NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 0000; **00**:1–16
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla

Numerical solution of the finite horizon stochastic linear quadratic control problem

Tobias Damm[1], Hermann Mena[2,3]* and Tony Stillfjord[4]

[1] *Technische Universität Kaiserslautern, Department of Mathematics, 67653 Kaiserslautern, Germany*
[2]*Institut für Mathematik, Universität Innsbruck, Technikerstraße 13a, 6020 Innsbruck, Austria*
[3]*School of Mathematical Sciences and Information Technology, Yachay Tech, Hacienda San José, San Miguel de Urcuquí, Ecuador*
[4]*Mathematical Sciences, Chalmers University of Technology and the University of Gothenburg, SE-412 96 Göteborg, Sweden*

SUMMARY

The treatment of the stochastic linear quadratic optimal control problem with finite time horizon requires the solution of stochastic differential Riccati equations (SRE). We propose efficient numerical methods which exploit the particular structure and can be applied for large-scale systems. They are based on numerical methods for ordinary differential equations such as Rosenbrock methods, Backward Differentiation Formulas (BDF) and splitting methods. The performance of our approach is tested in numerical experiments. Copyright © 0000 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Stochastic LQR problem; Stochastic Riccati equations; Rosenbrock methods; BDF methods; Splitting methods

1. INTRODUCTION

The stochastic linear quadratic regulator (SLQR) problem in finite dimensions has been first studied by Kushner (1962) [34] and Wonham (1968) [49, 50]. Control problems with stochastic coefficients and the corresponding backward stochastic Riccati equations (BSREs) have been treated in the finite-horizon and finite-dimensional case by many authors [12, 13, 30, 31, 32, 33]. In [51], one can find a complete treatment of the SLQR problem in finite dimensions along with a feedback characterization of the optimal control via a matrix Riccati equation. Note that although this equation is called *stochastic Riccati equation* (SRE), it is a deterministic differential matrix equation. Thus the feedback relation between the optimal control and the optimal state is deterministic, even though both are random processes.

The finite horizon SLQR control problem for a one-dimensional Brownian motion consists of the controlled state equation

$$\begin{aligned} dx(t) &= [A(t)x(t) + B(t)u(t) + b(t)]\,dt + [C(t)x(t) + D(t)u(t) + d(t)]\,dW(t), \\ x(0) &= y \in \mathbb{R}^n, \quad t \in [0, T], \end{aligned} \tag{1}$$

*Correspondence to: Hermann Mena, Institut für Mathematik, Universität Innsbruck, Technikerstraße 13a, 6020 Innsbruck, Austria. E-mail: hermann.mena@uibk.ac.at and the performance index

$$J(u) = \frac{1}{2}\mathbb{E}\Big(\int_0^T (\langle Q(t)x(t), x(t)\rangle + 2\langle Sx(t), u(t)\rangle + \langle R(t)u(t), u(t)\rangle)dt \\ + \langle Gx(T), x(T)\rangle \Big). \tag{2}$$

The objective is to find the minimum of the functional J over all possible controls u subject to the condition that x satisfies the state equation (1), where $T > 0$, A, B, C, D, d are deterministic matrix-valued functions of suitable dimensions $A, C \in L^\infty((0,T), \mathbb{R}^{n\times n})$, $B, D \in L^\infty((0,T), \mathbb{R}^{n\times k})$, $b, d \in L^2((0,T), \mathbb{R}^n)$ and $W(t)$ is a Brownian motion defined on $(\Omega, \mathcal{F}, \mathbb{P})$ over $t \in [0,T]$. We assume $Q \in L^\infty((0,T), \mathcal{S}^n)$, $S \in L^\infty((0,T), \mathbb{R}^{k\times n})$ and $R \in L^\infty((0,T), \mathcal{S}^k)$, $G \in \mathcal{S}^n$ and that all the coefficients depend on time t. Some of these assumptions can be weakened, [51].

If the SLQR problem is well-posed it can be reduced to that of solving a SRE and a backward stochastic differential equation (BSDE). The SRE has the form

$$\begin{cases} \dot{P} = -(A^TP + PA + Q + C^TPC - \\ \qquad - \left(B^TP + S + D^TPC\right)^T (R + D^TPD)^{-1} \left(B^TP + S + D^TPC\right)), \\ \qquad\qquad \text{for a.a. } t \in [0,T], \\ P(T) = G, \\ R(t) + D(t)^TP(t)D(t) > 0 \ \text{ for a.a. } t \in [0,T], \end{cases} \tag{3}$$

and the BSDE

$$\begin{cases} \dot{\varphi} + \left(A - B(R + D^TPD)^{-1}(B^TP + S + D^TPC)\right)^T \varphi + \\ \qquad + \left(C - D(R + D^TPD)^{-1}(B^TP + S + D^TPC)\right)^T Pd + Pb = 0, \\ \qquad\qquad \text{for a.a. } t \in [0,T] \\ \varphi(T) = 0. \end{cases} \tag{4}$$

The above derivation requires that $R(t) + D(t)^TP(t)D(t)$ is invertible. The existence and uniqueness of the solutions to these equations are available only for certain special cases. In many applications, there is no multiplicative noise associated with the control input u, i.e. $D = 0$, and there is no weighted cross term in the cost functional, i.e. $S = 0$. In this case, the SRE (3) has the simplified form

$$\begin{cases} \dot{P} = -(A^TP + PA + Q + C^TPC - PBR^{-1}B^TP), \quad \text{for a.a. } t \in [0,T], \\ P(T) = G, \\ R(t) > 0 \ \text{ for a.a. } t \in [0,T]. \end{cases} \tag{5}$$

Note that we can solve (5) forward in time, i.e. $P(0) = G$, and in this way state the equation in the usual format of an initial value problem.

In this paper we focus on solving (5) involving multidimensional Brownian motion, i.e. we study

$$\begin{cases} \dot{P} = A^TP + PA + Q + \sum_{j=1}^{\nu} C_j^TPC_j - PBR^{-1}B^TP \quad \text{for a.a. } t \in [0,T], \\ P(0) = G, \\ R(t) > 0 \ \text{ for a.a. } t \in [0,T]. \end{cases} \tag{6}$$

Our focus is on large-scale problems arising in the numerical treatment of SQLR problems governed by stochastic partial differential equations. The infinite dimensional SLQR problem was solved by Ichikawa in [29] using a dynamic programming approach. Da Prato [17] and Flandoli [19] later considered the stochastic LQR for systems driven by analytic semigroups with Dirichlet or Neumann boundary controls, but with disturbance in the state only. The infinite dimensional LQR with random coefficients has been investigated in [21, 22] along with the associated backward stochastic Riccati equation. In [39], a novel approach for solving the stochastic LQR based on the concept of chaos expansion from white noise analysis is proposed. For a class of control systems known as singular estimate control systems the stochastic analog of the linear quadratic problem has been first treated by Hafizoglu [24]. This class captures certain systems of coupled parabolic/hyperbolic PDEs, with boundary or point control actions [36, 37, 38]. Recently, a theoretical framework for the stochastic LQR has been laid out for singular estimates control systems in the presence of noise in the control and in the case of finite time penalization in the performance index [25]. Considering the general setting described in [25], an approximation scheme for solving the control problem and the associated Riccati equation has been proposed in [40].

The numerical solution of the SLQR relies on solving efficiently the associated Riccati equation. For the deterministic LQR problem, most of the methods for solving differential Riccati equations, e.g. [2, 8, 35], are based on a low-rank approximation of the solution, and their performance relies on the rapid decay of the singular values. This phenomenon is observed in singular estimate control systems in applications, and has been studied in detail by e.g. [1, 20, 42, 41, 47]. Here we generalize these methods for SREs assuming that the decay property also holds. The assumption is verified empirically in our numerical experiments.

