A new subspace iteration method for the algebraic Riccati equation

Yiding Lin\textsuperscript{1,2} and Valeria Simoncini\textsuperscript{2,*}\$\dagger$

\textsuperscript{1}School of Mathematical Sciences, Xiamen University, Xiamen, China
\textsuperscript{2}Dipartimento di Matematica, Università di Bologna, Bologna, Italy

SUMMARY

We consider the numerical solution of the continuous algebraic Riccati equation $A^*X + XA - XFX + G = 0$, with $F = F^*$, $G = G^*$ of low rank and $A$ large and sparse. We develop an algorithm for the low-rank approximation of $X$ by means of an invariant subspace iteration on a function of the associated Hamiltonian matrix. We show that the sought-after approximation can be obtained by a low-rank update, in the style of the well known Alternating Direction Implicit (ADI) iteration for the linear equation, from which the new method inherits many algebraic properties. Moreover, we establish new insightful matrix relations with emerging projection-type methods, which will help increase our understanding of this latter class of solution strategies. Copyright © 2014 John Wiley & Sons, Ltd.

Received 10 July 2013; Revised 31 March 2014; Accepted 31 March 2014

KEY WORDS: Riccati equation; ADI; rational Krylov subspace; invariant subspace iteration

1. INTRODUCTION

We are interested in the numerical solution of the continuous algebraic Riccati equation

$$A^*X + XA - XFX + G = 0, \quad F = F^*, \quad G = G^*,$$

(1)

where $A \in \mathbb{R}^{n \times n}$ has large dimensions, $F, G \in \mathbb{R}^{n \times n}$ have low rank, and $X$ is the unknown matrix to be approximated$^\dagger$. Here and in the following, $M^*$ denotes the conjugate transpose of the matrix $M$. We shall assume that $A$ is stable; that is, its eigenvalues all have strictly negative real part. The quadratic matrix equation in (1) has a dominant role in the solution and analysis of optimal control problems associated with dynamical systems, and it has attracted the interest of many researchers both for its elegance and its timeliness in applied field; we refer the reader to, for example, [1–6].

A matrix $X$ solution to (1) is such that the columns of the matrix

$$\begin{bmatrix} I_n \\ X \end{bmatrix},$$

(2)

where $I_n \in \mathbb{R}^{n \times n}$ is the identity matrix, generate an invariant subspace of the Hamiltonian matrix $[1]

$$\mathcal{H} = \begin{bmatrix} A & -F \\ -G & -A^* \end{bmatrix}.$$
In particular, we assume that the eigenvalues of $\mathcal{H}$ satisfy
\[
\Re(\lambda_1) \leq \Re(\lambda_2) \leq \ldots \leq \Re(\lambda_n) < 0 < \Re(\lambda_{n+1}) \leq \Re(\lambda_{n+2}) \leq \ldots \leq \Re(\lambda_{2n}),
\]
so that, in particular, no purely imaginary eigenvalues arise. We look for an approximation to the extremal solution $X_+$ of (1), associated with all eigenvalues of $\mathcal{H}$ with negative real part [1]. Such solution is called a stabilizing solution, being such that the matrix $A - FX_+$ is stable.

Many numerical procedures have been explored for solving the quadratic matrix equation (1); see, for example, [7] for a thorough survey; however, few can address the case when $A$ has large dimensions [7–9]. In this case, usually a symmetric low-rank approximation matrix is sought, in the form of the product of two matrices, such as $X = UU^*$, with $U$ having few columns. Such approach avoids storing the full $n \times n$ matrix, which would be prohibitive for large $n$. Among these strategies is the class of exact and inexact Newton methods: Newton’s iteration applied to (1) can be conveniently rewritten so as to update the low-rank approximate solution and its rank at each iteration. The approach requires the (in)exact solution of a linear matrix equation at each iteration [10, 11], which is performed by means of iterative methods, such as ADI or projection methods; we refer to [12] for a very recent survey. For large matrices stemming from sufficiently regular differential control problems, Newton strategies based on hierarchical matrices and nonlinear multigrid methods have also shown to be effective [13, 14].

Another class of methods has recently emerged as a competitive alternative to nonlinear (Newton) solvers: the general approach consists in extending well-established projection-type methods to the quadratic case, with no significant modifications [15–18]. Although projection methods have gained increasing popularity in the linear case, with thoroughly analyzed theoretical properties [19], their exploration in the quadratic case has only recently started, and much of their properties remains to be uncovered.

A less exercised class of methods is given by the doubling algorithm, which was recently explored in the Riccati context in [20]; however, its memory and computational requirements have not been fully analyzed for large nonsymmetric problems.

All these approaches attack (1) as a quadratic equation. We take a different viewpoint, which consists in approximating $X$ in the second block of the matrix in (2), whose columns span an invariant subspace of $\mathcal{H}$. Such strategy is quite popular in the small-scale case, when an explicit possibly structure-preserving eigendecomposition may be determined; see, for example, [21–24] and the extensive treatment in [7]. A possible adaptation to the large-scale setting was recently proposed in [25], where an approximation of the form $X_k = ZWY^*$ was derived, stemming from the approximation of selected stable eigenpairs of $\mathcal{H}$.

The aim of this paper is to develop an algorithm for the approximation of $X$ by means of an invariant subspace iteration on a function of the matrix $\mathcal{H}$ [23]. Typically, subspace iteration methods are based on $\mathcal{H}$. Here, we consider a subspace iteration method with a transformed matrix obtained using a Cayley transformation. For $\alpha$, so that $\mathcal{H} + \alpha I$ is nonsingular, the Cayley transformation is given by
\[
S(\alpha) = (\mathcal{H} + \alpha I)^{-1}(\mathcal{H} - \overline{\alpha} I),
\]
and it is usually employed in the Riccati equation context for accelerating the computation of the Schur form by means of a QR iteration [7, p.133], [26]. As a consequence of the transformation, the property (3) transforms into $|\sigma_1| \geq |\sigma_2| \geq \ldots \geq |\sigma_n| > 1 > |\sigma_{n+1}| \geq |\sigma_{n+2}| \geq \ldots \geq |\sigma_{2n}|$, for the eigenvalues $\sigma_j$ of $S(\alpha)$, and the columns of $[I_n; X_+]$ span the invariant subspace of $S(\alpha)$ associated with the $n$ eigenvalues largest in modulus. The transformation thus provides a more natural setting for a subspace iteration. We will show that whenever $F$ and $G$ are positive semidefinite and have low rank, such iteration can be written in terms of a fixed-point recurrence in the low-rank approximation matrix $X_k$, and a low-rank update can be performed. To the best of our knowledge, this iteration appears to be new, in particular with the simplification obtained in the low-rank case. From our derivation, it readily follows that this novel approach coincides with the ADI method in the linear case, namely, whenever $F = 0$, thus showing that ADI may be bonded to a subspace iteration method. The proposed method depends on parameters that can be deduced from known
properties of the problem or estimated a priori. In that respect, the method inherits the properties of its linear counterpart ADI.

We will also derive relations between the new subspace iteration and projection methods for the Riccati equation that use the rational Krylov subspace method (RKSM). These results provide new insights in the understanding of the convergence properties of RKSM when directly applied to (1).

We emphasize that our developments provide a new and insightful matrix framework that, on the one hand, will allow us to bridge the gap between methods for two closely related linear and quadratic equations and, on the other hand, will be a first step ahead in the understanding of projection methods for (1), not based on the Newton method for large-scale problems.

The following notation will be used throughout the manuscript. \( F \) is a Hermitian and positive (semi-)definite matrix; \( \mathbb{F} \) will denote the set of eigenvalues of \( A \), and its spectral radius. The Euclidean norm will be used for vectors, and the associated induced norm for matrices, denoted by \( \| \cdot \| \), together with the Frobenius norm, denoted by \( \| \cdot \|_F \). The notation \( \text{diag}(d) \) and \( \text{blkdiag}(D_1, D_2) \) will be used to denote a diagonal matrix with the entries of the vector \( d \) on the diagonal and a block diagonal matrix with block diagonal entries \( D_1, D_2 \), respectively. We will use MATLAB ([27]) notation for matrices and their subblocks whenever possible.

2. A SUBSPACE ITERATION WITH CAYLEY TRANSFORMATION

Given \( X_0 \in \mathbb{R}^{n \times n} \) and the parameters \( \alpha_k, k = 1, 2, \ldots \) with \( \Re(\alpha_k) > 0 \), such that \( (\mathcal{H} + \alpha_k I) \) is invertible\(^3\), we consider the following iteration to compute a sequence of approximations \( X_1, X_2, \ldots, X_k, \ldots \) to \( X^+ \).

For \( k = 1, 2, \ldots \)

Compute

\[
\begin{bmatrix}
M_k \\
N_k
\end{bmatrix} := S(\alpha_k) \begin{bmatrix}
I \\
X_{k-1}
\end{bmatrix} \quad \text{(with } S(\alpha_k) \text{ as in (4)})
\]

\( X_k := N_k M_k^{-1} \)

end

The iteration breaks down if \( M_k \) is singular at some iteration. In the following text, we shall find a sufficient condition that ensures all \( M_k \)'s are nonsingular; we will also show that this condition can be easily satisfied when, for instance, \( X_0 \) is chosen to be the zero matrix and \( F, G \) are positive semidefinite.

By eliminating the intermediate matrices \( M_k, N_k \), the aforementioned recursion can be rewritten as a fixed-point iteration with the recurrence matrix \( X_k \), from which we will derive the new low-rank recurrence. To this end, we need to express the statement in (5) in a more explicit way. For any \( \alpha \in \mathbb{C} \) such that \( \mathcal{H} + \alpha I \) is nonsingular, let

\[
\mathcal{H} + \alpha I = \begin{bmatrix}
A + \alpha I & -F \\
-G & -A^* + \alpha I
\end{bmatrix}.
\]

If \( A + \alpha I \) is nonsingular, then the Schur complement \( S_1(\alpha) := (-A^* + \alpha I) - G(A + \alpha I)^{-1} F \) is also nonsingular. Analogously, \( -A^* + \alpha I \) nonsingular implies \( S_2(\alpha) := (A + \alpha I) - F(-A^* + \alpha I)^{-1} G \) nonsingular. To simplify the notation, we shall often omit the dependence of \( S_1, S_2 \) on \( \alpha \). It can be readily verified that

\[
S_2^{-1} F(-A^* + \alpha I)^{-1} = (A + \alpha I)^{-1} FS_1^{-1},
\]

\[
S_1^{-1} G(A + \alpha I)^{-1} = (-A^* + \alpha I)^{-1} GS_2^{-1}.
\]

\(^3\)This condition will be relaxed in the sequel.
For later use, we notice that we can write
\[
(\mathcal{H} + \alpha I)^{-1} = \begin{bmatrix}
S_2^{-1} & S_2^{-1} F(-A^* + \alpha I)^{-1} \\
S_1^{-1} G(A + \alpha I)^{-1} & S_1^{-1}
\end{bmatrix}
= \begin{bmatrix}
S_2^{-1} \quad (A + \alpha I)^{-1} F S_1^{-1} \\
(-A^* + \alpha I)^{-1} G S_2^{-1} & S_1^{-1}
\end{bmatrix}.
\]

At the \(k\)th iteration, let \(\alpha_k = \alpha_k + \tau b_k\), with \(\alpha_k, b_k \in \mathbb{R}\); this notation will be used throughout the paper. In particular, from now on, we shall assume that \(\alpha_k > 0\) for all \(k\). Because \(S(\alpha_k) = I - 2\alpha_k (\mathcal{H} + \alpha_k I)^{-1}\), we can write the product in (5) as follows
\[
S(\alpha_k) \begin{bmatrix} I \\ X_{k-1} \end{bmatrix} = \left( I - 2\alpha_k (\mathcal{H} + \alpha_k I)^{-1} \right) \begin{bmatrix} I \\ X_{k-1} \end{bmatrix} = \begin{bmatrix} I - 2\alpha_k S_2^{-1} - 2\alpha_k S_2^{-1} F (-A^* + \alpha_k I)^{-1} X_{k-1} \\
-2\alpha_k S_1^{-1} G (A + \alpha_k I)^{-1} + (I - 2\alpha_k S_1^{-1}) X_{k-1} \end{bmatrix},
\]
so that the next iterate can be written by means of a fixed-point iteration:
\[
X_k = \left[ -2\alpha_k S_1^{-1} G (A + \alpha_k I)^{-1} + (I - 2\alpha_k S_1^{-1}) X_{k-1} \right]^{-1} \left[ I - 2\alpha_k S_2^{-1} - 2\alpha_k S_2^{-1} F (-A^* + \alpha_k I)^{-1} X_{k-1} \right]^{-1}. \tag{8}
\]

Notice that because of (6) and (7), it would be possible to write the iteration in four possible different but mathematically equivalent ways.

