SIAM J. Sci. Comput. © 2019 Society for Industrial and Applied Mathematics
Vol. 41, No. 4, pp. A2193–A2211

NONLINEAR LEAST-SQUARES APPROACH FOR LARGE-SCALE ALGEBRAIC RICCATI EQUATIONS[*]

KHALIDE JBILOU[†] AND MARCOS RAYDAN[‡]

Abstract. We propose a new optimization approach for solving large-scale continuous-time algebraic Riccati equations with a low-rank right-hand side. First, we project the problem onto a Krylov-type low-dimensional subspace. Then, instead of forcing the orthogonality conditions related to the Galerkin strategy, we minimize the residual to get a low-dimensional nonlinear matrix least-squares problem that will be solved to obtain an approximate factorized solution of the initial Riccati equation. To solve the low-order minimization problems, we propose a globalized Gauss–Newton matrix approach that exhibits a smooth convergence behavior and that guarantees global convergence to stationary points. This novel procedure involves the solution of a linear symmetric matrix problem per iteration that will be solved by direct or preconditioned iterative matrix methods. To illustrate the behavior of the combined scheme, we present numerical results on some test problems.

Key words. large-scale Riccati equations, Galerkin-projection methods, extended block Arnoldi subspaces, rational Krylov subspaces, Gauss–Newton method, global conjugate gradient method

AMS subject classifications. 65F10, 65F30, 65K10, 49N10

DOI. 10.1137/18M1198922

1. Introduction.

We consider the continuous-time algebraic Riccati equation (CARE) of the form

$$A^T X + XA - XBB^T X + C^T C = 0, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ is nonsingular, $B \in \mathbb{R}^{n\times p}$, and $C \in \mathbb{R}^{s\times n}$. The matrices B and C are assumed to be of full rank with $p \ll n$ and $s \ll n$. Riccati equations play a fundamental role in many areas such as control theory, model reduction, differential equations, and robust control problems [2, 15, 37, 45]. For a review, we refer the reader to [1, 9, 13, 32].

In many applications, such as in optimal control problems, we need to determine the stabilizing solution X of (1.1); i.e., X is symmetric positive semidefinite and the resulting closed-loop matrix $A - BB^T X$ is stable (a square matrix is stable if all its eigenvalues are in the open left half-plane). The stabilizing solution X exists and is unique under some specific assumptions on the three given matrices involved in (1.1); see, e.g., [10, 32]. For a review on the theoretical aspects of Riccati equations and the conditions under which solutions exist for (1.1), we recommend [1, 9, 10, 12, 32].

For small to medium-sized problems, many numerical methods for solving CARE (1.1) have been developed. The standard computational methods are based on the Schur and structure-preserving Schur methods [33, 43], matrix sign function methods [30, 39], and Newton-based methods [3, 11, 22, 31, 32, 42]. In practical applications, the matrices A, B, and C are obtained from the discretization of operators defined on infinite dimensional subspaces. Moreover, the matrix A is in general sparse, banded,

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section July 5, 2018; accepted for publication (in revised form) April 9, 2019; published electronically July 16, 2019.
https://doi.org/10.1137/18M1198922

Funding: The work of the second author was financially supported by the Portuguese Foundation for Science and Technology (FCT) through the project UID/MAT/00297/2019 (CMA).

[†]Laboratoire LMPA, 50 rue F. Buisson, ULCO, 62228, Calais, France (jbilou@univ-littoral.fr).
[‡]Centro de Matemática e Aplicações, FCT, UNL, 2829-516 Caparica, Portugal (m.raydan@fct.unl.pt).

and large. For such problems, several different approaches have been developed in the last few years. A well-known option is to use the so-called inexact Newton method, for which the large-scale Lyapunov equation at each outer iteration is solved approximately using iterative inner schemes; see, e.g., [7, 11, 20]. Another option for the outer iterations is the well-known low-rank Cholesky factorized Newton (LRCF-Newton) method [8], for which the large-scale Lyapunov matrix equation is solved via alternating direction implicit (ADI) based methods. The direct application of low-rank subspace ADI-type iterations has also been recently studied; see, e.g., [4, 5, 35, 36].

Another recent approach is based on projections onto Krylov-type subspaces methods using standard, rational, global, or extended block Arnoldi processes [24, 26, 27, 28, 29, 41, 42]. At every low-order Krylov subspace they use the so-called Galerkin orthogonality conditions to obtain an approximate solution on that subspace. In a recent paper [42], Simoncini, Szyld, and Monsalve observed that in practice the projection approach is superior in required computational work and storage when compared with the inexact Newton approach; see also [6]. However, there are two drawbacks when combining projection-type methods and the Galerkin strategy. First, although the initial conditions ensure the existence of the stabilizing solution, we have no guarantee that such a solution exists for the projected low-order subproblems that need to be solved. Second, the convergence of the projecting scheme may exhibit an erratic behavior after a finite number of iterations.

With the aim of avoiding these drawbacks, we present a projection-minimization method that allows us to compute factorized approximations to the stabilizing solution of (1.1). We project the original problem onto a sequence of nested Krylov-type subspaces. Then we minimize the residual at each subspace to get a low-dimensional nonlinear matrix least-squares problem that will be solved to obtain an approximate solution to the initial Riccati equation. To solve the low-order nonlinear matrix problems, we propose and analyze a globalized Gauss–Newton approach that exhibits a smooth convergence behavior. This procedure involves the solution of a linear symmetric matrix problem per iteration that will be solved by direct or (preconditioned) iterative matrix methods. In the linear case, for solving Lyapunov equations, projection-minimization schemes which require the solution of linear matrix least-squares problems have been already proposed [26, 34, 38].

The rest of the paper is organized as follows. In section 2, we present and describe the Krylov-type schemes that play a key role in our work, and we also recall the Galerkin-projection strategy for the computation of a low-dimensional approximate solution at each Krylov subspace. In section 3, we describe the minimum residual least-squares approach to be used instead of the Galerkin strategy, and we develop a matrix version of the Gauss–Newton approach for solving the nonlinear least-squares problems at every external projection step. We also add a globalization strategy, and we establish its convergence properties. In section 4, we develop a direct scheme based on the Kronecker product to solve the linear problems that appear at each Gauss–Newton iteration, as well as a preconditioned global conjugate gradient scheme for large-scale problems. We also propose a suitable preconditioning strategy. In section 5, we show how to extract a factorized form of the approximate solution, and we present a general framework for the whole scheme which includes both the Galerkin and the Gauss–Newton approaches. In section 6, we present our numerical experiments and results. In section 7, we present some final remarks.

Throughout this paper, we use the following notation: The Frobenius inner product of two matrices W_1 and W_2 is given by $\langle W_1, W_2\rangle_F = Tr(W_1^T W_2)$, where Tr denotes the trace of a matrix. The corresponding norm is the well-known Frobenius norm. The matrices I_r and $O_{r\times l}$ will denote the identity of size $r \times r$ and the null matrix of size $r \times l$, respectively.

2. Krylov-type schemes and the Galerkin-projection method.

The Galerkin-projection strategy generates a sequence of nested low-dimensional spaces, $\mathcal{K}_m \subseteq \mathcal{K}_{m+1}$, $m \geq 1$, and finds an approximate projected solution in each one of them, until a satisfactory low-rank approximate solution of (1.1) is found. Methods following this strategy differ in the way the sequence of the nested subspaces $\mathcal{K}_m$ is built. We focus our attention on two recently developed effective options: the extended block Arnoldi (EBA) method (see [16, 24, 40, 42]) and the rational Krylov subspace method (RKSM) (see [35, 41, 42]).

First, we recall the key features of the EBA method when applied to the pair (A^T, C^T) (the algorithmic steps of EBA for solving CARE are fully described in [24]). At step $m \geq 1$, it has built in an incremental way an orthonormal basis $\{V_1, V_2, \ldots, V_m\}$ ($V_i \in \mathbb{R}^{n\times 2s}$) of the extended Krylov subspace

$$\mathcal{K}_m^E(A^T, C^T) = \text{Range}(C^T, A^T C^T, \ldots, (A^T)^{m-1} C^T, A^{-T} C^T, \ldots, (A^{-T})^m C^T).$$

Let us denote the block matrix $\mathbb{V}_m = [V_1, V_2, \ldots, V_m]$, and let $\mathcal{T}_m = \mathbb{V}_m^T A^T \mathbb{V}_m \in \mathbb{R}^{2ms\times 2ms}$ be the restriction of the matrix A^T to the subspace $\mathcal{K}_m^E(A^T, C^T)$. Then it can be shown that $\mathcal{T}_m$ is a block upper Hessenberg matrix with $2s \times 2s$ blocks; see [24, 40]. The following algebraic relations are satisfied:

$$A^T \mathbb{V}_m = \mathbb{V}_{m+1} \widetilde{\mathcal{T}}_m = \mathbb{V}_m \mathcal{T}_m + V_{m+1} T_{m+1,m} \widetilde{E}_m^T, \tag{2.1}$$

where $\widetilde{\mathcal{T}}_m = \mathbb{V}_{m+1}^T A^T \mathbb{V}_m$, $T_{m+1,m}$ is the bottom-right $2s \times 2s$ block of $\widetilde{\mathcal{T}}_m$, and the matrix $\widetilde{E}_m = [O_{2s\times 2(m-1)s}, I_{2s}]^T$ is formed with the last $2s$ columns of the $2ms \times 2ms$ identity matrix I_{2ms}. The $(2ms+2s)\times 2ms$ block Hessenberg matrix $\widetilde{\mathcal{T}}_m$ can be written as

$$\widetilde{\mathcal{T}}_m = \begin{pmatrix} \mathcal{T}_m \\ h_m \end{pmatrix}, \tag{2.2}$$

where $h_m = T_{m+1,m} E_m^T$ represents the last $2s$ rows of $\widetilde{\mathcal{T}}_m$.