The paper is organized as follows. In Section 2 we solve SREs by ODE methods such as the Rosenbrock and BDF methods in matrix setting. Then, in Section 3, we do the same for first- and second-order exponential splitting schemes. In Section 4, the proposed methods are tested on a few numerical examples and their efficiency is compared. Finally, Section 5 presents our conclusions and outlook.

2. ODE METHODS IN MATRIX SETTING

In [8, 9] it is shown that standard ODE methods can efficiently be applied to the deterministic Riccati differential equation (DRE) in the matrix setting. Here, we discuss the application of these methods to the SRE. In particular, we focus on the Rosenbrock and BDF methods.

Let us first consider the Rosenbrock methods, which have proved to be efficient methods for autonomous DREs. For simplicity we consider the one-dimensional Brownian motion case, i.e. (5), but the same procedure can be applied to the multidimensional Brownian motion case. We discretize the time interval $[0, T]$ by the equidistant grid $\{t_k\}_{k=1}^{N_T}$ with step size $h = t_{k+1} - t_k$, and seek the approximation $P_k \approx P(t_k)$. In order to abbreviate the notation, we further introduce

$$S := BR^{-1}B^T$$

and represent the right-hand side of the equation by the operator $\mathcal{F} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$, given by

$$\mathcal{F}P = A^TP + PA + Q + C^TPC - PSP.$$

We note that the Frechét derivative of $\mathcal{F}$ at P_k is given by the generalized Lyapunov operator $\mathcal{F}^{'}(P_k) : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ with

$$\mathcal{F}^{'}(P_k)U = (A - SP_k)^TU + U(A - SP_k) + C^TUC.$$

The application of the linear implicit Euler method, as a matrix-valued algorithm, to the SRE (5) now yields

$$\begin{aligned} P_{k+1} &= P_k + hK_1, \\ K_1 - h(\mathcal{F}^{'}(P_k))(K_1) &= \mathcal{F}P_k. \end{aligned} \tag{7}$$

Note that K_1 represents a $n \times n$ matrix. Expanding $\mathcal{F}^{'}(P_k)$ in (7), we obtain

$$K_1 - h(A - SP_k)^TK_1 - hK_1(A - SP_k) - hC^TK_1C = \mathcal{F}P_k,$$

and re-arranging terms gives

$$(h(A - SP_k) - \frac{1}{2}I)^T K_1 + K_1(h(A - SP_k) - \frac{1}{2}I) + hC^T K_1 C = -\mathcal{F}P_k.$$

Denoting $\bar{A}_k = h(A - SP_k) - \frac{1}{2}I$, we can write the method as:

$$P_{k+1} = P_k + hK_1, \tag{8}$$
$$\bar{A}_k^T K_1 + K_1 \bar{A}_k + hC^T K_1 C = -\mathcal{F}P_k. \tag{9}$$

Hence, one generalized Lyapunov equation (9) has to be solved in every step. We consider such equations in Section 2.1. By writing $\mathcal{F}P_k$ as

$$\begin{aligned}\big(A - SP_k - \frac{1}{2h}I\big)^T P_k + P_k\big(A - SP_k - \frac{1}{2h}I\big)& \\ + Q + P_k S P_k + \frac{1}{h}P_k + C^T P_k C&,\end{aligned}$$

it is easily seen that P_{k+1} can be computed directly from

$$\tilde{A}_k^T P_{k+1} + P_{k+1}\tilde{A}_k + C^T P_{k+1} C = -Q - P_k S P_k - \frac{1}{h}P_k, \tag{10}$$

where $\tilde{A}_k = A - SP_k - \frac{1}{2h}I$. This avoids the additional step in (8) but is still a generalized Lyapunov equation.

The application of the Rosenbrock method of order two proposed in [8, 9], as a matrix-valued algorithm, to the SRE (5) yields

$$P_{k+1} = P_k + \frac{3}{2}hK_1 + \frac{1}{2}hK_2,$$
$$K_1 - \gamma h(\mathcal{F}'(P_k))(K_1) = \mathcal{F}P_k, \tag{11}$$
$$K_2 - \gamma h(\mathcal{F}'(P_k))(K_2) = \mathcal{F}(P_k + hK_1) - 2K_1. \tag{12}$$

Denoting $\hat{A}_k = \gamma h(A - SP_k) - \frac{1}{2}I$ and rewriting (11) and (12) similar to (9), we can write the method as:

$$P_{k+1} = P_k + \frac{3}{2}hK_1 + \frac{1}{2}hK_2,$$
$$\hat{A}_k^T K_1 + K_1\hat{A}_k + \gamma hC^T K_1 C = -\mathcal{F}P_k, \tag{13}$$
$$\hat{A}_k^T K_2 + K_2\hat{A}_k + \gamma hC^T K_2 C = -\mathcal{F}(P_k + hK_1) + 2K_1. \tag{14}$$

Thus, two generalized Lyapunov equations (13), (14) have to be solved in every step. Rewriting the right hand side of (14) as

$$\begin{aligned}-\mathcal{F}P_k + h^2 K_1 S K_1 + C^T(P_k + hK_1)C& \\ -(h(A_{k+1} - S_{k+1}P_k) - I)^T K_1 - K_1(h(A_{k+1} - S_{k+1}P_k) - I)&,\end{aligned}$$

the method can be written in an efficient way.

In the case of multidimensional Brownian motion, the application of a Rosenbrock method yields essentially the same equations as above, except that the generalized Lyapunov equations are now of the form

$$F_k^T P + PF + \sum_{j=1}^{\nu} C_j^T P C_j = -H.$$

Implicit methods. According to [16], the application of any implicit method to the deterministic DRE yields an algebraic Riccati equation (ARE) to be solved in every step. As the structure of the SRE differ from the DRE in the term C^TPC, the application of an implicit method to a SRE yields an ARE with an extra term to be solved in each stage. As an illustration let us consider the Backward Differentiation Formulae (BDF) method. Applying the BDF method to the SRE we obtain the matrix valued BDF scheme

$$P_{k+1} = \sum_{j=1}^{p} -\alpha_j P_{k+1-j} + h\beta \mathcal{F} P_{k+1},$$

where α_j, β are the coefficients of the p-step BDF formula, see [3]. This leads to the difference equation

$$\begin{aligned} -P_{k+1} + h\beta(Q{+}A^TP_{k+1} + P_{k+1}A - P_{k+1}SP_{k+1} + C^TP_{k+1}C) \\ -\sum_{j=1}^{p}\alpha_j P_{k+1-j} = 0, \end{aligned}$$

and after re-arranging terms we see that this is the generalized ARE

$$\begin{aligned} \mathcal{R}(P_{k+1}) := (h\beta A - \frac{1}{2}I)^TP_{k+1} + P_{k+1}(h\beta A - \frac{1}{2}I) + h\beta C^TP_{k+1}C \\ + (h\beta Q - \sum_{j=1}^{p}\alpha_j P_{k+1-j}) - P_{k+1}(h\beta S)P_{k+1} = 0\ . \end{aligned} \tag{15}$$

The computation of P_{k+1} by Newton's method yields a generalized Lyapunov equation to be solved in each step. If Δ denotes the increment $P_{k+1} \leftarrow P_{k+1} + \Delta$ of the Newton step, then these equations take the form

$$\breve{A}_{k+1}^T\Delta + \Delta\breve{A}_{k+1} + C^T\Delta C = \frac{1}{h\beta}\mathcal{R}(P_{k+1}) \tag{16}$$

with $\breve{A}_{k+1} = A - \frac{1}{2h\beta}I - SP_{k+1}$.

