial X}(t_k, X_k))K_i = &\mathcal{R}\big(t_{k,i}, X_k + \sum_{j=1}^{i-1} a_{ij}K_j\big) + \sum_{j=1}^{i-1}\frac{c_{ij}}{\tau}K_j + \gamma_i\tau\mathcal{R}_{t_k},\\ X_{k+1} = X_k &+ \sum_{j=1}^{p} m_j K_j,\end{aligned} \tag{7}$$

where $t_{k,i} = t_k + \alpha_i\tau$, $i = 1, \ldots, p$, and γ_{ii}, a_{ij}, c_{ij}, γ_i, m_j are the method coefficients, that are avilable in text books as, e.g. [24]. We denote by K_i the $n\times n$ matrix representing the solution of the i-th-stage of the method and abbreviate $\mathcal{R}_{t_k} = \frac{\partial\mathcal{R}}{\partial t}(t_k, X(t_k))$.

Algorithm 2.1 LRF 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: (Z_{k+1}, t_{k+1}) such that $X_{k+1} \approx Z_{k+1}Z_{k+1}^T$.

$t_0 = a$.
for $k = 0$ to $\lceil \frac{b-a}{\tau} \rceil$ **do**
  $t_{k+1} = t_k + h$.
  $\hat{A}_{k+1} = \tau\beta A_{k+1} - \frac{1}{2}E_{k+1}$.
  $\hat{B}_{k+1} = \tau\beta B_{k+1}$.
  $\hat{C}_{k+1}^T = [\,\sqrt{\tau\beta}C_{k+1}^T, \sqrt{-\alpha_1}E_{k+1}^T Z_k, \ldots, \sqrt{-\alpha_p}E_{k+1}^T Z_{k+1-p}\,]$.
  for $\ell = 1$ to ℓ_{max} **do**
    $G^{(\ell)} = [\hat{C}_{k+1}^T \sqrt{\tau\beta}K^{(\ell-1)}]$.
    Compute $Z^{(\ell)}$ such that $X^{(\ell)} \approx Z^{(\ell)}{Z^{(\ell)}}^T$ is the solution of

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

    .
    $K^{(\ell)} = E_{k+1}^T Z^{(\ell)}({Z^{(\ell)}}^T B_{k+1})$.
  end for
  $Z_{k+1} = Z^{(\ell_{max})}$.
end for

The derivative $\frac{\partial\mathcal{R}}{\partial X}(t_k, X_k)$ in (7) is given by the (Frechét) derivative of $\mathcal{R}$ at X_k, represented by the Lyapunov operator

$$ \frac{\partial\mathcal{R}}{\partial X}(t_k, X_k) : U \to (A_k - S_k X_k)^T U + U(A_k - S_k X_k), $$

where $X_k \approx X(t_k)$, $A_k \equiv A(t_k)$, $S_k \equiv S(t_k)$ and $U \in \mathbb{R}^{n\times n}$.

Replacing $\frac{\partial\mathcal{R}}{\partial X}(t_k, X_k)$ by the left hand side of the first equation in (7) we obtain,

$$ \frac{1}{\tau\gamma_{ii}}K_i - (A_k - S_k X_k)^T K_i - K_i(A_k - S_k X_k), $$

and re-arranging terms yields

$$ -\left(\big(A_k - S_k X_k - \frac{1}{2\tau\gamma_{ii}}I\big)^T K_i + K_i\big(A_k - S_k X_k - \frac{1}{2\tau\gamma_{ii}}I\big)\right). \tag{8} $$

Defining

$$ \bar{A}_k := A_k - S_k X_k - \frac{1}{2\tau\gamma_{ii}}I, $$

we can, then, write (7) for $i = 1, \ldots, s$ as

$$\begin{aligned} \bar{A}_k^T K_i + K_i \bar{A}_k = &- \mathcal{R}\big(t_{k,i}, X_k + \sum_{j=1}^{i-1} a_{ij} K_j\big) \\ &- \sum_{j=1}^{i-1} \frac{c_{ij}}{\tau} K_j - \gamma_i \tau \mathcal{R}_{t_k}, \\ X_{k+1} = &X_k + \sum_{j=1}^{s} m_j K_j, \end{aligned} \tag{9}$$

Hence, in each stage of every time step of the 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 (9), as in the BDF-case, we expect 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, we exemplary illustrate a first and a second order scheme for certain choices of the determining coefficients. First, we consider the 1-stage Rosenbrock scheme (Ros1)

$$\begin{aligned} X_{k+1} &= X_k + K_1, \\ \bar{A}_k^T K_1 + K_1 \bar{A}_k &= -\mathcal{R}(X_k), \\ &= -Q_k - A_k^T X_k - X_k A_k + X_k S_k X_k \end{aligned} \tag{10}$$

with $\gamma = 1$ and $\bar{A}_k = A_k - S_k X_k - \frac{1}{2\tau} I$. With $K_1 = X_{k+1} - X_k$, rewriting the right hand side of (10) as

$$-Q_k - (A_k - S_k X_k - \frac{1}{2\tau} I)^T X_k - X_k (A_k - S_k X_k - \frac{1}{2\tau} I) - X_k S_k X_k - X_k$$

and using the low-rank factorizations of Q_k, S_k in (5) we obtain the 1-stage scheme

$$\bar{A}_k^T X_{k+1} + X_{k+1} \bar{A}_k = -C_k^T C_k - X_k^T B_k B_k^T X_k - \frac{1}{\tau} X_k, \tag{11}$$

iterating explicitly on the solution X_{k+1} of RDE (1) at time integration step $k+1$. Now, using the standard low-rank splitting $-G_k G_k^T$ of the right hand side and of the solution $X_k = Z_k Z_k^T$, respectively, we end up with Algorithm 2.2. Using a higher order Rosenbrock scheme ($p > 1$) leads to an increasing number of ALEs to be solved as shown in Equation (9). As an illustration we consider the second order Rosenbrock scheme (Ros2) proposed in [16]. Therein, the second order method is applied to autonomous atmospheric dispersion problems describing photochemistry, advective,

Algorithm 2.2 1-stage Rosenbrock for RDEs (linear implicit Euler), [37]

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: X_{k+1} such that $X_{k+1} \approx Z_{k+1}Z_{k+1}^T$.