Concerning RKSM, at step $m \geq 1$ it has built in an incremental way an orthonormal basis $\{V_1, V_2, \ldots, V_m\}$ ($V_i \in \mathbb{R}^{n\times s}$) of the rational Krylov subspace

$$\mathcal{K}_m^R(A^T, C^T, \bar{s}) = \text{Range}\left(C^T, (A^T - \bar{s}_2 I)^{-1} C^T, \ldots, \prod_{j\leq m} (A^T - \bar{s}_j I)^{-1} C^T\right),$$

where the vector $\bar{s}$ contains the (possibly complex) parameters $\bar{s}_i$, $2 \leq i \leq m$, which are computed dynamically during the process such that $(A^T - \bar{s}_i I)$ is nonsingular for all i, as described in [17]. The particular dynamical choice of the parameters $\bar{s}_i$, described in [17], has proved to be successful in the majority of cases; see also [41, 42]. For this option, the block matrix $\mathbb{V}_m = [V_1, V_2, \ldots, V_m] \in \mathbb{R}^{n\times ms}$, and $\mathcal{T}_m = \mathbb{V}_m^T A^T \mathbb{V}_m \in \mathbb{R}^{ms\times ms}$ is the restriction of the matrix A^T to the subspace $\mathcal{K}_m^R(A^T, C^T, \bar{s})$. The algorithmic steps and the features that characterize RKSM, including the structure of the associated reduced matrices $\widetilde{\mathcal{T}}_m$ in (2.2) and the specific details of the corresponding recursion given in (2.1) for EBA, are fully described in [41, section 5]; see also [17]. In particular, we remark that the dimensions $2s$ and $2ms$ in EBA are reduced to s and ms, respectively, in RKSM.

Let us now recall the Galerkin-projection method introduced in [24] for the computation of approximate solutions to (1.1). At step m, we project the initial problem onto the considered Krylov-type subspace ($\mathcal{K}_m^E(A^T, C^T)$ or $\mathcal{K}_m^R(A^T, C^T, \bar{s})$) to obtain $\mathbb{V}_m$, $\mathcal{T}_m$, and $\widetilde{\mathcal{T}}_m$. Then the approximation X_m^G is defined as follows:

$$X_m^G = \mathbb{V}_m Y_m^G \mathbb{V}_m^T, \tag{2.3}$$

where the matrix Y_m^G is determined such that the residual $R(X_m^G) := A^T X_m^G + X_m^G A - X_m^G BB^T X_m^G + C^T C$ satisfies the Galerkin condition

$$\mathbb{V}_m^T R(X_m^G) \mathbb{V}_m = 0. \tag{2.4}$$

From (2.3) and (2.4), it can be shown that Y_m^G is the solution of the low-dimensional Riccati equation

$$\mathcal{T}_m Y_m^G + Y_m^G \mathcal{T}_m^T - Y_m^G B_m B_m^T Y_m^G + C_m C_m^T = 0, \tag{2.5}$$

where $\mathcal{T}_m = \mathbb{V}_m^T A^T \mathbb{V}_m$, $C_m = \mathbb{V}_m^T C^T$, and $B_m = \mathbb{V}_m^T B$. Conditions to guarantee the existence of a stabilizing solution Y_m^G of (2.5) can be found in [41].

From now on, we will use two positive integers to identify the two key dimensions that appear in the construction of the orthonormal basis of any of the considered Krylov subspaces: $\widehat{s}$, which indicates the number of columns of V_i, for all i, and $\widehat{ms}$, which indicates the size of the square matrix $\mathcal{T}_m$. In particular, $\widehat{s} = 2s$ and $\widehat{ms} = 2ms$ when using EBA, and $\widehat{s} = s$ and $\widehat{ms} = ms$ when using RKSM. Nevertheless, we would like to mention that any of the forthcoming algorithmic ideas and results could also be applied when any other Krylov-type subspace is used, as long as the subspace satisfies the basic properties described above.

Let us now present a useful and well-known result that allows us to compute the residual without computing the approximate solution X_m^G; see, e.g., [24, 26, 40].

THEOREM 2.1. *Let X_m^G be the obtained approximate Galerkin-projection solution; then we have*

$$\|R(X_m^G)\|_2 = \|T_{m+1,m} \tilde{Y}_m^G\|_2, \tag{2.6}$$

where $\tilde{Y}_m^G$ is the matrix corresponding to the last $\widehat{s}$ rows of Y_m^G.

3. The minimum residual approach and the Gauss–Newton method.

We now describe a new projection-minimization approach for solving (1.1). Let us recall that for the Galerkin-projection approach, we consider approximate solutions X_m^G of (1.1) given by (2.3). From now on, we drop the superscript G in X_m^G and Y_m^G, and instead of using a Galerkin condition on the residual for every m, Y_m will be computed by solving the minimization problem

$$\min_{X_m = \mathbb{V}_m Y_m \mathbb{V}_m^T} \left\| A^T X_m + X_m A - X_m BB^T X_m + C^T C \right\|_F, \tag{3.1}$$

where $Y_m = \mathbb{V}_m^T X_m \mathbb{V}_m$. The next result allows us to transform (3.1) into an equivalent minimization problem with small dimension. Similar results are obtained when minimal residual techniques are applied to linear matrix problems; see, e.g., [34, 40].

THEOREM 3.1. *The minimization problem* (3.1) *is equivalent to the following one:*

$$\min_{Y_m} \left\| \widetilde{\mathcal{T}}_m Y_m \widetilde{I} + \widetilde{I}^T Y_m \widetilde{\mathcal{T}}_m^T - \widetilde{I}^T Y_m B_m B_m^T Y_m \widetilde{I} + \begin{bmatrix} R_C R_C^T & 0 \\ 0 & 0 \end{bmatrix} \right\|_F, \tag{3.2}$$

where $\widetilde{I} = [I_{\widehat{ms}} \quad O_{\widehat{ms} \times \widehat{s}}]$ and $V_1 R_C$ is the QR decomposition of C^T ($V_1 R_C = C^T$).

Proof. Setting $X_m = \mathbb{V}_m Y_m \mathbb{V}_m^T$, we get the following equalities:

$$\begin{aligned}\|R(X_m)\|_F &= \|A^T X_m + X_m A - X_m B B^T X_m + C^T C\|_F \\ &= \|A^T \mathbb{V}_m Y_m \mathbb{V}_m^T + \mathbb{V}_m Y_m \mathbb{V}_m^T A - \mathbb{V}_m Y_m B_m B_m^T Y_m \mathbb{V}_m^T + V_1 R_C R_C^T V_1^T\|_F.\end{aligned}$$

Now, using the fact that $A^T \mathbb{V}_m = \mathbb{V}_{m+1} \widetilde{\mathcal{T}}_m$ and that the matrix $\mathbb{V}_{m+1}$ is orthonormal, it follows that

$$\|R(X_m)\|_F = \left\| \widetilde{\mathcal{T}}_m Y_m \widetilde{I} + \widetilde{I}^T Y_m \widetilde{\mathcal{T}}_m^T - \widetilde{I}^T Y_m B_m B_m^T Y_m \widetilde{I} + \left[\begin{array}{cc} R_C R_C^T & 0 \\ 0 & 0 \end{array}\right] \right\|_F. \quad \square \tag{3.3}$$

Notice that problem (3.2) can be written as

$$\min_Y f(Y), \tag{3.4}$$

where the function f is given by

$$f(Y) = \|\mathcal{F}(Y)\|_F^2 = \langle \mathcal{F}(Y), \mathcal{F}(Y) \rangle_F,$$

and the function $\mathcal{F}$ is defined as

$$\mathcal{F}(Y) = \widetilde{\mathcal{T}}_m Y \widetilde{I} + \widetilde{I}^T Y \widetilde{\mathcal{T}}_m^T + \widetilde{I}^T Y \mathcal{B}_m Y \widetilde{I} + \widetilde{\mathcal{R}}, \tag{3.5}$$

where

$$\mathcal{B}_m = -B_m B_m^T \quad \text{and} \quad \widetilde{\mathcal{R}} = \left[\begin{array}{cc} R_C R_C^T & 0 \\ 0 & 0 \end{array}\right].$$

Remark 3.1. Any matrix Y (positive semidefinite or not) for which $\mathcal{F}(Y) = 0$ is a global minimizer of $f(Y)$. Therefore, even when (2.5) does not have a stabilizing solution at a certain low-dimensional subspace, the optimization problem (3.4) can have several global solutions.

Let us now recall that, using properties of the trace operator, for any matrices W_1, W_2, and W_3, it follows that

$$\langle W_1, W_2 W_3 \rangle_F = \langle W_2^T W_1, W_3 \rangle_F = \langle W_1 W_3^T, W_2 \rangle_F. \tag{3.6}$$

We will now develop a version of the Gauss–Newton approach for solving the nonlinear least-squares matrix problem (3.4) where the nonlinear matrix map $\mathcal{F}(Y)$ is given by (3.5). For that we need the first and the second derivatives of $f(Y)$, which can be obtained as follows. Consider the auxiliary function $g : \mathbb{R} \to \mathbb{R}$, given by $g(t) = f(Y + tZ)$, for any arbitrary matrix Z. From basic calculus we know that $g'(0) = \langle f'(Y), Z \rangle_F$ and also that $g''(0) = \langle f''(Y)Z, Z \rangle_F$. After simple manipulations, using (3.6), we obtain

$$g'(0) = 2\langle \mathcal{F}(Y), \mathcal{F}'(Y)Z \rangle_F = 2\langle \mathcal{F}'(Y)^T \mathcal{F}(Y), Z \rangle_F,$$

and also that

$$\begin{aligned} g''(0) &= 2\langle \mathcal{F}'(Y)Z, \mathcal{F}'(Y)Z \rangle_F + 2\langle \mathcal{F}(Y)Z, \mathcal{F}''(Y)Z \rangle_F \\ &= 2\langle \mathcal{F}'(Y)^T (\mathcal{F}'(Y)Z), Z \rangle_F + 2\langle \mathcal{F}''(Y)^T (\mathcal{F}(Y)Z), Z \rangle_F. \end{aligned}$$

Therefore,

$$f'(Y) = 2\mathcal{F}'(Y)^T \mathcal{F}(Y) \quad \text{and} \quad f''(Y) = 2\mathcal{F}'(Y)^T \mathcal{F}'(Y) + 2\mathcal{F}''(Y)^T \mathcal{F}(Y).$$