Remark 2.1
If the nonlinear term vanishes, that is if \(F = 0\), then the Riccati equation becomes the (linear) Lyapunov equation \(G + A^* X + X A = 0\). In this case, the fixed-point iteration in (8) coincides with the ADI recursion for solving the Lyapunov equation; see, for example, [28, formula (4.1)]. We will return to this correspondence in later sections.

We next analyze the existence of the approximate solution at each step \(k\) of the subspace iteration, with \(X_k\) obtained as in (8). We shall also describe some properties that will allow us to define a factorized low-rank approximate solution in the next section.

Proposition 2.2
Assume that \(F, G \succeq 0\) and that \(A^* - \alpha_k I\) is nonsingular. In (5), assume that for some \(k > 0\), it holds that \(X_{k-1} \succeq 0\). Then,

(i) The matrix \(M_k\) is nonsingular.

(ii) The matrix \(X_k\) is well defined and satisfies \(X_k = X_k^*\).

Proof
From the definition of \(M_k\), we have
\[
M_k = S_2^{-1}(S_2 - 2\alpha_k I - 2\alpha_k F(-A^* + \alpha_k I)^{-1} X_{k-1})
= S_2^{-1}(A - \tilde{\alpha_k} I - F(-A^* + \alpha_k I)^{-1} G - 2\alpha_k F(-A^* + \alpha I)^{-1} X_{k-1})
= S_2^{-1}(A - \tilde{\alpha_k} I)[I - (A - \tilde{\alpha_k} I)^{-1} F(-A^* + \alpha_k I)^{-1} G + 2\alpha_k X_{k-1}]
= S_2^{-1}(A - \tilde{\alpha_k} I)[I + (-A + \tilde{\alpha_k} I)^{-1} F(-A^* + \alpha_k I)^{-1} G + 2\alpha_k X_{k-1}]. \tag{9}
\]

Then, we observe that the nonzero eigenvalues of \([(-A + \bar{a}_k I)^{-1} F (-A^* + \alpha_k I)^{-1})(G + 2\alpha X_{k-1})\]
are all real and positive, because the matrix is the product of two Hermitian and non-negative definite matrices. Therefore, the quantity in brackets in (9) is nonsingular, and the first result follows.

Because \(X_k = N_k M_k^{-1}\), the first result ensures that \(X_k\) is well defined. We only need to show that it is Hermitian, namely \(X_k = X_k^*\), which is equivalent to showing that \(M_k^* N_k = N_k^* M_k\). Let us write \(S = [M_1, M_2; N_1, N_2]\) with

\[
\begin{align*}
N_1 &:= -2\alpha_k (-A^* + \alpha_k I)^{-1} G S_2^{-1}, \\
N_2 &:= I - 2\alpha_k S_1^{-1}, \\
M_1 &:= I - 2\alpha_k S_2^{-1}, \\
M_2 &:= -2\alpha_k (A + \alpha_k I)^{-1} F S_1^{-1}.
\end{align*}
\]

We recall that because \(H\) is Hamiltonian, \(S\) is symplectic, so that from the definition of symplectic matrix, it follows that [7, p.24]

\[
M_1^* N_1 = N_1^* M_1, \quad M_2^* N_2 = N_2^* M_2, \quad M_2^* N_1 - N_2^* M_1 = -I. \tag{11}
\]

Moreover, \(X_k = N_k M_k^{-1} = (N_1 + N_2 X_{k-1})(M_1 + M_2 X_{k-1})^{-1}\). Together with \(X_{k-1} = X_{k-1}^*\), relations (11) show that \(M_k^* N_k = N_k^* M_k\), so that \(X_k = X_k^*\). \(\square\)

We note that the second result does not explicitly require that \(F\) and \(G\) be positive semidefinite. Moreover, the hypothesis that \(A^* - \alpha_k I\) is nonsingular is always satisfied for \(A\) real and stable, and \(\Re(\alpha_k) > 0\).

Next proposition derives a more convenient form for the iterate \(X_k\), from which we can deduce that \(X_k\) is positive semidefinite for any \(k > 0\), if \(X_0\) is. More precisely, a preliminary low-rank factored form is obtained, which shows that the dense iterates \(X_k\) do not have to be stored, but only their memory saving factors should be saved for the next iteration.

**Proposition 2.3**

Assume \(F^* = F\), \(G = C^* C\) and that for some \(k > 0\), \(X_{k-1}\) can be written as \(X_{k-1} = U_{k-1} T_{k-1}^{-1} U_{k-1}^*\) with \(T_{k-1}\) Hermitian and nonsingular. Suppose \(X_k\) is well defined and let

\[
T_k = \begin{bmatrix} T_{k-1} & 0 \\ 0 & 2\alpha_k I \end{bmatrix} + 2\alpha_k \begin{bmatrix} U_{k-1}^* \\ C \end{bmatrix} (-A + \bar{a}_k I)^{-1} F (-A^* + \alpha_k I)^{-1} \begin{bmatrix} U_{k-1} & C^* \end{bmatrix}. \tag{12}
\]

If \(T_k\) is nonsingular, then \(X_k = U_k T_k^{-1} U_k^*\), where

\[
U_k = \begin{bmatrix} (-A^* + \alpha_k I)^{-1} (-A^* - \bar{a}_k I) U_{k-1} & -2\alpha_k (-A^* + \alpha_k I)^{-1} C^* \end{bmatrix}. \tag{13}
\]

**Proof**
The proof is postponed to the Appendix. \(\square\)

**Corollary 2.4**

Assume that \(F \geq 0\) and \(G \geq 0\). If for some \(k > 0\), \(X_{k-1} \geq 0\), then \(X_k \geq 0\).

**Proof**
The assumption \(X_{k-1} \geq 0\) ensures that \(X_{k-1}\) can be written as \(X_{k-1} = U_{k-1} T_{k-1}^{-1} U_{k-1}^*\) with \(T_{k-1} \geq 0\). Proposition 2.3 thus shows that for \(G \geq 0\), \(X_k\) can be written as \(X_k = U_k T_k^{-1} U_k^*\) with \(T_k\) defined in (12). If in addition \(F \geq 0\), then \(T_k \geq 0\), which implies \(X_k \geq 0\). \(\square\)

We conclude this section by showing that the hypothesis that \(X_0 \geq 0\) is sufficient for all subsequent iterates to be well defined. Thanks to Corollary 2.4, all subsequent iterates will also be non-negative definite. In particular, the proof of Corollary 2.4 ensures that all matrices \(T_k\) are symmetric and positive definite.
Proposition 2.4
Suppose $F \succeq 0$ and $G \succeq 0$. Assume that all $\alpha_k$‘s have positive real part. If $X_0 \succeq 0$, then all matrices $M_k$, $k = 1, 2, \ldots$ produced by (5) are nonsingular.

Proof
Proposition 2.2 states that if $X_{k-1} \succeq 0$, then $M_k$ is nonsingular and $X_k$ is well defined. Corollary 2.4 states that if $X_{k-1} \succeq 0$, then $X_k \succeq 0$. Therefore, choosing $X_0 \succeq 0$, by induction, it follows that $M_k$, $k = 1, 2, \ldots$ produced by (5) will be nonsingular. \hfill \Box

3. FACTORED LOW-RANK SUBSPACE ITERATION FOR LARGE-SCALE DATA

Whenever the problem dimension is very large, the approximate solution matrix as expressed in (8) cannot be explicitly stored. However, if both $F$ and $G$ are low rank, Proposition 2.3 can be used to derive and update at each iteration a (low-rank) factorization of $X_k$ that can be handled more cheaply. Assuming $G = C^*C$ is low rank, the resulting recursion is given in Algorithm 1. We stress that any initial approximation $X_0$ written as $X_0 = U_0 T_0^{-1} U_0^*$ with $T_0 > 0$ can be used. Moreover, we notice that the algorithm will not break down if $\alpha_k$ is an eigenvalue of $\mathcal{H}$, as long as $-A^* + \alpha_k I$ is nonsingular, the latter being the only hypothesis required in practice; stability of $A$ and positivity of $\Re(\alpha_k)$ ensure that such a condition is verified.

Algorithm 1 LRSI: low-rank subspace iteration. Generic implementation.

1: INPUT: Given $A \in \mathbb{R}^{n \times n}$ stable, $U_0$ and $T_0 > 0$ such that $X_0 = U_0 T_0^{-1} U_0^* \succeq 0$ (e.g., $X_0 = 0$), and $\alpha_k$, $k = 1, 2, \ldots$, with $\alpha_k = \Re(\alpha_k) > 0$

2: for $k = 1, 2, 3, \ldots$ do

3: \hspace{1em} $U_k := \begin{bmatrix} (A^* + \alpha_k I)^{-1}(A^* - \bar{\alpha}_k I) & 0 \\ 0 & 2\alpha_k \end{bmatrix}^{-1} \begin{bmatrix} U_{k-1}^* \end{bmatrix}$

4: \hspace{1em} $T_k := \begin{bmatrix} T_{k-1} & 0 \\ 0 & 2\alpha_k \end{bmatrix}$ \hspace{1em} $+ \begin{bmatrix} U_{k-1}^* \\ 0 \end{bmatrix}$ \hspace{1em} $(-A + \bar{\alpha}_k I)^{-1} F (-A^* + \alpha_k I)^{-1} \begin{bmatrix} U_{k-1} & C^* \end{bmatrix}$

5: end for

6: OUTPUT: $U_k, T_k$ such that $X_k = U_k T_k^{-1} U_k^* \approx X_+$

A more effective low-rank recursion is obtained by noticing that the term

$$
\begin{bmatrix} U_{k-1}^* \\ C \end{bmatrix} (-A + \bar{\alpha}_k I)^{-1} F (-A^* + \alpha_k I)^{-1} \begin{bmatrix} U_{k-1} & C^* \end{bmatrix}
$$

can be computed without explicitly computing the $n \times n$ inner matrix. This operation is particularly cheap if $F = BB^*$ with $B$ having low column rank. A closer look at the recurrence matrix

$$
U_k := \begin{bmatrix} (A^* + \alpha_k I)^{-1}(A^* - \bar{\alpha}_k I) & 0 \\ 0 & 2\alpha_k \end{bmatrix}^{-1} \begin{bmatrix} U_{k-1}^* \end{bmatrix}
$$

reveals that, except for an innocuous scaling factor, this is precisely the same iteration matrix obtained when using Low-Rank ADI (LR-ADI) [28, formulas (4.6) and (4.7)], [29]. In particular, when the nonlinear term is zero ($F = 0$), the recurrence in Algorithm 1 corresponds to the LR-ADI iteration. As a consequence, we obtain that

$$
\text{Range}(U_k) = \text{Range}([(-A^* + \alpha_1 I)^{-1} C^*, \ldots, (-A^* + \alpha_k I)^{-1} C^*])
$$

namely the generated space is the rational Krylov subspace with poles $\alpha_1, \ldots, \alpha_k$, $k \geq 1$ [28, proposition 7.3]. From $X_k = U_k T_k^{-1} U_k^*$, it thus follows that a different basis for the Rational Krylov subspace could be selected to equivalently define $X_k$. More precisely, letting $Q_k$ be any nonsingular matrix of size equal to the number of columns of $U_k$, then the columns of $Q_k U_k$ are still a basis for...
the space, and \( X_k = (U_k Q_k)(Q_k^{-1} T_k^{-1} Q_k^{-*})(U_k Q_k)^* \). This property is particularly important, as the matrices \( U_k \) in (14) are not efficiently nested: the number of system solves per iteration increases with the number of iterations. In the next section, we derive a more economical low-rank recurrence.

