A LOW-RANK SOLUTION METHOD FOR RICCATI EQUATIONS WITH INDEFINITE QUADRATIC TERMS

PETER BENNER, JAN HEILAND, AND STEFFEN W. R. WERNER

ABSTRACT. Algebraic Riccati equations with indefinite quadratic terms play an important role in applications related to robust controller design. While there are many established approaches to solve these in case of small-scale dense coefficients, there is no approach available to compute solutions in the large-scale sparse setting. In this paper, we develop an iterative method to compute low-rank approximations of stabilizing solutions of large-scale sparse continuous-time algebraic Riccati equations with indefinite quadratic terms. We test the developed approach for dense examples in comparison to other established matrix equation solvers, and investigate the applicability and performance in large-scale sparse examples.

algebraic Riccati equation and large-scale sparse matrices and low-rank approximation and iterative numerical method

1. Introduction

Many concepts in systems and control theory are connected to solutions of algebraic Riccati equations. Prominent examples are the linear-quadratic regulator (LQR) and linear-quadratic Gaussian (LQG) controller design [32] and corresponding model order reduction methods [23] as well as the characterization of passivity and contractivity of input-output systems and properties-preserving model reduction methods for these [1, 35]. They also appear, for example, in applications with differential games [4, 17]. In this paper, we target the numerical solution of Riccati equations with indefinite quadratic terms

$$A^{\mathsf{T}}X + XA + X\left(\gamma^{-2}B_1B_1^{\mathsf{T}} - B_2B_2^{\mathsf{T}}\right)X + C^{\mathsf{T}}C = 0, \tag{1}$$

as they frequently occur in the state-space theory of $\mathcal{H}_\infty$-robust controller design [33, 47]. In this case, the matrices $A \in \mathbb{R}^{n\times n}$, $B_1 \in \mathbb{R}^{n\times m_1}$, $B_2 \in \mathbb{R}^{n\times m_2}$ and $C \in \mathbb{R}^{p\times n}$ describe a linear time-invariant system of the form

$$\dot{x}(t) = Ax(t) + B_1w(t) + B_2u(t), \quad y(t) = Cx(t), \tag{2}$$

with state $x(t) \in \mathbb{R}^n$, controller input $u(t) \in \mathbb{R}^{m_2}$, disturbances $w(t) \in \mathbb{R}^{m_1}$ and output $y(t) \in \mathbb{R}^p$, and we are interested in a symmetric positive semi-definite solution X to (1) that stabilizes (A, B_2) with a bound on perturbations induced by $B_1w(t)$ expressed by $\gamma > 0$.

In the case of coefficients of small size n and classical LQR/LQG Riccati equations, i.e., $B_1 = 0$ in (1) such that the quadratic term is negative semi-definite, there is a variety of different numerical approaches to compute the stabilizing solution: Direct approaches that compute eigenvalue decompositions of underlying Hamiltonian or even matrix pencils [2, 28], iterative methods that work on the underlying spectrum of the Hamiltonian matrix [9, 36], or iterative approaches, which compute a sequence of matrices converging to the stabilizing solution [24, 40]. The problem becomes more complicated in the opposite case with symmetric positive semi-definite quadratic term, i.e., setting $B_2 = 0$ in (1), which occurs in bounded-real and positive-real problems [1, 35]. This already reduces the number of applicable methods. While techniques from the classical setting that do not take the definiteness of the quadratic term into account can still be applied here [2, 28, 36], only few, iterative Newton-type methods, which converge to the desired solution, have been developed for this problem class [14, 44]. The amount of applicable approaches narrows down even further when considering the general case of indefinite quadratic terms like in (1). A new type of iterative method for the solution of (1) was developed in [26, 27] to overcome accuracy problems of classical solution approaches and the general lack of iterative methods for this problem type. This new approach computes a sequence of matrices converging to the stabilizing solution of (1) by solving classical Riccati equations with symmetric negative semi-definite quadratic terms.

However, in this paper, we will mainly focus on the case of large-scale sparse coefficient matrices in (1) arising, for example, from the discretization of partial differential equations, with $n \geq 10^5$ and low-rank quadratic term and right-hand side such that $m_1, m_2, p \ll n$. A high dimension of n leads to two major computational issues that forbids the use of the computation methods from above that were developed for dense coefficients of small or moderate size. Firstly, any transformation, like the eigenvalue decomposition, will necessarily transform the matrices into fully populated data arrays which quickly becomes unfeasible in terms of memory requirements. Secondly, these methods base on matrix operations that do not scale well with the problem dimension, which heavily increases the running time of algorithms. As in the dense case, there exists a variety of methods for the classical LQR/LQG case ($B_1 = 0$), for example, the low-rank Newton method [12, 45] with the low-rank alternating direction implicit method [12, 25, 30] for solving the occurring large-scale sparse Lyapunov equations (LR-Newton-ADI), the Riccati alternating direction implicit method (RADI) [7], the incremental low-rank subspace iteration (ILRSI) [31], and projection-based methods that construct approximating subspaces such that internally small-scale dense Riccati equations need to be solved, e.g., [22, 41]. See also [8, 13, 25] for overviews and numerical comparisons of large-scale sparse solvers for this special case of Riccati equations with negative semi-definite quadratic terms. For the opposite case of Riccati equations with positive semi-definite quadratic terms ($B_2 = 0$), only the Newton method from [14] is known to have a low-rank extension to the large-scale sparse matrix case. However, there are no methods known to solve the general case (1) with an indefinite quadratic term in the large-scale sparse setting.

The goal of this paper is to develop an extension of the approach described in [26, 27] that can be applied in the case of large-scale sparse coefficient matrices. With such an algorithm at hand, in particular, the design of so-called $\mathcal{H}_\infty$-robust controllers based on the state-space theory from the 1990s [33, 47] may become generally accessible for large-scale systems. This working hypotheses we find well backed up by the close connection between the existence of such controllers and the sufficient conditions for convergence of the algorithm of [26, 27]. Under some reasonable assumptions, it is known that if an $\mathcal{H}_\infty$-robust controller $\mathcal{K}\colon y \mapsto u$ that stabilizes the system (2) with a robustness bound γ_0 exists, then the Riccati equation (1) has a unique symmetric positive semi-definite solution X_∞ for any

$\gamma > \gamma_0$; cf. [47]. We will point out in detail that the sufficient conditions for convergence of the proposed algorithm coincide with the assumptions made for $\mathcal{H}_\infty$-controller design. Accordingly, we can state that success of the proposed method is in one-to-one correspondence with the existence of the relevant solution.

The contents of this paper are organized as follows. In the upcoming Section 2, we first recap the algorithm from [26,27] and summarize some important results. Afterwards, we reformulate the steps of the algorithm to fit the large-scale sparse system case and extend it further to coefficients arising from systems with differential-algebraic equations. In Section 3, we test the new algorithm first on Riccati equations with dense coefficients, for a comparison to other established methods, and afterwards on equations with large-scale sparse coefficients. The paper is concluded in Section 4.

2. Riccati iteration method

In this section, we describe the idea of the Riccati iteration method from [26,27] and extend the approach to large-scale sparse systems. For generality and brevity of the paper, we immediately consider the generalized formulation of (1) involving an additional $E \in \mathbb{R}^{n\times n}$ matrix as shown in Problem 2.1.

Thereby, we will use the following notation. We denote symmetric positive semi-definite matrices by $X = X^\mathsf{T} \geq 0$, for $X \in \mathbb{R}^{n\times n}$, and write $X_1 \geq X_2$ for two symmetric matrices $X_1, X_2 \in \mathbb{R}^{n\times n}$, if $X_1 - X_2 \geq 0$. We call a matrix triple (A, B, E) stabilizable, with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and $E \in \mathbb{R}^{n\times n}$ invertible, if there exists a feedback matrix $K \in \mathbb{R}^{m\times n}$ such that the matrix pencil $\lambda E - (A - BK)$ has only eigenvalues with negative real parts.

2.1. Basic algorithm.

In this section, we formulate the fundamental algorithm for the iterative computation of approximate solutions to the following general problem.

Problem 2.1 (Stabilizing solutions of indefinite Riccati equations)**.** Given matrices $A \in \mathbb{R}^{n\times n}$, $B_1 \in \mathbb{R}^{n\times m_1}$, $B_2 \in \mathbb{R}^{n\times m_2}$, $C \in \mathbb{R}^{p\times n}$, and $E \in \mathbb{R}^{n\times n}$ invertible, where $n, m_1, m_2, p \in \mathbb{N}$, compute a matrix $X = X_\infty \in \mathbb{R}^{n\times n}$, if it exists, which

(a) solves the Riccati equation with indefinite quadratic term

$$A^\mathsf{T} X E + E^\mathsf{T} X A + E^\mathsf{T} X \big(B_1 B_1^\mathsf{T} - B_2 B_2^\mathsf{T}\big) X E + C^\mathsf{T} C = 0, \tag{3}$$

and

(b) is symmetric positive semi-definite and such that the matrix pencil

$$\lambda E - \big(A + (B_1 B_1^\mathsf{T} - B_2 B_2^\mathsf{T}) X_\infty E\big)$$

is stable, i.e., all its eigenvalues lie in the open left half-plane.