Let us now consider the following expansion:

$$\begin{aligned}\mathcal{F}(Y+Z) &= \widetilde{\mathcal{T}}_m(Y+Z)\widetilde{I} + \widetilde{I}^T(Y+Z)\widetilde{\mathcal{T}}_m^T + \widetilde{I}^T(Y+Z)\mathcal{B}_m(Y+Z)\widetilde{I} + \widetilde{\mathcal{R}}\\ &= \mathcal{F}(Y) + \widetilde{\mathcal{T}}_m Z\widetilde{I} + \widetilde{I}^T Z\widetilde{\mathcal{T}}_m^T + \widetilde{I}^T Y\mathcal{B}_m Z\widetilde{I} + \widetilde{I}^T Z\mathcal{B}_m Y\widetilde{I} + O(Z^2).\end{aligned}$$

Hence, the Fréchet derivative $\mathcal{F}'(Y)$ applied to a general matrix Z is given by

$$\begin{aligned}\mathcal{F}'(Y)Z &= \widetilde{\mathcal{T}}_m Z\widetilde{I} + \widetilde{I}^T Z\widetilde{\mathcal{T}}_m^T + \widetilde{I}^T Y\mathcal{B}_m Z\widetilde{I} + \widetilde{I}^T Z\mathcal{B}_m Y\widetilde{I}\\ &= \widetilde{I}^T Z(\widetilde{\mathcal{T}}_m^T + \mathcal{B}_m Y\widetilde{I}) + (\widetilde{\mathcal{T}}_m + \widetilde{I}^T Y\mathcal{B}_m)Z\widetilde{I}.\end{aligned}$$

Notice that $\mathcal{B}_m$ is symmetric, and if Y is symmetric, we obtain that

$$\mathcal{F}'(Y)Z = \widetilde{I}^T Z\mathcal{L}(Y)^T + \mathcal{L}(Y)Z\widetilde{I},$$

where $\mathcal{L}(Y)$ is the matrix operator defined by

$$\mathcal{L}(Y) = \widetilde{\mathcal{T}}_m + \widetilde{I}^T Y\mathcal{B}_m. \tag{3.7}$$

The operator $\mathcal{L}$ given by (3.7) can be written as

$$\mathcal{L}(Y) = \begin{pmatrix}\mathcal{T}_m\\ h_m\end{pmatrix} + \begin{pmatrix} I\\ 0\end{pmatrix} Y\mathcal{B}_m = \begin{pmatrix}\widehat{\mathcal{L}}(Y)\\ h_m\end{pmatrix}, \tag{3.8}$$

where h_m is as defined in (2.2) and $\widehat{\mathcal{L}}(Y) = \mathcal{T}_m + Y\mathcal{B}_m$.

The full Newton method for solving (3.4), starting at a given symmetric $Y^{(0)}$, requires the solution of the linear matrix problem $f''(Y^{(k)})S^{(k)} = -f'(Y^{(k)})$ to obtain the step $S^{(k)}$ at every iteration, and then computes the next iterate as $Y^{(k+1)} = Y^{(k)} + S^{(k)}$. Notice that it involves both terms in the expression of $f''(Y) = 2\mathcal{F}'(Y)^T\mathcal{F}'(Y) + 2\mathcal{F}''(Y)^T\mathcal{F}(Y)$ for solving the linear problem. The Gauss–Newton method, on the other hand, simply discards the second-order terms $\mathcal{F}''(Y)$ in the second derivative of $f(Y)$; i.e., it does not take into account the term $2\mathcal{F}''(Y)^T\mathcal{F}(Y)$. The main motivation for using this approach is that those second-order terms will be close to zero during the convergence process, if $\|\mathcal{F}(Y^*)\|_F$ is near zero at the limit point Y^*, and will be exactly zero at the limit point for zero residual problems, i.e., if $\mathcal{F}(Y^*) = 0$; see, e.g., [14, 23]. Hence for the Gauss–Newton approach, the linear matrix problem that needs to be solved per iteration to obtain the step matrix $S^{(k)}$ reduces to solving for S the problem

$$\mathcal{F}'(Y^{(k)})^T(\mathcal{F}'(Y^{(k)})S) = -\mathcal{F}'(Y^{(k)})^T(\mathcal{F}(Y^{(k)})), \tag{3.9}$$

which in our particular setting is given by

$$\widetilde{I}(\mathcal{F}'(Y^{(k)})S)\mathcal{L}(Y^{(k)}) + \mathcal{L}(Y^{(k)})^T(\mathcal{F}'(Y^{(k)})S)\widetilde{I}^T = -\widetilde{I}\mathcal{F}(Y^{(k)})\mathcal{L}(Y^{(k)}) - \mathcal{L}(Y^{(k)})^T\mathcal{F}(Y^{(k)})\widetilde{I}^T.$$

By recalling that $\mathcal{F}'(Y^{(k)})S = \widetilde{I}^T S\mathcal{L}(Y^{(k)})^T + \mathcal{L}(Y^{(k)})S\widetilde{I}$, it becomes

$$\begin{aligned}&\widetilde{I}\left[\widetilde{I}^T S\mathcal{L}(Y^{(k)})^T + \mathcal{L}(Y^{(k)})S\widetilde{I}\right]\mathcal{L}(Y^{(k)}) + \mathcal{L}(Y^{(k)})^T\left[\widetilde{I}^T S\mathcal{L}(Y^{(k)})^T + \mathcal{L}(Y^{(k)})S\widetilde{I}\right]\widetilde{I}^T\\ &\qquad = -\widetilde{I}\mathcal{F}(Y^{(k)})\mathcal{L}(Y^{(k)}) - \mathcal{L}(Y^{(k)})^T\mathcal{F}(Y^{(k)})\widetilde{I}^T.\end{aligned} \tag{3.10}$$

Let us see how to derive a simplified expression of the linear matrix equation (3.10). Since $\widetilde{I}\widetilde{I}^T = I$, The linear matrix equation (3.10) can be written as

$$\mathcal{L}(Y^{(k)})^T\mathcal{L}(Y^{(k)})S + S\mathcal{L}(Y^{(k)})^T\mathcal{L}(Y^{(k)}) + \widetilde{I}\mathcal{L}(Y^{(k)})S\widetilde{I}\mathcal{L}(Y^{(k)}) + \mathcal{L}(Y^{(k)})^T\widetilde{I}^T S\mathcal{L}(Y^{(k)})^T\widetilde{I}^T = \mathcal{C}_k,$$

where $\mathcal{C}_k = -\widetilde{I}\mathcal{F}(Y^{(k)})\mathcal{L}(Y^{(k)}) - \mathcal{L}(Y^{(k)})^T\mathcal{F}(Y^{(k)})\widetilde{I}^T$. Using the preceding notation, this matrix linear equation can be written as

$$\mathcal{M}_k(S) = \mathcal{C}_k, \tag{3.11}$$

where

(3.12)

$$\mathcal{M}_k(S) = \mathcal{L}(Y^{(k)})^T\mathcal{L}(Y^{(k)})S + S\mathcal{L}(Y^{(k)})^T\mathcal{L}(Y^{(k)}) + \widehat{\mathcal{L}}(Y^{(k)})S\widehat{\mathcal{L}}(Y^{(k)}) + \widehat{\mathcal{L}}(Y^{(k)})^T S\widehat{\mathcal{L}}(Y^{(k)})^T.$$

Once (3.11) is solved for $S^{(k)}$, the next iterate is then obtained as $Y^{(k+1)} = Y^{(k)} + S^{(k)}$.

Our next result is concerned with the symmetry of the matrix $Y^{(k)}$ for all k.

PROPOSITION 3.2. *If the linear operator $\mathcal{M}_k$ is invertible for all k, and $Y^{(0)}$ is symmetric, then $Y^{(k)}$ is symmetric for all $k \geq 0$.*

Proof. We will prove that $Y^{(k)}$ is symmetric for all $k \geq 0$ by induction on k. The starting matrix is given such that $Y^{(0)} = (Y^{(0)})^T$. Let us now assume that $Y^{(k_1)}$ is symmetric for an integer $k_1 \geq 1$. Hence, using (3.5), $\mathcal{F}(Y^{(k_1)})$ is symmetric, and so $\mathcal{C}_{k_1}$ is also symmetric. Therefore, from (3.11) it follows that $(\mathcal{M}_{k_1}(S^{(k_1)}))^T = \mathcal{M}_{k_1}(S^{(k_1)})$. Now, using (3.12), we have that $(\mathcal{M}_{k_1}(S^{(k_1)}))^T = \mathcal{M}_{k_1}((S^{(k_1)})^T)$, and by the linearity of $\mathcal{M}_{k_1}$ we obtain that

$$\mathcal{M}_{k_1}(S^{(k_1)} - (S^{(k_1)})^T) = 0.$$

Since $\mathcal{M}_{k_1}$ is invertible, $(S^{(k_1)})^T = S^{(k_1)}$. Finally, recalling that $Y^{(k_1+1)} = Y^{(k_1)} + S^{(k_1)}$, we obtain that $Y^{(k_1+1)}$ is symmetric, and the result is established. ∎

Remark 3.2. From the proof of Proposition 3.2 we obtain that the matrices $S^{(k)}$ are also symmetric. However, we would like to mention that even if $Y^{(k)}$ is positive semidefinite for some k, the matrix $S^{(k)}$ that solves (3.11) may be indefinite, as well as the next iterate $Y^{(k+1)} = Y^{(k)} + S^{(k)}$. In that case, if convergence is observed, the limit $Y_m = \lim_{k\to\infty} Y^{(k)}$ might be an indefinite matrix (see also Remark 3.1).

Our next result establishes some key properties of the linear operator $\mathcal{M}_k$ that will guarantee a unique solution of (3.11) for every k and also that will allow us to use a global conjugate gradient method with a suitable preconditioner for solving the linear matrix problems.

