Max Planck Institute Magdeburg
Preprints

Peter Benner Patrick Kürschner

Computing Real Low-rank Solutions of Sylvester equations by the Factored ADI Method

MPIMD/13-05 **July 15, 2013**

Impressum:

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

www.mpi-magdeburg.mpg.de/preprints

Computing Real Low-rank Solutions of Sylvester equations by the Factored ADI Method.

Peter Benner, Patrick Kürschner
MPI Magdeburg, Germany

July 15, 2013

Abstract

We investigate the factored ADI iteration for large and sparse Sylvester equations. A novel low-rank expression for the associated Sylvester residual is established which enables cheap computations of the residual norm along the iteration, and which yields a reformulated factored ADI iteration. The application to generalized Sylvester equations is considered as well.

We also discuss the efficient handling of complex shift parameters and reveal interconnections between the ADI iterates w.r.t. to those complex shifts. This yields a further modification of the factored ADI iteration which employs only an absolutely necessary amount of complex arithmetic operations and storage, and which produces low-rank solution factors consisting of entirely real data.

Certain linear matrix equations, such as, e.g., cross-Gramian Sylvester, and discrete-time Lyapunov equations, are in fact special cases of generalized Sylvester equations and we show how specially tailored low-rank ADI iterations can be deduced from the generalized factored ADI iteration.

1 Introduction

We consider the numerical solution of Sylvester equations of the form

$$AX - XB = FG^T \tag{1}$$

with $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$, $F \in \mathbb{R}^{n\times r}$, $G \in \mathbb{R}^{m\times r}$, and the sought solution $X \in \mathbb{R}^{n\times m}$, as well as generalized Sylvester equations

$$AXC - EXB = FG^T \tag{2}$$

with nonsingular matrices $C \in \mathbb{R}^{m\times m}$, $E \in \mathbb{R}^{n\times n}$. We assume throughout this paper that the spectra $\Lambda(A,\ E)$ and $\Lambda(B,\ C)$ are disjoint which ensures the existence of a

unique solution of (1), (2), see e.g., [33, 29]. We also assume that A, E, C, B are large, sparse matrices. Special cases of the above matrix equations are, e.g., standard Lyapunov equations, $B = -A^T$, $G = -F$ in (1), and generalized Lyapunov equations, $B = -A^T$, $C = E^T$, $G = -F$ in (2).

For small to medium scale equations, methods based on (generalized) Schur, eigenvalue, or Hessenberg decompositions of the involved coefficient matrices can be applied [3, 46, 28, 26, 22, 25]. Another class of methods based on the matrix sign function iteration was investigated, e.g., in [39, 16]. Since these methods have a cubic complexity and quadratic storage requirements, they are not feasible for large and sparse matrices. There are special methods for the case when only one of the pairs $(A,\ E)$ or $(B,\ C)$ is large and sparse, but the other one is much smaller and dense [44, 9].

For large and sparse problems there is a variety of Krylov subspace methods for Sylvester equations, e.g., [21, 1, 2, 32, 30, 17]. Another approach based in some sense on $\mathcal{H}_2$ interpolation is proposed in [6].

Here, we focus on methods based on low-rank versions of the alternating directions implicit (ADI) iteration [47, 18] for matrix equations. The term low-rank refers to the computation of an approximate solution $\tilde{X} \approx X$ with $\operatorname{rank}(\tilde{X}) = t \ll n, m$. This is motivated by an often observed very small (numerical) rank of the exact solution X, provided the right hand side FG^T has a low rank, i.e., $r \ll n, m$. For Lyapunov equations this phenomena was also subject to theoretical investigations [45, 27]. Low-rank (Cholesky factor) ADI methods [38, 4, 40, 14, 34] exploit this low-rank property and represent frequently used and well understood iterative methods for large-scale Lyapunov equations. This approach was generalized to Sylvester equations in [15, 36] leading to the factored ADI method which is the subject of this paper. ADI based methods require a number of shift parameters to attain a fast convergence. These shifts are in one way or another related to $\Lambda(A,\ E)$ and $\Lambda(B,\ C)$, where we explicitly focus on the case when there are complex eigenvalues in at least one of these spectra. This might lead to complex shift parameters for the ADI, which will produce a complex (low-rank) solution, or complex solution factors. Since the original Sylvester equation involves only real data, but the factored ADI will then contain complex arithmetic operations and storage, which yields higher computational costs as in the real case, this is an undesirable property. Therefore, our goal is to investigate strategies for computing real low-rank solutions with modified versions of the factored ADI, which should employ no, or at least only an absolutely necessary amount of complex arithmetic operations and storage. For this purpose, we generalize results from [12] where a real low-rank ADI method for the symmetric case of Lyapunov equations is discussed.

