

Unconventional Reflexive Numerical Methods  
for Matrix Differential Riccati Equations <sup>1</sup>Ren-Cang Li<sup>2</sup>

November 2000

## ABSTRACT

Matrix Differential Riccati Equations (MDREs)

$$X' = A_{21} - XA_{11} + A_{22}X - XA_{12}X, \quad X(0) = X_0,$$

where  $A_{ij} \equiv A_{ij}(t)$ , appear frequently throughout applied mathematics, science, and engineering. MDREs play particularly important roles in optimal control, filtering, estimation, and in two-point linear boundary value problems. In the past a number of unconventional numerical methods that are suited only for time-invariant MDREs have been designed, but despite their special structure, no unconventional methods that are suited for time-varying MDREs have been constructed, except (carefully) re-designed conventional linear multistep methods and Runge-Kutta methods. Implicit conventional methods which are preferred to explicit ones for stiff systems require solving nonlinear systems of equations (of possibly much higher dimensions than the original problem itself for Runge-Kutta methods) which not only pose implementation difficulties but also may be expensive because they require solving non-linear matrix equations which may be costly. We propose new unconventional reflexive methods which are suited for both time-invariant and time-varying MDREs and which requires solving no nonlinear systems but linear ones like one Sylvester equation per time-step or linear matrix systems with the same size as the MDRE. The new methods are semi-implicit and thus have much better stability properties than explicit methods and more importantly, they can be easily tailored for engineering applications, for example, some new methods can be easily coded for large sparse MDREs without much programming complications. The new methods are reflexive and thus allow simple and easily implementable palindromic compositions or extrapolations to achieve highly accurate numerical solutions whenever necessary.

---

<sup>1</sup>This report, actually finished in *December 1997*, is available on the web at <http://www.ms.uky.edu/~rcli/>.

<sup>2</sup>Department of Mathematics, University of Kentucky, Lexington, KY 40506 (rcli@ms.uky.edu). This work was supported in part by the National Science Foundation under Grant No. ACI-9721388 and by the National Science Foundation CAREER award under Grant No. CCR-9875201.

# Unconventional Reflexive Numerical Methods for Matrix Differential Riccati Equations

Ren-Cang Li\*

December 9, 1997

## Abstract

Matrix Differential Riccati Equations (MDREs)

$$X' = A_{21} - XA_{11} + A_{22}X - XA_{12}X, \quad X(0) = X_0,$$

where  $A_{ij} \equiv A_{ij}(t)$ , appear frequently throughout applied mathematics, science, and engineering. MDREs play particularly important roles in optimal control, filtering, estimation, and in two-point linear boundary value problems. In the past a number of unconventional numerical methods that are suited only for time-invariant MDREs have been designed, but despite their special structure, no unconventional methods that are suited for time-varying MDREs have been constructed, except (carefully) re-designed conventional linear multistep methods and Runge-Kutta methods. Implicit conventional methods which are preferred to explicit ones for stiff systems require solving nonlinear systems of equations (of possibly much higher dimensions than the original problem itself for Runge-Kutta methods) which not only pose implementation difficulties but also may be expensive because they require solving non-linear matrix equations which may be costly. We propose new unconventional reflexive methods which are suited for both time-invariant and time-varying MDREs and which requires solving no nonlinear systems but linear ones like one Sylvester equation per time-step or linear matrix systems with the same size as the MDRE. The new methods are semi-implicit and thus have much better stability properties than explicit methods and more importantly, they can be easily tailored for engineering applications, for example, some new methods can be easily coded for large sparse MDREs without much programming complications. The new methods are reflexive and thus allow simple and easily implementable palindromic compositions or extrapolations to achieve highly accurate numerical solutions whenever necessary.

## 1 Introduction

Matrix Differential Riccati Equations (MDREs) arise frequently throughout applied mathematics, science and engineering. They in particular play major roles in optimal control,

---

\*Department of Mathematics, University of Kentucky, Lexington, KY 40506, *email*: na.rcli@na-net.ornl.gov. This material is based in part upon work supported by the Summer Faculty Research Fellowship of the University of Kentucky.

filtering and estimation [28] and in solving linear two point boundary value problems of ordinary differential equations (ODEs) [2, 3, 13, 14]. A number of algorithms have been proposed over the past 25 years for solving MDREs numerically. These include carefully re-designed conventional Runge-Kutta methods and Linear Multi-step Methods for ODEs by Choi and Laub [7] and by Dieci [10] and unconventional methods for MDREs arising from the optimal control theory by, e.g., [8, 27, 30, 29, 32, 34, 37, 40, 43, 45]. It is known that these unconventional methods are not suited for time-varying MDREs. While the re-designed conventional methods benefit greatly from past development of sophisticated general-purposed computer programs for ODEs, they could be easily evolved into complicated programs thousands of lines long, somewhat complicated interfaces. Implicit conventional methods which are preferred to explicit ones for stiff systems require solving nonlinear systems of equations (of possibly much higher dimensions than the original problem itself for Runge-Kutta methods) which not only pose implementation difficulties but also may be expensive. Babuška and Majer [3] proposed an unconventional method that applies to both time-invariant and time-varying MDREs, but their method is only of order 2 and is not *reflexive* (see §2 for the definition). Recently Dieci and Eirola [11, 12] launched an investigation on integrators that preserve positive definiteness and/or monotonicity when exact solutions do. A surprising theoretical, though a bit of disappointing, result of theirs is that MDRE integrators from direct discretization of differential Riccati equations and preserving positive definiteness and/or monotonicity can only be of order 1, a severe order barrier. The authors then went on proposing the other two (indirect) approaches, one of which is the well-known Linear System approach (see §3.3) via Bernoulli substitutions. Unfortunately the resulted Linear Systems are known to be dichotomic [2]. Efficient implementations of Dieci's and Eirola's methods remain to be seen as they remarked.

In this paper we propose new reflexive numerical methods to solve MDREs. Because of reflexivity, these methods are at least of order 2 and moreover their orders can be increased by simple palindromic composition schemes [24, 25, 35] or extrapolations [6, 20] whenever underlying applications call for. The proposed reflexive methods will be semi-implicit in the sense that they involves no nonlinear systems of equations but linear ones like Sylvester equations or linear matrix systems with the same size as the MDRE, and they can be easily turned into effective subprograms. Because of their semi-implicitness, the proposed methods have better stability properties than explicit methods, some of them can be implemented almost as easily as explicit methods. The simplicity of our methods make it easy for them to be tried on practical problems, e.g., some new methods can be easily coded for large sparse MDREs without much programming complications.

We are interested in MDRE:

$$X' = A_{21} - X A_{11} + A_{22}X - X A_{12}X, \quad X(0) = X_0, \quad (1.1)$$

where  $X$  is  $n$ -by- $m$  matrix-valued function of time  $t$ , and all  $A_{ij}$  are matrix-valued functions

of time  $t$  too, whose dimensions are determined by the following conformal partitioning

$$\begin{matrix} & m & n \\ m & \left( \begin{array}{cc} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array} \right) \\ n & & \end{matrix}.$$

For MDREs arising in optimal control and filtering problems,  $n = m$ ,  $A_{22} \equiv -A_{11}^*$ , and  $A_{21}^* \equiv A_{12}$ , where  $(\cdot)^*$  denotes (complex) conjugate transpose. Then it is called a *Symmetric Differential Riccati Equation*<sup>1</sup>. In this case, if  $X(0)$  is symmetric,  $X(t)$  is symmetric for all  $t$  that  $X(t)$  exists.

## 2 Why Reflexive Numerical Methods?

We shall now present some basics (that are necessary for our presentation) on *Reflexive Numerical Methods* for ordinary differential equations (ODE); see, e.g., [24, 26, 25, 35] for details. Consider the following *Autonomous Initial Value Problem* (AIVP)<sup>2</sup>

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}), \quad \text{with } \mathbf{y}(0) = \mathbf{y}_0. \quad (2.1)$$

In principle, any one-step method for solving the system yields an *updating formula*  $\mathbf{Q}(\theta, \mathbf{g})$  which advances  $\mathbf{g} \approx \mathbf{y}(\tau)$  to  $\mathbf{Q}(\theta, \mathbf{g}) \approx \mathbf{y}(\tau + \theta)$ , where  $\theta$  is the step-size. An updating formula  $\mathbf{Q}(\theta, \mathbf{g})$  is *Reflexive* if

$$\mathbf{Q}(-\theta, \mathbf{Q}(\theta, \mathbf{g})) = \mathbf{g}.$$

(It has been called *Symmetric*, but we feel that the term has already been overworked.) One example is *the Implicit Mid-point Rule*:  $\mathbf{Y} = \mathbf{y} + \theta \mathbf{f}\left(\frac{\mathbf{y} + \mathbf{Y}}{2}\right)$ , which usually requires solving a nonlinear system of equations per time-step. (In the coming section, we shall create several reflexive updating formulas for MDREs, which require solving no nonlinear systems of equations.) What distinguish a reflexive updating formula from other ODE solvers are its two unique properties which consequently make it a good choice from time to time. The two properties are

1. A consistent reflexive updating formula  $\mathbf{Q}(\theta, \mathbf{g})$  is at least of 2nd order convergence which is often sufficient for engineering applications and moreover the asymptotic series for  $\mathbf{Q}(\tau, \mathbf{y}_0) - \mathbf{y}(\tau)$  contains only even powers of  $\theta$  as  $\theta \rightarrow 0$ , where  $\tau = n\theta$ ; see [21, 22, 23]. This makes extrapolation techniques particularly efficient.
2. A reflexive updating formula  $\mathbf{Q}(\theta, \mathbf{g})$ 's order can be increased by simple and easily implementable palindromic composition schemes; see [24, 26, 25, 35]. By *palindromically composing* the existing reflexive updating formula  $\mathbf{Q}(\cdot, \cdot)$  to obtain higher order methods we mean that with appropriately chosen integer  $m$  and scalar  $\delta_j$ 's

$$\mathbf{Q}(\delta_m \theta, \mathbf{Q}(\delta_{m-1} \theta, \mathbf{Q}(\cdots, \mathbf{Q}(\delta_1 \theta, \mathbf{g}) \cdots))) \quad (2.2)$$

<sup>1</sup>a *Hermitian Riccati Differential Equation* for complex cases.

<sup>2</sup>Any given system where  $\mathbf{f}$  depends on time  $t$  can be rewritten in a way that suppresses all explicit references to  $t$ . So unless otherwise stated, we shall deal with AIVP only in this section in order to simplify the formulas that will arise.

approximates  $\mathbf{y}(\tau+\theta)$  (much) more accurately than  $\mathbf{Q}(\theta, \mathbf{g})$  does, where  $\delta_i = \delta_{m-i+1}$  for  $i = 1, 2, \dots, m$ . Composition schemes of these kinds have also been discovered in special contexts like symplectic integrators for separable Hamiltonian systems [41, 17, 46, 36], composing the implicit mid-point rule [42, 9] and decompositions of exponential operators [44]. The reader is referred to Li [35, 1995] for a short history of this. Palindromic schemes (2.2) of orders as high as 10 have been constructed in [25, 35].