THEOREM 3.3. *At any iteration k, the linear operator $\mathcal{M}_k$ defined by (3.12) is self-adjoint (symmetric) and positive; i.e., for all matrices W_1, W_2, and symmetric matrix $W \neq 0$,*

$$\langle \mathcal{M}_k(W_1), W_2\rangle_F = \langle W_1, \mathcal{M}_k(W_2)\rangle_F \quad \textit{and} \quad \langle \mathcal{M}_k(W), W\rangle_F \geq 0.$$

Moreover, if $\widehat{\mathcal{L}}(Y^{(k)})$ is stable, then $\mathcal{M}_k$ is strictly positive; i.e., $\langle \mathcal{M}_k(W), W\rangle_F > 0$.

Proof. Using the trace operator properties (3.6), we obtain, for any two matrices W_1 and W_2, that $\langle \mathcal{L}(Y^{(k)})^T\mathcal{L}(Y^{(k)})W_1, W_2\rangle_F = \langle W_1, \mathcal{L}(Y^{(k)})^T\mathcal{L}(Y^{(k)})W_2\rangle_F$ and also that

$$\langle \widehat{\mathcal{L}}(Y^{(k)})W_1\widehat{\mathcal{L}}(Y^{(k)}), W_2\rangle_F = \langle W_1, \widehat{\mathcal{L}}(Y^{(k)})^T W_2\widehat{\mathcal{L}}(Y^{(k)})^T\rangle_F.$$

Therefore, applying these properties in a systematic way using (3.12), it follows that

$$\langle \mathcal{M}_k(W_1), W_2\rangle_F = \langle W_1, \mathcal{M}_k(W_2)\rangle_F,$$

and the linear operator $\mathcal{M}_k$ is self-adjoint.

For the second part, let us consider the equivalent expression for $\mathcal{M}_k$ given by (3.10). Once again, applying in a systematic way the trace operator properties (3.6), we have that for any symmetric $W \neq 0$

$$\begin{aligned}\langle \mathcal{M}_k(W), W\rangle_F &= \left\langle \widetilde{I}\left[\widetilde{I}^T W\mathcal{L}(Y^{(k)})^T + \mathcal{L}(Y^{(k)})W\widetilde{I}\right]\mathcal{L}(Y^{(k)}), W\right\rangle_F \\ &\quad + \left\langle \mathcal{L}(Y^{(k)})^T\left[\widetilde{I}^T W\mathcal{L}(Y^{(k)})^T + \mathcal{L}(Y^{(k)})W\widetilde{I}\right]\widetilde{I}^T, W\right\rangle_F \\ &= \langle \widetilde{I}^T W\mathcal{L}(Y^{(k)})^T + \mathcal{L}(Y^{(k)})W\widetilde{I}\,,\, \widetilde{I}^T W\mathcal{L}(Y^{(k)})^T + \mathcal{L}(Y^{(k)})W\widetilde{I}\rangle_F \\ &= \|\widetilde{I}^T W\mathcal{L}(Y^{(k)})^T + \mathcal{L}(Y^{(k)})W\widetilde{I}\|_F^2 \geq 0,\end{aligned}$$

which proves that $\mathcal{M}_k$ is a positive linear operator.

Finally, using the structure of $\widetilde{I}$ and (3.8), $\widetilde{I}^T W\mathcal{L}(Y^{(k)})^T + \mathcal{L}(Y^{(k)})W\widetilde{I}$ can be written as a 2×2 block matrix whose left upper block is $W\widehat{\mathcal{L}}(Y^{(k)})^T + \widehat{\mathcal{L}}(Y^{(k)})W$. For $W \neq 0$, this matrix cannot be zero; for otherwise, using the Kronecker form, it holds that

$$\left[\widehat{\mathcal{L}}(Y^{(k)}) \otimes I + I \otimes \widehat{\mathcal{L}}(Y^{(k)})\right] vec(W) = 0, \tag{3.13}$$

where $vec(W)$ is obtained by stacking the columns of W from the left-top. Now, let $\{\mu_1, \ldots, \mu_{\widehat{ms}}\}$ be the eigenvalues of $\widehat{\mathcal{L}}(Y^{(k)})$. The eigenvalues of the Kronecker matrix in (3.13) are $\mu_i + \mu_j$ for all possible combinations of $1 \leq i \leq \widehat{ms}$ and $1 \leq j \leq \widehat{ms}$ (see [44, Theorem 6.19]). Thus, since $\widehat{\mathcal{L}}(Y^{(k)})$ is stable, all the eigenvalues of the Kronecker matrix in (3.13) have negative real part, and so it is a nonsingular matrix. Hence, $vec(W) = 0$, which contradicts that $W \neq 0$. Therefore, using properties of the Frobenius norm, we obtain

$$\langle \mathcal{M}_k(W), W\rangle_F = \|\widetilde{I}^T W\mathcal{L}(Y^{(k)})^T + \mathcal{L}(Y^{(k)})W\widetilde{I}\|_F^2 \geq \|W\widehat{\mathcal{L}}(Y^{(k)})^T + \widehat{\mathcal{L}}(Y^{(k)})W\|_F^2 > 0,$$

and the operator $\mathcal{M}_k$ is strictly positive. ∎

Remark 3.3. In Theorem 3.3, the stability of $\widehat{\mathcal{L}}(Y^{(k)})$ is a sufficient condition to guarantee that $\mathcal{M}_k$ is strictly positive. However, it is not a necessary condition. Indeed, even if $\widehat{\mathcal{L}}(Y^{(k)})$ is not stable at some iteration k, the Kronecker matrix in (3.13) could still be nonsingular, and so $\mathcal{M}_k$ could still be strictly positive. See also [41] for an insightful study of matrices of the form $\mathcal{T}_m^T - B_m B_m^T Y$ and their key role for solving Riccati equations.

Concerning the convergence of the Gauss–Newton scheme, under standard assumptions, it has been established for nonlinear least-squares vector problems that if the residual $\|\mathcal{F}(Y^*)\|_F = 0$, then it has the same local convergence as Newton's method; i.e., it is q-quadratically convergent. Moreover, if $\mathcal{F}''(Y^*)^T\mathcal{F}(Y^*)$ is small relative to $\mathcal{F}'(Y^*)^T\mathcal{F}'(Y^*)$, then it is locally q-linearly convergent; see [14, 23] for details. In addition, since the linear operator $\mathcal{M}_k$ is strictly positive, it is invertible, and the inverse operator is also strictly positive, and so, as in the vector case, the step $S^{(k)}$ is a descent direction for the function $f(Y)$ at $Y^{(k)}$; i.e.,

$$\langle f'(Y^{(k)}), S^{(k)}\rangle_F = -2\langle \mathcal{F}'(Y^{(k)})^T\mathcal{F}(Y^{(k)}), (\mathcal{F}'(Y^{(k)})^T\mathcal{F}'(Y^{(k)}))^{-1}\mathcal{F}'(Y^{(k)})^T\mathcal{F}(Y^{(k)})\rangle_F < 0. \tag{3.14}$$

Nevertheless, it is a local method, and so it might happen that $f(Y^{(k)} + S^{(k)}) > f(Y^{(k)})$ at some iterations. This suggests a line search *backtracking* globalization strategy in the descent direction $S^{(k)}$ to improve the global behavior of the Gauss–Newton algorithm. The idea is to obtain the next iterate as $Y^{(k+1)} = Y^{(k)} + \lambda_k S^{(k)}$, where $\lambda_k \in (0, 1]$ is a step length that guarantees a sufficient reduction in the value of the objective function $\mathcal{F}(Y^{(k+1)})$. The well-known Armijo backtracking approach (see, e.g., [14]) is included in our global Gauss–Newton algorithm.

Algorithm 1. Global Gauss–Newton algorithm.

```
• Given a symmetric positive semidefinite $Y^{(0)}$, $\gamma \in (0, 1/2)$, and $0 < \sigma_1 < \sigma_2 < 1$
• For $k = 0, 1, \ldots$ until convergence, do
    1. Solve the matrix linear problem (3.11) for $S_k$
    2. Set $\lambda_c = 1$
    3. (Armijo backtracking)
    4. While $f(Y^{(k)} + \lambda_c S^{(k)}) > f(Y^{(k)}) + \gamma\lambda_c \langle f'(Y^{(k)}), S^{(k)}\rangle_F$, do
    5.     Choose $\rho \in [\sigma_1, \sigma_2]$ and set $\lambda_c = \rho\lambda_c$
    6. End While
    7. Set $\lambda_k = \lambda_c$ and $Y^{(k+1)} = Y^{(k)} + \lambda_k S^{(k)}$
• End For
```

Our next result establishes that if $\mathcal{M}_k$ is strictly positive for all k, then the global Gauss–Newton algorithm is well defined and also a global convergence property under an additional uniform bound assumption on the sequence $\{\mathcal{M}_k\}$.

THEOREM 3.4. *If $\mathcal{M}_k$ is strictly positive for all k, then the global Gauss–Newton algorithm is well defined. Moreover, if there exist $0 < \lambda_{\min} < \lambda_{\max} < \infty$ such that at every iteration k and for any symmetric matrix $Z \neq 0$*

$$\lambda_{\min} \leq \frac{\langle \mathcal{M}_k(Z), Z\rangle_F}{\langle Z, Z\rangle_F} \leq \lambda_{\max}, \tag{3.15}$$

then $\lim_{k\to\infty} f'(Y^{(k)}) = 0$. Therefore, any limit point of the sequence $\{Y^{(k)}\}$ is a stationary point of $f(Y)$.

Proof. Since $\mathcal{M}_k$ is strictly positive, using (3.14), we have that $\langle f'(Y^{(k)}), S^{(k)}\rangle_F < 0$, and so the number of cycles at the backtracking process (steps 4, 5, and 6 of the algorithm) will be finite for every k, until a sufficiently small $\lambda_k > 0$ is obtained such that

$$f(Y^{(k)} + \lambda_k S^{(k)}) \leq f(Y^{(k)}) + \gamma\lambda_k \langle f'(Y^{(k)}), S^{(k)}\rangle_F.$$

In addition, since $\mathcal{M}_k$ is strictly positive, the linear matrix system (3.11) has a unique solution for every k, and hence the algorithm is well defined.