### 3.1. An incremental low-rank recursion

We next express the approximate solution in terms of a nested basis spanning the rational Krylov subspace, which only requires one system solve with \(-A^* + \alpha_k I\) at iteration \( k \) to expand the space. This is based on the observation that the given basis is nested for \( X_0 = 0 \).

To simplify the presentation here and in the following, we shall work with the corresponding rational function scalar bases. When employing matrices, the symbol \( \lambda \) should be replaced by \(-A^*\), while the matrix \( C^* \) should end each term: \( 1/(\lambda + \alpha_k) \) should therefore read \( (-A^* + \alpha_k I)^{-1} C^* \).

With this notation, we are going to employ the following basis:

\[
V_k := \begin{bmatrix}
-2a_1 \frac{\lambda - \bar{\alpha}_1}{\lambda + \bar{\alpha}_1}, & -2a_2 \frac{\lambda - \bar{\alpha}_1}{\lambda + \bar{\alpha}_2}, & \ldots & -2a_{k-1} \frac{\lambda - \bar{\alpha}_1}{\lambda + \bar{\alpha}_{k-1}}, & \frac{k-2}{\lambda} \frac{\lambda - \bar{\alpha}_i}{\lambda + \bar{\alpha}_i} \frac{k-1}{\lambda} \frac{\lambda - \bar{\alpha}_i}{\lambda + \bar{\alpha}_i} & \frac{k}{\lambda} \frac{\lambda - \bar{\alpha}_i}{\lambda + \bar{\alpha}_i} & \frac{k+1}{\lambda} \frac{\lambda - \bar{\alpha}_i}{\lambda + \bar{\alpha}_i} & \cdots & \frac{k}{\lambda} \frac{\lambda - \bar{\alpha}_i}{\lambda + \bar{\alpha}_i} & \frac{k+1}{\lambda} \frac{\lambda - \bar{\alpha}_i}{\lambda + \bar{\alpha}_i} & \frac{k+2}{\lambda} \frac{\lambda - \bar{\alpha}_i}{\lambda + \bar{\alpha}_i} & \frac{k+3}{\lambda} \frac{\lambda - \bar{\alpha}_i}{\lambda + \bar{\alpha}_i} \end{bmatrix}.
\]  

(15)

**Lemma 3.1**

Let \( V_k \) be the matrix associated with (15). If, for some \( k > 0 \), it holds that \( X_{k-1} = V_{k-1}^{-1} T_{k-1}^{-1} V_{k-1}^{-*} \), then there exists a nonsingular matrix \( Q_k \) such that \[ [-2a_k(-A^* + \alpha_k I)^{-1} V_{k-1}, -2a_k(-A^* + \alpha_k I)^{-1} C^*] Q_k = V_k \] and a nonsingular \( T_k \) such that \( X_k = V_k T_k^{-1} V_k^* \).

**Proof**

From the recursion, we obtain \( X_k = \tilde{U}_k \tilde{T}_k^{-1} \tilde{U}_k^* \), where

\[
\tilde{U}_k := \begin{bmatrix}
(-A^* + \alpha_k I)^{-1}(-A^* - \bar{\alpha}_k I) V_{k-1}, & -2a_k(-A^* + \alpha_k I)^{-1} C^*
\end{bmatrix} = \begin{bmatrix}
-2a_k(-A^* + \alpha_k I)^{-1} V_{k-1}, & -2a_k(-A^* + \alpha_k I)^{-1} C^* + [ V_{k-1}, 0 ]
\end{bmatrix},
\]

\[
\tilde{T}_k := \begin{bmatrix}
T_{k-1} & 0 \\
0 & 2a_k I
\end{bmatrix} + 2a_k \begin{bmatrix}
V_{k-1} \ C^*
\end{bmatrix} (-A + \bar{\alpha}_k I)^{-1} F(-A^* + \alpha_k I)^{-1} [ V_{k-1} C^* ]
\]

(16)

where \( Z^* = [-2a_k(-A^* + \alpha_k I)^{-1} V_{k-1}, -2a_k(-A^* + \alpha_k I)^{-1} C^*] \). Because \( \text{Range}(Z^*) = \text{Range}(V_k) \), there exists \( Q_k \) such that \( Z^* = V_k Q_k^{-1} \). Therefore, \( \tilde{U}_k = V_k P_k \), where \( P_k := Q_k^{-1} + \begin{bmatrix} I \\ 0 \end{bmatrix} \).

Then, \( X_k = \tilde{U}_k \tilde{T}_k^{-1} \tilde{U}_k^* = V_k P_k \tilde{T}_k^{-1} \tilde{P}_k^* V_k^* = V_k \left( P_k^{-*} \tilde{T}_k P_k^{-1} \right)^{-1} V_k^* \). After defining

\[
T_k := P_k^* \tilde{T}_k P_k^{-1},
\]

(17)

we obtain \( X_k = V_k T_k^{-1} V_k^* \).

\[ \square \]

**Lemma 3.1** shows that if we can find \( Q_k \) explicitly, then we can update \( X_k = V_k T_k^{-1} V_k^* \) from \( X_{k-1} = V_{k-1} T_{k-1}^{-1} V_{k-1}^* \).

In the following, we shall make repeated use of the following simple relation, which holds for any (not necessarily distinct) \( \alpha_i, \alpha_j \):

\[
\frac{\alpha_i - \alpha_j}{(\lambda + \alpha_i)(\lambda + \alpha_j)} + \frac{1}{\lambda + \alpha_i} = \frac{1}{\lambda + \alpha_j}.
\]
Proposition 3.2
Assume the hypotheses and notation of Lemma 3.1 hold. Then, for any \( k > 0 \), \( X_k = V_k T_k^{-1} V_k^* \) with \( T_k \) as defined in (17) and

\[
V_k = [-2a_1(-A^* + \alpha_1 I)^{-1} C^*, -2a_2(-A^* + \alpha_2 I)^{-1} \varphi_1(A) C*, \ldots, -2a_k(-A^* + \alpha_k I)^{-1} \prod_{i=1}^{k-1} \varphi_i(A) C*],
\]

with \( \varphi_i(\lambda) = (\lambda - \tilde{\alpha}_i)/(\lambda + \alpha_i) \).

Proof
We prove the assertion for \( C^* \) having one column. For more columns, the same result can be written by expanding the matrices \( Q_k \) and \( P_k \) defined in the succeeding text using Kronecker products (Algorithm 2). Let

\[
V_k := \begin{bmatrix}
-2a_1/\lambda + \alpha_1, & -2a_2/\lambda + \alpha_2 & \cdots & -2a_k/\lambda + \alpha_k \\
\end{bmatrix},
\]

and

\[
\hat{U}_k = \begin{bmatrix}
(1 + \lambda/\lambda_1) V_{k-1}, & -2a_k/\lambda + \alpha_k \\
\end{bmatrix}, \quad \hat{U}_k := \begin{bmatrix}
-2a_k/\lambda + \alpha_k V_{k-1}, & -2a_k/\lambda + \alpha_k \\
\end{bmatrix}.
\]

We need to find \( Q_k \) such that \( \hat{U}_k Q_k = V_k \). Then, \( \hat{U}_k = V_k P_k \), where \( P_k = Q_k^{-1} + \text{blkdiag}(I, 0) \).

Let \( \hat{Q} = \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} 1 & I \\
1 & I \\
\end{bmatrix} \end{bmatrix} \end{bmatrix}, \)

then

\[
U_k := \hat{U}_k \hat{Q} = \begin{bmatrix}
-2a_k/\lambda + \alpha_k, & -2a_k/\lambda + \alpha_k, & -2a_k/\lambda + \alpha_k, & \cdots, & -2a_k/\lambda + \alpha_k \\
\end{bmatrix}.
\]

Because for any \( s = 1, 2, \ldots, k - 1 \),

\[
\begin{bmatrix}
1/\lambda + \alpha_s, & 1 \\
1/\lambda + \alpha_s, & 1 \\
\end{bmatrix} \begin{bmatrix}
1 & 1 \\
1 & 1 \\
\end{bmatrix} \begin{bmatrix}
1 & 1 \\
1 & 1 \\
\end{bmatrix} = \begin{bmatrix}
\lambda/\lambda + \alpha_s, & 1 \\
\lambda/\lambda + \alpha_s, & 1 \\
\end{bmatrix},
\]

it holds that \( U_k Q = U_k \), where

\[
Q := \begin{bmatrix}
1 & 1 \\
I & I \\
\end{bmatrix} \cdots \begin{bmatrix}
1 & 1 \\
I & I \\
\end{bmatrix} \begin{bmatrix}
1 & 1 \\
I & I \\
\end{bmatrix}
\]

and

\[
U_k := \begin{bmatrix}
-2a_k/\lambda + \alpha_k, & -2a_k/\lambda + \alpha_k, & -2a_k/\lambda + \alpha_k, & \cdots, & -2a_k/\lambda + \alpha_k \\
\end{bmatrix}.
\]