The remainder of this paper is organized as follows. We review the factored ADI iteration in Section 2, where we also give new properties of the residual, modify the algorithm for generalized Sylvester equations, discuss the shift parameter problem, and discuss some ideas for stopping criteria. In Section 3 the handling of complex shifts and the generation of real low-rank solution factors is discussed. Modifications to solve certain special Sylvester equations are considered in Section 4. Numerical experiments showing the performance of the investigated approaches are given in Section 5 and Section 6 summarizes and gives future research directions.

The following notation is used in this paper: $\mathbb{R}$ and $\mathbb{C}$ denote the real and complex numbers and $\mathbb{R}_{\pm}$ refers to the set of strictly positive or negative numbers, where $\mathbb{C}_{\pm}$ stands for the set of complex numbers in the open right or left half plane (i.e., they have strictly positive or negative real parts). In the matrix case, $\mathbb{R}^{n\times m}$, $\mathbb{C}^{n\times m}$ denote $n \times m$ real and complex matrices, respectively. For any complex quantity $X = \operatorname{Re}(X) + \jmath \operatorname{Im}(X)$, $\operatorname{Re}(X)$, $\operatorname{Im}(X)$ are its real and imaginary parts with $\jmath$ being the imaginary unit. The complex conjugate of X is denoted by $\overline{X} = \operatorname{Re}(X) - \jmath \operatorname{Im}(X)$. The absolute value of $\xi \in \mathbb{C}$ is denoted by $|\xi|$. The matrix A^T is the transpose of a real $n \times m$ matrix and $X^H = \overline{X}^T$ is the complex conjugate transpose of a complex one. Note that we make explicit use of both ^{T} and ^{H} depending on the considered matrix being a real or complex one. The inverse of a nonsingular matrix X is denoted by X^{-1} and, moreover, $X^{-T} = (X^T)^{-1}$ and $X^{-H} = (X^H)^{-1}$. Throughout the paper $g = X^{-1}f$ should be understood as solving the linear system of equations $Xg = f$ for g. The identity matrix of dimension k is indicated by I_k. The symbol $\otimes$ denotes the Kronecker product. If not stated otherwise, $\|X\|$ is the spectral norm of X.

2 Factored ADI Iteration for Large-Scale Sylvester Equations

2.1 The ADI Iteration for Standard Sylvester Equations

The alternating directions implicit (ADI) iteration for (1) and two sets of shift parameters $\{\alpha_k\}$, $\{\beta_k\}$ is given by

$$\begin{aligned}(A - \beta_k I_n)X_{k-\frac{1}{2}} &= X_{k-1}(B - \beta_k I_m) + FG^T,\\ X_k(B - \alpha_k I_n) &= (A - \alpha_k I_n)X_{k-\frac{1}{2}} - FG^T,\end{aligned} \tag{3}$$

