Fuzzy Systems and Data Mining IX
A.J. Tallón-Ballesteros and R. Beltrán-Barba (Eds.)

doi:10.3233/FAIA231043

Low-Rank Exponential Integrators for Differential Riccati Equation

Jingyi Li [a] and Dongping Li [a,b,1]
[a] *Department of Mathematics, Changchun Normal University, Changchun 130032, PR China*
[b] *Department of Mathematics, Jilin University, Changchun 130012, PR China*

Abstract. Differential Riccati equation (DRE) is especially important in several fields such as optimal control, filtering, and model reduction. In this paper we present matrix-based exponential integrators for the solution of DRE. The methods are suitable for stiff DRE. In particular, we develop low-rank versions of these methods to solve large-scale DRE. The performance of the proposed methods is compared to existing typical integrators.

Keywords. Differential Riccati equation, Matrix-valued exponential integrators, Low-rank approximation, φ-functions.

1. Introduction

In this paper, we consider the solution of matrix differential Riccati equation (DRE) on the time interval $[t_0, T]$ of the form

$$X'(t) = AX(t) + X(t)A^T + Q - X(t)GX(t), \quad X(t_0) = X_0, \tag{1}$$

where $X(t) \in \mathbb{R}^{N\times N}$ is the unknown matrix-valued function and $A, Q, G \in \mathbb{R}^{N\times N}$ are given coefficient matrices and $X_0 \in \mathbb{R}^{N\times N}$ denotes the initial value with N being the dimension of the equation. The DRE plays a fundamental role in optimal control theory, filter design theory, H_∞-control of linear time-varying systems, model reduction problems, robust control problems and many more (see, e.g., [1,2,3,4,5]). In many practical applications, the coefficient matrix A of Eq. (1) results from the spatial discretization of the differential operator, and the fast and slow modes exist, which means that the associated DRE will be large and stiff.

For stiff DRE, although the most naive approach is to expand the DRE into a vector-valued ordinary differential equation and solve it using implicit time integrators, the method is not suitable for large stiff DRE due to the disadvantages of computationally expensive and considerable storage requirements. Over the past few years some other numerical methods have been proposed for solving the

[1]Corresponding Author: E-mail: lidp@ccsfu.edu.cn.

DRE, see, e.g. [6,7,8,9,10,11,12]). In particular, several low-rank approximations based on matrix versions of classical time integration schemes, such as the BDF method, the Rosenbrock method and the splitting method have been developed, see, e.g., [13,14,15,16]. More recently, a low-rank approximation based on matrix version of the exponential Rosenbrock-type integrator has been introduced in [17].

In this paper we present the matrix-valued versions of two types of exponential integrators: integrating factor method (IF) [18] and generalized integrating factor method (GIF) [19]. Both types of methods have shown good stability and computational efficiency in solving semi-linear problems. We also exploit their low-rank approximations for large DRE. Numerical experiments have shown that the methods proposed are suitable for solving large stiff DRE.

A brief overview of the paper follows. Section 2 focuses on the matrix-based integrating factor (MIF) method. In Section 3, we present the generalized matrix-based integrating factor method. In Section 4, we develop low-rank algorithms for all proposed exponential integrators based on LDL^T-type decompositions, which is the main objective of this paper. In Section 5, we provide some numerical experiments, a comparison of different methods to illustrate the accuracy and effectiveness of the proposed methods. Finally, we summarize some conclusions in Section 6.

2. Matrix-valued Integrating Factor Method

In this section, we describe the IF method for DRE in the matrix-valued form similar to [18] and discuss how to apply this to large-scale problems. For this purpose, we rewrite Eq. (1) as

$$X'(t) = F(X(t)) = \mathcal{L}[X] + \mathcal{N}(X), \tag{2}$$

where $\mathcal{L}$ and $\mathcal{N}$ are Lyapunov and nonlinear operators, respectively:

$$\mathcal{L}[X] = AX + XA^T, \quad \mathcal{N}(X) = Q - XGX. \tag{3}$$

Since the large stiff property of the original Eq. (1), a transformation of variables is considered to ameliorate the stiff part of the equation. For $X(t)$ around $t = t_0$, find a $V : \mathbb{R} \to \mathbb{R}^n$ such that:

$$V(\tau) = e^{-\tau\mathcal{L}}[X(t_0 + \tau)]. \tag{4}$$

Differentiating (4) and then insert it into (2), we have

$$V'(\tau) = g(V(\tau)) = e^{-\tau\mathcal{L}}[\mathcal{N}(e^{\tau\mathcal{L}}[V])], \quad V(\tau_0) = X_0. \tag{5}$$

The aim of this transformation of the differential equation is to remove the explicit dependence of the equation on the operator $\mathcal{L}$, except within the exponential. The exponential function will dampen the behavior of $\mathcal{L}$ removing the stiffness or highly oscillatory nature of the problem. Then, an s-stage explicit Runge-

Kutta method with coefficients b_i, c_i, a_{ij} satisfying the simplifying assumptions $c_1 = 0$ and

$$\sum_{j=1}^{s} b_j = 1, \quad \sum_{j=1}^{s} a_{ij} = c_i, \quad 1 \le i \le s, \tag{6}$$

is used to solve (5), and the process for its numerical solution defines a sequence V_{n+1} approximating $V(\tau_{n+1})$ by:

$$\begin{aligned} k_i &= g(V_n + \tau \sum_{j=1}^{s} a_{ij} k_j), \quad 1 \le i \le s, \\ V_{n+1} &= V_n + \tau \sum_{i=1}^{s} b_i k_i. \end{aligned} \tag{7}$$

By the right of (5) for $g(V(\tau))$, we have

$$\begin{aligned} k_i &= e^{-(\tau_n + c_i\tau)\mathcal{L}}[\mathcal{N}(e^{(\tau_n + c_i\tau)\mathcal{L}}[V_n + \tau \sum_{j=1}^{s} a_{ij} k_j])], \quad 1 \le i \le s, \\ V_{n+1} &= V_n + \tau \sum_{i=1}^{s} b_i k_i. \end{aligned} \tag{8}$$

Then use (4) to transform back to the original variables to obtain the general format for the IF method

$$\begin{aligned} k_i^* &= \mathcal{N}(e^{c_i h\mathcal{L}}[X_n] + h \sum_{j=1}^{s} a_{ij} e^{(c_i - c_j)h\mathcal{L}}[k_j^*]), \quad 1 \le i \le s, \\ X_{n+1} &= e^{h\mathcal{L}}[X_n] + h \sum_{i=1}^{s} b_i e^{(1-c_i)h\mathcal{L}}[k_i^*], \end{aligned} \tag{9}$$

where X_n is the numerical approximation to the exact solution $X(t)$ at time $t = t_n = nh$, and h is the step size. Let us apply the MIF method to Eq. (1), which leads to

$$\begin{aligned} k_i^* &= Q - (e^{c_i h\mathcal{L}}[X_n] + h \sum_{j=1}^{s} a_{ij} e^{(c_i - c_j)h\mathcal{L}}[k_j^*]) \\ &\quad \cdot G \cdot (e^{c_i h\mathcal{L}}[X_n] + h \sum_{j=1}^{s} a_{ij} e^{(c_i - c_j)h\mathcal{L}}[k_j^*]), \quad 1 \le i \le s, \\ X_{n+1} &= e^{h\mathcal{L}}[X_n] + h \sum_{i=1}^{s} b_i e^{(1-c_i)h\mathcal{L}}[k_i^*]. \end{aligned} \tag{10}$$