1: **for** $k = 0$ to $\lceil \frac{b-a}{\tau} \rceil$ **do**

2: $\quad \bar{A}_k = A_k - B_k((B_k^T Z_k)Z_k^T E_k) - \frac{1}{2\tau\gamma_{ii}} E_k$

3: $\quad G_k = [C_k^T,\ E_k^T Z_k Z_k^T B_k,\ \sqrt{\frac{1}{\tau}} E_k^T Z_k]$

4: $\quad$ Compute T_1 such that $K_1 \approx T_1 T_1^T$ is the solution of

$$\bar{A}_k^T K_1 E_k + E_k K_1 \bar{A}_k = -\, G_k G_k^T.$$

5: $\quad Z_{k+1} = \left[Z_k, \sqrt{\tau} T_1\right]$

6: **end for**

and turbulent diffusive transport. The 2-stage procedure is given in the form

$$\begin{aligned} X_{k+1} &= \frac{3}{2}\tau K_1 + \frac{1}{2}\tau K_2, \\ (I - \gamma\tau \frac{\partial \mathcal{R}}{\partial X}(X_k))K_1 &= \mathcal{R}(X_k), \\ (I - \gamma\tau \frac{\partial \mathcal{R}}{\partial X}(X_k))K_2 &= \mathcal{R}(X_k + \tau K_1) - 2K_1. \end{aligned} \tag{12}$$

An extension to the non-autonomous case is presented in [37]. The given representation in [16] slightly varies from the general scheme shown in (9). Hence, using the Frechét derivative of $\mathcal{R}$ as in (8) and re-arranging the terms yields

$$\begin{aligned} X_{k+1} &= \frac{3}{2}\tau K_1 + \frac{1}{2}\tau K_2, \\ \tilde{A}_k^T K_1 + K_1 \tilde{A}_k &= -\mathcal{R}(X_k), \\ \tilde{A}_k^T K_2 + K_2 \tilde{A}_k &= -\mathcal{R}(X_k + \tau K_1) + 2K_1 \end{aligned} \tag{13}$$

with $\tilde{A}_k := \gamma\tau(A_k - S_k X_k) - \frac{1}{2}I$. Following the reformulations in [37] (13) can be simplified to

$$\begin{aligned} X_{k+1} &= \frac{3}{2}\tau K_1 + \frac{1}{2}\tau K_2, \\ \tilde{A}_k^T K_1 + K_1 \tilde{A}_k &= -\mathcal{R}(X_k), \\ \tilde{A}_k^T K_{21} + K_{21} \tilde{A}_k &= +\tau^2 K_1 B_k B_k^T K_1 + (2 - \frac{1}{\gamma})K_1 \\ K_2 &= K_{21} + (1 - \frac{1}{\gamma})K_1 \end{aligned} \tag{14}$$

Again considering the low-rank splitting given in (5) the right hand side of the first stage in (14) becomes

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

Noting that terms of the form $A_k^T Z_k Z_k^T + Z_k Z_k^T A_k$ can be split as

$$A_k^T Z_k Z_k^T + Z_k Z_k^T A_k := \left(A_k^T Z_k + Z_k\right)\left(A_k^T Z_k + Z_k\right)^T - A_k^T Z_k Z_k A_k - Z_k Z_k^T, \quad (15)$$

as explained in [37], we consider the following two possible splittings of the form $-G_k G_k^T$. A partitioning

$$G_k = \begin{bmatrix} C_k^T, & A_k^T Z_k + Z_k, & i Z_k Z_k^T B_k, & i A_k^T Z_k, & i Z_k \end{bmatrix} \quad (16)$$

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

$$\begin{aligned} \bar{A}_k^T \bar{K}_1 + \bar{K}_1 \bar{A}_k &= -N_k N_k^T \\ \tilde{A}_k^T \tilde{K}_1 + \tilde{K}_1 \tilde{A}_k &= -U_k U_k^T \end{aligned} \quad (17)$$

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

$$N_k = \begin{bmatrix} C_k^T, & A_k^T Z_k + Z_k \end{bmatrix}, \quad U_k = \begin{bmatrix} Z_k Z_k^T B_k, & A_k^T Z_k, & Z_k \end{bmatrix}.$$

The right hand side of the second stage equation of the Rosenbrock scheme (14) in standard low-rank representation with $K_1 = T_1 T_1^T$, $T_1 \in \mathbb{R}^{n \times t_k}$ reads

$$G_k = \begin{bmatrix} \tau T_1 T_1^T B, & \sqrt{2 - \frac{1}{\gamma}} T_1 \end{bmatrix}.$$

Note that the two equations in (17) share the same Lyapunov operator and therefore, recycling inner solvers leads to basically the same computational cost as for the solution of only one Lyapunov equation. The splitting (15), however, introduces additional blocks to the factors of the right hand sides. Furthermore, numerical experiments have shown that the formation of K_1 may suffer from cancellation problems in finite arithmetic. That is, constructing the solution $K_1 := \bar{K}_1 - \tilde{K}_1$ is affected by numerical inaccuracies and therefore breaks the entire second order low-rank algorithm.

Other implicit methods: As stated in [21] the application of any implicit method to the RDE yields an ARE to be solved in every step. For illustration we will exemplary consider the Midpoint and Trapezoidal rules.

The Midpoint rule applied to the RDE (1) yields

$$X_{k+1} = X_k + \tau F\left(t_k + \frac{\tau}{2}, \frac{1}{2}(X_{k+1} + X_k)\right),$$

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

$$\begin{aligned}\left[\tau Q_{k'} + X_k + \frac{\tau}{2}\left(A_{k'}^T X_k + X_k A_{k'} - \frac{X_k S_{k'} X_k}{2}\right)\right] \\ +(\frac{\tau}{2}A_{k'} - \frac{\tau}{4}S_{k'}X_k - \frac{1}{2}I)^T X_{k+1} + X_{k+1}(\frac{\tau}{2}A_{k'} - \frac{\tau}{4}S_{k'}X_k - \frac{1}{2}I) \\ - X_{k+1}(\frac{\tau}{4}S_{k'})X_{k+1} = 0,\end{aligned} \tag{18}$$

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

Applying the Trapezoidal rule to the RDE (1) we obtain

$$X_{k+1} = X_k + \frac{\tau}{2}(F(t_k, X_k) + F(t_{k+1}, X_{k+1})).$$

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

$$\begin{aligned}\left[\frac{\tau}{2}Q_{k+1} + X_k + \frac{\tau}{2}\left(Q_k + A_k^T X_k + X_k A_k - X_k S_k X_k\right)\right] \\ +\left(\frac{\tau}{2}A_{k+1} - \frac{1}{2}I\right)^T X_{k+1} + X_{k+1}\left(\frac{\tau}{2}A_{k+1} - \frac{1}{2}I\right) \\ - X_{k+1}\left(\frac{\tau}{2}S_{k+1}\right)X_{k+1} = 0,\end{aligned} \tag{19}$$

as before. That is, 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.

We are interested in solving large-scale problems and therefore the Rosenbrock methods are more attractive to apply. Moreover, most codes for solving initial value problems are intended to deal with stiff or nonstiff problems but not both. Rosenbrock methods allow to deal with this issue choosing the coefficients of the method common to the ones of an explicit Runge-Kutta method. In [23] the authors investigated this idea for a fourth order method. The result is a Rosenbrock integrator of order four that contains an embedded explicit Runge-Kutta method. It switches from one to the other solver, when the solution leaves a stiff domain and enters a nonstiff domain and vice versa. Even though it is not a simple task to determine when the solution leaves or enters a stiff domain, this idea can be efficiently exploited to integrate a certain type of problems [23]. The exploitation of both the fourth order method and the mixed solver for solving large-scale matrix differential equations will be reported somewhere else. Especially, in the context of Section 3 additional difficulties appear in the determination of the optimal representation of the low-rank formulation of the right hand sides of the single stage equations.