see [47]. As detailed in [5, 36], rewriting the above two half steps into one single step, setting $X_0 = 0$, exploiting the structure of the iterates given by the low-rank right hand side FG^T, and reordering the shifts, leads to the factored ADI method (fADI) [15, 36] for solving (1) which is shown in Algorithm 1. It computes low-rank solution factors $Z_{k_{\max}} \in \mathbb{C}^{n\times rk_{\max}}$, $Y_{k_{\max}} \in \mathbb{C}^{m\times rk_{\max}}$, $D_{k_{\max}} \in \mathbb{C}^{rk_{\max}\times rk_{\max}}$ such that the product $Z_{k_{\max}} D_{k_{\max}} Y_{k_{\max}}^H \in \mathbb{C}^{n\times m}$ approximates the solution X. Note that for Lyapunov equations ($B = -A^T$, $G = -F$, $\beta_k = -\overline{\alpha_k}$) the above algorithm gives the low-rank (Cholesky factor) ADI method (LR-ADI) [38, 35, 40, 14]. In each iteration of Algorithm 1, r new columns are added to the low-rank factors Y_{k-1}, Z_{k-1}, and D_{k-1} is augmented by an $r \times r$ diagonal matrix. The main computational tasks are the solutions of the linear systems of equations with the shifted A, B matrices. We assume that A, B are large, but sparse matrices and we are able to employ sparse-direct or iterative Krylov subspace methods for solving the linear systems. Moreover, it should hold that $r \ll n$ since the column dimension r of F, G determines the number of right hand sides in the linear systems. A small value of r is also crucial for the existence of a low-rank solution of the Sylvester equation. The shift parameters $\{\alpha_k\}$, $\{\beta_k\}$ steer the convergence of the iteration and are discussed in Subsection 2.4

Algorithm 1: Original factored ADI (fADI) for (1)

Input : $A,\ B,\ F,\ G$ as in (1) and shift parameters $\{\alpha_1, \ldots, \alpha_{k_{\max}}\}$, $\{\beta_1, \ldots, \beta_{k_{\max}}\}$.

Output: $Z_{k_{\max}} \in \mathbb{C}^{n \times rk_{\max}}$, $Y_{k_{\max}} \in \mathbb{C}^{m \times rk_{\max}}$, $D_{k_{\max}} \in \mathbb{C}^{rk_{\max} \times rk_{\max}}$ such that $Z_{k_{\max}} D_{k_{\max}} Y_{k_{\max}}^H \approx X$.

for $k = 1, 2, \ldots, k_{\max}$ **do**
  if $k = 1$ **then**
    $V_1 = (A - \beta_1 I_n)^{-1} F$, $W_1 = (B - \alpha_1 I_m)^{-H} G$.
  else
    $V_k = V_{k-1} + (\beta_k - \alpha_{k-1})(A - \beta_k I_n)^{-1} V_{k-1}$.
    $W_k = W_{k-1} + \overline{(\alpha_k - \beta_{k-1})}(B - \alpha_k I_n)^{-H} W_{k-1}$.
  Update the low-rank solution factors

  $$Z_k = [Z_{k-1}, V_k],\ Y_k = [Y_{k-1}, W_k],\ D_k = \operatorname{diag}(D_{k-1}, \gamma_k I_r),\ \gamma_k := \beta_k - \alpha_k.$$

which is followed by some ideas for suitable stopping criteria in Subsection 2.5. The above algorithm obviously employs complex arithmetic operations and storage if some of the used shifts are complex numbers. This undesirable property is the main issue of this work and will be investigated in detail in Section 3. Before we continue we will investigate the structure of the Sylvester residual obtained with Algorithm 1 which will give us a novel reformulated version of fADI. After that we generalize Algorithm 1 for solving (2) in order to stick with this more general class of problems in the remainder.

2.2 The fADI Sylvester Residual

The following lemmas are helpful for providing insights into the fADI iteration and the structure of the residual.

Lemma 1. For each $M \in \mathbb{C}^{n \times n}$ and $\xi,\ \mu \in \mathbb{C}$ the matrices $(M \pm \xi I_n)^{\pm 1}$ and $(M \pm \nu I_n)^{\pm 1}$ commute, provided the inverses exist.