Because for any \( s = 1, 2, \ldots, k - 1 \),

\[
\begin{bmatrix}
1/\lambda + \alpha_s, & 1 \\
1/\lambda + \alpha_s, & 1 \\
\end{bmatrix} \begin{bmatrix}
\hat{\alpha}_s + a_k/\alpha_k \\
\hat{\alpha}_s - a_k/\alpha_k \\
\end{bmatrix} = \begin{bmatrix}
-2a_s/\lambda + \alpha_s, & -2a_s/\lambda + \alpha_s \\
\end{bmatrix},
\]

it holds that \( U_k Q = V_k \) where

\[
Q := \begin{bmatrix}
\hat{\alpha}_1 + a_k/\alpha_k \\
\hat{\alpha}_1 - a_k/\alpha_k \\
\hat{\alpha}_2 + a_k/\alpha_k \\
\hat{\alpha}_2 - a_k/\alpha_k \\
\cdots \\
\hat{\alpha}_{k-1} + a_k/\alpha_k \\
\hat{\alpha}_{k-1} - a_k/\alpha_k \\
\end{bmatrix}.
\]
This implies that we can determine $Q_k$ and $P_k$ such that $\tilde{U}_k Q_k = V_k$ and $\tilde{U}_k = V_k P_k$, that is

\[
Q_k = \begin{bmatrix}
I & \\
1 & \cdots & 1
\end{bmatrix}
\begin{bmatrix}
\frac{\alpha_1 + \alpha_k}{2\alpha_k} & \frac{\alpha_1 - \alpha_k}{2\alpha_k} & \frac{\alpha_2 + \alpha_k}{2\alpha_k} & \ldots & \frac{\alpha_{k-1} + \alpha_k}{2\alpha_k} \\
\frac{\alpha_1 - \alpha_k}{2\alpha_k} & \frac{\alpha_1 + \alpha_k}{2\alpha_k} & \frac{\alpha_2 - \alpha_k}{2\alpha_k} & \ldots & \frac{\alpha_{k-1} - \alpha_k}{2\alpha_k} \\
\frac{\alpha_2 + \alpha_k}{2\alpha_k} & \frac{\alpha_2 - \alpha_k}{2\alpha_k} & \frac{\alpha_3 + \alpha_k}{2\alpha_k} & \ldots & \frac{\alpha_{k-2} + \alpha_k}{2\alpha_k} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\frac{\alpha_{k-1} + \alpha_k}{2\alpha_k} & \frac{\alpha_{k-1} - \alpha_k}{2\alpha_k} & \frac{\alpha_{k-2} + \alpha_k}{2\alpha_k} & \ldots & \frac{\alpha_{k-3} + \alpha_k}{2\alpha_k}
\end{bmatrix}
\]

\[
P_k = Q_k^{-1} + \begin{bmatrix}
I \\
0
\end{bmatrix}.
\]

We summarize the resulting method in Algorithm 2. We remark that this implementation generates a Cholesky-Factor-ADI-like (CF-ADI) basis \cite[Algorithm 2]{Lin2015} (the algorithm will be the same for $F = 0$); more precisely, the CF-ADI algorithm uses

\[
\tilde{V}_k := \begin{bmatrix}
\frac{\sqrt{2\alpha_1} - \lambda}{\lambda + \alpha_1} & \frac{\sqrt{2\alpha_2} - \lambda}{\lambda + \alpha_2} & \ldots & \frac{\sqrt{2\alpha_{k-1}} - \lambda}{\lambda + \alpha_{k-1}} & \frac{\sqrt{2\alpha_k} - \lambda}{\lambda + \alpha_k}
\end{bmatrix} \begin{bmatrix}
\frac{\sqrt{2\alpha_1} - \lambda}{\lambda + \alpha_1} & \frac{\sqrt{2\alpha_2} - \lambda}{\lambda + \alpha_2} & \ldots & \frac{\sqrt{2\alpha_{k-1}} - \lambda}{\lambda + \alpha_{k-1}} & \frac{\sqrt{2\alpha_k} - \lambda}{\lambda + \alpha_k}
\end{bmatrix}^{-1}
\]

\[
\tilde{v}_1 = \sqrt{2\alpha_1}(A^* + \alpha_1 I)^{-1} C^*
\]

\[
\tilde{v}_{i+1} = \frac{\sqrt{2\alpha_{i+1}}}{\sqrt{2\alpha_i}} [I - (\alpha_{i+1} + \alpha_i)(-A^* + \alpha_{i+1} I)^{-1}] \tilde{v}_i
\]

for which a corresponding expression of $X_k$ can be derived. This selection of $v_1, T_1$ makes Algorithm 2 mathematically equivalent to the recurrence in (5) with $X_0 = 0$.

\begin{algorithm}
1: INPUT: $A \in \mathbb{R}^{n \times n}$ stable, $C \in \mathbb{R}^{p \times n}$, $F = BB^*$ with $B \in \mathbb{R}^{n \times d}$, $\alpha_k, k = 1, 2, \ldots$, with $\alpha_k = \mathbb{N}(\alpha_k) > 0$
2: $v_1 := -2\alpha_1(-A^* + \alpha_1 I)^{-1} C^*$, $V_1 := v_1$, $T_1 := [2\alpha_1 I + 2\alpha_1 C(-A + \alpha_1 I)^{-1} F(-A^* + \alpha_1 I)^{-1} C^*]$
3: for $k = 2, 3, \ldots$ do
4: $v_k := \frac{\alpha_k}{\alpha_{k-1}} (v_{k-1} - (\alpha_{k-1} + \alpha_k)(-A^* + \alpha_k I)^{-1} v_{k-1})$
5: $V_k := [V_{k-1}, v_k]$
6: $Q_k := \begin{bmatrix}
I & \\
1 & \cdots & 1
\end{bmatrix}
\begin{bmatrix}
\frac{\alpha_1 + \alpha_k}{2\alpha_k} & \frac{\alpha_1 - \alpha_k}{2\alpha_k} & \frac{\alpha_2 + \alpha_k}{2\alpha_k} & \ldots & \frac{\alpha_{k-1} + \alpha_k}{2\alpha_k} \\
\frac{\alpha_1 - \alpha_k}{2\alpha_k} & \frac{\alpha_1 + \alpha_k}{2\alpha_k} & \frac{\alpha_2 - \alpha_k}{2\alpha_k} & \ldots & \frac{\alpha_{k-1} - \alpha_k}{2\alpha_k} \\
\frac{\alpha_2 + \alpha_k}{2\alpha_k} & \frac{\alpha_2 - \alpha_k}{2\alpha_k} & \frac{\alpha_3 + \alpha_k}{2\alpha_k} & \ldots & \frac{\alpha_{k-2} + \alpha_k}{2\alpha_k} \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
\frac{\alpha_{k-1} + \alpha_k}{2\alpha_k} & \frac{\alpha_{k-1} - \alpha_k}{2\alpha_k} & \frac{\alpha_{k-2} + \alpha_k}{2\alpha_k} & \ldots & \frac{\alpha_{k-3} + \alpha_k}{2\alpha_k}
\end{bmatrix} \otimes I_p$
7: $P_k := Q_k^{-1} + \begin{bmatrix}
I \\
0
\end{bmatrix} \otimes I_p$
8: $T_k := P_k^{-1} \begin{bmatrix}
T_{k-1} & 0 \\
0 & 2\alpha_k I
\end{bmatrix} + \frac{1}{2\alpha_k} Q_k^{-1} V_k^* F V_k Q_k^{-1} P_k^{-1}$
9: end for
10: OUTPUT: $V_k, T_k$ s.t. $X_k = V_k T_k^{-1} V_k^* \approx X_+$
\end{algorithm}
The algorithm requires one solve with \(-A^* + \alpha_k I\) per iteration, while most other computations are performed with small matrices whose dimensions do not depend on the problem size. In addition to the memory requirements associated with such a solve, memory allocations with long vectors are limited to storing the matrix \(V_k\).

The algorithm sequentially expands the matrix \(V_k\) as the iteration progresses. If \(C^*\) has multiple columns, then the columns of \(V_k\) increases correspondingly, at each iteration. Regardless of the number of columns of \(C^*\), the matrix \(V_k\) becomes increasingly ill-conditioned, possibly losing numerical rank. Although this fact does not influence the stability of the method, the whole matrix \(V_k\) is required, so that memory requirements expand accordingly. However, \(V_k\) may be stored as \(V_k = V_kR_k\), with \(V_k\) of (smaller) full column numerical rank, and the small matrix \(R_k\) possibly having a larger number of columns than rows. This way, the much thinner matrix \(V_k\) can be saved in place of \(V_k\). We do not report the implementation details of this approach, which can be found in [30], and note that this implementation provides the same numerical results as the original one, up to the truncation tolerance used.

**Remark 3.3**
Algorithm 2 can be easily extended to handle the following generalized algebraic Riccati equation

\[
G + A^*XE + E^*XA - E^*XFXE = 0
\]

with \(E\) nonsingular. In particular, from \((CE^{-1})^*CE^{-1} + (AE^{-1})^*X + XAE^{-1} - XFX = 0\), it follows that \(C^*\) is substituted by \(E^{-*}C^*\), and \((-A^* + \alpha_i I)^{-1}\) by \((-A^* + \alpha_i E^*)^{-1}E^*\). As a consequence, only the lines 2 and 4 of Algorithm 2 require some modifications. In particular, these two lines are replaced by

\[
2'.v_1 := -2\alpha_1 (-A^* + \alpha_1 E^*)^{-1}C^*, V_1 := v_1,
\]

\[
T_1 := [2\alpha_1 I + 2\alpha_1 C(-A + \tilde{\alpha}_1 E)^{-1}F(-A^* + \alpha_1 E^*)^{-1}C^*]
\]

\[
4'.v_k := \frac{\alpha_k}{\alpha_{k-1}}(v_{k-1} - (\alpha_k - 1 + \tilde{\alpha}_k)(-A^* + \alpha_k E^*)^{-1}E^*v_{k-1})
\]

The rest of the algorithm is unchanged.

### 3.2. Shift selection and initialization

The shifts should be selected so as to speed up convergence. In this section, we provide theoretical ground for an effective shift selection, together with a practical implementation of such a strategy. Moreover, we describe a choice of the starting approximate solution \(X_0\) that ensures convergence of the whole process.

We first need to recall some definitions and known relations. Let \(D_n(\mathcal{H}^*)\) denote the left c-stable invariant subspace of \(\mathcal{H}\) [31, p.333]. Let \(\mathcal{H} = QTQ^*\) be the Schur decomposition of \(\mathcal{H}\), with

\[
Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}, \quad T = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}, \quad \sigma(T_{11}) \subset \mathbb{C}_-, \quad \sigma(T_{22}) \subset \mathbb{C}_+.
\]  

(20)

Then, for every \(k\) and \(\Re(\alpha_k) > 0\), the Cayley transformation has Schur decomposition \(S_k = QT(k)Q^*\), where \(T(k) := \begin{bmatrix} T_{11(k)} & T_{12(k)} \\ 0 & T_{22(k)} \end{bmatrix}\) with \(T_{11(k)} = (T_{11} + \alpha_k I)^{-1}(T_{11} - \tilde{\alpha}_k I)\) having all eigenvalues outside the unit disk, while \(T_{22(k)} = (T_{22} + \alpha_k I)^{-1}(T_{22} - \tilde{\alpha}_k I)\) has all eigenvalues in the unit disk.

Given two subspaces \(\mathcal{G}_1\) and \(\mathcal{G}_2\) of \(C^n\) of equal dimension, their distance is given by (see, e.g., [31, p.76]) \(\text{dist}(\mathcal{G}_1, \mathcal{G}_2) = \|P_1 - P_2\|_2\), where \(P_i\) is the orthogonal projection matrix onto \(\mathcal{G}_i\). Finally, (see, e.g., [31, p.325])
Next theorem provides a bound on the expected convergence rate. Its proof is an adaptation of the general proof in [31, Theorem 7.3.1, page 337] to our context and can be found in [32].