2.2 Application to LDEs

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

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

$$\begin{aligned} \dot{X}(t) &= Q(t) + X(t)A(t) + A^T(t)X(t) \equiv \mathcal{L}(t, X(t)), \\ X(t_0) &= X_0, \end{aligned} \tag{20}$$

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 LDE yields an algebraic Lyapunov equation to be solved in each step for X_{k+1},

$$(\tau\beta Q_{k+1} - \sum_{j=1}^{p} \alpha_j X_{k+1-j}) + (\tau\beta A_{k+1} - \frac{1}{2}I)^T X_{k+1} + X_{k+1}(\tau\beta A_{k+1} - \frac{1}{2}I) = 0 \tag{21}$$

where β, α_j are given in Table 1.
The algebraic equation (21) can be written in terms of low-rank factors similar to the Riccati case in (6). Note that the application of higher order methods will require complex arithmetic as complex numbers will arise in the factor of the constant term. This is due to the fact that Q_{k+1} and all X_{k+1-j} are positive (semi-)definite and thus the difference is in general indefinite. The same happens when Rosenbrock methods are applied to the LDE. There the problem appears when solving the algebraic Lyapunov equation corresponding to each stage of the method. In general the application of an implicit higher order method will require complex arithmetic. The latter can be avoided using the LDL^T-type algorithms which are described in Section 3.

2.3 Classical column compression

For all kinds of integration methods, explicit and implicit, the respective low-rank solution factor of either the previous or the current time step will appear in the right hand side 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 will increase drastically. 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 integration methods the right hand sides become indefinite and therefore we are faced with complex right hand side factors. This directly leads to the inadmissibility of the classic rank-revealing QR decomposition and SVD based column compressions. In the following we employ MATLAB notation to specify subblocks of a matrix and consider the ALE

$$F^T X + XF = -GG^T. \tag{22}$$

Note that the rank r in practice needs to be decided numerically or memory restrictions make a rank truncation necessary and 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

$$G_r G_r^T = U_r \Sigma_r^2 U_r^T = U_r \Sigma_r V_r^T V_r \Sigma_r U_r^T = U\Sigma V^T V \Sigma U^T = GG^T.$$

Column compression for complex data: Consider the ALE (22) to originate from a higher order ($p > 1$) time integration method. That is, the low-rank factor G of the right hand side is complex. 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 $G_r G_r^T$. This yields

$$G_r G_r^T = \Pi R_r^H \bar{R}_r \Pi^T \neq \Pi R_r^H Q_r^H Q_r R_r \Pi^T.$$

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

$$G_r G_r^T = U_r \Sigma_r V^H \bar{V} U_r^T \neq U_r \Sigma_r V^H V \Sigma_r U_r^T = U_r \Sigma_r^2 U_r^T.$$

Clearly using $G \in \mathbb{C}^{n\times k}$ 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 $G = QR$ with $Q \in \mathbb{C}^{n\times k}$, $Q^HQ = I_k$, $R \in \mathbb{C}^{k\times k}$.

ii) Compute a decomposition $RR^T = V\Lambda V^T$ 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 $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_k|$.

iii) Set the compressed factor $G_r := QV_r\Lambda_r^{\frac{1}{2}} \in \mathbb{C}^{n\times r}$, where $r \leq k$ and $|\lambda_{r+1}| \leq \varepsilon$.

Following the statements in [28, Theorem 4.4.13], the existence of the desired form $V^TV = I_k$ is guaranteed, since RR^T diagonalizable. This directly follows from the fact that GG^T is real symmetric and therefore a decomposition

$$GG^T = U\Lambda U^T$$

with diagonal Λ and an orthogonal U exists. That is, using Step i) from the above procedure we obtain

$$U\Lambda U^T = GG^T = QRR^TQ^T \Leftrightarrow RR^T = Q^HU\Lambda U^T\bar{Q}.$$

Note that the eigendecomposition (e.g. in MATLAB) for the complex symmetric matrix RR^T within Step ii) in general leads to

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

with eigenvectors $\tilde{v}_i \in \mathbb{C}^k$ (i.e. columns in $\tilde{V}$) satisfying the properties

$$\begin{aligned} \|\tilde{v}_i\|_2 = \tilde{v}_i^*\tilde{v}_i = 1, \qquad & \tilde{v}_i^T\tilde{v}_i \neq 1, \\ \tilde{v}_i^*\tilde{v}_j \neq 0, \qquad & \tilde{v}_i^T\tilde{v}_j = 0. \end{aligned} \tag{23}$$

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 (23), we have

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

Defining $V := \tilde{V}\tilde{D}$ with $\tilde{D} = \operatorname{diag}\left(\frac{1}{\sqrt{\tilde{v}_1^T\tilde{v}_1}}, \ldots, \frac{1}{\sqrt{\tilde{v}_k^T\tilde{v}_k}}\right)$ 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 $\tilde{v}_i$ via $\sqrt{\tilde{v}_i^T \tilde{v}_i},\ i = 1, \ldots, k$ 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 low-rank methods for the solution of the RDE.

3 LDL^T-type Lyapunov solvers

It is necessary to solve an ALE of dimension n of the form

$$F^T X + XF = -W \tag{24}$$

with an indefinite matrix W in every step of either the Rosenbrock method or the Newton method within the BDF scheme, the Midpoint or Trapezoidal rule to solve the RDE. 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 to in the form GSG^T with $G \in \mathbb{R}^{n \times k}, k \ll n$ and a small but indefinite matrix $S \in \mathbb{R}^{k \times k}$.

3.1 LDL^T-type ADI

Following the theory in [9] the one step iteration at step j of the ADI method becomes

$$\begin{aligned} L_{j+1} D_{j+1} L_{j+1}^T = &-2\mathrm{Re}(\mu_j)(F^T + \mu_j I)^{-1} G S G^T (F + \bar{\mu}_j I)^{-1} \\ &+ (F^T + \mu_j I)^{-1}(F^T - \mu_j I) L_j D_j L_j^T (F - \mu_j I)(F + \mu_j I)^{-1}. \end{aligned} \tag{25}$$

Using the inherent structure of (25) the factors L_{j+1}, D_{j+1} can be computed as follows:

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

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

$$\begin{aligned} L_{j+1} &= \left[R_{j+1}G,\, R_{j+1}T_{j+1}R_jG,\, \ldots,\, R_{j+1}T_{j+1}\ldots R_2T_2R_1G\right], \\ D_{j+1} &= \begin{bmatrix} -2\mathrm{Re}(\mu_{j+1})S & & & \\ & -2\mathrm{Re}(\mu_j)S & & \\ & & \ddots & \\ & & & -2\mathrm{Re}(\mu_1)S \end{bmatrix}. \end{aligned} \tag{26}$$

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

$$\begin{aligned} L_{j+1} &= \left[R_1G,\, R_2T_1(R_1G),\, \ldots,\, R_{j+1}T_j(R_jT_{j-1}\ldots R_2T_1R_1G)\right], \\ D_{j+1} &= \begin{bmatrix} -2\mathrm{Re}(\mu_1)S & & & \\ & -2\mathrm{Re}(\mu_2)S & & \\ & & \ddots & \\ & & & -2\mathrm{Re}(\mu_{j+1})S \end{bmatrix} \\ &= -2\mathrm{diag}\left(\mathrm{Re}(\mu_1), \ldots, \mathrm{Re}(\mu_{j+1})\right) \otimes S \end{aligned} \tag{27}$$

in complete analogy to the procedure first employed by Li and White for the ZZ^T case in [35]. Thus, the LDL^T-based factorization does not differ to much from the low-rank factored ADI as proposed in [6, 8, 7].