Using now (3.15), and since $f'(Y)$ is Lipschitz continuous and $f(Y)$ is bounded below by zero, the rest of the proof follows straightforwardly from Theorem 2.5 in [21, section 2.5]. $\square$

We note that the uniform bound (3.15) on the linear operators $\mathcal{M}_k$, which are approximating the second derivatives $f''(Y^{(k)})$, is a classical and standard assumption to establish global convergence whenever an iterative optimization method is embedded into an inexact globalization strategy; see, e.g., [14, 21]. We also note that the obtained global convergence result (limit points are stationary points) does not guarantee convergence to a specific desired minimization matrix, which in our case is the positive semidefinite one (see Remarks 3.1 and 3.2).

We are using, at every iteration of the global Gauss–Newton algorithm, the convenient initial trial $\lambda_c = 1$ to start with the full Gauss–Newton step in order to observe its fast local convergence once the iterations are near the solution. Notice also that the algorithm needs to solve the matrix linear problem (3.11) only once per iteration. The additional cost required at each backtracking step is only a function evaluation at $f(Y^{(k)} + \lambda_c S^{(k)})$. In practical implementations, the typical values of the key parameters are $\sigma_1 = 0.1$, $\sigma_2 = 0.9$, and $\gamma = 1. \times 10^{-4}$ (very small to avoid unnecessary backtracking cycles). Concerning $\rho \in [\sigma_1, \sigma_2]$, it can be chosen differently at every backtracking step, but in some cases the conservative fixed choice $\rho = 1/2$ works effectively.

4. Methods for solving the linear matrix problems.

4.1. Using the Kronecker form and a direct method. Using the Kronecker form, the linear matrix (3.11) is equivalent to solving the following linear system:

$$M_k\, x = c_k, \tag{4.1}$$

where

$$M_k = [I\otimes(\mathcal{L}(Y^{(k)})^T\mathcal{L}(Y^{(k)}))]+[(\mathcal{L}(Y^{(k)})^T\mathcal{L}(Y^{(k)}))\otimes I]+[\widehat{\mathcal{L}}(Y^{(k)})\otimes\widehat{\mathcal{L}}(Y^{(k)})^T]+[\widehat{\mathcal{L}}(Y^{(k)})^T\otimes\widehat{\mathcal{L}}(Y^{(k)})]$$

and $c_k = vec(\mathcal{C}_k)$. Likewise, once (4.1) is solved we recover the step $S^{(k)}$ required at every iteration of Algorithm 1 from $x = vec(S^{(k)})$. Hence, for small values of $\widehat{ms}$, we can use a direct method to solve the symmetric and positive definite linear system of equations (4.1).

4.2. The preconditioned global conjugate gradient method. For large values of $\widehat{ms}$ we can use the preconditioned global conjugate gradient (PGCG) method, described in [18], to solve the linear matrix problem (3.11). The PGCG method applies the operator $\mathcal{M}_k$ directly on the required matrix subspace, and so it avoids building the Kronecker matrix M_k in (4.1). The theoretical properties of $\mathcal{M}_k$, established in Theorem 3.3, guarantee the convergence of the PGCG method to obtain the step $S^{(k)}$ at every iteration of the Gauss–Newton scheme.

For a properly chosen preconditioner $\mathcal{P}_k \approx \mathcal{M}_k$, the application of the PGCG method for solving (3.11) is summarized in Algorithm 2.

Algorithm 2. The preconditioned global conjugate gradient (PGCG) algorithm.

```
• Choose a tolerance $tol$, a maximum number of $j_{max}$ iterations.
  Choose $\widetilde{S}_0$, set $\widehat{\mathcal{R}}_0 = \mathcal{C}_k - \mathcal{M}_k(\widetilde{S}_0)$, solve $\mathcal{P}_k Z_0 = \widehat{\mathcal{R}}_0$ for $Z_0$, and set $U_0 = Z_0$.
• For $j = 0, 1, 2, \dots, j_{max}$
    1. $W_j = \mathcal{M}_k(U_j)$
    2. $\alpha_j = \left\langle \widehat{\mathcal{R}}_j, Z_j \right\rangle_F / \langle W_j, U_j \rangle_F$
    3. $\widetilde{S}_{j+1} = \widetilde{S}_j + \alpha_j U_j$
    4. $\widehat{\mathcal{R}}_{j+1} = \widehat{\mathcal{R}}_j - \alpha_j W_j$
    5. If $\|\widehat{\mathcal{R}}_{j+1}\|_F < tol$ stop, else
    6. Solve $\mathcal{P}_k Z_{j+1} = \widehat{\mathcal{R}}_{j+1}$ for $Z_{j+1}$
    7. $\beta_j = \left\langle \widehat{\mathcal{R}}_{j+1}, Z_{j+1} \right\rangle_F / \left\langle \widehat{\mathcal{R}}_j, Z_j \right\rangle_F$
    8. $U_{j+1} = Z_{j+1} + \beta_j U_j$
• End For
```

Notice that, once Algorithm 2 stops at some iteration j, the current matrix $\widetilde{S}_j$ will be used as the step $S^{(k)}$ in the Gauss–Newton Algorithm 1. Notice also that the computation of $W_j = \mathcal{M}_k(U_j)$ at every j is obtained using (3.12).

Concerning a preconditioning strategy, we now describe a suitable option for obtaining $\mathcal{P}_k$ at every Gauss–Newton iteration, which can be viewed as an adapted version of the one developed in [34, section 3.4.1] for Lyapunov equations. Consider the Schur factorization of the symmetric and positive semidefinite matrix $\mathcal{L}(Y^{(k)})^T\mathcal{L}(Y^{(k)})$:

$$\mathcal{L}(Y^{(k)})^T\mathcal{L}(Y^{(k)}) = \mathcal{V}\mathcal{D}\mathcal{V}^T, \tag{4.2}$$

where $\mathcal{V}$ is an orthogonal matrix and $\mathcal{D}$ is the diagonal matrix of the eigenvalues. Therefore, the linear matrix equation (3.11) reduces to

$$\mathcal{V}\mathcal{D}\mathcal{V}^T S + S\,\mathcal{V}\mathcal{D}\mathcal{V}^T + \widehat{\mathcal{L}}(Y^{(k)})S\widehat{\mathcal{L}}(Y^{(k)}) + \widehat{\mathcal{L}}(Y^{(k)})^T S\widehat{\mathcal{L}}(Y^{(k)})^T = \mathcal{C}_k. \tag{4.3}$$

Setting $\widetilde{S} = \mathcal{V}^T S\mathcal{V}$ and multiplying both sides of (4.3) from the left by $\mathcal{V}^T$ and from the right by $\mathcal{V}$, we get

$$\mathcal{D}\widetilde{S} + \widetilde{S}\,\mathcal{D} + \mathcal{V}^T\widehat{\mathcal{L}}(Y^{(k)})S\widehat{\mathcal{L}}(Y^{(k)})\mathcal{V} + \mathcal{V}^T\widehat{\mathcal{L}}(Y^{(k)})^T S\widehat{\mathcal{L}}(Y^{(k)})^T\mathcal{V} = \mathcal{V}^T\mathcal{C}_k\mathcal{V}. \tag{4.4}$$

Setting $\bar{\mathcal{L}}_k = \mathcal{V}^T\widehat{\mathcal{L}}(Y^{(k)})\mathcal{V}$ and $\bar{\mathcal{C}}_k = \mathcal{V}^T\mathcal{C}_k\mathcal{V}$, it follows that problem (4.4) is equivalent to

$$\mathcal{N}_k(\widetilde{S}) = \bar{\mathcal{C}}_k, \tag{4.5}$$

where

$$\mathcal{N}_k(\widetilde{S}) = \mathcal{D}\widetilde{S} + \widetilde{S}\,\mathcal{D} + \bar{\mathcal{L}}_k\widetilde{S}\bar{\mathcal{L}}_k + \bar{\mathcal{L}}_k^T\widetilde{S}\bar{\mathcal{L}}_k^T.$$

Therefore, we can use the first part of (4.5) as a preconditioner in the PGCG Algorithm 2. For that we set $\mathcal{P}_k(\widetilde{S}) = \mathcal{V}(\mathcal{D}\widetilde{S} + \widetilde{S}\,\mathcal{D})\mathcal{V}^T$. Notice that with this preconditioning strategy, Algorithm 2 will require solving for $\widehat{Z}_j$, at every j, a linear problem of the form

$$\mathcal{D}\widehat{Z}_j + \widehat{Z}_j\,\mathcal{D} = \mathcal{V}^T\widehat{\mathcal{R}}_j\mathcal{V},$$

which will be done easily if $\mathcal{D}$ is nonsingular, since the matrix $\mathcal{D}$ is diagonal. Once $\widehat{Z}_j$ is obtained, we recover the matrix Z_j required in the algorithm by setting $Z_j = \mathcal{V}\widehat{Z}_j\mathcal{V}^T$.

5. A factorized form of the approximate solution.

A standard procedure for large-scale matrix problems, once convergence is achieved, is to obtain the approximate solution X_m as the product of two low-rank matrices, which is very important for storage requirements; see, e.g., [24, 34, 40]. Consider the eigenvalue decomposition of the obtained matrix $Y_m = U\Lambda\,U^T$, where Λ is the diagonal matrix of eigenvalues in decreasing order. Let Λ_l be the submatrix of Λ obtained by deleting the eigenvalues that are less than some fixed threshold $dtol > 0$, and let U_l be the corresponding matrix obtained from the orthogonal matrix U. Then $Y_m \approx U_l\,\Lambda_l\,{U_l}^T$, where $\Lambda_l = \mathrm{diag}[\lambda_1,\ldots,\lambda_l]$. Let $Z_m = \mathbb{V}_m\,U_l\,\Lambda_l^{1/2}$; then the approximate solution X_m can be given in a factored form

$$X_m \approx Z_m\,Z_m^T. \tag{5.1}$$

Summing up, the minimization approach for solving continuous matrix Riccati equations is described in Algorithm 3.