In order to derive the MIF method, a very careful local error analysis must be performed to determine the coefficients a_{ij} and b_i. Now, the specific form of the MIF method is listed as follows: First, we consider a second-order MIF method and its parameters are given as

$$RK21: \quad \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{11}$$

This yields the following method:

$$\begin{aligned} k_1^* &= \mathcal{N}(X_n), \\ k_2^* &= \mathcal{N}(e^{h\mathcal{L}}[X_n] + he^{h\mathcal{L}}[k_1^*]), \\ X_{n+1} &= e^{h\mathcal{L}}[X_n] + \frac{h}{2}e^{h\mathcal{L}}[k_1^*] + \frac{h}{2}k_2^*. \end{aligned} \tag{12}$$

Second, we consider a third-order MIF method and its parameters are given as

$$RK31: \quad \begin{array}{c|ccc} 0 & & & \\ \frac{2}{3} & \frac{2}{3} & & \\ \frac{2}{3} & \frac{1}{3} & \frac{1}{3} & \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array} \tag{13}$$

This yields the following method:

$$\begin{aligned} k_1^* &= \mathcal{N}(X_n), \\ k_2^* &= \mathcal{N}(e^{\frac{2h}{3}\mathcal{L}}[X_n] + \frac{2h}{3}e^{\frac{2h}{3}\mathcal{L}}[k_1^*]), \\ k_3^* &= \mathcal{N}(e^{\frac{2h}{3}\mathcal{L}}[X_n] + \frac{h}{3}e^{\frac{2h}{3}\mathcal{L}}[k_1^*] + \frac{h}{3}k_2^*), \\ X_{n+1} &= e^{h\mathcal{L}}[X_n] + \frac{h}{4}e^{h\mathcal{L}}[k_1^*] + \frac{3h}{4}e^{\frac{h}{3}\mathcal{L}}[k_3^*]. \end{aligned} \tag{14}$$

Third, we consider a fourth-order MIF method and its parameters are given as

$$RK41: \quad \begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{15}$$

This yields the following method:

$$
\begin{aligned}
k_1^* &= \mathcal{N}(X_n),\\
k_2^* &= \mathcal{N}(e^{\frac{h}{2}\mathcal{L}}[X_n] + \frac{h}{2}e^{\frac{h}{2}\mathcal{L}}[k_1^*]),\\
k_3^* &= \mathcal{N}(e^{\frac{h}{2}\mathcal{L}}[X_n] + \frac{h}{2}k_2^*),\\
k_4^* &= \mathcal{N}(e^{h\mathcal{L}}[X_n] + he^{\frac{h}{2}\mathcal{L}}[k_3^*]),\\
X_{n+1} &= e^{h\mathcal{L}}[X_n] + \frac{h}{6}e^{h\mathcal{L}}[k_1^*] + \frac{h}{3}e^{\frac{h}{2}\mathcal{L}}[k_2^*] + \frac{h}{3}e^{\frac{h}{2}\mathcal{L}}[k_3^*] + \frac{h}{6}k_4^*. \qquad (16)
\end{aligned}
$$

The disadvantage of the MIF method for numerically solving DRE is that there is large error coefficients for $\|\mathcal{L}| \gg 1$. To solve this problem, we will continue to generalize the MIF method.

3. Generalized Matrix-valued Integrating Factor Method (GMIF)

We found that the generalized MIF method in [19] is computationally superior to the standard MIF method, so the matrix-valued version of the generalized MIF method will be further presented in this section. The idea of the method is to approximate the solution of the original equation by solving a simpler ODE exactly. To find an ODE with the same key features as (1)

$$\tilde{X}'(\tau) = \tilde{F}(\tilde{X}(\tau)) = \mathcal{L}[X] + C(\tau), \quad \tilde{X}(\tau_0) = \tilde{X}_0, \qquad (17)$$

using the following approximation for the nonlinear remainder $\mathcal{N}(X)$:

$$C(\tau) = \sum_{j=0}^{s-1} \frac{\tau^j}{j!} c_j, \qquad (18)$$

where

$$c_j = \frac{1}{h^j} \sum_{i=0}^{s-1} \gamma_{ij} \mathcal{N}_{n-i}. \qquad (19)$$

Similar to the construction of the IF method, define the operator $\varphi_{\tau,\tilde{F}} : \mathbb{R}^n \to \mathbb{R}^n$ to represent the solution of (2) by $V(\tau)$ with $V(\tau_0) = X_0$ such that

$$X(t_0 + \tau) = \varphi_{\tau,\tilde{F}}(V(\tau)) = e^{\tau\mathcal{L}}[V] + \sum_{j=1}^{s} \tau^j \varphi_j(\tau\mathcal{L})[c_{j-1}], \qquad (20)$$

and then differentiate this relation with respect to τ leads to the ODE of the form

$$V'(\tau) = (D\varphi_{\tau,\tilde{F}}(V))^{-1}(F(\varphi_{\tau,\tilde{F}}(V)) - \tilde{F}(\varphi_{\tau,\tilde{F}}(V))), \quad V(\tau_0) = X_0. \qquad (21)$$

One can also easily find that by defining the modified vector field $\tilde{F}$ such that $\tilde{F}(X) = \mathcal{L}[X]$ we get the standard IF method. By observing that $D\varphi_{\tau,\tilde{F}}(V) = e^{\tau\mathcal{L}}$, (21) can be simplified

$$V'(\tau) = e^{-\tau\mathcal{L}}[\mathcal{N}(X(t_0+\tau)) - C(\tau)], \quad V(\tau_0) = X_0. \tag{22}$$

Applying the classical fourth-order Runge-Kutta method to the transformed Eq. (22), we obtain

$$\begin{aligned}
k_1 &= e^{-\tau_n\mathcal{L}}[\mathcal{N}[\varphi_{\tau_n,\tilde{F}}(V_n)] - C(\tau_n+\tau)],\\
k_2 &= e^{-(\tau_n+\frac{\tau}{2})\mathcal{L}}[\mathcal{N}(\varphi_{\tau_n+\frac{\tau}{2},\tilde{F}}(V_n+\frac{\tau}{2}k_1)) - C(\tau_n+\frac{\tau}{2})],\\
k_3 &= e^{-(\tau_n+\frac{\tau}{2})\mathcal{L}}[\mathcal{N}(\varphi_{\tau_n+\frac{\tau}{2},\tilde{F}}(V_n+\frac{\tau}{2}k_2)) - C(\tau_n+\frac{\tau}{2})],\\
k_4 &= e^{-(\tau_n+\tau)\mathcal{L}}[\mathcal{N}[\varphi_{\tau_n+\tau,\tilde{F}}(V_n+\tau k_3)] - C(\tau_n+\tau)],\\
V_{n+1} &= V_n + \frac{\tau}{6}k_1 + \frac{\tau}{3}k_2 + \frac{\tau}{3}k_3 + \frac{\tau}{6}k_4.
\end{aligned} \tag{23}$$

Using (21) to calculate the numerical solution in the original variable, that is

$$\begin{aligned}
a &= \varphi_{\frac{h}{2},\tilde{F}}(X_n),\\
b &= \varphi_{h,\tilde{F}}(X_n),\\
c &= a + \frac{h}{2}(\mathcal{N}_a - C(\frac{h}{2})),\\
d &= b + he^{\frac{h}{2}\mathcal{L}}[\mathcal{N}_c - C(\frac{h}{2})],\\
X_{n+1} &= b + \frac{h}{3}e^{\frac{h}{2}\mathcal{L}}[\mathcal{N}_a + \mathcal{N}_c - 2C(\frac{h}{2})] + \frac{h}{6}(\mathcal{N}_d - C(h)),
\end{aligned} \tag{24}$$

where $\mathcal{N}_n = \mathcal{N}(n), n = a, b, c, d$. The corresponding s-1st order interpolation polynomial $C_{s-1}(\tau)$, we call this method ETDs/RK4. In our context, we only consider the following three representative schemes and use them in our numerical experiments.