**Theorem 3.4**

Let \( \begin{bmatrix} I & X_0 \end{bmatrix} = U_0 R_0 \) be the skinny QR decomposition of \([I; X_0]\) and assume that \( X_0 \) is such that

\[
d = \text{dist} \left( D_n(H^*), \text{Range} \left( \begin{bmatrix} I \\ X_0 \end{bmatrix} \right) \right) < 1.
\]

If for any \( k > 0 \), the matrix \( M_k \) in the iteration (5) is nonsingular, then the associated iterate \( X_k \) satisfies

\[
\text{dist} \left( \text{Range} \left( \begin{bmatrix} I \\ X_+ \end{bmatrix} \right), \text{Range} \left( \begin{bmatrix} I \\ X_k \end{bmatrix} \right) \right) \leq \gamma \left\| \prod_{i=k}^{1} T_{22(i)} \right\|_2 \left\| \prod_{i=1}^{k} T_{11(i)}^{-1} \right\|_2
\]

where \( \gamma = \frac{\|K_0^{-1}\|_2}{\sqrt{1 - d^2}} \left( 1 + \frac{\|T_1\|_F}{\text{sep}(T_{11}, T_{22})} \right) \).

Theorem 3.4 shows that the distance between the exact and approximate subspaces is bounded in terms of the norms of the products of the \( T_{22(i)} \)'s and \( T_{11(i)}^{-1} \)'s. Unfortunately, we do not have a handle on the norm of these two matrices, and thus on their non-normality, whereas eigenvalue information can be used more easily. The next proposition gives more insight into the role of the parameters by using \( \rho(T_{22(i)}) \) and \( \rho(T_{11(i)}^{-1}) \) for \( i = 1, \ldots, k \), which we know to be less than one.

**Proposition 3.5**

With the notation of Theorem 3.4, assume that the \( \alpha_i \)'s are such that the matrices \( T_{22(i)}, T_{11(i)}^{-1} \), for all \( i = 1, \ldots, k \) are well defined. Then,

\[
\rho \left( \prod_{i=k}^{1} T_{22(i)} \right) = \rho \left( \prod_{i=1}^{k} T_{11(i)}^{-1} \right) = \max_{\lambda \in \lambda_+(H)} \prod_{i=1}^{k} \frac{\lambda - \overline{\alpha_i}}{\lambda + \alpha_i}.
\]

**Proof**

From \( T_{22(i)} = (T_{22} + \alpha_i I)^{-1}(T_{22} - \overline{\alpha_i} I) \), \( T_{11(i)} = (T_{11} + \alpha_i I)^{-1}(T_{11} - \overline{\alpha_i} I) \), for \( i = 1, \ldots, k \), it follows that

\[
\sigma \left( \prod_{i=k}^{1} T_{22(i)} \right) = \left\{ \prod_{i=1}^{k} \frac{\lambda - \overline{\alpha_i}}{\lambda + \alpha_i} : \lambda \in \lambda_+(H) \right\}, \quad \sigma \left( \prod_{i=k}^{1} T_{11(i)} \right) = \left\{ \prod_{i=1}^{k} \frac{\lambda - \overline{\alpha_i}}{\lambda + \alpha_i} : \lambda \in \lambda_-(H) \right\}.
\]

From \( \lambda_+(H) = -\overline{\lambda_-(H)} \), the result follows. \( \square \)

Proposition 3.5, together with the requirement that all \( \alpha_i \) have positive real part, motivates the computation of the parameters as

\[
\{\alpha_1, \ldots, \alpha_k\} = \arg \min_{\alpha_1, \ldots, \alpha_k > 0} \max_{\lambda \in \lambda_+(H)} \prod_{i=1}^{k} \frac{\lambda - \overline{\alpha_i}}{\lambda + \alpha_i}.
\]

Note that here and throughout the paper, we assume that the set of parameters is closed under conjugation, that is if \( \alpha \) belongs to the set, also \( \overline{\alpha} \) does. In case of complex data, this constraint is unnecessary.
The problem of selecting the parameters is quite similar to the one in ADI for the Lyapunov equation (see [33–36] and the discussion in [19]), except that now the maximization is performed with respect to $\mathcal{H}$ instead of $A$. We implemented a variant of Penzl’s algorithm in [36], which selects the best $m$ Ritz values of $\mathcal{H}$ with positive real part, among those obtained in the generated Krylov subspaces with $\mathcal{H}$ and $\mathcal{H}^{-1}$ of size $m_1$ and $m_2$, respectively. In our simple implementation, we did not make any special effort to preserve the symmetric spectral structure in the computation of the Ritz values, which should instead be taken into account in case accurate computation is required. Our numerical experience indicates that the subspace iteration strongly depends on the quality of these parameters and that different selection strategies than this one may be more effective; see section 5 for further discussion.

Theorem 3.4 also requires a condition on the initial approximation $X_0$. A very simple choice of $X_0$, the zero matrix, turns out to satisfy this hypothesis.

**Proposition 3.6**

Assume that $A$ is stable and $F, G \succeq 0$. If $X_0 = 0$, then $\text{dist} \left( D_n(\mathcal{H}^*), \text{Range} \left( \begin{bmatrix} I \\ X_0 \end{bmatrix} \right) \right) < 1$.

**Proof**

We have that $(A^*, G)$ is stabilizable (see, e.g., [7, p.12]); that is, there exists $Z_+ \succeq 0$ such that $F + AZ + ZA^* - ZGZ = 0$, with $\sigma(A^* - GZ_+) \subset \mathbb{C}^-$ and

$$
\begin{bmatrix}
A^* & -G \\
-F & -A \\
\end{bmatrix}
\begin{bmatrix}
I \\
Z_+ \\
\end{bmatrix} = 
\begin{bmatrix}
I \\
Z_+ \\
\end{bmatrix} (A^* - GZ_+).
$$

(24)

Therefore, $D_n(\mathcal{H}^*) = \text{Range} \left( \begin{bmatrix} I \\ Z_+ \end{bmatrix} \right) = \text{Range} \left( \begin{bmatrix} I \\ Z_+ \end{bmatrix} (I + Z_+ Z_+)^{-\frac{1}{2}} \right)$, where the last matrix in parentheses has orthonormal columns. Thus,

$$
\text{dist} \left( D_n(\mathcal{H}^*), \text{Range} \left( \begin{bmatrix} I \\ 0 \end{bmatrix} \right) \right) = \left\| \begin{bmatrix} 0 & I \\ Z_+ \\
\end{bmatrix} (I + Z_+ Z_+)^{-\frac{1}{2}} \right\|_2 < 1,
$$

where the strict inequality follows from the fact that the $(1,1)$ block of the orthonormal basis is nonsingular. $\square$

### 3.3. Computation of the residual norm

Unless the problem size is small, the square residual matrix should not be computed explicitly. Instead, following similar procedures already used in the literature (cf., e.g., [29]), the computation of the residual norm can be performed economically by exploiting the low-rank form of the approximate solution. At iteration $k$, using $X_k = V_k T_k V_k^*$ gives

$$
\| A^* X_k E + E^* X_k A - E^* X_k B B^* X_k E + C^* C \|_F
$$

$$
= \left\| \begin{bmatrix} C^* & A^* V_k & E^* V_k \\
I & 0 & 0 \\
0 & 0 & T_k^{-1} \\
0 & T_k^{-1} & T_k^{-1} V_k^* B B^* V_k T_k^{-1} \\
\end{bmatrix} \begin{bmatrix} C \\
V_k^* A \\
V_k^* E \end{bmatrix} \right\|_F
$$

$$
= \left\| R_k \begin{bmatrix} I & 0 & 0 \\
0 & 0 & T_k^{-1} \\
0 & T_k^{-1} & T_k^{-1} V_k^* B B^* V_k T_k^{-1} \\
\end{bmatrix} R_k^* \right\|_F,
$$

(25)

where $R_k$ is obtained from the economy-size QR decomposition of $[C^*, A^* V_k, E^* V_k]$. Because the basis in $V_k$ is nested, it is possible to update $R_k$ at each iteration by means of a Gram–Schmidt-type procedure, without recomputing the decomposition from scratch.
4. SUBSPACE ITERATION AND GALERKIN RATIONAL KRYLOW SUBSPACE METHODS

In the linear case (i.e., \( F = 0 \)), it is known that the ADI method and the Galerkin RKSM determine approximate solutions that belong to the same rational Krylov subspace [28]. More recently, in [37], it was proved that the two methods give exactly the same approximate solution if and only if the two spaces use the same shifts and these shifts coincide with the mirrored Ritz values of \( A \) onto the generated space.

In this section, we show that a natural generalization of this property also holds for our setting, leading to an equivalence between the subspace iteration and the Galerkin rational Krylov method applied to the Riccati equation. We recall here that RKSM determines a solution onto the rational Krylov subspace by requiring that the residual matrix associated with the approximate solution \( X_k^{(G)} \) be ‘orthogonal’ to the space; see, for example, [18]. More precisely, setting \( R_k^{(G)} := X_k^{(G)} A - X_k^{(G)} F X_k^{(G)} + G \), it holds that \( U_k^* R_k^{(G)} U_k = 0 \), where the orthonormal columns of \( U_k \) span the rational Krylov subspace. Writing \( X_k^{(G)} = U_k Y_k U_k^* \), the condition \( 0 = U_k^* R_k^{(G)} U_k \) corresponds to the reduced equation \( U_k^* A^* U_k Y_k + Y_k U_k^* A U_k - Y_k U_k^* F U_k Y_k + U_k^* G U_k = 0 \). This equation admits a unique stabilizable solution \( Y_k \) under the assumptions that \( U_k^* A^* U_k \) is stable and \( U_k^* F U_k \geq 0, U_k^* G U_k \geq 0 \). Therefore, in this section, we assume that \( A \) is passive; that is, \( (x^* A x)/(x^* x) < 0 \) for all \( x \neq 0 \), so that \( U_k^* A^* U_k \) is stable.

To prove the equivalence, we exploit yet another basis for the rational Krylov subspace, namely (see [37] for similar considerations) **

\[
V_k = \text{Range}([(-A^* + \alpha_1 I)^{-1} C^*, \ldots, (-A^* + \alpha_k I)^{-1} C^*]).
\]  

We next show how to generate the matrices \( Q_k \) and \( P_k \) so as to use \( V_k \) as reference basis; the derivation below could be obtained also for nested bases [30]. We assume that \( C^* \) has a single column; otherwise, a Kronecker form as in Algorithm 2 can be used. With the scalar rational function notation, we write

\[
V_k := \left[ \frac{1}{\lambda + \alpha_1}, \frac{1}{\lambda + \alpha_2}, \ldots, \frac{1}{\lambda + \alpha_{k-1}}, \frac{1}{\lambda + \alpha_k} \right],
\]

together with the definition of \( \hat{U}_k \) in (18). We observe that

\[
\begin{bmatrix}
-2\alpha_k/
\lambda + \alpha_k
\end{bmatrix}
\begin{bmatrix}
V_{k-1} \
-2\alpha_k/\lambda + \alpha_k
\end{bmatrix}
= \left[
\begin{array}{cccc}
-2\alpha_k/\lambda + \alpha_1 & -2\alpha_k/\lambda + \alpha_2 & \cdots & -2\alpha_k/\lambda + \alpha_{k-1} & -2\alpha_k/
\lambda + \alpha_k
\end{array}
\right].
\]

Moreover,

\[
\begin{bmatrix}
\frac{1}{\lambda + \alpha_k} & 1 \\
\frac{1}{\lambda + \alpha_1} & \frac{1}{\lambda + \alpha_2} & 1 & \frac{1}{\lambda + \alpha_k}
\end{bmatrix}
\begin{bmatrix}
\alpha_k - \alpha_s \\
\alpha_k - \alpha_1
\end{bmatrix}
\begin{bmatrix}
\alpha_k - \alpha_s \\
\alpha_k - \alpha_1
\end{bmatrix}
\]

for \( s = 1, \ldots, k - 1 \),

and

\[
\begin{bmatrix}
\frac{1}{\lambda + \alpha_k} & 1 \\
\frac{1}{\lambda + \alpha_1} & \frac{1}{\lambda + \alpha_2} & 1 & \frac{1}{\lambda + \alpha_k}
\end{bmatrix}
\begin{bmatrix}
\alpha_k - \alpha_s \\
\alpha_k - \alpha_1
\end{bmatrix}
\begin{bmatrix}
\alpha_k - \alpha_s \\
\alpha_k - \alpha_1
\end{bmatrix}
\]

for \( s = 1, \ldots, k - 1 \).