How do we construct reflexive updating formulas? It is generally problem-dependent. However, the following two guidelines are very useful. Later we shall see how they can be applied to MDREs.

The **Symmetrical Splitting** idea is to split  $\mathbf{f}(\mathbf{y})$  *symmetrically* into  $\mathcal{F}(\theta, \mathbf{u}, \mathbf{v})$  [24, 35] so as to satisfy both

$$\mathcal{F}(0, \mathbf{y}, \mathbf{y}) \equiv \mathbf{f}(\mathbf{y}) \quad \text{and} \quad \mathcal{F}(-\theta, \mathbf{v}, \mathbf{u}) \equiv \mathcal{F}(\theta, \mathbf{u}, \mathbf{v}). \quad (2.3)$$

Then solving

$$\mathbf{Q} = \mathbf{g} + \theta \mathcal{F}(\theta, \mathbf{g}, \mathbf{Q})$$

for  $\mathbf{Q} = \mathbf{Q}(\theta, \mathbf{g})$  gives a reflexive updating formula.

The **Partitioning** idea [23, 24, 35] is to partition a big complicated problem into a collection of simpler subproblems, each of which can be solved relatively easily for some variables assuming given constant values for the others. To solve the big problem, we iteratively solve the subproblems, using as data for each whatever approximations may be available from the solutions of the others.

Suppose we wish to solve numerically a big system of AIVP, say

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= \mathbf{a}(\mathbf{x}, \mathbf{y}, \mathbf{z}) \quad \text{with } \mathbf{x}(0) = \mathbf{x}_0, \\ \frac{d\mathbf{y}}{dt} &= \mathbf{b}(\mathbf{x}, \mathbf{y}, \mathbf{z}) \quad \text{with } \mathbf{y}(0) = \mathbf{y}_0, \\ \frac{d\mathbf{z}}{dt} &= \mathbf{c}(\mathbf{x}, \mathbf{y}, \mathbf{z}) \quad \text{with } \mathbf{z}(0) = \mathbf{z}_0, \end{aligned}$$

having already written subprograms to solve the subsystems into which the big system has been partitioned. Our subprograms have the form of consistent updating formulas that provide approximate solutions

$$\begin{aligned} \mathbf{x}(\tau) &\approx \mathbf{Q}_x(\tau, \mathbf{x}_0, \mathbf{y}, \mathbf{z}) \quad \text{for any } \tau \geq 0 \text{ and constant } \mathbf{x}_0, \mathbf{y} \text{ and } \mathbf{z}, \\ \mathbf{y}(\tau) &\approx \mathbf{Q}_y(\tau, \mathbf{x}, \mathbf{y}_0, \mathbf{z}) \quad \text{for any } \tau \geq 0 \text{ and constant } \mathbf{x}, \mathbf{y}_0 \text{ and } \mathbf{z}, \\ \mathbf{z}(\tau) &\approx \mathbf{Q}_z(\tau, \mathbf{x}, \mathbf{y}, \mathbf{z}_0) \quad \text{for any } \tau \geq 0 \text{ and constant } \mathbf{x}, \mathbf{y} \text{ and } \mathbf{z}_0. \end{aligned}$$

Assume the subprograms are *Reflexive*. Here *Reflexive* means that

$$\begin{aligned} \mathbf{Q}_x(-\theta, \mathbf{Q}_x(\theta, \mathbf{x}, \mathbf{y}, \mathbf{z}), \mathbf{y}, \mathbf{z}) &\equiv \mathbf{x}, \\ \mathbf{Q}_y(-\theta, \mathbf{x}, \mathbf{Q}_y(\theta, \mathbf{x}, \mathbf{y}, \mathbf{z}), \mathbf{z}) &\equiv \mathbf{y}, \\ \mathbf{Q}_z(-\theta, \mathbf{x}, \mathbf{y}, \mathbf{Q}_z(\theta, \mathbf{x}, \mathbf{y}, \mathbf{z})) &\equiv \mathbf{z}. \end{aligned}$$

Then a larger program that is also reflexive for the whole system is constructed thus:

Given approximations  $\mathbf{x} \approx \mathbf{x}(\tau)$ ,  $\mathbf{y} \approx \mathbf{y}(\tau)$ , and  $\mathbf{z} \approx \mathbf{z}(\tau)$ , we approximate in turn

$$\begin{aligned} \mathbf{x}(\tau + \theta/2) & \text{ by } \hat{\mathbf{X}} = \mathbf{Q}_{\mathbf{x}}(\theta/2, \mathbf{x}, \mathbf{y}, \mathbf{z}), \\ \mathbf{y}(\tau + \theta/2) & \text{ by } \hat{\mathbf{Y}} = \mathbf{Q}_{\mathbf{y}}(\theta/2, \hat{\mathbf{X}}, \mathbf{y}, \mathbf{z}), \\ \mathbf{z}(\tau + \theta) & \text{ by } \mathbf{Z} = \mathbf{Q}_{\mathbf{z}}(\theta, \hat{\mathbf{X}}, \hat{\mathbf{Y}}, \mathbf{z}), \\ \mathbf{y}(\tau + \theta) & \text{ by } \mathbf{Y} = \mathbf{Q}_{\mathbf{y}}(\theta/2, \hat{\mathbf{X}}, \hat{\mathbf{Y}}, \mathbf{Z}), \\ \mathbf{x}(\tau + \theta) & \text{ by } \mathbf{X} = \mathbf{Q}_{\mathbf{x}}(\theta/2, \hat{\mathbf{X}}, \mathbf{Y}, \mathbf{Z}). \end{aligned}$$

### 3 Reflexive Updating Formulas for Matrix Differential Riccati Equations

In this section, we shall explain our unconventional treatments for numerically solving MDREs using the general methodology of the previous section. Several 2nd order reflexive updating formulas will be constructed in the decreasing order of numerical complexities by the numbers of flops<sup>3</sup> each updating formula requires to complete one time-step in the generic case. By “*the generic case*”, we mean

$$\begin{aligned} & \text{All } A_{ij}(t) \text{'s are real functions of } t \text{ and are dense in the sense most} \\ & \text{of their entries are not identically zero functions. No special} \\ & \text{structures of } A_{ij} \text{ that may be used to reduce computational costs} \\ & \text{are known.} \end{aligned} \tag{3.1}$$

We point out that in applications, especially those from the control theory, special structures of  $A_{ij}$  do exist and known and can be exploited to save work (even quite substantially). We hope our complexity calculation for the generic case at least would give some idea about how expensive each updating formula might be. We shall exclude the costs for evaluating  $A_{ij}(t)$ 's in our flop counts since they are the same for each of our updating formulas.

We remark that the 2nd order convergences of these updating formulas play little restrictions on their practical applicability where very highly accurate numerical results are needed since as we argued the reflexiveness allow us to obtain highly accurate numerical solutions by either palindromic compositions or extrapolations.

#### 3.1 Symmetrical Splitting

The right-hand side of (1.1) is quadratic in  $X$ , and thus the idea of symmetric splitting [24, 25, 35] can be naturally applied to yield a *Reflexive Numerical Method of Order 2*

---

<sup>3</sup>A *flop* is defined to be the amount of work of a floating point operation [18, p.18]. One addition or multiplication of two real numbers is counted as 1 flop. A division typically costs as much as 5 additions or multiplications on most commercially significant machines, but since in our formulas the number of additions or multiplications dominates that of division, it does not matter how many flops a division may cost as far as numerical complexities of the entire updating formulas are concerned.

which requires solving one algebraic linear system – Sylvester Equation in this case – per time step.

Let  $X$  be an approximation to  $X(\tau)$  at time  $t = \tau$ . The next approximation  $\hat{X}$  to  $X(\tau + \theta)$  is obtained by solving

$$\frac{\hat{X} - X}{\theta} = A_{21} - \frac{X + \hat{X}}{2}A_{11} + A_{22}\frac{X + \hat{X}}{2} - \frac{1}{2}(XA_{12}\hat{X} + \hat{X}A_{12}X), \quad (3.2)$$

where we abused notation by still writing  $A_{ij} \equiv A_{ij}(t)|_{t=\tau+\theta/2}$  for short. Notice that this equation is invariant under substitution:  $X \leftarrow \hat{X}$ ,  $\hat{X} \leftarrow X$ ,  $\theta \leftarrow -\theta$ . (3.2) is equivalent to

$$\begin{aligned} (\hat{X} - X)\left(\frac{1}{\theta}I_m + A_{11} + A_{12}X\right) &+ \left(\frac{1}{\theta}I_n - A_{22} + XA_{12}\right)(\hat{X} - X) \\ &= 2(A_{21} - XA_{11} + A_{22}X - XA_{12}X). \end{aligned} \quad (3.3)$$

This equation can be solved by either Bartels-Stewart (B-S) algorithm [5] or Golub-Nash-Van Loan (G-N-V) algorithm [19]. The new updating formula is summarized in Figure 1. In the case of Symmetric Matrix Differential Riccati Equation, we have

**Updating Formula SS:**

**Input:**  $\tau, \theta, X \approx X(\tau)$ .

**Output:**  $\hat{X} \approx X(\tau + \theta)$ .

1 Evaluate  $A_{ij} \leftarrow A_{ij}(\tau + \theta/2)$ ;

2 Solve (3.3) for  $\hat{X}$ .

Figure 1: An updating formula based on the symmetrical splitting.

$$\begin{aligned} \left(\frac{1}{\theta}I_n - A_{22} + XA_{12}\right) &= \left(\frac{1}{\theta}I_n + A_{11} + A_{12}X\right)^*, \\ A_{21} - XA_{11} + A_{22}X - XA_{12}X &= (A_{21} - XA_{11} + A_{22}X - XA_{12}X)^*, \end{aligned}$$

provided  $X^* = X$ ; and thus  $\hat{X}^* = \hat{X}$ .

We shall discuss the cost of the updating formula. Forming the Sylvester equation (3.3) costs  $4mn(m+n) + 4mn \min\{m, n\}$  flops, depending on how  $XA_{12}X$  is computed; real Schur decompositions of a  $m \times m$  and a  $n \times n$  matrices cost approximately  $25(m^3 + n^3)$  flops [18, p.359] required by Bartels-Stewart algorithm and then afterwards solving the Sylvester equation to get  $\hat{X} - X$  costs  $5mn(m+n)$  flops. So the total cost is

$$25(m^3 + n^3) + 9mn(m+n) + 4mn \min\{m, n\} \text{ flops,}$$

which is  $72n^3$  when  $m = n$ . If however, Golub-Nash-Van Loan algorithm is used to solve the Sylvester equation, the total cost can be reduced to

$$\min\{25m^3 + 14n^3/3, 25n^3 + 14m^3/3\} + 9mn(m+n) + 4mn \min\{m, n\} \text{ flops,}$$

which is  $155n^3/3 \approx 51.6n^3$  when  $m = n$ . For the Symmetric Matrix Differential Riccati Equation, where  $m = n$ , potential saving is possible; it can be calculated that the cost is  $36n^3$  flops about half the cost for non-symmetric case.