The introduction of the potentially indefinite matrix S in the decomposition of the right hand side immediately avoids the necessity for 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 diagonal matrix S and the shift sequence which is done during the ADI anyway. Since the low-rank factors L and Z are computed by the same iteration sequence 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 $W_{j-1}SW_{j-1}^T \in \mathbb{R}^{n\times n}$ and $W_{j-1}^TW_{j-1}S \in \mathbb{R}^{k\times k}$ 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

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

Algorithm 3.1 LDL^T-factorization based ADI method

INPUT: ADI shifts $\mu_1, \ldots, \mu_\ell$, 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
$D_j = -2\operatorname{diag}(\operatorname{Re}(\mu_1), \ldots, \operatorname{Re}(\mu_j)) \otimes S$

3.2 LDL^T-type Krylov subspace method

Following the statements in [22, 44] the rational Krylov subspace method (RKSM) and the exteded Krylov subspace method (EKSM) compute a solution

$$X_s = V_s Y_s V_s^T \tag{28}$$

of the ALE (24) 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

$$\mathcal{K}_s(F, \hat{G}, p) = \{\hat{G}, (F^T - \mu_1 I)^{-1}\hat{G}, \ldots, \prod_{j=1}^{s}(F^T - \mu_j I)^{-1}\hat{G}\} \subset \mathbb{R}^{n \times sk} \text{ or}$$
$$\mathcal{K}_{2s}(F, F^{-s}\hat{G}) = \{F^{-s}\hat{G}, \ldots, F^{-1}\hat{G}, \hat{G}, F\hat{G}, \ldots, F^s\hat{G}\} \subset \mathbb{R}^{n \times 2sk},$$

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

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

That is, the RKSM and EKSM Lyapunov solvers directly compute the solution of (24) 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 constructing the splitting $\hat{G}\hat{G}^T$ out of the given right hand side GSG^T can be performed efficiently. Using a column compression

technique in order to reduce the block size of G and S will result in a diagonal block S_r (see Section 3.4). Therefore, we easily obtain $\hat{G}_r = G_r S_r^{\frac{1}{2}}$ with

$$\hat{G}_r \hat{G}_r^T \approx \hat{G}\hat{G}^T = GSG^T.$$

Note that using the Krylov subspace based methods the advantage of the LDL^T representation of solutions solely is the avoidance of the additional computation of an artificial ZZ^T factorization of the solution X_s.

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. For simplicity, we restrict ourselves to the autonomous case. For details on non-autonomous ODEs see e.g., [11, 37]. In particular, we will demonstrate the advantageous of the LDL^T-type splitting on the example of a general p-step BDF method, as well as for the first and second order Rosenbrock schemes presented in e.g., [11, 37] and the references therein. Further, we restrict the following illustration to the application of the above mentioned methods to the RDE.

Backward differentiation formulas: Again, consider the ARE

$$\begin{aligned}(\tau\beta C_{k+1}^T C_{k+1} - \sum_{j=1}^{p} \alpha_j X_{k+1-j}) + \hat{A}_{k+1}^T X_{k+1} + X_{k+1}\hat{A}_{k+1} \\ - X_{k+1}(\tau\beta B_{k+1}B_{k+1}^T)X_{k+1} = 0\end{aligned} \tag{29}$$

arising in the BDF method at time integration step $k+1$. Applying Newton's method to the ARE results in the solution of the ALE

$$\begin{aligned}(\hat{A}_{k+1} - \tau\beta B_{k+1}B_{k+1}^T X_{k+1}^{(\ell-1)})^T X_{k+1}^{(\ell)} + X_{k+1}^{(\ell)}(\hat{A}_{k+1} - \tau\beta B_{k+1}B_{k+1}^T X_{k+1}^{(\ell-1)}) \\ = -(\tau\beta C_{k+1}^T C_{k+1} - \sum_{j=1}^{p} \alpha_j X_{k+1-j}) - X_{k+1}^{(\ell-1)}(\tau\beta B_{k+1}B_{k+1}^T)X_{k+1}^{(\ell-1)}\end{aligned} \tag{30}$$

for $X_{k+1}^{(\ell)}$ in the ℓ-th Newton step. The p-step BDF coefficients $\alpha_j, j = 1, \ldots, p$ lead to an indefinite constant term in the ALE (30) for all schemes of order $p \geq 2$, see Table 1. That is, the solution of (29) via Newton's method, in particular the application of the inner solver to the ALE (30), needs to deal with complex arithmetic and data. Now, using the factorization $X_{k+1} := L_{k+1}D_{k+1}L_{k+1}^T$ instead of the standard low-rank

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

representation of the solution of the RDE, Algorithm 2.1 changes to Algorithm 3.2. That is, using the LDL^T factorization and the associated splitting GSG^T of the right hand side of (30) allows to put the coefficients $\alpha_j, j = 1, \ldots, p$ into the diagonal blocks of S. This avoids taking the square root of the non-positive α_j (see Table 1) and in turn removes complex data and arithmetic.

As mentioned in Section 2 the Midpoint or Trapezoidal rule also lead to the solution of an ARE in every time integration step. Having a closer look at the corresponding Equations (18) and (19), 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, where it can be treated the same way.

First order Rosenbrock method (linear implicit Euler): As given in Algorithm 2.2 the first order Rosenbrock scheme in standard low-rank formulation deals with the right hand side

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

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 $\tilde{G}_k \tilde{S}_k \tilde{G}_k^T$

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

for the solution factorization $X_k = L_k D_k L_k^T$ will improve the numerical computations, since re-arranging the blocks in the form

$$\begin{aligned} \tilde{G}_k &= \left[C_k^T, \quad L_k\right] && \in \mathbb{R}^{n \times q+z_k} \\ \tilde{S}_k &= \begin{bmatrix} I & \\ & D_k L_k^T B_k B_k L_k D_k + \frac{1}{\tau} D_k \end{bmatrix} && \in \mathbb{R}^{q+z_k \times q+z_k} \end{aligned} \tag{31}$$

leads to a factor $\tilde{G}_k$ of size $n \times q + z_k$ 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. That is, 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 $m \cdot n_{lyap} \cdot n_{ODE}$ less linear system solves during the solution of the RDE (1) 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 $z_k \times m$ and therefore do not require a significant amount of computation time as long as $z_k, m \ll n$, which is a required assumption for low-rank computations anyway.

