

# Efficient Solution of Large-Scale Algebraic Riccati Equations Associated with Index-2 DAEs via the Inexact Low-Rank Newton-ADI Method

Peter Benner<sup>a,d</sup>, Matthias Heinkenschloss<sup>b,1,\*</sup>, Jens Saak<sup>a</sup>, Heiko K. Weichelt<sup>c,2</sup>

<sup>a</sup>*Research Group Computational Methods in Systems and Control Theory (CSC),  
Max Planck Institute for Dynamics of Complex Technical Systems Magdeburg,  
Sandtorstr. 1, 39106 Magdeburg, Germany*

<sup>b</sup>*Department of Computational and Applied Mathematics (CAAM),  
Rice University, MS-134, 6100 Main Street, Houston, TX 77005-1892, USA*

<sup>c</sup>*The Mathworks Ltd., Matrix House, Cambridge Business Park,  
CB4 0HH Cambridge, United Kingdom*

<sup>d</sup>*Institut für Analysis und Numerik, Fakultät für Mathematik,  
Otto-von-Guericke Universität Magdeburg,  
Universitätsplatz 2, 39106 Magdeburg, Germany.*

---

## Abstract

This paper extends the algorithm of *Benner, Heinkenschloss, Saak, and Weichelt: An inexact low-rank Newton-ADI method for large-scale algebraic Riccati equations, Applied Numerical Mathematics Vol. 108 (2016), pp. 125–142, doi:10.1016/j.apnum.2016.05.006* to Riccati equations associated with Hessenberg index-2 Differential Algebraic Equation (DAE) systems. Such DAE systems arise, e.g., from semi-discretized, linearized (around steady state) Navier-Stokes equations. The solution of the associated Riccati equation is important, e.g., to compute feedback laws that stabilize the Navier-Stokes equations. Challenges in the numerical solution of the Riccati equation arise from the large-scale of the underlying systems, the algebraic constraint in the DAE system, and the fact that matrices arising in some subproblems may only be marginally stable. These challenges are met by a careful extension of the inexact low-rank Newton-ADI method to the case of DAE systems. A main ingredient in the extension to the DAE case is the projection onto the manifold of the algebraic constraints. In the algorithm, the equations are never explicitly projected, but the projection is only applied as needed. The performance of the algorithm is illustrated on a large-scale Riccati equation associated with the stabilization of Navier-Stokes flow around a cylinder.

*Keywords:* Riccati equation, Kleinman-Newton, Stokes, Navier-Stokes, low-rank ADI methods  
*2010 MSC:* 49M15, 49N35, 65F30, 65H10, 76D55, 93B52

---

## 1. Introduction

This paper introduces and analyzes an efficient algorithm for the solution of the generalized continuous algebraic Riccati equation (GCARE) associated with the solution of linear quadratic regulator (LQR) problems governed by Hessenberg index-2 Differential Algebraic Equations (DAEs). This problem arises, e.g., in the computation of feedback laws that stabilize Navier-Stokes flows. The numerical solution of the Riccati equation is challenging because the underlying systems are large-scale, because of the presence of algebraic

---

\*Corresponding author

*Email addresses:* [benner@mpi-magdeburg.mpg.de](mailto:benner@mpi-magdeburg.mpg.de) (Peter Benner), [heinken@rice.edu](mailto:heinken@rice.edu) (Matthias Heinkenschloss), [saak@mpi-magdeburg.mpg.de](mailto:saak@mpi-magdeburg.mpg.de) (Jens Saak), [heiko.weichelt@mathworks.co.uk](mailto:heiko.weichelt@mathworks.co.uk) (Heiko K. Weichelt)

<sup>1</sup>The research of this author was supported in part by NSF grant DMS-1522798 and by the DARPA EQUiPS Program, Award UTA15-001068.

<sup>2</sup>This paper is based on the PhD Thesis of this author, which was completed while he was with the Max Planck Institute for Dynamics of Complex Technical Systems Magdeburg, Germany.

constraints in the DAE system, and because some subproblems may only be marginally stable. To overcome these challenges we extend our inexact low-rank Newton-ADI method developed in [1] for problems governed by ordinary differential equations (ODEs) to the DAE case. The main idea is to use the structure of the Hessenberg index-2 DAE and apply the discrete version of the Leray projector (see Heinkenschloss et al. [2] and Bänsch, et al. [3]) to transform the LQR problem governed by the DAE into a classical LQR problem governed by an ODE. In principle, the standard LQR and Riccati theory as well as the inexact low-rank Newton-ADI method developed in our previous paper [1] can be applied to this ODE problem. This, however leads to a solution approach that is not practical because the projected systems are large-scale and, because of the projection, dense. To arrive at an efficient algorithm, the computations must be presented in terms of the original large-scale sparse system, and the structure of the governing DAE system must be exploited. This is done in this paper.

The LQR problem and associated Riccati equation considered in this paper have also been solved by Bänsch et al. [3]. However, the focus of [3] was the computation of feedback laws for Navier-Stokes flows, and a basic version of an inexact low-rank Newton-ADI method was applied. Our paper focusses on the solution of the Riccati equation and incorporates many recent improvements. As a result, the algorithm in this paper delivers an approximately 100-times speed-up over the algorithm used in [3]. Benner and Stykel [4] study the solution of projected Riccati equations, which are associated with DAEs. They use so-called spectral projectors, which project onto the right and left deflating subspaces. While these projectors can be applied to general DAEs defined by a regular pencil, in the general case “the projectors [...] are required in explicit form [and the] computation of these projectors is, in general, very expensive” [4, p. 590]. The projector used in our paper is specially designed for the index-2 DAE system arising for fluid flow problems and our Kleinman-Newton-ADI method contains many improvements not yet available in [4]. In principle it is possible to use rational Krylov subspace projection methods (see Simoncini et al. [5, 6]) to solve the Riccati equations, but extensions of this approach to the DAE case and numerical comparisons of the latest versions of both approaches are not yet available.

This paper is organized as follows. The next section, Section 2, introduces the LQR problem, uses projection onto the constraint manifold to derive a projected Riccati equation, and reviews existence results for both the projected Riccati equation and the LQR problem. Section 3 reviews the main components of our algorithm in [1] applied to the projected GCARE and Section 4 carefully exploits the special structure of the projected GCARE for an efficient numerical realization of the inexact low-rank Newton-ADI method. Finally, Section 5 illustrates the performance of our algorithm on a large-scale Riccati equation associated with the stabilization of Navier-Stokes flow around a cylinder – a problem also solved by Bänsch et al. [3]. As mentioned earlier, the algorithmic improvements in this paper lead to approximately 90-times speed-up over the algorithm used in [3].

**Notation.** Throughout the paper we consider the Hilbert space of matrices in  $\mathbb{R}^{n \times n}$  endowed with the inner product  $\langle M, N \rangle = \text{tr}(M^T N) = \sum_{i,j=1}^n M_{ij} N_{ij}$  and the corresponding (Frobenius) norm  $\|M\|_F = (\langle M, M \rangle)^{1/2} = (\sum_{i,j=1}^n M_{ij}^2)^{1/2}$ . Furthermore, given real symmetric matrices  $M, N$ , we write  $M \succeq N$  if and only if  $M - N$  is positive semi-definite, and  $M \succ N$  if and only if  $M - N$  is positive definite. The spectrum of a symmetric matrix  $M$  is denoted by  $\sigma(M)$ .

## 2. The LQR Problem and the Riccati Equation

In this section we present the mathematical statement of the LQR problem and the governing Hessenberg index-2 DAE, and we show how it can be transformed into a ‘standard’ LQR problem governed by an ODE using a projection onto the constraint manifold of the original DAE. Then we apply classical LQR theory to this transformed problem to compute, under standard conditions on the system, the solution of the LQR problem via the GCARE. As mentioned before, the problem transformation is performed to derive the solution, but the computations are done using the original DAE framework. The projection used to convert the DAE into an ODE was first used in a different context by Heinkenschloss et al. [2]. For DAEs derived from a finite element discretization of the Stokes or linearized Navier-Stokes system, Bänsch et al. [3] show that this projection is a discrete version of the Leray projector. Projections have also been used by Benner,

Stykel [4] to formulate and solve GCAREs associated with index-2 DAEs, although, as already noted in the introduction, the projection there is different. Except for some extensions in problem statement and notation the material in this section is mostly known from [2, 3], but is needed to provide the necessary background that allows us to switch between expressions using the original DAE system and the the corresponding expressions using the transformed ODE system. Compared to [3], this section also provides a more detailed link between the representations of the optimal control of the LQR problem derived using the original DAE and transformed ODE system.

### 2.1. The LQR Problem

Given matrices  $A, M \in \mathbb{R}^{n_v \times n_v}$ ,  $G \in \mathbb{R}^{n_v \times n_p}$ ,  $B \in \mathbb{R}^{n_v \times n_u}$ , and  $C \in \mathbb{R}^{n_y \times n_v}$  such that  $M$  is symmetric positive definite and  $G$  has rank  $n_p < n_v$ , we consider the LQR problem

$$\min_{\mathbf{u} \in L^2(0, \infty)} \int_0^\infty \|\mathbf{y}(t)\|_2^2 + \|\mathbf{u}(t)\|_2^2 dt, \quad (2.1)$$

where for given  $\mathbf{u} \in L^2(0, \infty)$ , the function  $\mathbf{y} \in L^2(0, \infty)$  is obtained as the output of the Hessenberg index-2 Differential Algebraic Equation system

$$M \frac{d}{dt} \mathbf{v}(t) = A\mathbf{v}(t) + G\mathbf{p}(t) + B\mathbf{u}(t), \quad (2.2a)$$

$$0 = G^T \mathbf{v}(t), \quad (2.2b)$$

$$\mathbf{y}(t) = C\mathbf{v}(t). \quad (2.2c)$$

To ensure well-posedness of the LQR, we will make additional assumptions on the system (2.2) in Section 2.2. In the cost functional, we may replace the Euclidian norms by any weighted norm induced by positive definite matrices  $Q_y$  and  $Q_u$ . Here, we set both weighting matrices to the appropriate identity for ease of notation. It is straightforward to include non-identity weighting matrices into the problem description and the computational framework.

The LQR problem (2.1, 2.2) arises, e.g., in feedback stabilization of the Navier-Stokes equations, see Bäscher et al. [3] or Raymond [9]. In this context, (2.2a, 2.2b) correspond to the linearized discretized Navier-Stokes equations, and  $\mathbf{v}$ ,  $\mathbf{p}$  correspond to velocity and pressure, respectively. The problem also arises in feedback stabilization of multi-field flow problems, see Bäscher et al. [10]. In this case, (2.2a) includes additional equations such as linearized reaction equations, and  $\mathbf{v}$  corresponds to velocities and the other fields, such as concentrations.

If we define

$$\mathbf{A} = \begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}, \quad \mathbf{C} = [C \quad 0], \quad (2.3)$$

and

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{v}(t) \\ \mathbf{p}(t) \end{bmatrix},$$

the DAE system (2.2) can be written in the compact form

$$\mathbf{M} \frac{d}{dt} \mathbf{x}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad (2.4a)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t). \quad (2.4b)$$

The structure of (2.2) can be used to convert the LQR problem (2.1, 2.2) into a classical one governed by an ODE. We proceed as in [2, 3]. The constraint (2.2b) and the variable  $\mathbf{p}$  can be eliminated from (2.2a, 2.2b) via the projection

$$\Pi = I_{n_v} - G(G^T M^{-1} G)^{-1} G^T M^{-1} \in \mathbb{R}^{n_v \times n_v}. \quad (2.5)$$

The matrix  $\Pi$  obeys  $\Pi^2 = \Pi$  and  $\Pi M = M\Pi^T$ , i.e., it is in fact an  $M$ -orthogonal projection. Furthermore,

$$\text{null}(\Pi^T) = \text{range}(M^{-1}G) \quad \text{and} \quad \text{range}(\Pi^T) = \text{null}(G^T), \quad (2.6)$$

which means that

$$0 = G^T \mathbf{v}(t) \quad \text{if and only if} \quad \mathbf{v}(t) = \Pi^T \mathbf{v}(t).$$

We use the latter property to enforce (2.2b) and multiply (2.2a) by  $\Pi$  to arrive at

$$\Pi M \Pi^T \frac{d}{dt} \mathbf{v}(t) = \Pi A \Pi^T \mathbf{v}(t) + \Pi B \mathbf{u}(t), \quad (2.7a)$$

$$\mathbf{y}(t) = C \Pi^T \mathbf{v}(t). \quad (2.7b)$$

If needed, the function  $\mathbf{p}$  can be computed from  $\mathbf{v}$ ,  $\mathbf{u}$  using

$$\mathbf{p}(t) = -(G^T M^{-1} G)^{-1} G^T M^{-1} A \mathbf{v}(t) - (G^T M^{-1} G)^{-1} G^T M^{-1} B \mathbf{u}(t). \quad (2.8)$$

Equation (2.8) is obtained multiplying (2.2a) by  $G^T M^{-1}$  and using (2.2b).