### 3.2 A Pseudo-Partitioning Technique

Here we first actually go in the other direction – by duplicating the system (1.1) to get

$$X' = A_{21} - XA_{11} + A_{22}Y - XA_{12}Y \stackrel{\text{def}}{=} F(X, Y), X(0) = X_0, \quad (3.4)$$

$$Y' = A_{21} - XA_{11} + A_{22}Y - XA_{12}Y, \quad Y(0) = X_0. \quad (3.5)$$

In theory, one of (3.4) and (3.5) is redundant since  $X(t) \equiv Y(t)$  for the time interval on which (1.1) has a solution, but numerically more tricks can be applied by considering both altogether. Notice that both (3.4) with frozen  $Y$  and (3.5) with frozen  $X$  are linear differential systems, and thus cheaper reflexive updating formulas can be constructed for each of them individually first and then we can concatenate the reflexive updating formulas so obtained to construct reflexive updating formulas for the whole system as explained in §2. It turns out that the resulting reflexive updating formulas for the whole system are actually cheaper than what we derived via the symmetrical splitting idea.

To illustrate our idea, we show how two reflexive updating formulas for integrating (3.4) with frozen  $Y$  from  $\tau$  to  $\tau + \theta$  may be constructed. The idea is as follows.

Evaluate  $A_{ij} \equiv A_{ij}(t)|_{t=\tau+\theta/2}$ , and compute

$$-V = A_{11} + A_{12}Y, \quad W = A_{21} + A_{22}Y,$$

where  $Y$  is given and frozen. Let  $X \approx X(\tau)$  be given, and let  $\hat{X} \approx X(\tau + \theta)$  that is to be computed.

- **Method 1** – the implicit mid-point rule. Solve

$$\frac{\hat{X} - X}{\theta} = \frac{\hat{X} + X}{2}V + W \quad \text{for } \hat{X},$$

i.e.  $(\hat{X} - X)[(2/\theta)I_m - V] = 2F(X, Y)$  requiring solving a linear matrix system of the same size as the MDRE.

- **Method 2**. Write  $V = L + U$ , where  $L$  is lower triangular and  $U$  is upper triangular<sup>4</sup>. Then solve

$$\frac{X_{1/2} - X}{\theta/2} = X_{1/2}L + XU + W \quad \text{for } X_{1/2}, \text{ and}$$

$$\frac{\hat{X} - X_{1/2}}{\theta/2} = X_{1/2}L + \hat{X}U + W \quad \text{for } \hat{X}.$$

We have  $(X_{1/2} - X)[(2/\theta)I_m - L]^{-1} = F(X, Y)$  and  $(\hat{X} - X_{1/2})[(2/\theta)I_m - U]^{-1} = F(X_{1/2}, Y)$  requiring solving two linear triangular matrix systems of the same size as the MDRE.

---

<sup>4</sup>There are many ways in choosing the diagonal entries of  $L$  and  $U$ . For our purpose, we should not made these diagonal entries too large in magnitudes as comparing to these of  $V$ . Such a decomposition is always implied in Figure 3.

Each of the method does not even require solving Sylvester Equations as all conventional implicit ODE solvers and **Updating Formula SS** do. Similar ideas can be applied to the subsystem (3.5). To save space, we shall omit the detail.

Now readily available are reflexive updating formulas for the whole system (3.4) and (3.5) by concatenating formulas for (3.4) and formulas for (3.5) as explained in §2. One disadvantage for doing so is that  $A_{ij}(t)$ 's have to be evaluated four times per time step at  $t - \tau = \theta/4, \theta/2, 3\theta/4$ . This may be too costly. Fortunately, a trick can be applied such that  $A_{ij}(t)$ 's are evaluated only once per time step; that is to freeze all  $A_{ij}(t)$ 's at  $t = \tau + \theta/2$  at the beginning of every time step from  $\tau$  to  $\tau + \theta$  for the whole system, and then treat both (3.4) and (3.5) as constant systems for the time interval. In concatenating updating formulas for (3.4) and (3.5), we did not find any particular reason to treat different subsystems differently. In what follows, we shall summarize two reflexive updating formulas for the whole system into Figures 2 and 3. Bearing in mind that Gaussian elimination for solving an  $m$ -dimensional linear algebraic system costs  $(2/3)m^3$ , we estimate the cost for the reflexive updating formula defined in Figure 2 to be

$$4m^3/3 + 2n^3/3 + 14m^2n + 10mn^2 \text{ flops,}$$

which is  $26n^3$  when  $m = n$ . The cost for the reflexive updating formula defined in Figure 3 is

$$14m^2n + 10mn^2 \text{ flops,}$$

which is  $24n^3$  when  $m = n$ .

**Updating Formula PPM:**

**Input:**  $\tau, \theta, X \approx X(\tau) \approx Y$ .

**Output:**  $\hat{X} \approx X(\tau + \theta) \approx \hat{Y}$ .

- 1 Evaluate  $A_{ij} \leftarrow A_{ij}(\tau + \theta/2)$ ;
- 2 Solve  $(X_{1/2} - X)[(4/\theta)I_m + (A_{11} + A_{12}Y)] = 2F(X, Y)$  for  $X_{1/2}$ ;
- 3 Solve  $[(2/\theta)I_n - (A_{22} - X_{1/2}A_{12})](\hat{Y} - Y) = 2F(X_{1/2}, Y)$  for  $\hat{Y}$ ;
- 4 Solve  $(\hat{X} - X_{1/2})[(4/\theta)I_m + (A_{11} + A_{12}\hat{Y})] = 2F(X_{1/2}, \hat{Y})$  for  $\hat{X}$ .

Figure 2: An updating formula based on the partitioning idea.

Notice that  $\hat{X}$  is generally not equal to  $\hat{Y}$ , but the two differ by an order of  $O(\theta^2)$  for all time  $t$ , which can be exploited to measure how accurate the computed solutions might be. Both reflexive updating formulas defined by Figures 2 and 3 are of 2nd order convergence and as we stated in §§1 and 2, higher order approximations can be obtained easily and efficiently by palindromic compositions (2.2) or extrapolations. In cases when memory is a limitation, it is tempting not to keep both  $\hat{X}$  and  $\hat{Y}$  after each time-step. Doing so provides no harm to 2nd order approximations alone, but in order not to destroy *reflexiveness* so that palindromic compositions (2.2) or extrapolations can be applied to enhance numerical efficiencies, care must be taken in this regard, namely, both  $\hat{X}$  and  $\hat{Y}$  must be kept before the completion of all inner-stages of a palindromic composition (2.2);

### Updating Formula PPR:

**Input:**  $\tau, \theta, X \approx X(\tau) \approx Y$ .

**Output:**  $\hat{X} \approx X(\tau + \theta) \approx \hat{Y}$ .

- 1  $A_{ij} \leftarrow A_{ij}(\tau + \theta/2)$ ;
- 2 a)  $-V \leftarrow A_{11} + A_{12}Y$ ;  
 b) Decompose  $V = L + U$ ;  
 c) Solve  $(X_{1/4} - X)[(4/\theta)I - L] = F(X, Y)$  for  $X_{1/4}$ ;  
 d) Solve  $(X_{1/2} - X_{1/4})[(4/\theta)I - U] = F(X_{1/4}, Y)$  for  $X_{1/2}$ ;
- 3 a)  $P \leftarrow A_{22} - X_{1/2}A_{12}$ ;  
 b) Decompose  $P = L_1 + U_1$ ;  
 c) Solve  $[(2/\theta)I - L_1](Y_{1/2} - Y) = F(X_{1/2}, Y)$  for  $Y_{1/2}$ ;  
 d) Solve  $[(2/\theta)I - U_1](\hat{Y} - Y_{1/2}) = F(X_{1/2}, Y_{1/2})$  for  $\hat{Y}$ ;
- 4 a)  $-\hat{V} \leftarrow A_{11} + A_{12}\hat{Y}$ ;  
 b) Decompose  $\hat{V} = \hat{L} + \hat{U}$ ;  
 c) Solve  $(X_{3/4} - X_{1/2})[(4/\theta)I - \hat{L}] = F(X_{1/2}, \hat{Y})$  for  $X_{3/4}$ ;  
 d) Solve  $(\hat{X} - X_{3/4})[(4/\theta)I - \hat{U}] = F(X_{3/4}, \hat{Y})$  for  $\hat{X}$ .

Figure 3: An updating formula based on the partitioning idea.

both  $\hat{X}$  and  $\hat{Y}$  must be kept until after integration process over entire integration interval is completed so that some extrapolation processes can be applied.

We notice in passing that **Updating Formula PPR** does not do matrix decompositions (like LU or eigenvalue decompositions) that could possibly mess up structures of involved matrices, except those matrix operations that are already presented in a MDRE. So we expect the formula would probably be a good solver to large sparse MDREs. We shall investigate the issue more carefully in the future.

In Figures<sup>6</sup> 2 and 3 concatenations are in the pattern **YX**. But the order **YXY** would probably do equally good.

### 3.3 A Linear System Approach

This approach is based on Bernoulli substitutions of the form  $X = ZY^{-1}$  that transform MDRE into a  $(m + n)$ -dimensional linear system of first order ODEs; see Reid [38].

**Theorem 3.1** *Consider Matrix Differential Riccati Equation (1.1) with  $X(0) = Z_0Y_0^{-1}$ , where  $Y_0$  is  $m \times m$  and  $Z_0$  is  $n \times m$ , and*

$$\frac{d}{dt} \begin{pmatrix} Y \\ Z \end{pmatrix} = \begin{matrix} m & n \\ A_{11} & A_{12} \\ n & A_{21} & A_{22} \end{matrix} \begin{matrix} m \\ Y \\ Z \end{matrix}, \quad \begin{pmatrix} Y \\ Z \end{pmatrix} \Big|_{t=0} = \begin{pmatrix} Y_0 \\ Z_0 \end{pmatrix}. \quad (3.6)$$

---

<sup>6</sup>PPM stands for *pseudo-partitioning and mid-point rule*. The formula defined by Figure 3 resembles an ancient idea of relaxation method for linear algebraic systems. For this reason, we associate the formula with **PPR** — *pseudo-partitioning and Relaxation*.