Remark 2.2 (Reformulated Riccati equation)**.** *If compared to the Riccati equation (1) from $\mathcal{H}_\infty$-control theory introduced before, the indefinite Riccati equation (3) that we use for the derivation and display of the solution approach, differs in the additional E matrix and the missing factor γ^{-2}. While the γ-term is a simple scaling of the B_1 matrix, an invertible E can be readily resolved in the standard form by considering AE^{-1} and CE^{-1} as coefficients rather than A and C. However, in many applications the inversion of E should be avoided, so that we rather state the equivalent equations with E explicitly given.*

Algorithm 1: Riccati iteration (RI).

```
Input: $A, B_1, B_2, C, E$ from (3), convergence tolerance $\tau$.
Output: Stabilizing approximate solution $X_k \approx X_\infty$ of (3), with
        $X_k = X_k^\mathsf{T} \geq 0$.
1  Initialize $X_0 = 0$, $k = 0$.
2  repeat
3      Update the iteration matrix $A_k = A + (B_1 B_1^\mathsf{T} - B_2 B_2^\mathsf{T}) X_k E$.
4      Solve the definite Riccati equation
           $A_k^\mathsf{T} W_{k+1} E + E^\mathsf{T} W_{k+1} A_k - E^\mathsf{T} W_{k+1} B_2 B_2^\mathsf{T} W_{k+1} E + \mathcal{R}(X_k) = 0$
       for the stabilizing solution $W_{k+1} \in \mathbb{R}^{n \times n}$.
5      Update the solution matrix $X_{k+1} = X_k + W_{k+1}$.
6      if $(A + B_1 B_1^\mathsf{T} X_{k+1} E, B_2, E)$ is stabilizable then
7          Increment $k \leftarrow k + 1$.
8      else
9          error // There is no stabilizing $X_\infty = X_\infty^\mathsf{T} \geq 0$ that
               solves (3).
10     end
11 until $\|B_1 W_k E\|_2 < \tau$
```

For computing the solution to Problem 2.1 numerically, the *Riccati iteration* method was first described in [26, 27] for the standard equation case ($E = I_n$). The underlying idea of the algorithm is to consider the Riccati operator

$$\mathcal{R}(X) := A^\mathsf{T} X E + E^\mathsf{T} X A - E^\mathsf{T} X (B_2 B_2^\mathsf{T} - B_1 B_1^\mathsf{T}) X E + C^\mathsf{T} C, \tag{4}$$

associated with (3), and a splitting of the final solution into the sum of consecutive solutions of updated algebraic Riccati equations with semi-definite quadratic terms. For two symmetric matrices $X_1 = X_1^\mathsf{T}$ and $X_2 = X_2^\mathsf{T}$, one can show that

$$\mathcal{R}(X_1 + X_2) = \mathcal{R}(X_1) + \widetilde{A}^\mathsf{T} X_2 E + E^\mathsf{T} X_2 \widetilde{A} - E^\mathsf{T} X_2 (B_2 B_2^\mathsf{T} - B_1 B_1^\mathsf{T}) X_2 E$$

holds, where $\widetilde{A} := A - (B_2 B_2^\mathsf{T} - B_1 B_1^\mathsf{T}) X_1 E$. Accordingly, for X_2 as a solution to the algebraic Riccati equation with negative semi-definite quadratic term

$$0 = \mathcal{R}(X_1) + \widetilde{A}^\mathsf{T} X_2 E + E^\mathsf{T} X_2 \widetilde{A} - E^\mathsf{T} X_2 B_2 B_2^\mathsf{T} X_2 E, \tag{5}$$

the residual reads

$$\mathcal{R}(X_1 + X_2) = E^\mathsf{T} X_2 B_1 B_1^\mathsf{T} X_2 E.$$

An iterative use of this relation (5), together with the initial approximate solution $X_0 = 0$, yields the *Riccati iteration (RI)* method. The resulting algorithm is summarized in Algorithm 1.

The following proposition lays out the theoretical foundation of the Riccati iteration method.

Proposition 2.3 (Properties of the Riccati iteration [26, 27])**.** *If (A, B_2, E) is stabilizable, (A, C, E) has no unobservable purely imaginary modes and there exists a stabilizing solution $X_\infty = X_\infty^\mathsf{T} \geq 0$ for (3), then the following statements hold for the iteration in Algorithm 1:*

(a) $(A + B_1 B_1^\mathsf{T} X_k E, B_2, E)$ is stabilizable for all $k = 0, 1, \ldots$,

(b) $W_k = W_k^{\mathsf{T}} \geq 0$ *for all* $k = 0, 1, \ldots,$
(c) *the eigenvalues of the matrix pencil* $\lambda E - (A + B_1 B_1^{\mathsf{T}} X_k E - B_2 B_2 X_{k+1} E)$ *lie in the left open half-plane for all* $k = 0, 1, \ldots,$
(d) $\mathcal{R}(X_{k+1}) = E^{\mathsf{T}} W_{k+1} B_1 B_1^{\mathsf{T}} W_{k+1} E$ *for all* $k = 0, 1, \ldots,$
(e) $X_\infty \geq \ldots \geq X_{k+1} \geq \ldots \geq X_1 \geq X_0 = 0,$
(f) *the iteration converges to the stabilizing solution of* (3), $\lim_{k\to\infty} X_k = X_\infty$, *and*
(g) *the convergence is locally quadratic.*

Basically, the conditions of Proposition 2.3 guarantee convergence of the iterates to the desired solution of Problem 2.1 as well as the stabilization property for every intermediate iteration step. The following remark gives some insight about the convergence of the stabilizing solutions of (3) and the convergence of the Riccati iteration.

Remark 2.4 (Definiteness of stabilizing solutions)**.** *The Riccati iteration converges only if the stabilizing solution exists and if it is symmetric positive semi-definite, which follows from Parts (e) and (f) of Proposition 2.3 and classical theory about the LQR/LQG Riccati equations that are solved in every iteration step. In general, if a real stabilizing solution of* (3) *exists, it does not need to be symmetric positive semi-definite; see examples in [26]. Other approaches that rely on the underlying Hamiltonian matrix pencil of* (3) *are capable of computing also indefinite stabilizing solutions.*

However, for the design of $\mathcal{H}_\infty$*-robust controllers, the existence of a symmetric positive semi-definite solution is both sufficient and necessary for the well-posedness of the controller design problem. In other words, such a controller exists if and only if the related indefinite Riccati equation has a symmetric positive semi-definite solution. Even more, the assumptions made for the convergence of the Riccati iteration are equivalent to the fundamental assumptions made for* $\mathcal{H}_\infty$*-controller synthesis. In fact, if only the solvability of the (regulator) Riccati equation* (1) *is discussed, the conditions listed in [47, Chap. 17.1] reduce to* (A, B_2) *being stabilizable and*

$$\begin{bmatrix} A - sI_n \\ C \end{bmatrix}$$

having full column rank for all purely imaginary $s \in \mathbb{C}$ *that are readily converted into the assumptions made for Proposition 2.3.*

2.2. Factorized low-rank formulation for large-scale sparse equations.

The Riccati iteration method in Algorithm 1 cannot be directly applied to the large-scale case. Most importantly, the iterates X_k will be dense $n \times n$-matrices such that memory will become a limiting factor already for moderate dimensions n. With the guarantee that all intermediate solutions X_k as well as all updates W_{k+1} are symmetric positive semi-definite (cf. Parts (a) and (e) of Proposition 2.3), the approximation by low-rank factorizations provides a potential remedy as in the case of Riccati equations with symmetric semi-definite quadratic terms; see, e.g., [8, 13, 25]. The basic idea is to rewrite the intermediate stages of the Riccati iteration via Cholesky-like low-rank factorizations such that $X_k \approx Z_k Z_k^{\mathsf{T}}$ and $W_{k+1} \approx Y_{k+1} Y_{k+1}^{\mathsf{T}}$, with $Z_k \in \mathbb{R}^{n \times r_k}$ and $Y_{k+1} \in \mathbb{R}^{n \times q_{k+1}}$ for all $k = 0, 1, \ldots,$ and the relevant equations in terms of the factors Z_k and Y_k. In particular, with the intermediate solution in

Step 6 of Algorithm 1 reformulated as

$$Z_{k+1}Z_{k+1}^{\mathsf{T}} \approx X_{k+1} = X_k + W_{k+1} \approx Z_k Z_k^{\mathsf{T}} + Y_{k+1}Y_{k+1}^{\mathsf{T}} = \begin{bmatrix} Z_k & Y_{k+1} \end{bmatrix} \begin{bmatrix} Z_k^{\mathsf{T}} \\ Y_{k+1}^{\mathsf{T}} \end{bmatrix},$$

we will develop an iteration for the solution factors without ever forming the full solution explicitly.