Since  $\Pi M \Pi^T \in \mathbb{R}^{n_v \times n_v}$  has an  $n_p$ -dimensional null-space and cannot be inverted, (2.7) is still not an ODE. However,  $\Pi^T \mathbf{v}(t) \in \mathbb{R}^{n_v}$  is contained in the  $n_v - n_p$  dimensional subspace  $\text{range}(\Pi^T)$  and we can explicitly express  $\Pi^T \mathbf{v}(t)$  as an element of this subspace. This is done using the decomposition

$$\Pi = \Theta_l \Theta_r^T \quad \text{such that} \quad \Theta_l^T \Theta_r = I_{n_v - n_p} \quad (2.9)$$

with  $\Theta_l, \Theta_r \in \mathbb{R}^{n_v \times (n_v - n_p)}$ . In particular

$$\text{range}(\Theta_r) = \text{range}(\Pi^T). \quad (2.10)$$

The new variable  $\tilde{\mathbf{v}}(t) = \Theta_l^T \mathbf{v}(t) \in \mathbb{R}^{n_v - n_p}$  satisfies

$$\Theta_r \tilde{\mathbf{v}}(t) = \Theta_r \Theta_l^T \mathbf{v}(t) = \Pi^T \mathbf{v}(t) = \mathbf{v}(t). \quad (2.11)$$

Using the decomposition (2.9), we define

$$\mathcal{M} := \Theta_r^T M \Theta_r, \quad \mathcal{A} := \Theta_r^T A \Theta_r, \quad \mathcal{B} := \Theta_r^T B, \quad \mathcal{C} := C \Theta_r, \quad (2.12)$$

and write the descriptor system (2.7) as

$$\mathcal{M} \frac{d}{dt} \tilde{\mathbf{v}}(t) = \mathcal{A} \tilde{\mathbf{v}}(t) + \mathcal{B} \mathbf{u}(t), \quad (2.13a)$$

$$\mathbf{y}(t) = \mathcal{C} \tilde{\mathbf{v}}(t). \quad (2.13b)$$

The DAE system (2.2) is equivalent to the ODE system (2.13). Furthermore, the LQR problem (2.1, 2.2) is equivalent to the classical LQR problem (2.1, 2.13). We summarize this result in the following proposition.

**Proposition 1.** *The functions  $\mathbf{v}$ ,  $\mathbf{p}$  solve (2.2a, 2.2b) if and only if  $\mathbf{v} = \Theta_r \tilde{\mathbf{v}}$ ,  $\tilde{\mathbf{v}}$  solves (2.13a), and (2.8) holds. Moreover, the control  $\mathbf{u}_* \in L^2(0, \infty)$  solves the LQR problem (2.1, 2.2) if and only if it solves the classical LQR problem (2.1, 2.13)*

The equivalence between the LQR problem (2.1, 2.2) and the classical LQR problem (2.1, 2.13), however, is only used theoretically. Even if the matrices  $A, \dots$  in (2.2) are sparse, the projected matrices  $\mathcal{A}, \dots$  in (2.13) are dense. We will use the equivalence between (2.1, 2.2) and (2.1, 2.13) to derive our algorithms, but always compute using the formulation (2.1, 2.2).

## 2.2. Solution of the LQR Problem and the Riccati Equation

If  $(\mathcal{A}, \mathcal{B}; \mathcal{M})$  is stabilizable and  $(\mathcal{C}, \mathcal{A}; \mathcal{M})$  is detectable, the classical LQR problem (2.1, 2.13) has a solution given as the feedback control law

$$\mathbf{u}_*(t) = - \underbrace{\mathcal{B}^T \mathcal{X}^{(*)} \mathcal{M}}_{=: \mathcal{K}^T} \tilde{\mathbf{v}}(t), \quad (2.14)$$

where  $\mathcal{X}^{(*)} = (\mathcal{X}^{(*)})^T \succeq 0 \in \mathbb{R}^{(n_v - n_p) \times (n_v - n_p)}$  is the unique stabilizing solution of the GCARE

$$\mathcal{C}^T \mathcal{C} + \mathcal{A}^T \mathcal{X} \mathcal{M} + \mathcal{M} \mathcal{X} \mathcal{A} - \mathcal{M} \mathcal{X} \mathcal{B} \mathcal{B}^T \mathcal{X} \mathcal{M} = 0. \quad (2.15)$$

See, e.g., Lancaster, Rodman [11].

The unique stabilizing solution of the GCARE is obtained by applying Newton's method to find a root of the quadratic operator

$$\mathcal{R}(\mathcal{X}) = \mathcal{C}^T \mathcal{C} + \mathcal{A}^T \mathcal{X} \mathcal{M} + \mathcal{M} \mathcal{X} \mathcal{A} - \mathcal{M} \mathcal{X} \mathcal{B} \mathcal{B}^T \mathcal{X} \mathcal{M}. \quad (2.16)$$

Given an approximate root  $\mathcal{X}^{(k)}$ , the new approximation is computed as the solution of

$$\mathcal{R}'(\mathcal{X}^{(k)}) \mathcal{X}^{(k+1)} = \mathcal{R}'(\mathcal{X}^{(k)}) \mathcal{X}^{(k)} - \mathcal{R}(\mathcal{X}^{(k)}). \quad (2.17)$$

This method is known as the Kleinman-Newton method. See the original paper by Kleinman [12] or the book by Lancaster, Rodman [11].

The system (2.17) is a Lyapunov equation and for large-scale problems the exact Kleinman-Newton method which is defined by (2.17) is impractical. This is particularly true for the Riccati equation (2.15) which is obtained from a large-scale DAE by projection. The projected matrices in (2.12) are not only large-scale, but because of the projections they are also dense. To overcome these difficulties, we need to 'undo' the projections in the numerical computations. We will discuss the details of our solution approach in the next section. In the remainder of this section we provide basic relationships between quantities for the projected problem and quantities for the original problem.

The Kleinman-Newton method applied to the projected Riccati equation (2.15) generates iterates

$$0 \preceq \mathcal{X}^{(k)} \in \mathbb{R}^{(n_v - n_p) \times (n_v - n_p)}$$

and corresponding feedback matrices

$$(\mathcal{K}^{(k)})^T = \mathcal{B}^T \mathcal{X}^{(k)} \mathcal{M} \in \mathbb{R}^{n_u \times (n_v - n_p)}. \quad (2.18)$$

We want to write the corresponding feedback law  $-(\mathcal{K}^{(k)})^T \tilde{\mathbf{v}}(t) = -\mathcal{B}^T \mathcal{X}^{(k)} \mathcal{M} \tilde{\mathbf{v}}(t)$  in terms of the original variable  $\mathbf{v} = \Theta_r \tilde{\mathbf{v}}$ , see Proposition 1. If we define

$$X^{(k)} = \Theta_r \mathcal{X}^{(k)} \Theta_r^T \in \mathbb{R}^{n_v \times n_v} \quad (2.19a)$$

and

$$(K^{(k)})^T = B^T X^{(k)} M \in \mathbb{R}^{n_u \times n_v}, \quad (2.19b)$$

then (2.12) and  $\mathbf{v} = \Theta_r \tilde{\mathbf{v}}$  imply

$$(K^{(k)})^T \Theta_r = (\mathcal{K}^{(k)})^T, \quad (2.19c)$$

and

$$-(\mathcal{K}^{(k)})^T \tilde{\mathbf{v}}(t) = -\mathcal{B}^T \mathcal{X}^{(k)} \mathcal{M} \tilde{\mathbf{v}}(t) = -B^T X^{(k)} M \mathbf{v}(t) = -(K^{(k)})^T \mathbf{v}(t). \quad (2.19d)$$

The convergence of the (exact) Kleinman-Newton method can now be expressed in the unprojected variables and in the context of the (2.2). First we show that the stability (detectability) of the system (2.13) is equivalent to the stability (detectability) of the system (2.2).

**Definition 2.**

1. A matrix pencil  $(\mathbf{A}, \mathbf{M})$  is called stable if it is regular and all the finite eigenvalues of  $(\mathbf{A}, \mathbf{M})$  lie in the open left half-plane.
2. Let  $\mathbf{A}, \mathbf{B}, \mathbf{M}$  be given by (2.12). The triple  $(\mathbf{A}, \mathbf{B}; \mathbf{M})$  is stabilizable if there exists a matrix  $K \in \mathbb{R}^{n_v \times n_u}$  such that all finite eigenvalues of the matrix pencil

$$\left( \begin{bmatrix} A - BK^T & G \\ G^T & 0 \end{bmatrix}, \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \right) \quad (2.20)$$

are contained in the open left half-plane. The triple  $(\mathbf{C}, \mathbf{A}; \mathbf{M})$  is called detectable if and only if  $(\mathbf{A}^T, \mathbf{C}^T; \mathbf{M})$  is stabilizable.

The following result is proven in [7, Lemma 4.4].

**Lemma 3.** The matrix triple  $(\mathcal{A}, \mathcal{B}; \mathcal{M})$  is stabilizable ( $(\mathcal{C}, \mathcal{A}; \mathcal{M})$  is detectable) if and only if  $(\mathbf{A}, \mathbf{B}; \mathbf{M})$  is stabilizable ( $(\mathbf{C}, \mathbf{A}; \mathbf{M})$  is detectable).

With these preparations, the following result is an immediate consequence of the classical Kleinman-Newton convergence result [12], [11]. See [7, Thm. 4.5] for a detailed proof.

**Theorem 4.** Assume  $(\mathbf{A}, \mathbf{B}; \mathbf{M})$  is stabilizable and  $(\mathbf{C}, \mathbf{A}; \mathbf{M})$  is detectable. There exists a maximal symmetric solution  $\mathcal{X}^{(*)} \in \mathbb{R}^{(n_v - n_p) \times (n_v - n_p)}$  of  $\mathcal{R}(\mathcal{X}) = 0$  for which

$$\left( \begin{bmatrix} A - BB^T X^{(*)} M & G \\ G^T & 0 \end{bmatrix}, \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \right) \quad (2.21)$$

is stable, where  $X^{(*)} = \Theta_r \mathcal{X}^{(*)} \Theta_r^T$ . Furthermore, let  $X^{(0)} = \Theta_r \mathcal{X}^{(0)} \Theta_r^T$  be symmetric and such that

$$\left( \begin{bmatrix} A - BB^T X^{(0)} M & G \\ G^T & 0 \end{bmatrix}, \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \right)$$

is stable, then the sequence  $\{X^{(k)}\}_{k=0}^{\infty}$  defined by  $X^{(k)} := \Theta_r \mathcal{X}^{(k)} \Theta_r^T$ , (2.17) satisfies

$$X^{(1)} \succeq X^{(2)} \succeq \dots \succeq X^{(k)} \succeq 0, \\ \lim_{k \rightarrow \infty} X^{(k)} = X^{(*)},$$

and there is a constant  $\kappa$  such that

$$\|X^{(k+1)} - X^{(*)}\|_F \leq \kappa \|X^{(k)} - X^{(*)}\|_F^2 \quad \text{for all } k.$$

**Remark 5.** If  $X^{(*)} = \Theta_r \mathcal{X}^{(*)} \Theta_r^T$  is the solution of the Riccati equation specified in Theorem 4 and

$$(K^{(*)})^T = B^T X^{(*)} M$$

is the corresponding feedback matrix, then  $\mathbf{u}_*(t) := -(K^{(*)})^T \mathbf{v}(t)$  solves the LQR problem (2.1, 2.2).

In principle, the large-scale projected GCARE (2.15) can be solved using the Kleinman-Newton method [12]. However, the size and special structure of (2.15) require the inexact solution of the Newton equation, a Lyapunov equation, in each step of the Kleinman-Newton method. Moreover, the explicit use of the large, dense projected matrices  $\mathcal{M}, \mathcal{A}, \mathcal{B}, \mathcal{C}$  (2.12) must be avoided in computations and the final algorithm must operate with the sparse matrices  $\mathbf{M}, \mathbf{A}, \mathbf{B}, \mathbf{C}$  (2.3) instead. To adopt our approach from [1] to efficiently solve the large-scale projected GCARE (2.15), we first need to review the main components of our approach in [1].

