Max Planck Institute Magdeburg
Preprints

Peter Benner Patrick Kürschner Jens Saak

Low-Rank Newton-ADI Methods for Large Nonsymmetric Algebraic Riccati Equations

MPIMD/14-21 **November 20, 2014**

Abstract

The numerical treatment of large-scale, nonsymmetric algebraic Riccati equations (NARE) by a low-rank variant of Newton's method is considered. We discuss a method to compute approximations to the solution of the NARE in a factorized form of low rank. The occurring large-scale Sylvester equations are dealt with using the factored alternating direction implicit iteration (fADI). Several performance enhancing strategies available for the factored ADI as well as the related Newton-ADI for symmetric algebraic Riccati equations are generalized to this combination. This includes the efficient computation of the norm of the residual matrix, adapted shift parameters strategies for fADI, and an acceleration of the Newton's scheme by means of a Galerkin projection. Numerical experiments illustrate the capabilities of the proposed method to solve high-dimensional NAREs.

Imprint:

Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg

Publisher:
Max Planck Institute for
Dynamics of Complex Technical Systems

Address:
Max Planck Institute for
Dynamics of Complex Technical Systems
Sandtorstr. 1
39106 Magdeburg

http://www.mpi-magdeburg.mpg.de/preprints/

1 Introduction

We consider non-symmetric algebraic Riccati equations (NARE) of the form

$$\mathcal{R}(X) = FG^T + AX + XB - XPQ^TX = 0 \tag{1}$$

with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$, $F \in \mathbb{R}^{n\times r}$, $G \in \mathbb{R}^{m\times r}$, $P \in \mathbb{R}^{m\times p}$, $Q \in \mathbb{R}^{n\times p}$, and the sought solution $X \in \mathbb{R}^{n\times m}$. If $B = A^T$, $G = F$ and $Q = P$, (1) becomes a continuous-time algebraic Riccati equation (CARE). NAREs arise, e.g., in fluid queue models [35], in the numerical treatment of transport equations [29], in the study of open loop Nash games [1], and in methods to compute or refine invariant subspaces of matrices or pencils [16]. Often, the block matrix

$$\mathcal{M} := \begin{bmatrix} B & -PQ^T \\ FG^T & A \end{bmatrix} \in \mathbb{R}^{m+n\times m+n}$$

is an M-matrix and, hence, the associated NAREs are typically referred to as M-NAREs. These were recently subject of extensive research in both theoretical and computational aspects, see, e.g., [13, 29, 24, 22, 33] and the references therein. One is typically interested in the minimal, nonnegative solution $X^{(\min)} \geq 0$ which satisfies $X^{(\min)} \leq X$ for all possible solutions X of (1). Here and from now on, $\leq$, $\geq$ refer to the element wise partial orderings. This minimal, nonnegative solution always exists if $\mathcal{M}$ is a nonsingular, or an irreducible, singular M-matrix [13, Theorem 2.9].

Here we assume that A, B are large and sparse matrices but the number of columns in F, G, P, Q is much smaller than the dimension of (1), i.e., r, $p \ll \min(n, m)$. This will enable the approximation of X by a low-rank matrix X_h with $\operatorname{rank}(X_h) = h \ll \min(n, m)$ such that also large-scale NAREs can be dealt with.

The main purpose of this article is, without using M-matrix properties, to present a Newton style method for large-scale NAREs which computes such a low-rank approximation X_h. The involved Sylvester equations are solved by the factored alternating directions implicit iteration (fADI) [9]. The remainder of the article is structured as follows: in Section 2 we briefly review Newton's method for NAREs and also consider an analogue to the Newton-Kleinman method for CAREs. The fADI algorithm for computing low-rank solutions of the occurring large-scale Sylvester equations is topic of Section 3. To this end, recent improvements of fADI [6] are included and several performance enhancing strategies which are known for the Newton-ADI for large-scale CAREs [8] are also adopted. We also give a new result regarding the structure of the NARE residual matrix and introduce an adapted shift parameter strategy for fADI. Some modification for dealing with NAREs of special structure or generalized versions of (1) are also given. The performance of the developed low-rank Newton-ADI method is evaluated in numerical experiments in Section 4. Section 5 summarizes and proposes some related future research topics.

Algorithm 1: Newton's Method for (1)

Input : A, B, F, G, P, Q as in (1), initial guess $X^{(0)}$.
Output: Approximate solution X.

for $k = 1, \ldots, k_{\max}$ **do**
  $\mathcal{R}(X^{(k-1)}) = FG^T + AX^{(k-1)} + X^{(k-1)}B - X^{(k-1)}PQ^TX^{(k-1)}$
  Solve the Sylvester equation

$$(A - X^{(k-1)}PQ^T)N^{(k)} + N^{(k)}(B - PQ^TX^{(k-1)}) = -\mathcal{R}(X^{(k-1)}) \tag{2}$$

  for $N^{(k)}$ and set $X^{(k)} = X^{(k-1)} + N^{(k)}$.

2 Newton Methods for NAREs

A Newton scheme for (1) is given by

$$X^{(k)} = X^{(k-1)} + N^{(k)}, \quad N^{(k)} := -(\mathcal{R}'_{X^{(k-1)}})^{-1}\mathcal{R}(X^{(k-1)}),$$

where $\mathcal{R}'_X$ is the Fréchet derivative of $\mathcal{R}$ evaluated at X. The increment $N^{(k)}$ is the solution of the linear matrix equation

$$\mathcal{R}'_{X^{(k-1)}}[N^{(k)}] = -\mathcal{R}(X^{(k-1)})$$

which turns out to be a Sylvester equation. The resulting iteration is given in Algorithm 1, see, e.g. [13, Listing 3.11]. It is shown in [24, 22] that if $\mathcal{M}$ is a nonsingular or irreducible, singular M-Matrix, then Algorithm 1 initialized with $X^{(0)} = 0$ produces a sequence of nonnegative matrices $X^{(0)} \leq X^{(1)} \leq \ldots \leq X^{(\min)}$ that converges to the minimal nonnegative solution $X^{(\min)}$. The convergence is quadratic provided $\mathcal{M}$ is a nonsingular M-matrix. For singular and irreducible M-matrices the convergence speed might in some cases only be linear but there are approaches to cure this: one can either work on a shifted NARE instead or start the Newton iteration with an appropriate initial guess X_0 [13].

Similar to the derivation of the Newton-Kleinman and Newton-Hewer method [30, 25] for CAREs and, respectively, discrete-time AREs (DAREs), we insert $N^{(k)} = X^{(k)} - X^{(k-1)}$ into (2) in Algorithm 1. This yields a Newton-Kleinman variant for NAREs which is illustrated in Algorithm 2. There, we introduced the matrices $K^{(k)} := X^{(k)}P \in \mathbb{R}^{n\times p}$, $L^{(k)} := {X^{(k)}}^T Q \in \mathbb{R}^{m\times p}$ which might be considered as complement to the feedback matrices in the CARE and DARE cases. If $K^{(0)} = X^{(0)}P$ and $L^{(0)} = (X^{(0)})^TQ$, the Algorithms 1 and 2 are mathematically equivalent and produce the same results.

The main computational work for both algorithms is the solution of the Sylvester equation in each step of the iteration. For small to moderate sized problems, methods based on eigenvalue, Schur, or Hessenberg decompositions of A, B [2, 39, 21], or the sign function iteration [36, 10] can be applied. Due to their in general cubic complexity and quadratic storage requirements, their application to large-scale problems is not

Algorithm 2: Newton-Kleinman Method for (1)

Input : $A,\ B,\ F,\ G,\ P,\ Q$ as in (1), initial guesses $K^{(0)},\ L^{(0)}$.
Output: Approximate solution X.

1 **for** $k = 1, \ldots, k_{\max}$ **do**
2 $\quad F^{(k)} := [F, K^{(k-1)}],\ G^{(k)} := [G, L^{(k-1)}]$.
3 $\quad$ Solve the Sylvester equation

$$(A - K^{(k-1)}Q^T)X^{(k)} + X^{(k)}(B - P(L^{(k-1)})^T) = -F^{(k)}(G^{(k)})^T \tag{3}$$

$\quad$ for $X^{(k)}$.
4 $\quad$ Set $K^{(k)} := X^{(k)}P,\ L^{(k)} := (X^{(k)})^T Q$.

feasible. The Sylvester equations in (2) and (3) are defined by the same coefficients but different right hand sides. It holds rank $\left(F^{(k)}\ G^{(k)\,T}\right) \leq r + p \ll n$ in (3) which enables us to employ low-rank solvers for Sylvester equations as first suggested in [3]. In contrast, the right hand sides in the original Newton scheme in Algorithm 1 do not share this property in general. In the following we therefore develop and investigate a low-rank version of Algorithm 2, where occurring large Sylvester equations are dealt with by the factored alternating directions implicit (ADI) iteration [9, 6]. This combination, low-rank variant of ADI within a Newton iteration, has been successfully applied for CAREs [8, 37, 27, 18] as well as DAREs [5].

3 Low-rank Newton-ADI for NAREs

Before we state our low-rank version of Algorithm 2 to deal with large-scale NAREs, we concisely review the low-rank variant of the ADI iteration for Sylvester equations as well as some recent improvements of it.

