Order reduction methods for solving large-scale differential matrix riccati equations

ORDER REDUCTION METHODS FOR SOLVING LARGE-SCALE DIFFERENTIAL MATRIX RICCATI EQUATIONS *

GERHARD KIRSTEN† AND VALERIA SIMONCINI‡

Abstract. We consider the numerical solution of large-scale symmetric differential matrix Riccati equations. Under certain hypotheses on the data, reduced order methods have recently arisen as a promising class of solution strategies, by forming low-rank approximations to the sought after solution at selected timesteps. We show that great computational and memory savings are obtained by a reduction process onto rational Krylov subspaces, as opposed to current approaches. By specifically addressing the solution of the reduced differential equation and reliable stopping criteria, we are able to obtain accurate final approximations at low memory and computational requirements. This is obtained by employing a two-phase strategy that separately enhances the accuracy of the algebraic approximation and the time integration. The new method allows us to numerically solve much larger problems than in the current literature. Numerical experiments on benchmark problems illustrate the effectiveness of the procedure with respect to existing solvers.

Key words. Differential Matrix Riccati, Rational Krylov, Extended Krylov, Linear Quadratic Regulator, Low-rank, BDF

1. Introduction.

We consider the solution of the continuous-time differential matrix Riccati equation (DRE in short) of the form

$$\dot{X}(t) = A^T X(t) + X(t)A - X(t)BB^T X(t) + C^T C, \quad X(0) = X_0, \tag{1.1}$$

in the unknown matrix $X(t) \in \mathbb{R}^{n\times n}$, where $X_0 = ZZ^T$ and $t \in [0, t_f]$. Here, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times s}$, $C \in \mathbb{R}^{p\times n}$ and $Z \in \mathbb{R}^{n\times q}$ are time invariant, and $s, p, q \ll n$. The matrix A is assumed to be large, sparse and nonsingular, whereas B, C and Z have full rank. In particular, we consider low-rank DREs, where both matrices $C^T C$ and X_0 have very low rank compared to n. Even though the matrix A is sparse, the solution $X(t)$ is typically dense and impossible to store when n is large. Under the considered hypotheses, numerical evidence seems to indicate that $X(t)$ usually has rapidly decaying singular values, hence a low-rank approximation to $X(t)$ may be considered, see e.g., [48]. For completeness, we also refer the reader to [21, 20] for results on the existence of low-rank solutions for the algebraic Sylvester and Lyapunov equations.

The DRE plays a fundamental role in optimal control theory, filter design theory, model reduction problems, as well as in differential games [2, 7, 11, 13, 38]. Equations of the form (1.1) are crucial in the numerical treatment of the linear quadratic regulator (LQR) problem [2, 13, 29]: given the state equation

$$\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t), \qquad x(0) = x_0 \tag{1.2}$$

consider the finite horizon case, where the finite time cost integral has the form

$$J(u) = x(t_f)^T P_f x(t_f) + \int_0^{t_f} \left(x(t)^T C^T C x(t) + u(t)^T u(t)\right) dt. \tag{1.3}$$

The matrix P_f is assumed to be symmetric and nonnegative definite. Assuming that the pair (A, B) is stabilizable and the pair (C, A) is detectable, the optimal input

* This version dated April 21, 2020

†Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato, 5, I-40127 Bologna, Italy, (gerhard.kirsten2@unibo.it)

‡Dipartimento di Matematica and AM^2, Università di Bologna, Piazza di Porta S. Donato, 5, I-40127 Bologna, Italy; and IMATI-CNR, Pavia Italy (valeria.simoncini@unibo.it).

$\tilde{u}(t)$, minimizing (1.3), can be determined as $\tilde{u}(t) = -B^T P(t)\tilde{x}(t)$, and the optimal trajectory is subject to $\dot{\tilde{x}} = (A - BB^T P(t))x(t)$. The matrix $P(t)$ is the solution to the DRE

$$\dot{P}(t) = A^T P(t) + P(t)A - P(t)BB^T P(t) + C^T C, \quad P(t_f) = P_f. \tag{1.4}$$

Using a common practice, we can transform (1.4) into the initial value problem (1.1) via the change of variables $X(t_f - t) = P(t)$.

Under certain assumptions, the exact solution of (1.1) can be expressed in integral form as (see e.g., [28, Theorem 8])

$$X(t) = e^{tA^T} ZZ^T e^{tA} + \int_0^t e^{(t-s)A^T} \left(C^T C - X(s)BB^T X(s)\right) e^{(t-s)A} ds, \tag{1.5}$$

so that when $t \to \infty$ the DRE reaches a steady state solution satisfying the algebraic Riccati equation (ARE)

$$0 = A^T X_\infty + X_\infty A - X_\infty BB^T X_\infty + C^T C. \tag{1.6}$$

In the framework of differential equations, the DRE is characterized by both fast and slow varying modes, hence it is classified as a stiff ordinary differential equation (ODE). The stiffness and the nonlinearity of the DRE are responsible for the difficulties in its numerical solution even on a small scale ($n < 10^3$). Several stiff integrators have been investigated, including the matrix generalizations of implicit ODE solvers [15, 12], linearization methods [14] and more recently matrix versions of splitting methods [35, 46, 47]. These methods are feasible on a small scale but fail to be efficient when n is large. In [41], iterative methods are implemented within the matrix generalization of standard implicit methods allowing for the computation of an approximate solution to the DRE when $n \gg 10^3$. These algorithms require the solution of a large *algebraic* Riccati equation at each timestep, which again raises big concerns as of storage and computational efforts.

A promising idea is to rely on a model order reduction strategy typically used in linear and nonlinear dynamical systems. In this setting, the original system is replaced with

$$\dot{\widehat{x}}(t) = A_m \widehat{x}(t) + B_m u(t), \quad y(t) = C_m \widehat{x}(t), \qquad \widehat{x}(0) = \widehat{x}_0 \tag{1.7}$$

where A_m, B_m and C_m are projections and restrictions of the original matrices onto a subspace of small dimension. The differential Riccati equation associated with this reduced order problem is solved, yielding an optimal corresponding cost function. This strategy allows for a natural low-rank approximation to the sought after DRE solution $X(t)$, obtained by interpolating the reduced order solution at selected time instances. One main feature is that a single space is used for all time snapshots, so that the approximate solutions can be kept in factored form with few memory allocations. We refer the reader to, e.g., [4] for a general presentation of algebraic reduction methods for linear dynamical systems, and to [43] for a detailed discussion motivating the reduction approach in the context of the algebraic Riccati equation.

A key ingredient for the success of the reduction methodology is the choice of the approximation space onto which the algebraic reduction is performed; [4] presents a comprehensive description of various space selections in the dynamical system setting. Following strategies already successfully adopted for the algebraic Riccati equations, the authors of [27] and [23] have independently used polynomial and extended Krylov

subspaces as approximation space, respectively, in the differential setting. A major characteristic of these spaces is that their dimension can be expanded iteratively, so that if the determined approximate solution is not sufficiently accurate, the Krylov space can be enlarged and the process continued. Several questions remain open in the methods proposed in [27],[23]. On the one hand, it is well known that polynomial Krylov subspaces require a very large dimension to satisfactorily solve real application problems, thus destroying the reduction advantages. On the other hand, the multiple timestepping proposed in the method in [23] only provides an accurate approximation at $t = t_f$, except when $X_0 = 0$. For $X_0 = ZZ^T \neq 0$ of low rank, memory requirements of the extended method grow significantly. These problems can be satisfactorily solved by using a general *rational* Krylov subspace, which is shown in various applications to be able to supply good spectral information on the involved matrices with much smaller dimensions than the polynomial and extended versions. Such gain has been experimentally reported in the literature in the solution of the *algebraic* Riccati equation. We show that great computational and memory savings can be obtained when projecting onto the fully rational Krylov subspace, and that with an appropriate implementation the extended Krylov subspace may also be competitive with certain data.

A related issue that has somehow been overlooked in the available literature is the expected final accuracy and thus the stopping criterion. Time dependence of the DRE makes the reduced problem trickier to handle than in the purely algebraic case; in particular, two intertwined issues arise: i) The accuracy of the approximate solution may vary considerably within the time interval $[0, t_f]$; ii) Throughout the reduction process the reduced ODE cannot be solved with high accuracy and, quite the opposite, low-order methods should be used to make the overall cost feasible. We analyze these difficulties in detail, and by exploiting the inherent structure of the reduced order model, we derive a two-phase strategy that first focuses on the reduction and then on the integration, in a way that is efficient for memory and CPU time usage, but also in terms of final expected accuracy.

We also discuss several algebraic properties of the approximate solution and its relation both with the solution $X(t)$ for $t \in [0, t_f]$, and with the steady state solution X_∞. These results continue a matrix analysis started in [27], where positivity and monotonicity properties of the approximate solution obtained by certain reduction methods are explored.

The paper is organized as follows. In section 2 we introduce reduction methods and discuss the use of Krylov subspace based strategies. Matrix-oriented BDF methods are recalled for the solution of the projected problem in section 3. In section 4 we devise a stopping criterion for the order reduction methods and illustrate its key role in the implementation. Section 5 is devoted to the analysis of matrix properties of the solution, as well as the reduced model, from a control theory perspective. Several numerical experiments are reported in section 6, where the new methods are also compared with state-of-the-art procedures. Our conclusions are discussed in section 7. Finally, in Appendix A and Appendix B we review some properties of the extended and rational Krylov subspaces.

Notation and definitions. Throughout the paper, the matrix I_n will denote the $n \times n$ identity matrix. In terms of norms, $\|\cdot\|$ refers to any induced matrix norm, where in particular the Frobenius norm is denoted by $\|\cdot\|_F$. A matrix A is stable (sometimes also called Hurwitz) if all its *eigenvalues* are contained in the left half open complex plane. A linear dynamical system, $\dot{x} = Ax$, is called dissipative if the real matrix A has its *field of values* contained in the left half open complex plane.

All reported experiments were performed using MATLAB 9.4 (R2018b) ([33]) on a MacBook Pro with 8-GB memory and a 2.3-GHz Intel core i5 processor.

2. Order reduction with Krylov-based subspaces. In this section, we review Krylov-based order reduction methods and show how they are applied to the DRE. Krylov subspaces that have been explored in the past years have the form

$$\mathcal{K}_m(A,N) = \text{range}\left\{[N, AN, A^2N, \ldots, A^{m-1}N]\right\} \qquad \text{polynomial}$$
$$\mathcal{EK}_m(A,N) = \mathcal{K}_m(A,N) + \mathcal{K}_m(A^{-1}, A^{-1}N) \qquad \text{extended}$$
$$\mathcal{RK}_m(A,N,\boldsymbol{s}) = \text{range}\left\{[N, (A-s_2I)^{-1}N, \ldots, \prod_{i=2}^{m}(A-s_iI)^{-1}N]\right\} \qquad \text{rational.}$$

where N is a tall matrix associated with the given problem. In the rational subspace, $\boldsymbol{s} = \{s_2, \ldots, s_m\}$ is a set of properly chosen real or complex shifts, whose computation can be performed a priori or dynamically during the generation of the subspace; we refer the reader to [44, 18] for more complete descriptions.

Krylov-based projection methods (in short generically denoted as $\mathcal{K}_m$) were first applied to ARE's in [25] (polynomial spaces) and later improved in [24] (extended space) and [45] (rational spaces). The two rational spaces prove to be far superior to the polynomial Krylov space in most reduction strategies where they are applied in the literature, as long as solving linear systems at each iteration is feasible. The differential Riccati equation has been attacked in [23] with the extended space, and in [27] with the polynomial space; here we close the gap, as far as Krylov subspaces are concerned. In addition, we address several implementation issues to make the final method computationally reliable and, to the best of our knowledge, a great competitor among the available methods for large-scale DRE problems.