**Updating Formula LA:**

**Input:**  $\tau$ ,  $\theta$ , and  $Y \approx Y(\tau)$  and  $Z \approx Z(\tau)$  such that  $ZY^{-1} \approx X(\tau)$ .

**Output:**  $\hat{Y} \approx Y(\tau + \theta)$  and  $\hat{Z} \approx Z(\tau + \theta)$  to approximate  $X(\tau + \theta)$  by  $\hat{Z}\hat{Y}^{-1}$ .

1 Evaluate  $A_{ij} \leftarrow A_{ij}(\tau + \theta/2)$ ;

2 a) Decompose  $A_{11} = L_1 + U_1$  and  $A_{22} = L_2 + U_2$ ;

b) Solve  $\frac{\begin{pmatrix} Y_{1/2} \\ Z_{1/2} \end{pmatrix} - \begin{pmatrix} Y \\ Z \end{pmatrix}}{\theta/2} = \begin{pmatrix} L_1 & \\ A_{21} & L_2 \end{pmatrix} \begin{pmatrix} Y_{1/2} \\ Z_{1/2} \end{pmatrix} + \begin{pmatrix} U_1 & A_{12} \\ & U_2 \end{pmatrix} \begin{pmatrix} Y \\ Z \end{pmatrix}$   
for  $Y_{1/2}$  and  $Z_{1/2}$ .

c) Solve  $\frac{\begin{pmatrix} \hat{Y} \\ \hat{Z} \end{pmatrix} - \begin{pmatrix} Y_{1/2} \\ Z_{1/2} \end{pmatrix}}{\theta/2} = \begin{pmatrix} L_1 & \\ A_{21} & L_2 \end{pmatrix} \begin{pmatrix} Y_{1/2} \\ Z_{1/2} \end{pmatrix} + \begin{pmatrix} U_1 & A_{12} \\ & U_2 \end{pmatrix} \begin{pmatrix} \hat{Y} \\ \hat{Z} \end{pmatrix}$   
for  $\hat{Y}$  and  $\hat{Z}$ ;

3<sup>a</sup> QR decomposition  $\hat{Y} = QR$ , and  $\hat{Z} \leftarrow \hat{Z}R^{-1}$ ,  $\hat{Y} \leftarrow Q$ .

---

<sup>a</sup>This step is for stabilization consideration in case when  $Y(t)$  may become increasingly nearly singular. (It does not cure the problem completely though, since there are MDREs for which this ill-conditioning problem is intrinsic.) The step must be used with care in order not to destroy the reflexiveness of the formula defined by Steps 1 and 2 only.

Figure 4: An updating formula based on the linear system approach.

The solution to (1.1) with  $X(0) = Z_0Y_0^{-1}$  and that to (3.6) exist for the same time interval and  $X(t) = Z(t)Y(t)^{-1}$ .

This theorem makes it possible that MDRE (1.1) may be solved through solving the linear system (3.6) by any suited numerical method<sup>7</sup>. However the system (3.6) is notoriously dichotomic [2] and thus *such an approach does often suffer a drawback, namely, the ill-conditioning in inverting  $Y(t)$* , especially for stiff systems when the exact solution converges to a stationary point and thus all  $Y(t)$ 's columns converge to a same constant vector which makes  $Y(t)$  close to a singular matrix. Nevertheless for some MDREs such an approach may work just fine. In practice, at the end of each time-step some kinds of stabilizations may be applied; see Step 3 in Figure 4 which presents a cheap and easily implementable 2nd order (reflexive) updating formula for (3.6), besides the known implicit mid-point rule. This updating formula without Step 3 costs

$$4m(m+n)^2 \text{ flops}$$

which is  $16n^3$  flops when  $m = n$ , and with Step 3 costs

$$4m(m+n)^2 + nm^2 + 8m^3/3 \text{ flops}$$

which is  $19.7n^3$  flops when  $m = n$ .

---

<sup>7</sup>Recently Dieci and Eirola [11, 12] showed that symplectic Runge-Kutta methods preserve positive definiteness and monotonicity.

## 4 Numerical Experiments

Three numerical examples will be presented to illustrate the effectiveness of our proposed methods, as well as their palindromic composition schemes (2.2). We shall use `sIodeJx` to denote an  $I$ -stage (i.e., the number of calls to  $\mathbf{Q}$ ) order  $J$  scheme. ( $\mathbf{x}$  distinguishes different schemes having same number of stages and orders.) For constants that define the schemes, see [25]. Before we detail our numerical examples, let's first discuss some of the features that the proposed formulas have.

### 4.1 Software design

The proposed updating formulas share a common and appealing feature that their computationally intensive parts, except evaluating  $A_{ij}(t)$ , have already been handled by the publically available Linear Algebra Package – LAPACK [1] efficiently written by experts in the field and (highly optimized) Basic Linear Algebra Subprograms – BLAS [16, 15, 33] which nowadays many computer vendors have implemented or will do. This feature allows one to quickly turn the formulas into correct and highly efficient subroutines.

Now we come to each individual updating formula. The following table details their flop counts for large  $m \approx n$ , assuming the MDRE is nonsymmetric, dense, and generic.

Updating Formula	SS (B-S)	SS (G-N-V)	PPM	PPR	LA
flop counts	$72n^3$	$51.6n^3$	$26n^3$	$24n^3$	$19.7n^3$

It is now well-known that fully utilizing level-3 BLAS operations, i.e., matrix-matrix operations, in designing numerical software is the key to speed even though it means more floating point operations sometimes. **Updating Formulas PPR**, and **LA** can be implemented entirely by calling level-3 BLAS operations. We expect much speed could be gained from this. Besides from level-3 BLAS operations, **Updating Formula PPM** also requires two LU decompositions per call. **Updating Formula SS** requires the most flop counts. It must call an eigen-decomposition subroutine like `xGEESX` from LAPACK too, which is not a BLAS operation. So we expect one call to **Updating Formula SS** would be much slower than one call to any other formulas. But on the other hand, preliminary numerical experiments indicate that with the same step-size  $\theta$ , one step of **Updating Formula SS** may be (much) more accurate than the other formulas; so there is a trade-off here. Which formula is better is going to be determined by a particular MDRE.

Once a subroutine for  $\mathbf{Q}$  is written, higher order approximation schemes are readily at hands by palindromic compositions (2.2), and again coding here is very simple.

General speaking reflexive formulas and their palindromic compositions are very much suited for fixed step-size implementations only because of lacking a cheap error estimation mechanism for automatic step-size selections. A typical remedy is to run the formula or the composition three times: one with step-size  $\theta$  and the other two with  $\theta/2$  for each time-step. Such an implementation sometimes is too costly to do. The proposed updating formulas via pseudo-partitioning, however, actually produce two different approximations  $X$  and  $Y$  which are supposed to approximate the same number numerically. This provides an instant and free error estimations. The only possible problem with this is that both

approximations  $X$  and  $Y$  are of the same order of accuracy. This idea has been coded and tested and it turns out to be very good, as we shall see later.

## 4.2 Stability Consideration

MDRE (1.1) is said to be *stiff* if its vectorized system is stiff [7, 10]. Stiff systems are recommended not to be solved by explicit methods but implicit ones. Typically BDF formulas [22] which require solving nonlinear systems at each time-step and thus could be very costly have to be employed.

The proposed **Updating Formula SS** is provably  $A$ -Stable. (This is because the vectorized MDRE falls into the category of the differential systems studied in Kahan and Li [26].) Thus it can be used to solve stiff MDRE and yet requires solving no nonlinear systems. **Updating Formula PPM** appears to be very stable too since typically MDRE (1.1) is stiff when  $A_{11} + A_{12}X$  and  $-A_{22} + XA_{12}$  have eigenvalues all with negative real parts some of which are large in magnitude relatively to the others. It is clear that in such cases, **Updating Formula PPM** is capable of taking large step-sizes. Other updating formulas are obtained by breaking coefficient matrices into upper and lower triangular parts which makes it difficult to analyze their eigenvalue properties; so it is not clear that how well these formulas handle stiff MDRE, but numerical experiments indicate that they are capable of handling at least mild stiff systems.

Composition schemes should be used with care when applied to stiff systems. Generally they are not stable unless  $\mathbf{Q}$  is cleverly designed [35]. Fortunately, often in practice 2nd methods is more cost-effective.

## 4.3 Numerical Examples

### 4.3.1 A Scalar Riccati Equation

This example is taken from Kahan [24]. Consider the following scalar *Riccati Equation*:

$$x' = t + x^2 \quad \text{for all } t \geq 0, x(0) = 0. \quad (4.7)$$

We shall compute  $x(10)$ . We shall pretend not to know that the solution may be expressed in terms of Bessel functions:

$$\begin{aligned} x(t) &= -\frac{d}{dt} \ln \left( \sqrt{t} J_{-1/3}(2t^{3/2}/3) \right) = \sqrt{t} \frac{J_{2/3}(2t^{3/2}/3)}{J_{-1/3}(2t^{3/2}/3)}, \\ x(10) &= -7.53121107313542534544973495802223 \dots \end{aligned}$$

See Figure 5. Instead we shall apply a numerical method that purports to solve the differential equation approximately and yet as accurately as we like if we spend enough time on it. We shall use<sup>8</sup>  $Q(\theta, \tau, g)$  to stand for an updating formula that advances  $g \approx x(\tau)$  to  $Q(\theta, \tau, g) \approx x(\tau + \theta)$ . A few unconventional updating formulas have been tested. Based

---

<sup>8</sup>We now use slightly different notation to accommodate that  $Q$  is a scalar and depends explicitly on  $t$ , which can be suppressed by transforming (4.7) into an AIVP of dimension 2; but we prefer not to do so to stick to a scalar equation.