3.1 The Factored ADI Method for Large-Scale Sylvester Equations

Here we consider Sylvester equations

$$AX + XB + FG^T = 0 \tag{4}$$

with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$, $F \in \mathbb{R}^{n\times r}$, $G \in \mathbb{R}^{m\times r}$. We assume that the spectra of A, B satisfy $\Lambda(A) \cap \Lambda(-B) = \emptyset$ to ensure the existence of a unique solution. The alternating directions implicit iteration [40, 42] for (4) is given by the two-step scheme

$$\begin{aligned} (A + \beta_j I_n)X_{j+\frac{1}{2}} &= -X_j(B + \beta_j I_m) - FG^T, \\ X_{j+1}(B + \alpha_j I_n) &= -(A + \alpha_j I_n)X_{j+\frac{1}{2}} + FG^T, \end{aligned} \tag{5}$$

where $\alpha_i \notin \Lambda(B)$, $\beta_i \notin \Lambda(A)$, $\beta_i \neq -\alpha_i$, $i \geq 1$ are shift parameters that steer the convergence speed of (5). Setting $X_0 = 0$ and rewriting (5) into a single-step scheme leads, as thoroughly described in [9], to the factored ADI iteration (fADI)

$$V_1 = (A + \beta_1 I_n)^{-1} F, \quad W_1 = (B + \alpha_1 I_m)^{-H} G, \tag{6a}$$

$$V_j = V_{j-1} - (\beta_j + \alpha_{j-1})(A + \beta_j I_n)^{-1} V_{j-1}, \tag{6b}$$

$$W_j = W_{j-1} - \overline{(\alpha_j + \beta_{j-1})}(B + \alpha_j I_n)^{-H} W_{j-1} \quad \text{for} \quad j > 1. \tag{6c}$$

It computes a low-rank approximation of X in a factored form $X_j = Z_j \Gamma_j Y_j^H$, where the factors are in the course of (6) build via

$$\begin{aligned} Z_j &= [Z_{j-1},\ V_j] \in \mathbb{C}^{n \times rj}, \quad Y_j = [Y_{j-1},\ W_j] \in \mathbb{C}^{m \times rj}, \\ \Gamma_j &= \operatorname{diag}(\Gamma_{j-1}, -(\beta_j + \alpha_j) I_r) \in \mathbb{C}^{rj \times rj} \end{aligned}$$

with Z_0, Γ_0, Y_0 set as empty arrays. The above iteration (6) is particularly efficient if $r \ll \min(n, m)$ because only r new columns have to be processed in each iteration step. The main computational effort is caused by the solution of two shifted linear systems with r right hands sides. In [6] it is shown that the residual at iteration step j is given as

$$\begin{aligned} \mathcal{S}(X_j) &= A X_j \Gamma_j (Y_j)^H + X_j \Gamma_j (Y_j)^H B + F G^T = S_j T_j^H, \\ S_j &= S_{j-1} - (\beta_j + \alpha_j) V_j \in \mathbb{C}^{n \times r}, \\ T_j &= T_{j-1} - \overline{(\beta_j + \alpha_j)} W_j \in \mathbb{C}^{m \times r}. \end{aligned}$$

Hence, although $\mathcal{S}(X_j)$ is in general a dense, $n \times m$ matrix, it is of rank at most r. By exploiting the above low-rank factorization of $\mathcal{S}(X_j)$, its norm can be efficiently computed via

$$\begin{aligned} \|\mathcal{S}(X_j)\| &= \|S_j T_j^H\| = \sigma_{\max}(S_j T_j^H) \\ &= \sqrt{\lambda_{\max}(T_j S_j^H S_j T_j^H)} = \sqrt{\lambda_{\max}((S_j^H S_j)(T_j^H T_j))}. \end{aligned} \tag{7}$$

For this relation we used the well known property that the nonzero eigenvalues of $\mathcal{AB}$ are the same as those of $\mathcal{BA}$ for all $\mathcal{A} \in \mathbb{C}^{n \times m}$, $\mathcal{B} \in \mathbb{C}^{m \times n}$, see, e.g., [26, Theorem 1.32]. Hence, computing the residual norm essentially requires the computation of the largest eigenvalue of the $r \times r$ matrix $(S_j^H S_j)(T_j^H T_j)$. Although the eigenvalues of this matrix are obviously real, roundoff errors might introduce very small, spurious imaginary parts such that it is wise to take the square root only over the real part of the computed largest eigenvalue. This strategy is slightly more efficient than using a QR factorization of either S_j or T_j as in [6]. With $S_0 := F$, $T_0 := G$, the residual factors S_j, T_j can be explicitly inserted into the low-rank iteration (6) which yields the modified version of the fADI method illustrated in Algorithm 3. In Line 2 it uses the scaled residual norm as stopping criterion via $\|\mathcal{S}(X_j)\| \leq \tau_{\text{ADI}} \|F G^T\|$.

Algorithm 3: Factored ADI iteration for (4)

Input : $A,\ B,\ F,\ G$ as in (4), shift parameters $\{\alpha_1, \ldots, \alpha_{j_{\max}}\}$, $\{\beta_1, \ldots, \beta_{j_{\max}}\}$, and stopping tolerance $\tau_{\text{ADI}} \ll 1$.

Output: $Z_j \in \mathbb{C}^{n\times rj}$, $Y_j \in \mathbb{C}^{m\times rj}$, $\Gamma_j \in \mathbb{C}^{rj\times rj}$ such that $Z_j\Gamma_j Y_j^H \approx X$.

$S_0 := F,\ T_0 := G,\ Z_0 = \Gamma_0 = Y_0 = [\,],\ j = 0.$

while $\|S_j T_j^T\| \geq \tau_{ADI}\|FG^T\|$ **do**

  $j = j + 1.$

  $V_j = (A + \beta_j I_n)^{-1} S_{j-1},\ W_j = (B + \alpha_j I_m)^{-H} T_{j-1}.$

  $S_j = S_{j-1} + \gamma_j V_j,\ T_j = T_{j-1} + \overline{\gamma_j} W_j,\ \gamma_j := -(\beta_j + \alpha_j).$

  Update the low-rank solution factors

$$Z_j = [Z_{j-1}, V_j],\ Y_j = [Y_{j-1}, W_j],\ \Gamma_j = \operatorname{diag}(\Gamma_{j-1}, \gamma_j I_r)\,.$$

Shift Parameters The shift parameters of (5), (6), and Algorithm 3 can be related to the rational optimization problem

$$\min_{\alpha_j,\beta_j\in\mathbb{C}} \left(\max_{\substack{1\leq \ell\leq n\\ 1\leq k\leq m}} \prod_{j=1}^{J} \left| \frac{(\lambda_\ell - \alpha_j)(\mu_k - \beta_j)}{(\lambda_\ell + \beta_j)(\mu_k + \alpha_j)} \right| \right), \quad \lambda_\ell \in \Lambda(A),\ \mu_k \in \Lambda(B), \tag{8}$$

see [41]. For large-scale Sylvester equations this problem is hard to solve as the involved spectra $\Lambda(A),\ \Lambda(B)$ are not efficiently available. Some strategies to compute a fixed number of optimal, approximate shifts a priori, i.e., before the actual iteration, can be found in [42, 38, 9]. One approach which we will employ in our numerical experiments is the heuristic strategy proposed in [32, 9]. The spectra $\Lambda(A)$, $\Lambda(B)$ in (8) are replaced by much smaller sets consisting of a fixed number $J \ll \min(n, m)$ of approximate eigenvalues of A, B. The approximations are typically chosen as k_+ and k_- Ritz-, and inverse Ritz values obtained from Arnoldi processes w.r.t A, B and, respectively A^{-1}, B^{-1}. These sets of Ritz values alone often provide good α- and β-shifts, [6]. Alternatively, one can use the Ritz values to solve (8) in an approximate way to get $k_+ + k_-$ α- and $k_+ + k_-$ β-shifts as illustrated, e.g, in [9, Algorithm 2]. Generating these heuristic shift parameters introduces additional costs because of the involved matrix vector products and linear systems solves with A and B. Also, there is no known rule for adjusting the setup parameters $J,\ k_+,\ k_-$ to get optimal results as it is criticized in [7]. Even slight changes in these values can yield a noticeable different convergence behavior of fADI.

As second shift strategy we will use the novel approach proposed in [7] for efficiently and automatically computing shifts in the course of the iteration. The main idea is to generate, at inner iteration step j, orthogonal matrices $U_A \in \mathbb{C}^{m\times r}$ and $U_B \in \mathbb{C}^{n\times r}$ corresponding to V_j and, respectively, W_j. Then, the shifts for the iteration steps $j+1$ to $j + r$ are taken as the eigenvalues of the projected matrices $U_A^H A U_A$ and $U_B^H B U_B$.

After these r iteration steps, the generation is repeated with V_{j+r} and W_{j+r}. These self-generating shifts are in the remainder called A-B-shifts. They are, obviously, very cheap to compute, because only a small number of matrix vector multiplications with the original large matrices A, B, but no linear system solves, are required.

Moreover, another big advantage is that the construction is completely automatic, since no setup parameters are involved. These A-B-shifts often outperform the heuristic and other approaches, especially for problems with complex spectra as reported in [7].