While for the algebraic Riccati equation $N = C^T$, in the differential context the starting matrix for generating these spaces is given by $N = [C^T, Z]$, where $X_0 = ZZ^T$. Both matrices C and Z play a crucial role in the closed-form DRE solution matrix and are thus included to generate the projection space. The idea of reduction methods is to first project the large DRE onto the smaller subspace $\mathcal{K}_m$, then solve the projected equation, and finally expand the solution back to the original space.

Let the columns of $\mathcal{V}_m \in \mathbb{R}^{n\times d}$ span the considered Krylov subspace. Then the following Arnoldi-type relation holds,

$$A^T\mathcal{V}_m = \mathcal{V}_m\mathcal{T}_m^T + \nu_{m+1}\tau_m^T, \tag{2.1}$$

where the actual values of $\nu_{m+1} \in \mathbb{R}^n$ and τ_m^T depend on the chosen subspace. Moreover, setting $\mathcal{V}_{m+1} = [\mathcal{V}_m, \nu_{m+1}]$ we have that $\mathcal{K}_{m+1} = \text{range}(\mathcal{V}_{m+1})$, which shows that Krylov subspaces are nested, that is $\mathcal{K}_m \subseteq \mathcal{K}_{m+1}$, resulting in a dimension increase after each iteration. Matrix relations leading to (2.1) for the extended and rational Krylov subspaces are recalled in Appendix A.

Assume that $\mathcal{V}_m$ has orthonormal columns. Following similar reduction methods in the dynamical system contexts, see, e.g., [4], the reduction process consists of first projecting and restricting the original data onto the approximation space as

$$\mathcal{T}_m = \mathcal{V}_m^TA\mathcal{V}_m, \quad B_m = \mathcal{V}_m^TB, \quad Z_m = \mathcal{V}_m^TZ \quad \text{and} \quad C_m = C\mathcal{V}_m.$$

Then the following low order differential Riccati equation needs to be solved,

$$\begin{aligned}\dot{Y}_m(t) &= \mathcal{T}_m^TY_m(t) + Y_m(t)\mathcal{T}_m - Y_m(t)B_mB_m^TY_m(t) + C_m^TC_m\\ Y_m(0) &= Z_mZ_m^T,\end{aligned} \tag{2.2}$$

for $t \in [0, t_f]$. This low-dimensional DRE admits a unique solution for $t_f < \infty$, see e.g., [28]. Restrictions on the data to allow for positive, stabilizing solutions are discussed in more detail in section 5.1. An approximation to the sought after solution is then written as

$$X_m(t) = \mathcal{V}_m Y_m(t) \mathcal{V}_m^T \approx X(t), \qquad t \in [0, t_f]. \tag{2.3}$$

We stress that $X_m(t)$ is never explicitly computed, but always referred to via the matrix $\mathcal{V}_m$ and the set of matrices $Y_m(t)$ at given time instances. In fact, the matrices $Y_m(t)$ may also be numerically low rank, so that at the end of the whole process a further reduction can be performed by truncating the eigendecomposition of $Y_m(t)$ for each t.

REMARK 2.1. *The approach we have derived is solely based on the order reduction of the dynamical system (1.2). Nonetheless, and with some abuse of notation, the reduced DRE could have been formally obtained by means of a Galerkin condition on the differential equation. For $t \in [0, t_f]$ let*

$$\mathcal{R}_m(t) := \dot{X}_m(t) - A^T X_m(t) - X_m(t) A + X_m(t) B B^T X_m(t) - C^T C$$

be the residual matrix for $X_m(t) = \mathcal{V}_m Y_m \mathcal{V}_m^T$. The matrix $Y_m(t)$ is thus determined by imposing that the residual satisfies the following Galerkin condition

$$\mathcal{V}_m^T \mathcal{R}_m(t) \, \mathcal{V}_m = 0, \quad t \in [0, t_f], \tag{2.4}$$

that is, $\mathcal{R}_m(t) \perp \mathcal{K}_m$ in a matrix sense, so that the residual is forced to belong to a smaller and smaller subspace as $\mathcal{K}_m$ grows. Substituting $X_m(t) = \mathcal{V}_m Y_m(t) \mathcal{V}_m^T$ into the residual matrix, the application of the Galerkin condition results in the projected system

$$\mathcal{V}_m^T(\mathcal{V}_m \dot{Y}_m(t) \mathcal{V}_m^T - A^T \mathcal{V}_m Y_m(t) \mathcal{V}_m^T - \mathcal{V}_m Y_m(t) \mathcal{V}_m^T A + \mathcal{V}_m Y_m(t) \mathcal{V}_m^T B B^T \mathcal{V}_m Y_m(t) \mathcal{V}_m^T - C^T C) \mathcal{V}_m = 0,$$

which corresponds to (2.2). This is rigorous as long as $\dot{X}_m = \mathcal{V}_m \dot{Y}_m \mathcal{V}_m^T$ holds. □

It is crucial to realize that, as opposed to some available methods in the literature (such as, for instance, [41],[47] and the time-invariant algorithms in [30]), the approximation space is independent of the time stepping, that is a single space range($\mathcal{V}_m$) is used for all time steps. This provides enormous memory savings whenever the approximate solution is required at different time instances in $[0, t_f]$. Theoretical motivation for keeping the approximation space independent of the time-stepping is contained in [5], where it is shown that the solution of the DRE lives in an invariant Krylov-subspace[1].

The class of numerical methods we used for solving the reduced DRE is described in the next section. In the rest of this paper, we specialize the generic derivation above to the extended and rational Krylov subspaces, which greatly outperformed polynomial spaces both in terms of CPU time and memory requirements. More information on these spaces and their properties are given in Appendix A; in particular, we discuss the generation of a real rational Krylov basis in the presence of non-real shifts.

[1] We also refer the reader to the recent manuscript [6], which appeared on-line briefly before the first round of our revision.

3. BDF methods for the DRE. The numerical solution of the small-scale DRE is a well-studied topic, see, e.g., [34, 14, 35, 46]. Among the explored methods are matrix generalizations of the BDF methods [34, 15], which are computationally appealing only for small problems. Due to the reduction strength of rational Krylov subspaces, we expect the reduced DRE in (2.2) to be small enough to allow for efficient use of matrix-based BDF methods, which we are going to summarize next. For simplicity of exposition, in the rest of this section we omit the subscript in Y_m, and denote $Y^{(k+1)} = Y(t_{k+1})$. If we define

$$\mathbf{F}\left(Y^{(k+1)}\right) = \mathcal{T}_m^T Y^{(k+1)} + Y^{(k+1)}\mathcal{T}_m - Y^{(k+1)} B_m B_m^T Y^{(k+1)} + C_m^T C_m, \tag{3.1}$$

the approximation of $Y^{(k+1)}$ is given by the implicit relation

$$Y^{(k+1)} = \sum_{i=0}^{b-1} \alpha_i Y^{(k-i)} + h\beta \mathbf{F}(Y^{(k+1)}), \tag{3.2}$$

where $h = t_{k+1} - t_k$ is the stepsize and the respective α_i's and β are the coefficients of the b-step BDF method for $b \le 3$ and are given below.

p	β	α_0	α_1	α_2
1	1	1		
2	2/3	4/3	−1/3	
3	6/11	18/11	−9/11	2/11

Substituting (3.1) into (3.2) results in the following nonlinear matrix equation

$$-Y^{(k+1)} + h\beta\left(\mathcal{T}_m^T Y^{(k+1)} + Y^{(k+1)}\mathcal{T}_m - Y^{(k+1)} B_m B_m^T Y^{(k+1)} + C_m^T C_m\right) + \sum_{i=0}^{b-1} \alpha_i Y^{(k-i)} = 0,$$

which can be reformulated as the following continuous-time ARE

$$\widehat{\mathcal{T}}_m^T Y^{(k+1)} + Y^{(k+1)}\widehat{\mathcal{T}}_m - Y^{(k+1)} \widehat{B}_m \widehat{B}_m^T Y^{(k+1)} + \widehat{Q}_m = 0. \tag{3.3}$$

The coefficient matrices are given by

$$\widehat{\mathcal{T}}_m = h\beta\mathcal{T}_m - \frac{1}{2} I_m, \quad \widehat{B}_m = \sqrt{h\beta} B_m, \quad \widehat{Q}_m = h\beta C_m^T C_m + \sum_{i=0}^{b-1} \alpha_i Y^{(k-i)}.$$

The Riccati equation (3.3) can be solved using "direct" methods; see, e.g., [9]. In our experiments we used the MATLAB solver `care` from the control systems toolbox. A brief sketch of the b-step BDF method is reported in Algorithm 3.1; other approaches are discussed, e.g., in [34, 30].

We conclude the section by depicting the typical convergence behavior of the BDF methods in our context. We consider an example from [35], where the $n \times n$ matrix A stems from the spatial finite difference discretization of the following advection-diffusion equation

$$\partial_t w = \Delta w - 10 x w_x - 100 y w_y, \qquad w|_{\partial\Omega} = 0$$

on $\Omega = (0,1)^2$ with homogeneous Dirichlet boundary conditions. The choices of $B \in \mathbb{R}^{n\times 1}$ and $C \in \mathbb{R}^{1\times n}$ are given binomially as described in [35]. The initial condition is taken to be the zero matrix, that is $Z = \mathbf{0}_{n\times 1}$. We compare the obtained

Algorithm 3.1 b-step BDF method – BDF(b, ℓ)

Require: $\mathcal{T}_m \in \mathbb{R}^{d\times d}$, $B_m \in \mathbb{R}^{d\times s}$, $C_m \in \mathbb{R}^{p\times d}$, $Z_m \in \mathbb{R}^{d\times q}$, final time t_f, number of timesteps ℓ, initial approximations $Y^{(0)}, \dots, Y^{(b-1)}$.

$h = t_f/\ell$, $\widehat{\mathcal{T}}_m = h\beta\mathcal{T}_m - \frac{1}{2}I_m$, $\widehat{B}_m = \sqrt{h\beta}B_m$

for $k = 0$ **to** ℓ **do**

  $\widehat{Q}_m = h\beta C_m^T C_m + \sum_{i=0}^{b-1} \alpha_i Y^{(k-i)}$

  Solve $\widehat{\mathcal{T}}_m^T Y^{(k+1)} + Y^{(k+1)}\widehat{\mathcal{T}}_m - Y^{(k+1)}\widehat{B}_m\widehat{B}_m^T Y^{(k+1)} + \widehat{Q}_m = 0$

end for

return $Y^{(k)} \approx Y(t_k)$, $t_k = 0, h, \dots, t_f$

solution with a "reference" numerical solution $Y_{ref}(t)$ computed by an accurate but expensive method (the MATLAB function `ode23s` in our experiments), so that n is kept small, $n = 49$. The convergence behavior for $b = 1, 2, 3$ and ℓ timesteps, with $\ell = 10, 100, 1000$ is displayed in Figure 3.1. The left plot shows the error $\|Y(t) - Y_{ref}(t)\|$ as a function of t, for different values of ℓ. The right plot shows the evolution of the (1,1) component of the solution throughout the time span for the most accurate choice of BDF method, compared with that of the reference solution. These plots

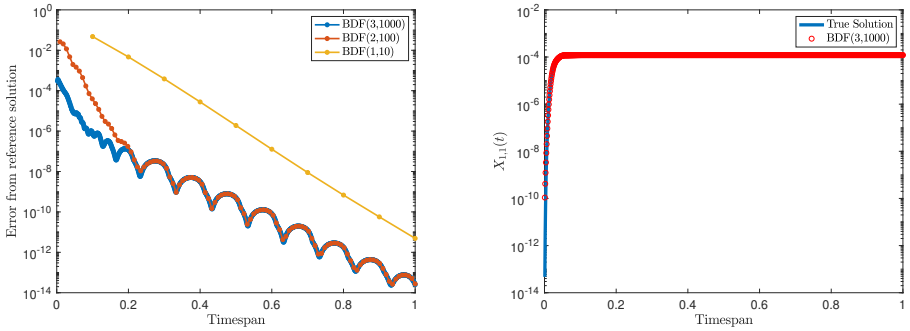

FIG. 3.1. *Typical convergence behavior of BDF methods (left) and evolution of the $X_{1,1}$ component of the reference and BDF$(3, 1000)$ solution (right).*

illustrate that we cannot expect an overall high accuracy of the projection method as long as the reduced differential equation is not solved with sufficiently good accuracy. The importance of this is discussed in more detail in the following section.