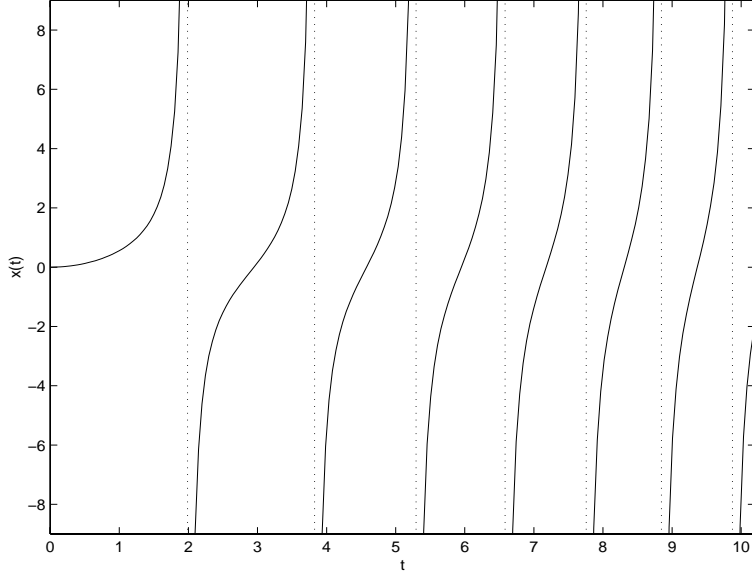


Figure 5: The solution to Riccati Equation  $x' = t + x^2$  with  $x(0) = 0$ .

on Figure 1, we have  $Q_{SS}$  and its variation<sup>9</sup>  $Q_T$  due to Kahan [24]:

$$Q_{SS}(\theta, \tau, g) \stackrel{\text{def}}{=} g + \frac{\tau + \theta/2 + g^2}{1/\theta - g} \quad \text{and} \quad Q_T(\theta, \tau, g) \stackrel{\text{def}}{=} g + \frac{\tau + \theta/2 + g^2}{\frac{\sqrt{\tau + \theta/2}}{\tan(\theta\sqrt{\tau + \theta/2})} - g};$$

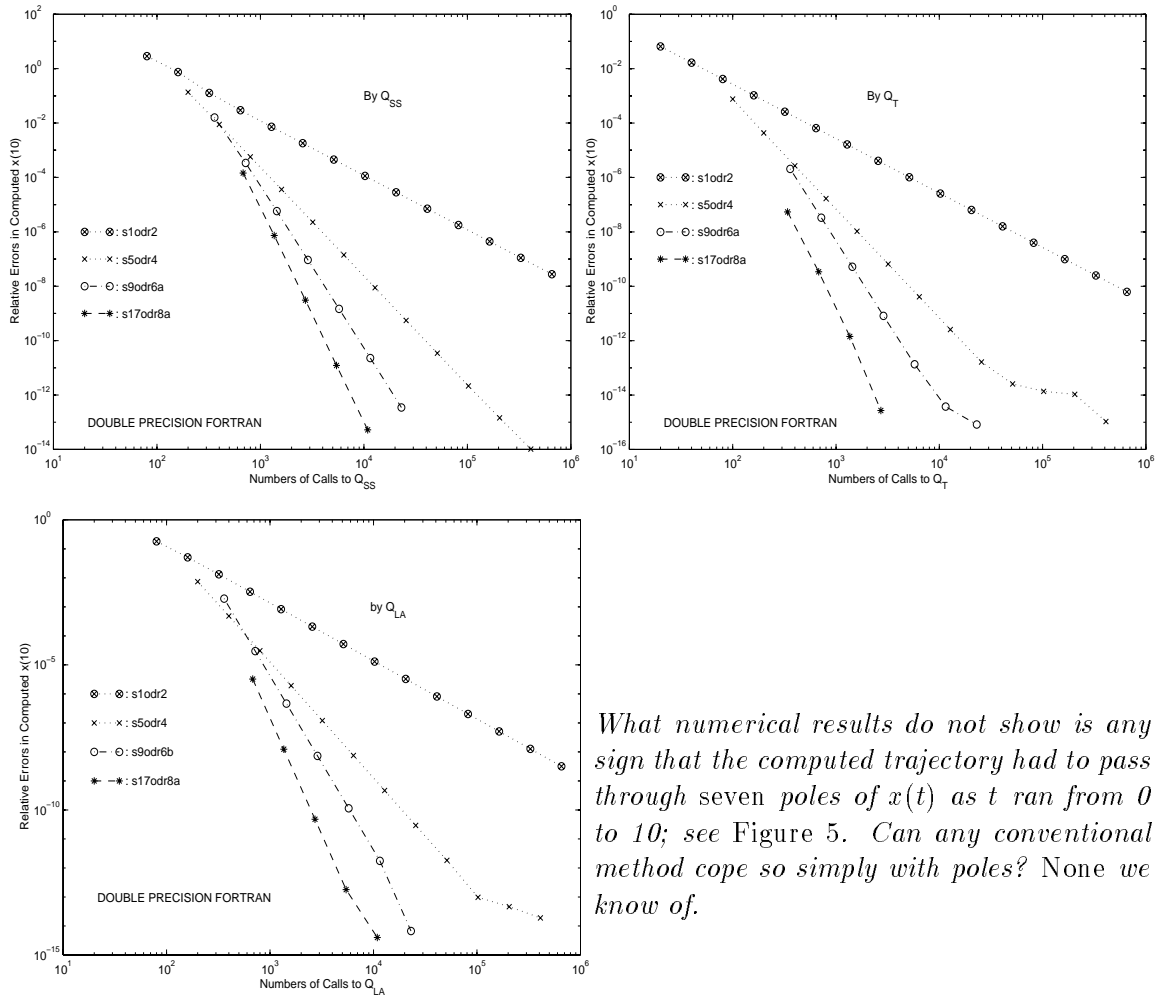
Figure 4 without Step 3 gives  $Q_{LA}(\theta, \tau, [y, z]) \stackrel{\text{def}}{=} [\hat{y}, \hat{z}]$ :

$$y_{1/2} = y - (\theta/2)z, \quad \hat{z} = z + \theta(\tau + \theta/2)y_{1/2}, \quad \hat{y} = y_{1/2} - (\theta/2)\hat{z}.$$

Figure 6 shows the behaviors of our palindromic schemes based on above mentioned 2nd order reflexive updating formulas. Composition schemes up to order 10 work perfectly well and clearly indicate their orders of convergence, but the order 10 schemes are not competitive in this example and thus are not shown in Figure 6. Also tested are **Updating Formulas PPM** and **PPR** defined by Figures 2 and 3. It turns out they only work fine before integrations near the first pole  $1.9863527 \dots$  of the exact solution, but they behave badly near and in attempting to cross the pole as other formulas do.

---

<sup>9</sup> $Q_T$  is obtained from only semi-discretizing (4.7) as  $x' = (\tau + \tau + \theta)/2 + x^2$ ,  $x(\tau) = g$  and then solving it exactly.



*What numerical results do not show is any sign that the computed trajectory had to pass through seven poles of  $x(t)$  as  $t$  ran from 0 to 10; see Figure 5. Can any conventional method cope so simply with poles? None we know of.*

Figure 6: Relative errors in computed  $x(10)$  .vs. the numbers of calls to  $Q_{SS}$  (left) or  $Q_T$  (right). (Fixed step-size implementation)

### 4.3.2 A Missile Autopilot Problem

This problem is taken from Kenney and Leipnik [27] and has a physical significance – modeling a missile autopilot problem. It is a symmetric MDRE:

$$P' = -CC^T - PB - B^T P + PGG^T P, \quad P(0) = 0,$$

where  $C = I_4$ ,  $G = (0 \ 0 \ 127 \ 0)^*$ , and

$$B = \begin{pmatrix} 0 & -\frac{5t^2+13t+15}{175-17t} & -\frac{52t^2+126t+150}{175-17t} & 0 \\ 1 & -\frac{21t^2+50t+61}{-754t^2+9880t+13000} & -\frac{52t^2+126t+150}{-754t^2+9880t+13000} & 0 \\ 0 & 0 & -127 & 0 \\ 0 & -\frac{21t^2+50t+61}{600-42t} & \frac{5t^2+13t+15}{600-42t} & 0 \end{pmatrix}.$$

We shall integrate the MDRE from  $t = 0$  up to  $t = 1$ . The exact solution<sup>10</sup> at  $t = 1$  (correct up to 17 decimal digits) is (notice that the matrix is symmetric)

$(i, j)$ entry	up to 17 decimal digits
(1,1)	-0.76158129792536009D+0
(2,1)	0.13592872478918398D+0
(3,1)	-0.86915180259185969D-2
(4,1)	0.10766667962629911D-1
(2,2)	-0.92255002992371799D+0
(3,2)	0.15162471148073925D-2
(4,2)	-0.87573365492494723D-1
(3,3)	-0.19109215612078873D-1
(4,3)	0.44250780807439288D-3
(4,4)	-0.99936462480739237D+0

Figure 7 shows the numerical behavior of **Updating Formulas SS**, **PPM**, and **PPR** and their composition schemes for their fixed step-size implementations. All computations are carried out in MATLAB. In this particular example, as far as flop counts are concerned, a call to **Updating Formula PPM** costs about 65% of that to **Updating Formula SS**, and a call to **Updating Formula PPR** about 75%. Despite all these, **Updating Formula SS** and its palindromic composition schemes are the most efficient. **Updating Formula LA** and its palindromic composition schemes do not work for this problem due to the ill-conditioning in inverting the  $R$  (see Step 3 in Figure 4).

We have also done variable step-size implementations for **Updating Formulas PPM** and **PPR** and their compositions based on the error estimation mechanism described previously. The actual implementations go as follows. Let **rto1** and **ato1** be prescribed relative and absolute tolerances. (We let **ato1** = **rto1**<sup>2</sup>.) At the beginning of time step  $t = \tau$ , we have approximation  $X$ . To start, we let  $Y = X$ , and then one or many (for a composition scheme) calls to one of the updating formulas are made; at the end we have

<sup>10</sup>It is computed using FORTRAN REAL \*16.