Generating real solution factors in the presence of complex shifts The shifts can, regardless of the method used for their generation, be complex numbers, especially if at least one of the spectra $\Lambda(A)$, $\Lambda(B)$ contains complex eigenvalues. Then, Algorithm 3 will produce complex iterates and consequently also complex low-rank solution factors. Since this increases the computation cost and memory requirements of the fADI method, a further modification of the algorithm was introduced in [6, Algorithm 2] to circumvent this issue. Under the assumption that complex shifts always occur in complex conjugated pairs, the modified fADI iteration generates real low-rank solution factors at a significantly reduced amount of complex arithmetic operations. Since the involved formulas are rather long and complicated, but not important for the remainder, for the sake of brevity we keep the simpler representation given in Algorithm 3. However, in our numerical examples, whenever the fADI method is applied to solve (4), this should be understood as the usage of the modified version [6, Algorithm 2] such that real, low-rank solution factors $Z_j \in \mathbb{R}^{n\times rj}$, $Y_j \in \mathbb{R}^{m\times rj}$, $\Gamma_j \in \mathbb{R}^{rj\times rj}$ are efficiently obtained in the end.

3.2 Combination of factored ADI and the Newton-Kleinman method for NAREs

Employing Algorithm 3 to solve the Sylvester equation in Line 3 of the Newton-Kleinman scheme (Algorithm 2) for NAREs yields the low-rank Newton-ADI (LR-NADI-N) method which is illustrated in Algorithm 4. This algorithm can be seen as inner-outer method as it consists of an outer (the Newton iteration) and an inner iteration (the fADI iteration). To distinguish these two stages we will from now on use the notation that subscripts $_j$ and bracketed superscripts $^{(k)}$ will refer to quantities associated to the inner and, respectively, outer iteration.

We will now discuss some of the major steps of Algorithm 3 in more detail.

3.2.1 Solving the Linear Systems

The coefficient matrices $(A^{(k)} + \beta_j^{(k)} I_n)$, $(B^{(k)} + \alpha_j^{(k)} I_m)$ defining the linear systems in Line 6 will in general be dense matrices, even if A and B are sparse. In that case the coefficient matrices are given as a sum of a sparse matrix and a low-rank update. It is often mandatory to employ the Sherman-Morrison-Woodbury formula [20] which

Algorithm 4: Low-Rank Newton-ADI for NAREs (LR-NADI-N)

Input : Matrices $A,\ B,\ F,\ G,\ P,\ Q$ defining (1), initial guesses $K^{(0)},\ L^{(0)}$, and stopping tolerance $\tau_{\text{ADI}} \ll 1$.

Output: $Z_{k_{\max}} \in \mathbb{C}^{n\times(r+p)j}$, $Y_{k_{\max}} \in \mathbb{C}^{m\times(r+p)j}$, $\Gamma_{k_{\max}} \in \mathbb{C}^{(r+p)j\times(r+p)j}$ such that $Z_{k_{\max}}\Gamma_{k_{\max}}Y_{k_{\max}}^H \approx X$.

for $k = 1, 2, \ldots, k_{\max}$ **do**

  Determine shifts $\{\alpha_1^{(k)}, \ldots, \alpha_J^{(k)}\}$, $\{\beta_1^{(k)}, \ldots, \beta_J^{(k)}\}$ w.r.t. $A^{(k)} := A - K^{(k-1)}Q^T$ and $B^{(k)} := B - P\,L^{(k-1)\,H}$.

  $S_0^{(k)} = [F, K^{(k-1)}],\ T_0^{(k)} = [G, L^{(k-1)}],\ Z_0^{(k)} = \Gamma_0^{(k)} = Y_0^{(k)} = [\,],\ j = 0$

  while $\|S_j^{(k)}(T_j^{(k)})^H\| > \tau_{ADI}\|S_0^{(k)}(T_0^{(k)})^H\|$ **do**

    $j = j + 1$

    Solve

$$(A^{(k)} + \beta_j^{(k)} I_n)V_j^{(k)} = S_{j-1}^{(k)}, \quad (B^{(k)} + \alpha_j^{(k)} I_m)^H W_j^{(k)} = T_{j-1}^{(k)}$$

    for $V_j^{(k)},\ W_j^{(k)}$.

    Update low-rank factors of Sylvester residual

    $S_j^{(k)} = S_{j-1}^{(k)} + \gamma_j V_j^{(k)},\ T_j^{(k)} = T_{j-1}^{(k)} + \overline{\gamma_j} W_j^{(k)},\ \gamma_j = -(\beta_j^{(k)} + \alpha_j^{(k)})$.

    Augment the low-rank solution factors

$$Z_j^{(k)} = [Z_{j-1}^{(k)},\ V_j^{(k)}],\ Y_j^{(k)} = [Y_{j-1}^{(k)},\ W_j^{(k)}],\ \Gamma_j^{(k)} = \operatorname{diag}\left(\Gamma_{j-1}^{(k)},\ \gamma_j I_r\right).$$

  $K^{(k)} = Z_j^{(k)}\Gamma_j^{(k)}((Y_j^{(k)})^H P)\ ,\ L^{(k)} = Y_j^{(k)}(\Gamma_j^{(k)})^H(Z_j^{(k)})^H Q.$

amounts, e.g., for obtaining $V_j^{(k)}$ to

$$\begin{aligned}[V_S,\ V_K] &= (A + \beta_j^{(k)} I_n)^{-1}[S_{j-1}^{(k)},\ K^{(k-1)}],\\ V_j^{(k)} &= V_S + V_K\left(I_p - Q^T V_K\right)^{-1}(Q^T V_S).\end{aligned} \tag{9}$$

The equations for generating $W_j^{(k)}$ are similar. Hence, $r + 2p$ linear systems with the sparse coefficient matrices $(A+\beta_j^{(k)} I_n)$, $(B+\alpha_j^{(k)} I_m)$ have to be solved which makes this approach especially helpful if sparse-direct solvers are applied. Iterative solvers that work only with matrix vector products of the coefficient matrices might also directly be applied to $(A^{(k)} + \beta_j^{(k)} I_n)$, $(B^{(k)} + \alpha_j^{(k)} I_m)$. However, the low-rank updates might severely increase the condition number such that the Sherman-Morrison-Woodbury formula might still be preferable.

3.2.2 Implicit Updates of $K^{(k)}$, $L^{(k)}$

The approximate solution of the Sylvester equation is constructed via

$$X_j^{(k)} = Z_j^{(k)}\Gamma_j^{(k)}(Y_j^{(k)})^H = \sum_{i=1}^{j}\gamma_i V_i^{(k)}(W_i^{(k)})^H = X_{j-1}^{(k)} + \gamma_j V_j^{(k)}(W_j^{(k)})^H$$

in each inner iteration step. It is possible to recursively update $K^{(k)}$, $L^{(k)}$ in every inner iteration step as well:

$$\begin{aligned} K_j^{(k)} &:= X_j^{(k)}U = K_{j-1}^{(k)} + \gamma_j V_j^{(k)}(W_j^{(k)})^H P, \\ L_j^{(k)} &:= (X_j^{(k)})^H Q = L_{j-1}^{(k)} + \overline{\gamma_j} W_j^{(k)}(V_j^{(k)})^H Q. \end{aligned} \tag{10}$$

Hence, if only $K^{(k)}$, $L^{(k)}$ are of interest, storing the low-rank solution factors can be omitted. Also Line 9 in Algorithm 4 is not required, and Line 10 is moved into the fADI loop in the form (10). This can greatly reduce the storage requirements of the LR-NADI-N method and is the analogous situation to the implicit low-rank Newton-ADI method for CAREs [8, Algorithm 6].

3.2.3 Stopping Criteria

Both the inner and outer iteration require suitable conditions when to stop because the number of required steps is typically not known in advance. Here we employ stopping criteria based on the norm of the residual matrices. For fADI as inner iteration, this is already indicated in Line 4 of Algorithm 4, where $\|S_j^{(k)}(T_j^{(k)})^H\|$ can be efficiently computed by (7).

The following theorem shows that, as a consequence of the low-rank factorization of the Sylvester residual matrix, also the NARE residual matrix is given in a similar factored form of low rank. It can be seen as a generalizations of results from [12, 6].

Theorem 1. The NARE residual matrix w.r.t. the solution $X_j^{(k)} = Z_j^{(k)}\Gamma_j^{(k)}(Y_j^{(k)})^H$ produced at outer iteration $k \geq 1$ and inner iteration j in Algorithm 4 has at most rank $2p + r$ and is given by

$$\mathcal{R}(X_j^{(k)}) = C_j^{(k)}(D_j^{(k)})^H, \tag{11a}$$

where

$$C_j^{(k)} := [S_j^{(k)},\ K^{(k-1)} - K_j^{(k)}], \quad D_j^{(k)} := [T_j^{(k)},\ L_j^{(k)} - L^{(k-1)}] \tag{11b}$$

with $K_j^{(k)}$, $L_j^{(k)}$ defined as in (10).