4. Stopping criterion and the complete algorithm. To complete the reduction algorithm of section 2, we need to introduce a stopping criterion. We found that it is crucial to take into account the accuracy of the numerical method employed to solve the reduced DRE, as discussed in section 3.

To derive our stopping criterion we were inspired by those in [27, 23], however, we made some important modifications. In both cited references, the authors assume that the inner problem (2.2) is solved exactly, which is not true in general. We thus consider that the numerical method solves the reduced problem with residual matrix $R_m^{(I)}(t) := \dot{Y}_m(t) - \mathbf{F}(Y_m(t))$, so that the final DRE residual can be split into two components.

PROPOSITION 4.1. *Let $X_m(t) = V_m Y_m(t) V_m^T$ be the Krylov-based approximate solution after m iterations, where $Y_m(t)$ approximately solves the reduced problem*

(2.2). *With the previous notation, the residual matrix* $\mathcal{R}_m(t) = \dot{X}_m(t) - \mathbf{F}(X_m(t))$ *satisfies*

$$\|\mathcal{R}_m(t)\|_F^2 = \|R_m^{(I)}(t)\|_F^2 + 2\|R_m^{(O)}(t)\|_F^2, \tag{4.1}$$

where $R_m^{(I)}(t) = \dot{Y}_m(t) - \mathbf{F}(Y_m(t))$ *and* $R_m^{(O)}(t) = \tau_m^T Y_m(t)$ *with* τ_m *as in (2.1).*

Proof. Substituting (2.3) into the residual $\mathcal{R}_m(t)$ we obtain

$$\begin{aligned}\mathcal{R}_m(t) = \mathcal{V}_m\dot{Y}_m(t)\mathcal{V}_m^T - A^T\mathcal{V}_m Y_m(t)\mathcal{V}_m^T - \mathcal{V}_m Y_m(t)\mathcal{V}_m^T A \\ + \mathcal{V}_m Y_m(t)\mathcal{V}_m^T BB^T\mathcal{V}_m Y_m(t)\mathcal{V}_m^T - C^T C.\end{aligned} \tag{4.2}$$

Since C^T belongs to range$(\mathcal{V}_m)$, we can write $C^T = \mathcal{V}_m C_m^T$. Using (2.1), we get

$$\begin{aligned}\mathcal{R}_m(t) = \mathcal{V}_m\dot{Y}_m(t)\mathcal{V}_m^T - (\mathcal{V}_m\mathcal{T}_m^T + \nu_{m+1}\tau_m^T)Y_m(t)\mathcal{V}_m^T - \mathcal{V}_m Y_m(t)(\mathcal{T}_m\mathcal{V}_m^T + \tau_m\nu_{m+1}^T) \\ + \mathcal{V}_m Y_m(t)\mathcal{V}_m^T BB^T\mathcal{V}_m Y_m(t)\mathcal{V}_m^T - \mathcal{V}_m C_m^T C_m\mathcal{V}_m^T.\end{aligned}$$

Since $\mathcal{V}_{m+1} = [\mathcal{V}_m, \nu_{m+1}]$, we can write $\mathcal{R}_m(t) = \mathcal{V}_{m+1}\mathcal{J}_m(t)\mathcal{V}_{m+1}^T$, where

$$\mathcal{J}_m(t) = \left[\begin{array}{c|c} \dot{Y}_m(t) - \mathcal{T}_m^T Y_m(t) - Y_m(t)\mathcal{T}_m + Y_m(t)B_m B_m^T Y_m(t) - C_m^T C_m & Y_m(t)\tau_m \\ \hline \tau_m^T Y_m(t) & \mathbf{0}\end{array}\right].$$

Let $R_m^{(I)}(t)$ be the residual of the numerical ODE inner solver. Then

$$\mathcal{J}_m(t) = \left[\begin{array}{c|c} R_m^{(I)}(t) & Y_m(t)\tau_m \\ \hline \tau_m^T Y_m(t) & \mathbf{0}\end{array}\right].$$

Since the columns of $\mathcal{V}_{m+1}$ are orthonormal,

$$\begin{aligned}\|\mathcal{R}_m(t)\|_F^2 &= \|\mathcal{V}_{m+1}\mathcal{J}_m(t)\mathcal{V}_{m+1}^T\|_F^2 = \|\mathcal{J}_m(t)\|_F^2 \\ &= \mathrm{Tr}\left(R_m^{(I)}(t)^T R_m^{(I)}(t) + 2(Y_m(t)\tau_m)(\tau_m^T Y_m(t))\right),\end{aligned}$$

that is, $\|\mathcal{R}_m(t)\|_F^2 = \|R_m^{(I)}(t)\|_F^2 + 2\|\tau_m^T Y_m(t)\|_F^2$, and the result follows. ∎

The expression for $\mathcal{J}_m(t)$ emphasizes that at each iteration m the matrix $Y_m(t)$ is the exact solution of

$$\dot{Y}_m(t) - \mathcal{T}_m^T Y_m(t) - Y_m(t)\mathcal{T}_m + Y_m(t)B_m B_m^T Y_m(t) - C_m C_m^T - R_m^{(I)}(t) = 0.$$

Hence, as long as $\|R_m^{(I)}(t)\|_F$ is not very small, the increase of m aims at more and more accurately approximating a "nearby" differential problem to the truly projected one, with a term $R_m^{(I)}(t)$ that varies with m. Hence, $X_m(t) = \mathcal{V}_m Y_m \mathcal{V}_m^T$ is an approximation not to $X(t)$, but to the solution of a differential problem with an additional term whose projection onto the space is $R_m^{(I)}(t)$.

Proposition 4.1 also implies that we cannot expect an overall small residual norm if either of the two partial residual norms $\|R_m^{(I)}(t)\|_F$, $\|R_m^{(O)}(t))\|_F$ is not small. In particular, we observe that the two residuals can be made small independently. Therefore we propose the following practical strategy:

(i) Run the algorithm as presented, with a low-order cheap ODE inner solver (i.e., BDF(1, ℓ) with ℓ relatively small) and use $R_m^{(O)}(t)$ in the stopping criterion;

(ii) Once completed step (i) after $\widehat{m}$ iterations, use the matrices $T_{\widehat{m}}, C_{\widehat{m}}, B_{\widehat{m}}$ and $Z_{\widehat{m}}$ to refine the ODE inner solution by using a higher-order ODE solver for the projected system.

The final matrix $Y_{\widehat{m}}(t)$ obtained in step (ii) will provide a more accurate solution matrix than what would have been obtained at the end of step (i). We emphasize that *any* ODE method for small and medium scale DREs could be used at steps (i) and (ii). Our choice of BDF(1, ℓ) is due to its good trade-off between accuracy and computational effort; other approaches could be considered.

To complete the description of the stopping criterion, we recall that $R_m^{(O)}(t)$ depends on t, so that we need to estimate the integral over the whole time interval by means of a quadrature formula, that is

$$\left\| \int_0^{t_f} R_m^{(O)}(\gamma) d\gamma \right\|_F = \left\| \tau_m^T \int_0^{t_f} Y_m(\gamma) d\gamma \right\|_F \approx \left\| \tau_m^T \sum_{j=1}^{\ell} \frac{t_f}{\ell} Y_m(t_j) \right\|_F =: \rho_m. \tag{4.3}$$

where the interval $[0, t_f]$ has been divided into ℓ intervals with nodes t_j.

The overall algorithm[2] based on the rational Krylov subspace method is reported in Algorithm 4.1, while the algorithm based on the extended method is postponed to Appendix B. Several implementation issues of Algorithm 4.1 are also described in Appendix A, such as the use of a real basis in case of complex shifts s_j in the basis construction.

Algorithm 4.1 RKSM-DRE

Require: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times s}$, $C \in \mathbb{R}^{p\times n}$, $Z \in \mathbb{R}^{n\times q}$, *tol*, t_f, ℓ, $\mathbf{s_0} = \{s_0^{(1)}, s_0^{(2)}\}$
(i) Perform reduced QR: $[C^T, Z] = V_1 \Lambda_1$
Set $\mathcal{V}_1 \equiv V_1$
for $m = 2, 3 \ldots$
Compute the next shift and add it to $\mathbf{s_0}$
Compute the next *real* basis block V_m
Set $\mathcal{V}_m = [\mathcal{V}_{m-1}, V_m]$
Update $\mathcal{T}_m = \mathcal{V}_m^T A \mathcal{V}_m$ and $B_m = \mathcal{V}_m^T B$, $Z_m = \mathcal{V}_m^T Z$ and $C_m = C\mathcal{V}_m$
Integrate (2.2) from 0 to t_f using BDF$(1, \ell)$
Compute ρ_m using (4.3) where $\tau_m^T = G_m^T$
if $\rho_m < tol$
go to (ii)
end if
end for
(ii) Refinement: solve (2.2) with a more accurate integrator
Compute $Y_m(t_j) = \widehat{Y}_m(t_j)\widehat{Y}_m(t_j)^T$, $j = 1, \ldots, \ell$ using the truncated SVD
return $\mathcal{V}_m \in \mathbb{R}^{n\times m(p+q)}$ and ℓ factors $\widehat{Y}_m(t_j) \in \mathbb{R}^{m(p+q)\times r}$, $j = 1, \ldots, \ell$

5. Stability analysis and error bounds. In this section, we provide a few results on the spectral and convergence properties of the obtained approximate solution We first inspect some properties of the asymptotic matrix solution, which solves

[2] A Matlab implementation of Algorithm 4.1 can be downloaded from either https://github.com/Gerhard-Kirsten/Differential-Riccati-RKSM or http://www.dm.unibo.it/~simoncin/software.

the algebraic Riccati equation. Then we propose a bound for the error matrix, in an appropriate functional norm.

5.1. Properties of the (steady state) algebraic Riccati equation. Properties associated with the algebraic Riccati equation – as asymptotic solution to the DRE – are well known in linear quadratic optimal control, see, e.g., [29, 13]. In particular, classical uniqueness and stabilization properties of the solution (see, e.g., [13, Lemma 12.7.2]), can directly be extended to the reduced DRE (2.2).

COROLLARY 5.1. *Let $(\mathcal{T}_m, B_m, C_m)$ be stabilizable and detectable system. Let $Y_m(t)$ be the solution of (2.2) at time t and let $Y_m^\infty = \lim_{t\to\infty} Y_m(t)$. Then Y_m^∞ is the unique symmetric nonnegative definite solution and the only stabilizing solution to the (reduced) algebraic Riccati equation*

$$0 = \mathcal{T}_m^T Y_m^\infty + Y_m^\infty \mathcal{T}_m - Y_m^\infty B_m B_m^T Y_m^\infty + C_m^T C_m. \tag{5.1}$$

Moreover, if the pair $(C_m, \mathcal{T}_m)$ is observable, Y_m^∞ is strictly positive definite.