Lemma 2 (Generalization of [31, Lemma 3.1.1], [23, Lemma 5.1])**.** For every complex $\beta \notin \Lambda(A),\ \alpha \notin \Lambda(B),\ \alpha \neq \beta$, (1) is equivalent to the discrete-time Sylvester equation

$$X = \mathcal{C}(A, \beta, \alpha) X \mathcal{C}(B, \alpha, \beta) + T(\alpha, \beta), \tag{4}$$

where $T(\alpha, \beta) := (\beta - \alpha)(A - \beta I_n)^{-1} F G^T (B - \alpha I_m)^{-1}$ and $\mathcal{C}$ is a generalized Cayley type transformation:

$$\mathcal{C}(M, \xi, \nu) := (M - \xi I)^{-1}(M - \nu I). \tag{5}$$

Proof. Equation (1) is obviously equivalent to

$$(A - \beta I_n) X (B - \alpha I_m) - (A - \alpha I_n) X (B - \beta I_m) - (\beta - \alpha) F G^T = 0$$

from which the result follows since the inverses of $A - \beta I_n$, $B - \alpha I_m$ exist. □

Let in the remainder $\beta_j \notin \Lambda(A)$, $\alpha_j \notin \Lambda(B)$, $\alpha_j \neq \beta_j$ hold for all $j = 1, 2, \ldots, k_{\max}$ in (3).

Lemma 3 (Generalization of [31, Lemma 3.5.1], [23, Lemma 5.2])**.** The error after k steps of the Sylvester ADI can be expressed as

$$X_k - X = \left[\prod_{j=1}^{k} \mathcal{C}(A, \beta_j, \alpha_j)\right] (X_0 - X) \left[\prod_{j=1}^{k} \mathcal{C}(B, \alpha_j, \beta_j)\right]. \tag{6}$$

Proof. From (3) we have, using the notation of the above lemma, that the ADI approximation at iteration k can be expressed as

$$X_k = \mathcal{C}(A, \beta_k, \alpha_k) X_{k-1} \mathcal{C}(B, \alpha_k, \beta_k) + T(\alpha_k, \beta_k)$$

which yields with (4)

$$X_k - X = \mathcal{C}(A, \beta_k, \alpha_k)(X_{k-1} - X)\mathcal{C}(B, \alpha_k, \beta_k).$$

The result follows from an repeated application of this identity. □

Lemma 4 (Generalization of [31, Lemma 3.5.2], [23, Lemma 5.3])**.** For the residual R_k after k iterations of (3) it holds

$$R_k = AX_k - X_kB - FG^T = \left[\prod_{j=1}^{k} \mathcal{C}(A, \beta_j, \alpha_j)\right] R_0 \left[\prod_{j=1}^{k} \mathcal{C}(B, \alpha_j, \beta_j)\right]. \tag{7}$$

Proof. Using

$$AX_k - X_kB - FG^T = A(X_k - X) - (X_k - X)B$$

yields with (6) the sought expression because A and $\mathcal{C}(A, \beta_j, \alpha_j)$, as well as B and $\mathcal{C}(B, \alpha_j, \beta_j)$, commute for all j. □

The following theorem generalizes a result for the LR-ADI iteration applied to Lyapunov equations [13].

Theorem 5 (Generalization of [13, Theorem 1])**.** Assume that $\operatorname{rank}(G) = \operatorname{rank}(F) = r$. Then the residual at step k of fADI (Algorithm 1) is of rank at most r and given by

$$\begin{aligned} R_k &:= AZ_kD_kY_k^H - Z_kD_kY_k^HB - FG^T = -\hat{V}_k\hat{W}_k^H, \\ \hat{V}_k &:= \hat{V}_{k-1} + \gamma_k V_k \in \mathbb{C}^{n\times r},\ \hat{V}_0 := F, \\ \hat{W}_k &:= \hat{W}_{k-1} - \overline{\gamma_k} W_k \in \mathbb{C}^{n\times r},\ \hat{W}_0 := G, \end{aligned}$$

where $\gamma_k := \beta_k - \alpha_k$.