The residual Riccati equation in Step 4 of Algorithm 1 that defines the update W_{k+1} is given via

$$A_k^{\mathsf{T}} W_{k+1} E + E^{\mathsf{T}} W_{k+1} A_k - E^{\mathsf{T}} W_{k+1} B_2 B_2^{\mathsf{T}} W_{k+1} E + \mathcal{R}(X_k) = 0, \tag{6}$$

and we observe that

(1) in the initial step with $X_0 = 0$, we have $\mathcal{R}(X_0) = C^{\mathsf{T}} C$, and
(2) for all all further steps, with Part (d) of Proposition 2.3, it holds

$$\mathcal{R}(X_{k+1}) = E^{\mathsf{T}} W_{k+1} B_1 B_1^{\mathsf{T}} W_{k+1} E = \underbrace{\left(B_1^{\mathsf{T}} W_{k+1} E\right)^{\mathsf{T}}}_{\in \mathbb{R}^{n \times m_1}} \underbrace{\left(B_1^{\mathsf{T}} W_{k+1} E\right)}_{\in \mathbb{R}^{m_1 \times n}},$$

for all $k = 0, 1, \ldots$, and where $W_{k+1} \approx Y_{k+1} Y_{k+1}^{\mathsf{T}}$.

In both cases, this central step of the algorithm requires the solve of a standard (semi-definite) Riccati equation with low-rank factorized quadratic and constant terms. Accordingly, a low-rank factorized approximation to X_{k+1} can be computed by established Riccati equation solvers, like the LR-Newton-ADI method [12, 45], RADI [7], ILRSI [31], or projection-based methods [22, 41].

Remark 2.5 (Low-rank solutions)**.** *By the low-rank structure of the right hand-side and the quadratic terms, $m_1, m_2, p \ll n$, we can also expect the solution and update terms to be well approximated by low-rank factors such that $r_k, q_k \ll n$; see [6, 42].*

Combining all these ideas leads now to the *low-rank Riccati iteration (LR-RI)* method in Algorithm 2. Algorithm 2 has been implemented for dense coefficients in the MORLAB toolbox [16] and for the large-scale sparse case in the M-M.E.S.S. toolbox [38].

Another difference of Algorithm 2 compared to Algorithm 1 is the stabilizability test for $(A + B_1 B_1^{\mathsf{T}} X_{k+1} E, B_2, E)$. This additional test is expensive in the large-scale setting but can, in principle, be omitted. In case of $(A + B_1 B_1^{\mathsf{T}} X_{k+1} E, B_2, E)$ not being stabilizable in intermediate steps, the iteration will diverge and there might be no stabilizing solutions for the intermediate Riccati equations with negative semi-definite quadratic terms anymore. The following remarks state further computational features of the new low-rank iteration method in Algorithm 2.

Remark 2.6 (Unstable closed-loop matrix pencils)**.** *The intermediate closed-loop matrix pencils $sE - A_k$ can potentially have unstable eigenvalues [27]. However, in case there exists a stabilizing solution $X_\infty = X_\infty^{\mathsf{T}} \geq 0$ of (3), (A_k, B_2, E) will be stabilizable. Many of the low-rank solvers for definite Riccati equations need a stabilizing initial solution or a corresponding feedback matrix. In the large-scale sparse case, this can be computed using a sparse eigenvalue solver to compute the eigenvectors corresponding to the unstable eigenvalues of $sE - A_k$ and to apply a partial stabilization approach [5] on the projected problem using the computed eigenvalue basis [3]. Note that if $sE - A_{k_0}$ is stable for some k_0 and the iteration converges, then $sE - A_k$ will also be stable for all $k \geq k_0$; cf. Proposition 2.3.*

Algorithm 2: Low-rank Riccati iteration (LR-RI).

```
Input: $A, B_1, B_2, C, E$ from (3), convergence tolerance $\tau$.
Output: Low-rank factor $Z_k$ s.t. $Z_k Z_k^{\mathsf{T}} \approx X_\infty$ is the stabilizing solution
        of (3).
1 Solve the Riccati equation with negative semi-definite quadratic term
      $A^{\mathsf{T}} W_0 E + E^{\mathsf{T}} W_0 A - E^{\mathsf{T}} W_0 B_2 B_2^{\mathsf{T}} W_0 E + C^{\mathsf{T}} C = 0,$
  for the low-rank factor $Y_0$ such that $Y_0 Y_0^{\mathsf{T}} \approx W_0$.
2 Initialize $Z_0 = Y_0$, $U = \begin{bmatrix} B_1 & -B_2 \end{bmatrix}$ and $k = 0$.
3 while $\|B_1^{\mathsf{T}} Y_k Y_k^{\mathsf{T}} E\|_2 > \tau$ do
4     Set $A_k = A - U V_k^{\mathsf{T}}$ with the updated low-rank factor
          $V_k = \begin{bmatrix} E^{\mathsf{T}} Z_k Z_k^{\mathsf{T}} B_1 & E^{\mathsf{T}} Z_k Z_k^{\mathsf{T}} B_2 \end{bmatrix}.$
5     Solve the Riccati equation with negative semi-definite quadratic term
          $A_k^{\mathsf{T}} W_{k+1} E + E^{\mathsf{T}} W_{k+1} A_k - E^{\mathsf{T}} W_{k+1} B_2 B_2^{\mathsf{T}} W_{k+1} E + \left(E^{\mathsf{T}} W_k B_1\right)\left(E^{\mathsf{T}} W_k B_1\right)^{\mathsf{T}} = 0,$
      for the low-rank factor $Y_{k+1}$ with $Y_{k+1} Y_{k+1}^{\mathsf{T}} \approx W_{k+1}$.
6     Update $Z_{k+1} = \begin{bmatrix} Z_k & Y_{k+1} \end{bmatrix}$ and increment $k \leftarrow k + 1$.
7 end
```

Remark 2.7 (Riccati equations with positive semi-definite quadratic terms)**.** *In principle, the Riccati iteration method can be used as an alternative approach to solve Riccati equations with positive semi-definite quadratic terms by setting $B_2 = 0$:*

$$A^{\mathsf{T}} X E + E^{\mathsf{T}} X A + E^{\mathsf{T}} X B_1 B_1^{\mathsf{T}} X E + C^{\mathsf{T}} C = 0,$$

which occur, for example, in [1, 35]. A necessary condition for applying the Riccati iteration method is in this case the stability of the matrix pencil $sE - A$, since otherwise the stabilizability of (A, B_2, E) will never be fulfilled. This restriction does not occur in the Newton iteration from [14, 44], where only a stabilizing initial feedback is needed.

2.3. **Realization of linear solves in factored form.**

In general, the coefficients $A_k = A + (B_1 B_1^{\mathsf{T}} - B_2 B_2^{\mathsf{T}}) X_k E$, for $k = 1, 2, \ldots$, of the intermediate Riccati equations in Step 4 of Algorithm 2 are $n \times n$-matrices without sparsity structures such that an explicit realization would make the approach infeasible in the large-scale setting. In this case, the coefficients need to be rewritten as the sparse system matrix A plus low-rank update:

$$A_k = A + (B_1 B_1^{\mathsf{T}} - B_2 B_2^{\mathsf{T}}) X_k E = A + \begin{bmatrix} B_1 & -B_2 \end{bmatrix} \begin{bmatrix} B_1^{\mathsf{T}} X_k E \\ B_2^{\mathsf{T}} X_k E \end{bmatrix} =: A + U V_k^{\mathsf{T}}, \tag{7}$$

with $U, V_k \in \mathbb{R}^{n \times (m_1 + m_2)}$, for all $k = 1, 2, \ldots$. Thereby, matrix-vector multiplications with A_k can be performed without resorting to an explicit formation of A_k.

For the use within sparse direct solvers, this factored representation of A_k in (7) can be exploited as follows. Consider the shifted linear system

$$(\sigma E^{\mathsf{T}} - A_k^{\mathsf{T}}) X = (\sigma E^{\mathsf{T}} - A^{\mathsf{T}} - V_k U^{\mathsf{T}}) X = F, \tag{8}$$

for a *slim* right-hand side $F \in \mathbb{R}^{n \times \ell}$, with $\ell \ll n$, and a shift $\sigma \in \mathbb{C}$. The solution of such linear systems (8) are the backbone of standard iterative solvers for large-scale Riccati equations. By the *Sherman-Morrison-Woodbury formula* for matrix inversion [19] and with the abbreviation $\Phi(\sigma) := (\sigma E^\mathsf{T} - A^\mathsf{T})$, the inverse of the shifted matrix in (8) is given by

(9)
$$\big(\Phi(\sigma) - V_k U^\mathsf{T}\big)^{-1} = \Phi(\sigma)^{-1} + \Phi(\sigma)^{-1} V_k \big(I_{m_1+m_2} - U^\mathsf{T}\Phi(\sigma)^{-1} V_k\big)^{-1} U^\mathsf{T}\Phi(\sigma)^{-1},$$

which actually amounts to 2ℓ sparse linear solves with $\sigma E^\mathsf{T} - A^\mathsf{T}$ and one solve with an $(m_1+m_2)\times(m_1+m_2)$-dimensional matrix. Thus, in a practical realization of (9) for (8), one would compute

$$X = Z_1 + Z_2\big(I_{m_1+m_2} - U^\mathsf{T} Z_2\big)^{-1} U^\mathsf{T} Z_1,$$

with $(\sigma E^\mathsf{T} - A^\mathsf{T})Z_1 = F$, and $(\sigma E^\mathsf{T} - A^\mathsf{T})Z_2 = V_k$.