2.1. *Generalized algebraic Lyapunov and Riccati equations*

The most basic operation in both Rosenbrock and BDF methods applied to SREs requires the solution of generalized Lyapunov equations such as

$$F^TX + XF + \sum_{j=1}^{\nu} C_j^TXC_j = -Y\ , \tag{17}$$

where F is replaced e.g. by $\bar{A}_k$ in (9), $\tilde{A}_k$ in (10), $\hat{A}_k$ in (13) and (14), or $\breve{A}_{k+1}$ in (16). Typically, the right hand side is assumed to be nonpositive definite, i.e. $Y \geq 0$ and the solution X is required to be nonnegative definite, i.e. $X \geq 0$. This is relevant, for instance, when we consider (low rank) factorizations $X = ZZ^T$. In the following let $F, C_j \in \mathbb{R}^{n\times n}$, $j = 1, \ldots, \nu$ and define the linear mappings $\mathcal{L}, \Pi : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ by

$$\mathcal{L}(X) = F^TX + XF\ , \quad \Pi(X) = \sum_{j=1}^{\nu} C_j^TXC_j\ . \tag{18}$$

Here $\mathcal{L}$ is a Lyapunov operator, and Π is positive in the sense that $X \geq 0$ implies $\Pi(X) \geq 0$. By $\sigma(T)$ we denote the spectrum of a linear mapping T, and by $\rho(T)$ its spectral radius. It is well-known that $\sigma(\mathcal{L}) = \{\lambda + \bar{\mu} \mid \lambda, \mu \in \sigma(F)\}$, see e.g. [27, Theorem 4.4.5]. According to a generalization of Lyapunov's matrix theorem in [43], equation (17) has a unique solution $X \geq 0$, if $\sigma(F) \subset \mathbb{C}_-$ and $\rho(\mathcal{L}^{-1}\Pi) < 1$, which we will assume in the following. It is easy to see that this assumption holds for suffciently small $h > 0$ in all our equations (9), (10), (13), (14), and (16). Conversely, our assumption implies an upper bound for the admissible step size in the methods described above. A direct solution of (17), e.g. based on a Kronecker product representation of $\mathcal{L}$ and Π has prohibitive complexity $\mathcal{O}(n^6)$. Therefore, iterative methods are preferable. Since for the standard Lyapunov equation (without Π) efficient methods like the Bartels-Stewart algorithm or low-rank alternating direction iteration (ADI) techniques are available (with complexity $\mathcal{O}(n^3)$ or less), it is natural to exploit the given splitting $\mathcal{L} + \Pi$ and to use $\mathcal{L}$ as a preconditioner. The iterative scheme

$$X_{k+1} = -\mathcal{L}^{-1}\Pi(X_k) - \mathcal{L}^{-1}(Y) \tag{19}$$

is convergent, because $\rho(\mathcal{L}^{-1}\Pi) < 1$. A common variant of (19) is obtained if the preconditioner $\mathcal{L}$ is replaced by an ADI-approximation, see [15]. In [4] and [44] these approaches have been extended to allow for low-rank approximations to X. We sketch the low-rank ADI-based variant suggested by [4], which we use in our computations. The underlying assumption is that Y and X can be approximated well by low rank factorizations $Y \approx Y_0 Y_0^T$ and $X \approx ZZ^T$.

For given positive shift parameters $p_0, p_1, \ldots$, consider the iteration

$$\begin{aligned} X_{k+1} &= (F - p_k I)^{-T}(F + p_k I)^T X_k (F + p_k I)(F - p_k I)^{-1} \\ &\quad + 2p_k (F - p_k I)^{-T} \left(\Pi(X_k) + Y_0^T Y_0\right)(F - p_k I)^{-1}\,, \end{aligned} \tag{20}$$

which is a convergent modification of (19). If $X_0 = 0$ then $X_1 = Z_1 Z_1^T$ with $Z_1 = \sqrt{2p_k}(F - p_k I)^{-1} Y_0$ and $X_{k+1} = Z_{k+1} Z_{k+1}^T$ with

$$Z_{k+1} = (F - p_k I)^{-1}\left[(F + p_k I)Z_k, \sqrt{2p_k}C_1 Z_k, \ldots, \sqrt{2p_k}C_\nu Z_k, \sqrt{2p_k}Y_0\right]. \tag{21}$$

Thus the iteration can be carried out for the factors Z_k of X_k.

Remark 2.1. (a) Note that if Y_0 has r columns, then Z_k has $(\nu + 2)^k r$ columns. To keep the rank of X_k small, in each step a column compression is applied, where Z_k is replaced by a truncated singular value decomposition with fixed error tolerance.

(b) Note also that the special structure of F can be exploited in evaluating (21). In our applications, F is of the form $F = A - SP$, where both S and P are of low rank. This low-rank perturbation can be efficiently handled by use of the Woodbury matrix inversion formula. Furthermore, the matrix A typically arises from the discretization of a partial differential operator. The evaluation of the remaining $(A - p_k I)^{-1}$ can thus be done efficiently using specialized sparse solvers.

(c) A good choice of ADI-parameters is also essential. Following [4] we use the optimal shifts for the *standard case* from [48], see also [15].

We summarize these low-rank implementation details in Algorithm 1 for our typical case, $F = (A - cI - SP)$, where c is a positive constant. The main difference to [4] is the use of the Woodbury matrix inversion formula to handle the SP term. We note that the ordering of the computations indicated by the parentheses in e.g. $Z(Z^T B R^{-1}(B^T V_i))$ are critical to minimize the storage requirements and computational effort.

Apart from the generalized Lyapunov equation, the generalized algebraic Riccati equation (15) plays a role in the BDF method. It is of the general form

$$\mathcal{R}(X) = F^T X + XF + \sum_{j=1}^{\nu} C_j^T X C_j + Q - XSX = 0\,. \tag{22}$$

Hamiltonian methods designed for standard Riccati equations can not be extended to (22) in an obvious way. Instead, Newton type iterations are typically used see e.g. [14, 50]. These exhibit a remarkable non-local convergence behaviour. Let again $\mathcal{L}$ and Π be defined by (18). If $\sigma(\mathcal{L} + \Pi) \subset \mathbb{C}_-$ (which in our applications is fulfilled for sufficiently small h), then the Newton iteration applied to (22) starting at $X_0 = 0$ is guaranteed to converge to the largest solution of (22). As mentioned above, each step of the Newton iteration requires the solution of a generalized Lyapunov equation. Pseudo-code for this in the context of the first-order BDF method is given in Algorithm 3, while the simpler first-order Rosenbrock method is outlined in Algorithm 2.

Algorithm 1 Solving $F^TX + XF + \sum_{j=1}^{\nu} C_j^T X C_j = -Y$ with $F = (A - cI - SP)$ in low-rank form

```
Input: Low-rank factors $Z \in \mathbb{R}^{N\times r_Z}$ and $W \in \mathbb{R}^{N\times r_W}$ such that $P = ZZ^T$ and $Y = WW^T$, shifts
    $p_0, p_1, \ldots$, relative tolerance TOL, maximum number of iterations $M$.
Output: Low-rank factor $V$ such that $X = VV^T$.
 1. Set $V_1 = 0$
 2. for $i = 1, \ldots, M$ do
 3.   Set $U_0 = (A + (p_i - c)I)V_i - Z(Z^TBR^{-1}(B^TV_i))$
 4.   Form $U = [U_0, \sqrt{2p_i}C_1V_i, \ldots, \sqrt{2p_i}C_\nu V_i, \sqrt{2p_i}W]$
 5.   Solve $(A - (c + p_i)I)Y_1 = U$ and $(A - (c + p_i)I)Y_2 = Z$
 6.   Solve $(I - B^TY_2(Z^TBR^{-1}))Y_3 = B^TY_1$
 7.   Column compress $V_{i+1} \approx Y_1 + Y_2(Z^TBR^{-1}Y_3)$
 8.   {Compute residual of $V_{i+1}V_{i+1}^T$, where $V_{i+1} \in \mathbb{R}^{N\times r}$}:
 9.   Set $R_L = \big[(A - cI)V_{i+1} - Z(Z^TBR^{-1}(B^TV_{i+1})), V_{i+1}, C_1V_{i+1}, \ldots, C_\nu V_{i+1}, W\big]$
10.   Set $R_D = \begin{bmatrix} 0 & I_r & 0 \\ I_r & 0 & 0 \\ 0 & 0 & I_{\nu r + r_W} \end{bmatrix}$
11.   if $\sqrt{\operatorname{trace}\left((R_L^TR_LR_D)^2\right)} < \text{TOL}$ then
12.     break
13.   end if
14. end for
15. $V = V_{i+1}$
```