Proof. Note that Algorithm 1 starts from $X_0 = 0$. Thus, $R_0 = -FG^T$ and the expression (7) yields already $R_k = -\hat{V}_k \hat{W}_k^H$ with

$$\hat{V}_k = \left[\prod_{j=1}^{k} \mathcal{C}(A, \beta_j, \alpha_j)\right] F, \quad \hat{W}_k = \left[\prod_{j=1}^{k} \mathcal{C}(B, \alpha_j, \beta_j)^H\right] G. \tag{8}$$

This shows that $\operatorname{rank}(R_k) = r$ if $\beta_j,\ \alpha_j \notin \Lambda(A) \cup \Lambda(B)$. If $\alpha_j \in \Lambda(A)$ or $\beta_j \in \Lambda(B)$ for some j, the inverses still exist but $A - \alpha_j I_n$, or respectively $B - \beta_j I_m$, is singular, such that the column rank of $\hat{V}_k$ or $\hat{W}_k$ can be smaller than r. Exploiting again the commutativity relations from Lemma 1, the increments V_k, W_k in steps 5, 6 of Algorithm 1 can be expressed as

$$\begin{aligned} V_k &= (A - \alpha_{k-1} I_n)(A - \beta_k I_n)^{-1} V_{k-1} \\ &= (A - \beta_k I_n)^{-1}(A - \alpha_{k-1} I_n)(A - \alpha_{k-2} I_n)(A - \beta_{k-1} I_n)^{-1} V_{k-2} \\ &= (A - \beta_k I_n)^{-1} \mathcal{C}(A, \beta_{k-1}, \alpha_{k-1})(A - \alpha_{k-2} I_n) V_{k-2} \\ &= \ldots = (A - \beta_k I_n)^{-1} \left[\prod_{j=1}^{k-1} \mathcal{C}(A, \beta_j, \alpha_j)\right] F, \end{aligned} \tag{9a}$$

$$W_k = (B - \alpha_k I_m)^{-H} \left[\prod_{j=1}^{k-1} \mathcal{C}(B, \alpha_j, \beta_j)^H\right] G. \tag{9b}$$

A comparison of (8) with (9) yields $\forall k \geq 1$

$$\hat{V}_k = (A - \alpha_k I_n) V_k, \quad \hat{W}_k = (B - \beta_k I_n)^H W_k$$

from which we conclude that the increments can be written as

$$V_k = (A - \beta_k I_n)^{-1} \hat{V}_{k-1}, \quad W_k = (B - \alpha_k I_n)^{-H} \hat{W}_k. \tag{10}$$

Consequently,

$$\begin{aligned} \hat{V}_k &= (A - \alpha_k I_n)(A - \beta_k I_n)^{-1} \hat{V}_{k-1} \\ &= (I_n + (\beta_k - \alpha_k)(A - \beta_k I_n)^{-1}) \hat{V}_{k-1} = \hat{V}_{k-1} + \gamma_k V_k, \end{aligned} \tag{11a}$$

$$\begin{aligned} \hat{W}_k &= (B - \beta_k I_m)^H (B - \alpha_k I_m)^{-H} \hat{W}_{k-1} \\ &= (I_m + \overline{(\alpha_k - \beta_k)}(B - \alpha_k I_m)^{-H}) \hat{W}_{k-1} = \hat{W}_{k-1} - \overline{\gamma_k} W_k, \end{aligned} \tag{11b}$$

which are the desired formulas for the low-rank factors of R_k. □

With (10) and (11), Algorithm 1 can be reformulated as given in Algorithm 2, where the low-rank factors $\hat{V}_k$, $\hat{W}_k$ of the residual matrix R_k are now integral part of the iteration. Note that this result enables cheap evaluations of $\|R_k\| = \|\hat{V}_k \hat{W}_k^H\|$ such that Algorithm 2 can be terminated using a stopping criterion based on the normalized residual norm, see Section 2.5.

Algorithm 2: Reformulated Factored ADI iteration (fADI) for (1)

Input : $A,\ B,\ F,\ G$ as in (1) and shift parameters $\{\alpha_1, \ldots, \alpha_{k_{\max}}\}$, $\{\beta_1, \ldots, \beta_{k_{\max}}\}$, tolerance $0 < \tau \ll 1$.