An alternative to the Sherman-Morrison-Woodbury formula is the augmented matrix approach, which makes use of the *block matrix inversion formula.* This approach is, often times, more stable than the Sherman-Morrison-Woodbury formula. Thereby, the solution of (8) is given as the solution of an augmented system of linear equations with

$$\begin{bmatrix} \sigma E^\mathsf{T} - A^\mathsf{T} & V_k \\ U^\mathsf{T} & I_{m_1+m_2} \end{bmatrix} \begin{bmatrix} X \\ X^\perp \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix}, \tag{10}$$

where the lower block of the solution, $X^\perp$, is an auxiliary variable with no relevance for the solution of (8). Under the assumption that U and V_k have much fewer columns than n, systems of the form (10) can still be solved with standard sparse solvers.

2.4. **Singular E matrices and projected Riccati equations.**

A regularly occurring case in applications involves singular E matrices. These especially occur in control problems with differential-algebraic equations (DAEs). In general, the presence of singular E matrices changes the concepts of stabilizability of matrix pencils and solvability of Riccati equations; see, e.g., [10,34] and references therein. However, in many cases, the singular part of E does not play any substantial role and it is actually enough to consider the solution of Riccati equations restricted to subspaces corresponding to the finite eigenvalues of the matrix pencil $\lambda E - A$, i.e., a restriction to the invertible part of E. This can be realized in two different concepts:

(1) consider a projected version of the Riccati equation (3) with additional constraints on the stabilizing solution,
(2) consider a truncated version of the Riccati equation (3).

In the following subsections, we consider first the general ideas for projected and truncated Riccati equations and, afterwards, the setup of incompressible flows as a particular example.

2.4.1. *Projected and truncated equations.*

In the first concept, appropriate spectral projections $\mathcal{P}_r, \mathcal{P}_\ell \in \mathbb{R}^{n \times n}$ onto the right and left deflating subspaces corresponding to the finite eigenvalues of $\lambda E - A$ are explicitly introduced in (3) such that

$$A^{\mathsf{T}} X E + E^{\mathsf{T}} X A + E^{\mathsf{T}} X \left(B_1 B_1^{\mathsf{T}} - B_2 B_2^{\mathsf{T}} \right) X E + \mathcal{P}_r^{\mathsf{T}} C^{\mathsf{T}} C \mathcal{P}_r = 0, \tag{11a}$$

$$\mathcal{P}_\ell^{\mathsf{T}} X \mathcal{P}_\ell = X, \tag{11b}$$

needs to be solved instead. While in general, such projections $\mathcal{P}_r, \mathcal{P}_\ell$ can be constructed using decompositions of the matrix pencil $\lambda E - A$ that resemble the *Weierstrass canonical form*, they are in fact known for several specially structured problems that arise in the large-scale sparse setting; see the examples in [43]. Note that the E matrix in (11) is still singular, but the equation and its solutions are restricted to the appropriate underlying subspace. Iterative methods like Algorithm 2 can be used to solve (11), where only the application of the spectral projections $\mathcal{P}_r$ and $\mathcal{P}_\ell$ is needed in the intermediate computational steps. This is also the common drawback of solving (11), since the repeated application of the projections can quickly become expensive, especially in the large-scale sparse case.

The second concept of truncated Riccati equations is more commonly used in practice. Thereby, we consider in general Riccati equations of the form

$$\widehat{A}^{\mathsf{T}} \widehat{X} \widehat{E} + \widehat{E}^{\mathsf{T}} \widehat{X} \widehat{A} + \widehat{E}^{\mathsf{T}} \widehat{X} \left(\widehat{B}_1 \widehat{B}_1^{\mathsf{T}} - \widehat{B}_2 \widehat{B}_2^{\mathsf{T}} \right) \widehat{X} \widehat{E} + \widehat{C}^{\mathsf{T}} \widehat{C} = 0, \tag{12}$$

where the truncated matrices are constructed by

$$\widehat{A} = W^{\mathsf{T}} A V, \quad \widehat{E} = W^{\mathsf{T}} E V, \quad \widehat{B} = W^{\mathsf{T}} B, \quad \widehat{C} = C V, \tag{13}$$

with the coefficient matrices from (3) and $V, W \in \mathbb{R}^{n \times r}$, basis matrices of the right and left deflating subspaces corresponding to the finite eigenvalues of $\lambda E - A$. By construction, the $\widehat{E}$ matrix in (12) is nonsingular and the solution techniques mentioned so far for (3) can also be applied to (12). These basis matrices V and W can be computed and applied explicitly to (3) up to medium-scale coefficient matrices. This is the basis of the implementation of model reduction methods for descriptor systems in the MORLAB toolbox [16] and explained in [15].

However, the explicit computation of the basis matrices V and W and of the resulting truncated coefficient matrices (13) is usually not possible in the large-scale sparse setting because of computation time and memory limitations. In this case, for many special sparse structures, the truncated Riccati equation (12) can be realized implicitly during the computations, i.e., instead of applying Algorithm 2 directly to (12), the original sparse coefficients from (3) are used and, by means of their structure, V and W are implicitly applied during the computational steps. Examples for this implicit truncation for different sparsity structures are given in [3, 18, 21, 39], which are also implemented in the function handle framework of the M-M.E.S.S. toolbox [38]. Consequently, the implementation of Algorithm 2 in the M-M.E.S.S. toolbox can make use of the implicit truncation in the case of singular E matrices. As a particular example, the following subsection considers the case of structured coefficient matrices arising from incompressible flows.

2.4.2. *Implicit realization of truncations in case of flow problems.* Riccati equations with indefinite quadratic terms (3) are of particular interest in the design of robust controllers for the stabilization of incompressible flows modeled by linearization of the Navier-Stokes equations [11]. We briefly touch this particular case as an example for implicit truncation since we will consider such a numerical example later. In this case, the coefficient matrices of (3) are structured and given by

$$A = \begin{bmatrix} \widetilde{A} & J^{\mathsf{T}} \\ J & 0 \end{bmatrix}, \quad E = \begin{bmatrix} \widetilde{E} & 0 \\ 0 & 0 \end{bmatrix}, \quad B_1 = \begin{bmatrix} \widetilde{B}_1 \\ 0 \end{bmatrix}, \quad B_2 = \begin{bmatrix} \widetilde{B}_2 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} \widetilde{C} & 0 \end{bmatrix}, \tag{14}$$

with $\widetilde{E}$ symmetric and invertible. The key to implicitly truncate and project Riccati equations with coefficients like (14) is the *discrete Leray projection*

$$\Pi = I_{\tilde{n}} - \widetilde{E}^{-1} J^{\mathsf{T}} (J \widetilde{E}^{-1} J^{\mathsf{T}})^{-1} J; \tag{15}$$

see [21] for the most general case. Then, the truncated and projected Riccati equation that needs to be solved has the form

$$\Pi^{\mathsf{T}} \widetilde{A}^{\mathsf{T}} \Pi \widehat{X} \widetilde{E} + \widetilde{E}^{\mathsf{T}} \widehat{X} \Pi^{T} \widetilde{A} \Pi + \widetilde{E}^{\mathsf{T}} \widehat{X} \Pi^{\mathsf{T}} \left(\widetilde{B}_1 \widetilde{B}_1^{\mathsf{T}} - \widetilde{B}_2 \widetilde{B}_2^{\mathsf{T}} \right) \Pi \widehat{X} \widetilde{E} + \Pi^{\mathsf{T}} \widetilde{C}^{\mathsf{T}} \widetilde{C} \Pi = 0, \tag{16}$$

where $\widehat{X} = \Pi \widehat{X} \Pi^{\mathsf{T}}$; see [11, Sec. 3.1].