### 3. Inexact Kleinman-Newton for Algebraic Riccati Equations

Our approach in [1] is based on an inexact Kleinman-Newton method with line search. Although the exact and, under additional conditions, inexact Kleinman-Newton method converges with step size fixed to one (see, e.g., Kleinman [12] or Feitzinger et al. [13]), variable step sizes can hugely improve the performance (Benner, Byers [14], Benner et al. [1]). We will also observe this in our numerical tests, see Figure 2 in Section 5. The line search method and analysis in [14] are based on exact Lyapunov equation solves, which guarantees that some favorable properties of the Kleinman-Newton iterates are automatically preserved. Our paper [1] extends line search algorithms and their analyses to inexact solves. An inexact Kleinman-Newton method without line search is analyzed in [13], but some assumptions made in [13] do not hold when low-rank methods are applied to solve the Lyapunov equation iteratively. We extended the inexact Kleinman-Newton method and analysis to integrate the efficient low-rank ADI solver in [1]. This section reviews the main algorithmic components of [1] applied to the projected GCARE (2.15). The following Section 4 then carefully exploits the special structure of the projected GCARE (2.15) for an efficient numerical realization.

#### 3.1. Inexact Kleinman-Newton Method

At its core our method is an inexact Newton method applied to the GCARE  $\mathcal{R}(\mathcal{X}) = 0$ , where  $\mathcal{R}(\mathcal{X})$  is the Riccati residual (2.16). Given an approximate solution  $\mathcal{X}^{(k)} \in \mathbb{R}^{(n_v - n_p) \times (n_v - n_p)}$  and a so-called forcing parameter  $\eta_k \in (0, 1)$ , we compute a step  $\mathcal{S}^{(k)} \in \mathbb{R}^{(n_v - n_p) \times (n_v - n_p)}$  that satisfies

$$\|\mathcal{R}'(\mathcal{X}^{(k)})\mathcal{S}^{(k)} + \mathcal{R}(\mathcal{X}^{(k)})\|_F \leq \eta_k \|\mathcal{R}(\mathcal{X}^{(k)})\|_F. \quad (3.1)$$

Then we compute a step size  $\xi_k \in (0, 1]$  such that the sufficient decrease condition

$$\|\mathcal{R}(\mathcal{X}^{(k)} + \xi_k \mathcal{S}^{(k)})\|_F \leq (1 - \xi_k \beta) \|\mathcal{R}(\mathcal{X}^{(k)})\|_F \quad (3.2)$$

is satisfied, where  $\beta > 0$  is a given parameter. The new iterate is

$$\mathcal{X}^{(k+1)} = \mathcal{X}^{(k)} + \xi_k \mathcal{S}^{(k)}. \quad (3.3)$$

We will discuss below how we compute an  $\mathcal{S}^{(k)}$  that satisfies (3.1). As we have shown in [1], if the forcing parameters in (3.1) are limited by

$$\eta_k \leq \bar{\eta} < 1 \quad \text{and} \quad \beta \in (0, 1 - \bar{\eta}),$$

then the sufficient decrease condition (3.2) is satisfied for all step sizes  $\xi_k$

$$0 < \xi_k \leq (1 - \bar{\eta} - \beta) \frac{\|\mathcal{R}(\mathcal{X}^{(k)})\|_F}{\|\mathcal{M}\mathcal{S}^{(k)}\mathcal{B}\mathcal{B}^T\mathcal{S}^{(k)}\mathcal{M}\|_F}. \quad (3.4)$$

To ensure convergence of the sequence of iterates  $\{\mathcal{X}^{(k)}\}$ , the step sizes  $\xi_k$  also need to be bounded away from zero. We will state the precise convergence result later, see Theorem 6 below. We use the Armijo rule to compute the step sizes  $\xi_k$ . This step size rule and others are discussed in [1], as well as conditions that ensure  $\xi_k \geq \xi_{\min} > 0$  for all  $k$ .

Instead of computing the new iterate  $\mathcal{S}^{(k)}$  as an approximate solution of  $\mathcal{R}'(\mathcal{X}^{(k)})\mathcal{S}^{(k)} = -\mathcal{R}(\mathcal{X}^{(k)})$ , it is more favorable for our purposes to compute

$$\tilde{\mathcal{X}}^{(k+1)} := \mathcal{X}^{(k)} + \mathcal{S}^{(k)} \quad (3.5)$$

as an approximate solution of  $\mathcal{R}'(\mathcal{X}^{(k)})\tilde{\mathcal{X}}^{(k+1)} = -\mathcal{R}(\mathcal{X}^{(k)}) + \mathcal{R}'(\mathcal{X}^{(k)})\mathcal{X}^{(k)}$ . Both equations  $\mathcal{R}'(\mathcal{X}^{(k)})\mathcal{S}^{(k)} = -\mathcal{R}(\mathcal{X}^{(k)})$  and  $\mathcal{R}'(\mathcal{X}^{(k)})\tilde{\mathcal{X}}^{(k+1)} = -\mathcal{R}(\mathcal{X}^{(k)}) + \mathcal{R}'(\mathcal{X}^{(k)})\mathcal{X}^{(k)}$  are Lyapunov equations, but the right hand side of the latter equation,

$$-\mathcal{R}(\mathcal{X}^{(k)}) + \mathcal{R}'(\mathcal{X}^{(k)})\mathcal{X}^{(k)} = -\mathcal{C}^T\mathcal{C} - \mathcal{M}\mathcal{X}^{(k)}\mathcal{B}\mathcal{B}^T\mathcal{X}^{(k)}\mathcal{M} = -[\mathcal{C}^T \quad \mathcal{K}^{(k)}] [\mathcal{C}^T \quad \mathcal{K}^{(k)}]^T,$$

where  $\mathcal{K}^{(k)}$  is defined in (2.18), is low-rank and this will allow the application of the efficient low-rank ADI method (discussed in the next section) to compute  $\tilde{\mathcal{X}}^{(k+1)}$ . Note that

$$\mathcal{R}'(\mathcal{X}^{(k)})\tilde{\mathcal{X}}^{(k+1)} = (\mathcal{A}^{(k)})^T \tilde{\mathcal{X}}^{(k+1)} \mathcal{M} + \mathcal{M} \tilde{\mathcal{X}}^{(k+1)} \mathcal{A}^{(k)},$$

where

$$\mathcal{A}^{(k)} = \mathcal{A} - \mathcal{B}\mathcal{B}^T \mathcal{X}^{(k)} \mathcal{M} = \mathcal{A} - \mathcal{B} (\mathcal{K}^{(k)})^T. \quad (3.6)$$

We define the projected Lyapunov residual at any  $\tilde{\mathcal{X}}^{(k+1)}$  by

$$\mathcal{L}(\tilde{\mathcal{X}}^{(k+1)}) := \mathcal{R}'(\mathcal{X}^{(k)})\tilde{\mathcal{X}}^{(k+1)} + \mathcal{R}(\mathcal{X}^{(k)}) - \mathcal{R}'(\mathcal{X}^{(k)})\mathcal{X}^{(k)} = \mathcal{R}'(\mathcal{X}^{(k)})\mathcal{S}^{(k)} + \mathcal{R}(\mathcal{X}^{(k)}). \quad (3.7)$$

The inexactness condition (3.1) means that we have to compute  $\tilde{\mathcal{X}}^{(k+1)}$  with

$$(\mathcal{A}^{(k)})^T \tilde{\mathcal{X}}^{(k+1)} \mathcal{M} + \mathcal{M} \tilde{\mathcal{X}}^{(k+1)} \mathcal{A}^{(k)} = - [\mathcal{C}^T \quad \mathcal{K}^{(k)}] [\mathcal{C}^T \quad \mathcal{K}^{(k)}]^T + \mathcal{L}(\tilde{\mathcal{X}}^{(k+1)}) \quad (3.8)$$

such that the corresponding projected Lyapunov residual satisfies

$$\|\mathcal{L}(\tilde{\mathcal{X}}^{(k+1)})\|_F \leq \eta_k \|\mathcal{R}(\mathcal{X}^{(k)})\|_F. \quad (3.9)$$

Using the definition (2.16), (3.5), and (3.8), the residual of the projected CARE at (3.3) can be written as

$$\begin{aligned} \mathcal{R}(\mathcal{X}^{(k)} + \xi_k \mathcal{S}^{(k)}) &= \mathcal{R}(\mathcal{X}^{(k)}) + \xi_k \mathcal{R}'(\mathcal{X}^{(k)})\mathcal{S}^{(k)} + \frac{\xi_k^2}{2} \mathcal{R}''(\mathcal{X}^{(k)})(\mathcal{S}^{(k)}, \mathcal{S}^{(k)}) \\ &= (1 - \xi_k) \mathcal{R}(\mathcal{X}^{(k)}) + \xi_k \mathcal{L}(\tilde{\mathcal{X}}^{(k+1)}) - \xi_k^2 \mathcal{M} \mathcal{S}^{(k)} \mathcal{B} \mathcal{B}^T \mathcal{S}^{(k)} \mathcal{M}, \end{aligned} \quad (3.10)$$

which can be evaluated efficiently for any  $\xi_k$ , and therefore can be used to efficiently compute a step size  $\xi_k > 0$  that satisfies (3.2).

The inexact Kleinman-Newton method with line search is summarized in Algorithm 1 below.

---

**Algorithm 1** Inexact Kleinman-Newton method with line search

---

**Input:**  $\mathcal{A}, \mathcal{M}, \mathcal{B}, \mathcal{C}$ ,  $tol_{\text{Newton}}$ , initial stabilizing iterate  $\mathcal{X}^{(0)}$ ,  $\bar{\eta} \in (0, 1)$ , and  $\beta \in (0, 1 - \bar{\eta})$

**Output:** Approximate unique stabilizing solution  $\mathcal{X}^{(*)}$  of GCARE (2.15)

- 1: Set  $k = 0$ .
- 2: **while**  $\|\mathcal{R}(\mathcal{X}^{(k)})\|_F > tol_{\text{Newton}}$  **do**
- 3:  $\mathcal{K}^{(k)} = \mathcal{M} \mathcal{X}^{(k)} \mathcal{B}$
- 4: Set  $\mathcal{A}^{(k)} = \mathcal{A} - \mathcal{B} (\mathcal{K}^{(k)})^T$ .
- 5: Select  $\eta_k \in (0, \bar{\eta}]$ .
- 6: Compute  $\tilde{\mathcal{X}}^{(k+1)}$  that solves the inexact Lyapunov equation

$$(\mathcal{A}^{(k)})^T \tilde{\mathcal{X}}^{(k+1)} \mathcal{M} + \mathcal{M} \tilde{\mathcal{X}}^{(k+1)} \mathcal{A}^{(k)} = - [\mathcal{C}^T \quad \mathcal{K}^{(k)}] [\mathcal{C}^T \quad \mathcal{K}^{(k)}]^T + \mathcal{L}(\tilde{\mathcal{X}}^{(k+1)}) \quad (3.11a)$$

$$\text{with } \|\mathcal{L}(\tilde{\mathcal{X}}^{(k+1)})\|_F \leq \eta_k \|\mathcal{R}(\mathcal{X}^{(k)})\|_F. \quad (3.11b)$$

- 7: Compute  $\xi_k \in (0, 1)$  such that  $\|\mathcal{R}((1 - \xi_k)\mathcal{X}^{(k)} + \xi_k \tilde{\mathcal{X}}^{(k+1)})\|_F \leq (1 - \xi_k \beta) \|\mathcal{R}(\mathcal{X}^{(k)})\|_F$ .
  - 8: Set  $\mathcal{X}^{(k+1)} = (1 - \xi_k)\mathcal{X}^{(k)} + \xi_k \tilde{\mathcal{X}}^{(k+1)}$ .
  - 9:  $k = k + 1$
  - 10: **end while**
  - 11:  $\mathcal{X}^{(*)} = \mathcal{X}^{(k)}$
- 

The following convergence theorem for the iterates generated by Algorithm 1 is adopted from [1, Thm. 10] to match the notation of the projected Riccati equation (2.15).



**Theorem 6.** Let  $(\mathcal{A}, \mathcal{B}; \mathcal{M})$  be stabilizable, let  $(\mathcal{C}, \mathcal{A}; \mathcal{M})$  be detectable and assume that for all  $k$ , there exists a symmetric positive semi-definite  $\tilde{\mathcal{X}}^{(k+1)}$  such that (3.8) and (3.9) hold. Furthermore, let  $\mathcal{X}^{(k)}$  be the iterates generated by Algorithm 1 and  $\mathcal{A}^{(k)} = \mathcal{A} - \mathcal{B}(\mathcal{M}\mathcal{X}^{(k)}\mathcal{B})^T$ .