Algorithm 3. Combined scheme for solving CARE.

```
• Choose a tolerance $\epsilon$ and a maximum number of allowed iterations $mmax$
• For $m = 1, 2, \ldots, mmax$
    1. Apply EBA or RKSM to expand the approximate subspace, and then
       update $\mathbb{V}_m$, $\widetilde{\mathcal{T}}_m$, $\mathcal{T}_m$, and $B_m$
    2. Use the Global Gauss–Newton Algorithm 1 to solve (3.4) and obtain $Y_m$
    3. Compute the norm of the residual $r_m = \|R(X_m)\|_F$ using (3.3)
    4. If $r_m/\|CC^T\|_F < \epsilon$ stop, and get the matrix $X_m$ in a factored form as
       in (5.1)
• End For
```

Note that the Galerkin-projection scheme for solving CARE can also be embedded into the framework of Algorithm 3 by changing step 2 and step 3 as follows:

2. Solve the low-dimensional Riccati equation (2.5) and obtain Y_m
3. Compute the norm of the residual $r_m = \|R(X_m^G)\|_2$ using (2.6).

We would like to remark that the formulas used to evaluate the residual norm at each iteration of Algorithm 3 ((2.6) for the Galerkin approach and (3.3) for the Gauss–Newton approach) do not need the explicit knowledge of X_m^G and X_m, respectively. We also note that (2.6), which takes advantage of the orthogonality Galerkin condition (2.4), requires less computational work than (3.3).

6. Numerical experiments.

To give further insight into the nonlinear least-squares Gauss–Newton approach, and to illustrate its performance when solving (1.1), we present the results of some numerical experiments. All computations were performed in MATLAB 7.3, which has unit roundoff $\mu \approx 1.1 \times 10^{-16}$. All the runs were carried out on an Intel Core i5-2450M at 2.5 GHz with 4 GB of RAM.

We compared the behavior of the Galerkin approach and the Gauss–Newton approach, both combined with EBA and RKSM, within the framework of Algorithm 3. For EBA we used the implementation described in [24] (see also [40]), and for RKSM we used the implementation described in [17], using complex arithmetic when complex shifts occur. For the Galerkin approach, we used the control system toolbox for calling the MATLAB `care` function for solving the low-dimensional Riccati equations. For the Gauss–Newton approach we considered the Kronecker, the global conjugate gradient (GCG), and the preconditioned global conjugate gradient (PGCG) methods for solving the linear matrix problems (3.11). For the GCG method, we set $Z_j = \widehat{\mathcal{R}}_j$ for all j in Algorithm 2. For building the preconditioner, in Algorithm 2, we used the MATLAB `schur` function to obtain $\mathcal{V}$ and $\mathcal{D}$ in (4.2). For the (preconditioned) GCG method the initial matrix was always chosen as $\widetilde{S}_0 = 0$.

Concerning the symmetric positive semidefinite initial guess for Algorithm 1, we used $I_{\widehat{s}}$ the very first time it was called ($m = 1$), and then for all $m > 1$ we used $\widehat{Y}_m$, which is the closest positive semidefinite approximation of the obtained Y_m from the previous external iteration as the top-left block, and we used $I_{\widehat{s}}$ at the bottom-right corner, and zeros elsewhere. The matrix $\widehat{Y}_m$ is obtained by projecting Y_m onto the closed cone of symmetric positive semidefinite matrices. This projection can be easily obtained by computing only the negative eigenvalues and the corresponding eigenvectors of Y_m as follows. Let $\{\lambda_1, \ldots, \lambda_q\}$ be the negative eigenvalues of Y_m and $\{Z_1, \ldots, Z_q\}$ be the corresponding eigenvectors. Then $\widehat{Y}_m = Y_m - \sum_{i=1}^{q} \lambda_i Z_i Z_i^T$; see [19, Theorem 3.1]. Notice that if Y_m is already positive semidefinite, i.e., if $q = 0$, then $\widehat{Y}_m = Y_m$.

The same stopping criterion was used for both approaches, and the computations in Algorithm 3 were stopped using $\epsilon = 10^{-7}$. We stopped the iterations of the global Gauss–Newton method (Algorithm 1) when the norm of $f'(Y^{(k)}) = 2\mathcal{F}'(Y^{(k)})^T\mathcal{F}(Y^{(k)}) = -2\mathcal{C}_k$ was less than 10^{-4}, or equivalently when $\|\mathcal{C}_k\|_F \leq 0.5 \times 10^{-4}$. For the PGCG and the GCG methods we stopped the iterations using $tol = 10^{-6}$, and we set $j_{max} = (\widehat{ms})^2$. In the eigenvalue decomposition of $Y_m \in \mathbb{R}^{\widehat{ms}\times\widehat{ms}}$, to obtain the approximation X_m in a factored form as in (5.1), we used $dtol = 10^{-11}$ to discard the columns of U corresponding to diagonal elements of Λ less than $dtol$. It is worth mentioning that all the considered options seem to converge to the same solution matrix. We performed some additional calculations for the experiments with small dimensions (reported in Table 6.1), and we noticed that the distance in Frobenius norm among all the approximate solutions was always less than 10^{-4}.

Unless otherwise specified, throughout our experiments, the matrix A is generated from the five-point centered finite difference discretization of the operator

$$\Delta u - f_1(x,y)\frac{\partial u}{\partial x} - f_2(x,y)\frac{\partial u}{\partial y} - g(x,y)u \tag{6.1}$$

on the unit square $[0,1]\times[0,1]$ with homogeneous Dirichlet boundary conditions. The dimension of the matrix A is $n = n_0^2$, where n_0 is the number of inner grid points in each direction. In the tables below, we report the number of iterations (IT) and the CPU time in seconds (Time) required by Algorithm 3. We also report the dimension of the subspace (Spc Dim) in which the approximate solution was found, and the rank (Rk(X_m)) of the final factorized matrix X_m. For the Gauss–Newton approach we report the average number of iterations of the Gauss–Newton method for each value of m, and if the (preconditioned) GCG method is used for solving the linear problems, we also report the average number of (preconditioned) GCG iterations per each GN iteration. They are both reported under the label (Avg GN/GCG It).

For our first experiment, we choose A setting $f_1(x,y) = e^{xy}$, $f_2(x,y) = \sin(xy)$, and $g(x,y) = y^2 - x^2$ in (6.1). For B and C we use low-rank matrices whose entries are randomly chosen in $[0,1]$ with a uniform distribution. In Table 6.1 we report the performance of both approaches for different values of p and s, and small values of n. We notice that the Kronecker option, for solving the linear problems, requires in general more CPU time than the other options, and as a consequence, for the next experiments for which the dimensions will be increased we no longer consider it. Similarly, we observe that the GCG method without using the preconditioner requires more internal iterations and computational work than the preconditioned version to obtain the same solution. From now on when using the Gauss–Newton method we will only report the PGCG option.

For our second experiment, we choose A setting $f_1(x,y) = e^{x^2+y}$, $f_2(x,y) = \cos(xy)$, and $g(x,y) = y^2 - x^2$ in (6.1). The entries of B and C are randomly chosen in $[0,1]$. In Table 6.2 we report the performance of both approaches for different values of p and s, and larger values of n.

From Tables 6.1 and 6.2 we notice that for the matrices A, B, and C, used so far, the Galerkin approach has no difficulties converging to the solution in a small-dimension subspace, and we also notice that all the different options using the Gauss–Newton approach also converge to the same approximate solution in the same subspace, with a very small difference in the rank of the final factorized matrix X_m. For these experiments both approaches converge smoothly. We can also observe that the GN-PGCG options require an increase in CPU time when compared to the corre-

Table 6.1

Performance of the Galerkin approach and the Gauss–Newton approach when $f_1(x,y) = e^{xy}$*,* $f_2(x,y) = \sin(xy)$*, and* $g(x,y) = y^2 - x^2$ *in* (6.1) *for different values of* n*,* p*, and* s*. The entries of* B *and* C *are randomly chosen in* $[0,1]$*.*

	Method	It	Avg GN/GCG It	Time	Spc Dim	Rk(X_m)
$n = 400$	Galerkin-EBA	9		.21	54	35
$p = 6$	GN-EBA-Kron	9	4.6	38.3	54	36
$s = 3$	GN-EBA-GCG	9	4.6/155	4.0	54	36
	GN-EBA-PGCG	9	4.6/6.8	.64	54	36
$n = 400$	Galerkin-RKSM	11		.31	33	33
$p = 6$	GN-RKSM-Kron	11	4.2	3.7	33	33
$s = 3$	GN-RKSM-GCG	11	4.2/148	2.4	33	33
	GN-RKSM-PGCG	11	4.2/8.1	.45	33	33
$n = 900$	Galerkin-EBA	11		.26	44	29
$p = 4$	GN-EBA-Kron	11	4.6	14.3	44	30
$s = 2$	GN-EBA-GCG	11	4.6/275	7.8	44	30
	GN-EBA-PGCG	11	4.6/8.4	.64	44	30
$n = 900$	Galerkin-RKSM	13		.35	26	26
$p = 4$	GN-RKSM-Kron	13	4.1	1.8	26	26
$s = 2$	GN-RKSM-GCG	13	4.1/167	3.7	26	26
	GN-RKSM-PGCG	13	4.1/8.1	.44	26	26

Table 6.2

Performance of the Galerkin approach and the Gauss–Newton approach when $f_1(x,y) = e^{x^2+y}$*,* $f_2(x,y) = \cos(xy)$*, and* $g(x,y) = y^2 - x^2$ *in* (6.1) *for different values of* n*,* p*, and* s*. The entries of* B *and* C *are randomly chosen in* $[0,1]$*.*

	Method	It	Avg GN/GCG It	Time	Spc Dim	Rk(X_m)
$n = 10000$	Galerkin-EBA	19		1.82	76	39
$p = 4$	GN-EBA-PGCG	19	4.8/9.8	3.61	76	41
$s = 2$	Galerkin-RKSM	17		2.43	34	34
	GN-RKSM-PGCG	17	3.8/9.8	2.85	34	34
$n = 90000$	Galerkin-EBA	19		76.2	228	132
$p = 10$	GN-EBA-PGCG	19	5.1/11.3	112	228	130
$s = 6$	Galerkin-RKSM	21		132	126	120
	GN-RKSM-PGCG	21	3.9/11.4	153	126	120

sponding Galerkin options. Finally, we can observe that using RKSM the solution was consistently found in a smaller rank subspace as compared with the related one obtained with EBA, which is a very valuable feature. Based on that, for the next experiments we will only report the results obtained using RKSM.