Therefore, with

\[
Q_k = \frac{1}{-2\alpha_k}
\begin{bmatrix}
\alpha_k - \alpha_1 \\
\alpha_k - \alpha_2 \\
\vdots \\
\alpha_k - \alpha_{k-1}
\end{bmatrix}
\begin{bmatrix}
1 \\
1 \\
\vdots \\
1
\end{bmatrix},
\]

For the basis to be full rank, a necessary condition is that all shifts be distinct. In practice, by using the nested space construction, it is readily seen that this condition may be relaxed by allowing higher negative powers of \( (-A^* + \alpha_i I) \) in case \( \alpha_i \) is a multiple shift.
we obtain

\[
P_k = Q_k^{-1} + \begin{bmatrix} I & 0 \end{bmatrix} = \begin{bmatrix}
\frac{a_1-\alpha_k}{\alpha_1+\alpha_k} & \frac{a_2-\alpha_k}{a_2-\alpha_k} & \cdots & \frac{a_{k-1}-\alpha_k}{a_{k-1}-\alpha_k} & -2a_k \\
\frac{2a_k}{\alpha_k-\alpha_1} & \frac{2a_k}{\alpha_k-\alpha_2} & \cdots & \frac{2a_k}{\alpha_k-\alpha_{k-1}} & 0
\end{bmatrix},
\]

so that

\[
P_k^{-1} = \begin{bmatrix}
\frac{a_1-\alpha_k}{\alpha_1+\alpha_k} & \frac{a_2-\alpha_k}{a_2-\alpha_k} & \cdots & \frac{a_{k-1}-\alpha_k}{a_{k-1}-\alpha_k} \\
\frac{-1}{\alpha_1+\alpha_k} & \frac{-1}{\alpha_2+\alpha_k} & \cdots & \frac{-1}{\alpha_{k-1}+\alpha_k}
\end{bmatrix}.
\]

As already mentioned, the approximation \(X_k\) can be written in terms of the new basis and representation matrix as \(X_k = V_k T_k^{-1} V_k^*\) with \(V_k\) as in (26) and

\[
T_k = P_k^{-*} \begin{bmatrix}
T_{k-1} & 0 & 2a_k I \\
0 & 2a_k I
\end{bmatrix} + \frac{1}{2a_k} Q_k^{-*} V_k^* F V_k Q_k^{-1},
\]

where \(P_k^{-1}\) and \(Q_k\) are as defined in (30) and (28), respectively. Here, we focus on the use of this formulation for demonstrating the connection of our approach with RKSM. We first show that the reduced matrix \(T_k\) satisfies a linear matrix equation.

**Lemma 1**

Let \(V_k\) and \(T_k\) be as in (26) and (31), respectively, and \(\alpha_k = \text{diag}(\alpha_1, \ldots, \alpha_k)\). Let \(I = [1, \ldots, 1]^*\). Then,

\[
\alpha_k^* T_k + T_k \alpha_k - V_k^* F V_k - 11^* = 0.
\]

**Proof**

With \(P_k^{-1}\) in (30) and \(Q_k^{-1}\) expressed via (29), we first observe that

\[
Q_k^{-1} P_k^{-1} = \left( P_k - \begin{bmatrix} I & 0 \end{bmatrix} \right) P_k^{-1} = \text{diag} \left( \frac{2a_k}{\alpha_1+\alpha_k}, \ldots, \frac{2a_k}{\alpha_{k-1}+\alpha_k}, 1 \right).
\]

We next prove that \(T_k(i, j) = \frac{1+\tilde{F}_{ij}}{\tilde{a}_i+\tilde{a}_j}\) for \(i, j \leq k\) by induction on \(k\), where \(\tilde{F}_{ij} = V_k^* F V_k\). For \(k = 1\), it can be verified that \(T_1 = \frac{1}{\tilde{a}_1} + \frac{V_1^* F V_1}{2\tilde{a}_1}\). Assume that the relation holds for \(T_{k-1}\). Noticing the structure of \(P_k^{-1}\) and \(Q_k^{-1} P_k^{-1}\), for \(i < k, j < k\), we have

\[
T_k(i, j) = e_i^* P_k^{-1} \begin{bmatrix} T_{k-1} & 0 \\
0 & 2a_k
\end{bmatrix} P_k^{-1} e_j + \frac{1}{2a_k} e_i^* P_k^{-*} Q_k^{-*} V_k^* F V_k Q_k^{-1} P_k^{-1} e_j
\]

\[
= \left[ \frac{\tilde{a}_i-\tilde{a}_k}{\tilde{a}_i+\tilde{a}_k} \right] \left[ \frac{1+\tilde{F}_{ij}}{\tilde{a}_i+\tilde{a}_j} \right] \left[ \frac{\alpha_j-\alpha_k}{\alpha_j+\alpha_k} \right] + \frac{1}{2a_k} \tilde{F}_{ij} \left[ \frac{2a_k}{\tilde{a}_i+\tilde{a}_j} \right] + \frac{2a_k}{\tilde{a}_i+\tilde{a}_j} \left[ \frac{\tilde{a}_i-\tilde{a}_k}{\tilde{a}_i+\tilde{a}_j} \right]
\]

\[
= \frac{\tilde{a}_i-\tilde{a}_k}{\tilde{a}_i+\tilde{a}_j} \left[ \frac{1+\tilde{F}_{ij}}{\tilde{a}_i+\tilde{a}_j} \right] + \frac{2a_k}{(\tilde{a}_i+\tilde{a}_j)(\alpha_j+\tilde{a}_k)} + \frac{2a_k}{(\tilde{a}_i+\tilde{a}_j)(\alpha_j+\tilde{a}_k)}
\]

\[
= 1 + \tilde{F}_{ij} \frac{\tilde{a}_i+\alpha_k}{\tilde{a}_i+\alpha_j}.
\]
For \( j = k \) and \( i \leq k \), we obtain

\[
T_k(i, k) = e_i^* T_k e_k = e_i^* P_k^{-*} \begin{bmatrix} T_{k-1} & 0 \\ 0 & 2\alpha_k \end{bmatrix} P_k^{-1} e_k + \frac{1}{2\alpha_k} e_i^* P_k^{-*} Q_k^{-*} V_k^* F V_k Q_k^{-1} P_k^{-1} e_k
\]

\[
= e_i^* P_k^{-*} (-e_k) + \frac{1}{2\alpha_k} \tilde{F}_{ik} \tilde{e}_k = \frac{1}{\alpha_i + \alpha_k}.
\]

The structure of \( T_k(k, j) \), \( j \leq k \) is obtained by symmetry, completing the proof. \( \square \)

We notice that Lemma 1 also shows that the principal \((k - 1) \times (k - 1)\) diagonal block of \( T_k \) coincides with \( T_{k-1} \). As an immediate consequence of this fact, we show that the approximate solution \( X_k \) can be updated from \( X_{k-1} \) with a rank-one matrix (a rank-\( p \) matrix if \( C^* \) has \( p \) columns); this is similar to what one finds with CF-ADI. In addition, the approximation sequence is weakly monotonically increasing.

**Proposition 3.1**

For \( k > 0 \), the approximate solution \( X_k \) is such that \( X_k - X_{k-1} \) has rank one. Moreover, for all \( k > 0 \), \( X_k \geq X_{k-1} \).

**Proof**

Let \( X_k = V_k T_k^{-1} V^* \), with \( V_k \) and \( T_k \) as in (26) and (31), respectively. Because \( T_k(1 : k - 1, 1 : k - 1) = T_{k-1} \), we have \( X_k - X_{k-1} = V_k \left( T_k^{-1} - \begin{bmatrix} T_{k-1}^{-1} \\ 0 \end{bmatrix} \right) V_k^* \).

We next show that the matrix in parentheses has rank one. Let \( T_{k-1} = LL^* \) be the Cholesky decomposition of \( T_{k-1} \). Then,

\[
T_k = \begin{bmatrix} L & 0 \\ L^* & l \end{bmatrix}, \quad T_k^{-1} = \begin{bmatrix} L^{-*} & -L^{-*}l \tilde{m}^{-1} \\ 0 & \tilde{m}^{-1} \end{bmatrix}, \quad T_{k-1}^{-1} = \begin{bmatrix} L^{-1} & 0 \\ -l^{-1}L^{-1}m^{-1} & m^{-1} \end{bmatrix}.
\]

By explicitly writing down the \((1,1)\) block of \( T_k^{-1} \), it follows

\[
T_{k-1}^{-1} - T_{k-1}^{-1} \begin{bmatrix} T_{k-1}^{-1} \\ 0 \end{bmatrix} = \begin{bmatrix} -l^{-1}L^{-1}m^{-1} \\ m^{-1} \end{bmatrix} =: \ell \ell^*.
\]

which has rank one, as stated. Finally, \( X_k = X_{k-1} + V_k \ell \ell^* V_k^* \) with \( V_k \ell \ell^* V_k^* \geq 0 \), thus completing the proof. \( \square \)

With these results in hand, we are able to show that for \( C^* \) having a single column, the Riccati equation residual associated with \( X_k \) is also a rank-one matrix.

**Proposition 4.3**

Assume \( C \) is rank one. Then, the residual matrix \( R_k = C^* C + A^* X_k + X_k A - X_k F X_k \) is also rank one.

**Proof**

Recalling the notation leading to (32), we can write \( A^* V_k = -C^* 1^* + V_k \alpha_k \), so that

\[
\left[ C^*, A^* V_k, V_k \right] = [C^*, V_k] \begin{bmatrix} 1 & 1 \\ -1^* & 0 \end{bmatrix} + [C^*, V_k] \begin{bmatrix} 0 & \alpha_k \end{bmatrix} \begin{bmatrix} 1 \\ l \end{bmatrix}.
\]
Using (32),

\[
R_k = [C^*, A^* V_k, V_k] [\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & T_k^{-1} \\ 0 & T_k^{-1} & -T_k^{-1} V_k^* F V_k T_k^{-1} \end{bmatrix}] [C^*, A^* V_k, V_k]^*
\]

\[
= [C^*, V_k] [\begin{bmatrix} \frac{-1}{T_k} I & 0 \\ 0 & 0 \frac{-1}{T_k} I \\ 1 & 0 \end{bmatrix} [C^*, V_k]^*
\]

(33)

which is a rank-one matrix.

We can thus state the main result of this section, which gives necessary and sufficient conditions for the subspace iteration and RKSM for the Riccati equation to be mathematically equivalent. The equivalence follows from the uniqueness of the Galerkin solution onto the given space, determined by RKSM, following from the uniqueness of the stabilizing solution of the reduced problem.