- (i) If the step sizes are bounded away from zero, i.e.,  $\xi_k \geq \xi_{\min} > 0$  for all  $k$ , then  $\|\mathcal{R}(\mathcal{X}^{(k)})\|_F \rightarrow 0$ .
- (ii) If in addition the pencils  $(\mathcal{A}^{(k)}, \mathcal{M})$  are stable for  $k \geq k_0$ , and  $\mathcal{X}^{(k)} \succeq 0$  for all  $k \geq k_0$ , then  $\mathcal{X}^{(k)} \rightarrow \mathcal{X}^{(*)}$ , where  $\mathcal{X}^{(*)} \succeq 0$  is the unique stabilizing solution of the GCARE (2.15).

### 3.2. Improved Low-Rank ADI Method

The main expense in the inexact Kleinman-Newton Algorithm 1 is in Step 6. We apply the real low-rank ADI method, which is detailed in [1] and in [7, Sec. 6.3.1]. This method generates a low-rank approximate solution  $\tilde{\mathcal{X}}^{(k+1)}$  of the Lyapunov equation in factored form. Compared to the original version of the ADI method [15, 16], which is also the version used in Bansch et al. [3], we use two important modifications of the original ADI method. The first reorganizes the computation to obtain a low-rank representation of the Lyapunov residual in the ADI iterations [17], and the second exploits the fact that the ADI shifts must occur either as a real number or as a pair of complex conjugate numbers to write almost all<sup>3</sup> matrices in the ADI iterations as real matrices [17]. Most importantly, the improved method generates a real matrices  $\mathcal{Z}$  and  $\tilde{\mathcal{W}}_\ell$ , each with few columns, such that  $\mathcal{Z}\mathcal{Z}^T = \tilde{\mathcal{X}}^{(k+1)}$  satisfies (3.11a) and the corresponding Lyapunov residual  $\mathcal{L}(\tilde{\mathcal{X}}^{(k+1)}) = \tilde{\mathcal{W}}_\ell \tilde{\mathcal{W}}_\ell^T$  obeys (3.11). We refer to [1] or [7, Sec. 6.3.1] for details on the derivation of the real low-rank ADI method. The detailed listing of this method is given in Algorithm 2 below.

---

#### Algorithm 2 Generalized real-valued low-rank residual ADI method

---

**Input:**  $\mathcal{A}^{(k)}, \mathcal{K}^{(k)}, \mathcal{C}$ , shifts  $\{q_i\}_{i=1}^\ell = \overline{\{q_i\}_{i=1}^\ell} \in \mathbb{C}^-$   
**Output:**  $\mathcal{Z}$  such that  $\mathcal{Z}\mathcal{Z}^T = \tilde{\mathcal{X}}^{(k+1)}$  and  $\mathcal{L}(\tilde{\mathcal{X}}^{(k+1)}) = \tilde{\mathcal{W}}_\ell \tilde{\mathcal{W}}_\ell^T$  satisfy (3.11)

- 1: Set  $\ell = 1$ ,  $\mathcal{Z} = []$ ,  $\tilde{\mathcal{W}}_0 = [\mathcal{C}^T \quad \mathcal{K}^{(k)}]$ .
- 2: **while**  $\|\tilde{\mathcal{W}}_{\ell-1}^T \tilde{\mathcal{W}}_{\ell-1}\|_F > \eta_k \|\mathcal{R}(\mathcal{X}^{(k)})\|_F$  **do**
- 3:    $\mathcal{V}_\ell = \left( (\mathcal{A}^{(k)})^T + q_\ell \mathcal{M} \right)^{-1} \tilde{\mathcal{W}}_{\ell-1}$
- 4:   **if**  $\text{Im}(q_\ell) = 0$  **then**
- 5:      $\tilde{\mathcal{W}}_\ell = \tilde{\mathcal{W}}_{\ell-1} - 2q_\ell \mathcal{M} \mathcal{V}_\ell$
- 6:      $\tilde{\mathcal{V}}_\ell = \sqrt{-2q_\ell} \mathcal{V}_\ell$
- 7:   **else**
- 8:      $\gamma_\ell = 2\sqrt{-\text{Re}(q_\ell)}$ ,    $\delta_\ell = \text{Re}(q_\ell) / \text{Im}(q_\ell)$
- 9:      $\tilde{\mathcal{W}}_\ell = \tilde{\mathcal{W}}_{\ell-1} + \gamma_\ell^2 \mathcal{M} (\text{Re}(\mathcal{V}_\ell) + \delta_\ell \text{Im}(\mathcal{V}_\ell))$
- 10:      $\tilde{\mathcal{V}}_{\ell+1} = [\gamma_\ell (\text{Re}(\mathcal{V}_\ell) + \delta_\ell \text{Im}(\mathcal{V}_\ell)) \quad \gamma_\ell \sqrt{(\delta_\ell^2 + 1)} \text{Im}(\mathcal{V}_\ell)]$
- 11:      $\ell = \ell + 1$
- 12:   **end if**
- 13:    $\mathcal{Z} = \begin{bmatrix} \mathcal{Z} & \tilde{\mathcal{V}}_\ell \end{bmatrix}$
- 14:    $\ell = \ell + 1$
- 15: **end while**

---

Algorithms 1 and 2 work with the projected matrices, but need to be implemented operating on the matrices  $\mathbf{M}, \mathbf{A}, \mathbf{B}, \mathbf{C}$ . This transformation will be described in the next section.

---

<sup>3</sup>The linear system solve still has a complex coefficient matrix and thus the intermediate  $\mathcal{V}_\ell$  is complex. This can be avoided along the lines of [18, Remark 4.4], but is not done in our implementation.

#### 4. Inexact Kleinman-Newton for Algebraic Riccati Equations Associated with Index-2 DAEs

The inexact Kleinman-Newton Algorithm 1 and the improved ADI Algorithm 2 are derived and stated in terms of the projected matrices in (2.12). As stated before, these matrix are dense, expensive to compute with and the explicit use of the projection needs to be avoided. As before, we use calligraphic font, like  $\mathcal{X}^{(k)}$ , to denote projected quantities, and roman font, like  $X^{(k)}$ , to denote the corresponding quantities without projection.

Regarding the transformation of the iterates in the inexact Kleinman-Newton Algorithm 1, we already know from (2.19) that

$$X^{(k)} = \Theta_r \mathcal{X}^{(k)} \Theta_r^T \in \mathbb{R}^{n_v \times n_v}, \quad (4.1a)$$

$$(K^{(k)})^T = B^T X^{(k)} M \in \mathbb{R}^{n_u \times n_v}, \quad \text{and} \quad (K^{(k)})^T \Theta_r = (\mathcal{K}^{(k)})^T. \quad (4.1b)$$

To undo the projections, we multiply the Lyapunov equations and the Riccati residuals from the left by  $\Theta_l$  and from the right by  $\Theta_l^T$  and replace Steps 6 and 7 in Algorithm 1 by the following.

6: Compute  $\tilde{\mathcal{X}}^{(k+1)}$  that solves the inexact Lyapunov equation

$$\begin{aligned} & \Theta_l (\mathcal{A}^{(k)})^T \tilde{\mathcal{X}}^{(k+1)} \mathcal{M} \Theta_l^T + \Theta_l \mathcal{M} \tilde{\mathcal{X}}^{(k+1)} \mathcal{A}^{(k)} \Theta_l^T \\ &= -\Theta_l^T [\mathcal{C}^T \quad \mathcal{K}^{(k)}] [\mathcal{C}^T \quad \mathcal{K}^{(k)}]^T \Theta_l^T + \Theta_l \mathcal{L}(\tilde{\mathcal{X}}^{(k+1)}) \Theta_l^T \end{aligned} \quad (4.2a)$$

with

$$\|\Theta_l \mathcal{L}(\tilde{\mathcal{X}}^{(k+1)}) \Theta_l^T\|_F \leq \eta_k \|\Theta_l \mathcal{R}(\mathcal{X}^{(k)}) \Theta_l^T\|_F. \quad (4.2b)$$

7: Compute  $\xi_k \in (0, 1)$  such that  $\|\Theta_l \mathcal{R}((1 - \xi_k) \mathcal{X}^{(k)} + \xi_k \tilde{\mathcal{X}}^{(k+1)}) \Theta_l^T\|_F \leq (1 - \xi_k \beta) \|\Theta_l \mathcal{R}(\mathcal{X}^{(k)}) \Theta_l^T\|_F$ .

For any symmetric matrix  $\mathcal{S} \in \mathbb{R}^{(n_v - n_p) \times (n_v - n_p)}$ , because  $\Theta_l \in \mathbb{R}^{n_v \times (n_v - n_p)}$  has rank  $n_v - n_p$ ,  $\Theta_l \mathcal{S} \Theta_l^T = 0$  if and only if  $\mathcal{S} = 0$ . Thus, replacing Steps 6 and 7 in Algorithm 1 by the Steps 6 and 7 above replaces the Frobenius norm  $\|\cdot\|_F$  by the weighted Frobenius norm  $\|\Theta_l \cdot \Theta_l^T\|_F$ . While this change in norm influences the iterates (e.g., because the residual norm is changed when the inexact Lyapunov equation is solved), it does not change the fundamental convergence behavior. In particular, Theorem 6 remains valid when the weighted Frobenius norm is used.

The reason for multiplying by  $\Theta_l$  and  $\Theta_l^T$  is that the projection  $\Pi$  emerges. In fact, using (2.12), (2.9), and (4.1), the left hand side in (4.2a) becomes

$$\Theta_l (\mathcal{A}^{(k)})^T \tilde{\mathcal{X}}^{(k+1)} \mathcal{M} \Theta_l^T + \Theta_l \mathcal{M} \tilde{\mathcal{X}}^{(k+1)} \mathcal{A}^{(k)} \Theta_l^T = \Pi (\mathcal{A}^{(k)})^T \tilde{X}^{(k+1)} M \Pi^T + \Pi M \tilde{X}^{(k+1)} \mathcal{A}^{(k)} \Pi^T, \quad (4.3)$$

where

$$A^{(k)} = A - B(K^{(k)})^T.$$

Although the projection  $\Pi$  emerges in (4.3), it will not be computed and used explicitly. We outline the main ideas in the following subsections.

##### 4.1. Low-Rank Residual ADI for Index-2 DAE Systems

Recall (2.9) and (2.12). We have

$$[\mathcal{C}^T \quad \mathcal{K}^{(k)}] = \Theta_r^T [\mathcal{C}^T \quad K^{(k)}]. \quad (4.4)$$

To transform the matrices in the improved ADI Algorithm 2 we set

$$\tilde{W}_{\ell-1} = \Theta_r^T \tilde{W}_{\ell-1}, \quad \ell \geq 1 \quad \text{and} \quad \tilde{W}_0 := [\mathcal{C}^T \quad K^{(k)}]. \quad (4.5)$$

Using (2.12) and (4.1), the linear system in Step 3 of Algorithm 2 is transformed into

$$(\mathcal{A}^T + q_\ell \mathcal{M} - \mathcal{K}^{(k)} \mathcal{B}^T) \mathcal{V}_\ell = \Theta_r^T \left( A^T + q_\ell M - K^{(k)} B^T \right) \Theta_r \mathcal{V}_\ell = \Theta_r^T \widetilde{W}_{\ell-1} = \widetilde{W}_{\ell-1}. \quad (4.6)$$

We define

$$V_\ell = \Theta_r \mathcal{V}_\ell, \quad \ell \geq 1. \quad (4.7a)$$

From (2.9) it follows that

$$\Pi^T V_\ell = V_\ell, \quad \ell \geq 1. \quad (4.7b)$$

Finally, multiplying (4.6) by  $\Theta_l$  from the left, using (2.9), (4.7a) and (4.7b), the linear system in Step 3 of Algorithm 2 is written as

$$\Pi \left( A^T + q_\ell M - K^{(k)} B^T \right) \Pi^T V_\ell = \Pi \widetilde{W}_{\ell-1}. \quad (4.8)$$

As it is shown by Heinkenschloss et al. [2] and Bänsch et al. [3] the solution of the projected system (4.8) is equivalent to the solution of the  $2 \times 2$  block system

$$\begin{bmatrix} A^T + q_\ell M - K^{(k)} B^T & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} V_\ell \\ * \end{bmatrix} = \begin{bmatrix} \widetilde{W}_{\ell-1} \\ 0 \end{bmatrix}, \quad (4.9)$$

where “\*” indicates that the second block of the solution matrix is not needed. Finally, since  $K^{(k)} B^T$  is dense, the matrix in (4.9) is written as a low-rank perturbation

$$\begin{bmatrix} A^T + q_\ell M - K^{(k)} B^T & G \\ G^T & 0 \end{bmatrix} = \begin{bmatrix} A^T + q_\ell M & G \\ G^T & 0 \end{bmatrix} - \begin{bmatrix} K^{(k)} \\ 0 \end{bmatrix} \begin{bmatrix} B^T & 0 \end{bmatrix}$$

and the solution of (4.9) is computed using the Sherman-Morrison-Woodbury formula. See Bänsch et al. [3] or Weichelt [7, p. 67].