- **ETD1/RK4** A zeroth-order approximation $C_0(\tau) = \mathcal{N}_n$ to the nonlinear terms $\mathcal{N}(X)$, leads to the ETD1/RK4 method

$$\begin{aligned}
a &= \varphi_{\frac{h}{2},\tilde{F}}(X_n),\\
b &= \varphi_{h,\tilde{F}}(X_n),\\
c &= a + \frac{h}{2}(\mathcal{N}_a - \mathcal{N}_n),\\
d &= b + he^{\frac{h}{2}\mathcal{L}}[\mathcal{N}_c - \mathcal{N}_n],\\
X_{n+1} &= b + \frac{h}{3}e^{\frac{h}{2}\mathcal{L}}[\mathcal{N}_a + \mathcal{N}_c - 2\mathcal{N}_n] + \frac{h}{6}(\mathcal{N}_d - \mathcal{N}_n).
\end{aligned} \tag{25}$$

An interesting property to observe about the ETD1/RK4 method is that when $\mathcal{L} = 0$, the method reduces to the classical fourth-order Runge-Kutta method.

- **ETD2/RK4** A first-order approximation to the nonlinear terms $\mathcal{N}(X)$ of the form

$$C_1(\tau) = \mathcal{N}_n + \tau(\frac{\mathcal{N}_n - \mathcal{N}_{n-1}}{h}), \tag{26}$$

leads to the ETD2/RK4 method

$$\begin{aligned}
a &= \varphi_{\frac{h}{2},\tilde{F}}(X_n),\\
b &= \varphi_{h,\tilde{F}}(X_n),\\
c &= a + \frac{h}{2}(\mathcal{N}_a - \frac{3}{2}\mathcal{N}_n + \frac{1}{2}\mathcal{N}_{n-1}),\\
d &= b + he^{\frac{h}{2}\mathcal{L}}[\mathcal{N}_c - \frac{3}{2}\mathcal{N}_n + \frac{1}{2}\mathcal{N}_{n-1}],\\
X_{n+1} &= b + \frac{h}{3}e^{\frac{h}{2}\mathcal{L}}[\mathcal{N}_a + \mathcal{N}_c - 3\mathcal{N}_n + \mathcal{N}_{n-1}]\\
&\quad + \frac{h}{6}(\mathcal{N}_d - 2\mathcal{N}_n + \mathcal{N}_{n-1}).
\end{aligned} \tag{27}$$

- **ETD3/RK4** A second-order approximation to the nonlinear terms $\mathcal{N}(X)$ of the form

$$\begin{aligned}
C_2(\tau) = \mathcal{N}_n &+ \tau(\frac{\frac{1}{2}\mathcal{N}_{n-2} - 2\mathcal{N}_{n-1} + \frac{3}{2}\mathcal{N}_n}{h})\\
&+ \frac{\tau^2}{2}(\frac{N_{n-2} - 2\mathcal{N}_{n-1} + \mathcal{N}_n}{h^2}),
\end{aligned} \tag{28}$$

leads to the ETD3/RK4 method

$$\begin{aligned}
a &= \varphi_{\frac{h}{2},\tilde{F}}(X_n),\\
b &= \varphi_{h,\tilde{F}}(X_n),\\
c &= a + \frac{h}{2}(\mathcal{N}_a - \frac{15}{8}\mathcal{N}_n + \frac{5}{4}\mathcal{N}_{n-1} - \frac{3}{8}\mathcal{N}_{n-2}),\\
d &= b + he^{\frac{h}{2}\mathcal{L}}[\mathcal{N}_c - \frac{15}{8}\mathcal{N}_n + \frac{5}{4}\mathcal{N}_{n-1} - \frac{3}{8}\mathcal{N}_{n-2}],\\
X_{n+1} &= b + \frac{h}{3}e^{\frac{h}{2}\mathcal{L}}[\mathcal{N}_a + \mathcal{N}_c - \frac{15}{4}\mathcal{N}_n + \frac{5}{2}\mathcal{N}_{n-1} - \frac{3}{4}\mathcal{N}_{n-2}]\\
&\quad + \frac{h}{6}(\mathcal{N}_d - 3\mathcal{N}_n + 3\mathcal{N}_{n-1} - \mathcal{N}_{n-2}).
\end{aligned} \tag{29}$$

This method has two interesting properties. First, it is an exponential general linear method that requires the transfer of two quantities X_n and $\mathcal{N}_{n-k}$ from one

step to another. Second, the coefficient matrices corresponding to the four stages are different from those of the standard integrating factor methods. This property applies to all generalized integrating factor methods, which use the Rung-Kutta methods to approximate the modified initial conditions.

4. Low-rank Methods

4.1. Low-rank Matrix-valued Integrating Factor Method

In order to efficiently implement the MIF method, we consider solving the low-rank formulation of the solution $X(t)$. First, focusing on that Q, G and X_0 in (1) are low-rank positive definite cases, where

$$Q = C^T C, \quad G = BB^T, \quad X_0 = L_0 \Gamma_0 L_0^T, \tag{30}$$

for some $C \in \mathbb{R}^{l\times N}$, $B \in \mathbb{R}^{N\times q}$, $L_0 \in \mathbb{R}^{N\times r}$, and $\Gamma_0 \in \mathbb{R}^{r\times r}$, l, q, $r \ll N$. Let X_n be the desired approximate solution to (1) given as

$$X_n = L_n \Gamma_n L_n^T, \tag{31}$$

where $L_n \in \mathbb{R}^{N\times r}$, $\Gamma_n \in \mathbb{R}^{r\times r}$. Then, k_i^* in (9) can be written as

$$k_i^* = \mathcal{N}(e^{c_i h\mathcal{L}}[L_n \Gamma_n L_n^T] + h\sum_{j=1}^{s} a_{ij} e^{(c_i-c_j)h\mathcal{L}}[L_j \Gamma_j L_j^T]), \quad 1 \le i \le s. \tag{32}$$

In the above formula, k_j^* has the same splitting factor as k_i^*, and its splitting process is described subsequently. The decompositions $L_a \Gamma_n L_a^T$ to $e^{c_i h\mathcal{L}}[X_n]$ and $\hat{L}_a \hat{\Gamma}_a \hat{L}_a^T$ to $h\sum_{j=1}^{s} a_{ij} e^{(c_i-c_j)h\mathcal{L}}[k_j^*]$ are given by the factors

$$\begin{aligned}
&L_a = e^{c_i hA} L_n,\\
&\hat{L}_a = [e^{(c_i-c_0)hA} L_0, e^{(c_i-c_1)hA} L_1, \cdots, e^{(c_i-c_{s-1})hA} L_{s-1}],\\
&\hat{\Gamma}_a = \mathrm{blkdiag}(\gamma_0 \Gamma_0, \gamma_1 \Gamma_1, \cdots, \gamma_{s-1} \Gamma_{s-1}), \quad \gamma_{i-1} = h a_{i,i-1}, \quad 1 \le i \le s.
\end{aligned} \tag{33}$$

Note that the new matrix L_i has more columns than L_j and more than the rank. This means that the size of the blocks of the new matrix will increase dramatically over time during the decomposition process and the computational cost becomes expensive. It is necessary to find more suitable low-rank factors by column compression strategies. Then, inserting (33) into $e^{c_i h\mathcal{L}}[X_n] + h\sum_{j=1}^{s} a_{ij} e^{(c_i-c_j)h\mathcal{L}}[k_j^*]$ yields a decomposition of form $\tilde{L}_a \tilde{\Gamma}_a \tilde{L}_a^T$ with

$$\tilde{L}_a = [L_a, \hat{L}_a], \quad \tilde{\Gamma}_a = \mathrm{blkdiag}(\Gamma_n, \hat{\Gamma}_a). \tag{34}$$

Again, inserting the splitting factors $\tilde{L}_a$ and $\tilde{\Gamma}_a$ into (32) and direct calculation shows that

Algorithm 1 Low-rank approximation of the MIF method for DRE.