We notice that the stabilizability and detectability properties of $(\mathcal{T}_m, B_m, C_m)$ are not necessarily implied by those on (A, B, C). Nevertheless, it is shown in [43] that if there exists a feedback matrix K, such that the linear dynamical system $\dot{x} = (A - BK)x$ is dissipative, then the pair $(\mathcal{T}_m, B_m)$ is stabilizable. A similar result can be formulated for the detectability of $(C_m, \mathcal{T}_m)$, since by duality reasoning, $(C_m, \mathcal{T}_m)$ is detectable if $(\mathcal{T}_m^T, C_m^T)$ is stabilizable. The question regarding the existence of such a feedback matrix, with respect to A and B (A^T and C^T), is addressed in [22].

With these results, we can relate the asymptotic solution of the original and projected problems. Let $X_m(t) = \mathcal{V}_m Y_m(t) \mathcal{V}_m^T$ and $X_m^a = \mathcal{V}_m Y_m^\infty \mathcal{V}_m^T$ respectively be approximate solutions to (1.1) and (1.6) by a projection onto range$(\mathcal{V}_m)$. If there exist matrices K and L such that the systems $\dot{x} = (A - BK)x$ and $\dot{x} = (A^T - C^T L)x$ are dissipative, then

$$\lim_{t\to\infty} X_m(t) = \mathcal{V}_m \lim_{t\to\infty} Y_m(t) \mathcal{V}_m^T = \mathcal{V}_m Y_m^\infty \mathcal{V}_m^T = X_m^a, \tag{5.2}$$

that is, X_m^a is the steady state solution of $X_m(t)$ when projected onto the same basis.

Under the hypotheses that (A, B, C) is a stabilizable and detectable system, there exists a unique non-negative and stabilizing solution X_∞ to (1.6) (see, e.g., [28, Theorem 5]). In [43] a bound was derived for the error $X_\infty - X_m^a$ in terms of the matrix residual norm. Here we complete the argument by stating that in exact arithmetic and if the whole space can be spanned, the obtained approximate solution equals X_∞.

PROPOSITION 5.2. *Suppose (A, B, C) is stabilizable and detectable. Assume it is possible to determine m_* such that dim(range$(\mathcal{V}_{m_*})) = n$, and let $X_{m_*}^a = \mathcal{V}_{m_*} Y_{m_*}^\infty \mathcal{V}_{m_*}^T$ be the obtained approximate solution of (1.6) after m_* iterations. Then, $X_{m_*}^a = X_\infty$.*

Proof. Since $\mathcal{V}_{m_*}$ is square and orthogonal the projected ARE is given by

$$0 = \mathcal{V}_{m_*}^T A^T \mathcal{V}_{m_*} Y_{m_*}^\infty + Y_{m_*}^\infty \mathcal{V}_{m_*}^T A \mathcal{V}_{m_*} - Y_{m_*}^\infty \mathcal{V}_{m_*}^T B B^T \mathcal{V}_{m_*} Y_{m_*}^\infty + \mathcal{V}_{m_*}^T C^T C \mathcal{V}_{m_*}.$$

From (A, B, C) stabilizable and detectable it follows that $(\mathcal{V}_{m_*}^T A \mathcal{V}_{m_*}, \mathcal{V}_{m_*}^T B, C\mathcal{V}_{m_*})$ is also stabilizable and detectable, so that $Y_{m_*}^\infty \geq 0$ and stabilizing. Multiplying by $\mathcal{V}_{m*}$ (by $\mathcal{V}_{m*}^T$) from the left (right), we obtain $0 = A^T X_{m_*}^a + X_{m_*}^a A - X_{m_*}^a B B^T X_{m_*}^a + C^T C$, that is, $X_{m_*}^a \geq 0$ is a solution to the original ARE. Since X_∞ is the unique nonnegative definite solution, it must be $X_{m_*}^a = X_\infty$. ∎

5.2. Error bound for the differential Riccati equation. In this section we derive a bound for the maximum error obtained by the reduction process, in terms of the residual

$$R_m(t) = A^T X_m(t) + X_m(t)A - X_m(t)BB^T X_m(t) + C^T C - \dot{X}_m(t). \tag{5.3}$$

Note that $R_m(t)$ is the residual matrix with respect to the exact solution of the reduced differential problem, that is, it also includes the discretization error. A similar bound on the error has been derived for the nonsymmetric DRE in [3], which used matrix perturbation techniques from [26].

PROPOSITION 5.3. *For $t \in [0, t_f]$ let $\mathcal{E}_m(t) = X(t) - X_m(t)$ and assume that $\mathcal{A}(t) := A - BB^T X(t)$ is stable for all $t \in [0, t_f]$. Denote*

$$\nu := \max_{t\in[0,t_f]} \left\{ \int_0^t \|\Phi_{\mathcal{A}^T}(t,s)\| \, \|\Phi_{\mathcal{A}}(t,s)\| ds \right\},$$

where $\Phi_{\mathcal{A}}$ is the state-transition matrix satisfying $\frac{\partial \Phi_{\mathcal{A}}(t,s)}{\partial t} = \mathcal{A}(t)\Phi_{\mathcal{A}}(t,s), \Phi_{\mathcal{A}}(s,s) = I$.

If $4\nu^2\|B\|^2\|R_m\|_{\infty_t} < 1$, then

$$\|\mathcal{E}_m\|_{\infty_t} \le 2\nu\|R_m\|_{\infty_t},$$

where $\|L\|_{\infty_t} = \max_{t\in[0,t_f]} \|L(t)\|$ for any continuous matrix function $L(t)$.

Proof. By subtracting (5.3) from (1.1) and manipulating terms we obtain

$$\dot{\mathcal{E}}_m(t) = (A - BB^T X(t))^T \mathcal{E}_m(t) + \mathcal{E}_m(t)(A - BB^T X(t)) + \mathcal{E}_m(t)BB^T \mathcal{E}_m(t) + R_m(t),$$

with $\mathcal{E}_m(0) = 0$. Therefore, by the variation of constants formula (see, e.g., [28])

$$\mathcal{E}_m(t) = \int_0^t \Phi_{\mathcal{A}^T}(t,s) \left(R_m(s) + \mathcal{E}_m(s)BB^T \mathcal{E}_m(s) \right) \Phi_{\mathcal{A}}(t,s) ds.$$

Taking norms yields

$$\|\mathcal{E}_m(t)\|_{\infty_t} \le \max_{t\in[0,t_f]} \int_0^t \|\Phi_{\mathcal{A}^T}(t,s)\| \, \|\Phi_{\mathcal{A}}(t,s)\| \left(\|R_m(s)\| + \|\mathcal{E}_m(s)\|^2 \|B\|^2 \right) ds,$$

so that $\|\mathcal{E}_m(t)\|_{\infty_t} \le \nu \left(\|R_m(t)\|_{\infty_t} + \|\mathcal{E}_m(t)\|^2_{\infty_t} \|B\|^2 \right)$. Solving this quadratic inequality yields

$$\|\mathcal{E}_m(t)\|_{\infty_t} \le \frac{1 - \sqrt{1 - 4\nu^2\|B\|^2\|R_m\|_{\infty_t}}}{2\nu\|B\|^2}.$$

The result follows from multiplying and dividing by $(1 + \sqrt{1 - 4\nu^2\|B\|^2\|R_m\|_{\infty_t}})$ and noticing that at the denominator this quantity can be bounded from below by 1. ∎

We conclude with a remark on the intuitive fact that if the approximation space spans the whole space, the obtained solution by projection necessarily coincides with the sought after solution of the DRE.

REMARK 5.4. *If it is possible to determine m_* such that $dim(\mathcal{V}_{m_*}) = n$, then the approximate solution $X_{m_*}(t)$ coincides with $X(t)$ for all $t \ge 0$. Indeed, let us write $X_{m_*}(t) = \mathcal{V}_{m_*} Y_{m_*}(t) \mathcal{V}_{m_*}^T$, where $\mathcal{V}_{m_*}$ is square and orthogonal. The reduced DRE is given by*

$$\dot{Y}_{m_*} = \mathcal{V}_{m_*}^T A^T \mathcal{V}_{m_*} Y_{m_*} + Y_{m_*} \mathcal{V}_{m_*}^T A \mathcal{V}_{m_*} - Y_{m_*} \mathcal{V}_{m_*}^T BB^T \mathcal{V}_{m_*} Y_{m_*} + \mathcal{V}_{m_*}^T C^T C \mathcal{V}_{m_*}$$

with $Y_{m_*} = Y_{m_*}(t)$. Multiplying by $\mathcal{V}_{m*}$ (by $\mathcal{V}_{m*}^T$) from the left (right), we obtain

$$\dot{X}_{m_*}(t) = A^T X_{m_*}(t) + X_{m_*}(t)A - X_{m_*}(t)BB^T X_{m_*}(t) + C^T C,$$

hence, $X_{m_*}(t) \geq 0$ is a solution of (1.1). Since $X(t)$ is the unique nonnegative definite solution of (1.1) for any $X_0 \geq 0$ (see, e.g., [28]), then $X_{m_*}(t) = X(t)$ for $t \geq 0$. □

6. Numerical experiments. In this section we report on our numerical experience with the developed techniques. We consider two artificial symmetric and nonsymmetric model problems, as well as three (of which two are nonsymmetric) standard benchmark problems. Information about the considered data is contained in Table 1. For the first two datasets displayed in Table 1, the matrix A stems from the finite difference discretization with homogenous Dirichlet boundary conditions on the unit square and unit cube, respectively. The first matrix (SYM2D) comes from the finite difference discretization of the two-dimensional Laplace operator in the unit square with homogeneous boundary conditions, while the second matrix (NSYM3D) stems from the finite difference discretization of the three-dimensional differential operator

$$\mathcal{L}(u) = e^{xy}(u_x)_x + e^{xy}(u_y)_y + (u_z)_z + (1+x)e^{-x}u_x + y^2 u_y + 10(x+y)u_z,$$

in the unit cube, with homogeneous boundary conditions. For both datasets, the matrices B, C and Z are selected randomly with normally distributed entries. The realizations of the random matrices are fixed for both examples using the MATLAB command `rng`: for B, C and Z we use `rng`(7), `rng`(2) and `rng`(3), respectively. The following two datasets (CHIP and FLOW) are taken from [1], and all coefficient matrices ($\widehat{A}$, $\widehat{B}$, $\widehat{C}$ and $\widehat{E}$) are contained in the datasets, which stem from the dynamical system

$$\widehat{E}\dot{\widehat{x}} = \widehat{A}\widehat{x} + \widehat{B}u, \qquad \widehat{y} = \widehat{C}\widehat{x}.$$

Since $\widehat{E}$ is diagonal and nonsingular, it is incorporated as $A = \widehat{E}^{-\frac{1}{2}}\widehat{A}\widehat{E}^{-\frac{1}{2}}$, while $\widehat{B}$ and $\widehat{C}$ are updated accordingly to form B and C.