Algorithm 2 Step k of the low-rank Ros1 method

```
Input: Low-rank factors $Z_k \in \mathbb{R}^{N\times r_Z}$ and $Q_0 \in \mathbb{R}^{N\times r_Q}$ such that $P_k = Z_kZ_k^T$ and $Q = Q_0Q_0^T$
Output: Low-rank factor $Z_{k+1}$ such that $P_{k+1} = Z_{k+1}Z_{k+1}^T$
 1. Column-compress $W \approx \big[Q_0, Z_k/\sqrt{h}, Z_k(Z_k^TB)\big]$
 2. Solve $(A - \frac{1}{2h}I - SX_k)^TX + X(A - \frac{1}{2h}I - SX_k) + \sum_{j=1}^{\nu} C_j^TXC_j = -WW^T$ via Algorithm 1 for $X = Z_{k+1}Z_{k+1}^T$
```

3. SPLITTING METHODS

The BDF and Rosenbrock methods treat either the full vector field $\mathcal{F}$ or a linearization thereof, but the final step in both methods is to split a number of generalized Lyapunov equations. In contrast to this, a splitting method directly splits the right-hand side into parts, and considers each part separately, see e.g. [28, Section IV] or [23, Section II.5]. The splitting is thus in this case at the top level, rather than at the bottom level.

In the SRE case, we have the natural three-term decomposition

$$\mathcal{F} = \mathcal{F}_1 + \mathcal{F}_2 + \mathcal{F}_3,$$

Algorithm 3 Step k of the low-rank BDF1 method

```
Input: Low-rank factors $Z_k \in \mathbb{R}^{N\times r_Z}$ and $Q_0 \in \mathbb{R}^{N\times r_Q}$ such that $P_k = Z_k Z_k^T$ and $Q = Q_0 Q_0^T$,
    Newton tolerance TOL, maximum number of iterations $M$.
Output: Low-rank factor $Z_{k+1}$ such that $P_{k+1} = Z_{k+1} Z_{k+1}^T$
 1. Column-compress $W \approx [Q_0, Z_k/\sqrt{h}]$
 2. Set $V_1 = Z_k$
 3. for $i = 1, \dots, M$ do
 4.    Solve $\check{A}^T X + X\check{A}^T + \sum_{j=1}^{\nu} C_j^T X C_j = -WW^T$ with $\check{A} = A - \frac{1}{2h}I - SV_jV_j^T$, via Algorithm 1, for $X = V_{j+1}V_{j+1}^T$
 5.    {Compute residual of $X$, where $V_{i+1} \in \mathbb{R}^{N\times r}$}:
 6.    Set $R_L = \left[A^T V_{i+1}, V_{i+1}, C_1 V_{i+1}, \dots, C_\nu V_{i+1}, Q_0, V_{i+1}\left(V_{i+1}^T B\right)\right]$
 7.    Set $R_D = \begin{bmatrix} 0 & I_r & 0 & 0 \\ I_r & 0 & 0 & 0 \\ 0 & 0 & I_{\nu r + r_Q} & 0 \\ 0 & 0 & 0 & R^{-1} \end{bmatrix}$
 8.    if $\sqrt{\operatorname{trace}\left((R_L^T R_L R_D)^2\right)} <$ TOL then
 9.       break
10.    end if
11. end for
12. $Z_{k+1} = V_{j+1}$
```

where

$$
\begin{aligned}
\mathcal{F}_1 P &= A^T P + PA + Q, \\
\mathcal{F}_2 P &= \sum_{j=1}^{\nu} C_j^T P C_j = \Pi(P) \quad \text{and} \\
\mathcal{F}_3 P &= -PSP.
\end{aligned}
$$

In the following we consider only the case of one-dimensional Brownian motion, i.e. $\mathcal{F}_2 P = C^T PC$, but the approach extends directly to the multidimensional case.

The idea is that it is easier or cheaper to approximate the solution to each of the subproblems $\dot{P} = \mathcal{F}_k P$ than to the full problem $\dot{P} = \mathcal{F}P$. Let $\mathcal{T}_k(t)P_0$ denote the solution to the subproblem $\dot{P} = \mathcal{F}_k P$, $P(0) = P_0$. Then the simplest splitting scheme is given by the iteration

$$P_{k+1} = \mathcal{T}_1(h)\mathcal{T}_2(h)\mathcal{T}_3(h)P_k,$$

i.e. we simply evolve the dynamics of $\mathcal{F}_3$ over one time step, then consider the $\mathcal{F}_2$ dynamics and finally $\mathcal{F}_1$.

As shown in [46], we can give explicit representations for $\mathcal{T}_1(t)$ and $\mathcal{T}_3(t)$, and these can be efficiently evaluated in a low-rank setting. Unfortunately, there does not seem to be a similarly useful representation for $\mathcal{T}_2(t)$. However, as the matrix C typically only contains a few small elements and does not give rise to a stiff $\mathcal{F}_2$, we can approximate the action of $\mathcal{T}_2(t)$ well with an explicit numerical method. We therefore consider the following modified Lie and Strang splitting schemes, given by

$$
\begin{aligned}
P_{k+1} &= \mathcal{T}_1(h)(I + h\mathcal{F}_2)\mathcal{T}_3(h)P_k \quad \text{and} \\
P_{k+1} &= \mathcal{T}_1(h/2)\mathcal{T}_3(h/2)(I + h\mathcal{F}_2 + h^2/2\mathcal{F}_2^2)\mathcal{T}_3(h/2)\mathcal{T}_1(h/2)P_k,
\end{aligned}
$$

respectively. Here, either the explicit Euler method or the midpoint rule is used to approximate the action of $\mathcal{T}_2(h)$.

In the rest of this section, we briefly recap how to implement these methods in a low-rank fashion. For details, we refer to [46]. As previously, let $P = ZZ^T$ and $Q = Q_0 Q_0^T$ be given low-rank factorizations, and consider first $\mathcal{T}_1(h)P$. This can be written as

$$\mathcal{T}_1(h)P = \mathrm{e}^{hA^T} P \mathrm{e}^{hA} + \int_0^h \mathrm{e}^{sA^T} Q \mathrm{e}^{sA} \mathrm{d}s.$$

By approximating the integral by a (high-order) quadrature formula with weights w_k and nodes τ_k we see that a low-rank approximation WW^T to $\mathcal{T}_1(h)ZZ^T$ is given by

$$W = \left[\mathrm{e}^{hA^T} Z, \sqrt{hw_1}\mathrm{e}^{\tau_1 A^T} Q_0, \sqrt{hw_2}\mathrm{e}^{\tau_2 A^T} Q_0, \ldots, \sqrt{hw_s}\mathrm{e}^{\tau_s A^T} Q_0\right].$$

The above notation means that the matrices have been placed side by side. As it is highly probable that this introduces many linearly dependent columns, a column compression technique should be applied to W to find a more suitable low-rank factor.