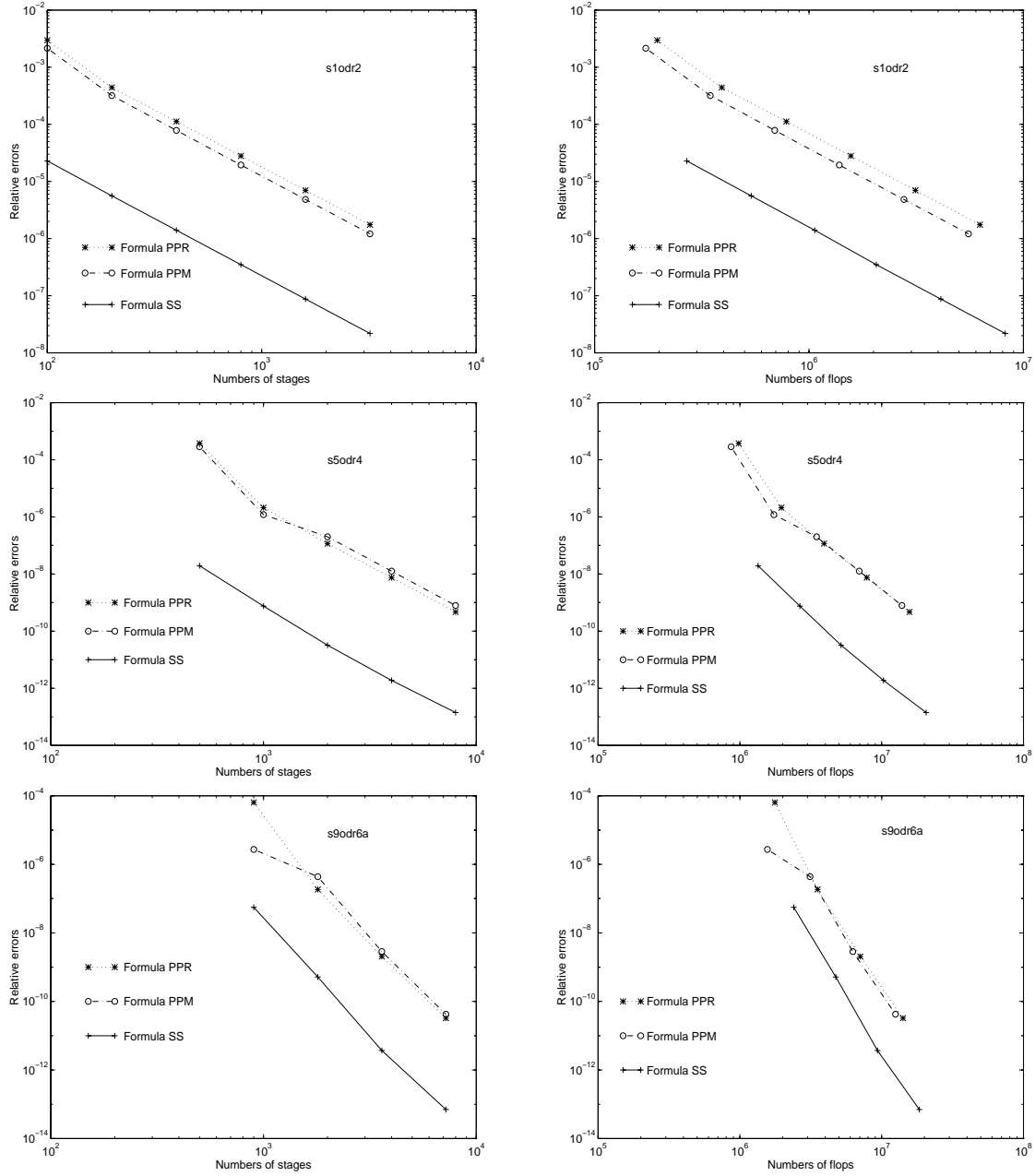


Figure 7: Relative errors  $\frac{\|(\text{Computed})X(1) - (\text{exact})X(1)\|_F}{\|(\text{exact})X(1)\|_F}$  vs. the numbers of stages (calls to an updating formula) or the numbers of flops obtained by MATLAB's built-in flop counter. (Fixed step-size implementation)

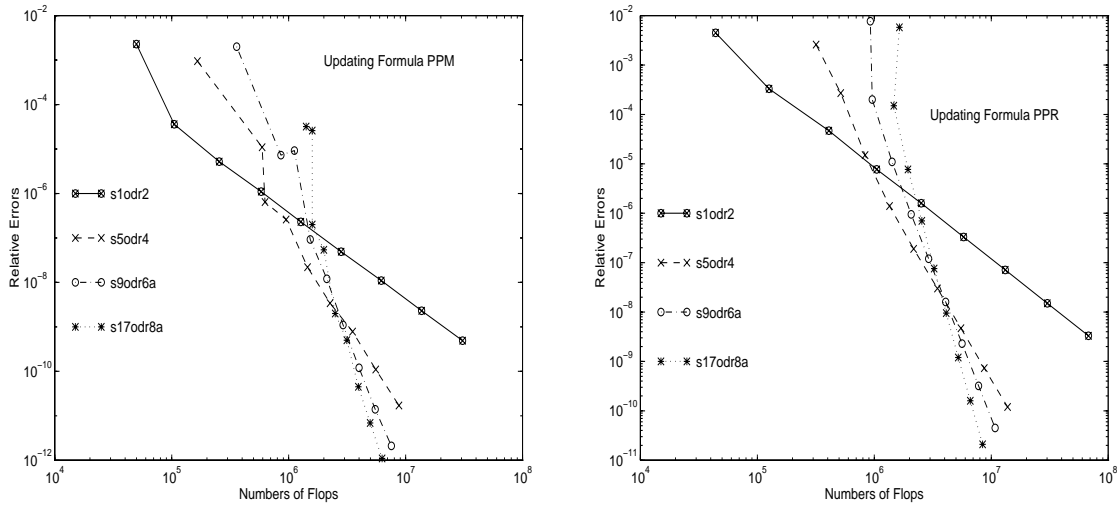


Figure 8: Relative errors  $\frac{\|(\text{Computed})X(1) - (\text{exact})X(1)\|_F}{\|(\text{exact})X(1)\|_F}$  vs. the numbers of flops obtained by MATLAB's built-in flop counter. (Variable step-size implementation)

$\hat{X}$  and  $\hat{Y}$  both approximate the true one at the next time step  $t = \tau + \theta$ . In MATLAB notation, we then do

```
Xtmp = 0.5 * ( $\hat{X}$  +  $\hat{Y}$ );
err = abs( $\hat{X}$  -  $\hat{Y}$ )./(rtol * abs(Xtmp) + atol * ones(4, 4));
err = max(max(err));
```

Now if<sup>11</sup>  $\text{err} \leq 4$ , we accept  $\text{Xtmp}$  as the computed approximation at time  $\tau + \theta$  and estimate a new step-size  $\theta$  and proceed; otherwise reject the just computed  $\text{Xtmp}$  and replace the current step-size  $\theta$  by

$$\text{theta} = \max(0.1, 0.8/(\text{err}^{1/(\text{order}+1)})) * \text{theta}; \quad (4.8)$$

and redo it again. Upon acceptance, the step-size for the very next time step is calculated either by

$$\text{theta} = \max(0.1, \min(2, 0.9/(\text{err}^{1/(\text{order}+1)}))) * \text{theta};$$

if no rejection in the current and previous time steps, or by (4.8) otherwise.

The following table displays numerical behaviors of our varied step-size implementations for  $\text{rtol} = 1\text{e} - 3$ , where  $\text{ns}$  stands for the numbers of successful integration steps,  $\text{fs}$  for the numbers of rejected steps,  $\text{relerr}$  for the relative errors at  $t = 1$ . We also plot selectively Figure 8 for a few schemes to show typical performances; not all tested schemes are plotted for otherwise the pictures would be too messy to read.

<sup>11</sup>The number 4 is somehow arbitrary. We choose it as the dimension of the problem. A judicial choice would reduce the numbers of rejected step while not hurting accuracy.

rtol = 1e - 3 and atol = rtol <sup>2</sup>					
Scheme	Formula	ns	nf	flops	relerr
s1odr2	<b>PPM</b>	139	0	2.55e+05	5.2e-06
	<b>PPR</b>	196	0	4.09e+05	4.7e-05
s3odr4	<b>PPM</b>	154	0	8.10e+05	1.1e-06
	<b>PPR</b>	181	0	1.08e+06	3.8e-05
s5odr4	<b>PPM</b>	73	0	6.28e+05	6.4e-07
	<b>PPR</b>	85	0	8.35e+05	1.5e-05
s5odr4a	<b>PPM</b>	103	0	8.90e+05	9.5e-07
	<b>PPR</b>	119	0	1.17e+06	2.4e-05
s7odr6	<b>PPM</b>	100	0	1.20e+06	1.3e-06
	<b>PPR</b>	107	0	1.47e+06	2.4e-05
s9odr6a	<b>PPM</b>	73	0	1.12e+06	9.2e-06
	<b>PPR</b>	81	0	1.42e+06	1.1e-05
s9odr6b	<b>PPM</b>	73	1	1.14e+06	9.9e-07
	<b>PPR</b>	81	0	1.42e+06	1.1e-05
s15odr8	<b>PPM</b>	69	0	1.76e+06	5.0e-07
	<b>PPR</b>	75	0	2.19e+06	1.1e-05
s17odr8a	<b>PPM</b>	55	0	1.59e+06	2.0e-07
	<b>PPR</b>	59	0	1.94e+06	7.7e-06
s17odr8b	<b>PPM</b>	55	0	1.59e+06	2.2e-07
	<b>PPR</b>	59	0	1.94e+06	7.8e-06

### 4.3.3 A control problem governed by the heat equation

Discretizing optimal control problems governed by Partial Differential Equations (PDEs) [4, 31, 39] typically yields large MDREs of the form

$$P' = -CC^T - PA - A^T P + PBB^T P, \quad (4.1)$$

where  $B$  and  $C$  have only a few columns, and  $A$  appears in the form of  $HQ^{-1}$ , where  $H$  and  $Q$  are banded with very narrow bandwidths and  $Q$  is symmetric positive definite.

Such MDREs if time-invariant (i.e.,  $C$ ,  $A$ , and  $B$  are constant matrices) may be solved indirectly by integrating the associated Chandrasekhar systems. Doing so won't yield approximations to  $P(t)$  directly, but it may be good enough in certain circumstances. Here, however, we shall consider the situations where either the approximations to  $P(t)$  are indeed needed or Chandrasekhar method is simply not applicable such as for time-varying systems or  $C$  have too many columns that through solving the associated Chandrasekhar system is not much advantageous even though the MDRE is time-invariant. Indeed the example we are going to discuss is in the category where the Chandrasekhar system method may be best suited, but our purpose here is to provide an illustrative example showing the effectiveness of **Updating Formula PPM** on, but not limited to, MDREs with above-mentioned structural properties.

We shall apply solve the MDREs arising from discretizing a control problem governed by the heat equation [4]. The original problem is as follows: minimizing the cost functional



Rewrite (4.6) as

$$\mathbf{z}'(t) = \tilde{A}\mathbf{z}(t) + \tilde{b}u(t), \quad (4.7)$$

where  $\tilde{A} = Q^{-1}H$  and  $\tilde{b} = Q^{-1}b$  (both should be left in their factor forms.).

The optimal solution is given by  $u_{\text{opt}}(t) = -\tilde{b}^T \tilde{P}(t)\mathbf{z}(t)$ , where  $\tilde{P}(t)$  satisfies the following Matrix Differential Riccati equation

$$\tilde{P}' = -\tilde{c}\tilde{c}^T - \tilde{P}\tilde{A} - \tilde{A}^T\tilde{P} + \tilde{P}\tilde{b}\tilde{b}^T\tilde{P},$$

Substitute  $\tilde{A} = Q^{-1}H$  and  $\tilde{b} = Q^{-1}b$ , and pre- and post-multiply by  $Q^{-1}$  to get

$$P' = -cc^T - PA - A^T P + Pbb^T P, \quad P(0) = 0. \quad (4.8)$$

where  $P = Q^{-1}\tilde{P}Q^{-1}$ ,  $A = HQ^{-1}$ , and  $c = Q^{-1}\tilde{c}$ .

MDRE (4.8) shares the following common structural properties with MDREs arising from discretizing control problems governed by PDEs:

1.  $A$  appears in the form  $HQ^{-1}$  with (symmetrically) narrow banded  $Q$  and  $H$  and positive definite  $Q$ . Their bandwidths depend on the choices of the trial functions.
2.  $B$  and  $C$  have only a few columns (just 1 column for (4.8)), and  $B$  has only a few nonzero entries (here  $b = (-1, 0, \dots, 0)^T$ ).