Output: $Z_{k_{\max}} \in \mathbb{C}^{n \times rk_{\max}}$, $Y_{k_{\max}} \in \mathbb{C}^{m \times rk_{\max}}$, $D_{k_{\max}} \in \mathbb{C}^{rk_{\max} \times rk_{\max}}$ such that $Z_{k_{\max}} D_{k_{\max}} Y_{k_{\max}}^H \approx X$.

$\hat{V}_0 = F,\ \hat{W}_0 = G,\ k = 1.$

while $\|\hat{V}_{k-1}\hat{W}_{k-1}^H\| \geq \tau \|FG^T\|$ **do**

  $V_k = (A - \beta_k I_n)^{-1}\hat{V}_{k-1},\ W_k = (B - \alpha_k I_m)^{-H}\hat{W}_{k-1}.$

  $\hat{V}_k = \hat{V}_{k-1} + \gamma_k V_k,\ \hat{W}_k = \hat{W}_{k-1} - \overline{\gamma_k} W_k,\ \gamma_k = \beta_k - \alpha_k.$

  Update the low-rank solution factors

$$Z_k = [Z_{k-1}, V_k],\ Y_k = [Y_{k-1}, W_k],\ D_k = \operatorname{diag}(D_{k-1}, \gamma_k I_r).$$

  $k = k + 1.$

2.3 Application to Generalized Sylvester Equations

Motivated by the derivation of the generalized low-rank ADI (G-LR-ADI) iteration [4, 40] for computing low-rank solution factors of generalized Lyapunov equations, we investigate how Algorithm 2 can be modified accordingly to treat generalized Sylvester equations (2). Due to the assumed nonsingularity of E and C, (2) is equivalent to the standard Sylvester equation

$$\begin{aligned} &\tilde{A}X - X\tilde{B} = \tilde{F}\tilde{G}^T \\ \text{with}\quad &\tilde{A} := E^{-1}A,\ \tilde{B} := BC^{-1},\ \tilde{F} := E^{-1}F,\ \tilde{G} := C^{-T}G \end{aligned} \tag{12}$$

to which Algorithm 1 can be applied directly. However, in a large scale setting forming $\tilde{A}$, $\tilde{B}$ is infeasible and moreover, depending on the sparsity pattern of E and C, the sparsity of $\tilde{A}$ and $\tilde{B}$ might be worse or even completely lost. This can be circumvented by rewriting the associated steps, especially the linear systems in Algorithm 2. The residual of the equivalent standard Sylvester equation is $\tilde{R}_k = \tilde{A}X_k - X_k\tilde{B} - \tilde{F}\tilde{G}^T = -\check{V}_k\check{W}_k^H$, where

$$\check{V}_k = \check{V}_{k-1} + \gamma_k V_k, \quad \check{W}_k = \check{W}_{k-1} - \overline{\gamma_k} W_k, \quad \check{V}_0 := \tilde{F}, \quad \check{W}_0 := \tilde{G}.$$

Clearly, it holds for the residual w.r.t. (2) that $R_k = E\tilde{R}_k C = -\hat{V}_k\hat{W}_k$ with

$$\hat{V}_k := E\check{V}_k = \hat{V}_{k-1} + \gamma_k E V_k, \quad \hat{W}_k := C^T\check{W}_k = \hat{W}_{k-1} - \overline{\gamma_k} C^T W_k, \tag{13}$$

where $V_k,\ W_k$ are obtained from the linear systems

$$\begin{aligned} V_k &= (\tilde{A} - \beta_k I_n)^{-1}\check{V}_{k-1} = (A - \beta_k E)^{-1}\hat{V}_{k-1}, \\ W_k &= (\tilde{B} - \alpha_k I_m)^{-H}\check{W}_{k-1} = (B - \alpha_k C)^{-H}\hat{W}_{k-1}. \end{aligned}$$

Algorithm 3: Generalized factored ADI iteration (G-fADI) for (2)

Input : $A,\ B,\ E,\ C,\ F,\ G$ as in (2) and shift parameters $\{\alpha_1, \ldots, \alpha_{k_{\max}}\}$, $\{\beta_1, \ldots, \beta_{k_{\max}}\}$, tolerance $0 < \tau \ll 1$.