Input: $A \in \mathbb{R}^{N\times N}, C \in \mathbb{R}^{l\times N}$ such that $Q = C^TC, B \in \mathbb{R}^{N\times q}$ such that $G = BB^T, L_0 \in \mathbb{R}^{N\times r}, \Gamma_0 \in \mathbb{R}^{r\times r}$ such that $X_0 = L_0\Gamma_0L_0^T, t \in [a, b]$, and the stepsize h.

1: **for** $n = 0$ to $[\frac{b-a}{h}]$ **do**
2: **for** $i = 1$ to s **do**
3: $L_a = e^{c_ihA}L_n$;
4: $\hat{L}_a = [e^{(c_i-c_0)hA}L_0, e^{(c_i-c_1)hA}L_1, \cdots, e^{(c_i-c_{s-1})hA}L_{s-1}]$;
5: $\hat{\Gamma}_a = \begin{pmatrix} \gamma_0\Gamma_0 & & & \\ & \gamma_1\Gamma_1 & & \\ & & \ddots & \\ & & & \gamma_{s-1}\Gamma_{s-1} \end{pmatrix}, \gamma_j = ha_{i,j}, 1 \le i \le s.$
6: Column-compress L_a, $\hat{L}_a$ and $\hat{\Gamma}_a$.
7: Form $\tilde{L}_a = [L_a, \hat{L}_a]$ and $\tilde{\Gamma}_a = \text{blkdiag}(\Gamma_n, \hat{\Gamma}_a)$.
8: Column-compress $\tilde{L}_a$ and $\tilde{\Gamma}_a$.
9: Form $L_i = [C^T, \tilde{L}_a]$ and $\Gamma_i = \text{blkdiag}(I, -(\tilde{\Gamma}_a\tilde{L}_a^TB)(\tilde{\Gamma}_a\tilde{L}_a^TB)^T)$.
10: Column-compress L_i and Γ_i.
11: $L_b = e^{hA}L_n$;
12: $\tilde{L}_b = [e^{(1-c_1)hA}L_1, e^{(1-c_2)hA}L_2, \cdots, e^{(1-c_s)hA}L_s]$;
13: $\tilde{\Gamma}_b = \text{blkdiag}(\beta_1\Gamma_1, \beta_2\Gamma_2, \cdots, \beta_s\Gamma_s), \beta_i = hb_i, 1 \le i \le s.$
14: Column-compress L_b, $\tilde{L}_b$ and $\tilde{\Gamma}_b$.
15: **end for**
16: Form $L_{n+1} = [L_b, \tilde{L}_b]$ and $\Gamma_{n+1} = \text{blkdiag}(\Gamma_n, \tilde{\Gamma}_b)$.
17: Column-compress L_{n+1} and Γ_{n+1}.
18: **end for**

Output: L_n and Γ_n.

$$\begin{aligned} k_i^* &= \mathcal{N}(\tilde{L}_a\tilde{\Gamma}_a\tilde{L}_a^T) \\ &= C^TC - \tilde{L}_a\tilde{\Gamma}_a\tilde{L}_a^TBB^T\tilde{L}_a\tilde{\Gamma}_a\tilde{L}_a^T \\ &= L_i\Gamma_iL_i^T, \end{aligned} \tag{35}$$

with

$$L_i = [C^T, \tilde{L}_a], \quad \Gamma_i = \text{blkdiag}(I, -(\tilde{\Gamma}_a\tilde{L}_a^TB)(\tilde{\Gamma}_a\tilde{L}_a^TB)^T). \tag{36}$$

Under the LDL^T-type splitting with $k_i^* = L_i\Gamma_iL_i^T$, it can be proved that

$$\begin{aligned} X_{n+1} &= e^{h\mathcal{L}}[L_n\Gamma_nL_n^T] + h\sum_{i=1}^{s} b_ie^{(1-c_i)h\mathcal{L}}[L_i\Gamma_iL_i^T] \\ &= L_b\Gamma_nL_b^T + \tilde{L}_b\tilde{\Gamma}_b\tilde{L}_b^T, \end{aligned} \tag{37}$$

with

$$
\begin{aligned}
&L_b = e^{hA}L_n,\\
&\tilde{L}_b = [e^{(1-c_1)hA}L_1, e^{(1-c_2)hA}L_2, \cdots, e^{(1-c_s)hA}L_s],\\
&\tilde{\Gamma}_b = \text{blkdiag}(\beta_1\Gamma_1, \beta_2\Gamma_2, \cdots, \beta_s\Gamma_s), \quad \beta_i = hb_i, \quad 1 \le i \le s.
\end{aligned} \tag{38}
$$

Now, we obtain the LDL^T-type splitting with $X_{n+1} \approx L_{n+1}\Gamma_{n+1}L_{n+1}$, and then use a column compression strategy to obtain low-rank splitting factors:

$$
L_{n+1} = [L_b, \tilde{L}_b], \quad \Gamma_{n+1} = \text{blkdiag}(\Gamma_n, \tilde{\Gamma}_b). \tag{39}
$$

For completeness, we summarize the above solution process in **Algorithm 1**.

4.2. Low-rank Generalized Matrix-valued Integrating Factor Method

For the ETDs/RK4 method, we similarly assume that the previous solution approximation $X_n = L_n\Gamma_n L_n^T$ with $L_n \in \mathbb{R}^{N\times r}$, $\Gamma_n \in \mathbb{R}^{r\times r}$. We assume that the $\varphi_j(\frac{h}{2}\mathcal{L})[c_{j-1}]$ term can be decomposed into the form $L_C\Gamma_C L_C^T$, $1 \le C \le s$, and then the first stage value a can be written as the form of LDL^T-type:

$$
\begin{aligned}
a &= \varphi_{\frac{h}{2},\tilde{F}}(X_n)\\
&= e^{\frac{h}{2}\mathcal{L}}[X_n] + \sum_{j=1}^{s}(\frac{h}{2})^j\varphi_j(\frac{h}{2}\mathcal{L})[c_{j-1}]\\
&= [e^{\frac{h}{2}A}L_n, L_1, L_2, \cdots, L_s]
\begin{pmatrix}
\Gamma_n & & & & \\
& \frac{h}{2}\Gamma_1 & & & \\
& & \frac{h^2}{4}\Gamma_2 & & \\
& & & \ddots & \\
& & & & \frac{h^s}{2^s}\Gamma_s
\end{pmatrix}
[e^{\frac{h}{2}A}L_n, L_1, L_2, \cdots, L_s]^T\\
&= L_a\Gamma_a L_a^T.
\end{aligned} \tag{40}
$$