The system (4.8) is normally stable backwards in time and its solution (quickly) goes to the non-negative definite solution that solves the associated Algebraic Riccati Equation  $-cc^T - PA - A^T P + Pbb^T P = 0$ . To the best interest of the control problem, it should be integrated backwards in time. In applying **Updating Formula PPM**, linear matrix systems like

$$(\hat{X} - X)[(2/\theta)I + (A - bb^T Y)] = 2F(X, Y), \quad (4.9)$$

$$[(2/\theta)I + (A - bb^T X)]^T(\hat{Y} - Y) = 2F(X, Y), \quad (4.10)$$

are solved repeatedly, given  $X$  and  $Y$ , where  $F(X, Y) \stackrel{\text{def}}{=} -cc^T - XA - A^T Y + Xbb^T Y$ . Both can be solved at costs of  $O(N^2)$  flops. We shall use (4.9) as an example to show how this can be done. Multiply (4.9) by  $Q$  from the right to get

$$(\hat{X} - X)[(2/\theta)Q + H - bb^T YQ] = 2F(X, Y)Q.$$

Write  $M = (2/\theta)Q + H$  which is a symmetric negative definite tridiagonal matrix for  $\theta < 0$ .  $-M$  has a Cholesky decomposition, and it applied to a vector costs  $O(N)$  flops and to an  $(N + 1) \times (N + 1)$  matrix costs  $O(N^2)$  flops. Notice that

$$[M - bb^T YQ]^{-1} = M^{-1} + \frac{1}{1 - (b^T YQ)M^{-1}b} M^{-1}b(b^T YQ)M^{-1};$$

so  $\hat{X} - X = 2F(X, Y)Q[M - bb^T YQ]^{-1}$  can be computed in  $O(N^2)$  flops.

Consider  $c(x) \equiv 1$  in (4.3). Then the vector  $c$ 's entries are all ones, and the matrix  $P$  again with entries all ones is the critical point to which numerical solutions are supposed to converge to as time goes to  $-\infty$ .

We have run our code implemented in MATLAB for various  $N$ , and  $\theta$  (fixed step-size in time). It turns out that **Updating Formula PPM** is very effective on MDRE (4.8) and enjoys remarkable stability backwards in time, which is due to the fact that from the optimal control perspective eventually  $A - bb^T X$  and  $A - bb^T Y$  have eigenvalues all with negative real parts provided  $X$  and  $Y$  are fairly close to the exact  $P$ . It is fairly fast for modest accuracy requirement. Figure 9 gives a typical picture of the solution curve to (4.8). The capability of allowing large  $|\theta|$  makes the formula particularly attractive

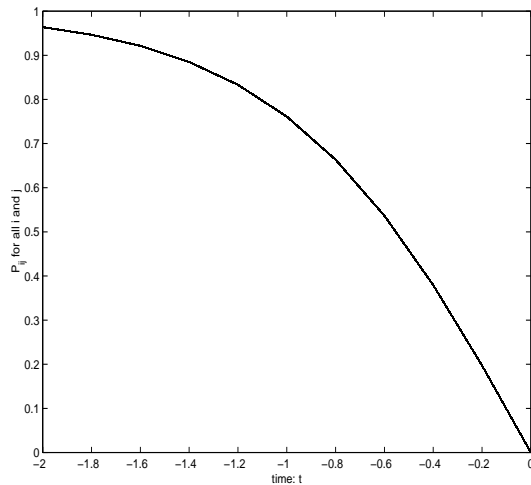


Figure 9: Computed  $P$  with  $\theta = 0.2$  vs. time. Apparently all entries grow at the same rate.

from the control perspective to seeking the steady solution. The following table displays numerical results by one call to **Updating Formula PPM** with  $\theta = -2$  at  $t = 0$ . These results clearly reach a close neighborhood of the critical point. In the table  $F(P) \stackrel{\text{def}}{=} -cc^T - PA - A^T P + Pbb^T P$ ,  $P_X$  and  $P_Y$  are obtained from the  $X$ - and  $Y$ -part numerical approximations by symmetrization (like  $(X + X^T)/2$ ).  $P_{\text{crt}}$  is the critical point to the MDRE, and  $\|\cdot\|_{\max}$  is the largest entry in magnitude.

$N$	$\ F(P_X)\ _{\max}$	$\ P_X - P_{\text{crt}}\ _{\max}$	$\ F(P_Y)\ _{\max}$	$\ P_Y - P_{\text{crt}}\ _{\max}$	flops/step ( $N^2$ )
5	3.89e-12	1.54e-14	4.73e-12	2.09e-14	188.7
10	4.06e-11	4.24e-14	5.87e-11	6.18e-14	155.5
20	6.14e-10	2.22e-13	1.19e-09	3.16e-13	140.3
40	8.85e-09	6.90e-13	2.04e-08	1.59e-12	133.0
80	1.90e-07	3.84e-12	3.01e-07	6.55e-12	129.5
160	3.21e-06	1.42e-11	4.85e-06	2.14e-11	127.7
320	6.90e-05	7.37e-11	1.34e-04	1.39e-10	126.9
640	9.23e-04	2.62e-10	1.73e-03	4.84e-10	126.4

## 5 Conclusions and Future Work

We have presented several unconventional methods for solving Matrix Differential Riccati Equations (MDRE). The methods are easy to use and can be quickly and efficiently implemented on modern computer architectures due to the fact that the methods rely entirely on currently publicly available Linear Algebra Package (LAPACK) and Basic Linear Algebra Subprograms (BLAS). These methods fall into the category of reflexive methods, and thus are at least 2nd order convergence. If necessary, efficient extrapolations or palindromic compositions can be used to raise the orders. We also exploit variable step-size implementation on two updating formulas and their palindromic compositions. Preliminary numerical tests are encouraging. Our future studies will include the following.

1. One of the most expensive parts of **Updating Formula SS** (Figure 1) for a *large* MDRE (1.1) is to compute (real) Schur decompositions of a  $m \times m$  and/or a  $n \times n$  matrices in solving the Sylvester equation (3.3). It could take as much as 57% to 69% of the total cost per time-step depending on the algorithm for solving (3.3). In practice, costs of Schur decompositions may dominate the rest even more because possibly special structures allow one to form the Sylvester equation (3.3) cheaply. On the other hand, computing Schur decompositions of matrices of small sizes are fast. This gives us a reason to consider partitioning the whole (1.1) into small subsystems. Here is an outline. To facilitate our notation, we shall write MDRE (1.1) as

$$X' = Q - XB + CX - XPX \stackrel{\text{def}}{=} F(X), \quad X(0) = X_0. \quad (5.1)$$

In the other word, new notation has been given to  $A_{ij}$  as in

$$\begin{matrix} & m & n \\ m & \left( \begin{array}{cc} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array} \right) & \\ n & & \end{matrix} = \begin{matrix} & m & n \\ m & \left( \begin{array}{cc} B & P \\ Q & C \end{array} \right) & \\ n & & \end{matrix}.$$

We partition this MDRE as follows. Let integers  $m_1, m_2, \dots, m_s > 1$  and  $n_1, n_2, \dots, n_t > 1$  such that  $\sum_{i=1}^s m_i = m$  and  $\sum_{i=1}^t n_i = n$ . (For efficient partitioning, we shall make all  $m_i$ 's almost equal and all  $n_i$ 's almost equal.) Partition  $X$  in the same way as to  $Q$  and conformally other matrices as in

$$\begin{matrix} & m_1 & \cdots & m_s & n_1 & \cdots & n_t \\ m_1 & \left( \begin{array}{cccccc} B_{11} & \cdots & B_{1s} & P_{11} & \cdots & P_{1t} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ B_{s1} & \cdots & B_{ss} & P_{s1} & \cdots & P_{st} \\ Q_{11} & \cdots & Q_{1s} & C_{11} & \cdots & C_{1t} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ Q_{t1} & \cdots & Q_{ts} & C_{t1} & \cdots & C_{tt} \end{array} \right) & \\ \vdots & & & & & & \\ n_t & & & & & & \end{matrix}.$$

MDRE (5.1) is now a collection of the following  $st$  small MDREs:

$$\begin{aligned}
X'_{ij} = & \left( Q_{ij} - \sum_{k \neq j} X_{ik} B_{kj} + \sum_{\ell \neq i} C_{i\ell} X_{\ell j} - \sum_{k \neq j, \ell \neq i} X_{ik} P_{k\ell} X_{\ell j} \right) \\
& - X_{ij} \left( B_{jj} + \sum_{\ell \neq i} P_{j\ell} X_{\ell j} \right) + \left( C_{ii} - \sum_{k \neq j} X_{ik} P_{ki} \right) X_{ij} \\
& + X_{ij} P_{ji} X_{ij}, \text{ for } i = 1, \dots, t, j = 1, \dots, s.
\end{aligned}$$

Each of these small MDREs can be solved rather quickly by **Updating Formula SS** for some matrix variable assuming given constant values for the others:

$$\begin{aligned}
(\hat{X}_{ij} - X_{ij}) \left( \frac{1}{\theta} I + B_{jj} + (PX)_{jj} \right) \\
+ \left( \frac{1}{\theta} I - C_{ii} + (XP)_{ii} \right) (\hat{X}_{ij} - X_{ij}) = 2F(X)_{ij}, \tag{5.2}
\end{aligned}$$

where  $(PX)_{jj}$  is the  $(j, j)$ th block of  $PX$  partitioned in the same way as for  $B$ ,  $(XP)_{ii}$  is the  $(i, i)$ th block of  $XP$  partitioned in the same way as for  $C$ , and  $F(X)_{ij}$  is the  $(i, j)$ th block of  $F(X)$  partitioned in the same way as for  $X$ . A reflexive method for the whole MDRE (5.1) can then be obtained by concatenating programs for each sub-MDREs as follows. (One can easily modify it to take symmetry into consideration when (5.1) is a symmetric MDRE.)