As mentioned in Section 2.4.1, in the large-scale setting, we can neither explicitly compute the matrices in (16) nor the projection (15). However, instead one can use the original coefficient matrices with their special structure (14). As mentioned in Section 2.3, we practically need to solve linear systems like (8) in every step of the low-rank Riccati iteration. If we consider the coefficients of (16) in this setting, the low-rank update matrix (7) has the form

$$\begin{aligned} &\Pi^{\mathsf{T}} \widetilde{A}^{\mathsf{T}} \Pi + \widetilde{E}^{\mathsf{T}} \widehat{X}_k \Pi^{\mathsf{T}} (\widetilde{B}_1 \widetilde{B}_1^{\mathsf{T}} - \widetilde{B}_2 \widetilde{B}_2^{\mathsf{T}}) \Pi \\ &= \Pi^{\mathsf{T}} (\widetilde{A}^{\mathsf{T}} + \widetilde{E}^{\mathsf{T}} \widehat{X}_k (\widetilde{B}_1 \widetilde{B}_1^{\mathsf{T}} - \widetilde{B}_2 \widetilde{B}_2^{\mathsf{T}})) \Pi \\ &= \Pi^{\mathsf{T}} \widetilde{A}_k \Pi, \end{aligned}$$

where we also used that $\widetilde{E} \Pi = \Pi^{\mathsf{T}} \widetilde{E}$ and $\Pi^2 = \Pi$. By observing that all occurring right-hand sides will satisfy $F = \Pi F$, the solution of $(\sigma \widetilde{E}^{\mathsf{T}} - \Pi^{\mathsf{T}} \widetilde{A}_k^{\mathsf{T}} \Pi) X = F$ is alternatively given by

$$\begin{bmatrix} \sigma \widetilde{E}^{\mathsf{T}} - \widetilde{A}^{\mathsf{T}} + \widetilde{E}^{\mathsf{T}} \widehat{X}_k (\widetilde{B}_1 \widetilde{B}_1^{\mathsf{T}} - \widetilde{B}_2 \widetilde{B}_2^{\mathsf{T}}) & -J^{\mathsf{T}} \\ -J & 0 \end{bmatrix} \begin{bmatrix} X \\ X^{\perp} \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix}, \tag{17}$$

where $X^{\perp}$ is an auxiliary variable that does not play any role. This ensures that $X = \Pi X$ as required during the iteration; see [21, Lem. 5.2]; and respects the sparsity structure of (14), since (17) is actually given by

$$\left(\sigma E^{\mathsf{T}} - A^{\mathsf{T}} + E^{\mathsf{T}} \begin{bmatrix} \widehat{X}_k & 0 \\ 0 & 0 \end{bmatrix} (B_1 B_1^{\mathsf{T}} - B_2 B_2^{\mathsf{T}}) \right) \begin{bmatrix} X \\ X^{\perp} \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix},$$

using the matrices from (14).

3. Numerical experiments

The numerical experiments reported in this section have been executed on a machine with 2 Intel(R) Xeon(R) Silver 4110 CPU processors running at 2.10 GHz and equipped with 192 GB total main memory. The computer runs on CentOS Linux release 7.5.1804 (Core) with MATLAB 9.9.0.1467703 (R2020b).

The benchmark data used can be found in Table 1, where the first four columns state the dimensions of the considered problem; cf. (3); $\#\lambda_{\text{unstab}}$ denotes the number of unstable eigenvalues of the matrix pencil $sE - A$ and in the data column the example type is denoted with either dense or sparse. The last column shows

Table 1. Results for the aircraft and cable mass benchmarks.

	n	m_1	m_2	p	$\#\lambda_{\text{unstab}}$	data	γ
`AC10`	55	3	2	5	2	dense	2.00
`CM6`	960	1	1	3	76	dense	2.00
`rand512`	512	2	6	5	8	dense	8.00
`rand1024`	1 024	4	2	3	6	dense	8.00
`rand2048`	2 048	3	3	1	2	dense	7.00
`rand4096`	4 096	4	3	6	1	dense	10.00
`rail`	79 841	3	3	6	0	sparse	2.00
`cylinderwake`	47 136	1	1	6	4	sparse	50.00

γ-values; cf. (1); by which the B_1 matrices have been scaled such that the considered Riccati equations have symmetric positive semi-definite stabilizing low-rank solutions.

For the comparison of results between different algorithms and benchmarks, we will report different information about the performance of the applied solvers in upcoming tables such as the used iteration steps, the overall runtime in seconds, the rank of the resulting solution factor, the final normalized residual of the Riccati iteration as given by $\|B_1 Y_k Y_k^{\mathsf{T}} E\|_2^2 / \|CC^{\mathsf{T}}\|_2$, the relative residual of the solution factor given by $\|\mathcal{R}(Z_k Z_k^{\mathsf{T}})\|_2 / \|Z_k^{\mathsf{T}} Z_k\|_2$ and the normalized residual of the solution factor computed as $\|\mathcal{R}(Z_k Z_k^{\mathsf{T}})\|_2 / \|CC^{\mathsf{T}}\|_2$ using the Riccati operator $\mathcal{R}$ from (4), and lastly the norm of the full solution $\|Z_k^{\mathsf{T}} Z_k\|_2$.

3.1. **Factorized method for dense examples.**

Before we actually consider the large-scale sparse case, we test the new method from Algorithm 2 on medium-scale dense benchmarks and compare the results with two other commonly known solvers that can be applied to (3). We use the dense low-rank factor version of Algorithm 2 as implemented in `ml_icare_ric_fac` of [16], which we denote further on as LRRI, but we replace the internal Riccati equation solver by the factorized sign function approach [9] implemented in the very same toolbox [16] as `ml_caredl_sgn_fac`. For comparison, we use the Hamiltonian eigenvalue approach from [2] as it is implemented in the new MATLAB function `icare` from the Control System Toolbox™, further denoted by ICARE, and the classical (unfactorized) sign function solver [36] from the MORLAB toolbox [16] in `ml_caredl_sgn`, further on as SIGN. Since SIGN and ICARE can only compute unfactorized solutions of (3), we perform eigendecompisitions of the computed solutions to check the positive semi-definiteness and to compute low-rank approximations of the solutions by truncating all components corresponding to non-positive eigenvalues.

Considering the benchmarks, the first two data sets in Table 1 are practical examples taken from [29], where `AC10` is an aircraft model and `CM6` a cable mass model with low damping. The data was taken over with exactly the same naming as given in [29], where we have set $C = C_1$. The results of the different algorithms on these two data sets are shown in Table 2. First, we recognize that for `AC10`, LRRI performs visibly worse than the other two approaches in terms of accuracy, since we are loosing three orders of magnitude in the relative and normalized residuals.

Table 2. Results for the aircraft and cable mass benchmarks.

	AC10			CM6		
	LRRI	ICARE	SIGN	LRRI	ICARE	SIGN
Iteration steps	5	—	19	4	—	23
Runtime (s)	0.92801	0.47971	0.07705	29.9208	140.136	7.24722
Rank Z_k	53	55	55	569	758	781
Final res.	5.545e-25	—	—	1.873e-15	—	—
Relative res.	2.599e-07	9.617e-10	1.183e-09	1.910e-09	5.019e-08	2.125e-07
Normalized res.	1.554e-03	5.752e-06	7.074e-06	1.667e-05	4.381e-04	1.855e-03
$\lVert Z_k^{\mathsf{T}} Z_k \rVert_2$	1.457e+01	1.457e+01	1.457e+01	1.253e+04	1.253e+04	1.253e+04

Table 3. Results for the smaller random examples.

	rand512			rand1024		
	LRRI	ICARE	SIGN	LRRI	ICARE	SIGN
Iteration steps	8	—	14	7	—	8
Runtime (s)	6.94609	23.0720	1.39393	10.3911	217.230	2.96977
Rank Z_k	96	298	294	89	522	512
Final res.	7.889e-23	—	—	2.600e-14	—	—
Relative res.	7.544e-10	2.926e-11	3.508e-11	6.737e-11	6.980e-11	1.074e-10
Normalized res.	3.930e-10	1.524e-11	1.827e-11	6.164e-12	6.387e-12	9.830e-12
$\lVert Z_k^{\mathsf{T}} Z_k \rVert_2$	3.202e+02	3.202e+02	3.202e+02	1.058e+02	1.058e+02	1.058e+02

We assume this comes from the bad conditioning of the A matrix in this example and the repeated solution of Riccati equations with the matrix. However, for the CM6 example, this turned around, as LRRI performs now an order of magnitude better than ICARE and two orders better than SIGN. Concerning the computation times we see that LRRI performs reasonably well thanks to the efficient inner Riccati equation solver. While on the smaller data set AC10, the runtime of LRRI is in the same order of magnitude as ICARE, LRRI is already four times faster than ICARE for CM6. We are not able to outperform SIGN with LRRI in these examples.

To further investigate accuracy and performance of LRRI, we created random examples denoted by rand* in Table 1 using the randn function in MATLAB. The results of the computations can be found in Tables 3 and 4. Overall, LRRI competes very well against ICARE and SIGN in terms of accuracy. For all four presented random examples, the residuals of the solution factors lie in the same order of magnitude for all methods with only minor exceptions. In terms of runtimes, we see the same relations that we already recognized in Table 2. LRRI easily outperforms ICARE due to its efficient inner factorized sign function solver. For increasing problem size also this speed-up tremendously increases further up to the largest example rand4096, where LRRI is 113 times faster than ICARE. Compared to SIGN, LRRI is still not able to outperform the classical sign function iteration method. However, we can observe that the difference in runtimes is getting smaller and smaller, i.e.,

TABLE 4. Results for the larger random examples.