For our third experiment, we choose A setting $f_1(x,y) = e^{-11xy}$, $f_2(x,y) = e^{11xy}$, and $g(x,y) = -15(x+y)$ in (6.1). A similar stable and slightly nonpassive matrix has already been used in [17, 34] (a square matrix is passive if its field of values lies in the open left half-plane; see [25]). The entries of B and C are randomly chosen in $[0,1]$, and we set $s = 2$, $p = 2$, $n = 100$ (left), and $n = 400$ (right). The residual norm of both approaches (using RKSM) is plotted on a log scale versus the space dimension in Figure 6.1. For this particular setting, we note that when the subspace dimension $\widehat{ms}$ is small, a few eigenvalues of the projected matrix $\mathcal{T}_m$ have positive real part, and we also observe that the Galerkin approach has serious difficulties finding the stabilizing solution, and hence it exhibits an erratic behavior at the beginning of the process. We also note that for the same small values of $\widehat{ms}$, the Gauss–Newton approach quickly and smoothly finds a symmetric but indefinite global minimizer Y_m, with very few small-sized negative eigenvalues. Nevertheless, after a certain value of $\widehat{ms}$ (approximately 20 for $n = 100$ and 40 for $n = 400$), $\mathcal{T}_m$ becomes stable and the Galerkin approach manages to obtain the same approximate stabilizing solutions obtained by the Gauss–Newton approach whose convergence behavior is smooth and controlled during the whole process. For $n = 100$, the Galerkin approach finds the approximate solution on a subspace of dimension 62 with $\mathrm{Rk}(X_m) = 40$, whereas the Gauss–Newton approach finds the solution on a subspace of dimension 58 with $\mathrm{Rk}(X_m) = 40$. For $n = 400$, the Galerkin approach finds the approximate solution on a subspace of dimension 140 with $\mathrm{Rk}(X_m) = 82$, and the Gauss–Newton approach finds the solution on a subspace of dimension 134 with $\mathrm{Rk}(X_m) = 82$. We note that, for many other different values of n, at the beginning of the process the two residual curves are significantly different, keeping several orders of magnitude from each other. However, after a certain subspace dimension they both behave similarly, showing a smooth performance.

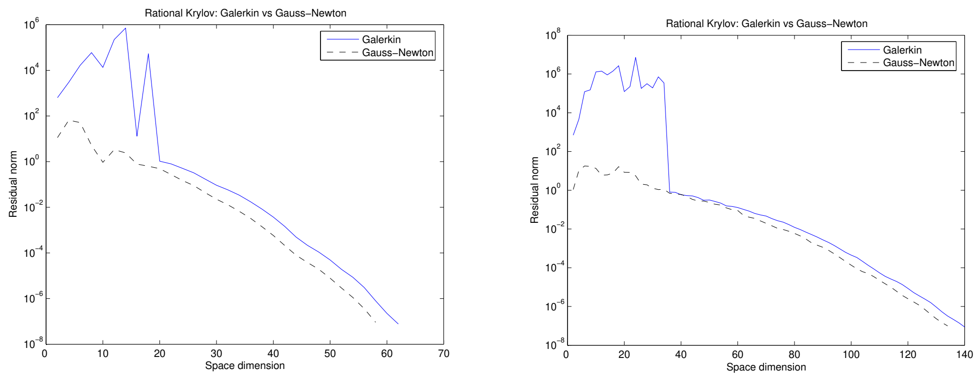

FIG. 6.1. *Residual norm versus space dimension for the Galerkin approach and the Gauss–Newton approach, both using RKSM, when* $f_1(x, y) = e^{-11xy}$, $f_2(x, y) = e^{11xy}$, $g(x, y) = -15(x+y)$, $n = 100$ *(left),* $n = 400$ *(right),* $s = 2$, *and* $p = 2$. *The entries of* B *and* C *are randomly chosen in* $[0, 1]$.

For our fourth and last experiment we choose the 900×900 real symmetric matrix $A = T \otimes I_{30} + I_{30} \otimes T$, where $T = \mathtt{toeplitz}([-2, \beta, \mathtt{zeros}(1, 28)])$. We set $s = 2$ and $p = 2$. The entries of the first row of C are all ones, and the second row of C is equal to $[1, -2, 1, -2, 1, -2, \ldots]$. The entries of the columns of B are chosen uniformly distributed in the interval $[0, 1]$: $B(:, 1) = \mathtt{linspace}(0, 1, 900)'$ and $B(:, 2) = \mathtt{linspace}(1, 0, 900)'$. We note that for $\beta = 1$, the matrix A is stable, but as soon as β slightly increases, say $\beta = 1 + \epsilon_0$ for $0 < \epsilon_0 \ll 1$, very few and very small positive eigenvalues appear in the spectrum of A, i.e., A becomes moderately nonstable, which could occur due to numerical perturbations. We are interested in the use of β slightly bigger than one to see the effect that such a small numerical perturbation can produce in the behavior of the Galerkin and the Gauss–Newton approaches. In Table 6.3 we report the performance of both schemes for different values of $\beta \geq 1$. We note that for $\beta > 1$, the Galerkin approach can still solve the Riccati equations on the low-dimensional subspaces but incorporates in the approximate solution Y_m a few more positive eigenvalues, above the threshold $dtol > 0$, than the ones present in the Y_m's obtained by the Gauss–Newton approach. As a consequence, a higher value of $\mathrm{Rk}(X_m)$ is obtained by the Galerkin approach as compared with the one obtained by the Gauss–Newton approach. Moreover, the difference between them increases when ϵ_0 increases.

Table 6.3
Performance of the Galerkin approach and the Gauss–Newton approach for our fourth experiment, when $A = T \otimes I_{30} + I_{30} \otimes T$ and $T =$ `toeplitz`$([-2, \beta,$ `zeros`$(1, 28)])$ for different values of $\beta \geq 1$.

	Method	It	Time	Spc Dim	Rk(X_m)
$\beta = 1$	Galerkin-RKSM	9	0.23	18	18
	GN-RKSM-PGCG	9	0.28	18	18
$\beta = 1.01$	Galerkin-RKSM	10	.28	20	20
	GN-RKSM-PGCG	10	.31	20	20
$\beta = 1.03$	Galerkin-RKSM	15	.42	30	27
	GN-RKSM-PGCG	15	.87	30	24
$\beta = 1.05$	Galerkin-RKSM	18	.54	36	33
	GN-RKSM-PGCG	18	1.92	36	26

We close this section with some general comments concerning the performance of both approaches, based on the reported results and also on many additional experiments. For relatively easy problems, regardless of the dimension of the matrices, the Galerkin approach has no difficulty finding the stabilizing solution in each one of the subspaces. For those problems, the Gauss–Newton approach converges most of the time to the same stabilizing solution in each one of the subspaces and very seldom converges to a nearby global solution Y_m of (3.4) with very few small-sized negative eigenvalues that fall below the truncation threshold *dtol* (see Remark 3.1). In that case, the projection of Y_m onto the cone of symmetric positive semidefinite matrices, to obtain $\widehat{Y}_m$, which is used to build the next initial guess for Algorithm 1, is very convenient to guarantee the convergence towards the final low-rank stabilizing solution of problem (1.1), which coincides with the final solution obtained by the Galerkin approach in the same subspace, and with a quite similar value of Rk(X_k).

In the case of solving harder problems for which the Galerkin approach struggles to find a stabilizing solution on the low-dimensional subspaces, the Gauss–Newton approach has no difficulty finding a suitable Y_m with few negative eigenvalues, which is a global solution of (3.4). Those negative eigenvalues tend to decrease in magnitude when m increases in such a way that when the right subspace has been reached, they fall once again below the truncation tolerance *dtol*. As before, the convenient way of building the next initial guess for Algorithm 1 plays a key role in guaranteeing the convergence to the low-rank stabilizing solution of problem (1.1). In other words, the fact that only the stabilizing solution in each one of the subspaces is acceptable clearly represents a drawback of the Galerkin approach. On the other hand, the fact that the Gauss–Newton approach is able to accept suitable indefinite global solutions at the intermediate subspaces represents a flexible feature that, combined with the right way of starting Algorithm 1 at each new subspace, accounts for the smooth convergence that we have observed towards the stabilizing low-rank solution of (1.1).

It is also worth noticing that in all cases, whether the problem is hard or easy, or large or small, the Gauss–Newton method (Algorithm 1) converges to a global solution for which the value of f is zero. In other words, so far we have not observed convergence to a local nonglobal solution of (3.4). As a consequence, the convergence of the Gauss–Newton scheme is very fast, usually showing a q-quadratic rate of convergence.

7. Concluding remarks.

For the same low-dimensional subspace scenario for which the Galerkin-projection approach has been used for solving Riccati equations, we have presented and analyzed a novel matrix nonlinear least-squares approach. Our experiments show that the new approach, combined with a PGCG method, is competitive with the Galerkin-projection approach when both schemes produce converging sequences to an approximate low-rank solution, and is an available robust option for those cases when the Galerkin approach has difficulty solving the subproblems on the Krylov subspaces. In our numerical section we presented and discussed several of those cases.

Our main contribution in this paper was the development of a Gauss–Newton method for solving matrix nonlinear least-squares problems that, as far as we know, has not been addressed in the literature for solving nonlinear matrix problems. A key advantage of using this approach is that it constructs an approximation to the second derivative of the objective function using only first-order information. For the Gauss–Newton method, we proposed a convenient way of choosing a symmetric and positive semidefinite initial guess at each low-dimensional subspace, and also a line search globalization strategy to guarantee global convergence to stationary points. We would like to mention that another globalization option, usually associated with the Gauss–Newton method for vector problems, is the Levenberg–Marquardt strategy [14, 21, 23], but in our matrix setting that option is too expensive since it involves the dynamical choice of a parameter λ. For each $\lambda > 0$ a new linear operator matrix problem needs to be solved, whereas for the line search backtracking option only one such linear problem needs to be solved per Gauss–Newton iteration. In all of our numerical experiments the proposed line search strategy was enough to obtain a smooth global convergence and a fast local behavior.