We use (2.9), (4.5) to write the projected Lyapunov residual

$$\Theta_l \mathcal{L}(\widetilde{\mathcal{X}}_\ell^{(k+1)}) \Theta_l^T = \Theta_l \widetilde{W}_\ell \widetilde{W}_\ell^T \Theta_l^T = \Pi \widetilde{W}_\ell \widetilde{W}_\ell^T \Pi^T =: \overline{W}_\ell \overline{W}_\ell^T. \quad (4.10)$$

Rather than computing  $\widetilde{W}_\ell$  and then multiplying by  $\Pi$ , we can update  $\overline{W}_\ell = \Pi \widetilde{W}_\ell$  directly. In fact, multiplying line 5 in Algorithm 2 with  $\Theta_l$  from the left and using (4.5), (4.7a) yields

$$\Pi \widetilde{W}_\ell = \Theta_l \Theta_r^T \widetilde{W}_\ell = \Theta_l \Theta_r^T \widetilde{W}_{\ell-1} - 2q_\ell \Theta_l \Theta_r^T M \Theta_r \mathcal{V}_\ell = \Pi \widetilde{W}_{\ell-1} - 2q_\ell \Pi M V_\ell = \Pi \widetilde{W}_{\ell-1} - 2q_\ell M V_\ell,$$

where in the last step we have used the  $M$ -orthogonality of  $\Pi$ , i.e.,  $\Pi M = M \Pi^T$  and (4.7b). Thus, the projected low-rank residual factor can be accumulated via

$$\overline{W}_\ell = \overline{W}_{\ell-1} - 2q_\ell M V_\ell \quad (4.11)$$

without using any explicit projections. Only the initial right hand side  $W^{(k)}$  needs to be projected to define

$$\overline{W}_0 := \Pi \begin{bmatrix} C^T & K^{(k)} \end{bmatrix}. \quad (4.12)$$

This one projection at the beginning of the ADI method is computed by first solving

$$\begin{bmatrix} M & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} W \\ * \end{bmatrix} = \begin{bmatrix} C^T & K^{(k)} \\ 0 \end{bmatrix}$$

(again, “\*” indicates that the second block of the solution is not used) and then setting

$$\overline{W}_0 = MW.$$

See Heinkenschloss et al. [2] or Weichelt [7, Lemma 4.1]. This projection is less expensive than a single ADI step and does not considerably increase the overall computation costs. Moreover, the right-hand side  $\widetilde{W}_{\ell-1}$  in (4.8), (4.9) can be replaced by  $\overline{W}_{\ell-1}$ , since

$$\Pi \widetilde{W}_{\ell-1} = \Pi \Pi \widetilde{W}_{\ell-1} = \Pi \overline{W}_{\ell-1}.$$

To incorporate this improved ADI method into Algorithm 1, some remaining issues, such as the storage of the Newton step and the projected Riccati residual, need to be addressed. This is done in the next subsection, expanding the statements in [1, Sec. 5.2].

#### 4.2. Low-Rank Riccati Residual for Index-2 DAE systems

The Newton step  $S^{(k)} = \Theta_r \mathcal{S}^{(k)} \Theta_r^T$  is only used in the computation of the step size  $\xi_k$ , since the inexact Kleinman-Newton step (3.8) directly iterates over the preliminary solution  $\widetilde{X}^{(k)} = \Theta_r \widetilde{\mathcal{X}}^{(k)} \Theta_r^T$ . Furthermore,  $\mathcal{S}^{(k)}$  always occurs in products  $\mathcal{M} \mathcal{S}^{(k)} \mathcal{B} \in \mathbb{R}^{(n_v - n_p) \times n_r}$ . Using (3.3), (3.5), and the definition of the feedback matrix in (2.14), this product can be written as

$$\mathcal{M} \mathcal{S}^{(k)} \mathcal{B} = \begin{cases} \mathcal{M} \widetilde{\mathcal{X}}^{(k+1)} \mathcal{B} - \mathcal{M} \mathcal{X}^{(k)} \mathcal{B} =: \widetilde{\mathcal{K}}^{(k+1)} - \mathcal{K}^{(k)} =: \Delta \widetilde{\mathcal{K}}^{(k+1)}, & \xi_k \neq 1, \\ \mathcal{M} \mathcal{X}^{(k+1)} \mathcal{B} - \mathcal{M} \mathcal{X}^{(k)} \mathcal{B} =: \mathcal{K}^{(k+1)} - \mathcal{K}^{(k)} =: \Delta \mathcal{K}^{(k+1)}, & \xi_k = 1, \end{cases} \quad (4.13)$$

which characterizes the feedback change corresponding to the preliminary or definite new iterate  $\widetilde{\mathcal{X}}^{(k+1)}$  or  $\mathcal{X}^{(k+1)}$ . Using (2.12), (4.1), and  $S^{(k)} = \Theta_r \mathcal{S}^{(k)} \Theta_r^T$ , (4.13) becomes

$$M S^{(k)} B = \begin{cases} M \widetilde{X}^{(k+1)} B - M X^{(k)} B =: \widetilde{K}^{(k+1)} - K^{(k)} =: \Delta \widetilde{K}^{(k+1)}, & \xi_k \neq 1, \\ M X^{(k+1)} B - M X^{(k)} B =: K^{(k+1)} - K^{(k)} =: \Delta K^{(k+1)}, & \xi_k = 1, \end{cases} \quad (4.14)$$

which characterizes the feedback change corresponding to the preliminary new iterate or  $\widetilde{X}^{(k+1)}$  or new iterate  $X^{(k+1)}$ . Hence, the dense Newton step  $S^{(k)}$  is never formed explicitly.

The definition  $\widetilde{\mathcal{X}}^{(k+1)} = \mathcal{Z} \mathcal{Z}^T$  and update in Step 13 of Algorithm 2 implies the formula

$$\widetilde{\mathcal{X}}_\ell^{(k+1)} = \widetilde{\mathcal{X}}_{\ell-1}^{(k+1)} + \widetilde{V}_\ell \widetilde{V}_\ell^T, \quad \ell \geq 1, \quad (4.15)$$

for the implicit iterate  $\widetilde{\mathcal{X}}_\ell^{(k+1)}$  in Algorithm 2. Algorithm 2 and (4.7a) lead to the definition

$$\widetilde{V}_\ell = \Theta_r \widetilde{V}_\ell, \quad \ell \geq 1. \quad (4.16)$$

Finally, (4.14), (4.15), (4.1), and (4.7a) imply that the feedback change can be accumulated during the ADI algorithm as follows

$$\begin{aligned} \Delta \widetilde{K}_\ell^{(k+1)} &= \widetilde{K}_\ell^{(k+1)} - K^{(k)} = \widetilde{K}_{\ell-1}^{(k+1)} + M \widetilde{V}_\ell (\widetilde{V}_\ell^T B) - K^{(k)} \\ &= \Delta \widetilde{K}_{\ell-1}^{(k+1)} + M \widetilde{V}_\ell (\widetilde{V}_\ell^T B), \end{aligned} \quad \forall \ell \geq 1 \quad (4.17)$$

with  $\Delta \widetilde{K}_0^{(k+1)} = -K^{(k)}$ ; compare [1, Sec. 5.2]. If we consider the feedback change at the final ADI iteration  $\ell$ , we simply write  $\Delta \widetilde{K}^{(k+1)}$  instead of  $\Delta \widetilde{K}_\ell^{(k+1)}$ .

The Riccati residual can be written in low-rank form as

$$\mathcal{R}(\mathcal{X}^{(k)}) = \mathcal{W}^{(k)} \left( \mathcal{W}^{(k)} \right)^T - \Delta \mathcal{K}^{(k)} \left( \Delta \mathcal{K}^{(k)} \right)^T =: \mathcal{U}^{(k)} \mathcal{D} \left( \mathcal{U}^{(k)} \right)^T \quad (4.18a)$$

with

$$\mathcal{U}^{(k+1)} = [\mathcal{W}^{(k+1)} \quad \Delta \mathcal{K}^{(k+1)}], \quad \mathcal{D} = \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix}. \quad (4.18b)$$

This representation can be used to efficiently compute  $\|\Theta_l \mathcal{R}(\mathcal{X}^{(k)}) \Theta_l^T\|_F$ .

In the initial iteration  $k = 0$  with  $\mathcal{X}^{(0)} = 0$ , (4.18) holds with  $\mathcal{W}^{(0)} = \mathcal{C}^T$  and  $\Delta \mathcal{K}^{(k)} = 0$ . Equation (3.10) and  $\mathcal{L}(\tilde{\mathcal{X}}^{(k+1)}) = \tilde{\mathcal{W}}_\ell \tilde{\mathcal{W}}_\ell^T$  imply

$$\begin{aligned}
\mathcal{R}(\mathcal{X}^{(k+1)}) &= \mathcal{R}(\mathcal{X}^{(k)} + \xi_k \mathcal{S}^{(k)}) \\
&= (1 - \xi_k) \mathcal{U}^{(k)} \mathcal{D} \left( \mathcal{U}^{(k)} \right)^T + \xi_k \tilde{\mathcal{W}}_\ell \tilde{\mathcal{W}}_\ell^T - \xi_k^2 \Delta \tilde{\mathcal{K}}^{(k+1)} \left( \Delta \tilde{\mathcal{K}}^{(k+1)} \right)^T \\
&= (1 - \xi_k) \left( \mathcal{W}^{(k)} \left( \mathcal{W}^{(k)} \right)^T - \Delta \mathcal{K}^{(k)} \left( \Delta \mathcal{K}^{(k)} \right)^T \right) + \xi_k \tilde{\mathcal{W}}_\ell \tilde{\mathcal{W}}_\ell^T - \xi_k^2 \Delta \tilde{\mathcal{K}}^{(k+1)} \left( \Delta \tilde{\mathcal{K}}^{(k+1)} \right)^T \\
&= \left[ \begin{array}{cc} \sqrt{(1 - \xi_k)} \mathcal{W}^{(k)} & \sqrt{\xi_k} \tilde{\mathcal{W}}_\ell \end{array} \right] \left[ \begin{array}{cc} \sqrt{(1 - \xi_k)} \Delta \mathcal{K}^{(k)} & \xi_k \Delta \tilde{\mathcal{K}}^{(k+1)} \end{array} \right] \times \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} \\
&\quad \times \left[ \begin{array}{cc} \sqrt{(1 - \xi_k)} \mathcal{W}^{(k)} & \sqrt{\xi_k} \tilde{\mathcal{W}}_\ell \end{array} \right] \left[ \begin{array}{cc} \sqrt{(1 - \xi_k)} \Delta \mathcal{K}^{(k)} & \xi_k \Delta \tilde{\mathcal{K}}^{(k+1)} \end{array} \right]^T, \tag{4.19}
\end{aligned}$$

which is of the form (4.18) with

$$\mathcal{W}^{(k+1)} := \left[ \begin{array}{cc} \sqrt{(1 - \xi_k)} \mathcal{W}^{(k)} & \sqrt{\xi_k} \tilde{\mathcal{W}}_\ell \end{array} \right], \quad \Delta \mathcal{K}^{(k+1)} := \left[ \begin{array}{cc} \sqrt{(1 - \xi_k)} \Delta \mathcal{K}^{(k)} & \xi_k \Delta \tilde{\mathcal{K}}^{(k+1)} \end{array} \right].$$

Using (4.5), (4.19) the projected Riccati residual  $\mathcal{R}(X^{(k+1)}) := \Theta_l \mathcal{R}(\mathcal{X}^{(k+1)}) \Theta_l^T$  can be written as

$$\begin{aligned}
\mathcal{R}(X^{(k+1)}) &:= \Theta_l \mathcal{R}(\mathcal{X}^{(k+1)}) \Theta_l^T \\
&= \Theta_l \mathcal{W}^{(k+1)} \left( \mathcal{W}^{(k+1)} \right)^T \Theta_l^T - \Theta_l \Delta \mathcal{K}^{(k+1)} \left( \Delta \mathcal{K}^{(k+1)} \right)^T \Theta_l^T \\
&= \Pi W^{(k+1)} \left( W^{(k+1)} \right)^T \Pi^T - \Pi \Delta K^{(k+1)} \left( \Delta K^{(k+1)} \right)^T \Pi^T \\
&= \bar{W}^{(k+1)} \left( \bar{W}^{(k+1)} \right)^T - \Delta K^{(k+1)} \left( \Delta K^{(k+1)} \right)^T =: U^{(k+1)} \mathcal{D} \left( U^{(k+1)} \right)^T \tag{4.20}
\end{aligned}$$

with  $U^{(k+1)} = \left[ \begin{array}{cc} \bar{W}^{(k+1)} & \Delta K^{(k+1)} \end{array} \right]$ . In the second to last equation in (4.20) we have used the identity

$$\Pi \Delta K^{(k+1)} = \Pi M (X^{(k+1)} - X^{(k)}) B = M \Pi^T (X^{(k+1)} - X^{(k)}) B = \Delta K^{(k+1)}, \tag{4.21}$$

which follows from the  $M$ -orthogonality of  $\Pi$  and  $\Pi^T (X^{(k+1)} - X^{(k)}) = X^{(k+1)} - X^{(k)}$  (cf. (4.7b)). The updates of  $\mathcal{W}^{(k+1)}$  and  $\Delta \mathcal{K}^{(k+1)}$  imply

$$\begin{aligned}
\bar{W}^{(k+1)} &:= \left[ \begin{array}{cc} \sqrt{1 - \xi_k} \bar{W}^{(k)} & \sqrt{\xi_k} \tilde{W}_\ell \end{array} \right], \quad \xi_k \in (0, 1], \\
\bar{W}^{(0)} &:= \Pi \left[ \begin{array}{cc} C^T & K^{(0)} \end{array} \right], \\
\Delta K^{(k+1)} &:= \left[ \begin{array}{cc} \sqrt{1 - \xi_k} \Delta K^{(k)} & \xi_k \Delta \tilde{K}^{(k+1)} \end{array} \right], \quad \xi_k \in (0, 1], \tag{4.22}
\end{aligned}$$

where  $K^{(0)}$  is an initial stabilizing feedback.