	`rand2048`			`rand4096`		
	LRRI	ICARE	SIGN	LRRI	ICARE	SIGN
Iteration steps	5	—	11	4	—	9
Runtime (s)	34.8321	2043.78	21.4784	139.670	16042.8	125.835
Rank Z_k	59	992	967	72	1984	1961
Final res.	2.450e−21	—	—	2.000e−14	—	—
Relative res.	4.997e−10	2.977e−10	1.628e−10	1.811e−10	1.804e−09	1.813e−10
Normalized res.	3.402e−11	2.027e−11	1.108e−11	2.812e−11	2.803e−10	2.816e−11
$\lVert Z_k^{\mathsf{T}} Z_k \rVert_2$	1.642e+02	1.642e+02	1.642e+02	6.760e+02	6.760e+02	6.760e+02

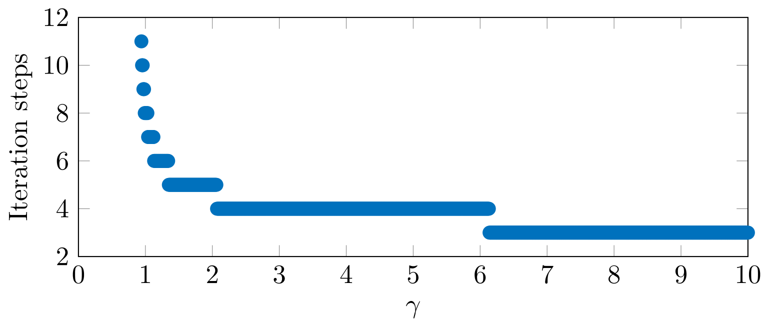
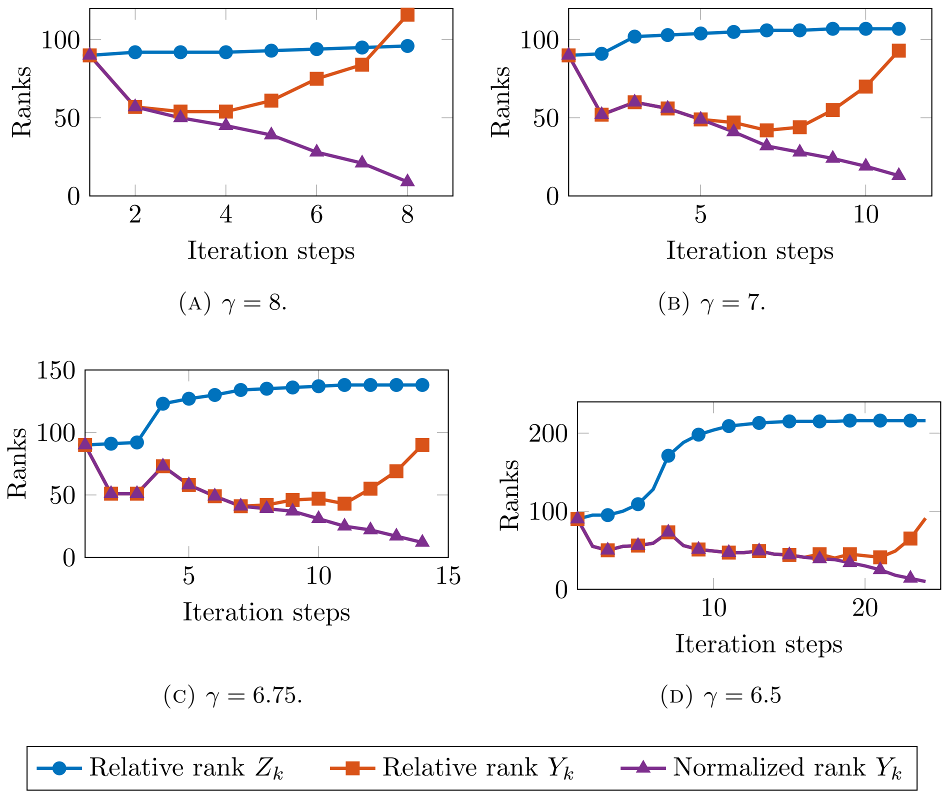


FIGURE 1. LRRI iteration steps for changing γ in the `AC10` example close to existence limit of semi-definite stabilizing solutions.

we can expect LRRI to outperform SIGN for larger problems. Also, these results reveal the strong dependence of LRRI on the inner Riccati equation solver and the power that comes from the ability to use sophisticated implementations for the inner Riccati equations with negative semi-definite quadratic terms.

Besides the general performance of the new LRRI approach, we have investigated its behavior for Riccati equations close to the existence limits of symmetric positive semi-definite stabilizing solutions. Therefore, we have varied the scaling constant γ; cf. Table 1. Figure 1 shows the iteration steps of LRRI in the `AC10` example over a range of different γ-values. The change in iteration steps depicted has also been observed for the other examples. Decreasing γ leads to an increase in the number of iteration steps since the scaled term $\frac{1}{\gamma}B_1$ becomes larger and the indefinite Riccati equation deviates further from the classical LQR/LQG equation, which is determined by B_2. The number of iteration steps tend to infinity for γ approaching its smallest possible value γ_0. For smaller values, the solution of the indefinite Riccati equation is not stabilizing anymore, or symmetric positive semi-definite, or both. This is in accordance with the convergence theory in Proposition 2.3 and Remark 2.4.

Similarly, we have obtained results for the ranks of the computed Riccati solution factors in the iteration steps of LRRI. Figure 2 shows the ranks of the outer and



(A) $\gamma = 8$.

(B) $\gamma = 7$.

(C) $\gamma = 6.75$.

(D) $\gamma = 6.5$

FIGURE 2. Rank development in LRRI in the `rand512` example for different γ.

inner solution factors in the `rand512` example for four different γ-values, where Z_k denotes the solution factor of the indefinite Riccati equation and Y_k of the semi-definite equation that is during the iteration step; cf. Algorithm 2 for the notation. Let $\sigma_{\max}(Z_k)$ and $\sigma_{\max}(Y_k)$ denote the largest singular values of the matrices Z_k and Y_k, respectively. The relative ranks of Z_k and Y_k are the numbers of singular values which are larger than $\sigma_{\max}(Z_k) \cdot \tau$ and $\sigma_{\max}(Y_k) \cdot \tau$ using the tolerance $\tau = 10^{-2} \cdot \sqrt{n\epsilon}$, with the double precision machine epsilon ϵ. The normalized rank of Y_k is given by the number of singular values of Y_k that are larger than $\sigma_{\max}(Z_1)\cdot\tau$. We can observe that for decreasing γ, the rank of Z_k increases and that the ranks of Y_k seem to lie relatively stable around 50 for an increasing number of iteration steps. The difference of the normalized rank from the relative towards the ends of the iterations indicates that it may be possible to solve these residual equations less accurate than the equations at the beginning of LRRI. However, an inexact version of the LRRI needs a careful analysis of the different errors and residuals in Algorithm 2 and is subject to future work.

3.2. **Low-rank approach for large-scale sparse examples.** Now we come to the case of large-scale sparse Riccati equations with indefinite quadratic terms. The LRRI method from Algorithm 2 is to our knowledge the only method suited to solve equations like (3) in the large-scale sparse setting. Therefore, we cannot compare

TABLE 5. Results of LRRI for the large-scale sparse examples.

	`rail`	`cylinderwake`
Iteration steps	3	3
Runtime (s)	72.7378	3469.59
Rank Z_k	169	418
Final res.	1.297e−19	2.184e−21
Relative res.	2.125e−21	1.996e−14
Normalized res.	9.766e−11	1.622e−03
$\Vert Z_k^{\mathsf{T}} Z_k \Vert_2$	6.866e+11	5.056e+08

our results to other methods. However, from the previous section we expect LRRI to yield reasonably accurate results in comparison to alternative Riccati equation solvers and the runtimes to be mainly dependent on the inner Riccati equation solver. We use the implementation of LRRI from [38] with the option to have RADI [7] or LR-Newton-ADI [12] as inner solvers or to switch between them during runtime if suitable.

As first example, we consider the optimal cooling problem of a steel profile; see, e.g., [37]; with the data available in [38]. For the different matrices in the quadratic term of (3), we consider the boundary control of the three lower segments of the profile edges as disturbances to give us B_1 and the rest to be control inputs in B_2. The resulting dimensions are given in Table 1. For LRRI, we use only the RADI method as inner Riccati equation solver and the results of the iteration can be seen in the `rail` column of Table 5. The iteration converges quickly and yields a very accurate solution of low rank. The behavior of the outer and inner iteration methods is shown in terms of normalized residuals in Figure 3. The inner solver has been used with the same convergence tolerance as LRRI.