Proof. Writing out the NARE residual and including the definitions of $K^{(k-1)}$, $L^{(k-1)}$

as well as $K_j^{(k)}$, $L_j^{(k)}$ yields

$$\begin{aligned}
\mathcal{R}(X_j^{(k)}) &= AX_j^{(k)} + X_j^{(k)}B - X_j^{(k)}PQ^TX_j^{(k)} + FG^T \\
&= (A - K^{(k-1)}Q^T)X_j^{(k)} + X_j^{(k)}(B - P(L^{(k-1)})^H) \\
&\quad + K^{(k-1)}Q^TX_j^{(k)} + X_j^{(k)}P(L^{(k-1)})^H - X_j^{(k)}PQ^TX_j^{(k)} \\
&\quad + FG^T + K^{(k-1)}(L^{(k-1)})^H - K^{(k-1)}(L^{(k-1)})^H \\
&= A^{(k)}X_j^{(k)} + X_j^{(k)}B^{(k)} + S_0^{(k)}(T_0^{(k)})^H \\
&\quad + K^{(k-1)}(L_j^{(k)})^H + K_j^{(k)}(L^{(k-1)})^H - K_j^{(k)}(L_j^{(k)})^H - K^{(k-1)}(L^{(k-1)})^H \\
&= S_j^{(k)}(T_j^{(k)})^H + [K^{(k-1)} - K_j^{(k)}][L_j^{(k)} - L^{(k-1)}]^H
\end{aligned}$$

from which (11) follows. Thus, rank $\left(\mathcal{R}(X_j^{(k)})\right) \leq 2p + r$ since $C_j^{(k)}$ and $D_j^{(k)}$ are of rank at most $2p + r$. □

There are some possibilities for rank deficiencies in $C_j^{(k)}$ and $D_j^{(k)}$. For instance, consider $C_j^{(k)}$ and observe that if $K^{(0)} = 0$ we have rank $\left(S_j^{(1)}\right) \leq r$, and hence, rank $\left(C_j^{(1)}\right) \leq p + r$. Similarly, rank $\left(D_j^{(1)}\right) \leq p + r$ if $L^{(0)} = 0$. Furthermore, simple manipulations reveal that the low-rank factors of the Sylvester residual matrix can be equivalently represented as

$$S_j^{(k)} = (A^{(k)} - \alpha_j^{(k)}I_n)V_j^{(k)}, \quad T_j^{(k)} = (B^{(k)} - \beta_j^{(k)}I_m)^H W_j^{(k)}.$$

Consequently, if $\alpha_j \in \Lambda(A^{(k)})$ or $\beta_j \in \Lambda(B^{(k)})$, then $(A^{(k)} - \alpha_j^{(k)}I_n)$ or $(B^{(k)} - \beta_j^{(k)}I_m)$ are singular, and, thus, $S_j^{(k)}$ or $T_j^{(k)}$ can be of rank smaller than $r + p$, see also [6, Theorem 4] for a similar discussion. Of course, solving the inner Sylvester equation exactly, i.e. $S_j^{(k)}(T_j^{(k)})^H = 0$, will also lead to rank $\left(\mathcal{R}(X_j^{(k)})\right) \leq p$.

The result of the above theorem can be equivalently proven as in [27, 12] by using a Taylor expansion of $\mathcal{R}(X)$ of order two.

Computing the norm of $\mathcal{R}(X_j^{(k)})$ can then be done in the same fashion as (7) for the Sylvester residual matrix:

$$\|\mathcal{R}(X_j^{(k)})\| = \|C_j^{(k)}(D_j^{(k)})^H\| = \sqrt{\lambda_{\max}((C_j^{(k)})^H C_j^{(k)}(D_j^{(k)})^H D_j^{(k)})},$$

where some of the matrices from the computation of $\|S_j^{(k)}(T_j^{(k)})^H\|$ can be reused for forming $(C_j^{(k)})^H C_j^{(k)}(D_j^{(k)})^H D_j^{(k)}$. Hence, computing $\|\mathcal{R}(X_j^{(k)})\|$ boils down to finding the largest eigenvalue of a matrix of size $r + 2p$. The norm of the NARE residual matrix can then be used to terminate the outer iteration, e.g., via

$$\|\mathcal{R}(X_j^{(k)})\| \leq \tau_{\mathrm{NM}}\|FG^T\|, \quad 0 < \tau_{\mathrm{NM}} \ll 1. \tag{12}$$

This criterion can actually also be monitored during each single or every couple of steps of the inner iteration. This allows to stop the fADI iteration when (12) is satisfied, regardless of the magnitude of $\mathcal{S}(X_j)$. Especially in the last outer iteration this can lead to a reduction of the number of carried out inner iterations and, thus, to computational savings. Motivated by the expected quadratic convergence rate of the Newton process, we propose to start monitoring the inner NARE residual norm when $\|\mathcal{R}(X^{(k-1)})\| \leq \sqrt{\tau_{\mathrm{NM}}}\|FG^T\|$. Alternatively, one can terminate the inner iteration when a stagnation of $\|\mathcal{R}(X_j^{(k)})\|$ is detected.

For a given τ_{NM} one typically chooses $\tau_{\mathrm{ADI}} < \tau_{\mathrm{NM}}$ to ensure that the required accuracy can be achieved. From the theory of inexact Newton methods it might conceptually be possible to solve the Sylvester equation in each outer iteration less accurately and still maintain quadratic convergence of the Newton process. In the context of algebraic Riccati equations this appears to be less straightforward because one has to additionally ensure that the Newton method still converges to the desired minimal solution. Since this issue is, to the authors knowledge, not sufficiently well understood and subject of ongoing research initiated, e.g., by [18], we will not further consider inexact solves of the inner Sylvester equations in the sense $\tau_{\mathrm{ADI}} > \tau_{\mathrm{NM}}$.

3.2.4 Shift Parameters for the Inner Iteration

As mentioned above, fADI requires two sets of shift parameters $\{\alpha_1^{(k)}, \ldots, \alpha_J^{(k)}\}, \{\beta_1^{(k)}, \ldots, \beta_J^{(k)}\}$ corresponding to the matrices $A^{(k)} := A - K^{(k-1)}Q^T$ and $B^{(k)} := B - P(L^{(k-1)})^H$. In Line 2 these are in each outer iteration step exemplary computed before the fADI iteration is started. As proposed in [37], this could be relaxed by keeping the same sets of shifts for a couple of outer iterations. Moreover, the shift computation can be moved outside the Newton-loop such that only the shift parameters w.r.t. $A^{(1)}$ and $B^{(1)}$ are used in the whole process. This might, depending on the magnitude of the changes in $K^{(k)}$ and $L^{(k)}$, slow down the fADI convergence speed but save some execution time due to skipping the shift generation. In case of an M-NARE, after convergence the spectra of the matrices $A^{(k)}$, $B^{(k)}$ will be located in the right half plane [13, Theorem 2.11.] such that it might be wise to neglect or negate the occasionally computed anti-stable eigenvalues of the projected matrices.

The self-generating A-B-shift parameters from [7] which we mentioned before can be included into LR-NADI-N in a straightforward manner by carrying out the projections using the matrices $A^{(k)}$, $B^{(k)}$. The same remarks as above concerning computed stable eigenvalues apply here as well.

An alternative but similar approach is based on the following proposition which is established very easily.

Proposition 2 (Generalization of [4, Theorem 5(c)])**.** The error $\mathcal{E}_j^{(k)} := X - X_j^{(k)}$ at the inner and outer iteration steps j and k of Algorithm 4 is the solution of the *residual NARE*

$$A_j^{(k)}\mathcal{E}_j^{(k)} + \mathcal{E}_j^{(k)}B_j^{(k)} - \mathcal{E}_j^{(k)}PQ^T\mathcal{E}_j^{(k)} + C_j^{(k)}(D_j^{(k)})^H = 0, \tag{13}$$

where $A_j^{(k)} := A - K_j^{(k)}Q^T$ and $B_j^{(k)} := B - P(L_j^{(k)})^H$.

The matrix

$$\hat{H}_j^{(k)} := \begin{bmatrix} B_j^{(k)} & PQ^T \\ C_j^{(k)}(D_j^{(k)})^H & -A_j^{(k)} \end{bmatrix} \in \mathbb{C}^{n+m \times m+n}$$

is associated to (13). Let $\hat{Q}^H \hat{H}_j^{(k)} \hat{Q} = \tilde{T}$ be the Schur form of $\hat{H}$ with $\hat{Q}^T \hat{Q} = I_{m+n}$ and partition the Schur vectors as $\hat{q}_h = \begin{bmatrix} \hat{u}_h \\ \hat{v}_h \end{bmatrix}$, $\hat{u}_h \in \mathbb{C}^m$, $\hat{v}_j \in \mathbb{C}^n$ for $h = 1 \ldots, m+n$. As Algorithm 4 converges, the lower left block of $\hat{H}_j^{(k)}$ will become smaller and smaller, and in the limit (13) will have a trivial solution. Hence, the norms $\|\hat{v}_h\|$ and $\|\hat{u}_h\|$ corresponding to the stable and, respectively, anti-stable eigenvalues of $\hat{H}_j^{(k)}$ will also converge towards zero. Assume for simplicity that (13) is associated to a nonsingular M-matrix. Then by [13, Theorem 2.11], $\hat{H}_j^{(k)}$ has exactly n stable and m anti-stable eigenvalues which are also identical to the spectra $-\Lambda(A_j^{(k)} - \mathcal{E}_j^{(k)} PQ^T)$ and $\Lambda(B_j^{(k)} - PQ^T \mathcal{E}_j^{(k)})$, respectively. In case of a singular M-matrix, there will be at least one zero eigenvalue in one of these sets. Extending the approach mentioned in [4, Section 3, Example 6-7], a natural idea is to select the next shift α_{j+1} as the negative of the stable eigenvalue of $\hat{H}_j^{(k)}$ whose $\hat{v}_h$ has the largest norm. Likewise, β_{j+1} is taken as the anti-stable eigenvalue of $\hat{H}_j^{(k)}$ with the largest norm of $\tilde{u}_h$. However, since $\hat{H}_j^{(k)}$ is a high-dimensional matrix, this generation of α_{j+1}, β_{j+1} is not feasible. Instead we propose to work with the reduced, at most $r+p$ dimensional matrix

$$\tilde{H}_j^{(k)} := \begin{bmatrix} U_B^H B_j^{(k)} U_B & U_B^H PQ^T U_A \\ U_A^H C_j^{(k)} (D_j^{(k)})^H U_B & -U_A^H A_j^{(k)} U_A \end{bmatrix},$$

where U_A and U_B are again orthogonal matrices associated to $\operatorname{span}(V_j^{(k)})$ and $\operatorname{span}(W_j^{(k)})$. We will call the shift parameters obtained with this strategy H-shifts.