The final considered dataset (RAIL) stems from a semi-discretized heat transfer problem for optimal cooling of steel profiles[3] [8]. We consider the largest of the four available discretizations (file `rail_79841_c60` containing $\widehat{A}, \widehat{B}, \widehat{C}$ and $\widehat{E}$) with $n = 79841$. The symmetric and positive definite mass matrix $\widehat{E}$ has a sparsity pattern very similar to $\widehat{A}$. Both matrices are therefore reordered by the same approximate minimum degree (RKSM-DRE) or reverse Cuthill-McKee (EKSM-DRE) permutation to limit fill-in. The state-space transformation is done using the Cholesky factorization of $\widehat{E}$. More precisely, let $\widehat{E} = \widehat{E}_L\widehat{E}_L^T$ with $\widehat{E}_L$ lower triangular, and consider the transformed state $x = \widehat{E}_L^T\widehat{x}$. Then

$$\dot{x} = Ax + Bu, \qquad y = Cx,$$

with $A = \widehat{E}_L^{-1}\widehat{A}\widehat{E}_L^{-T}$, $B = \widehat{E}_L^{-1}\widehat{B}$ and $C = \widehat{C}\widehat{E}_L^{-T}$. These matrices are *never* explicitly formed, rather they are commonly applied implicitly by solves with the factor $\widehat{E}_L$ at each iteration; see, e.g., [18, 42].

The initial low-rank factors are selected as the zero vector for FLOW, $Z = \sin g$ for CHIP and $Z = \cos g$ for RAIL, where $g \in \mathbb{R}^{n\times 1}$ is a vector with entries in $[0, 2\pi]$. Other sufficiently general choices were tried during our numerical investigation however results did not significantly differ from the ones we report.

[3] Data available at `http://modelreduction.org/index.php/Steel_Profile`

TABLE 1
Relevant information concerning the experimental data

Name	n	$p/s/q$	$\|\|A\|\|_F$	$\|\|B\|\|_F$	$\|\|C\|\|_F$	$\|\|Z\|\|_F$	$\|\|E\|\|_F$
SYM2D	640000	5/1/1	$3.6 \cdot 10^3$	$8.0 \cdot 10^2$	$1.8 \cdot 10^3$	$8.0 \cdot 10^2$	$8 \cdot 10^2$
NSYM3D	64000	6/1/3	$2.0 \cdot 10^3$	$2.5 \cdot 10^2$	$6.2 \cdot 10^2$	$2.8 \cdot 10^2$	$2.5 \cdot 10^2$
Name	n	$p/s/q$	$\|\|\widehat{A}\|\|_F$	$\|\|\widehat{B}\|\|_F$	$\|\|\widehat{C}\|\|_F$	$\|\|Z\|\|_F$	$\|\|\widehat{E}\|\|_F$
CHIP	20082	5/1/1	$2.2 \cdot 10^6$	$1.7 \cdot 10^2$	$3.3 \cdot 10^4$	$1.0 \cdot 10^2$	$2 \cdot 10^{-4}$
FLOW	9669	5/1/1	$4.5 \cdot 10^6$	$2.0 \cdot 10^4$	$1.2 \cdot 10^3$	–	$6.8 \cdot 10^0$
RAIL	79841	7/6/1	$7 \cdot 10^{-3}$	$1 \cdot 10^{-7}$	$6.2 \cdot 10^0$	$1.9 \cdot 10^2$	$8 \cdot 10^{-4}$

Performance of the projection methods. We first investigate the convergence behavior of the outer solver. The quantity we monitor in our stopping criterion is the backward error in an integral norm given by

$$\frac{\rho_m}{t_f \|C\|_F^2 + 2\xi_m + \psi_m}, \tag{6.1}$$

with ρ_m as in (4.3) and

$$\xi_m = \left\| A^T \mathcal{V}_m \int_0^{t_f} Y_m(\gamma)\, d\gamma \right\|_F \quad \text{and} \quad \psi_m = \left\| \int_0^{t_f} Y_m(\gamma) \mathcal{V}_m^T B B^T \mathcal{V}_m Y_m(\gamma) d\gamma \right\|_F .$$

The integrals are approximated by a quadrature formula in a similar fashion to (4.3), and we note that ξ_m can be cheaply computed by using the Arnoldi-type relation.

For all datasets, the stopping tolerance was chosen as 10^{-7}. For the first four datasets, $t_f = 1$ and BDF(1,10) is used as inner solver. For RAIL, $t_f = 4500$ (see e.g., [8] for further details about the setting) and BDF(1,45) is used as inner solver. Figures 6.1 to 6.5 display the convergence of the rational Krylov subspace method (Algorithm 4.1, RKSM-DRE) and of the extended Krylov subspace method (Algorithm B.1, EKSM-DRE). The left plots report the history of the backward error as the approximation space dimension increases, while the right plots display the same history versus the total computational time (in seconds) as the iterations proceed. We notice that the cost of the refinement step is not taken into account in these first tests.

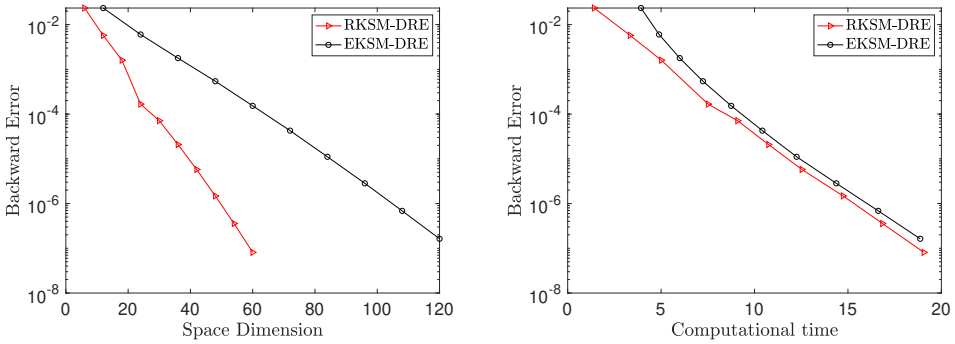

FIG. 6.1. SYM2D: *Convergence history for* EKSM-DRE *and* RKSM-DRE. *Left: backward error versus space dimension. Right: backward error versus computational time.*

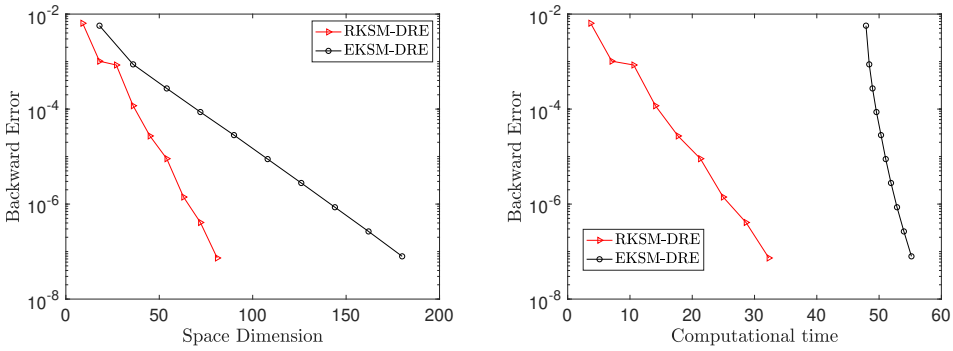

FIG. 6.2. NSYM3D: *Convergence history for* EKSM-DRE *and* RKSM-DRE. *Left: backward error versus space dimension. Right: backward error versus computational time.*

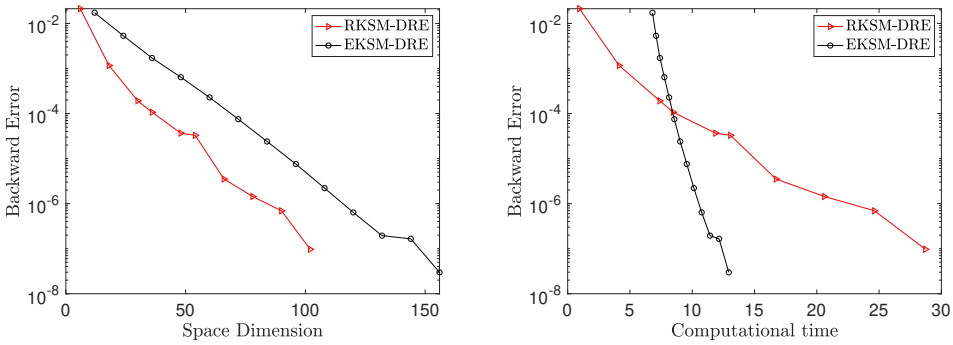

FIG. 6.3. CHIP: *Convergence history for* EKSM-DRE *and* RKSM-DRE. *Left: backward error versus space dimension. Right: backward error versus computational time.*

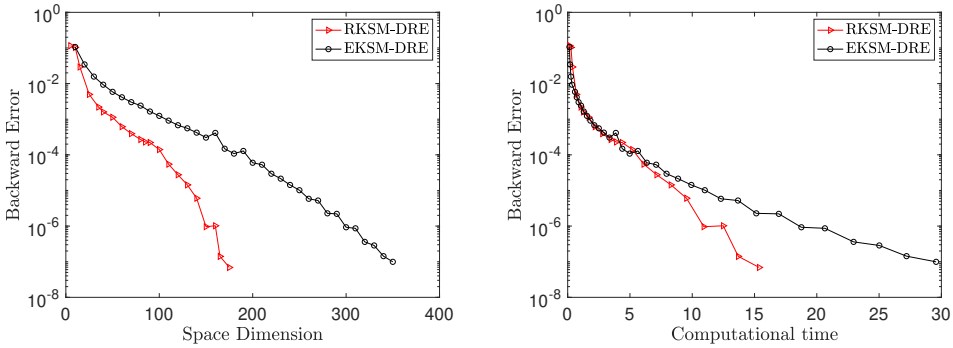

FIG. 6.4. FLOW: *Convergence history for* EKSM-DRE *and* RKSM-DRE. *Left: backward error versus space dimension. Right: backward error versus computational time.*

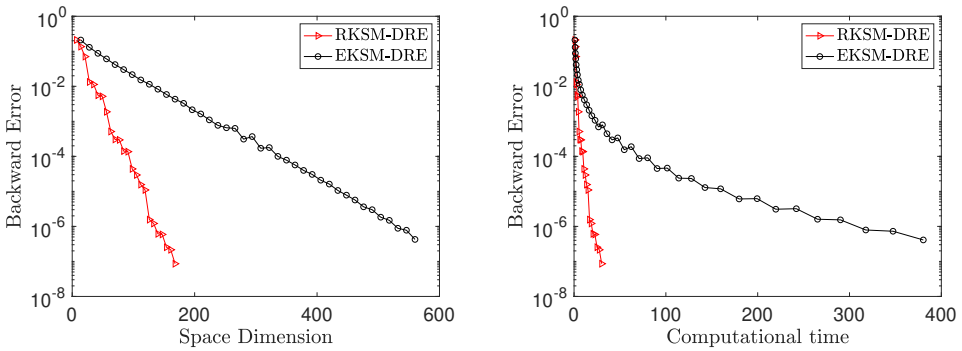

FIG. 6.5. RAIL: *Convergence history for* EKSM-DRE *and* RKSM-DRE. *Left: backward error versus space dimension. Right: backward error versus computational time.*

For the dataset SYM2D, the large algebraic linear system in RKSM-DRE was iteratively solved by implementing a block conjugate gradient algorithm, with an inner tolerance of 10^{-10}, preconditioned with an incomplete Cholesky factorization with drop tolerance 10^{-4}. For all other datasets, the MATLAB built-in backslash operator was used. For EKSM-DRE the coefficient matrix A used to generate the Krylov space remains constant, hence a sparse reordered Cholesky (for SYM2D and RAIL) or LU (for all other datasets) factorization was performed once and for all at the start of the algorithm. Therefore, only sparse triangular solves are required at each iteration. Clearly, the cost of the initial factorization depends on the size and density of the coefficient matrix. These two cost stages are particularly noticeable in the right plots of Figure 6.2 and Figure 6.3, where the EKSM-DRE curve starts towards the right of the plot, while the rest of the computation throughout the iterations is significantly faster.