**Algorithm:** One time-step for integrating (5.1)

**Input:** Current time  $\tau$ , step-size  $\theta$ , and current approximation  $X$

**Output:** Approximation  $X$  at time  $t = \tau + \theta$ .

**begin:**

1. If it is time-varying MDRE, evaluate  $Q$ ,  $B$ ,  $C$ , and  $P$  at  $t = \tau + \theta/2$ ;
2. For  $i$  from 1 to  $t$  do
  - For  $j$  from 1 to  $s$  do
    - Integrate the  $(i, j)$ th sub-MDRE by (5.2) but with *step-size*  $\theta/2$ , and overwrite  $X$ 's  $(i, j)$ th block with  $\hat{X}_{ij}$ ;
  - EndFor
- EndFor
3. For  $i$  from  $t$  down to 1 do
  - For  $j$  from  $s$  down to 1 do
    - Integrate the  $(i, j)$ th sub-MDRE by (5.2) but with *step-size*  $\theta/2$ , and overwrite  $X$ 's  $(i, j)$ th block with  $\hat{X}_{ij}$ ;
  - EndFor
- EndFor

**end**

To see roughly how much we may save in the Schur decomposition part, we consider the square case:  $m = n$ ,  $s = t$ , and  $m_i = n_i \approx m/s$ . **Updating Formula SS** requires computing 1 (for Golub-Nash-Van Loan algorithm) or 2 (for Bartels-Stewart algorithm) Schur

decompositions of  $n \times n$  matrices; and the method here requires computing  $2s^2$  (for Golub-Nash-Van Loan algorithm) or  $4s^2$  (for Bartels-Stewart algorithm) Schur decompositions of  $n/s \times n/s$  matrices. Noticing that

$$2s^2(n/s)^3 = (2/s)n^3,$$

we conclude roughly a save in flops by a factor of  $s/2$ .

The above approach is attractive only if it does not introduce substantially more work in evaluating all  $F(X)_{ij}$  in comparing with just evaluating  $F(X)$  once. However for MDREs as discussed in §4.3.3, it does.

**2.** Our variable step-size implementations of **Updating Formulas PPM** and **PPR** and their compositions are based on approximations of the same orders. This may be a risky practice in general ODE integrations. Further studies are needed in order to assess the applicability of this approach in MDRE integrations, although the presented numerical results look fine so far.

**3.** We will address more about integrating large sparse MDREs. Our work in §4.3.3 shows the effectiveness of **Updating Formulas PPM** on that kind of MDRE. It is conceivable that our proposed methods may have to be tailored to suit a particular application.

## Acknowledgment

The author would like to thank Prof. W. Kahan (University of California, Berkeley) for his constant encouragement and many constructive suggestions. He thanks Prof. John Burns and Mr. Kevin Hulsing (Virginia Polytechnic Institute and State University) for the fruitful conversations, Prof. Alan Laub (University of California, Davis), Prof. Charles Kenney (University of California, Santa Barbara), and Prof. Gary Rosen (University of Southern California) for their detailed email correspondences regarding large sparse MDREs arising from control problems governed by PDEs.

## References

- [1] E. ANDERSON, Z. BAI, C. BISCHOF, J. DEMMEL, J. DONGARRA, J. D. CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, S. OSTROUCHOV, AND D. SORENSEN, *LAPACK Users' Guide*, SIAM, Philadelphia, 1995.
- [2] U. M. ASCHER, R. M. MATTHEIJ, AND R. D. RUSSELL, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- [3] I. BABUŠKA AND V. MAJER, *The factorization method for the numerical solution of two point boundary value problems for linear ODE's*, SIAM Journal on Numerical Analysis, 24 (1987), pp. 1301–1334.
- [4] H. T. BANKS AND K. ITO, *A numerical algorithm for optimal feedback gains in high dimensional linear quadratic regulator problems*, SIAM Journal on Control and Optimization, 29 (1991), pp. 499–515.

- [5] R. H. BARTELS AND G. W. STEWART, *Algorithm 432: The solution of the matrix equation  $AX - BX = C$* , Communications of the ACM, 8 (1972), pp. 820–826.
- [6] R. BULIRSCH AND J. STOER, *Numerical treatment of ordinary differential equations by extrapolation methods*, Numerische Mathematik, 8 (1966), pp. 1–13.
- [7] C. H. CHOI AND A. J. LAUB, *Efficient matrix-valued algorithm for solving stiff Riccati differential equations*, IEEE Transactions on Automatic Control, 35 (1990), pp. 770–776.
- [8] E. J. DAVISON AND M. C. MAKI, *The numerical solution of the matrix differential Riccati equation*, IEEE Trans. Automat. Contr., AC-18 (1973), pp. 71–73.
- [9] J. DE FRUTOS AND J. M. SANZ-SERNA, *An easily implementable fourth-order method for the time integration of wave problems*, Journal of Computational Physics, 103 (1992), pp. 160–168.
- [10] L. DIECI, *Numerical integration of the differential Riccati equation and some related issues*, SIAM Journal on Numerical Analysis, 29 (1992), pp. 781–815.
- [11] L. DIECI AND T. EIROLA, *Positive definiteness in the numerical solution of Riccati differential equations*, Numer. Math., 67 (1994), pp. 303–313.
- [12] ———, *Preserving monotonicity in the numerical solution of Riccati differential equations*, Numer. Math., 74 (1996), pp. 35–47.
- [13] L. DIECI, M. R. OSBORNE, AND R. D. RUSSELL, *A Riccati transformation method for solving linear BVPs. I: Theoretical aspects*, SIAM Journal on Numerical Analysis, 25 (1988), pp. 1055–1073.
- [14] ———, *A Riccati transformation method for solving linear BVPs. II: Computational aspects*, SIAM Journal on Numerical Analysis, 25 (1988), pp. 1074–1092.
- [15] J. DONGARRA, J. D. CROZ, I. DUFF, AND S. HAMMARING, *A set of level 3 Basic Linear Algebra Subprograms*, ACM Trans. Math. Software, 16 (1990), pp. 1–17.
- [16] J. DONGARRA, J. D. CROZ, S. HAMMARING, AND R. HANSON, *An extended set of Fortran Basic Linear Algebra Subprograms*, ACM Trans. Math. Software, 14 (1988), pp. 1–17.
- [17] E. FOREST AND R. D. RUTH, *Fourth-order symplectic integrator*, Physica D, 43 (1990), pp. 105–117.
- [18] G. H. GOLUB AND C. F. V. LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 3rd ed., 1996.
- [19] G. H. GOLUB, S. NASH, AND C. VAN LOAN, *Hessenberg–Schur method for the problem  $AX + XB = C$* , IEEE Transactions on Automatic Control, AC-24 (1979), pp. 909–913.
- [20] W. GRAGG, *On extrapolation methods for ordinary initial-value problems*, SIAM Journal on Numerical Analysis, 2 (1965), pp. 384–403.
- [21] E. HAIRER, S. P. NØRSETT, AND G. WANNER, *Solving Ordinary Differential Equations I*, Springer-Verlag, New York, second ed., 1992.
- [22] E. HAIRER AND G. WANNER, *Solving Ordinary Differential Equations II*, Springer-Verlag, New York, 1991.
- [23] W. KAHAN, *Relaxation methods for solving systems of ordinary differential equations*, manuscript, CS Division, Department of EECS, University of California at Berkeley, Oct. 1977.

- [24] ———, *Unconventional numerical methods for trajectory calculations*, Lectures Notes, CS Division, Department of EECS, University of California at Berkeley, Oct. 1993.
- [25] W. KAHAN AND R.-C. LI, *Composition constants for raising the orders of unconventional schemes for ordinary differential equations*, *Mathematics of Computation*, 66 (1997), pp. 1089–1099.
- [26] ———, *Unconventional schemes for a class of ordinary differential equations—with applications to the Korteweg-de Vries*, *J. Comp. Phys.*, 134 (1997), pp. 316–331.
- [27] C. S. KENNEY AND R. B. LEIPNIK, *Numerical integration of the differential matrix Riccati equation*, *IEEE Transactions on Automatic Control*, AC-30 (1985), pp. 962–970.
- [28] H. KWAKERNAAK AND R. SIVAN, *Linear Optimal Control Systems*, Wiley Interscience, New York, 1972.
- [29] D. G. LAINIOTIS, *Generalized Chandrasekhar algorithms: Time-varying models*, *IEEE Trans. Automat. Contr.*, AC-21 (1976), pp. 728–732.
- [30] ———, *Partitioned Riccati solutions and integration-free doubling algorithms*, *IEEE Trans. Automat. Contr.*, AC-21 (1976), pp. 677–689.
- [31] I. LASIECKA AND R. TRIGGIANI, *Differential and Algebra Riccati Equations with Application to Boundary/Point Control Problems: Continuous Theory and Approximation Theory*, *Lecture Notes in Control and Information Sciences*, vol. 164, Springer-Verlag, New York, 1991.
- [32] A. J. LAUB, *Schur techniques for Riccati differential equations*, in *Feedback Control of Linear and Nonlinear Systems*, D. Hinrichsen and A. Isidori, eds., New York, 1982, Springer-Verlag.
- [33] C. LAWSON, R. HANSON, D. KINCAID, AND F. KROGH, *Basic linear algebra subprograms for Fortran usage*, *ACM Trans. Math. Software*, 5 (1979), pp. 308–323.
- [34] R. B. LEIPNIK, *A canonical form and solution for the matrix Riccati differential equation*, *Bull. Australian Math. Soc.*, 26 (1985), pp. 355–361.
- [35] R.-C. LI, *Raising the Orders of Unconventional Schemes for Ordinary Differential Equations*, PhD thesis, Department of Mathematics, University of California at Berkeley, CA, 1995.
- [36] R. I. MCLACHLAN, *On the numerical integration of ordinary differential equations by symmetric composition methods*, *SIAM Journal on Scientific Computing*, 16 (1995), pp. 151–168.
- [37] D. W. RAND AND P. WINTERNITZ, *Nonlinear superposition principles: A new numerical method for solving matrix Riccati equations*, *Comput. Phys. Commun.*, 33 (1984), pp. 305–328.
- [38] W. T. REID, *Riccati Differential Equations*, Academic Press, New York, 1972.
- [39] A. D. RUBIO, *Distributed Parameter Control of Thermal Fluids*, PhD thesis, Department of Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1997.
- [40] I. RUSNAK, *Almost analytic representation for the solution of the differential matrix Riccati equation*, *IEEE Trans. Automat. Contr.*, 33 (1988), pp. 191–193.
- [41] R. D. RUTH, *A canonical integration technique*, *IEEE Transactions on Nuclear Science*, NS-30 (1983), pp. 2669–2671.
- [42] J. M. SANZ-SERNA AND L. ABIA, *Order conditions for canonical Runge-Kutta schemes*, *SIAM Journal on Numerical Analysis*, 28 (1991), pp. 1081–1096.

- [43] M. SORINE AND P. WINTERNITZ, *Superposition laws for solutions of differential matrix Riccati equations arising in control theory*, IEEE Trans. Automat. Contr., AC-30 (1985), pp. 266–272.
- [44] M. SUZUKI, *General theory of higher-order decomposition of exponential operators and symplectic integrators*, Physics Letters A, 165 (1992), pp. 387–395.
- [45] D. R. VAUGHAN, *A negative exponential solution for the matrix Riccati equation*, IEEE Trans. Automat. Contr., AC-14 (1969), pp. 72–75.
- [46] H. YOSHIDA, *Construction of higher order symplectic integrators*, Physics Letters A, 150 (1990), pp. 262–268.