Since the matrices $A^{(k)}$, $B^{(k)}$ and $\hat{H}_j^{(k)}$ are in general not symmetric even if A, B are, they might often have complex eigenvalues which in turn leads to complex shift parameters, regardless of the actual generation strategy. The efficient handling of these complex shift parameters can be carried out exactly as shown in [6] for the standalone fADI for solving Sylvester equations. If this reformulated fADI method is used within Algorithm 4 then next to the matrices $Z_j^{(k)}$, $\Gamma_j^{(k)}$, $Y_j^{(k)}$, $S_j^{(k)}$, $T_j^{(k)}$ also $K_j^{(k)}$ and $L_j^{(k)}$ will be real matrices. As before we keep the complex formulation to simplify the presentation, but in our numerical examples the real fADI version [6, Algorithm 4] will be used when needed.

3.2.5 Accelerating the Outer Iteration

In [37, 11] the outer iteration of the low-rank Newton-ADI method for GCAREs is accelerated by performing a Galerkin projection onto the space spanned by the low-rank solution factor. This often leads to an impressive convergence boost where the number of outer iterations is reduced to one or two. This projection based approach

can be incorporated into Algorithm 4. Let for this $U_Z^{(k)} \in \mathbb{C}^{n\times g}$ and $U_Y^{(k)} \in \mathbb{C}^{n\times h}$ be rectangular, orthonormal matrices for the spaces spanned by the low-rank solution factors $Z^{(k)}$ and $Y^{(k)}$ after the inner iteration in the outer iteration step k has been finished. A clever orthogonalization routine should neglect nearly linearly independent columns in $Z^{(k)}$, $Y^{(k)}$ such that $g \leq (r+p)j_{\text{it}}^{(k)}$ is possible, where $j_{\text{it}}^{(k)}$ is the number of performed inner iteration steps. Assuming we look for an approximate solution of (1) of the form $X_{\text{pr}} = U_Z^{(k)}\tilde{X}(U_Y^{(k)})^H$ for which the residual $\mathcal{R}(X_{\text{pr}})$ satisfies the Galerkin condition $(U_Z^{(k)})^T\mathcal{R}(X_{\text{pr}})U_Y^{(k)} = 0$. This is equivalent to $\tilde{X} \in \mathbb{C}^{g\times h}$ being the solution of the projected NARE

$$\tilde{A}\tilde{X} + \tilde{X}\tilde{B} - \tilde{X}\tilde{S}\tilde{T}^T\tilde{X} + \tilde{F}\tilde{G}^T = 0 \tag{14}$$

with $\tilde{A} := (U_Z^{(k)})^H A U_Z^{(k)}$, $\tilde{B} := (U_Y^{(k)})^H B U_Y^{(k)}$, $\tilde{S} := (U_Y^{(k)})^H S$, $\tilde{T} := (U_Z^{(k)})^H T$, $\tilde{F} := (U_Z^{(k)})^H F$, and $\tilde{G} := (U_Y^{(k)})^H G$. Since (14) is of much smaller dimension than the original NARE (1) it can be solve directly, e.g., by the Newton methods in Algorithms 1, 2, or by the Schur vector method [13, Listing 3.5]. The latter method can run into numerical problems when (14) is associated to a singular M-matrix. In that case a modified and more stable variant of the Schur vector method proposed in [23] should be used.

Once $\tilde{X}$ is computed, the next outer iteration step $k+1$ in Algorithm 4 is started with the updated matrices $K_{\text{pr}} := X_{\text{pr}}S = U_Z^{(k)}\tilde{X}\tilde{S}$ and $L_{\text{pr}} := X_{\text{pr}}^H T = U_Y^{(k)}\tilde{X}^H\tilde{T}$. It is important to note that the NARE residual $\mathcal{R}(X_{\text{pr}})$ will no longer have the low-rank structure (11) from Theorem 1. Hence, $\|\mathcal{R}(X_{\text{pr}})\|$ has the be computed differently, e.g., via applying a Lanczos process to $\mathcal{R}(X_{\text{pr}})^H\mathcal{R}(X_{\text{pr}})$ to get an approximation of the largest singular value which coincides with the spectral norm. Our numerical experiments confirm that this Galerkin projection indeed decreases the required number of outer iteration steps significantly. However, in contrast to the projection for the self-generating shift parameters above which use $V_j^{(k)}$ and $W_j^{(k)}$, computing orthonormal bases for $Z^{(k)}$, $Y^{(k)}$ is usually noticeable more expensive because the number of columns to be orthogonalized is a multiple of $r+p$. Hence, regarding the computational costs the acceleration of the outer iteration will only pay off if the costs for the orthogonalization of $Z^{(k)}$, $Y^{(k)}$ are not too high. One obvious way to achieve this is by keeping the number of processed inner iteration small, e.g., by using high quality shift parameters. Another way is to employ sophisticated or implicit orthogonalization routines [37].

Remark 1. The presented acceleration via a Galerkin projection is a straightforward generalization of the idea in [9, Section 4], where the authors use basically the same approach for accelerating the convergence of the fADI iteration. Hence, in principle one could also implement the Galerkin projection within the inner iteration of Algorithm 4. We refrain from this idea because of the following reasons. At first, since orthonormal spaces for the low-rank solutions factors are required, one could have used a projection method as inner iteration from the start. Popular representatives of these methods are Krylov subspace methods, e.g., [17, 28, 15], which produce the required orthonormal

matrices anyway. In [19] it is shown that the fADI method is in fact a rational Krylov method, although it works without any orthogonalization. Hence, from this viewpoint adding explicit orthogonal bases for the low-rank solution factors is not expected to give significantly better results than the plain fADI iteration given in Algorithm 3. A similar argumentation can, for the low-rank ADI iteration for Lyapunov equations, be found in [43].

3.2.6 Complexity Assessment

Here we briefly give a rough estimate of the computational costs for Algorithm 4. Suppose the linear systems with a single right hand side and coefficient matrices A, B can be dealt with by sparse direct or iterative solvers. Let us therefore assume that the computational costs for computing matrix vector products and solving linear systems with A, B (as well as their shifted versions $(A+\beta_j I_n)$), $(B+\alpha_j I_m)$) can be estimated by $\mathcal{O}(c)$, where $c = \max(n, m)$. We begin by considering the inner iteration, where in each inner iteration step of Algorithm 4 the shifted linear systems (Line 6) have to be solved. If (9) is used, $r + 2p$ right hand sides have to be processed. Constructing the small p dimensional, dense matrix requires p^2 inner products of vectors of length m or n. It can therefore be done in $p^2\mathcal{O}(c)$. Solving the corresponding linear systems by Gaussian elimination in Line 6 adds further $\mathcal{O}(p^3)$. We assume that $p \ll \min(m, n)$ and such that $\mathcal{O}(p^3)$ does not dominate $\mathcal{O}(c)$. The product $Q^T V_S$ is slightly more expensive since it require $p(r + p)$ inner products. Note, however, that both matrix-matrix products of dense rectangular matrices can be performed in highly optimized BLAS level-3 calls in contrast to the memory bandwidth limited operations with the sparse quadratic matrices. Therefore, the relatively large constants in the corresponding $\mathcal{O}(c)$ expressions will be observed much smaller in practice. Independent of this fact, $V_j^{(k)}$ and $W_j^{(k)}$ are obtained at an approximate complexity $\mathcal{O}(c)(r + (r + 2)p + p^2)$, which, since also $r \ll \min(m, n)$ is still $\mathcal{O}(c)$. In step k of the outer iteration $j_{\text{it}}^{(k)}$ inner fADI iteration steps to achieve the desired accuracy are required. Of course, these numbers will in a reasonable implementation be restricted to a maximal number, say $j_{\max}$ inner iteration steps. Hence, the computational effort of one complete run of fADI in a single step of the outer iteration can be estimated as $\theta_{\text{inner}}^{(k)} := j_{\max}\mathcal{O}(c)$. Thus, k_{it} outer iteration steps of Algorithm 4 are roughly of complexity $\theta_{\text{outer}} := k_{\text{it}} j_{\max}\mathcal{O}(c)$. To conclude, the computational costs of LR-NADI-N are essentially dominated by the solutions of the linear systems.

The costs for the shift parameter generation can also be influential. Generating the heuristic shift parameters, as described above, before each other iteration step by k_+, k_- steps of Arnoldi processes w.r.t. $A^{(k)}$, $B^{(k)}$ and their inverses will introduce additional costs of order $\mathcal{O}(c)(k_+ + k_-)k_{\text{it}}$. Since solving linear systems is typically significantly more costly than performing matrix vector products, using the A-B-, or H-shifts is clearly cheaper although the computations for their generation appear more often than for the heuristic shifts. For this we assume by a similar reasoning as above that the involved eigenvalue computations (of order $\mathcal{O}(k_+^3 + k_-^3)$ and $\mathcal{O}((p + r)^3)$) are negligible.

Using the Galerkin projection from Section 3.2.5 yields additional costs for the construction of the orthonormal bases of $Z_j^{(k)}$, $Y_j^{(k)}$. This is because the number of columns $j_{\text{it}}^{(k)}(p+r)$ to be processed might still be smaller than $\min(n,m)$, but the effort for the orthogonalization, estimated as
$\mathcal{O}(c)(j_{\text{it}}^{(k)}(p+r))^2$ if QR-decompositions are used, which are often noticeable. This is observed in one of the examples in Section 4.