Equation (4.20) shows that the Riccati residual  $\mathcal{R}(X^{(k+1)})$  can be computed without any additional explicit projection.

The representation (3.10) shows that  $\mathcal{R}(X^{(k)} + \xi_k \mathcal{S}^{(k)})$  is a quartic polynomial with scalar coefficients. Just as in [1, Sec. 5] this is used for an efficient implementation of the line search computation.

The final feedback at the end of the  $k+1$ -st Newton step is defined via

$$K^{(k+1)} = (1 - \xi_k) K^{(k)} + \xi_k \Delta \tilde{K}^{(k+1)}. \tag{4.23}$$

---

**Algorithm 3** Inexact low-rank Kleinman-Newton-ADI for index-2 DAE systems

---

**Input:**  $M, A, G, B, C$ , initial feedback  $K^{(0)}$ ,  $tol_{\text{Newton}}$ ,  $\bar{\eta} \in (0, 1)$ , and  $\beta \in (0, 1 - \bar{\eta})$

**Output:** feedback matrix  $K$

- 1: Set  $\bar{W}^{(0)} = H [C^T \quad K^{(0)}]$ ,  $\Delta K^{(0)} = 0$ ,  $U^{(0)} = \begin{bmatrix} \bar{W}^{(0)} & \Delta K^{(0)} \end{bmatrix}$ .
  - 2: Set  $k = 0$ .
  - 3: **while**  $\left( \|U^{(k)} \mathcal{D}(U^{(k)})^T\|_F > tol_{\text{Newton}} \|U^{(0)} \mathcal{D}(U^{(0)})^T\|_F \right)$  **do**
  - 4:   Compute ADI shifts  $\{q_i\}_{i=1}^{n_{\text{ADI}}} = \overline{\{q_i\}_{i=1}^{n_{\text{ADI}}}} \subset \mathbb{C}^-$  ordered such that complex pairs form consecutive entries and choose  $\eta_k \in (0, \bar{\eta}]$ .
  - 5:   Set  $\tilde{W}_0 = H [C^T \quad K^{(k)}]$ ,  $\Delta \tilde{K}_0 = -K^{(k)}$ .
  - 6:   Set  $\ell = 1$ .
  - 7:   **while**  $\left( \|\tilde{W}_{\ell-1}^T \tilde{W}_{\ell-1}\|_F > \eta_k \|U^{(k)} \mathcal{D}(U^{(k)})\|_F \right)$  **do**
  - 8:     Get  $V_\ell$  by solving
 
$$\begin{bmatrix} A^T - K^{(k)} B^T + q_\ell M & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} V_\ell \\ * \end{bmatrix} = \begin{bmatrix} \tilde{W}_{\ell-1} \\ 0 \end{bmatrix}.$$
  - 9:     **if**  $\text{Im}(q_\ell) = 0$  **then**
  - 10:        $\tilde{W}_\ell = \tilde{W}_{\ell-1} - 2q_\ell M V_\ell$
  - 11:        $\tilde{V}_\ell = \sqrt{-2q_\ell} V_\ell$
  - 12:        $\Delta \tilde{K}_{\ell+1} = \Delta \tilde{K}_{\ell-1} + M \tilde{V}_\ell (\tilde{V}_\ell^T B)$
  - 13:     **else**
  - 14:        $\gamma_\ell = 2\sqrt{-\text{Re}(q_\ell)}$ ,  $\delta_\ell = \text{Re}(q_\ell) / \text{Im}(q_\ell)$
  - 15:        $\tilde{W}_{\ell+1} = \tilde{W}_{\ell-1} + \gamma_\ell^2 M (\text{Re}(V_\ell) + \delta_\ell \text{Im}(V_\ell))$
  - 16:        $\tilde{V}_{\ell+1} = [\gamma_\ell (\text{Re}(V_\ell) + \delta_\ell \text{Im}(V_\ell)) \quad \gamma_\ell \sqrt{(\delta_\ell^2 + 1)} \text{Im}(V_\ell)]$
  - 17:        $\ell = \ell + 1$
  - 18:        $\Delta \tilde{K}_{\ell+1} = \Delta \tilde{K}_{\ell-2} + M \tilde{V}_\ell (\tilde{V}_\ell^T B)$
  - 19:     **end if**
  - 20:      $\tilde{U}_{\ell+1} = \begin{bmatrix} \tilde{W}_{\ell+1} & \Delta \tilde{K}_{\ell+1} \end{bmatrix}$
  - 21:      $\ell = \ell + 1$
  - 22:   **end while**
  - 23:   **if**  $\|\tilde{U}_\ell \mathcal{D} \tilde{U}_\ell^T\|_F > (1 - \beta) \|U^{(k)} \mathcal{D}(U^{(k)})^T\|_F$  **then**
  - 24:     Compute  $\xi_k \in (0, 1)$  using, e.g., the Armijo rule.
  - 25:   **else**
  - 26:      $\xi_k = 1$ .
  - 27:   **end if**
  - 28:    $\bar{W}^{(k+1)} = \begin{bmatrix} \sqrt{1 - \xi_k} \bar{W}^{(k)} & \sqrt{\xi_k} \tilde{W}_\ell \end{bmatrix}$
  - 29:    $\Delta K^{(k+1)} = \begin{bmatrix} \sqrt{1 - \xi_k} \Delta K^{(k)} & \xi_k \Delta \tilde{K}_\ell \end{bmatrix}$
  - 30:    $U^{(k+1)} = \begin{bmatrix} \bar{W}^{(k+1)} & \Delta K^{(k+1)} \end{bmatrix}$
  - 31:    $K^{(k+1)} = (1 - \xi_k) K^{(k)} + \xi_k \Delta \tilde{K}_\ell$
  - 32:    $k = k + 1$
  - 33: **end while**
  - 34:  $K = K^{(k)}$
-

Only the feedback matrix is needed, but if desired the Riccati iterate can be computed in low-rank form as follows. Assuming the previous Riccati iterate is defined via  $X^{(k)} = Z^{(k)}(Z^{(k)})^T$  and the preliminary solution is defined via  $\tilde{X}^{(k+1)} = \tilde{Z}^{(k+1)}(\tilde{Z}^{(k+1)})^T$ , the new Riccati iterate can be written as

$$\begin{aligned} X^{(k+1)} &= (1 - \xi_k)X^{(k)} + \xi_k\tilde{X}^{(k+1)} \\ &= (1 - \xi_k)Z^{(k)}(Z^{(k)})^T + \xi_k\tilde{Z}^{(k+1)}(\tilde{Z}^{(k+1)})^T \\ &= \begin{bmatrix} \sqrt{1 - \xi_k} Z^{(k)} & \sqrt{\xi_k} \tilde{Z}^{(k+1)} \end{bmatrix} \begin{bmatrix} \sqrt{1 - \xi_k} Z^{(k)} & \sqrt{\xi_k} \tilde{Z}^{(k+1)} \end{bmatrix}^T, \end{aligned} \quad (4.24)$$

whose size depends on the number of ADI steps in the  $k$ -th and  $k+1$ -st Newton iteration.

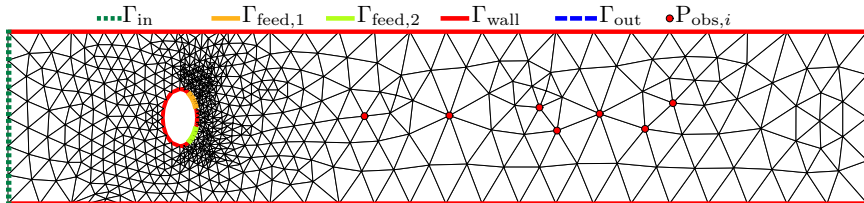
The entire process of the inexact low-rank KN-ADI method is depicted in Algorithm 3.

## 5. Numerical Experiments

We illustrate the benefits of Algorithm 3 to solve the GCARE associated with the solution of LQR problem (2.1, 2.2) governed by the linearized Navier-Stokes equation. Since our problem set-up is identical to that in the paper by Bänsch et al. [3] and in Weichelt's PhD Thesis [7], we only sketch it here and refer to [3, 7] for details. Additional numerical results can be found in [7].

The domain on which the Navier-Stokes and linearized Navier-Stokes equations are posed is shown in Figure 1. Inflow boundary conditions are posed on the left boundary, no-slip conditions are posed on part

Figure 1: Domain on which the linearized Navier-Stokes equation is posed and coarsest (level 1) triangulation



of the cylinder boundary and on the top and bottom boundary, and outflow conditions are imposed on the right boundary. Controls are applied on two segments on the cylinder wall (indicated by  $\Gamma_{\text{feed},1}$ ,  $\Gamma_{\text{feed},2}$ ). Specifically, for each segment a spatial profile is specified, so that the number of inputs in (2.2) is  $n_u = 2$ . As described in detail in [3, Sec. 2.7], [7, Sec. 4.1.3], an operator is constructed that converts these Dirichlet boundary controls to distributed controls, such that

$$B \in \mathbb{R}^{n_v \times 2}, \quad n_u = 2$$

in (2.2). The observations are chosen to be the vertical velocities of the linearized Navier-Stokes equations at the seven points indicated by  $P_{\text{obs},i}$ . Thus,  $\mathbf{y}(t) \in \mathbb{R}^7$ ,  $n_y = 7$ . Moreover, we penalize the output by  $\alpha > 0$ , i.e., the output equation (2.2c) takes the concrete form

$$\mathbf{y}(t) = \alpha C \mathbf{v}(t) \quad \text{with} \quad C \in \mathbb{R}^{7 \times n_v}$$

specified in [3, p. A855], [7, Sec. 4.4.1].

The solution to the steady state Navier-Stokes equation around which is linearized, as well as the linearized Navier-Stokes equations, i.e., the matrices in (2.1, 2.2) are computed using the finite element flow solver NAVIER [19], which uses  $\mathcal{P}_2$ - $\mathcal{P}_1$  Taylor-Hood elements and is written in FORTRAN90. The matrices in (2.1, 2.2) are generated using NAVIER and then stored using the so-called matrix market format [20].

Table 1: Finite element discretization levels and corresponding matrix sizes

Level	$n_v$	$n_p$
1	4,796	672
2	12,292	1,650
3	28,914	3,784
4	64,634	8,318
5	140,110	17,878
6	296,888	37,601

The computations for the resulting matrix equations are performed with MATLAB R2012b on a 64-bit CentOS 5.5 server with Intel Xeon X5650 at 2.67GHz, with 2 CPUs, 12 cores (6 cores per CPU), and 48 GB main memory available.

We conduct experiments with Reynolds number  $Re = 100, 200, 300, 400, 500$ , and we use six finite element discretization levels, with Level 1 being the coarsest (shown in Figure 1). The matrix sizes corresponding to these discretizations are listed in Table 1.