Output: $Z_{k_{\max}} \in \mathbb{C}^{n \times rk_{\max}}$, $Y_{k_{\max}} \in \mathbb{C}^{m \times rk_{\max}}$, $D_{k_{\max}} \in \mathbb{C}^{rk_{\max} \times rk_{\max}}$ such that $Z_{k_{\max}} D_{k_{\max}} (Y_{k_{\max}})^H \approx X$.

$\hat{V}_0 = F,\ \hat{W}_0 = G,\ k = 1.$

while $\|\hat{V}_{k-1}\hat{W}_{k-1}^H\| \geq \tau \|FG^T\|$ **do**

  $V_k = (A - \beta_k E)^{-1}\hat{V}_{k-1},\ W_k = (B - \alpha_k C)^{-H}\hat{W}_{k-1}.$

  $\hat{V}_k = \hat{V}_{k-1} + \gamma_k E V_k,\ \hat{W}_k = \hat{W}_{k-1} - \overline{\gamma_k} C^T W_k,\ \gamma_k = \beta_k - \alpha_k.$

  Update the low-rank solution factors

$$Z_k = [Z_{k-1}, V_k],\ Y_k = [Y_{k-1}, W_k],\ D_k = \operatorname{diag}\left(D_{k-1}, \gamma_k I_r\right).$$

  $k = k + 1.$

This result is the generalized factored ADI (G-fADI) iteration given in Algorithm 3. It reduces for generalized Lyapunov equations ($B = -A^T$, $G = -F$, $C = E^T$, $\beta_i = -\overline{\alpha_i}$) to the G-LR-ADI iteration [4]. All the other results of Section 2.2 can be carried over to the generalized case by employing a generalized Cayley type transformation

$$\mathcal{C}(M, N, \xi, \nu) := (M - \xi N)^{-1}(M - \nu N) \tag{14}$$

with M, N replaced by A, E and B, C as required.

2.4 Shift Parameters

Good shift parameters are essential for a fast convergence of the method. For normal matrices (i.e. the left coincide with the right eigenvectors) A and B of (1), the shift parameters for k iterations of Algorithm 1 satisfy the optimization problem

$$\min_{\substack{\alpha_j \in \mathbb{C} \\ \beta_j \in \mathbb{C}}} \max_{\substack{\lambda \in \mathbb{E} \\ \mu \in \mathbb{F}}} \prod_{j=1}^{k} \left| \frac{(\lambda - \alpha_j)(\mu - \beta_j)}{(\lambda - \beta_j)(\mu - \alpha_j)} \right|, \tag{15}$$

where $\mathbb{E} = \Lambda(A)$ and $\mathbb{F} = \Lambda(B)$, see [41]. For the generalized Sylvester equation involving normal pencils $A - \lambda E$, $B - \mu C$ and Algorithm 3, one uses $\mathbb{E} = \Lambda(A,\ E)$, $\mathbb{F} = \Lambda(B,\ C)$ in (15). For large-scale matrices, $\mathbb{E}$, $\mathbb{F}$ are not known and not cheaply available. Hence, as proposed in [15, 36] the heuristic approach by Penzl [38] for the LR-ADI shifts can be easily extended for (G-)fADI by setting $\mathbb{E}$, $\mathbb{F}$ in (15) as sets containing a small number of approximate eigenvalues of $(A,\ E)$, $(B,\ C)$, and then solving this reduced min-max problem. Usually, the approximate eigenvalues are chosen as Ritz values generated with a few steps of an Arnoldi process. The complete algorithm of this strategy is given in [15, Algorithm 2] and more detailed information regarding its efficient implementation can be found in [36]. In our examples we just

took a number of the obtained Ritz values as shifts since this seemed to work quiet well. Moreover, this work is concerned with the handling of complex shift parameters within Algorithms 1 and 3 no matter how they are derived. Another rather costly strategy for computing suboptimal shifts was recently proposed in [6, 24].