In the implementation of RKSM-DRE it is possible to decide a priori whether to use only real or generically complex shifts. Our experiments showed that complex shifts were unnecessary for SYM2D and NSYM3D and, in fact, slowed down convergence when used. On the other hand, the use of general complex shifts proved to be crucial for the efficient convergence of RKSM-DRE for CHIP and FLOW. For the symmetric data in RAIL no complex shifts were used. We mention in passing that both algorithms are implemented so that the inner solves of (2.2) and the residual computations are performed at each iteration; for more demanding data we would advise a user to perform these computations only periodically to save on computational time.

Comparing performance, we observe that the two algorithms have alternating leadership in terms of computational time, but that RKSM-DRE almost consistently requires half the space dimension of EKSM-DRE. This is expected as the space dimension of EKSM-DRE increases with twice the number of columns per iteration, in comparison to RKSM-DRE. This observation is crucial at the refinement step, where it could be considerably more expensive to accurately integrate a DRE of dimension $2m(p+q)$ in comparison to a DRE with approximately half the dimension.

To have a clearer picture of how the various steps influence the performance of the methods, Table 2 depicts the overall computational time for the system solves, the orthogonalization steps and the integration of the reduced systems for each algorithm. For EKSM-DRE the CPU time required for the Cholesky and LU factorizations are

included in the solving time, but indicated in brackets as well. It is particularly interesting to notice the small percentage of time required by RKSM-DRE in comparison to EKSM-DRE for integrating the reduced system, confirming the comment made in the previous paragraph.

TABLE 2
A breakdown of the computational time for the considered methods for the first two datasets.

Data	Method	System solves (s)	Orthogonalisation steps (s)	Integration steps (s)
SYM2D	RKSM-DRE	6.1	6.9	0.4
	EKSM-DRE	8.6 (2.7)	12.1	1.3
NSYM3D	RKSM-DRE	38.3	0.9	0.8
	EKSM-DRE	48.6 (43.5)	1.6	4.0

Comparisons with other BDF based methods. We compare the two projection methods RKSM-DRE and EKSM-DRE with low-rank methods that have been developed following different strategies. The package M.E.S.S. [41], for instance, can solve Lyapunov and Riccati equations, and perform model reduction of systems in state space and structured differential algebraic form, with time-variant and time-invariant data. For our purposes, the solvers in M.E.S.S. first discretize the time interval, and then solve the algebraic Riccati equation resulting from the ODE solver at each time step. Therefore, the approximation strategy employed at each time iteration to solve the algebraic problem is completely independent, and the obtained low-rank numerical solution needs to be stored separately. More precisely, if ℓ timesteps are performed, the procedure requires solving at least ℓ AREs of large dimensions, delivering the corresponding low-rank approximate solutions. Moreover, the rank of the constant term in the ARE increases with the time step, due to the way the ODE solver is structured, further increasing the complexity of the ARE numerical treatment. In our experiments with M.E.S.S. we only requested the approximate solution at the final stage. If the whole approximate solution matrix is requested at different time instances, the memory requirements will grow linearly with that. The overall strategy appears to be memory and computational time consuming, therefore we considered datasets of reduced size for our comparisons, as displayed in Table 3. The considered timespans were left unchanged.

TABLE 3
Data information for comparisons between projection-based methods and M.E.S.S.

Name	n	$p/s/q$	$\|\|A\|\|_F$	$\|\|B\|\|_F$	$\|\|C\|\|_F$	$\|\|Z\|\|_F$	$\|\|E\|\|_F$
SYM2D	40000	5/1/1	$1.3 \cdot 10^3$	$3.0 \cdot 10^2$	$6.7 \cdot 10^2$	$3.0 \cdot 10^2$	$2 \cdot 10^2$
NSYM3D	8000	6/1/3	$6.1 \cdot 10^2$	$7.7 \cdot 10^1$	$1.9 \cdot 10^2$	$8.3 \cdot 10^1$	$2.8 \cdot 10^2$
Name	n	$p/s/q$	$\|\|\widehat{A}\|\|_F$	$\|\|\widehat{B}\|\|_F$	$\|\|\widehat{C}\|\|_F$	$\|\|Z\|\|_F$	$\|\|\widehat{E}\|\|_F$
FLOW	9669	5/1/1	$4.5 \cdot 10^6$	$2.0 \cdot 10^4$	$1.2 \cdot 10^3$	0	$6.8 \cdot 10^0$
RAIL	20209	7/6/1	$4 \cdot 10^{-3}$	$2.1 \cdot 10^{-7}$	$6.2 \cdot 10^0$	$1.9 \cdot 10^2$	$2 \cdot 10^{-4}$

Our experimental results are displayed in Tables 4 to 7; we remark that now also the refinement cost is taken into account in the projection methods. In all tables, the code BDF(b, ℓ) refers to the BDF method implemented in the refinement procedure of the reduction methods and in the time discretization procedure of M.E.S.S.

TABLE 4

SYM2D: *Storage and computational time comparison of* RKSM-DRE, EKSM-DRE *and* M.E.S.S.. *Reduction phase performed with* BDF*(1,10), refinement phase with* BDF*(2,100). In* M.E.S.S. *only the approximate solution at the final time is stored, with no solutions at intermediate time instances returned.*

Method	# n-long Vecs	Min/Max rank	Reduction phase(s)	Refine phase(s)	Tot CPU time(s)
RKSM-DRE	54	23/43	1.4	0.15	1.6
EKSM-DRE	120	23/43	1.7	1.9	3.6
M.E.S.S.-BDF(1,10)	988	58/75			319.9
M.E.S.S.-BDF(2,100)	1032	58/86			4005.4

The tables show the storage requirements in terms of n-length vectors, the minimum and maximum approximate solution rank (with a truncation tolerance 10^{-8} for the projection methods) within the set of solutions, the CPU time break out of projection and refinement phases for the two projected methods, and finally the total CPU time. The stopping tolerance for all algebraic methods – that is the two projection methods and the Newton–Kleinmann-type method used in M.E.S.S. to solve each ARE – is set to 10^{-7}.

In the M.E.S.S. software the user can either select a stopping tolerance (to be used for all solvers within the Newton–Kleinmann strategy) or a maximum number of iterations. We have experimented with both cases, where the maximum number of iterations was detected (a-posteriori) as the maximum number of iterations required within M.E.S.S to reach the tolerance of 10^{-7}. It was observed that, in the majority of cases, avoiding the residual computation may, in fact, slow down the computational procedure. This is due to the possibility of performing several unnecessary iterations at some timesteps after the desired accuracy has in fact been reached. We therefore only report the results of the more realistic, reliable case where a stopping tolerance is selected beforehand. Galerkin acceleration is used to boost the performance of Newton–Kleinmann.

TABLE 5

NSYM3D: *Storage and computational time comparison of* RKSM-DRE, EKSM-DRE *and* M.E.S.S.. *Reduction phase performed with* BDF*(1,10), refinement phase with* BDF*(2,100). In* M.E.S.S. *only the approximate solution at the final time is stored, with no solutions at intermediate time instances returned.*

Method	# n-long Vecs	Min/Max rank	Reduction phase(s)	Refine phase(s)	Tot CPU time(s)
RKSM-DRE	90	36/66	2.4	2.8	5.2
EKSM-DRE	180	36/66	2.6	5.4	8.0
M.E.S.S.-BDF(1,10)	1116	71/90			431.0
M.E.S.S.-BDF(2,100)	1152	67/94			4965.0

All numbers in the tables illustrate the large computational costs of M.E.S.S., as expected by the strategy "first time-discretize, then solve", whereas both projection methods require just a few seconds of CPU in most cases.

The storage requirements of both reduction methods is independent of the number of timesteps where the solution is required. This is due to the fact that only a few n-long basis vectors need to be generated and stored, while only the reduced problem

solution $Y_m(t)$ changes at the timesteps t. The memory requirements of M.E.S.S. are measured as the dimensions of the low-rank factor returned by the Newton-Kleinmann procedure, before column compression, at the final timestep. The dimension decreases significantly with the column compression. In our experiments we only stored the approximate solution at the last time step, however memory will be correspondingly higher if the whole approximation matrix is required at more instances (memory will thus grow linearly with the number of time instances to be monitored).

Between the two projection methods, we observe that the extended space yields a significantly larger basis than the actual approximate solution rank it produces. This means that the approximate solution belongs to a much smaller space than the one constructed by EKSM-DRE. This is far less so with RKSM-DRE. The different behavior confirms what has been already observed for the two methods in the ARE case [45].

TABLE 6

FLOW: *Storage and computational time comparison of* RKSM-DRE, EKSM-DRE *and* M.E.S.S.. *Reduction phase performed with* BDF*(1,10), refinement phase with* BDF*(2,100). In* M.E.S.S. *only the approximate solution at the final time is stored, with no solutions at intermediate time instances returned.*

Method	# n-long Vecs	Min/Max rank	Reduction phase(s)	Refine phase(s)	Tot CPU time(s)
RKSM-DRE	175	95/100	11.8	4.5	16.3
EKSM-DRE	350	95/100	27.4	23.5	50.9
M.E.S.S.-BDF(1,10)	1280	87/106			431.7

TABLE 7

RAIL: *Storage and computational time comparison of* RKSM-DRE, EKSM-DRE *and* M.E.S.S.. *Reduction phase performed with* BDF*(1,10), refinement phase with* BDF*(2,100). In* M.E.S.S. *only the approximate solution at the final time is stored, with no solutions at intermediate time instances returned.*

Method	# n-long Vecs	Min/Max rank	Reduction phase(s)	Refine phase(s)	Tot CPU time(s)
RKSM-DRE	168	153/160	6.4	3.3	9.7
EKSM-DRE	462	153/160	39.2	5.7	44.9
M.E.S.S.-BDF(1,10)	6345	151/158			705.3
M.E.S.S.-BDF(2,100)	4023	124/158			3396.5

Comparisons with splitting methods. We next compare RKSM-DRE with the fourth order additive splitting method (SPLIT-ADD4(ℓ)) developed in [47]. The method is based on splitting the DRE into the linear and non-linear subproblems, for which respective closed form solutions exist and are explicitly approximated. The numerical solutions to the subproblems are then recombined to approximate the solution to the full problem, by means of an additive splitting scheme. The main computational effort is due to the repeated evaluation of matrix exponentials, which has been resolved by using a Krylov-based matrix exponential approximation. Similar to the issue discussed with M.E.S.S. in the previous section, the ℓ (factored) solution matrices are independently calculated at each timestep, leading to significant memory requirements.

To ensure that we are comparing methods with similar approximation accuracies, we generate reference solutions $X_{ref}(t_j)$ for the selected time instances t_j. This is done by using RKSM-DRE with a stopping tolerance of 10^{-10}, plus a refinement process with BDF$(4, 10^4)$ from [41]. To allow for such accurate approximations, we consider slightly smaller problem dimensions for the first two datasets, and we set $p = s = 1$ and $X_0 = 0$.