**Theorem 4.4**

Assume \( A \) is passive and the notation and assumption of Proposition 4.3 hold. Let \( K_k := (V_k^* V_k)^{-1} V_k^* A^* V_k, g_k := (V_k^* V_k)^{-1} V_k^* C^* \). Then,

(i) The subspace iteration provides a Galerkin method on Range(\( V_k \)), namely, \( V_k^* R_k V_k = 0 \), if and only if \( g_k = T_k^{-1} \).

(ii) \( V_k^* R_k V_k = 0 \) if and only if \( \alpha_k T_k + T_k K_k - V_k^* F V_k = 0 \); in particular, the poles are the mirrored Ritz values of \( A^* - X_k F \), namely,

\[ \alpha_j = -\lambda_j, \quad j = 1, \ldots, k \]

where \( \lambda_j \) are the properly sorted eigenvalues of \( (V_k^* V_k)^{-1} V_k^* (A^* - V_k T_k^{-1} V_k^* F) V_k \).

**Proof**

Using the relations in the proof of Proposition 4.3, we first notice that the relations \( V_k^* A^* V_k = -V_k C^* I + V_k^* V_k \alpha_k \) and \( K_k = -g_k I^* + \alpha_k \) hold. Then, from (33) we obtain

\[
V_k^* R_k V_k = [V_k^* C^* - (V_k^* V_k) T_k^{-1} I] [V_k^* C^* - (V_k^* V_k) T_k^{-1} I]^*
\]

\[
= (V_k^* V_k) [g_k - T_k^{-1} I] [g_k - T_k^{-1} I]^* (V_k^* V_k),
\]

from which the necessary and sufficient condition in (i) follows.

For proving (ii), let us first assume that \( V_k^* R_k V_k = 0 \). Then, using \( T_k g_k = I \) in (i) and \( K_k = -g_k I^* + \alpha_k \), we have from Lemma 1

\[
0 = \alpha_k^* T_k + T_k \alpha_k - V_k^* F V_k - I^*
\]

\[
= \alpha_k^* T_k + T_k \alpha_k - V_k^* F V_k - T_k g_k I^*
\]

\[
= \alpha_k^* T_k + T_k \alpha_k - V_k^* F V_k + T_k (K_k - \alpha_k)
\]

\[
= \alpha_k^* T_k + T_k K_k - V_k^* F V_k.
\]

To prove the opposite direction, we start from \( 0 = \alpha_k^* T_k + T_k K_k - V_k^* F V_k \) and go backward to \( 0 = \alpha_k^* T_k + T_k \alpha_k - V_k^* F V_k - T_k (V_k^* V_k)^{-1} V_k^* C^* I^* \). Because our iterates satisfy (32), it must follow that \( T_k (V_k^* V_k)^{-1} V_k^* C^* = I \), that is, \( g_k = T_k^{-1} \).
Finally, we notice that (34) is equivalent to 
\[ T_k^{-1} \alpha^* T_k = T_k^{-1} V_k^* F V_k - K_k, \]
and
\[ T_k^{-1} \alpha^* T_k = (V_k^* V_k)^{-1} V_k^* V_k T_k^{-1} V_k^* F V_k - K_k \]
so that the eigenvalues of the first and last matrices coincide, and the eigenvalues of
\[ (V_k^* V_k)^{-1} V_k^* (V_k T_k^{-1} V_k^* F - A) V_k \]
coincide with those of
\[ (V_k^* V_k)^{-1/2} V_k^* (V_k T_k^{-1} V_k^* F - A) V_k (V_k^* V_k)^{-1/2}, \]
where the columns of
\[ V_k (V_k^* V_k)^{-1/2} \]
define an orthogonal basis for the space. Therefore, these are the Ritz values of
\[ V_k T_k^{-1} V_k^* F - A \]
on the space \( \text{Range}(V_k) \).

**Remark 4.5**

The previous theorem provides insight into the estimation of the poles of RKSM, when a greedy algorithm is used to generate a pole sequence ‘on the fly’; in the linear case, poles are estimated by an optimization strategy of a scalar rational function on a certain region of the complex plane. The function has poles at the already computed shifts and zeros at the Ritz values of \(-A\) in the current space [38]. The results of Theorem 4.4 suggest that in the quadratic case, an alternative choice could be given by the Ritz values of \(-A^* + \bar{X}_k^{(G)} F\), where \(X_k^{(G)}\) is the current approximate solution. The very preliminary experiments reported in Example 1 seem to encourage the use of this strategy when \(A\) is non-normal and \(X_k^{(G)} F\) is sizable in norm.

In the case when the Ritz values of \(A, \lambda_j\) are considered, the condition \(\alpha_j = -\bar{\lambda}_j\) is associated with the optimality of the generated rational Krylov subspace as a model order reduction process for a linear dynamical system; see, for example, [39]. Whether different optimality results could be shown in our setting remains an open problem.

## 5. NUMERICAL EXPERIMENTS

In this section, we report on our numerical experience with the subspace iteration described in Algorithm 2. Experiments were performed in MATLAB [27] with version 7.13 (R2011b) of the software.

We do not report these numerical experiments to propose the method as a valid competitor of, for example, rational Krylov subspace solvers, as the large majority of our experiments showed otherwise. Having the extra feature of the Galerkin projection, RKSM with the same poles will in general be superior to subspace iteration, both in terms of number of iterations and memory requirements. Instead, our purpose is to explore what the expected performance of the method will be, and highlight the relations with the Galerkin procedure, specifically in connection with the pole selection. This analysis also leads us to the derivation of a possibly more effective pole selection for RKSM, compared with what was used, for example, in [18]. All experiments are performed with \(F\) and \(G\) of rank one. Similar results may be obtained with matrices of larger rank. All plots report the computed residual norm, according to Section 3.3, versus the space dimension. In fact, for ILRSI, this refers to the number of columns in the matrix \(V_k\) in Algorithm 2, because the numerical rank of that matrix may be lower.

We do not report experimental comparisons with other methods such as inexact Newton, as they are available in [18], at least with respect to projection-type methods.

In all our examples with the subspace iteration algorithm ILRSI, the poles are computed a priori. Unless explicitly stated otherwise, these are computed using Penzl’s algorithm [36] on the matrix \(A\) (when used for these specific problems, the variant of Penzl’s algorithm using \(H\) mentioned in Section 3.2 did not give appreciably better results). In the first two examples, the performance of the new method is compared with that of adaptive RKSM, as used, for instance, in [18], where the poles are computed adaptively. We notice that the main computational cost per iteration, namely, the solution of the shifted system with \(A\), is the same for both methods; therefore, the number of solves may represent a good measure for the comparison.
Example 5.1
We consider the (scaled) discretization of the Laplace operator on the unit square, with 100 interior points in each direction, so that the resulting matrix $A$ has dimension $n = 10,000$. The matrices $F$ and $G$ are given as $F = bb^*$ and $G = c^*c$ with $b = 1$ and $c^* = e_1$. The performance of ILRSI is reported in Figure 1, together with that of RKSM. The convergence rate is similar for the two methods, although RKSM consistently shows smaller residual norm.

Example 5.2
In this example, we consider the data set FLOW from the Oberwolfach collection [40], with $n = 9669$; $B$ and $C^*$ have a single column. The convergence histories of the subspace iteration and of adaptive RKSM are reported in Figure 2. The left plot shows adaptive RKSM and ILRSI, where for the latter, the poles were pre-computed with Penzl’s algorithm on $A$. For this example, the adaptive RKSM is able to obtain an accurate solution appreciably earlier than the new method. In the right plot, subspace iteration was run with the poles adaptively generated by RKSM, showing a convergence history very similar to that of RKSM. Such different performance confirms what one usually finds in the linear case: the behavior of ADI is very sensitive to the poles choice.

In the following example, we explore the considerations of Remark 4.5 on the shifts used by subspace iteration and RKSM. To be able to compute all spectral quantities of interest, a small matrix is considered.
Example 1
We consider the $500 \times 500$ Toeplitz matrix

$$A = \begin{bmatrix}
2.5 & 1 & 1 & 1 & 0 \\
-1 & 2.5 & 1 & 1 & 1 \\
0 & -1 & 2.5 & 1 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
0 & -1 & 2.5 & 1 & 1 \\
\end{bmatrix},$$

with $C = [1, -2, 1, -2, \ldots]$, while $B = 1$ normalized or non-normalized. This type of matrices is known to be very non-normal, which implies that at small perturbations of the entries, there may correspond very large spectral perturbations; see, for example, [41, ch.7]. Figure 3 reports the convergence history with adaptive RKSM and ILRSI, when the latter uses the poles computed by the former. The left-most plot stems from using $B/\|B\|$ in place of $B$, whereas the middle plot refers to the unnormalized case. While the performance of RKSM only slightly degrades in the unnormalized case, that of subspace iteration drastically changes, showing almost complete stagnation. Indeed, two very large in modulus eigenvalues of $H$ are mapped by Cayley’s transformation to an area very close to the unit circle, for all parameters $\alpha_k$, thus causing very slow convergence. The right-most plot shows the performance of the methods with $B = 1$ (unnormalized), when the parameters in RKSM were computed by using the current Ritz values of $A - BB^*X_k^{(G)}$ instead of those of $A$ (cf. Theorem 4.4). We can readily see that performance of both methods is significantly improved and in particular, no complete stagnation occurs for subspace iteration. A closer look reveals that for $X$ exact, $A - BB^*X$ has an isolate eigenvalue close to $-250$ (apparently caused by the modification induced by the norm of $B$), which is not captured by the Ritz values of $A$ alone. When $B$ is normalized, the Ritz values of $A - BB^*X_k^{(G)}$ do not differ significantly from those of $A$, and thus, performance does not differ much. So, in this case, where the spectrum of $A - BB^*X$ differs significantly from that of $A$, using the Ritz values of $A - BB^*X_k^{(G)}$ for the adaptive computation of the parameters yields significantly better performance. This phenomenon deserves further study.

6. CONCLUSIONS

We have derived a computationally feasible subspace iteration algorithm for the approximation of the solution to the large-scale algebraic Riccati equation, when the matrices $F$ and $G$ have low
rank. The new method coincides with the ADI method in the linear equation case. Consequently, the performance of the new method depends on certain parameters, whose selection follows similar reasonings than those used for ADI. Our derivation also shows that ADI may be viewed as a subspace iteration method for the Hamiltonian matrix with $F = 0$. Other issues deserve further future analysis, such as the choice of the initial approximation $X_0$, which, together with a refined shift selection, could considerably speed up the process. Although we have worked throughout with real data, the method is also well suited for complex data, as long as the poles are chosen in a suitable manner.

We have also derived a new insightful connection of the proposed method with the Galerkin rational Krylov subspace scheme, which aims at generalizing known equivalence in the linear case. Such connection opens up a new venue for the understanding of the convergence properties of RKSM, which is a competitive alternative to Newton-based approaches. We plan to explore this problem in future work.