Second order Rosenbrock method: As introduced in Equations (16) and (17) for the first stage equation of the second order method we either have to deal with the complex right hand side

$$G_k = \begin{bmatrix} C_k^T, & A_k^T Z_k + Z_k, & iZ_k Z_k^T B_k, & iA_k^T Z_k, & iZ_k \end{bmatrix}, \quad \in \mathbb{R}^{n \times q+m+3z_k}$$

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

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

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 (15), again, we consider the LDL^T-type splitting. Hence, the right hand side of the first stage equation becomes

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

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

$$\begin{aligned} \tilde{G}_k &= \begin{bmatrix} C_k^T, & A_k^T L, & L_k, & L_k \end{bmatrix}, \\ \tilde{S}_k &= \begin{bmatrix} I_q & & & \\ & D_k & & \\ & & D_k & \\ & & & D_k L_k^T B_k B_k^T L_k D_k \end{bmatrix}. \end{aligned} \tag{32}$$

Re-arranging blocks, similar to (31), leads to

$$\begin{aligned} \tilde{G}_k &= \begin{bmatrix} C_k^T, & A_k^T L_k, & L_k \end{bmatrix} && \in \mathbb{R}^{n \times q+2z_k}, \\ \tilde{S}_k &= \begin{bmatrix} I_q & & \\ & & D_k \\ & D_k & D_k L_k^T B_k B_k^T L_k D_k \end{bmatrix} && \in \mathbb{R}^{q+2z_k \times q+2z_k}. \end{aligned} \tag{33}$$

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

Furthermore, the right hand side of the second stage equation of the Rosenbrock scheme (14) in standard low-rank representation with the factorization $K_1 = T_1 T_1^T$, $T_1 \in \mathbb{R}^{n \times t_k}$ of the first stage solution reads

$$G_k = \begin{bmatrix} \tau T_1 T_1^T B, & \sqrt{2 - \frac{1}{\gamma}} T_1 \end{bmatrix} \in \mathbb{R}^{n \times m+t_k}.$$

Now, using the LDL^T-type splitting with $K_1 = T_1 D_1 T_1^T$ we obtain

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

For the solution of the second stage equation we save another m linear system solves.

In total this leads to saving $2m + z_k$ 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, also increases. On the one hand, the number of system solves will increase as well. On the other hand, analogous block re-arrangements will lead to similar savings per stage. 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 RDE solutions. Consider the matrix GSG^T, where $G \in \mathbb{R}^{n \times k}, S \in \mathbb{R}^{k \times k}$. Following the statements in Section 6.3.3 in [17] 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 $RSR^T = V \Lambda V^T$ with $V \in \mathbb{R}^{k \times k}$ and a diagonal matrix $\Lambda \in \mathbb{R}^{k \times k}$ with diagonal entries $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_k|$.

iii) Set the compressed factors $G_r := QV_r \in \mathbb{R}^{n \times r}, S_r := \Lambda_r$ with $r \leq k$ and $|\lambda_{r+1}| \leq \varepsilon$.

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

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

$$\begin{aligned} \min_u \; J(t;\, y, u) &= \int_{t_0}^{t_f} y^T(t) Q y(t) + u^T(t) R u(t) \; dt + y(t_f)^T M y(t_f), \\ \text{s.t.} \quad E\dot{x}(t) &= Ax(t) + Bu(t), \\ y &= Cx(t) \end{aligned} \tag{34}$$

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 RDE. The optimal solution to (34) is given by the feedback law (e.g., [36])

$$u = -R^{-1} B^T X(t) E x(t) = -K(t) x(t). \tag{35}$$

This means, in order to compute the optimal solution u of (34), we need to find a matrix valued function $X(t)$, which is given as the solution of the generalized RDE

$$\begin{aligned} E^T \dot{X} E &= Q + A^T X E + E^T X A - E^T X B B^T X E \\ E^T X(t_f) E &= 0 \end{aligned} \tag{36}$$

Note that the RDE arising from an LQR problem has to be solved backwards in time. Therefore, the following results, in particular the convergence behavior of the RDE to the ARE needs to be interpreted starting from the end point of the corresponding time interval.

4.1 ADI based Lyapunov solvers

The following examples show the evolution of one component of the feedback matrix $K = -R^{-1} B^T X(t) E$ in equation (35), where $X(t)$ is the solution of the RDE (36) computed via an ADI iteration based Lyapunov solver inside the time integration schemes.

4.1.1 Example 1: Steel profile

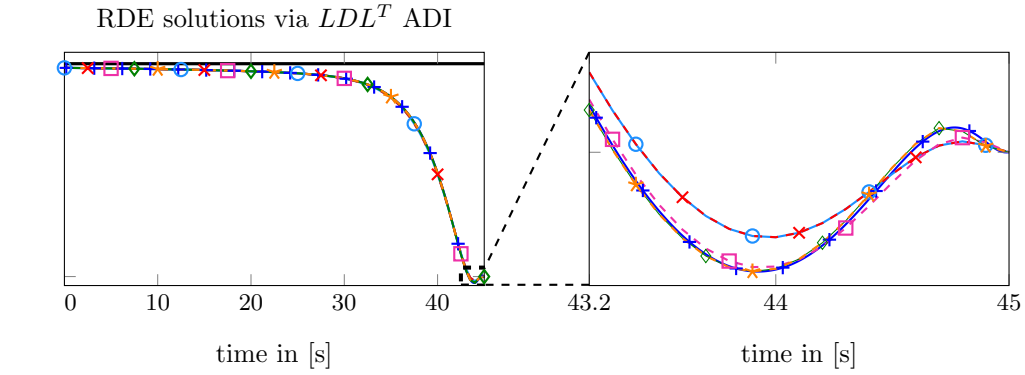
We consider the semi-discretized heat transfer model described in [12]. The model is given with $m = 7$ inputs and $q = 6$ outputs. The solution is computed on the time interval $[0, 45]s$.

In order to be able to compare the results of the different low rank RDE solvers to a classical dense 4th order Rosenbrock scheme (Ros4), we start with the smallest

available state space dimension $n = 371$. Figure 1 shows component $K_{1,77}(t)$ of the reference solution computed via the Ros4 with fixed time step size $\tau = 1e-4$ compared to the LDL^T based solutions of the BDF method of order $p = 1$, 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. Table 2 presents the timings for the different methods, as well as the relative error between the classical low-rank methods and the LDL^T based algorithms. From that we can see that the LDL^T based RDE solvers achieve a speed-up up to a factor of almost 4 for the second order Rosenbrock method. Note that the rather small time saving in the case of the first order 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 numerical errors. Still, there seems to be a problem with the Ros2 method. This has to be further investigated in the future.

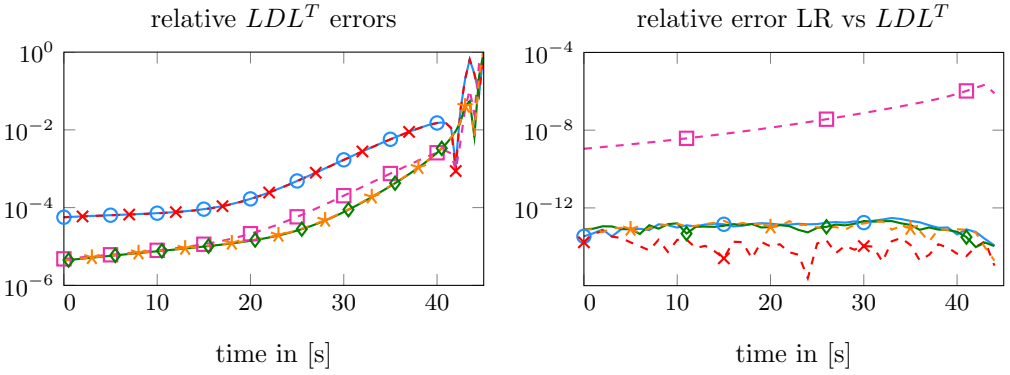
	time in s		avg. rel. err.
	LR	LDL	LRvsLDL
BDF1	1 909.53	1 299.17	1.87e-14
Ros1	845.74	658.11	1.85e-15
Ros2	4 514.19	1 242.30	2.97e-08
Midpoint	2 598.54	1 494.33	1.50e-14
Trapezoidal	2 602.14	1 180.55	1.53e-14

Table 2: Timings, average relative errors between standard low-rank and the LDL^T based methods of the steel profile example with $n = 371$ on the time interval $[0, 45]$ s, τ=1e-1.