Consider next the middle term $\mathcal{F}_2$. Clearly,

$$W = \left[Z, \sqrt{h}C^T Z\right] \quad \text{and} \quad W = \left[Z, \sqrt{h}C^T Z, h/\sqrt{2}\, C^T C^T Z\right]$$

are low-rank factors of $(I + h\mathcal{F}_2)P$ and $(I + h\mathcal{F}_2 + h^2/2\mathcal{F}_2^2)$, respectively. Again, column compression should be applied after forming these factors.

Finally, one can show that the solution to the third subproblem is given by

$$\mathcal{T}_3(h)P = (I + hPS)^{-1}P,$$

and an application of the Woodbury matrix inversion formula shows that

$$(I + hZZ^T S)^{-1} ZZ^T = Z(I + hZ^T SZ)^{-1} Z^T.$$

Since S is positive semi-definite, we can Cholesky factorize $(I + hZ^T SZ)^{-1} = LL^T$, and thereby acquire a low-rank factorization $W = ZL$. This operation is cheap since $(I + hZ^T SZ)^{-1}$ is a small matrix when Z has few columns. We note that no column compression is needed here, as the new low-rank factor W has exactly the same rank as Z.

These three computations constitute one step of the Lie splitting procedure, which we summarize in Algorithm 4. The Strang splitting case is analogous. Both methods require that a low-rank factor of the integral term is given, and this should only be computed once at the start of the integration. For completeness, we summarize also this in Algorithm 5.

Algorithm 4 Step k of the low-rank Lie splitting method

```
Input: Low-rank factors $Y$ and $Z_k$ such that $YY^T \approx \int_0^h \mathrm{e}^{sA^T} Q \mathrm{e}^{sA} \mathrm{d}s$ and $P_k = Z_k Z_k^T$
Output: Low-rank factor $Z_{k+1}$ such that $P_{k+1} = Z_{k+1} Z_{k+1}^T$
1. Cholesky factorize $I + hZ_k^T S Z_k =: LL^T$
2. Solve $W_3 L^T = Z_k$
3. Column-compress $W_2 \approx [W_3, \sqrt{h}C^T W_3]$
4. Compute $X = \mathrm{e}^{hA^T} W_2$
5. Column-compress $Z_{k+1} \approx [X, Y]$
```

Remark 3.1. We could also consider a splitting where $\mathcal{F}_1 P = A^T P + PA$ and $\mathcal{F}_2 P = C^T PC + Q$. This means that the evaluation of the action of $\mathcal{T}_1(h)$ is simplified, as the integral term disappears. However, experimentally this leads to an error amplification with a factor 3-4 for the modified Lie splitting and 6-8 for modified Strang. We therefore argue that the one-time approximation of the integral term is worthwhile.

Algorithm 5 Computing the low-rank factor of $\int_0^h \mathrm{e}^{sA^T} Q \mathrm{e}^{sA} \mathrm{d}s$

```
Input: Low-rank factor $Q_0$ such that $Q = Q_0 Q_0^T$. Quadrature weights $w_k$ and nodes $\tau_k$ for $k = 0, \dots, s$.
Output: Low-rank factor $Y$ such that $YY^T \approx \int_0^h \mathrm{e}^{sA^T} Q \mathrm{e}^{sA} \mathrm{d}s$
1. for $k = 0$ to $s$ do
2.     Compute $X_k = \mathrm{e}^{\tau_k A^T} Q_0$
3. end for
4. Column-compress $Y \approx \left[\sqrt{hw_1} X_1, \sqrt{hw_2} X_2, \dots, \sqrt{hw_s} X_s\right]$
```

Remark 3.2. If C is symmetric with many zero eigenvalues, we can use the exact solution for the $\mathcal{F}_2$ subproblem rather than explicit approximations. Denote by $\operatorname{vec} P$ the vectorization of P, i.e. the vector formed by stacking its columns on top of each other. By the well-known formula

$$\operatorname{vec} ABC = (C^T \otimes A) \operatorname{vec} B,$$

see e.g. [27, Lemma 4.3.1], we get

$$\operatorname{vec} \mathcal{T}_2(t) P_0 = \mathrm{e}^{tC^T \otimes C^T} \operatorname{vec} P_0. \tag{23}$$

Consider first a diagonal C such that $C_{i,i} = 0$ for $i > k$. In this case, also $\mathrm{e}^{tC^T \otimes C^T}$ is diagonal and $\mathcal{T}_2$ modifies only the upper left $k \times k$ part of P_0. We can write this modification as $E \odot (ZZ^T)_{1:k,1:k}$, where $E_{i,j} = \mathrm{e}^{tc_i c_j} - 1$. By the Schur product theorem [26, Theorem 7.5.3 and Corollary 7.5.9], both E and $E \odot (ZZ^T)_{1:k,1:k}$ are positive semi-definite. Hence if k is reasonably small, we can cheaply compute a low-rank factorization

$$WW^T := E \odot (Z_{1:k,:} Z_{1:k,:}^T).$$

A low-rank factorization YY^T of $\mathcal{T}_2(t) ZZ^T$ is then given by

$$Y = \begin{bmatrix} Z, W_0 \end{bmatrix},$$

where W_0 is formed by adding $N - k$ zeros in every column of W.

Now assume that $C = VDV^T$ is a reduced eigendecomposition, where $D \in \mathbb{R}^{k \times k}$ is diagonal and $V \in \mathbb{R}^{N \times k}$ is such that $V^T V = I$. Then it is easily verified that

$$\mathrm{e}^{hC^T \otimes C^T} = (V \otimes V) \mathrm{e}^{hD \otimes D} (V^T \otimes V^T).$$

Hence, this case reduces to the procedure described above for diagonal C, with the difference that it should be applied to $V^T Z$ instead of Z and one should form VW instead of W. Unfortunately, it is unclear how to generalize this approach to the nonsymmetric case.

4. NUMERICAL EXPERIMENTS

In order to validate the accuracy of the described methods, as well as compare their respective efficiency, we present the results of three different numerical experiments. The algorithms described in the previous sections were all implemented in MATLAB.

To compute the errors of our approximations, we need a reference solution to compare to. We are not aware of any non-trivial example where the exact solution is known, so we are faced with two possibilities: compute a reference solution by using our own methods with a smaller step size, or use a different, highly accurate method.

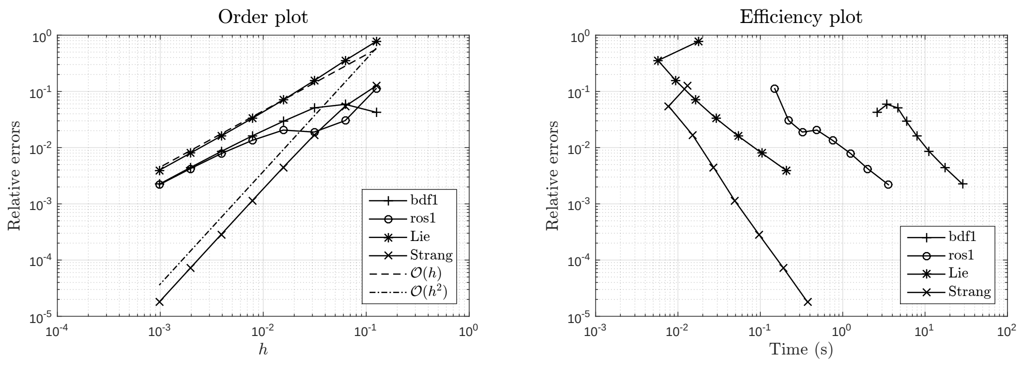

Figure 1. Left: Error versus temporal step size for the problem in Section 4.1. The methods converge with the expected orders. Right: Error versus computation time for the same problem. The splitting methods are clearly superior to the Rosenbrock method, which is itself much better than the BDF1 method. The strange bends in the curves for large step sizes are due to inaccurate timing for such small computation times.