APPENDIX A: IN THIS APPENDIX, WE PROVE PROPOSITION 2.3

Proof

Using (6), we write

$$X_k = \left[ -2\alpha_k S_1^{-1} G(A + \alpha I)^{-1} + (I - 2\alpha_k S_1^{-1}) X_{k-1} \right] M_k^{-1}$$

and with $G = C^* C$, we can write

$$\begin{bmatrix}
-2\alpha_k (-A^* + \alpha I)^{-1} G S_2^{-1} + (I - 2\alpha_k S_1^{-1}) X_{k-1}
\end{bmatrix}
= \begin{bmatrix}
-2\alpha_k (-A^* + \alpha I)^{-1} C^*, (I - 2\alpha_k S_1^{-1}) U_{k-1}
\end{bmatrix}
\begin{bmatrix}
C S_2^{-1}
\end{bmatrix}
\begin{bmatrix}
L
T_{k-1}^{-1} U_{k-1}
\end{bmatrix}
R
$$

so that $X_k = L( R M_k^{-1} )$. Because $X_k$ is Hermitian (Proposition 2.3), $L$ and $M_k^{-*} R^*$ have the same column space; therefore, there exists $\tilde{T}$ such that $M_k^{-*} R^* \tilde{T} = L$.

Writing $M_k^* L = R^* \tilde{T}$, it is possible to recover $\tilde{T}$ explicitly (we omit the tedious algebraic computations), namely,

$$\tilde{T} = \begin{bmatrix}
2\alpha_k I \\
T_{k-1}
\end{bmatrix}
+ \begin{bmatrix}
2\alpha_k C(-A + \alpha_k I)^{-1} F(-A^* + \alpha_k I)^{-1} C^* \\
-2\alpha_k U_{k-1}^* (-A + \alpha_k I)^{-1} F S_2^{-1} (I - 2\alpha_k S_1^{-1}) U_{k-1}
\end{bmatrix}
\begin{bmatrix}
C S_2^{-1}
\end{bmatrix}
\begin{bmatrix}
L
T_{k-1}^{-1} U_{k-1}
\end{bmatrix}
R
$$

The symmetry of $\tilde{T}$ can be obtained after substituting (6) into the (2,1) block and using (11) for the (2,2) block.

Let $\mathcal{P} := [I - C(A + \alpha_k I)^{-1} F(-A^* + \alpha_k I)^{-1} C^*]^{-1} C(A + \alpha_k I)^{-1} F(-A^* + \alpha_k I)^{-1}$. Applying the Sherman–Morrison–Woodbury formula to $S_1$, we obtain

$$S_1^{-1} = (-A^* + \alpha_k I)^{-1} + (-A^* + \alpha_k I)^{-1} C^* \mathcal{P}$$

$$I - 2\alpha_k S_1^{-1} = (-A^* + \alpha_k I)^{-1} (-A^* - \alpha_k I) - 2\alpha_k (-A^* + \alpha_k I)^{-1} C^* \mathcal{P}.$$
Hence,
\[
L = \left[ -2\alpha_k (A^* + \alpha_k I)^{-1} C^*, (I - 2\alpha_k S_1^{-1}) U_{k-1} \right]
\]
\[
= \left[ -2\alpha_k (A^* + \alpha_k I)^{-1} C^*, (A^* + \alpha_k I)^{-1} (A^* - \hat{\alpha}_k I) U_{k-1} \right]
\]
\[
= \left[ (A^* + \alpha_k I)^{-1} (A^* - \hat{\alpha}_k I) U_{k-1}, -2\alpha_k (A^* + \alpha_k I)^{-1} C^* \right]
\]
\[
=: U_k \begin{bmatrix} 0 & I \\ I & \mathcal{P} U_{k-1} \end{bmatrix}.
\]

Explicit computation gives \( T_k \) in (13) (explicit details are omitted):
\[
\begin{bmatrix} -U_{k-1}^* \mathcal{P}^* & I \\ I & 0 \end{bmatrix} \tilde{T} \begin{bmatrix} -\mathcal{P} U_{k-1}^* & I \\ I & 0 \end{bmatrix} = T_k.
\]

Note that \( T_k \) nonsingular is equivalent to \( \tilde{T} \) nonsingular. Finally,
\[
X_k = L \tilde{T}^{-1} L^* = U_k \begin{bmatrix} 0 & I \\ I & \mathcal{P} U_{k-1} \end{bmatrix} \tilde{T}^{-1} \begin{bmatrix} 0 & I \\ I & U_{k-1}^* \mathcal{P}^* \end{bmatrix} U_k^*
\]
\[
= U_k \left( \begin{bmatrix} -U_{k-1}^* \mathcal{P}^* & I \\ I & 0 \end{bmatrix} \tilde{T} \begin{bmatrix} -\mathcal{P} U_{k-1}^* & I \\ I & 0 \end{bmatrix} \right)^{-1} U_k^* = U_k T_k^{-1} U_k^*,
\]

which gives the sought-after result. \( \square \)

ACKNOWLEDGEMENTS

We thank two anonymous reviewers, whose suggestions helped us improve the presentation. This work was performed while the first author was visiting the Department of Mathematics of the Università di Bologna during the period September 2011–August 2013, supported by fellowship 2011631028 from the China Scholarship Council (CSC).

REFERENCES

1. Lancaster P, Rodman L. Algebraic Riccati Equations. Oxford University Press: Oxford, 1995.
2. Antoulas AC. Approximation of Large-scale Dynamical Systems. Advances in Design and Control. SIAM: Philadelphia, 2005.
3. Konstantinov M, Gu D, Mehrmann V, Petkov P. Perturbation Theory for Matrix Equations, Studies in Computational Mathematics 9. Elsevier: Amsterdam, 2003.
4. Benner P, Mehrmann V, Sorensen D. Dimension Reduction of Large-Scale Systems (eds), Lecture Notes in Computational Science and Engineering. Springer-Verlag: Berlin/Heidelberg, 2005.
5. Schilders WHA, van der Vorst HA, Rommes J. Model Order Reduction: Theory, Research Aspects and Applications. Springer-Verlag: Berlin/Heidelberg, 2008.
6. Antoulas AC. Approximation of Large-scale Dynamical Systems. Advances in Design and Control. SIAM: Philadelphia, 2005.
7. Schilders WHA, van der Vorst HA, Rommes J. Model Order Reduction: Theory, Research Aspects and Applications. Springer-Verlag: Berlin/Heidelberg, 2008.
8. Benner P, Mehrmann V, Sorensen D. Dimension Reduction of Large-Scale Systems (eds), Lecture Notes in Computational Science and Engineering. Springer-Verlag: Berlin/Heidelberg, 2005.
9. Schilders WHA, van der Vorst HA, Rommes J. Model Order Reduction: Theory, Research Aspects and Applications. Springer-Verlag: Berlin/Heidelberg, 2008.
10. Bini DA, Iannazzo B, Meini B. Numerical Solution of Algebraic Riccati Equations. SIAM: Philadelphia, 2012.
11. Benner P, Li JR, Penzl T. Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic-optimal control problems. Numerical Linear Algebra with Applications 2008; 15:1–23.
12. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
13. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
14. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
15. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
16. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
17. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
18. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
19. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
20. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
21. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
22. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
23. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
24. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
25. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
26. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
27. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
28. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
29. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
30. Benner P, Saak J. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations. Technical Report SPP1253-090, Deutsche Forschungsgemeinschaft - Priority Program 1253, 2010.
15. Jbilou K. Block Krylov subspace methods for large algebraic Riccati equations. *Numerical Algorithms* 2003; 34: 339–353.
16. Heyouni M, Jbilou K. An extended block Krylov method for large-scale continuous-time algebraic Riccati equations. *ETNA* 2008-2009; 33: 53–62.
17. Bouhamidi A, Heyouni M, Jbilou K. Block Arnoldi-based methods for large scale discrete-time algebraic Riccati equations. *Journal of Computational and Applied Mathematics* 2011; 236(6): 1531–1542.
18. Simoncini V, Szyld DB, Monsalve M. On two numerical methods for the solution of large-scale algebraic Riccati equations. *IMA Journal of Numerical Analysis*, posted on 2013. DOI: 10.1093/imanum/drt015.
19. Simoncini V. Computational methods for linear matrix equations. *Technical Report*, Università di Bologna: Bologna, March 2013.
20. Li T, Wah Chu EK, Lin WW, Weng P. Solving large-scale continuous-time algebraic Riccati equations by doubling. *Journal of Computational and Applied Mathematics* 2013; 237(1): 373–383.
21. Byers R. Solving the algebraic Riccati equation with the matrix sign function. *Linear Algebra and Applications* 1987; 85: 267–279.
22. Arnold WF, III, Laub AJ. Generalized eigenproblem algorithms and software for algebraic Riccati equations. *Proceedings of the IEEE* 1984; 72(12): 1746–1754.
23. Laub A. Invariant subspace methods for the numerical solution of Riccati equations. In *The Riccati Equation*, Bittanti S, Laub A, Willems J (eds). Springer-Verlag: Berlin, 1991; 163–196.
24. Benner P, Fassbender H. An implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem. *Linear Algebra and its Applications* 1997; 263: 75–111.
25. Amodei L, Buchot JM. An invariant subspace method for large-scale algebraic Riccati equation. *Applied Numerical Mathematics* 2010; 60: 1067–1082.
26. Byers R. A Hamiltonian QR algorithm. *SIAM Journal on Scientific and Statistical Computing* 1986; 7(1): 212–229.
27. The MathWorks. Inc. *MATLAB 7*, September 2004.
28. Li JR, White J. Low-rank solutions of Lyapunov equations. *SIAM Journal on Matrix Analysis and Applications* 2002; 24(1): 260–280.
29. Penzl T. A cyclic low-rank Smith method for large sparse Lyapunov equations. *SIAM Journal on Scientific Computing* 1999/00; 21(4): 1401–1418. (electronic).
30. Lin Y. On the numerical solution of large scale Lyapunov and Riccati matrix equations. *PhD thesis*, School of Mathematical Sciences, China, 2013. In preparation.
31. Golub GH, Van Loan CF. *Matrix Computations* (3rd edn.) Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press: Baltimore, MD, 1996.
32. Lin Y, Simoncini V. A new subspace iteration method for the algebraic Riccati equation. *Arxiv manuscript arXiv:1307.3843*, 15 July 2013.
33. Lu A, Wachspress EL. Solution of Lyapunov equations by alternating direction implicit iteration. *Computers & Mathematics with Applications* 1991; 21(9): 43–58.
34. Ellner NS, Wachspress EL. Alternating direction implicit iteration for systems with complex spectra. *SIAM Journal on Numerical Analysis* 1991; 23(3): 859–870.
35. Benner P, Kürschner P, Saak J. Efficient handling of complex shift parameters in the low-rank Cholesky factor ADI method. *Numerical Algorithms* 2013; 62(2): 225–251.
36. Penzl T. Lyapack users guide. *Technical Report SFB393/00-33*, TU Chemnitz, 09107 Chemnitz, D, 2000. (Available from: http://www.tu-chemnitz.de/sfb393/00-33/pr.html).
37. Druskin V, Knizhnerman L, Simoncini V. Analysis of the rational Krylov subspace and ADI methods for solving the Lyapunov equation. *SIAM Journal on Numerical Analysis* 2011; 49: 1875–1898.
38. Druskin V, Simoncini V. Adaptive rational Krylov subspaces for large-scaled dynamical systems. *Systems and Control Letters* 2011; 60: 546–560.
39. Gugercin S, Antoulas AC, Beattie C. $H_2$ model reduction for large-scale linear dynamical systems. *SIAM Journal on Matrix Analysis and Applications* 2008; 30: 609–638.
40. Benchmark Collection. *Oberwolfach model reduction benchmark collection*, 2003. (Available from: http://www.imtek.de/simulation/benchmark).
41. Trefethen LN, Embree M. *Spectra and Pseudospectra. The Behavior of Nonnormal Matrices and Operators*. Princeton University Press: Princeton, 2005.