(a) Convergence behavior of solution component $K_{1,77}$ on the time interval $[0, 45]$ s.

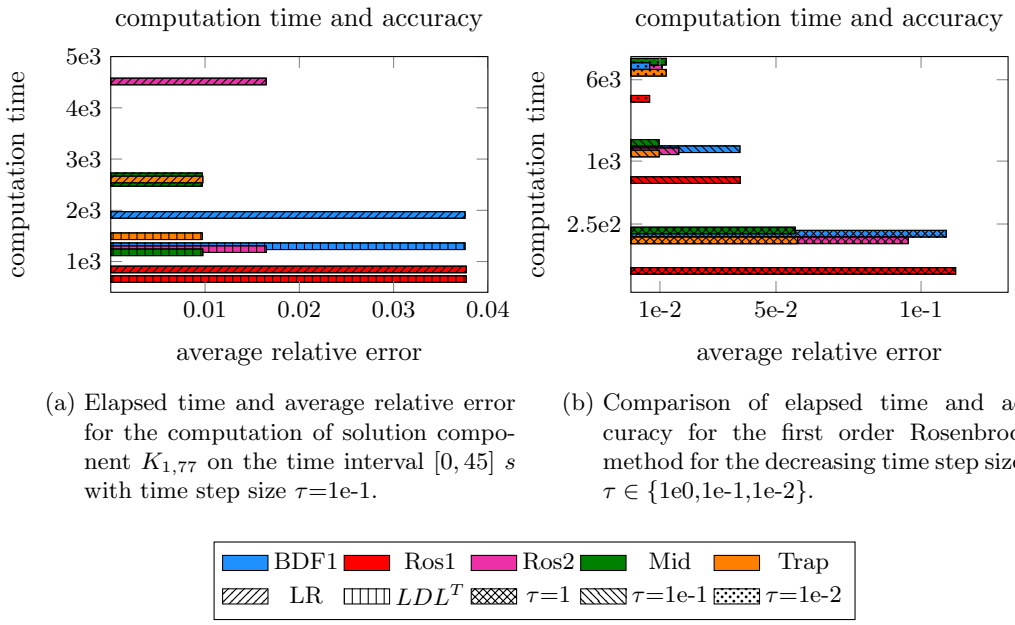
(b) Zoom into the time interval $t \in [43.2, 45]$ s of Figure 1a: Deviation of 1st and 2nd order methods



(c) Relative error of the LDL^T based methods compared to the reference solution.

(d) Relative error between the classical low-rank and the LDL^T based methods.

Figure 1: Comparison of the dense $4th$ order Rosenbrock reference solution computed with step size τ=1e-4 and the LDL^T BDF method of order 1, the Midpoint and Trapezoidal rules and the Rosenbrock methods of order p=1,2 computed with a fixed step size τ=1e-1.



(a) Elapsed time and average relative error for the computation of solution component $K_{1,77}$ on the time interval $[0, 45]$ s with time step size τ=1e-1.

(b) Comparison of elapsed time and accuracy for the first order Rosenbrock method for the decreasing time step sizes $\tau \in \{1e0,1e-1,1e-2\}$.

Figure 2: Accuracy investigations with respect to the time integration method 2a and decreasing time step size τ 2b.

Figure 2 presents some accuracy results with respect to the chosen time integration methods and the time step size. In Figure 2a the comparison of the computation times and the achieved accuracy is given for the first order BDF, first and second order Rosenbrock methods, the Midpoint and Trapezoidal rules for both, the classical low-rank and the LDL^T based integration schemes. According to Table 2, we also observe the superiority of the LDL^T based methods with respect to the computation times. Figure 2b shows the increasing accuracy for decreasing time step sizes τ of the LDL^T based algorithms. We observe that reducing the time step size increases the overall computation time while the accuracy of the resulting solution increases, as expected. We also realize that the average relative errors for the Midpoint and Trapezoidal rules for τ=1e-2 is slightly larger than the one for τ=1e-1. For now, the reason for this is obscure.

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. The savings of the system solves within the LDL^T based scheme (see Section 3.3) are approximately compensated by the column compression technique of lower computational cost (see Section 2.3) for the standard low-rank splitting of the real definite right

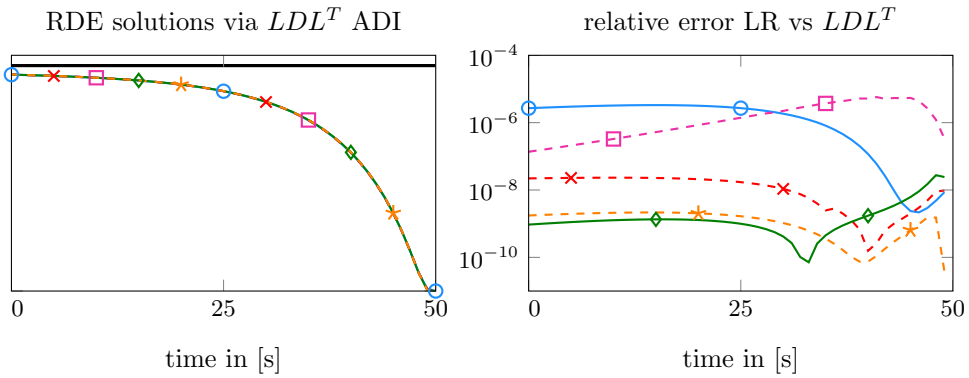
	time in s		avg. rel. err.
	LR	LDL	LRvsLDL
BDF1	1 682.59	1 409.83	2.76 e-15
Ros1	864.19	857.93	9.89 e-15
Ros2	12 796.24	1 788.25	2.48 e-07
Midpoint	2 625.05	1 524.23	1.28 e-15
Trapezoidal	2 652.60	1 316.43	6.28 e-15

Table 3: Timings, the average relative errors between low-rank and LDL^T methods for $K_{1,77}$ of the steel profile with n=1 357 on the time interval $[0, 45]$ s, τ=1.

hand sides arising in the first order Rosenbrock method. Using a higher order method, as one can easily observe in the case of Ros2, the LDL^T routines can benefit from all their advantages, i.e., avoiding the complex data and the removing of redundant information by re-arranging the middle block S of the right hand sides. Therefore, the LDL^T version achieves a significant time saving.

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 we have $m = 1$ input and $q = 9$ outputs. Similar to the above example Figure 3a shows the convergence behavior of the solutions of the different time integration methods to the solution of the ARE. Furthermore, Figure 3b depicts the relative errors of the solutions of the LR methods compared to the LDL^T results.



(a) Convergence behavior of solution component $K_{1,11}$ on the time interval $[0, 50]$ s.

(b) Relative error between the classical low-rank and the LDL^T based methods.

Figure 3: Comparison of the LDL^T BDF method of order 1, the Midpoint and Trapezoidal rules and the Rosenbrock methods of order p=1,2 computed with a fixed step size τ=1e-1.