Then N_a has the LDL^T-type splitting $\tilde{L}_a\tilde{\Gamma}_a\tilde{L}_a^T$ with

$$
\tilde{L}_a = [C^T, L_a], \quad \tilde{\Gamma}_a = \text{blkdiag}(I, -(\Gamma_a L_a B)(\Gamma_a L_a B)^T). \tag{41}
$$

A similar assumption on $\varphi_j(h\mathcal{L})[c_{j-1}]$ yields that it has the decomposition $\tilde{L}_C\tilde{\Gamma}_C\tilde{L}_C^T$ and the approximation $L_b\Gamma_b L_b^T$ to b is given by

$$
L_b = [e^{hA}L_n, \tilde{L}_1, \tilde{L}_2, \cdots, \tilde{L}_s], \quad \Gamma_b = \text{blkdiag}(\Gamma_n, h\tilde{\Gamma}_1, h^2\tilde{\Gamma}_2, \cdots, h^s\tilde{\Gamma}_s). \tag{42}
$$

Obviously, the process of finding the splitting factor of c involves the polynomial $C(\frac{h}{2})$, so the splitting of the stage value c is achieved by first splitting the polynomial c_j in LDL^T-type format, then performing an LDL^T factorization of the interpolated polynomial $C(\frac{h}{2})$, and finally forming a low-order factor of c by collecting the three terms on the right-hand side. The polynomials c_j cor-

responding to the value of $C(\frac{h}{2})$ in (24) has a low-rank splitting form $\tilde{C}_j\tilde{D}_j\tilde{C}_j^T$ with

$$\begin{aligned}\tilde{C}_j &= [\tilde{L}_0, \tilde{L}_1, \cdots, \tilde{L}_{n-s+1}],\\ \tilde{D}_j &= \text{blkdiag}(\frac{\gamma_{0j}}{h^j}\tilde{\Gamma}_0, \frac{\gamma_{1j}}{h^j}\tilde{\Gamma}_1, \cdots, \frac{\gamma_{(s-1)j}}{h^j}\tilde{\Gamma}_{n-s+1}), \quad 0 \le j \le s-1. \end{aligned} \tag{43}$$

Substitute c_j by the above splitting, we obtain the splitting factors $\bar{C}_j$, $\bar{D}_j$ of $C(\frac{h}{2})$

$$\begin{aligned}\bar{C}_j &= [\tilde{C}_0, \tilde{C}_1, \tilde{C}_2, \cdots, \tilde{C}_{s-1}],\\ \bar{D}_j &= \text{blkdiag}(\tilde{D}_0, \frac{\tau}{2}\tilde{D}_1, \frac{(\frac{\tau}{2})^2}{2!}\tilde{D}_2, \cdots, \frac{(\frac{\tau}{2})^{s-1}}{(s-1)!}\tilde{D}_{s-1}). \end{aligned} \tag{44}$$

And then c can be written as the low-rank form $L_c\Gamma_c L_c^T$ with

$$L_c = [L_a, \tilde{L}_a, \bar{C}_j], \quad \Gamma_c = \text{blkdiag}(\Gamma_a, \frac{h}{2}\tilde{\Gamma}_a, -\frac{h}{2}\bar{D}_j). \tag{45}$$

A proeduce similar to (41) can be applied to N_c yields the splitting factors

$$\tilde{L}_c = [C^T, L_c], \quad \tilde{\Gamma}_c = \text{blkdiag}(I, -(\Gamma_c L_c B)(\Gamma_c L_c B)^T). \tag{46}$$

Using the spliting $C(\frac{h}{2}) = \bar{C}_j\bar{D}_j\bar{C}_j^T$, it follows that d has the following splitting factors

$$L_d = [L_b, e^{\frac{h}{2}A}\tilde{L}_c, e^{\frac{h}{2}A}\bar{C}_j], \quad \Gamma_d = \text{blkdiag}(\Gamma_b, h\tilde{\Gamma}_c, -h\bar{D}_j). \tag{47}$$

In the same way, the splitting factors of N_d can be written as

$$\tilde{L}_d = [C^T, L_d], \quad \tilde{\Gamma}_d = \text{blkdiag}(I, -(\Gamma_d L_d B)(\Gamma_d L_d B)^T). \tag{48}$$

Finally, we obtain the LDL^T-type splitting $C(h) = \hat{C}_j\hat{D}_j\hat{C}_j^T$ with

$$\begin{aligned}\hat{C}_j &= [\tilde{C}_0, \tilde{C}_1, \tilde{C}_2, \cdots, \tilde{C}_{s-1}],\\ \hat{D}_j &= \text{blkdiag}(\tilde{D}_0, \tau\tilde{D}_1, \frac{\tau^2}{2!}\tilde{D}_2 \cdots, \frac{\tau^{s-1}}{(s-1)!}\tilde{D}_{s-1}). \end{aligned} \tag{49}$$

Now, the low-ranking approximation $L_{n+1}\Gamma_{n+1}L_{n+1}^T$ to X_{n+1} is given by forming

$$\begin{aligned}L_{n+1} &= [L_b, e^{\frac{h}{2}A}\tilde{L}_a, e^{\frac{h}{2}A}\tilde{L}_c, e^{\frac{h}{2}A}\bar{C}_j, \tilde{L}_d, \hat{C}_j],\\ \Gamma_{n+1} &= \text{blkdiag}(\Gamma_b, \frac{h}{3}\tilde{\Gamma}_a, \frac{h}{3}\tilde{\Gamma}_c, -\frac{2h}{3}\bar{D}_j, \frac{h}{6}\tilde{\Gamma}_d, -\frac{h}{6}\hat{D}_j). \end{aligned} \tag{50}$$

The details on procedure are summarized in **Algorithm 2**.

Algorithm 2 Low-rank approximation of the ETDs/RK4 method for DREs.

Input: $A \in \mathbb{R}^{N\times N}, C \in \mathbb{R}^{l\times N}$ such that $Q = C^TC, B \in \mathbb{R}^{N\times q}$ such that $G = BB^T, L_0 \in \mathbb{R}^{N\times r}, \Gamma_0 \in \mathbb{R}^{r\times r}$ such that $X_0 = L_0\Gamma_0L_0^T$, $t \in [a, b]$ and the stepsize h.