We have also included a preconditioning strategy based on the Schur factorization of the symmetric and positive operator $\mathcal{L}(Y_k)^T\mathcal{L}(Y_k)$, to accelerate the convergence of the GCG algorithm. It is worth mentioning that any other preconditioning strategy can be supplied by the user, as long as the supplied linear operator $\mathcal{P}_k$ is invertible for each k.

Finally, we would like to mention that a MATLAB version of all algorithms developed by the authors is available upon request.

Acknowledgments. We would like to thank two referees for pointing out some key references and for their helpful comments and suggestions that helped us to improve the final version of this paper. The second author would like to thank the Laboratory LMPA at Université du Littoral Côte d'Opale, Calais, France, for the hospitality during a one-month visit in June 2018.

REFERENCES

[1] H. Abou-Kandil, G. Freiling, V. Ionescu, and G. Jank, *Matrix Riccati Equations in Control and Systems Theory*, Syst. Control Found. Appl., Birkhäuser, Basel, 2003.

[2] B.D.O. Anderson and J.B. Moore, *Optimal Control—Linear Quadratic Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1990.

[3] W.F. Arnold, III, and A.J. Laub, *Generalized eigenproblem algorithms and software for algebraic Riccati equations*, Proc. IEEE, 72 (1984), pp. 1746–1754, https://doi.org/10.1109/PROC.1984.13083.

[4] P. Benner and Z. Bujanović, *On the solution of large-scale algebraic Riccati equations by using low-dimensional invariant subspaces*, Linear Algebra Appl., 488 (2016), pp. 430–459, https://doi.org/10.1016/j.laa.2015.09.027.

[5] P. Benner, Z. Bujanović, P. Kürschner, and J. Saak, *RADI: A low-rank ADI-type algorithm for large scale algebraic Riccati equations*, Numer. Math., 138 (2018), pp. 301–330, https://doi.org/10.1007/s00211-017-0907-5.

[6] P. Benner, Z. Bujanović, P. Kürschner, and J. Saak, *A Numerical Comparison of Solvers for Large-Scale, Continuous-Time Algebraic Riccati Equations*, preprint, https://arxiv.org/abs/1811.00850v1, 2018.

[7] P. Benner, M. Heinkenschloss, J. Saak, and H.K. Weichelt, *An inexact low-rank Newton-ADI method for large-scale algebraic Riccati equations*, Appl. Numer. Math., 108 (2016), pp. 125–142, https://doi.org/10.1016/j.apnum.2016.05.006.

[8] P. Benner, J. Li, and T. Penzl, *Numerical solution of large Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems*, Numer. Linear Algebra Appl., 15 (2008), pp. 755–777, https://doi.org/10.1002/nla.622.

[9] S. Bittanti, A. Laub, and J.C. Willems, eds., *The Riccati Equation*, Comm. Control Engrg., Springer-Verlag, Berlin, 1991.

[10] J.L. Casti, *Linear Dynamical Systems*, 2nd ed., Math. Sci. Engrg. 135, Academic Press, Boston, 1987.

[11] J.P. Chehab and M. Raydan, *Inexact Newton's method with inner implicit preconditioning for algebraic Riccati equations*, Comput. Appl. Math., 36 (2017), pp. 955–969, https://doi.org/10.1007/s40314-015-0274-8.

[12] M.J. Corless and A.E. Frazho, *Linear Systems and Control—An Operator Perspective*, Marcel Dekker, New York, 2003.

[13] B.N. Datta, *Numerical Methods for Linear Control Systems Design and Analysis*, Elsevier/Academic Press, New York, 2003.

[14] J.E. Dennis, Jr., and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Classics Appl. Math. 16, SIAM, Philadelphia, 1996, https://doi.org/10.1137/1.9781611971200.

[15] E.J.C. Doyle, K. Glover, P.P. Khargonekar, and B.A. Francis, *State space solutions to standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control problems*, IEEE Trans. Automat. Control, 34 (1989), pp. 831–846, https://doi.org/10.1109/9.29425.

[16] 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, https://doi.org/10.1137/S0895479895292400.

[17] V. Druskin and V. Simoncini, *Adaptive rational Krylov subspaces for large-scale dynamical systems*, Systems Control Lett., 60 (2011), pp. 546–560, https://doi.org/10.1016/j.sysconle.2011.04.013.

[18] A. El Guennouni, K. Jbilou, and A.J. Riquet, *Block Krylov methods for large Sylvester equations*, Numer. Alg., 29 (2002), pp. 75–96, https://doi.org/10.1023/A:1014807923223.

[19] R. Escalante and M. Raydan, *Dykstra's algorithm for constrained least-squares rectangular matrix problems*, Comput. Math. Appl., 35 (1998), pp. 73–79, https://doi.org/10.1016/S0898-1221(98)00020-0.

[20] F. Feitzinger, T. Hylla, and E.W. Sachs, *Inexact Kleinman–Newton method for Riccati equations*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 272–288, https://doi.org/10.1137/070700978.

[21] N.I.M. Gould, *An Introduction to Algorithms for Continuous Optimization*, http://www.numerical.rl.ac.uk/people/nimg/course/lectures/paper/paper.pdf, 2006.

[22] A.C.H. Guo and P. Lancaster, *Analysis and modification of Newton's method for algebraic Riccati equations*, Math. Comp., 67 (1998), pp. 1089–1105, https://doi.org/10.1090/S0025-5718-98-00947-8.

[23] P.C. Hansen, V. Pereyra, and G. Scherer, *Least-Squares Data Fitting with Applications*, The Johns Hopkins University Press, Baltimore, MD, 2013.

[24] M. Heyouni and K. Jbilou, *An extended block Arnoldi algorithm for large-scale solutions of the continuous-time algebraic Riccati equation*, Electron. Trans. Numer. Anal., 33 (2008/2009), pp. 53–62, http://etna.mcs.kent.edu/volumes/2001-2010/vol33/abstract.php?vol=33&pages=53-62.

[25] R.A. Horn and C.R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, New York, 1991, https://doi.org/10.1017/CBO9780511840371.

[26] I.M. Jaimoukha and E.M. Kasenally, *Krylov subspace methods for solving large Lyapunov equations*, SIAM J. Numer. Anal., 31 (1994), pp. 227–251, https://doi.org/10.1137/0731012.

[27] K. Jbilou, *Block Krylov subspace methods for large algebraic Riccati equations*, Numer. Alg., 34 (2003), pp. 339–353, https://doi.org/10.1023/B:NUMA.0000005349.18793.28.

[28] K. Jbilou, *An Arnoldi based algorithm for large algebraic Riccati equations*, Appl. Math. Lett., 19 (2006), pp. 437–444, https://doi.org/10.1016/j.aml.2005.07.001.

[29] K. Jbilou, A. Messaoudi, and H. Sadok, *Global FOM and GMRES algorithms for matrix equations*, Appl. Numer. Math., 31 (1999), pp. 49–63, https://doi.org/10.1016/S0168-9274(98)00094-4.

[30] C.S. Kenny, A.J. Laub, and P. Papadopoulos, *Matrix sign function algorithms for Riccati equations*, in Proceedings of the IMA Conference on Control: Modelling, Computation,

Information, IEEE, Washington, DC, 1992, pp. 1–10.

[31] D.L. Kleinman, *On an iterative technique for Riccati equation computations*, IEEE Trans. Automat. Control, 13 (1968), pp. 114–115, https://doi.org/10.1109/TAC.1968.1098829.

[32] P. Lancaster and L. Rodman, *Algebraic Riccati Equations*, Clarendon Press, Oxford, UK, 1995.

[33] A.J. Laub, *A Schur method for solving algebraic Riccati equations*, IEEE Trans. Automat. Control, 24 (1979), pp. 913–921, https://doi.org/10.1109/TAC.1979.1102178.

[34] Y. Lin and V. Simoncini, *Minimal residual methods for large scale Lyapunov equations*, Appl. Numer. Math., 72 (2013), pp. 52–71, https://doi.org/10.1016/j.apnum.2013.04.004.

[35] Y. Lin and V. Simoncini, *A new subspace iteration method for the algebraic Riccati equation*, Numer. Linear Algebra Appl., 22 (2015), pp. 26–47, https://doi.org/10.1002/nla.1936.

[36] A. Massoudi, M.R. Opmeer, and T. Reis, *Analysis of an iteration method for the algebraic Riccati equation*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 624–648, https://doi.org/10.1137/140985792.

[37] V. Mehrmann, *The Autonomous Linear Quadratic Control Problem: Theory and Numerical Solution*, Lecture Notes in Control and Inform. Sci. 163, Springer-Verlag, Berlin, 1991.

[38] A.J. Riquet, *Méthodes de Krylov par blocs pour les équations matricielles en théorie du contrôle*, Ph.D. thesis, Université du Littoral Côte d'Opale, Calais, France, 2002.

[39] D.J. Roberts, *Linear model reduction and solution of the algebraic Riccati equation by use of the sign function*, Internat. J. Control, 32 (1980), pp. 677–687, https://doi.org/10.1080/00207178008922881.

[40] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288, https://doi.org/10.1137/06066120X.

[41] V. Simoncini, *Analysis of the rational Krylov subspace projection method for large-scale algebraic Riccati equations*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 1655–1674, https://doi.org/10.1137/16M1059382.

[42] V. Simoncini, D.B. Szyld, and M. Monsalve, *On two numerical methods for the solution of large-scale algebraic Riccati equations*, IMA J. Numer. Anal., 34 (2014), pp. 904–920, https://doi.org/10.1093/imanum/drt015.

[43] P. Van Dooren, *A generalized eigenvalue approach for solving Riccati equations*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 121–135, https://doi.org/10.1137/0902010.

[44] F. Zhang, *Matrix Theory—Basic Results and Techniques*, Springer-Verlag, New York, 1999.

[45] K. Zhou, J.C. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice-Hall, Upper Saddle River, NJ, 1995.