In the first experiment, we follow the latter approach by unrolling the matrices into vectors and using MATLAB's built-in solver ODE15s. This is extremely expensive (in fact, this is one of the motivations for having an efficient large-scale solver) and we can therefore only consider a very small-scale problem. For the two latter experiments, we compute the reference solution by the Strang splitting method, with a temporal step size that is half as large as the one used for the most accurate approximation. In all cases, we compute relative errors in the temporal max-norm, that is, the error for the approximation u is given by

$$\max_{j=1,\ldots,N_T} \|u(j) - u_{\text{ref}}(j)\| \Big/ \max_{j=1,\ldots,N_T} \|u_{\text{ref}}(j)\|.$$

Additionally, we note that in all the experiments we scale the column compression tolerance by the number of steps, so that it better corresponds to the global error.

4.1. *A small-scale verification problem*

We first considere a small-scale case of 10×10 matrices, where we set A, B, C and Q to all consist of random numbers, in such a way that A is negative definite and Q is positive definite. The initial value P_0 is taken to be the zero matrix and is represented by the $n \times 1$ zero vector. We integrate the problem over the time interval $[0, 1]$, use a column compression tolerance of 10^{-6} and choose the tolerance 10^{-5} for both the ADI and Newton iterations. For the reference solution computed by MATLAB's ODE15s, we use an absolute tolerance of 10^{-10} and a relative tolerance of 10^{-8}. We apply all the different methods for each of the different temporal step sizes $h = 2^{-k}$, $k = 3, \ldots, 10$.

The results are shown in Figure 1. The left plot shows the error plotted against the step size in a loglog-scale, which demonstrates that the methods all converge to the solution of the problem. The splitting methods exhibit the expected order of convergence for all the step sizes, while the BDF and Rosenbrock have some trouble for the larger step sizes. This behavior is handled somewhat by using a lower tolerance for the Lyapunov equation solver, but this of course also increases the computational cost. The right plot shows the error plotted against the computation time, which demonstrates that the splitting schemes are better in terms of efficiency. For errors close to $4 \cdot 10^{-3}$, the BDF1 method is almost 10 times as slow as the Rosenbrock method which in turn is about 10 times as slow as the Lie splitting. As expected, the second-order Strang splitting is faster than all the other first-order methods for small step sizes, but also for the large step sizes.

4.2. *A medium-scale problem*

Next, we consider a SLQR problem as described in Section 1, for a stochastic heat transfer model. We let A be the discretization of the Laplacian on the unit square $[0, 1]^2$ and employ fixed boundary conditions given by $x = u_j$, $j = 1, 2, 3$, on three of the edges. On the final edge, we use the Robin

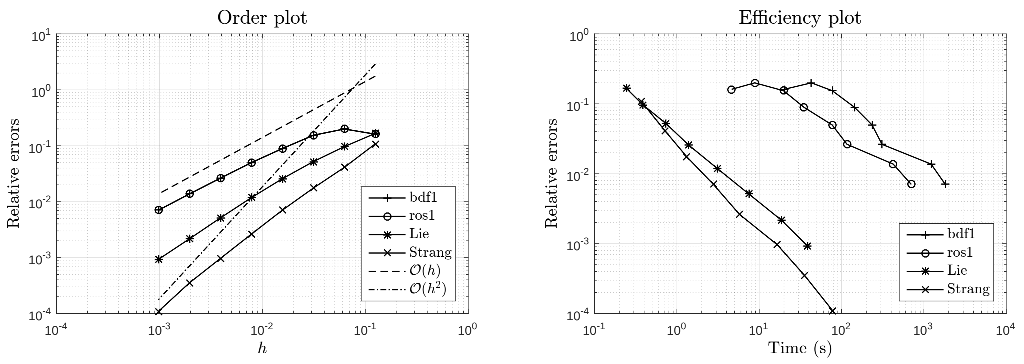

Figure 2. Left: Error versus temporal step size for the problem in Section 4.2. The methods converge with the expected orders except for the Strang splitting that exhibits order reduction for larger step sizes. Right: Error versus computation time for the same problem. Again the splitting methods are superior to the Rosenbrock method, which is itself better than the BDF1 method.

condition $n\nabla x = 0.5(0.5 + dW)x$. That is, the temperature can be directly controlled on three sides, and the leakage along the final edge is noise-dependent. After spatial discretization, this gives a full matrix $B \in \mathbb{R}^{n\times 3}$ and a matrix $C = C_1 \in \mathbb{R}^{n\times n}$ with only n nonzero elements that are located on the diagonal. The output is taken to be the mean of x, which gives $Q = Q_0 Q_0^T$ where $Q_0 \in \mathbb{R}^{n\times 1}$ has constant elements. Finally, we choose $R = I$. This problem has also been used in e.g. [6] and a similar problem has been described in detail in e.g. [5]. We therefore refer to the latter article for details on the spatial discretization.

We take $n_x = 25$ points in each spatial direction, which gives a system of size $n = 25^2 = 625$. Further, we use the temporal step sizes $h = 2^{-k}$, $k = 3, \dots, 10$, on the time interval $[0, 1]$, a tolerance of 10^{-4} for the Lyapunov equation solver and BDF Newton iteration, and a column compression tolerance of 10^{-12}. The results are shown in Figure 2. The left plot demonstrates that all the first-order methods converge to the solution of the problem as expected, but the Strang splitting seems to suffer from a slight order reduction except for at the smallest step sizes. This calls for a proper error analysis, but that is out of the scope of this article. The efficiency plot to the right is similar to that of the previous problem in that the splitting methods are clearly superior and that the BDF method is slower than the Rosenbrock method.

In the above, we have used a very low column compression tolerance. This means that only the extra linear dependency introduced by e.g. concatenating matrices in the ADI iteration are eliminated. Increasing this tolerance means that the column compression becomes a true truncation operator, where not all the information in the problem is retained. However, if the error introduced by this operator is sufficiently small, it will not affect the convergence of the methods. For the splitting schemes, it is enough that the error is of the same size as the local error, while for the BDF and Rosenbrock methods also the influence on the ADI iteration needs to be taken into account. In lieu of a rigorous error analysis, we present here the results of using two larger tolerances but with all other parameters unchanged. Figure 3 corresponds to a tolerance of 10^{-6} and Figure 4 to 10^{-4}; using even larger values lead to ADI convergence failures. We observe that the errors are essentially identical, while all the methods become faster due to working with approximations of lower rank. The cost of the column compression itself also decreases with smaller rank, but this is not a major part of the computational effort. In this example, it constitutes between 5 and 20% of the overall computation time, depending on method and tolerance. Regardless of these issues, the main conclusion is still that splitting methods are clearly superior for problems such as these.

We note that the tolerances for the ADI and Newton iteration cannot be chosen much larger without affecting the global error here, but in general these are also design parameters. See e.g. [7, 18] for ideas on how to choose these appropriately in the ARE case, and how to improve the basic Newton iteration in general. It seems likely that this theory extends to the generalized setting.