For larger Reynolds number, the matrix pencil  $(\mathbf{A}, \mathbf{M})$  is not stable (see [3, Fig. 2], [7, Sec. 4.2.3]) and a nonzero initial feedback is needed. We construct the initial feedback  $K^{(0)}$  as specified in [3, Sec. 2.7], [7, Sec. 4.2.3].

First, we illustrate the impact of the line search. Figure 2 shows the convergence of the ‘exact’ Kleinman-Newton method (i.e., the Lyapunov equation is solved with fixed high residual tolerance) and the inexact Kleinman-Newton method (Algorithm 1 with  $\eta_k = \min\{0.1, 0.9 \cdot \|\mathcal{R}(X^{(k)})\|_F\}$ ) both with and without line search for the LQR problem governed by the discretized linearized Navier-Stokes equations with  $Re = 500$ , output weight  $\alpha = 10^4$ , and discretization level 1. There is little difference in the Riccati residuals between the exact and the inexact Kleinman-Newton method. However, there is a big difference between the method with and without line search. Without line search the relative residual grows dramatically in the initial ( $k = 0$ ) iteration. With line search, the line search is active  $\xi_k < 1$  for iterations  $k = 0, 1, 2$  (exact Kleinman-Newton) and iterations  $k = 0, 1$  (inexact Kleinman-Newton). Figure 2 also shows the Riccati residuals corresponding to  $\xi_k = 1$  for the iterations where the line search is active. That the line search is typically only active in the first few iterations has also been observed in other applications of Riccati equations (see, e.g., [14]).

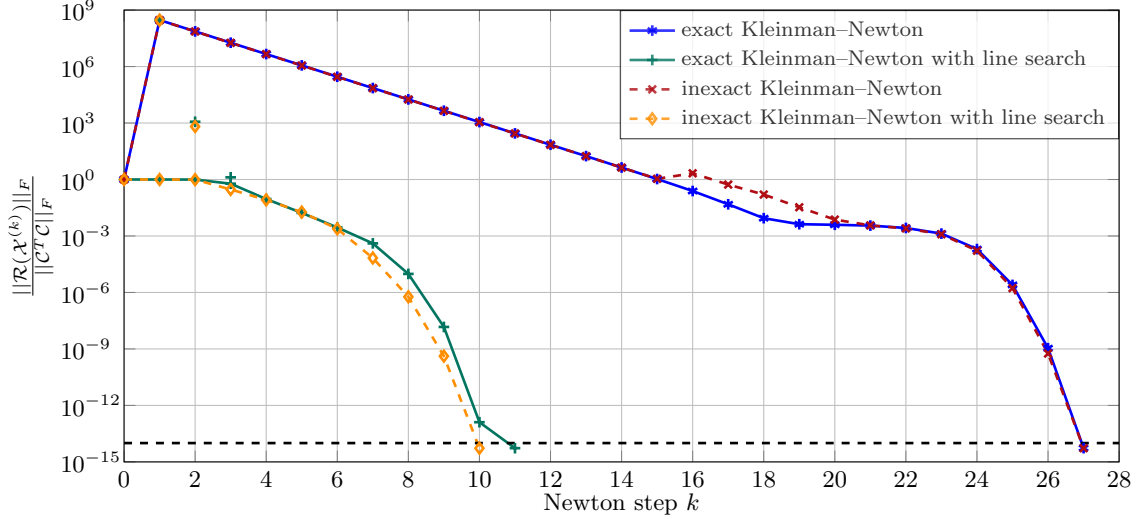
Next, we illustrate the influence of the various improvements to the overall performance of the Algorithm 3. Specifically we compare five set-ups, where ‘Setup i’ corresponds to a basic version of the Kleinman-Newton-ADI method, and ‘Setup v’ corresponds to the most efficient version, which is Algorithm 3. Setup i was used to compute the feedback controls in the paper by Bänsch et al. [3] without explicitly computing the projected residuals (cf., [7, p. 147]).

The set-ups are given as follows.

- i: Kleinman-Newton-ADI method, using the ‘classic’ low-rank ADI formulation, with fixed relative 2-norm Lyapunov residual tolerance  $tol_{\text{ADI}} = 10^{-10}$ . ADI shifts are computed heuristically as described in [21], requiring two short Arnoldi processes to approximate the large and small magnitude eigenvalues, i.e. several multiplications and solves with the pencil matrices (cf. [7, Sec. 2.2.3]). Lyapunov and Riccati residual norms are computed explicitly (cf. [7, Sec. 4.3.2]). This algorithm is detailed in [7, Sec. 4.2].
- ii: Kleinman-Newton-ADI method with fixed relative 2-norm Lyapunov residual tolerance  $tol_{\text{ADI}} = 10^{-8}$ , real-valued low-rank ADI, heuristic shifts, explicit computation of the projected Riccati residual norm. This explicit residual norm computation is not necessary, but demonstrates the accuracy of the low-rank Riccati residual.
- iii: Same setup as in ii, except that the low-rank Riccati residual updates are used as in (4.20). The Kleinman-Newton and the ADI iterations should be the same in Setup ii and Setup iii.



Figure 2: Impact of the line search on the convergence of the ‘exact’ and inexact Kleinman-Newton method for the problem with  $\text{Re} = 500$ , Level 1,  $\alpha = 10^4$ ,  $\text{tol}_{\text{Newton}} = 10^{-14}$ . There is little difference in the Riccati residuals between the exact and the inexact Kleinman-Newton method. Line search is active  $\xi_k < 1$  for iterations  $k = 0, 1$  and leads to a dramatic decrease in exact and inexact Kleinman-Newton iterations.



- iv: Same setup as in iii, except that the heuristic shifts are replaced by a modified version of the adaptive shifts in [22]. At most 15 ADI shifts are adaptively computed in each call. During the first call (in each Newton step), the projected pencil has  $n_y + n_u$  eigenvalues, since we are using the right hand side  $\widetilde{W}_0$  for projection. Those eigenvalues are passed into the `lp_mnmx` routine from [23] to determine  $r = \min\{15, n_y + n_u\}$  shifts.

After all shifts have been used we update the set. To this end, the blocks  $V_\ell$  are stored during the ADI iteration until all previously determined shifts have been used. The entire block  $Z_{\text{tmp}} = [V_1, \dots, V_r]$  is then used in the adaptive shift computation method. A thin QR-decomposition (using `qr(Z_tmp, 0)` in MATLAB<sup>®</sup>) is performed to determine the new projection basis and again upto 15 ADI shifts are determined via `lp_mnmx`.

- v: Algorithm 3 with  $\eta_k = \min\{0.1, 0.9 \cdot \|\mathcal{R}(X^{(k)})\|_F\}$ , adaptive shift selection in the ADI method and Armijo line search method. (Since the choice  $\eta_k = \min\{0.1, 0.9 \cdot \|\mathcal{R}(X^{(k)})\|_F\}$  of the forcing parameter leads to quadratic convergence of the inexact Kleinman-Newton method [1], this setup will also be referred to as ‘iKNqLS’ (inexact Kleinman Newton with quadratic forcing factor and Line Search).

For each setup, the detailed iteration numbers (the number of Newton iterations  $\#\text{Newt}$ , the number of ADI iterations  $\#\text{ADI}$ , and the number of Newton iterations where the line search was less than one  $\#\text{LS}$ ) and the various timings are depicted in Table 2. In Algorithm 3,  $k$  is the Newton iteration counter and  $\ell$  is the ADI iteration counter within a Newton iteration. Note that complex shifts appear as consecutive pairs for which we solve only one system (in Step 8 of Algorithm 3). Still, the ADI iteration counter is increased by two (in Steps 17 and 21 of Algorithm 3).

Comparing Setup i and Setup ii in Table 2 shows that incorporation of the real-valued ADI formulation in Setup ii reduces the number of linear solves ( $\#\text{lin\_solve}$ ) and, therefore, the time to solve these systems ( $\text{time}_{\text{lin\_solve}}$ ) drastically. Furthermore, the costs to compute the projected residuals are reduced by at least two magnitudes. Comparing Setup ii and Setup iii shows that avoiding the explicit computation of the projected residuals decreases the costs further, since the costs to evaluate the low-rank residuals are another magnitude smaller. The adaptive ADI shifts determination in Setup iv leads to another dramatic improvement in overall performance. These adaptive shifts reduce the number of ADI iterations and linear

Table 2: Performance of the various Kleinman-Newton-ADI methods. Iteration numbers and timings in seconds for the different Kleinman-Newton-ADI methods specified in Setup i to iv. applied to the problems with  $Re = 500$ , refinement level 1,  $tol_{Newton} = 10^{-8}$ , and  $\alpha = 10^0$ .

	#KN	#ADI	#lin_solve	#LS	time <sub>lin_solve</sub>	time <sub>shift</sub>	time <sub>proj-res</sub>	time <sub>total</sub>
i	8	3067	3067	–	$1.4 \cdot 10^3$	$3.6 \cdot 10^1$	$5.4 \cdot 10^3$	$6.8 \cdot 10^3$
ii	8	3031	1721	–	$7.0 \cdot 10^2$	$3.6 \cdot 10^1$	$1.0 \cdot 10^1$	$7.5 \cdot 10^2$
iii	8	3031	1721	–	$7.0 \cdot 10^2$	$3.7 \cdot 10^1$	–	$7.4 \cdot 10^2$
iv	8	600	346	–	$1.4 \cdot 10^2$	$2.8 \cdot 10^0$	–	$1.5 \cdot 10^2$
v	7	305	176	1	$7.3 \cdot 10^1$	$1.9 \cdot 10^0$	–	$7.5 \cdot 10^1$

solves by a factor of five. Additionally, the computation of these adaptive ADI shifts is one magnitude less expensive than the heuristic shift computation.

Finally, adding the line-search in Setup v improves the method further. The number of ADI iterations and linear system solves is reduced by a factor of two. The reduction in ADI iterations also reduced the time for the shift computation. The line search is less than one only in the first iteration and the cost of step size computation is negligible. Comparing the total computation times shows that the algorithm specified in Setup v is 90-times faster than the algorithm specified in Setup i. As we will see next, the solution of the Riccati equation becomes more difficult as the output weighting  $\alpha$  increases. In those cases the speedup of Setup v over Setup i is even more important.

The following numerical tests focus on Algorithm 3, with Setup v. As mentioned before, Algorithm 3 with Setup v will be referred to as ‘iKNqLS’. Table 3 documents the performance of iKNqLS applied to our test problem as Reynolds number  $Re$ , output weight  $\alpha$ , and discretization level change. Table 3a shows that the number of Kleinman-Newton iterations increases moderately with an increasing  $\alpha$  and increasing Reynolds number. Similarly, the number of total ADI iterations needed to approximately solve the Lyapunov equations increases with an increasing  $\alpha$  and increasing Reynolds number. Furthermore, line search is only necessary for higher Reynolds numbers and higher output weights.

Table 3b shows that for  $Re \leq 300$ , the number of Newton iterations remains nearly constant as the refinement level is increased. For  $Re \leq 300$  and refinement level greater than two the number of iterations where the step size is less than one is unusually large. We believe that this effect is a result of the instability of the matrix pencil. Solving the first Newton step inexactly might yield an intermediate solution that is slightly (especially in finite precision arithmetic) not stabilizing. Therefore, the following ADI iteration tends to diverge. Our algorithm detects this behavior by monitoring the Riccati and Lyapunov residual continuously. Although this behavior is not covered by the convergence proof in Theorem 6, where a stabilizing solution for  $k \geq k_0$  is required, our algorithm handles this situation by deleting the last ADI step and performing a line search. This yields convergence in all examples we considered. The relative Riccati residual seems to stagnate for a couple of iterations and, hence, an increasing amount of line search runs is required.

Convergence theory for the exact Kleinman-Newton method guarantees that the matrix pencils are stable if the initial matrix pencil is stable. Thus, another approach to circumvent the appearance of a possibly unstable pencil arising from inexact Lyapunov equation solution is to use a smaller fixed ADI tolerance for the first Newton step. Rather than using the Lyapunov residual tolerance  $tol_{ADI} = \eta_k \|\mathcal{R}(X^{(k+1)})\|$ , we set  $tol_{ADI} = 10^{-2}$  for the first two Newton iterations. If the relative Riccati residual decreases and drops below  $5 \cdot 10^{-1}$ , the method switches to the iKNqLS scheme (i.e.,  $tol_{ADI} = \eta_k \|\mathcal{R}(X^{(k+1)})\|$ ). This is referred to as ‘exact’ start. Using the  $tol_{ADI} = \eta_k \|\mathcal{R}(X^{(k+1)})\|$  in all iterations is referred to as inexact start. Table 4 compares both starting procedures for  $Re \geq 300$  and refinements Level 3–6. The ‘exact’ start prevents the stagnation of the relative Riccati residual and reduces the number of Newton iterations. The line search is used in at most one iteration. However, the ‘exact’ solves in the first Newton iterations increase the number of ADI iterations. Therefore, in most cases a decrease in the number of Newton iterations does not translate

Table 3: Number of Kleinman-Newton iterations ( $\#KN$ ), ADI iterations ( $\#ADI$ ), and iterations in which line search was active ( $\#LS$ ) during the ‘iKNqLS’ process ( $tol_{\text{Newton}} = 10^{-8}$ ,  $\eta_k = \min\{0.1, 0.9 \cdot \|\mathcal{R}(X^{(k)})\|_F\}$ , Armijo method).