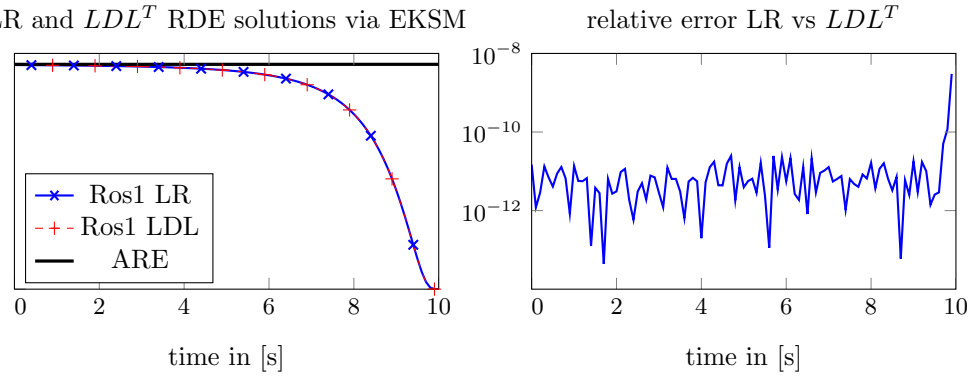
Table 4 shows the timings of the standard low-rank algorithms and the LDL^T based schemes, as well as the average relative errors between both. Here we observe that in the case of the first order Rosenbrock method the classical low-rank version computes the solution of the RDE in less time compared to the LDL^T scheme. The saving of $m = 1$ system solves, given by the single input system, within every ADI step at each time integration step cannot counterbalance the additional computational effort of the column compression for the LDL^T factorization.

	time in s		avg. rel. err.
	LR	LDL	LRvsLDL
BDF1	10 255.16	6 369.10	9.12 e-07
Ros1	3 574.56	3 959.63	7.98 e-09
Ros2	12 876.42	6 946.71	8.99 e-07
Midpoint	9 968.02	6 636.28	2.14 e-09
Trapezoidal	9 939.94	5 540.53	1.10 e-09

Table 4: Timings, average relative errors between low-rank and LDL^T methods for $K_{1,11}$ of the diffusion model with n=1 089 on the time interval $[0, 50]$ s, τ=1e-1.

	time in s		avg. rel. err.
	LR	LDL	LRvsLDL
Ros1	421.98	254.57	2.05e-08

Table 5: Timings, average relative error between low-rank and LDL^T method for $K_{1,1}$ of the carex model with n=1 000 on the time interval [0,10], τ=1e-2.



(a) Convergence behavior of solution component $K_{1,1}$ on the time interval $[0, 10]$ s.

(b) Relative error between the classical low-rank and the LDL^T based methods.

Figure 4: Comparison of the standard low-rank and LDL^T Rosenbrock method of order 1, computed with a fixed step size τ=1e-2.

4.2 Krylov based Lyapunov solvers

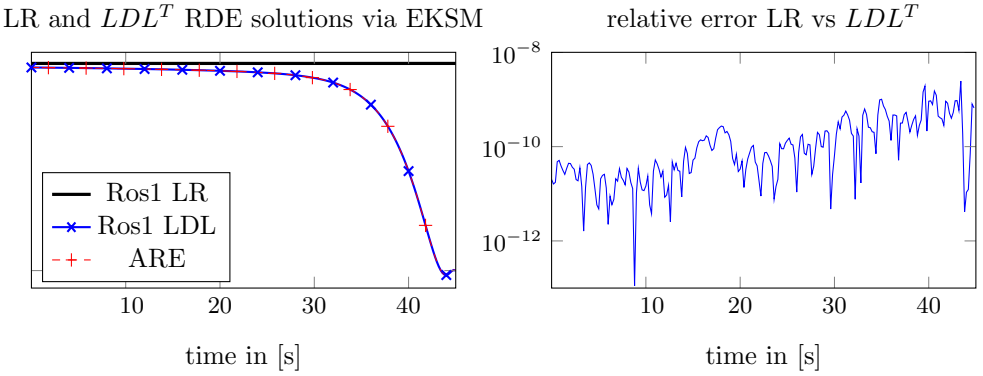
The following example presents the numerical results achieved for an EKSM based Lyapunov solver.

4.2.1 Example 3: carex model

The third example originates from the CAREX benchmark collection for continuous-time algebraic Riccati equations [1]. The model is a single-input-single-output (SISO) system with $E, A \in \mathbb{R}^{n\times n}$, $B = C^T \in \mathbb{R}^n$ and $n = 1\,000$. Figure 4 shows component $K_{1,1}$ computed via the classical low-rank Ros1 scheme compared to the LDL^T based version. The corresponding computation times and the average relative error are presented in Table 5.

	time in s		avg. rel. err.
	LR	LDL	LRvsLDL
Ros1	197.73	118.58	2.03e-10

Table 6: Timings, average relative errors between standard low-rank and the LDL^T based methods of the steel profile example with $n = 371$ on the time interval $[0, 45]$ s, τ=1e-1.



(a) Convergence behavior of solution component $K_{1,77}$ on the time interval $[0, 45]$ s.

(b) Relative error between the classical low-rank and the LDL^T based methods.

Figure 5: Comparison of the standard low-rank and LDL^T Rosenbrock method of order 1, computed with a fixed step size τ=1e-2.

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 $n = 371$, τ=1e-1 on the time interval $[0, 45]s$. Similar to Example 4.2.1 Figure 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 is given and shows the equality of the algorithms except for numerical deviations.

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 RDE 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 RDE. Our main contribution is the presentation of an LDL^T based decomposition of the solution of the RDE and the right hand side of the arising ALEs. This special type of factorization naturally avoids all of the aforementioned problems.

The theoretical stated advantages have been numerically validated for a number of examples. Here we compared the accuracy, as well as the computational timings of the classical low-rank and 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 can not compensate the extra cost of the column compression for the LDL^T decompositions. In Section 3.3 we showed that this directly depends on the number of inputs of the underlying state-space system as we also observe from the multiple input Example in Section 4.1.1 and the single input Example 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. A more direct comparison of the ADI and EKSM based Lyapunov solvers inside the time integration methods is postponed and will be reported somewhere else. This is due to the fact that the currently running experimental codes will be further optimized in the future and at the same time the comparability, e.g., with respect to comparable stopping criteria, at the most efficient level needs to be guaranteed.

References

[1] J. Abels and P. Benner, *CAREX – a collection of benchmark examples for continuous-time algebraic Riccati equations (version 2.0)*, SLICOT Working Note 1999-14, Faculty 3, Mathematics, Universitt Bremen, Nov. 1999. Available from www.slicot.org.

[2] H. Abou-Kandil, G. Freiling, V. Ionescu, and G. Jank, *Matrix Riccati Equations in Control and Systems Theory*, Birkhäuser, Basel, Switzerland, 2003.

[3] ——, *Matrix Riccati Equations in Control and Systems Theory*, Birkhäuser, Basel, Switzerland, 2003.

[4] A. Antoulas, *Approximation of Large-Scale Dynamical Systems (Advances in*

Design and Control), Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005.

[5] U. Ascher and L. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, PA, SIAM, Philadelphia, 1998.

[6] P. Benner, P. Kürschner, and J. Saak, *Efficient Handling of Complex Shift Parameters in the Low-Rank Cholesky Factor ADI Method*, Numerical Algorithms, 62 (2013), pp. 225–251. 10.1007/s11075-012-9569-7.