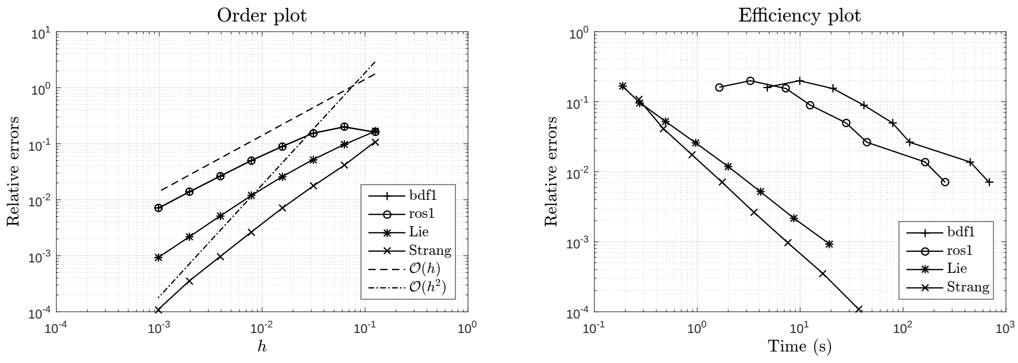

Figure 3. The results of the same experiment as in Figure 2, except for using a column compression tolerance of 10^{-6}. We note that the errors are essentially identical, while all the methods become faster.

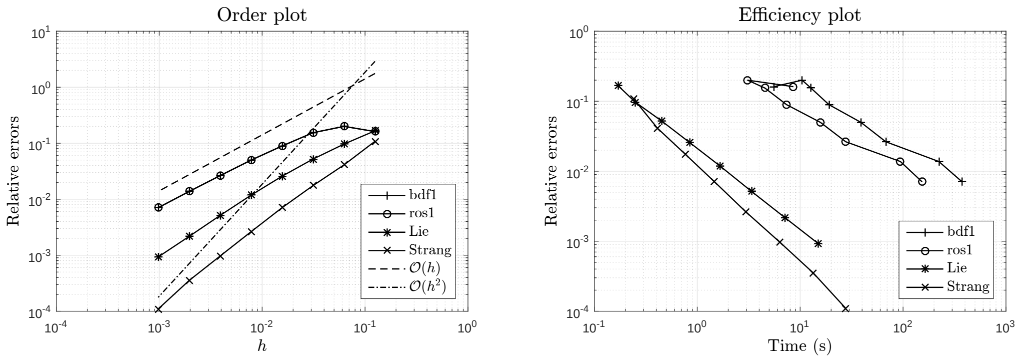

Figure 4. The results of the same experiment as in Figure 2, except for using a column compression tolerance of 10^{-4}. We note that the errors are essentially identical, while all the methods become faster.

Finally, we note that this problem is not very large either, and this is due to the excessive computation times required for the BDF and Rosenbrock methods. Essentially, the BDF and Rosenbrock methods have to solve an algebraic Riccati equation in each time step, and the computation time per step is in line with previous results on algebraic Riccati equations, see e.g. [11]. Hence we do not expect to be able to significantly improve the efficiency of the implementation for these methods, even by utilising recent algorithmic progress in solving large-scale AREs and Lyapunov equations [10].

4.3. A large-scale problem

As a final experiment, we consider also the previous problem but with $n_x = 100$. This yields a problem size of $n = 10000$. In view of the previous section, applying the BDF and Rosenbrock methods is unfeasible and we therefore only use the splitting methods in this case. We again use the temporal step sizes $h = 2^{-k}$, $k = 3, \dots, 12$, on the time interval $[0, 1]$, and a column compression tolerance of 10^{-6}. The results are shown in Figure 5.

We observe that the Lie splitting exhibits approximately order 1 convergence, and for small step sizes the Strang splitting approaches the expected order 2 behaviour. However, for larger step sizes, the order reduction that was already observed in the previous example is more pronounced. In addition, for the largest step sizes loss of stability also seems to be a problem. However, since these temporal step sizes are considerably larger than the spatial mesh size, this is a minor issue. Again, this behaviour clearly calls for a proper error analysis.

It is also interesting to note that due to the peculiar error behaviour, the Strang splitting is now only more efficient when small errors are required. If less accurate approximations are required, the Lie splitting can be used instead. It should be noted, however, that the version of Strang splitting

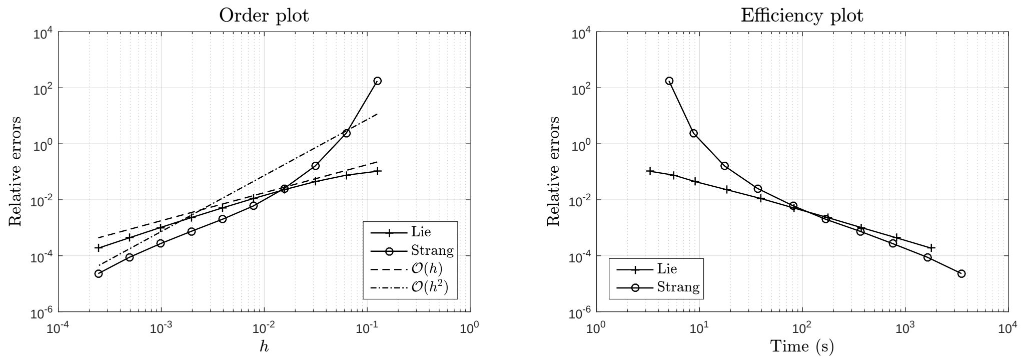

Figure 5. Left: Error versus temporal step size for the problem in Section 4.3. The splitting methods converge to the solution with convergence orders that for small step sizes are approximately 1 and 2, respectively. For larger step sizes, the Strang splitting seems to be affected by order reduction as well as stability issues. Right: Error versus computation time for the same problem. We see that now the Strang splitting is only more efficient if small errors are required, and for larger errors the Lie splitting is more efficient.

proposed here uses the most expensive ordering of the operators. This choice was made in order that the stiff operator F_1 be evaluated last. This is often beneficial, see e.g. [45], but the effects of different, less expensive, orderings should be investigated when considering a specific application.

5. CONCLUSIONS

We have studied the numerical solution of the finite horizon stochastic linear quadratic optimal control problem. The focus was on large-scale Riccati equations arising in problems governed by stochastic partial differential equations and we have proposed methods based on low-rank matrix versions of the Rosenbrock, Backward Differentiation Formulas and splitting methods. In general, splitting methods seem to be one order faster than the other methods in this matrix setting. Having an efficient solver for SREs will enable one to deal with real life applications.

ACKNOWLEDGEMENTS

The authors would like to thank Peter Benner and Tobias Breiten for providing their code for computing generalized Lyapunov and generalized algebraic Riccati equations. H. Mena was supported by the project *Solution of large-scale Lyapunov Differential Equations* (P 27926) founded by the Austrian Science Foundation FWF. T. Stillfjord was partially supported by the Swedish Research Council.

REFERENCES

1. Antoulas A, Sorensen D, Zhou Y. On the decay rate of Hankel singular values and related issues. *Syst. Control Lett.* 2000; 46:323–342.
2. Arias E, Hernández V, Ibanes J, Peinado J. A family of BDF algorithms for solving Differential Matrix Riccati Equations using adaptive techniques. *Procedia Computer Science* 2010; 1:2569–2577.
3. Ascher U, Petzold L. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations* SIAM, Philadelphia; 1998.
4. Benner P, Breiten T. Low rank methods for a class of generalized Lyapunov equations and related issues. *Numer. Math.* 2013; 124(3):441–470.
5. Benner P, Damm T. Lyapunov equations, energy functionals, and model order reduction of bilinear and stochastic systems. *SIAM J. Control Optim.* 2011, 49; 2:686–711.
6. Benner P, Damm T, Rodriguez Cruz Y. Dual pairs of generalized Lyapunov inequalities and balanced truncation of stochastic linear systems. *IEEE Trans. Autom. Control* 2016, accepted for publication.
7. Benner P, Heinkenschloss M, Saak J, Weichelt H. K. An inexact low-rank Newton-ADI method for large-scale algebraic Riccati equations. *Appl. Numer. Math.* 2016, 108:125–142
8. Benner P, Mena H. Numerical solution of the infinite-dimensional LQR-problem and the associated differential Riccati equations. *J. Numer. Math.* 2016, accepted for publication. DOI: 10.1515/jnma-2016-1039