(a) Influence of output weighting  $\alpha$  during the ‘iKNqLS’ process (refinement: Level 1).

$\alpha$ \ Re	100			200			300			400			500		
	#KN	#ADI	#LS	#KN	#ADI	#LS	#KN	#ADI	#LS	#KN	#ADI	#LS	#KN	#ADI	#LS
$10^{-2}$	3	38	–	4	74	–	4	73	–	4	87	–	5	79	–
$10^{-1}$	4	53	–	5	109	–	5	84	–	4	74	–	5	109	–
$10^0$	5	80	–	6	118	–	7	119	–	6	115	1	7	176	1
$10^1$	7	98	–	7	134	–	8	153	1	10	212	2	9	201	2
$10^2$	7	109	–	9	199	1	12	296	3	12	331	3	12	340	4

(b) Influence of refinement levels during the ‘iKNqLS’ process ( $\alpha = 1$ ).

Re \ Level	100			200			300			400			500		
	#KN	#ADI	#LS	#KN	#ADI	#LS	#KN	#ADI	#LS	#KN	#ADI	#LS	#KN	#ADI	#LS
Level 1	5	80	–	6	118	–	7	119	–	6	115	1	7	176	1
Level 2	4	73	–	6	118	1	7	144	1	7	148	1	7	168	1
Level 3	5	99	–	5	124	–	10	221	3	8	200	2	7	183	–
Level 4	4	72	–	6	176	1	11	198	6	10	199	5	10	243	3
Level 5	5	126	–	6	160	1	11	244	4	11	273	4	10	267	3
Level 6	6	189	–	6	184	1	11	280	4	11	279	4	13	344	6

into a significant decrease in the total number of ADI iterations (and therefore significant decrease in overall computing time) when the ‘exact’ start is used.

Overall iKNqLS is able to solve the Riccati equation in all cases. Although there is no theoretical justification, our numerics indicate that the inclusion of line search and computationally inexpensive monitoring of the low-rank Riccati and Lyapunov residuals enables the algorithm to successfully cope with intermediate iterates that are nearly not stabilizing.

## 6. Conclusions

We have extended our inexact Kleinman-Newton method low-rank ADI solver and line search from [1] to Riccati equations governed by Hessenberg index-2 DAEs. Using the projection idea from Heinkenschloss et al. [2] and Bänsch, Benner [8] we transform the problem governed by the DAE into a ‘classical’ problem governed by an ODE. Our algorithm in [1] is then applied to this transformed problem. However, the projected ODE is never computed in practice. Instead, a careful exploitation of the problem structure allows the formulation of the algorithm in the original DAE context. We have demonstrated the performance of our Riccati solver to a problem arising in feedback stabilization of Navier-Stokes flow around a cylinder. The numerical results document the impact of various algorithmic components on the overall performance. The algorithmic improvements in this paper lead to approximately 90-times speed-up over a previously used Kleinman-Newton-ADI method. Moreover, we have explored the performance of the new algorithm for various Reynolds numbers, mesh refinement levels and output weights. The new algorithm was able to solve all instances. Moreover, although there is no theoretical justification, our numerics indicate that the inclusion of line search and computationally inexpensive monitoring of the low-rank Riccati and Lyapunov residuals enables the new algorithm to successfully cope with intermediate iterates that are (slightly) not stabilizing.

Table 4: Comparison of ‘exact’ and inexact start  
 $(tol_{\text{Newton}} = 10^{-8}, \eta_k = \min\{0.1, 0.9 \cdot \|\mathcal{R}(X^{(k)})\|_F\})$ , Armijo method,  $\alpha = 1$ ).

		start inexact				start "exact" with $tol_{\text{ADI}} = 10^{-2}$			
		#KN	#ADI	#LS	time <sub>total</sub>	#KN	#ADI	#LS	time <sub>total</sub>
Re = 300									
Level	3	10	221	3	$7.2 \cdot 10^2$	8	186	1	$5.9 \cdot 10^2$
	4	11	198	6	$1.6 \cdot 10^3$	8	177	0	$1.4 \cdot 10^3$
	5	11	244	4	$4.8 \cdot 10^3$	8	215	0	$4.1 \cdot 10^3$
	6	11	280	4	$1.2 \cdot 10^4$	9	259	0	$1.2 \cdot 10^4$
Re = 400									
Level	3	8	200	2	$6.1 \cdot 10^2$	6	158	1	$5.2 \cdot 10^2$
	4	10	199	5	$1.5 \cdot 10^3$	7	197	1	$1.6 \cdot 10^3$
	5	11	273	4	$5.4 \cdot 10^3$	8	244	1	$4.6 \cdot 10^3$
	6	11	279	4	$1.3 \cdot 10^4$	8	272	1	$1.3 \cdot 10^4$
Re = 500									
Level	3	7	183	0	$6.2 \cdot 10^2$	7	179	1	$6.0 \cdot 10^2$
	4	10	243	3	$2.0 \cdot 10^3$	8	192	1	$1.6 \cdot 10^3$
	5	10	267	3	$5.5 \cdot 10^3$	9	261	1	$5.5 \cdot 10^3$
	6	13	344	6	$1.6 \cdot 10^4$	7	248	1	$1.2 \cdot 10^4$

## References

- [1] P. Benner, M. Heinkenschloss, J. Saak, H. K. Weichelt, An inexact low-rank Newton-ADI method for large-scale algebraic Riccati equations, *Appl. Numer. Math.* 108 (2016) 125–142. [doi:10.1016/j.apnum.2016.05.006](https://doi.org/10.1016/j.apnum.2016.05.006).
- [2] M. Heinkenschloss, D. C. Sorensen, K. Sun, Balanced truncation model reduction for a class of descriptor systems with application to the Oseen equations, *SIAM J. Sci. Comput.* 30 (2) (2008) 1038–1063. [doi:10.1137/070681910](https://doi.org/10.1137/070681910).
- [3] E. Bänsch, P. Benner, J. Saak, H. K. Weichelt, Riccati-based boundary feedback stabilization of incompressible Navier-Stokes flows, *SIAM J. Sci. Comput.* 37 (2) (2015) A832–A858. [doi:10.1137/140980016](https://doi.org/10.1137/140980016).
- [4] P. Benner, T. Stykel, Numerical solution of projected algebraic Riccati equations, *SIAM J. Numer. Anal.* 52 (2) (2014) 581–600. [doi:10.1137/130923993](https://doi.org/10.1137/130923993).
- [5] V. Simoncini, Analysis of the rational Krylov subspace projection method for large-scale algebraic Riccati equations, *SIAM J. Matrix Anal. Appl.* 37 (4) (2016) 1655–1674. [doi:10.1137/16M1059382](https://doi.org/10.1137/16M1059382).
- [6] V. Simoncini, D. B. Szyld, M. Monsalve, On two numerical methods for the solution of large-scale algebraic Riccati equations, *IMA J. Numer. Anal.* 34 (3) (2014) 904–920. [doi:10.1093/imanum/drt015](https://doi.org/10.1093/imanum/drt015).
- [7] H. K. Weichelt, Numerical aspects of flow stabilization by Riccati feedback, Dissertation, Otto-von-Guericke-Universität (Jan. 2016). URL <http://nbn-resolving.de/urn:nbn:de:gbv:ma9:1-8693>
- [8] E. Bänsch, P. Benner, Stabilization of incompressible flow problems by Riccati-based feedback, in: G. Leugering, S. Engell, A. Griewank, M. Hinze, R. Rannacher, V. Schulz, M. Ulbrich, S. Ulbrich (Eds.), *Constrained Optimization and Optimal Control for Partial Differential Equations*, Vol. 160 of *Internat. Ser. Numer. Math.* 160, Birkhäuser, Basel, 2012, pp. 5–20. [doi:10.1007/978-3-0348-0133-1](https://doi.org/10.1007/978-3-0348-0133-1).
- [9] J.-P. Raymond, Feedback boundary stabilization of the two-dimensional Navier-Stokes equations, *SIAM J. Cont. Optim.* 45 (3) (2006) 790–828. [doi:10.1137/050628726](https://doi.org/10.1137/050628726).
- [10] E. Bänsch, P. Benner, J. Saak, H. K. Weichelt, Optimal control-based feedback stabilization of multi-field flow problems, in: G. Leugering, P. Benner, S. Engell, A. Griewank, H. Harbrecht, M. Hinze, R. Rannacher, S. Ulbrich, G. Leugering, P. Benner, S. Engell, A. Griewank, H. Harbrecht, M. Hinze, R. Rannacher, S. Ulbrich (Eds.), *Trends in PDE Constrained Optimization*, Vol. 165 of *Internat. Ser. Numer. Math.* 165, Birkhäuser, 2014, pp. 173–188. [doi:10.1007/978-3-319-05083-6\\_11](https://doi.org/10.1007/978-3-319-05083-6_11).
- [11] P. Lancaster, L. Rodman, *Algebraic Riccati equations*, Oxford Science Publications, The Clarendon Press, Oxford University Press, New York, 1995.

- [12] D. L. Kleinman, On an iterative technique for Riccati equation computations, *IEEE Trans. Automat. Control* 13 (1) (1968) 114–115. doi:10.1109/TAC.1968.1098829.
- [13] F. Feitzinger, T. Hylla, E. W. Sachs, Inexact Kleinman-Newton method for Riccati equations, *SIAM J. Matrix Anal. Appl.* 31 (2) (2009) 272–288. doi:10.1137/070700978.
- [14] P. Benner, R. Byers, An exact line search method for solving generalized continuous-time algebraic Riccati equations, *IEEE Trans. Automat. Control* 43 (1) (1998) 101–107. doi:10.1109/9.654908.
- [15] P. Benner, J.-R. Li, T. Penzl, Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems, *Numer. Lin. Alg. Appl.* 15 (9) (2008) 755–777. doi:10.1002/nla.622.
- [16] J.-R. Li, J. White, Low rank solution of Lyapunov equations, *SIAM J. Matrix Anal. Appl.* 24 (1) (2002) 260–280. doi:10.1137/S0895479801384937.
- [17] P. Benner, P. Kürschner, J. Saak, A reformulated low-rank ADI iteration with explicit residual factors, *Proc. Appl. Math. Mech.* 13 (1) (2013) 585–586. doi:10.1002/pamm.201310273.
- [18] P. Kürschner, Efficient low-rank solution of large-scale matrix equations, Dissertation, Otto-von-Guericke-Universität, (Apr. 2016). URL <http://hdl.handle.net/11858/00-001M-0000-0029-CE18-2>
- [19] E. Bänsch, Simulation of instationary, incompressible flows, in: *Proceedings of the Algoritmy'97 Conference on Scientific Computing (Zuberec)*, Vol. 67, 1998, pp. 101–114.
- [20] R. F. Boisvert, R. Pozo, K. A. Remington, The Matrix Market Exchange Formats: Initial Design, NIST Interim Report 5935, National Institute of Standards and Technology (Dec. 1996). URL <http://math.nist.gov/MatrixMarket/reports/MMformat.ps> (accessed Jan. 19, 2018).
- [21] T. Penzl, LYAPACK Users Guide, Tech. Rep. SFB393/00-33, Sonderforschungsbereich 393 *Numerische Simulation auf massiv parallelen Rechnern*, TU Chemnitz, 09107 Chemnitz, Germany, available from <http://www.tu-chemnitz.de/sfb393/sfb00pr.html> (accessed Jan. 19, 2018). (2000).
- [22] P. Benner, P. Kürschner, J. Saak, Self-generating and efficient shift parameters in ADI methods for large Lyapunov and Sylvester equations, *Electron. Trans. Numer. Anal.* 43 (2014) 142–162. URL <http://etna.mcs.kent.edu/volumes/2011-2020/vol143/abstract.php?vol=43&pages=142-162>
- [23] T. Penzl, Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case, *Syst. Cont. Lett.* 40 (2000) 139–144. doi:10.1016/S0167-6911(00)00010-4.