for $n = 0$ to $\left[\frac{b-a}{h}\right]$ **do**
  for $j = 1$ to s **do**
    Compute the low-rank approximation $L_C\Gamma_CL_C^T$ to $\varphi_j(\frac{h}{2}\mathcal{L})C_{j-1}$ based on a numerical quadrature formula.
    Compute the low-rank approximation $\tilde{L}_C\tilde{\Gamma}_C\tilde{L}_C^T$ to $\varphi_j(h\mathcal{L})C_{j-1}$ based on a numerical quadrature formula.
  end for
  Compute the LDL^T splitting $L_a\Gamma_aL_a^T$ of a with
  $L_a = [e^{\frac{h}{2}A}L_n, L_1, L_2, \cdots, L_s]$, $\Gamma_a = \text{blkdiag}(\Gamma_n, \frac{h}{2}\Gamma_1, \frac{h^2}{4}\Gamma_2, \cdots, \frac{h^s}{2^s}\Gamma_s)$.
  Column-compress L_a and Γ_a.
  Form $\tilde{L}_a = [C^T, L_a]$ and $\tilde{\Gamma}_a = \text{blkdiag}(I, -(\Gamma_aL_aB)(\Gamma_aL_aB)^T)$.
  Column-compress $\tilde{L}_a$ and $\tilde{\Gamma}_a$.
  Compute the LDL^T splitting $L_b\Gamma_bL_b^T$ of b with
  $L_b = [e^{hA}L_n, \tilde{L}_1, \tilde{L}_2, \cdots, \tilde{L}_s]$ and $\Gamma_b = \text{blkdiag}(\Gamma_n, h\tilde{\Gamma}_1, h^2\tilde{\Gamma}_2, \cdots, h^s\tilde{\Gamma}_s)$.
  Column-compress L_b and Γ_b.
  for $j = 0$ to s-1 **do**
    Compute the LDL^T splitting $\tilde{C}_j\tilde{D}_j\tilde{C}_j^T$ of C_j with
    $\tilde{C}_j = [\tilde{L}_0, \tilde{L}_1, \cdots, \tilde{L}_{n-s+1}]$,
    $\tilde{D}_j = \text{blkdiag}(\frac{\gamma_{0j}}{h^j}\tilde{\Gamma}_0, \frac{\gamma_{1j}}{h^j}\tilde{\Gamma}_1, \cdots, \frac{\gamma_{(s-1)j}}{h^j}\tilde{\Gamma}_{n-s+1})$.
    Column-compress $\tilde{C}_j$ and $\tilde{D}_j$.
  end for
  Form $\bar{C}_j = [\tilde{C}_0, \tilde{C}_1, \tilde{C}_2, ..., \tilde{C}_{s-1}]$ and
  $\bar{D}_j = \text{blkdiag}(\tilde{D}_0, \frac{\tau}{2}\tilde{D}_1, \frac{(\frac{\tau}{2})^2}{2!}\tilde{D}_2, \cdots, \frac{(\frac{\tau}{2})^{s-1}}{(s-1)!}\tilde{D}_{s-1})$.
  Column-compress $\bar{C}_j$ and $\bar{D}_j$.
  Compute the LDL^T splitting $L_c\Gamma_cL_c^T$ of c with
  $L_c = [L_a, \tilde{L}_a, \bar{C}_j]$ and $\Gamma_c = \text{blkdiag}(\Gamma_a, \frac{h}{2}\tilde{\Gamma}_a, -\frac{h}{2}\bar{D}_j)$.
  Column-compress L_c and Γ_c.
  Form $\tilde{L}_c = [C^T, L_c]$ and $\tilde{\Gamma}_c = \text{blkdiag}(I, -(\Gamma_cL_cB)(\Gamma_cL_cB)^T)$.
  Column-compress $\tilde{L}_c$ and $\tilde{\Gamma}_c$.
  Compute the LDL^T splitting $L_d\ \Gamma_dL_d^T$ of d with
  $L_d = [L_b, e^{\frac{h}{2}A}\tilde{L}_c, e^{\frac{h}{2}A}\bar{C}_j]$ and $\Gamma_d = \text{blkdiag}(\Gamma_b, h\tilde{\Gamma}_c, -h\bar{D}_j)$.
  Column-compress L_d and Γ_d.
  Form $\tilde{L}_d = [C^T, L_d]$ and $\tilde{\Gamma}_d = \text{blkdiag}(I, -(\Gamma_dL_dB)(\Gamma_dL_dB)^T)$.
  Column-compress $\tilde{L}_b$ and $\tilde{\Gamma}_d$.
  Form $\hat{C}_j = [\tilde{C}_0, \tilde{C}_1, \tilde{C}_2, ..., \tilde{C}_{s-1}]$ and
  $\hat{D}_j = \text{blkdiag}(\tilde{D}_0, \tau\tilde{D}_1, \frac{\tau^2}{2!}\tilde{D}_2 \cdots, \frac{\tau^{s-1}}{(s-1)!}\tilde{D}_{s-1})$.
  Column-compress $\hat{C}_j$ and $\hat{D}_j$.
  Form $L_{n+1} = [L_b, e^{\frac{h}{2}A}\tilde{L}_a, e^{\frac{h}{2}A}\tilde{L}_c, e^{\frac{h}{2}A}\bar{C}_j, \tilde{L}_d, \hat{C}_j]$ and
  $\Gamma_{n+1} = \text{blkdiag}(\Gamma_b, \frac{h}{3}\tilde{\Gamma}_a, \frac{h}{3}\tilde{\Gamma}_c, -\frac{2h}{3}\bar{D}_j, \frac{h}{6}\tilde{\Gamma}_d, -\frac{h}{6}\hat{D}_j)$.
  Column-compress L_{n+1} and Γ_{n+1}.
end for

Output: L_n and Γ_n.

5. Numerical experiments

In the following subsections, we test the performance of the method presented through a number of numerical experiments. All tests were performed under Windows 10 and MATLAB R2021b, running on a computer equipped with an Intel Core i5 processor (1.6 GHz) and 8 GB of RAM. The relative error is measured in the F-norm, i.e.,

$$Error = \frac{||Y - \hat{Y}||_F}{||Y||_F}, \tag{51}$$

where $\hat{Y}$ is the computed solution and Y is the reference solution, respectively.

In the experiment, we use the MATLAB functions recurLrlyap [20] and expmv in [21] to evaluate φ-functions of the form $\varphi(\mathcal{L})[Q]$ and matrix functions of the form $e^A L$, respectively. The MATLAB function recurLrlyap employs the scaling and recursion method, while the MATLAB function expmv employs the scaling and squaring method based on the Taylor series. For the low-rank implementation, the column compression strategy terminates at a tolerance of $N \cdot \epsilon$. Here, N is the system dimension and ϵ denotes the machine accuracy.

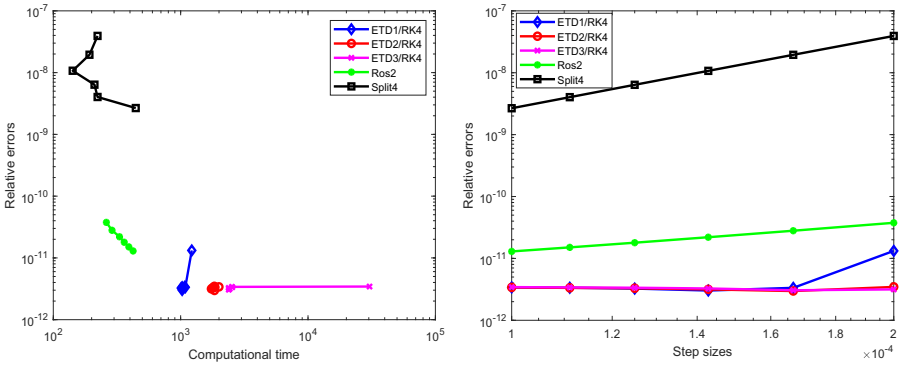
Experiment 1. The matrix A was obtained from the 5-point discretization of the following advection-diffusion operator

$$L[u] = \triangle u - f_1(x,y)\frac{\partial u}{\partial x} - f_2(x,y)\frac{\partial u}{\partial y}, \tag{52}$$

on the unit square $[0,1] \times [0,1]$ with homogeneous Dirichlet boundary conditions. The system matrices A, B and C can be generated directly by the MATLAB functions fdm_2d_matrix and fdm_2d_vector in the LYAPACK toolbox [22], respectively. In this test, we set two different function values $f_1(x,y) = 10x$ and $f_2(x,y) = 100y$. The domains of B and C are restricted to the input region of $x, y \in (0.1, 0.3)$ and the output region of $x, y \in (0.7, 0.9)$. The low-rank factor L_0 of the initial value X_0 was randomly generated by the MATLAB function rand. Further, there are $n_0 = 40$ equidistant grid points in each spatial dimension, and that the dimension of the matrix A is n_0^2. The reference solution of equation (1) at time t can be expressed exactly in terms of the fourth-order symmetric splitting scheme for the scheme with time step size $h = 10^{-4}$.