The input parameters are tailored so that the approximate solutions from different methods have relatable accuracies. In particular, RKSM-DRE is solved with an outer stopping tolerance of 10^{-6} and with BDF(3,1000) in the refinement process. The number of timesteps utilized in SPLIT-ADD4 is selected as $\ell = 500$. The expected approximation errors relative to the reference solution, measured as

$$\frac{\|X_{approx}(t) - X_{ref}(t)\|_F}{\|X_{ref}(t)\|_F},$$

are illustrated in Figure 6.6 (dataset SYM2D in the left plot, dataset NSYM3D in the right plot). The figures indicate that we compare methods having approximation

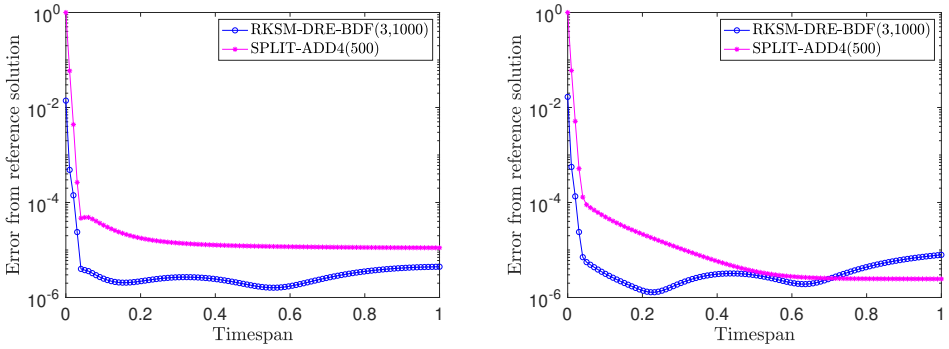

FIG. 6.6. *Expected approximation error for* RKSM-DRE *and* SPLIT-ADD4(500). *Left: Dataset* SYM2D. *Right: Dataset* NSYM3D.

errors of similar order. The performance results are contained in Table 8 for two different discretizations of SYM2D and NSYM3D.

All numbers indicate the competitiveness of RKSM-DRE in terms of storage and computational time. The memory requirements for SPLIT-ADD4 is measured as the dimension of the solution factor at the final timestep, before column compression. If the solution is required at more time instances, then these memory requirements will increase accordingly.

We also mention that we have experimented with the dynamic splitting methods introduced in [35], however the algorithms proposed by the authors[4] in [35] appeared to be better suited for small to medium size problems.

Discussion on the refinement step. In previous sections, we have stressed that the two approximation stages of the projection method are independent, and we have focused on determining an effective approximation space. Here we linger over the

[4]We thank Chiara Piazzola for providing us with her Matlab implementation of the method.

TABLE 8

Storage and computational time comparison of RKSM-DRE *and* SPLIT-ADD4(500). *Reduction phase performed with* BDF*(1,10), refinement phase with* BDF*(3,1000). In* SPLIT-ADD4 *only the approximate solution at the final time is stored.*

Data (n)	Method	# n-long Vecs	Min/Max rank	Tot CPU time (s)
SYM2D (10^4)	RKSM-DRE	8	3/6	0.6
	SPLIT-ADD4(500)	28	3/7	34.9
NSYM3D ($8 \cdot 10^3$)	RKSM-DRE	10	4/7	2.2
	SPLIT-ADD4(500)	36	3/9	37.9
SYM2D ($9 \cdot 10^4$)	RKSM-DRE	6	3/4	1.2
	SPLIT-ADD4(500)	28	3/7	330.0
NSYM3D ($2.7 \cdot 10^4$)	RKSM-DRE	10	4/8	10.1
	SPLIT-ADD4(500)	36	3/9	127.8

TABLE 9

SYM2D *(of size* 10^6*): Results with* RKSM-DRE*, using different refinement strategies. Reduction phase performed with* BDF*(1,10) and tolerance* 10^{-8}.

Refinement Method	# n-long Vecs	Soln. rank	Reduction phase(s)	Refinement phase(s)	Tot CPU time(s)
BDF(2,100)	72	55	37.7	1.1	38.8
BDF(3,1000)	72	55	37.7	9.6	47.3
BDF(4,10000)	72	55	37.7	95.5	133.2
SPLIT-ADD4(500)	72	55	37.7	1.9	39.6
SPLIT-ADD8(500)	72	55	37.7	5.1	42.8
SPLIT-ADAPT8	72	55	37.7	23.1	60.8

accuracy of the second stage, the refinement step. Exploiting the far smaller problem size of the reduced problem, it is possible to allow for a much more accurate integration phase than what was done during the iteration of the reduction step. This crucial fact is already illustrated in the time break down of Tables 4 to 6, where especially for RKSM-DRE the refinement phase employs a fraction of the overall computational time, while still allowing for a rather accurate final solution.

We next explore in more detail these advantages with RKSM-DRE on SYM2D, where the discretization was further refined to get a coefficient matrix of dimension 10^6. The dimensions of the other corresponding matrices remain as presented in Table 1. We investigate the time taken by DRE solvers with different accuracies to emphasize the advantages and flexibility of the refinement procedure. Table 9 reports the timings for a refinement step performed by three different BDF methods and three splitting methods. The 8th order adaptive splitting method (SPLIT-ADAPT8) also comes from [47] and is performed with a tolerance of 10^{-7}. We emphasize that in the refinement phase we have utilized some of the most accurate integrators available, and nevertheless the high-dimensional ($n = 10^6$) problem is approximated in less than 150 seconds for all integrators.

7. Conclusions and open problems. We have devised a rational Krylov subspace based order reduction method for solving the symmetric differential Riccati equation, providing a low-rank approximate solution matrix at selected time steps. A

single projection space is generated for all time instances, and the space is expanded until the solution is sufficiently accurate. We stress that our approach is very general, and that it could be applied to subspaces other than Krylov-based ones, as long as the spaces are nested, so that they keep growing as the iterations proceed. This methodology could then be employed for more complex settings, such as parameter dependent problems, where the involved approximation space may require the inclusion of some parameter sampling.

Like in typical model order reduction strategies, in our methodology time stepping is only performed at the reduced level, so that the integration cost is drastically lower than what one would have by applying the time stepping on the original large dimensional problem. We have derived a new stopping criterion that takes into account the different approximation behavior of the algebraic and differential portions of the problem, together with a refinement procedure that is able to improve the final approximate solution by using a high-order integrator. These enhancement strategies have also been applied to the extended Krylov subspace approach. We have analyzed the asymptotic behavior of the reduced order solution, so as to ensure that the generated approximation behaves like the sought after time-dependent solution.

Although our numerical results are promising, there are still several open issues associated with the reduced order solution of the DRE. In particular, while stability and other matrix properties associated with the solutions $X(t)$ have been thoroughly studied [10, 19, 37, 16], the analysis of corresponding properties for the approximate solution $X_m(t) = \mathcal{V}_m Y_m(t) \mathcal{V}_m^T$ for $t \in [0, t_f]$ is still a largely open problem. In [27] some interesting monotonicity properties have been shown when the polynomial Krylov subspace is used together with particular ODE solvers; a complete analysis for $X_m(t)$ in a more general setting would be desirable.

Acknowledgements. We thank Khalide Jbilou for helpful explanations on [3], and Jens Saak for assistance in using the software M.E.S.S. v.1.0.1 [41]. We are also grateful to Chiara Piazzola for making her code from [35] available and to Tony Stillfjord for pointing us towards his codes from [47]. We thank Maximilian Behr for some helpful criticism on a previous version of this manuscript. The first author acknowledges the insightful comments and kind hospitality of the group of Peter Benner at the Max Planck Institute in Magdeburg (D), during a short visit in February 2019. Furthermore, we thank two anonymous referees for their careful reading and insightful comments.

This research is supported in part by the Alma Idea grant of the Alma Mater Studiorum Università di Bologna, and by INdAM-GNCS under the 2018 Project "Metodi numerici per equazioni lineari, non lineari e matriciali con applicazioni". The second author is a member of INdAM-GNCS.

Appendix A. Krylov subspace properties. In this Appendix we review some properties of extended and rational Krylov subspaces. As in section 2 we denote $N = [C^T, Z]$.

Extended Krylov subspace. The extended Krylov subspace $\mathcal{EK}_m(A^T, N)$ takes the form discussed in section 2. The orthonormal basis $\mathcal{V}_m \in \mathbb{R}^{n \times 2m(p+q)}$ spanning the subspace is formed using the extended Arnoldi algorithm [17]. Let

$$\widetilde{\mathcal{T}}_m^T = \mathcal{V}_{m+1}^T A^T \mathcal{V}_m = \begin{bmatrix} \mathcal{T}_m^T \\ t_{m+1,m} E_{2m}^T \end{bmatrix} \in \mathbb{R}^{2(m+1)(p+q) \times 2m(p+q)}, \tag{A.1}$$

where $\mathcal{V}_{m+1} = [\mathcal{V}_m \; V_{m+1}] \in \mathbb{R}^{n \times 2(m+1)(p+q)}$ and E_{2m} is the last $2(p+q)$ columns of

$I_{2m(p+q)}$. The extended Arnoldi algorithm produces the Arnoldi-type relation

$$A^T\mathcal{V}_m = \mathcal{V}_{m+1}\widetilde{\mathcal{T}}_m^T = \mathcal{V}_m\mathcal{T}_m^T + V_{m+1}t_{m+1,m}E_{2m}^T. \tag{A.2}$$

Rational Krylov subspace. The rational Krylov subspace was originally proposed in the eigenvalue context in [39]. Its use in our context is motivated by [45] and later [43], where its effectiveness in the solution of the algebraic Riccati equation is amply discussed.

Assume that A is Hurwitz. Given $\boldsymbol{s} = \{s_1, s_2, \dots\}$, with $s_j \in \mathbb{C}^+$, the rational Krylov subspace is given by $\mathcal{RK}_m(A, N, \boldsymbol{s})$ as defined in section 2. The approximation effectiveness of this subspace depends on the choice of shifts $\boldsymbol{s}$, and this issue has been investigated in the literature; see, e.g., [36], [18]. The adaptive choice of shifts was tailored to the ARE in [32] by the inclusion of information of the term BB^T during the shift selection; see also [43] for a more detailed discussion[5]. In our numerical experiments we used this last adaptive strategy, where the approximate solution at timestep t_f is used.

The algorithm presented in [18] forms a complex basis, when the shifts are not all real. In short, when $s_j \in \mathbb{C}^+$, the original approach would be to use the shift s_j to form the next block V_j and to then let the following shift be given by $s_{j+1} = \overline{s}_j$, where $\overline{s}_j$ denotes the complex conjugate of s_j. This results in both V_j and V_{j+1} being complex. As a consequence, the reduced DRE has complex coefficient matrices, although the final resulting approximations $X_m(t)$ will be real. Standard ODE solvers do not handle complex arithmetic well, hence we implemented an all-real basis using the method introduced in [40], which works as follows. If the shift s_j is complex then the block $W_j = (A - s_jI)^{-1}V_{j-1}$ is also complex, hence we split it into its real and complex parts, that is $W_j = W_j^{(r)} + W_j^{(c)}\imath$. The block V_j is then formed by orthogonalizing $W_j^{(r)}$ with respect to all vectors in the already computed basis, after which V_{j+1} is formed by orthogonalizing $W_j^{(c)}$ with respect to all previous vectors in in the computed basis, and in V_j. This determines the same space, since $\text{span}\{W_j, \bar{W}_j\} = \text{span}\{V_j, V_{j+1}\}$. The resulting *real* basis of the rational Krylov subspace is given by $\mathcal{V}_m = [V_1, \dots, V_m] \in \mathbb{R}^{n\times m(p+q)}$. We also define the matrices $\mathcal{V}_{m+1} = [\mathcal{V}_m, V_{m+1}] \in \mathbb{R}^{n\times(m+1)(p+q)}$ and the matrix

$$\widetilde{\mathcal{H}}_m = \begin{bmatrix} \mathcal{H}_m \\ r_{m+1,m}E_m^T \end{bmatrix} \in \mathbb{R}^{(m+1)(p+q)\times m(p+q)}, \tag{A.3}$$

where $r_{m+1,m} \in \mathbb{R}^{(p+q)\times(p+q)}$ and E_m holds the last $(p+q)$ columns of $I_{m(p+q)}$. The matrix $\widetilde{\mathcal{H}}_m$ contains the orthogonalization coefficients obtained during the rational Arnoldi algorithm.