9. Benner P, Mena H. Rosenbrock methods for solving differential Riccati equations. *IEEE Trans. Autom. Control* 2013; 58:2950–2957.
10. Benner P, Kürschner P, Saak J. Efficient handling of complex shift parameters in the low-rank cholesky factor ADI method. *Numer. Algorithms*, 62: 225–251.
11. Benner P, Li JR, Penzl T. Numerical solution of large Lyapunov equations, Ricatti equations, and linear-quadratic control problems. *Numer. Linear Algebra Appl.* 2008, 15; 9:755–777.
12. Bismut JM. Linear quadratic optimal stochastic control with random coefficients. *SIAM J. Control. Optim.* 1976; 14:419-444.
13. Bismut JM. *Contrôle des systémes linéaires quadratiques: applications de l'intégrale stochastique.* In: Séminaire de Probabilités, XII. Lecture Notes in Math., 649. Springer, Berlin; 1978.
14. Damm T. *Rational Matrix Equations in Stochastic Control.* Number 297 in Lecture Notes in Control and Information Sciences, Springer; 2004.
15. Damm T. Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations. *Numer. Linear Algebra Appl.* 2008, 15; 7:853–871.
16. Dieci L. Numerical integration of the differential Riccati equation and some related issues. *SIAM J. Numer. Anal.* 1992, 29; 3:781–815.
17. Da Prato G. Direct solution of a riccati equation arising in stochastic control theory. *Appl. Math. Optim.* 1984; 11: 191–208.
18. Feitzinger F, Hylla T, Sachs E. W. Inexact Kleinman-Newton method for Riccati equations. *SIAM J. Matrix Anal. Appl.* 2009; 31(2):272–288.
19. Flandoli F. Direct solution of a Ricatti equation arising in a stochastic control problem with control and observation on the boundary. *Appl. Math. Optim.* 1986; 14:107–129.
20. Grasedyck L. Existence of a low rank or H-matrix approximant to the solution of a sylvester equation. *Numer. Linear Algebra Appl.* 2004; 11:371–389.
21. Guatteri G, Tessitore G. On the bacward stochastic Ricatti equation in infinite dimmensions. *SIAM J. Control Optim.* 2005, 44, 1:159–194.
22. Guatteri G, Tessitore G. Backward stochastic Ricatti equations in infinite horizon l-q optimal control with infinite dimensional state space and random coefficients. *Appl. Math. Optim.* 2008; 57:207–235.
23. Hairer E, Lubich C, Wanner G. *Geometric Numerical Integration*, vol. 31, Springer Series in Comput. Math. Berlin, Germany: Springer; 2006.
24. Hafizoglu C. *Linear quadratic regulatory boundary/point control of stochastic partial differential equations systems with unbounded coefficients.* Ph.D. Thesis, University of Virginia, US; 2006.
25. Hafizoglu C, Lasiecka I, Levajković T, Mena H, Tuffaha A. The stochastic linear quadratic problem with singular estimates. *SIAM Journal on Control and Optimization* 2016; accepted for publication.
26. Horn RA, Johnson CR. *Matrix Analysis*, 2nd ed. Cambridge, U. K. : Cambridge University Press; 1985.
27. Horn RA, Johnson CR. *Topics in Matrix Analysis*. Cambridge, U. K. : Cambridge University Press; 1991.
28. Hundsdorfer W, Verwer JG, *Numerical solution of time-dependent advection-diffusion-reaction equations*. vol. 33, Springer Series in Comput. Math. Berlin, Germany: Springer; 2003.
29. Ichikawa A. Dynamic programming approach to stochastic evolution equations. *SIAM J. Control. Optim.* 1979 17, 1:152–174.
30. Kohlmann M, Tang S. *New developments in backward stochastic Ricatti equations and their applications*. In: Mathematical Finance, Konstanz. Trends Math. Birkhäuser, Basel; 2001.
31. Kohlmann M, Tang S. Global adapted solution of one-dimensional backward stochastic Ricatti equations with application to the mean-variance hedging. *Stoch. Process. Appl.* 2002, 97:1255–1288.
32. Kohlmann M, Tang S. Multidimensional backward stochastic Ricatti equations and applications. *SIAM J. Control. Optim.* 2003, 41, 1696–1721.
33. Kohlmann M, Zhou XY. Relationship between backward stochastic differential equations and stochastic controls: a linear-quadratic approach. *SIAM J. Control. Optim.* 2000; 38:1392–1407.
34. Kushner HJ. Optimal stochastic control. *IRE Trans. Autom. Control* 1962; AC-7:120–122.
35. Lang N, Mena H, Saak J. On the benefits of the LDL^T factorization for large-scale differential matrix equation solvers. *Linear Algebra Appl.* 2015, 480:44–71.
36. I. Lasiecka. *Optimal control problems and Ricatti equations for systems with unbounded controls and partially analytic generators: applications to boundary and point control problems*. Springer Verlag Lecture Notes 1855; 2004.
37. Lasiecka I, Triggiani R. Optimal control and differential Ricatti equations under singular estimates for $e^{At}B$ in the absence of analyticity. *Adv. Dyn. Control* 2004, Special Volume dedicated to A. V. Balakrishnan, Chapman and Hall/CRC Press; 271–309.
38. Lasiecka I, Triggiani R. *Control theory for partial differential equations: continuous and approximation theories I.* Abstract parabolic systems. Cambridge University Press, Cambridge, UK; 2000.
39. Levajković T, Mena H, Tuffaha A. The Stochastic Linear Quadratic Control Problem: A Chaos Expansion Approach. *Evol. Equ. Control Theory* 2016, 5; 1:105–134.
40. Levajković T, Mena H, Tuffaha A. A Numerical Approximation Framework for the Stochastic Linear Quadratic Regulator on Hilbert Spaces. *Applied Mathematics and Optimization* 2016, Springer Online First. DOI: 10.1007/s00245-016-9339-3
41. Penzl T. Eigenvalue decay bounds for solutions of Lyapunov equations: the symetric case. *Syst. Control Lett.* 2000, 40:139–144.
42. Sabino J. *Solution of Large-Scale Lyapunov Equations via the Block Modified Smith Method*. PhD thesis, Rice University, Houston, Texas, 2007.
43. Schneider H. Positive operators and an inertia theorem. *Numer. Math.* 1965; 7:11–17.
44. Shank S, Simoncini V, Szyld D. Efficient low-rank solutions of generalized Lyapunov equations. *Numer. Math.* 2016; 134:327-342.

45. Sportisse B. An Analysis of Operator Splitting Techniques in the Stiff Case. *J. Comput. Phys.* 2000; 161:140–168.
46. Stillfjord T. Low-rank second-order splitting of large-scale differential Ricatti equations. *IEEE Trans. Autom. Control* 2015, 60; 10:2791-2796.
47. Truhar N, Veselić K. Bounds on the trace of a solution to the Lyapunov equation with a general stable matrix. *Syst. Control Lett.* 2007; 56:493–503.
48. Wachspress E. *The ADI Model Problem. Computational Intelligence and Complexity*, Springer; 2013.
49. Wonham WM. On the separation theorem of stochastic control. *SIAM J. Control* 1968; 6:312–326.
50. Wonham WM. On a matrix Riccati equation of stochastic control. *SIAM J. Control Optim.* 1968; 6:681–698.
51. Yong J, Zhou XY. *Stochastic controls - Hamiltonian systems and HJB equations*. Applications of Mathematics, Stochastic Modelling and Applied Probability 43. Springer, New York; 1999.