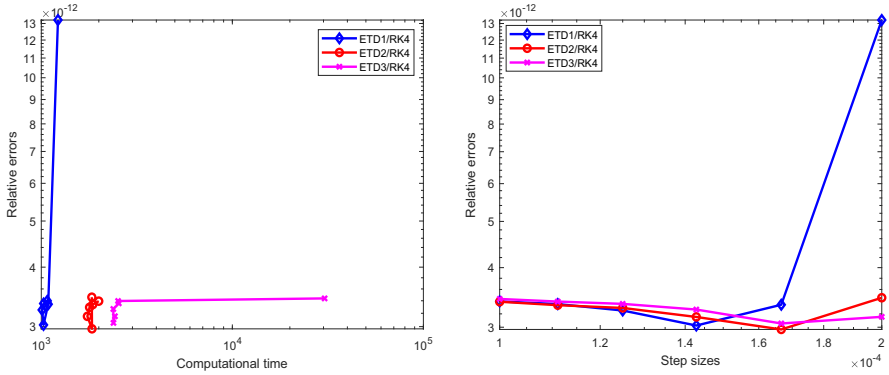
To test the performance of the GMIF method, we compared it with two classes of low-rank integrators developed in [23], including the second-order Rosenbrock method and fourth-order symmetric additive splitting method. We abbreviate them as Ros2 and Split4, respectively.

Figure 1 shows the accuracy and efficiency plots for ETD1/RK4, ETD2/RK4, ETD3/RK4, Ros2 and Split4 for each system on the integration interval $[0, 0.1]$ with time step sizes $h \in \{\frac{1}{500}, \frac{1}{600}, \frac{1}{700}, \frac{1}{800}, \frac{1}{900}, \frac{1}{1000}\}$. We can see that ETD1/RK4, ETD2/RK4 and ETD3/RK4 are more accurate than Ros2 and Split4 for the same time step, on the other hand they are more expensive. To get a better view as well as to compare the three methods, they are again plotted in Figure 2 with the same integration interval and number of steps.



a.Efficiency plot b.Order plot

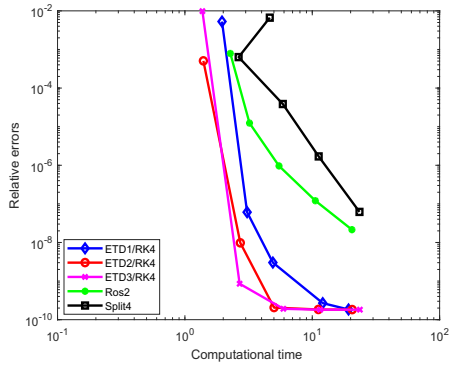
Figure 1. Experimental results of DRE for Experiment 1. Left: The relative errors versus computational time at t = 0.1. Right: The relative errors versus time step sizes $h \in \{\frac{1}{500}, \frac{1}{600}, \frac{1}{700}, \frac{1}{800}, \frac{1}{900}, \frac{1}{1000}\}$ for the same problem.



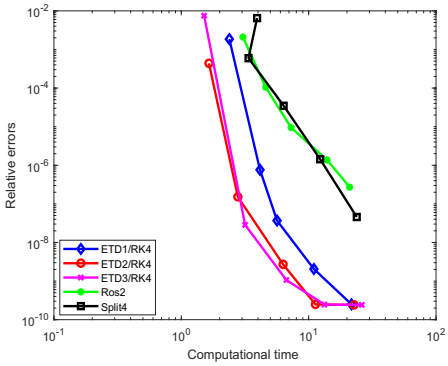
a.Efficiency plot b.Order plot

Figure 2. Same experimental setup as in Fig 1, but now for comparison purposes only the three methods ETD1/RK4, ETD2/RK4, ETD3/RK4 are shown.

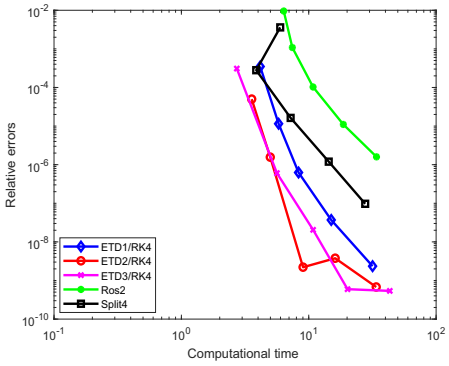
Experiment 2. As a second numerical experiment, We consider an artificial symmetric model problem acting on the unit square $\omega = (0, 1)^2$ with $N = n_0^2 = 1600$ degrees of freedom. The matrix $A \in R^{n_0^2 \times n_0^2}$ can be viewed as a finite difference discretization of the two-dimensional Laplace operator in a unitary square matrix with uniform boundary conditions. The matrices $B \in R^{n_0^2 \times 1}$, $C \in R^{1 \times n_0^2}$ and $L_0 \in R^{n_0^2 \times 1}$ are randomly generated using the MATLAB function randn with normally distributed terms. In order to determine the exact reference solution, we use a fourth-order symmetric splitting approach with time step size $h = 10^{-3}$. To test the relative errors and computation times of the GMIF method, we compared it with Ros2 and split4 developed in M.E.S.S. Toolbox [23]. We have carried out



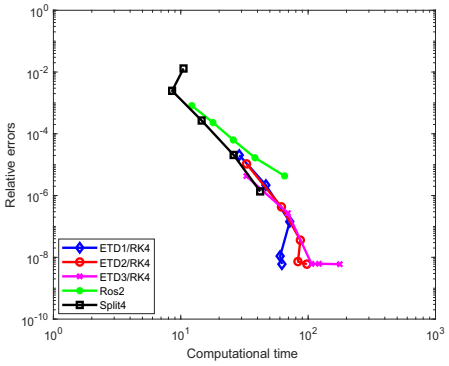
a.Efficiency plot for $\alpha = 2 \cdot 10^{-2}$



b.Efficiency plot for $\alpha = 2 \cdot 10^{-1}$



a.Efficiency plot for $\alpha = 2$

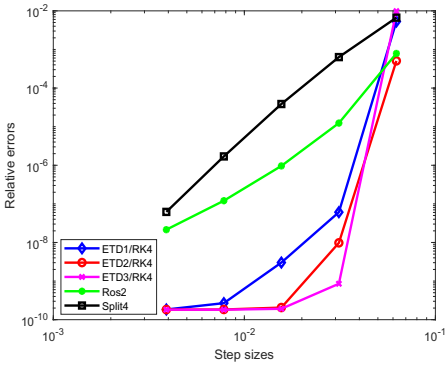


b.Efficiency plot for $\alpha = 2 \cdot 10$

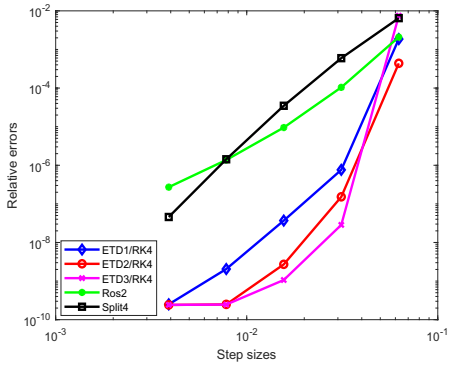
Figure 3. The relative errors of ETD1/RK4, ETD2/RK4, ETD3/RK4, Ros2 and Split4 versus the computation time when integrating each of the equations for Example 2 for t = 1.

numerical experiments with four different values of the coefficient $\alpha = 2 \cdot 10^{-2}$, $2 \cdot 10^{-1}$, 2, $2 \cdot 10$. The stiffness of the problem increases with increasing values of α.