3.3 Miscellaneous

3.3.1 Problems with Special Structure

In some applications, e.g. [29], the matrices A, B are of the form

$$A = \Psi - st^T, \quad B = \hat{\Psi} - xy^T, \tag{15a}$$

where $\Psi \in \mathbb{R}^{n\times n}$, $\hat{\Psi} \in \mathbb{R}^{m\times m}$ are diagonal matrices and $s,\ t \in \mathbb{R}^n$, $x,\ y \in \mathbb{R}^m$. In that case the solutions of the linear system are, by another use of the Sherman-Morrison-Woodbury formula, given by

$$\begin{aligned} V_j^{(k)} &= \tilde{V}_S + \tilde{V}_K(I_{p+1} - [t,T]^T\tilde{V}_K)^{-1}[t,T]^T\tilde{V}_S, \\ \tilde{V}_S &:= (\Psi+\beta_j I_n)^{-1}S_{j-1}^{(k)}, \quad \tilde{V}_K := (\Psi+\beta_j I_n)^{-1}[s,K] \end{aligned} \tag{15b}$$

and similarly for $W_j^{(k)}$. Since the inversions of $\Psi+\beta_j I_n$ and $\hat{\Psi}+\alpha_j I_m$ come basically for free, this leads to a very low computational effort for solving the linear systems and, thus, also for the overall algorithm. More precisely, the complexity for solving a linear system defined by a diagonal matrix of dimension n and r right hands sides can described adequately by $2nr$. Hence, obtaining $V_j^{(k)}$, $W_j^{(k)}$ using (15b) will require costs of order $(2p+r+1)c$ with $c = \max(n,m)$ as in Section 3.2.6. Hence, the complete algorithm will basically have a linear complexity. Tricks similar to (15b) have also been applied in other methods for M-NAREs, see, e.g., [24, 29, 14, 34].

3.3.2 Generalized Equations

Everything we discussed so far can also be carried over to NAREs of the form

$$\mathcal{R}(X) = FG^T + AXC + EXB - EXPQ^TXH = 0 \tag{16}$$

which we shall refer to as generalized NAREs (GNARE). For simplicity we assume that $E \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{m\times m}$ are nonsingular. We only list the changes in LR-NADI-N (Algorithm 4) in Algorithm 5 which can be derived easily by following, e.g., the manipulations done in [6] for generalized Sylvester equations.

4 Numerical Experiments

The experiments are done in MATLAB® 8.0.0.783 on an Intel®Core™2 E8400 CPU with 3.00 GHz and 4 GB RAM. The linear systems in the fADI method are solved with

Algorithm 5: Low-Rank Newton-ADI for GNAREs

2' Determine shifts $\{\alpha_1^{(k)}, \ldots, \alpha_J^{(k)}\}$, $\{\beta_1^{(k)}, \ldots, \beta_J^{(k)}\}$ w.r.t. the matrix pairs $(A^{(k)}, E)$ and $(B^{(k)}, C)$.

6' Solve $(A^{(k)} + \beta_j^{(k)} E)V_j^{(k)} = S_{j-1}^{(k)}$, $(B^{(k)} + \alpha_j^{(k)} C)^H W_j^{(k)} = T_{j-1}^{(k)}$ for $V_j^{(k)}$, $W_j^{(k)}$.

8' $S_j^{(k)} = S_{j-1}^{(k)} + \gamma_j E V_j^{(k)}$, $T_j^{(k)} = T_{j-1}^{(k)} + \overline{\gamma_j} C^T W_j^{(k)}$, $\gamma_j = -(\beta_j^{(k)} + \alpha_j^{(k)})$.

11' $K^{(k)} = E Z_j^{(k)} \Gamma_j^{(k)} ((Y_j^{(k)})^H P)$, $L^{(k)} = C^T Y_j^{(k)} (\Gamma_j^{(k)})^H (Z_j^{(k)})^H Q$.

the sparse direct solvers provided by the MATLAB backslash operator. If applicable, the Sherman-Morrison-Woodbury formula is used as explained in Section 3.2.1. The outer iteration is terminated when $\rho^{(k)} := \|\mathcal{R}(X^{(k)})\|/\|FG^T\| \leq \tau_{\mathrm{NM}}$ and, likewise, the inner iteration is stopped using the criterion employed in Algorithm 4 with τ_{ADI} or when $j_{\max}$ iteration steps have been performed. Additionally, the scaled NARE residual norm is also monitored within the inner iteration. If not stated otherwise, the initial guess $X^{(0)} = 0$ is used, i.e., $K^{(0)} = 0$ and $L^{(0)} = 0$. We use the following examples for evaluating the performance of LR-NADI-N.

Example 1. As an adaptation of [13, Example 3.18] we take

$$A = \begin{bmatrix} 3 & -1 & & \\ \ddots & \ddots & & \\ & 3 & -1 & \\ -1 & & & 1.9 \end{bmatrix} \in \mathbb{R}^{n\times n}, \quad B = \begin{bmatrix} 2 & -1 & & \\ & 3 & \ddots & \\ & & \ddots & -1 \\ -1 & & & 3 \end{bmatrix} \in \mathbb{R}^{m\times m},$$

$$F = \begin{bmatrix} F_1 \\ 0 \end{bmatrix} \in \mathbb{R}^{n\times r}, \; F_1 = \begin{bmatrix} -1 & -1 & & \\ \ddots & \ddots & & \\ & -1 & -1 & \\ & & & -0.9 \end{bmatrix} \in \mathbb{R}^{r\times r}, \quad G = I_{m,r},$$

$$P = \begin{bmatrix} P_1 \\ 0 \end{bmatrix} \in \mathbb{R}^{m\times p}, \; P_1 = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ \ddots & \ddots & & \\ & 1 & 1 & \end{bmatrix} \in \mathbb{R}^{p\times p}, \; Q = I_{n,p}.$$

with $n = 30000$, $m = 20000$, $r = 3$, and $p = 5$.

Example 2. A setting from [29] which inherits the structure (15a) with

$$\Psi = \operatorname{diag}(\delta_1, \ldots, \delta_n), \quad \hat{\Psi} = \operatorname{diag}\left(\hat{\delta}_1, \ldots, \hat{\delta}_n\right),$$
$$t = x = F = G = (q_1, \ldots, q_n)^T, \quad s = y = P = -Q = \mathbf{1}_n,$$
$$\delta_i = \frac{1}{c\xi_i(1+\nu)}, \quad \hat{\delta}_i = \frac{1}{c\xi_i(1-\nu)}, \quad q_i = \frac{\omega_i}{2\xi_i} \quad \text{for} \quad i = 1, \ldots, n.$$

There, $0 < c \leq 1$, $0 \leq \nu < 1$, ξ_i, ω_i are the quadrature nodes and weights corresponding to a Gaussian quadrature on $[0, 1]$, and $\mathbf{1}_h$ denotes the column vector of length h where all entries are equal to one. It obviously holds $r = p = 1$ and the other dimension is set

to $n = m = 20000$. We set the other defining constants to $c = 0.5$ and $\nu = 0.3$. The obtained NARE is an M-NARE and the relation (15b) is used to solve the occurring shifted linear systems.

Example 3. Consider 5-point, centered finite difference discretizations of the operators

$$\mathcal{A}(x) := \Delta x + \mathrm{e}^{\xi_1 \xi_2} \frac{\partial x}{\partial \xi_1} + \sin(\xi_1 \xi_2) \frac{\partial x}{\partial \xi_2} + (\xi_2^2 - \xi_1^2) x,$$
$$\mathcal{B}(x) := \Delta x + 100\mathrm{e}^{\xi_1} \frac{\partial x}{\partial \xi_1} + 10(\xi_1 + \xi_2) \frac{\partial x}{\partial \xi_2} + \sqrt{\xi_2^1 + \xi_2^2} x$$

for $x = x(\xi_1,\ \xi_2)$ defined on $\Omega = (0, 1)^2$ with homogeneous Dirichlet boundary conditions. The matrices A and B are obtained from using 110 and, respectively, 90 equidistant grid points for each spatial dimension such that $n = 12100$, $m = 8100$. The matrices $F,\ G,\ P,\ Q$ are random matrices with uniformly distributed entries and $r = 10$ and $p = 5$. This example is very close to the Sylvester equations used in [28, Example 2].

We were not able to acquire implementations of other methods for large-scale NAREs, e.g., [33], yet. Therefore, we leave the comparison to other methods for future work.

4.1 Influence of Different Shift Generation Strategies

At first we test the performance of LR-NADI-N for these examples when different shift strategies for the inner fADI iteration are used. We employ the heuristic Ritz value based shifts [9], the A-B-shifts [7], as well as the $\hat{H}$-shifts introduced above in Section 3.2.4. Other approaches, e.g. the one proposed in [38, Algorithm 2.1] or the `parsyl`[1] routine provided in [42] were not able to compete with this those strategies. The necessary orthonormal bases for the A-B-, and H-shifts were constructed using the MATLAB built-in `orth` routine. The setup parameters as well as the results are summarized in Table 1. There, heur($k_+,\ k_-$) refers to the heuristic shifts mentioned above which are generated by using k_+ and k_- Ritz and inverse Ritz values. The final rank, $r_f = \operatorname{rank}\left(X^{(k_{\mathrm{it}})}\right)$ of the obtained approximate solution is also listed.