As second example, we consider a laminar flow in a wake with a cylinder obstacle at Reynolds number 60, as described in [11] and with the data available at [20]. The example has two inputs modeling the suction and injection of fluid at the back of the cylinder placed in the beginning of the wake. We consider the case that one of the outlets is defect and produces only noise that needs to be compensated. Therefore, the defect outlet gives us the B_1 matrix and the control outlet gives us the matrix B_2. Also, the considered Riccati equation has exactly the structure as in (14) resulting from the underlying dynamical system of differential-algebraic equations. Therefore, we use the implicit truncation approach mentioned in Section 2.4.2, which is implemented in the `dae_2` function handles in [38]. Also, we note that the matrix pencil of this example has unstable eigenvalues. Since RADI is difficult to use for such a problem since it needs a stabilizing initial solution that produces a positive semi-definite residual in the Riccati operator, we switch only for the first iteration step of LRRI to the LR-Newton-ADI method and use a stabilizing Bernoulli feedback the same way as in [3]. The results of LRRI can be seen in the `cylinderwake` column of Table 5 and the normalized residuals of the outer and inner iterations are shown in Figure 4. We can observe that LRRI as well the inner iteration methods quickly converge. We want to note that the change from the Newton to the RADI solver after the first step of LRRI is in our favor, since RADI

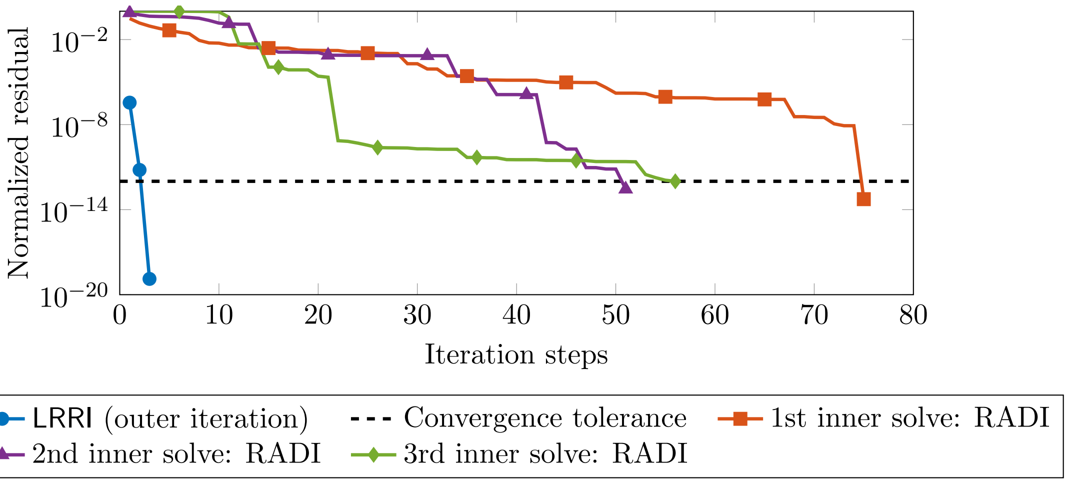


FIGURE 3. Convergence of Riccati iteration and inner solvers for the `rail` example.

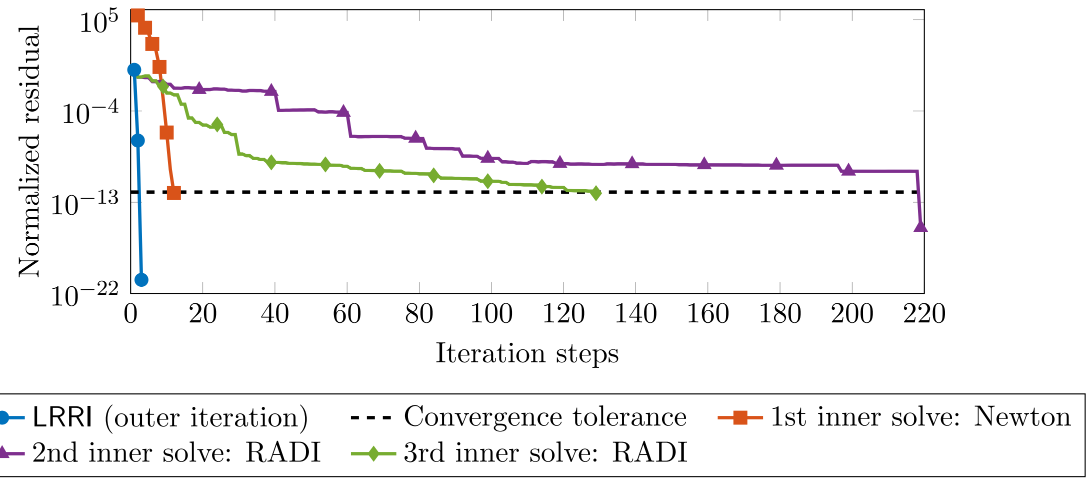


FIGURE 4. Convergence of Riccati iteration and inner solvers for the cylinder wake example.

performs significantly faster than Newton. The Newton method applies an iterative Lyapunov solver in each step such that a single Newton step is in this example by far more expensive than several hundred RADI steps. While the relative residual is comfortably small again for this example due to the very large norm of the stabilizing solution, the normalized residual is still quite high. This likely comes from the generally poor conditioning of the problem and the small norm of the right-hand side matrix. But overall, the computed solution has been obtained with reasonable accuracy and in a reasonable amount of time.

4. Conclusions

We have developed a low-rank iterative method for solving large-scale sparse Riccati equations with indefinite quadratic terms, which is based on solutions of Riccati equations with negative semi-definite quadratic terms. Numerical examples have illustrated that, in the dense case, we can expect a similar accuracy and good performance in comparison to other established Riccati equation solvers, and that for large-scale sparse equations, the method also yields reasonably good results. We have also extended the LR-RI approach to indefinite algebraic Riccati equations related to descriptor systems with singular E matrix.

To our knowledge, the low-rank Riccati iteration is currently the only approach to solve Riccati equations with indefinite quadratic terms in the large-scale sparse case. Another idea that might be directly extendable to this problem are projection-based methods. However, a problem occurring already in the case of classical Riccati equations with semi-definite quadratic terms is the solvability of the projected equations. Only recently, new results have been obtained under which conditions constructed projection spaces preserve the existence of stabilizing solutions in projected Riccati equations with negative semi-definite quadratic terms [46]. This problem becomes even more complicated when dealing with Riccati equations with indefinite quadratic terms, and so far, there are neither theoretical nor numerical results available considering this.

Declarations

Funding. This work was supported by the German Research Foundation (DFG) Research Training Group 2297 "Mathematical Complexity Reduction (MathCoRe)", Magdeburg.

Code and data availability. The source code of the implementations used to compute the presented results, the used data and computed results are available at https://doi.org/10.5281/zenodo.6308400 under the BSD-2-Clause license and authored by Steffen W. R. Werner.

Competing interests. Parts of this work were carried out while Werner was at the Max Planck Institute for Dynamics of Complex Technical Systems in Magdeburg, Germany. Benner is a member of the editorial board of Numerical Algorithms.

The authors declare to have no competing interests related to this work.

References

[1] Anderson, B.D.O., Vongpanitlerd, B.: Network Analysis and Synthesis: A Modern Systems Approach. Networks Series. Prentice-Hall, Englewood Cliffs, NJ (1972)

[2] Arnold, W.F., Laub, A.J.: Generalized eigenproblem algorithms and software for algebraic Riccati equations. Proc. IEEE **72**(12), 1746–1754 (1984). https://doi.org/10.1109/PROC.1984.13083

[3] Bänsch, E., Benner, P., Saak, J., Weichelt, H.K.: Riccati-based boundary feedback stabilization of incompressible Navier-Stokes flows. SIAM J. Sci. Comput. **37**(2), A832–A858 (2015). https://doi.org/10.1137/140980016

[4] Başar, T., Moon, J.: Riccati equations in Nash and Stackelberg differential and dynamic games. IFAC-Pap. **50**(1), 9547–9554 (2017). https://doi.org/10.1016/j.ifacol.2017.08.1625. 20th IFAC World Congress

[5] Benner, P.: Partial stabilization of descriptor systems using spectral projectors. In: P. Van Dooren, S.P. Bhattacharyya, R.H. Chan, V. Olshevsky, A. Routray (eds.) Numerical Linear Algebra in Signals, Systems and Control, *Lect. Notes Electr. Eng.*, vol. 80, pp. 55–76. Springer, Dodrecht (2011). https://doi.org/10.1007/978-94-007-0602-6_3

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

[7] Benner, P., Bujanović, Z., Kürschner, P., Saak, J.: RADI: a low-rank ADI-type algorithm for large scale algebraic Riccati equations. Numer. Math. **138**(2), 301–330 (2018). https://doi.org/10.1007/s00211-017-0907-5

[8] Benner, P., Bujanović, Z., Kürschner, P., Saak, J.: A numerical comparison of different solvers for large-scale, continuous-time algebraic Riccati equations and LQR problems. SIAM J. Sci. Comput. **42**(2), A957–A996 (2020). https://doi.org/10.1137/18M1220960