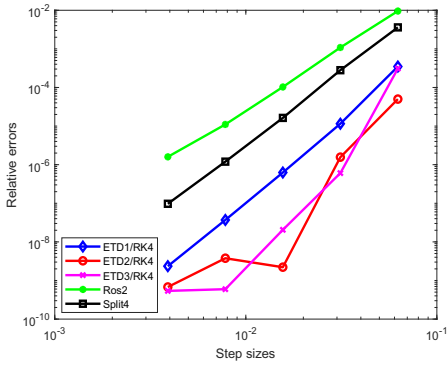
Figure 3 and Figure 4 examine the efficiency plots and accuracy plots of the methods with different coefficients for time steps of $h = 2^{-k}, k = 4, 5, \cdots, 8$ under the integration interval $[0, 1]$, respectively. In Figure 3, we can see that ETD1/RK4, ETD2/RK4 and ETD3/RK4 are more effective than Ros2 and Split4 for the same time step corresponding to a smaller stiffness. As the problem becomes more rigid, our relative errors are always significantly smaller than those of Ros2 and spilt4 , although our method performs slightly worse in terms of validity. Observe in Figure 4 that ETD1/RK4, ETD2/RK4 and ETD3/RK4 have higher accuracy when using the same step size. In particular, they remain more accurate than Ros2 and Split4 in the case of strongly rigid problems.



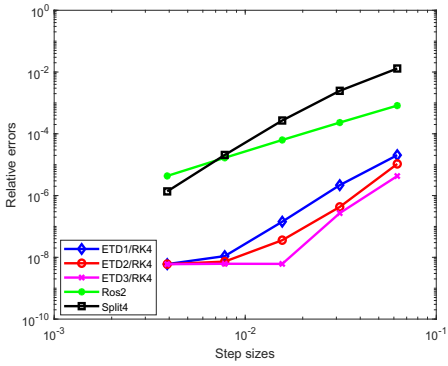
a.order plot for $\alpha = 2 \cdot 10^{-2}$



b.order plot for $\alpha = 2 \cdot 10^{-1}$



a.order plot for $\alpha = 2$



b.order plot for $\alpha = 2 \cdot 10$

Figure 4. The relative errors of ETD1/RK4, ETD2/RK4, ETD3/RK4, Ros2 and Split4 versus the variable number of time step sizes $h = 2^{-k}, k = 4, 5, \cdots, 8$ when integrating each of equations for Experiment 2 on [0, 1].

6. Conclusion

In this paper, we show how two types of exponential integrators can be applied to matrix-valued DRE, one of which is the integrating factor method and the other is the generalized integrating factor method. Based on this, we further construct two efficient algorithms based on low-rank decomposition. The performance is tested in numerical experiments with two different examples, and the results show that the proposed methods are more effective for large-scale rigid problems. In addition, the core of the computation of the generalized integral factor method is the computation of the function. Therefore, we hope that more accurate function computation methods can be proposed in the future to improve the performance.

Acknowledgements

This work was supported in part by the National Nature Science Foundation of China (Grant No. 12371455) and Jilin Scientific and Technological Development Program (Grant No. 20200201276JC) and the Natural Science Foundation of Changchun Normal University (Grant No. 2021001).

References

[1] Abou-Kandil H, Freiling G, Ionescu V, Jank G. Matrix Riccati Equations in Control and Systems Theory. Switzerland: Birkhäuser, Basel; 2003.
[2] Ichikawa A, Katayama H. Remarks on the time-varying H_∞ Riccati equations. Systems Control Lett. 1999;37:335-345.
[3] Jacobs OLR. Introduction to Control Theory. UK: Oxford Science Publications; 1993.
[4] Lancaster P, Rodman L. Algebraic Riccati equations. New York: Oxford Science Publications/The Clarendon Press Oxford University Press; 1995.
[5] Mehrmann V. The Autonomous Linear Quadratic Control Problem: Theory and Numerical Solution. 1991.
[6] Choi CH, Laub AJ. Efficient matrix-valued algorithms for solving stiff Riccati differential equations. IEEE Trans. Automat. Control. 1990;35:770-776.
[7] Lang N, Mena H, Saak J. On the benefits of the LDL^T factorization for large-scale differential matrix equation solvers. Linear Algebra and its Applications. 2015;480:44-71.
[8] Davison EJ, Maki M. The numerical solution of the matrix Riccati differential equation. IEEE Trans. Automat. Control. 1973;18:71-73.
[9] Dieci L. Numerical integration of the differential Riccati equation and some related issues. SIAM J. Numer. Anal. 1992;29:781-815.
[10] Kenne C, Leipnik R. Numerical integration of the differential matrix Riccati equation. IEEE Trans. Automat. Control. 1985;30:962-970.
[11] Kirsten G, Simoncini V. Order reduction methods for solving large-scale differential matrix Riccati equations. SIAM J. Sci. Comput. 2020;42:2182-2205.
[12] Laub AJ. A schur method for solving algebraic Riccati equations. IEEE Trans. Autom. Control. 1979;24:913-921.
[13] Benner P, Mena H. BDF methods for large-scale differential Riccati equations. proc of mathematical theory of network & systems mtns. 2009.
[14] Benner P, Mena H. Rosenbrock Methods for Solving Riccati Differential Equations. IEEE Trans, Automat. Control. 2013;58:2950-2956.
[15] Stillfjord T. Adaptive high-order splitting schemes for large-scale differential Riccati equations. Numer. Algor. 2018;78:1129-1151.
[16] Stillfjord T. Low-rank second-order splitting of large-scale differential Riccati equations. IEEE Trans. Automat. Control. 2015;60:2791-2796.
[17] Li DP, Zhang XY, Liu RY. Exponential integrators for large-scale stiff Riccati differential equation. J. Comput. Appl. Math. 2021;389:113360.
[18] Lawson D. Generalized RungeKutta processes for stable systems with large lipschitz constants. SIAM J. Numer. Anal. 1967;4:372-380.
[19] Krogstad S. Generalized integrating factor methods for stiff PDEs. Journal of Computational Physics. 2005;203:72-88.
[20] Li DP, Zhang XY. A low-rank algorithm for solving Lyapunov operator φ-functions within the matrix-valued exponential integrators. 2022. arXiv: 2212.02408.
[21] Al-Mohy A, Higham N. A new scaling and modified squaring algorithm for matrix functions. SIAM J. Matrix Anal. Appl. 2009;31:970-989.
[22] Penzl T. LYAPACK A MATLAB Toolbox for Large Lyapunov and Riccati Equations, Model Reduction Problems, and Linear-Quadratic Optimal Control Problems Users. 2000.
[23] Saak J, Koehler M, Benner P. M-M.E.S.S.-1.0.1-The Matrix Equations Sparse Solvers Library. 2016.