Apparently, regardless of the used shift parameters strategy, the LR-NADI-N algorithm is able to compute low-rank solutions of the the large-scale NAREs in a small amount of time.

For Example 1, the A-B-shifts required the smallest number of inner iterations j_{it}, closely followed by the H-shifts. This evidently yields, compared to the heuristic shifts, smaller execution times.

Similar observations can be made for Example 2, where the heuristic shift approach required a high numbers $k_+,\ k_-$ of Ritz values to provide convergence of the inner iteration within $j_{\max}$ steps. Due to the computation costs for generating the heuristic shifts, and because the required inner iteration steps are still larger compared to the

[1] Available at http://extras.springer.com/2013/978-1-4614-5121-1.

Table 1: Results and setup parameters for the examples using different shift strategies for the inner iteration: tolerance for outer and inner iteration τ_{NM}, τ_{ADI}, maximum number of fADI iteration steps $j_{\max}$, required outer and inner iteration steps k_{it}, j_{it}. The numbers in brackets behind j_{it} are the average numbers of inner iterations. We also list the scaled NARE residual norm $\rho^{(k_{\mathrm{it}})}$, the rank r_f of the approximated solution, as well as the computation time spend in seconds.

Ex.	τ_{NM}, τ_{ADI}, $j_{\max}$	shifts	k_{it}	j_{it} (avg.)	$\rho^{(k_{\mathrm{it}})}$	r_f	time
1	10^{-10}, 10^{-12}, 40	heur(10,10)	5	77 (15.4)	$1.44{\cdot}10^{-12}$	11	16.7
		A-B-shift	5	48 (9.6)	$1.41{\cdot}10^{-11}$	11	7.7
		H-shift	5	55 (11.0)	$1.11{\cdot}10^{-13}$	11	9.6
2	10^{-9}, 10^{-10}, 300	heur(50,40)	3	413 (137.7)	$9.26{\cdot}10^{-10}$	38	16.2
		A-B-shift	3	236 (78.7)	$5.28{\cdot}10^{-10}$	40	2.8
		H-shift	3	290 (96.7)	$2.11{\cdot}10^{-10}$	39	3.9
3	10^{-8}, 10^{-10}, 100	heur(10,10)	11	408 (37.1)	$5.23{\cdot}10^{-9}$	120	177.9
		A-B-shift	11	476 (43.3)	$3.07{\cdot}10^{-9}$	126	179.5
		H-shift	11	621 (56.5)	$4.34{\cdot}10^{-9}$	126	251.0

A-B-, and H-shifts, this approach leads to significantly higher computation times. The A-B-shift strategy leads again to the best performance.

For this M-matrix example the obtained low-rank approximations

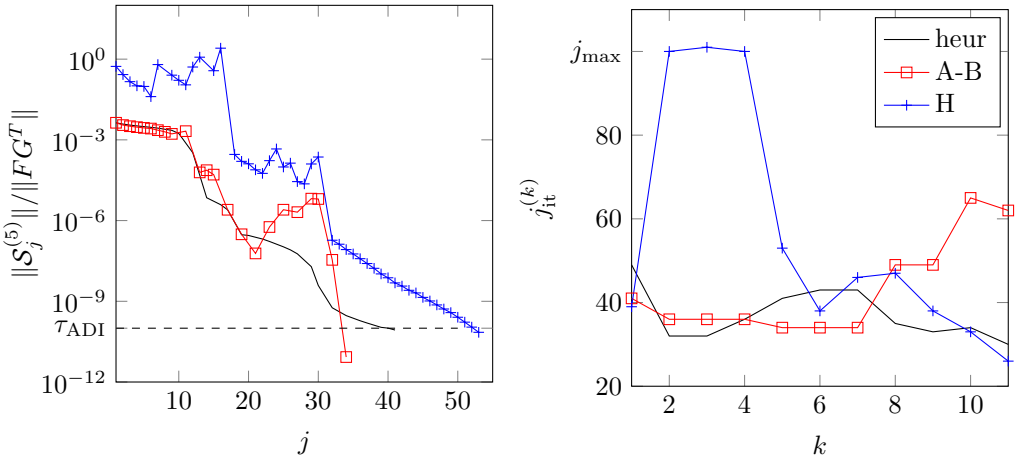
$$X^{(k)} = Z^{(k)}_{j^{(k)}_{\mathrm{it}}} \Gamma^{(k)}_{j^{(k)}_{\mathrm{it}}} \left(Y^{(k)}_{j^{(k)}_{\mathrm{it}}}\right)^T, \quad k = 1, \ldots, k_{\mathrm{it}}$$

were indeed nonnegative matrices. Hence, as for the Algorithms 1 and 2, the LR-NADI-N method produced a sequence of nonnegative approximations. The low-rank solution factors alone were, however, not nonnegative.

Example 3 appears to be the hardest one for the LR-NADI-N method resulting in the largest number of outer iteration steps k_{it}. The usage of an appropriately chosen initial guess $X^{(0)}$ could be of great value. Here, the heuristic shifts lead to a slightly better performance of the inner iteration compared to the A-B-shifts. The latter ones can, however, be generated cheaper such that the computation time is still smaller than for the heuristic shifts. We observed that the A-B-, and H-shifts lead in some outer iteration steps to a highly oscillatory behavior of the inner Sylvester residual norm. A similar observation is mentioned in [6, Section 3.6]. The H-shifts had difficulties to steer the inner iteration towards convergence within $j_{\max}$ steps. This was especially apparent in the early stage of the outer iteration when the approximate solution is still too far from the real solution. Hence, the H-shifts lead to the largest number of total inner iterations j_{it} as well as the highest execution times.

In Figure 1 we illustrate these issues. The left plot shows the progress of the scaled inner Sylvester residual norm at the fifth outer iteration for all three shift approaches. The right picture shows how the number of total required inner steps $j^{(k)}_{\mathrm{it}}$ at outer

Figure 1: Problematic behavior of the inner iteration for Example 3. Left plot: History of scaled Sylvester residual norm $\|\mathcal{S}_j^{(5)}\|/\|FG^T\|$ of fADI in the fifth outer iteration step. Right plot: number of required inner iteration steps $j_{\text{it}}^{(k)}$ against outer iteration index k.



iteration step k changes as the outer iteration proceeds.

The used shift parameters strategy does largely not effect the progress of the outer iteration, as is it can be seen by the constant number of required outer iteration steps k_{it}. The exception is the last outer iteration step, where the final obtained accuracy differs to some extend. It appears that the H-shift lead to slightly smaller residual norms, at the expense of an often slower convergence of the inner iteration compared to the A-B-shifts. It is also noteworthy that in all examples the estimated rank $\tilde{r}$ of the approximate solution is smaller than the column dimension of $Z^{k_{\text{it}}}$, $Y^{k_{\text{it}}}$. This points towards the conclusion that the used shift parameters approaches used are not yet the best possible ones. As we remarked earlier, the numerical efficient generation of high quality shift parameters for fADI is in some aspects still under current research. One could include an additional rank truncation step by, e.g., employing the approach given in [31] after or in between the outer iteration steps to throw away nearly linearly dependent columns in $Z^{k_{\text{it}}}$, $Y^{k_{\text{it}}}$ and free up the unnecessary storage.

4.2 Effect of the Galerkin Acceleration in the Outer Iteration

Now we repeat the complete above example series but employ the Galerkin acceleration described in Section 3.2.5 in each outer iteration step after the inner iteration is finished. The required orthonormal bases of $Z^{(k)}$, $Y^{(k)}$ are also computed with the `orth` routine. The small, projected NAREs are dealt with by a basic version of the Newton-Kleinman method (Algorithm 2). The settings for τ_{NM}, τ_{ADI}, $j_{\max}$ and for the shift parameter generation are kept unchanged. The results are summarized in

Table 2: Results for the examples with Galerkin projection in each outer iteration step.

Ex.	τ_{NM}, τ_{ADI}, $j_{\max}$	shifts	k_{it}	j_{it} (avg.)	$\rho^{(k_{\mathrm{it}})}$	r_f	time
1	10^{-10}, 10^{-12}, 40	heur(10,10)	1	11 (11.0)	$3.59{\cdot}10^{-12}$	13	2.2
		A-B-shift	1	10 (10.0)	$1.81{\cdot}10^{-11}$	12	1.2
		H-shift	1	10 (10.0)	$1.03{\cdot}10^{-13}$	12	1.3
2	10^{-9}, 10^{-10}, 300	heur(50,40)	2	265 (132.5)	$9.52{\cdot}10^{-10}$	38	18.1
		A-B-shift	2	147 (73.5)	$5.48{\cdot}10^{-10}$	40	4.1
		H-shift	2	150 (75.0)	$1.74{\cdot}10^{-10}$	40	4.5
3	10^{-8}, 10^{-10}, 100	heur(10,10)	2	79 (39.5)	$5.23{\cdot}10^{-9}$	120	27.4
		A-B-shift	1	41 (41.0)	$3.53{\cdot}10^{-9}$	129	9.5
		H-shift	1	39 (39.0)	$2.34{\cdot}10^{-9}$	126	8.5

Table 2. It is apparent that the Galerkin projection reduces the number of required outer iteration steps down to one or two. This naturally also leads to a reduction of the computation time compared to the results in Table 1. The exception is Example 2, where the acceleration of the outer iteration does not pay off because the numerical costs of the required orthogonalization procedure overwhelm the extraordinarily cheap solution of the linear systems by (9). Employing more efficient orthogonalization methods than `orth` or using an implicit orthogonalization can help to reduce this cost.