[9] Benner, P., Ezzatti, P., Quintana-Ortí, E.S., Remón, A.: A factored variant of the Newton iteration for the solution of algebraic Riccati equations via the matrix sign function. Numer. Algorithms **66**(2), 363–377 (2014). https://doi.org/10.1007/s11075-013-9739-2

[10] Benner, P., Heiland, J.: Equivalence of Riccati-based robust controller design for index-1 descriptor systems and standard plants with feedthrough. In: 2020 European Control Conference (ECC), pp. 402–407 (2020). https://doi.org/10.23919/ECC51009.2020.9143771

[11] Benner, P., Heiland, J., Werner, S.W.R.: Robust output-feedback stabilization for incompressible flows using low-dimensional $\mathcal{H}_\infty$-controllers. Comput. Optim. Appl. (2022). https://doi.org/10.1007/s10589-022-00359-x

[12] Benner, P., Li, J.R., Penzl, T.: Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems. Numer. Lin. Alg. Appl. **15**(9), 755–777 (2008). https://doi.org/10.1002/nla.622

[13] Benner, P., Saak, J.: Numerical solution of large and sparse continuous time algebraic matrix Riccati and lyapunov equations: a state of the art survey. GAMM-Mitt. **36**(1), 32–52 (2013). https://doi.org/10.1002/gamm.201310003

[14] Benner, P., Stykel, T.: Numerical solution of projected algebraic Riccati equations. SIAM J. Numer. Anal. **52**(2), 581–600 (2014). https://doi.org/10.1137/130923993

[15] Benner, P., Werner, S.W.R.: Model reduction of descriptor systems with the MORLAB toolbox. IFAC-Pap. **51**(2), 547–552 (2018). https://doi.org/10.1016/j.ifacol.2018.03.092. 9th Vienna International Conference on Mathematical Modelling MATHMOD 2018

[16] Benner, P., Werner, S.W.R.: MORLAB – Model Order Reduction LABoratory (version 5.0) (2019). https://doi.org/10.5281/zenodo.3332716. See also: `https://www.mpi-magdeburg.mpg.de/projects/morlab`

[17] Delfour, M.C.: Linear quadratic differential games: Saddle point and Riccati differential equation. SIAM J. Control Optim. **46**(2), 750–774 (2007). https://doi.org/10.1137/050639089

[18] Freitas, F., Rommes, J., Martins, N.: Gramian-based reduction method applied to large sparse power system descriptor models. IEEE Trans. Power Syst. **23**(3), 1258–1270 (2008). https://doi.org/10.1109/TPWRS.2008.926693

[19] Golub, G.H., Van Loan, C.F.: Matrix Computations, fourth edn. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore (2013)

[20] Heiland, J., Werner, S.W.R.: Code, data and results for numerical experiments in “Robust output-feedback stabilization for incompressible flows using low-dimensional $\mathcal{H}_\infty$-controllers” (version 2.0) (2021). https://doi.org/10.5281/zenodo.5532539

[21] Heinkenschloss, M., Sorensen, D.C., Sun, K.: Balanced truncation model reduction for a class of descriptor systems with application to the Oseen equations. SIAM J. Sci. Comput. **30**(2), 1038–1063 (2008). https://doi.org/10.1137/070681910

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

[23] Jonckheere, E.A., Silverman, L.M.: A new set of invariants for linear systems–application to reduced order compensator design. IEEE Trans. Autom. Control **28**(10), 953–964 (1983). https://doi.org/10.1109/TAC.1983.1103159

[24] Kleinman, D.L.: On an iterative technique for Riccati equation computations. IEEE Trans. Autom. Control **13**(1), 114–115 (1968). https://doi.org/10.1109/TAC.1968.1098829

[25] Kürschner, P.: Efficient low-rank solution of large-scale matrix equations. Dissertation, Otto-von-Guericke-Universität, Magdeburg, Germany (2016). URL `http://hdl.handle.net/11858/00-001M-0000-0029-CE18-2`

[26] Lanzon, A., Feng, Y., Anderson, B.D.O.: An iterative algorithm to solve algebraic Riccati equations with an indefinite quadratic term. In: 2007 European Control Conference (ECC), pp. 3033–3039 (2007). https://doi.org/10.23919/ecc.2007.7068239

[27] Lanzon, A., Feng, Y., Anderson, B.D.O., Rotkowitz, M.: Computing the positive stabilizing solution to algebraic Riccati equations with an indefinite quadratic term via a recursive method. IEEE Trans. Autom. Control **53**(10), 2280–2291 (2008). https://doi.org/10.1109/TAC.2008.2006108

[28] Laub, A.J.: A Schur method for solving algebraic Riccati equations. IEEE Trans. Autom. Control **24**(6), 913–921 (1979). https://doi.org/10.1109/TAC.1979.1102178

[29] Leibfritz, F.: $COMPl_eib$: COnstrained Matrix-optimization Problem library – a collection of test examples for nonlinear semidefinite programs, control system design and related problems. Tech.-report, University of Trier (2004). URL `http://www.friedemann-leibfritz.de/COMPlib_Data/COMPlib_Main_Paper.pdf`

[30] Li, J.R., White, J.: Low rank solution of Lyapunov equations. SIAM J. Matrix Anal. Appl. **24**(1), 260–280 (2002). https://doi.org/10.1137/S0895479801384937

[31] Lin, Y., Simoncini, V.: A new subspace iteration method for the algebraic Riccati equation. Numer. Linear Algebra Appl. **22**(1), 26–47 (2015). https://doi.org/10.1002/nla.1936

[32] Locatelli, A.: Optimal Control: An Introduction. Birkhäuser, Basel (2001)

[33] McFarlane, D.C., Glover, K.: Robust Controller Design Using Normalized Coprime Factor Plant Descriptions, *Lect. Notes Control Inf. Sci.*, vol. 138. Springer, Berlin, Heidelberg (1990). https://doi.org/10.1007/BFB0043199

[34] Möckel, J., Reis, T., Stykel, T.: Linear-quadratic Gaussian balancing for model reduction of differential-algebraic systems. Internat. J. Control **84**(10), 1627–1643 (2011). https://doi.org/10.1080/00207179.2011.622791

[35] Opdenacker, P.C., Jonckheere, E.A.: A contraction mapping preserving balanced reduction scheme and its infinity norm error bounds. IEEE Trans. Circuits Syst. **35**(2), 184–189 (1988). https://doi.org/10.1109/31.1720

[36] Roberts, J.D.: Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. Internat. J. Control **32**(4), 677–687 (1980). https://doi.org/10.1080/00207178008922881. Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971

[37] Saak, J.: Efficient numerical solution of large scale algebraic matrix equations in PDE control and model order reduction. Dissertation, Technische Universität Chemnitz, Germany (2009). URL `https://nbn-resolving.org/urn:nbn:de:bsz:ch1-200901642`

[38] Saak, J., Köhler, M., Benner, P.: M-M.E.S.S. – The Matrix Equations Sparse Solvers library (version 2.1) (2021). https://doi.org/10.5281/zenodo.4719688. See also: `https://www.mpi-magdeburg.mpg.de/projects/mess`

[39] Saak, J., Voigt, M.: Model reduction of constrained mechanical systems in M-M.E.S.S. IFAC-Pap. **51**(2), 661–666 (2018). https://doi.org/10.1016/j.ifacol.2018.03.112. 9th Vienna International Conference on Mathematical Modelling MATHMOD 2018

[40] Sandell, N.: On Newton's method for Riccati equation solution. IEEE Trans. Autom. Control **19**(3), 254–255 (1974). https://doi.org/10.1109/TAC.1974.1100536

[41] Simoncini, V.: Analysis of the rational Krylov subspace projection method for large-scale algebraic Riccati equations. SIAM J. Matrix Anal. Appl. **37**(4), 1655–1674 (2016). https://doi.org/10.1137/16M1059382

[42] Stillfjord, T.: Singular value decay of operator-valued differential Lyapunov and Riccati equations. SIAM J. Control Optim. **56**(5), 3598–3618 (2018). https://doi.org/10.1137/18M1178815

[43] Stykel, T.: Low-rank iterative methods for projected generalized Lyapunov equations. Electron. Trans. Numer. Anal. **30**, 187–202 (2008). URL `https://etna.math.kent.edu/volumes/2001-2010/vol30/abstract.php?vol=30&pages=187-202`

[44] Varga, A.: On computing high accuracy solutions of a class of Riccati equations. Control–Theory and Advanced Technology **10**(4), 2005–2016 (1995)

[45] Weichelt, H.K.: Numerical aspects of flow stabilization by Riccati feedback. Dissertation, Otto-von-Guericke-Universität, Magdeburg, Germany (2016). https://doi.org/10.25673/4493

[46] Zhang, L., Fan, H.Y., Chu, E.K.: Inheritance properties of Krylov subspace methods for continuous-time algebraic Riccati equations. J. Comput. Appl. Math. **371**, 112685 (2020). https://doi.org/10.1016/j.cam.2019.112685

[47] Zhou, K., Doyle, J.C., Glover, K.: Robust and Optimal Control. Prentice-Hall, Upper Saddle River, NJ (1996)