[7] P. Benner, P. Kürschner, and J. Saak, *An improved numerical method for balanced truncation for symmetric second order systems*, Math. Comput. Model. Dyn. Syst., 19 (2013), pp. 593–615.

[8] P. Benner, P. Kürschner, and J. Saak, *Self-generating and efficient shift parameters in ADI methods for large Lyapunov and Sylvester equations*, Preprint MPIMD/13-18, Max Planck Institute Magdeburg, October 2013. Available from http://www.mpi-magdeburg.mpg.de/preprints/.

[9] P. Benner, R.-C. Li, and N. Truhar, *On the ADI method for Sylvester equations*, J. Comput. Appl. Math., 233 (2009), pp. 1035–1045.

[10] P. Benner and H. Mena, *Numerical solution of the infinite-dimensional LQR-problem and the associated differential Riccati equations*, Tech. Rep. MPIMD/12-13, MPI Magdeburg Preprint, 2012.

[11] P. Benner and H. Mena, *Rosenbrock methods for solving differential Riccati equations*, IEEE Transactions on Automatic Control, 58 (2013), pp. 2950–2957.

[12] P. Benner and J. Saak, *A semi-discretized heat transfer model for optimal cooling of steel profiles*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D. Sorensen, eds., Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 353–356.

[13] ——, *Numerical solution of large and sparse continuous time algebraic matrix Riccati and Lyapunov equations: a state of the art survey*, GAMM Mitteilungen, 36 (2013), pp. 32–52.

[14] P. Benner, Z. Tomljanović, and N. Truhar, *Optimal damping of selected eigenfrequencies using dimension reduction*, Numerical Linear Algebra with Applications, 20 (2013), pp. 1–17.

[15] F. Blanchini, D. Casagrende, P. Gardonio, and S. Miani, *Constant and switching gains in semiactive damping of vibrating structures*, International Journal of Control, 85 (2012), pp. 1886–1897.

[16] J. Blom, W. Hundsdorfer, E. Spee, and J. Verwer, *A second order Rosenbrock method applied to photochemical dispersion problems*, SIAM J. Sci. Comput., 20(4) (1999), pp. 1456–1480.

[17] M. Bollhöffer and A. Eppler, *Low-rank Cholesky factor Krylov subspace methods for generalized projected Lyapunov equations*, in System Reduction for Nanoscale IC Design, P. Benner, ed., vol. 20 of Mathematics in Industry, Springer-Verlag, Berlin/Heidelberg, Germany, 2014. in press.

[18] C. Choi and A. Laub, *Efficient matrix-valued algorithms for solving stiff Riccati differential equations*, IEEE Trans. Automat. Control, 35 (1990), pp. 770–776.

[19] B. Datta, *Numerical Methods for Linear Control Systems Design and Analysis*, Elsevier Academic Press, 2003.

[20] E. Davison and M. Maki, *The numerical solution of the matrix Riccati differential equation*, IEEE Trans. Automat. Control, 18 (1973), pp. 71–73.

[21] L. Dieci, *Numerical integration of the differential Riccati equation and some related issues*, SIAM J. Numer. Anal., 29(3) (1992), pp. 781–815.

[22] V. Druskin and V. Simoncini, *Adaptive rational Krylov subspaces for large-scale dynamical systems*, Systems and Control Letters, 60 (2011), pp. 546–560.

[23] A. Freed and I. Iskovitz, *Development and applications of a Rosenbrock integrator*, Tech. Rep. NASA Tech. Memorandum 4709, Lewis Research Center, Cleveland, Ohio, 1996.

[24] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II-Stiff and Differerntial Algebraic Problems*, Springer Series in Computational Mathematics, Springer-Verlag, New York, 2000.

[25] E. Hansen and T. Stillfjord, *Convergence analysis for splitting of the abstract differential riccati equation*, Preprint 4363490, Lund University, 2014. Preprint.

[26] M. Heyouni and K. Jbilou, *An extended block Arnoldi algorithm for large-scale solutions of the continuous-time algebraic Riccati equation*, Electr. Trans. Num. Anal., 33 (2009), pp. 53–62.

[27] J. Hoepffner, *Stability and control of shear flows subject to stochastic excitations*, PhD thesis, KTH Royal Institute of Technology, Sweden, 2006.

[28] R. Horn and C. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1985.

[29] A. Ichikawa and H. Katayama, *Remarks on the time-varying H_∞ Riccati equations*, Sys. Cont. Lett., 37(5) (1999), pp. 335–345.

[30] O. Jacobs, *Introduction to Control Theory*, Oxford Science Publication, Oxford, 2nd ed., 1993.

[31] K. Jbilou, *Block Krylov subspace methods for large algebraic Riccati equations*, Numer. Algorithms, 34 (2003), pp. 339–353.

[32] K. Jbilou, *An arnoldi based algorithm for large algebraic Riccati equations*, tech.

rep., L.M.P.A., Jan. 2006.

[33] C. Kenney and R. Leipnik, *Numerical integration of the differential matrix Riccati equation*, IEEE Trans. Automat. Control, AC-30 (1985), pp. 962–970.

[34] C. Kjelgaard-Mikkelsen, *Numerical methods for large Lyapunov equations*, PhD thesis, Purdue University, 2009.

[35] J. Li and J. White, *Low rank solution of Lyapunov equations*, SIAM J. Matrix Anal. Appl., 24(1) (2002), pp. 260–280.

[36] A. Locatelli, *Optimal Control*, Birkhäuser, Basel, Boston, Berlin, 2001.

[37] H. Mena, *Numerical Solution of Differential Riccati Equations Arising in Optimal Control Problems for Parabolic Partial Differential Equations*, PhD thesis, Escuela Politecnica Nacional, 2007.

[38] C. Penland, M. Fluegel, and P. Chang, *The role of stochastic forcing in modulating ENSO predictability*, J. Climate, 17 (2004), pp. 3125–3140.

[39] C. Penland and P. D. Sardeshmukh, *The optimal growth of tropical sea surface temperature anomalies*, J. Climate, 8 (1995), pp. 1999–2024.

[40] T. Penzl, *A cyclic low rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 21 (2000), pp. 1401–1418.

[41] I. R. Petersen, V. A. Ugrinovskii, and A. V.Savkin, *Robust Control Design Using H^∞ Methods*, Springer-Verlag, London, UK, 2000.

[42] H. Sandberg, *Model Reduction for Linear Time-Varying Systems*, PhD thesis, Lund Institute of Technology, Sweden, 2004.

[43] ——, *A case study in model reduction of linear time-varying systems*, Automatica, 43 (2006), pp. 467–472.

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

[45] ——, *Computational methods for linear matrix equations*, tech. rep., Universita di Bologna, Preprint, March 2013.

[46] V. Simoncini, V. Druskin, and L. Knizhnerman, *Analysis of the rational Krylov subspace and the ADI methods for solving the Lyapunov equation*, SIAM J. Numer. Anal., 49(5) (2011), pp. 1875–1898.

[47] V. Simoncini, D. B. Szyld, and M. Monsalve, *On the numerical solution of large-scale Riccati equations*, IMA J. Numer. Anal., (2013, to appear).

[48] B. Vandereycken and S. Vandewalle, *A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 2553–2579.