5 Conclusions

We investigated the numerical solution for large NAREs by a low-rank Newton iteration. Motivated by similar approaches for other matrix equations, the core idea is to approximate the NARE solution by a factorization of very low rank. In each step of the Newton iteration a large Sylvester equation has to be solved, for which we employed the fADI method [9]. It efficiently computes low-rank approximate solutions of the Sylvester equations. Several improvements known for fADI [6] have been adopted in this combination of Newton scheme and fADI. For instance, the residual of the NARE w.r.t. the approximate solution at any stage of the iterative process can be expressed explicitly as low-rank factorization which allows the cheap computations of its norm. The efficiency of the fADI method heavily relies of certain shift parameters for which we adapted existing strategies known for the fADI itself [9, 7], but also for the Newton-ADI for CAREs [4]. An acceleration strategy for the Newton iteration based on a Galerkin projection [11] has also been presented and shows remarkable speedups in the execution time at essentially no accuracy loss.

The proposed Newton-ADI method for NAREs does in principle only require that linear systems of equations with the coefficient matrices defining the NARE can be solved. It does not rely on a certain additional structures of the coefficient matrices. However, special cases given in some applications [29], e.g. diagonal matrices with a low-rank update, can be incorporated right away. Numerical experiments showed

that the proposed method can solve large-scale NAREs very efficiently in a short amount of time. Using the Galerkin acceleration can reduce the computation time even further. Although the proposed shift parameter strategies for the fADI iteration worked satisfactory there is still room for improvement. The numerically feasible generation of high quality shift parameters for ADI based methods for various kinds of matrix equations is a currently active research topic [7]. strategy

References

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

[2] R. Bartels and G. Stewart, *Solution of the Matrix Equation $AX + XB = C$: Algorithm 432*, Comm. ACM, 15 (1972), pp. 820–826.

[3] P. Benner, *Factorized Solution of Sylvester Equations with Applications in Control*, in Proc. Intl. Symp. Math. Theory Networks and Syst. MTNS 2004, 2004.

[4] P. Benner and Z. Bujanović, *On the solution of large-scale algebraic Riccati equations by using low-dimensional invariant subspaces*, Preprint MPIMD/14-15, Max Planck Institute Magdeburg, Aug. 2014. Available from http://www.mpi-magdeburg.mpg.de/preprints/.

[5] P. Benner and H. Fassbender, *On the numerical solution of large-scale sparse discrete-time Riccati equations*, Adv. Comput. Math., 35 (2011), pp. 119–147. 10.1007/s10444-011-9174-7.

[6] P. Benner and P. Kürschner, *Computing Real Low-rank Solutions of Sylvester equations by the Factored ADI Method*, Comput. Math. Appl., 67 (2014), pp. 1656–1672.

[7] P. Benner, P. Kürschner, and J. Saak, *Self-Generating and Efficient Shift Parameters in ADI Methods for Large Lyapunov and Sylvester Equations*, Elect. Trans. Numer. Anal. to appear.

[8] P. Benner, J.-R. Li, and T. Penzl, *Numerical Solution of Large Lyapunov equations, Riccati Equations, and Linear-Quadratic Control Problems*, Numer. Lin. Alg. Appl., 15 (2008), pp. 755–777.

[9] P. Benner, R.-C. Li, and N. Truhar, *On the ADI Method for Sylvester Equations*, J. Comput. Appl. Math., 233 (2009), pp. 1035–1045.

[10] P. Benner, E. Quintana-Ortí, and G. Quintana-Ortí, *Solving Stable Sylvester Equations via Rational Iterative Schemes*, J. Sci. Comp., 28 (2005 (electronic)), pp. 51–83.

[11] P. Benner and J. Saak, *A Galerkin-Newton-ADI Method for Solving Large-Scale Algebraic Riccati Equations*, Preprint SPP 1253-090, DFG-SPP153, 2010.

[12] ——, *Numerical solution of large and sparse continuous time algebraic matrix Riccati and Lyapunov equations: a state of the art survey*, GAMM Mitteilungen, 36 (2013), pp. 32–52.

[13] D. A. Bini, B. Iannazzo, and B. Meini, *Numerical Solution of Algebraic Riccati Equations*, Society for Industrial and Applied Mathematics, 2011.

[14] D. A. Bini, B. Iannazzo, and F. Poloni, *A Fast Newton's Method for a Nonsymmetric Algebraic Riccati Equation*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 276–290.

[15] A. Bouhamidi, M. Hached, M. Heyouni, and K. Jbilou, *A preconditioned block Arnoldi method for large Sylvester matrix equations*, Numer. Lin. Alg. Appl., 20 (2011), pp. 208–219.

[16] J. W. Demmel, *Three methods for refining estimates of invariant subspaces*, Computing, 38 (1987), pp. 43–57.

[17] A. El Guennouni, K. Jbilou, and A. Riquet, *Block Krylov Subspace Methods for Solving Large Sylvester Equations*, Numerical Algorithms, 29 (2002), pp. 75–96.

[18] F. Feitzinger, T. Hylla, and E. W. Sachs, *Inexact Kleinman-Newton Method for Riccati Equations*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 272–288.

[19] G. M. Flagg and S. Gugercin, *On the ADI method for the Sylvester equation and the optimal-$\mathcal{H}_2$ points*, Appl. Numer. Math., 64 (2013), pp. 50–58.

[20] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, fourth ed., 2013.

[21] G. H. Golub, S. Nash, and C. F. Van Loan, *A Hessenberg–Schur method for the problem $AX + XB = C$*, IEEE Trans. Automat. Control, AC-24 (1979), pp. 909–913.

[22] C. Guo and N. Higham, *Iterative Solution of a Nonsymmetric Algebraic Riccati Equation*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 396–412.

[23] C.-H. Guo, *Efficient methods for solving a nonsymmetric algebraic Riccati equation arising in stochastic fluid models*, 192 (2006), pp. 353–373.

[24] C.-H. Guo and A. J. Laub, *On the Iterative Solution of a Class of Nonsymmetric Algebraic Riccati Equations*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 376–391.

[25] G. A. Hewer, *An Iterative Technique for the Computation of Steady State Gains for the Discrete Optimal Regulator*, IEEE Trans. Automat. Control, AC-16 (1971), pp. 382–384.

[26] N. J. Higham, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[27] T. Hylla, *Extension of inexact Kleinman-Newton methods to a general monotonicity preserving convergence theory*, PhD thesis, Universität Trier, 2011.

[28] K. Jbilou, *Low rank approximate solutions to large Sylvester matrix equations*, Appl. Math. Comput., 177 (2006), pp. 365–376.

[29] J. Juang and I. D. Chen, *Iterative solution for a certain class of algebraic matrix Riccati equations arising in transport theory*, Transport Theory and Statistical Physics, 22 (1993), pp. 65–80.

[30] D. Kleinman, *On an Iterative Technique for Riccati Equation Computations*, IEEE Trans. Automat. Control, AC-13 (1968), pp. 114–115.

[31] D. Kressner and C. Tobler, *Krylov Subspace Methods for Linear Systems with Tensor Product Structure*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1688–1714.

[32] R.-C. Li and N. Truhar, *On the ADI Method for Sylvester Equations*, Technical Report 2008-2, Department of Mathematics, University of Texas at Arlington, 2008. available at http://www.uta.edu/math/preprint/rep2008_02.pdf.

[33] T. Li, E. K.-w. Chu, Y. Kuo, and W. Lin, *Solving Large-Scale Nonsymmetric Algebraic Riccati Equations by Doubling*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1129–1147.

[34] V. Mehrmann and H. Xu, *Explicit Solutions for a Riccati Equation from Transport Theory*, SIAM J. Matrix Anal. Appl., 30 (2009), pp. 1339–1357.

[35] V. Ramaswami, *Matrix analytic methods for stochastic fluid flows*, in Teletraffic Engineering in a Competitive World, D. Smith and P. Key, eds., Proc. of the 16th International Teletraffic Congress, Edinburgh, UK, 1999, Elsevier Science, pp. 1019–1030.

[36] J. Roberts, *Linear Model Reduction and Solution of the Algebraic Riccati Equation by Use of the Sign Function*, Internat. J. Control, 32 (1980), pp. 677–687. (Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971).

[37] J. Saak, *Efficient Numerical Solution of Large Scale Algebraic Matrix Equations in PDE Control and Model Order Reduction*, PhD thesis, TU Chemnitz, July 2009. Available from http://nbn-resolving.de/urn:nbn:de:bsz:ch1-200901642.

[38] J. Sabino, *Solution of Large-Scale Lyapunov Equations via the Block Modified Smith Method*, PhD thesis, Rice University, Houston, Texas, June 2007. Available from http://www.caam.rice.edu/tech_reports/2006/TR06-08.pdf.

[39] D. Sorensen and Y. Zhou, *Direct methods for matrix Sylvester and Lyapunov equations*, J. Appl. Math, (2002), pp. 277–303.

[40] E. L. Wachspress, *Iterative solution of the Lyapunov matrix equation*, Appl. Math. Lett., 107 (1988), pp. 87–90.

[41] ——, *Optimum parameters for two-variable ADI iteration*, Ann. Nuc. Ener., 19 (1992), pp. 765–778.

[42] ——, *The ADI Model Problem*, Springer New York, 2013.

[43] T. Wolf and H. Panzer, *The ADI iteration for Lyapunov equations implicitly performs H2 pseudo-optimal model order reduction*, Tech. Rep. 1309.3985, ArXiv e-prints, 2013.