Let $\mathcal{T}_m^T = \mathcal{V}_m^T A^T \mathcal{V}_m \in \mathbb{R}^{m(p+q)\times m(p+q)}$. The rational Krylov basis satisfies the Arnoldi-type relation

$$A^T\mathcal{V}_m = \mathcal{V}_m\mathcal{T}_m^T + \widehat{V}_{m+1}G_m^T, \tag{A.4}$$

where $G_m^T = \gamma r_{m+1,m}E_m^T\mathcal{H}_m^{-1}$ and the matrix $\widehat{V}_{m+1}$ is an orthonormal matrix such that

$$\widehat{V}_{m+1}\gamma = V_{m+1}s_m - (I_n - \mathcal{V}_m\mathcal{V}_m^T)A^TV_{m+1} \tag{A.5}$$

[5]The Matlab code of the rational Krylov subspace method for ARE is available at `http://www.dm.unibo.it/~simoncin/software`

is the QR decomposition of the matrix on the right (see [18, 31]). The rational Krylov procedure requires as an extra input the (usually real) values $s_0^{(1)}, s_0^{(2)}$, which form a rough approximation of a spectral region used to compute the next shift. The reader is referred to [18, 43] for implementation details. Further, for the computation of the term $G_m^T Y_m(t)$ contained in the residual computation of RKSM-DRE, we follow an accelerated computation technique presented in [18].

Appendix B. Extended Krylov subspace based method. The extended Krylov (EKSM-DRE) subspace method for solving (1.1) is presented in Algorithm B.1.

Algorithm B.1 EKSM-DRE

Require: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times s}$, $C \in \mathbb{R}^{p\times n}$, $Z \in \mathbb{R}^{n\times q}$, tol, t_f, ℓ

(i) Perform reduced QR: $\big([C^T, Z], A^{-1}[C^T, Z]\big) = V_1\Lambda_1$

Set $\mathcal{V}_1 \equiv V_1$

for $m = 2, 3\ldots$

Compute the next basis block V_m

Set $\mathcal{V}_m = [\mathcal{V}_{m-1}, V_m]$

Update $\mathcal{T}_m$ as in [42] and $B_m = \mathcal{V}_m^T B$, $Z_m = \mathcal{V}_m^T Z$ and $C_m = C\mathcal{V}_m$

Integrate (2.2) from 0 to t_f using BDF$(1, \ell)$

Compute ρ_m using (4.3) where $\tau_m^T = t_{m+1,m} E_{2m}^T$

if $\rho_m < tol$

go to (ii)

end if

end for

(ii) Refinement: solve (2.2) with a more accurate integrator

Compute $Y_m(t_j) = \widehat{Y}_m(t_j)\widehat{Y}_m(t_j)^T$, $j = 1, \ldots, \ell$ using the truncated SVD

return $\mathcal{V}_m \in \mathbb{R}^{n\times 2m(p+q)}$ and ℓ factors $\widehat{Y}_m(t_j) \in \mathbb{R}^{2m(p+q)\times r}$, $j = 1, \ldots, \ell$.

REFERENCES

[1] *Oberwolfach model reduction benchmark collection.*, 2003.

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

[3] V. Angelova, M. Hached, and K. Jbilou, *Approximate solutions to large nonsymmetric differential Riccati problems with applications to transport theory*, Numer. Linear Algebra Appl., 27 (2020), p. e2272.

[4] A. C. Antoulas, *Approximation of large-scale dynamical systems*, vol. 6, SIAM, 2005.

[5] M. Behr, P. Benner, and J. Heiland, *On an invariance principle for the solution space of the differential Riccati equation*, Proc. Appl. Math. Mech., 18 (2018), p. e201800031.

[6] ———, *Invariant Galerkin ansatz spaces and Davison-Maki methods for the numerical solution of differential Riccati equations*, arXiv preprint arXiv:1910.13362, (2019).

[7] P. Benner, A. Cohen, M. Ohlberger, and K. Willcox, *Model reduction and approximation theory and algorithms*, SIAM, Philidelphia, 2017.

[8] P. Benner, V. Mehrmann, and D. Sorensen, *Dimension Reduction of Large-Scale Systems*, Springer-Verlag, Berlin/Heidelberg, Germany, 2005.

[9] D. A. Bini, B. Lannazzo, and B. Meini, *Numerical solution of algebraic Riccati equations*, vol. 9, SIAM, 2012.

[10] R. Bitmead, M. Gevers, I. Petersen, and J. Kaye, *Monotonicity and stabilizability-properties of solutions of the Riccati difference equation: Propositions, lemmas, theorems, fallacious conjectures and counterexamples*, Systems & Control Letters, 5 (1985), pp. 309–315.

[11] S. Blanes, *High order structure preserving explicit methods for solving linear-quadratic optimal control problems and differential games*, Numerical Algorithms, 69 (2015), pp. 271—-290.
[12] C. Choi and A. Laub, *Efficient matrix-valued algorithms for solving stiff Riccati differential equations*, IEEE Trans. Automat. Control, 35 (1990), p. 770–776.
[13] M. J. Corless and A. E. Frazho, *Linear systems and control - an operator perspective*, Marcel Dekker, New York Basel, 2003.
[14] E. Davison and M. Maki, *The numerical solution of the matrix Riccati differential equation*, IEEE Trans Autom Control, (1973), pp. 71–73.
[15] L. Dieci, *Numerical integration of the differential Riccati equation and some related issues*, SIAM J. Numer. Anal., 29 (1992), pp. 781–815.
[16] L. Dieci and T. Eirola, *Preserving monotonicity in the numerical solution of Riccati differential equations*, Numer Math, 74 (1996), pp. 35–47.
[17] V. Druskin and L. Knizhnerman, *Extended Krylov subspaces: approximation of the matrix square root and related functions*, SIAM J Matrix Anal Appl, 19 (1998), pp. 755–771.
[18] V. Druskin and V. Simoncini, *Adaptive rational Krylov subspaces for large-scale dynamical systems*, Systems & Control Letters, 60 (2011), pp. 546–560.
[19] M. Gevers, R. Bitmead, I. Petersen, and J. Kaye, *When is the solution of the Riccati equation stabilizing at every instant?*, in Frequency Domain and State Space Methods for Linear Systems, North-Holland, 1986, pp. 531–540.
[20] L. Grasedyck, *Existence and computation of low Kronecker-rank approximations for large linear systems of tensor product structure*, Computing., 72 (2004), pp. 247–265.
[21] ———, *Existence of a low rank or $\mathcal{H}$-matrix approximant to the solution of a Sylvester equation*, Numer. Linear Algebra Appl., 11 (2004), pp. 371–389.
[22] N. Guglielmi and V. Simoncini, *On the existence and approximation of a dissipating feedback*, arXiv preprint arXiv:1811.00069, (2019).
[23] Y. Guldogan, M. Hached, K. Jbilou, and M. Kurulaya, *Low rank approximate solutions to large-scale differential matrix Riccati equations*, Applicationes Mathematicae, 45 (2018), pp. 233–254.
[24] M. Heyouni and K. Jbilou, *An extended block arnoldi algorithm for large-scale solutions of the continuoustimealgebraic riccati equation*, Electron Trans Numer Anal, 33 (2009), pp. 53–62.
[25] I. M. Jaimoukha and E. M. Kasenally, *Krylov subspace methods for solving large Lyapunov equations*, SIAM J. Numer. Anal., 31 (1994), pp. 227–251.
[26] M. Konstantinov and G. Pelova, *Sensitivity of the solutions to differential matrix Riccati equations*, IEEE Trans. Autom. Control, 36 (1991), pp. 213–215.
[27] A. Koskela and H. Mena, *Analysis of Krylov subspace approximation to large scale differential Riccati equations*, arXiv preprint arXiv:1705.07507, (2018).
[28] V. Kucera, *A review of the matrix Riccati equation*, Kibernetika, 9 (1973), pp. 42–61.
[29] H. Kwakernaak and R. Sivan, *Linear optimal control systems*, vol. 1, Wiley-interscience New York, 1972.
[30] N. Lang, *Numerical methods for large-scale linear time-varying control systems and related differential matrix equations*, PhD thesis, Technische Universitaet Chemnitz, 2017.
[31] Y. Lin and V. Simoncini, *Minimal residual methods for large scale Lyapunov equations*, Applied Num Math, 72 (2013), pp. 52–71.
[32] ———, *A new subspace iteration method for the algebraic riccati equation*, Numerical Linear Algebra w/Appl., 22 (2015), pp. 26–47.
[33] The MathWorks, *MATLAB 7*, r2013b ed., 2013.
[34] H. Mena, *Numerical Solution of Differential Riccati Equations Arising in Optimal Control Problems for Parabolic Partial Differential Equations*, PhD thesis, Escuela Politécnica Nacional, 2007.
[35] H. Mena, A. Ostermann, L.-M. Pfurtscheller, and C. Piazzola, *Numerical low-rank approximation of matrix dierential equations*, J. Comput. Appl. Math., 340 (2018), pp. 602–614.
[36] T. Penzl, *A cyclic low-rank smith method for large sparse lyapunov equations*, SIAM J Sci Comput, 21 (2000), pp. 1401–1418.
[37] M.-A. Poubelle, R. Bitmead, and M. Gevers, *Fake algebraic Riccati techniques and stability*, IEEE Trans. Autom. Control, 33 (1988), pp. 379–381.
[38] W. Reid, *Riccati differential equations*, Academic Press, New York, 1972.
[39] A. Ruhe, *Rational Krylov sequence methods for eigenvalue computation*, Lin. Alg. Appl., 58 (1984), pp. 391–405.
[40] ———, *The rational Krylov algorithm for nonsymmetric eigenvalue problems. III:Complex shifts for real matrices*, BIT, 34 (1994), pp. 165–176.

[41] J. Saak, M. Köhler, and P. Benner, *M-m.e.s.s.-1.0.1 – the matrix equations sparse solvers library*. DOI:10.5281/zenodo.50575, Apr. 2016. see also:www.mpi-magdeburg.mpg.de/projects/mess.
[42] V. Simoncini, *A new iterative method for solving large-scale lyapunov matrix equations*, SIAM J Sci Comput, 29 (2007), pp. 1268–1288.
[43] ———, *Analysis of the rational Krylov subspace projection method for large-scale algebraic Riccati equations*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 1655–1674.
[44] ———, *Computational methods for linear matrix equations*, SIAM Rev, 58 (2016), pp. 377–441.
[45] V. Simoncini, D. B. Szyld, and M. Monsalve, *On two numerical methods for the solution of large-scale algebraic Riccati equations*, IMA Journal of Numerical Analysis, 34 (2014), pp. 904–920.
[46] T. Stillfjord, *Low-rank second-order splitting of large-scale differential Riccati equations*, IEEE Trans Automat Control, 60 (2015), pp. 2791–2796.
[47] ———, *Adaptive high-order splitting schemes for large-scale differential Riccati equations*, Numer Algor, 78 (2018), pp. 1129–1151.
[48] ———, *Singular value decay of operator-valued differential Lyapunov and Riccati equations*, SIAM J. Control Optim., 56 (2018), pp. 3598–3618.