2.5 Stopping Criteria

One common way to stop Algorithms 2,3 is when the residual

$$R_k := A(Z_k D_k Y_k^H)C - E(Z_k D_k Y_k^H)B - FG^T \in \mathbb{C}^{n\times m}$$

is small enough, e.g., $\|R_k\| < \tau\|FG^T\|$ in some matrix norm where τ is a small prescribed tolerance. Using (11), the spectral- and Frobenius norm can be computed efficiently via

$$\|R_k\| = \sqrt{\|R_k^H R_k\|} = \sqrt{\|\hat{W}_k(\hat{V}_k^H\hat{V}_k)\hat{W}_k^H\|} = \sqrt{\|H_k\hat{W}_k^H\hat{W}_k H_k^H\|}, \tag{16a}$$

$$= \sqrt{\|R_k R_k^H\|} = \sqrt{\|\hat{V}_k(\hat{W}_k^H\hat{W}_k)\hat{V}_k^H\|} = \sqrt{\|J_k\hat{V}_k^H\hat{V}_k J_k^H\|}, \tag{16b}$$

where H_k, J_k are the lower triangular factors of thin QR factorizations of $\hat{V}_k$, $\hat{W}_k$. This essentially reduces the residual norm computation to the computation of the norm of an $r\times r$ matrix ($H_k^H\hat{W}_k^H\hat{W}_k H_k$ or $J_k^H\hat{V}_k^H\hat{V}_k J_k$). Counting the required flops shows that (16a) and (16b) should be chosen when $m < n$ and $m > n$, respectively. This approach is significantly cheaper than, e.g., estimating the spectral norm indirectly via a Lanczos process on $R_k^H R_k$ or $R_k R_k^H$. There, the main work for each Lanczos iteration would be matrix-vector products $y = R_k x$, $x \in \mathbb{C}^m$, and $z = R_k^H y$, $y \in \mathbb{C}^n$. These products can be formed without an explicit construction of R_k, which requires 2 matrix-vector products with an $m \times m$ as well as an $n \times n$ sparse matrix, and 6 with thin, dense rectangular matrices ($G^T u$, Fu, $(D_k Y_k^H)u$, $Z_k u$ with u having a suitable dimension in each case). In total, a matrix-vector product $z = R_k^H(R_k x)$ requires 4 $n \times n$, 4 $m \times m$ and 12 rectangular matrix-vector products. Additionally, a diagonalization of the produced tridiagonal matrix T_k for finding an approximation of $\lambda_{\max}$ in needed. This approach might, however, still be useful if the low-rank solution is improved by Galerkin projection approaches [15] since then (11) does not hold anymore. Some numerical evidence that computing the residual norm via (16) is more efficient than using Lanczos can be found in Section 5 and in [13, Section 5] for generalized Lyapunov equations.

Alternatively, the relative changes of the low-rank factors Z, Y can be used as stopping criteria, as it is also used in the LR-ADI for Lyapunov equations [14]. There one stops the iteration if

$$\frac{\|V_k\|_F}{\|Z_k\|_F} \le \varepsilon_{\mathrm{rc}}, \quad \frac{\|W_k\|_F}{\|Y_k\|_F} \le \varepsilon_{\mathrm{rc}},$$

where $\varepsilon_{\mathrm{rc}}$ is a tiny prescribed tolerance. Using the Frobenius-norm allows a cheap accumulation since $\|Z_k\|_F^2 = \|V_k\|_F^2 + \|Z_{k-1}\|_F^2$ and similarly for $\|Y_k\|_F^2$. However, it is

theoretically possible that this stopping criterion is fulfilled for the Z-, but not for the Y-factors, or wise-versa. There, one could in principle still continue the Y-iterations alone which would, after fulfilling the relative change criterion there as well, produce Z- and Y-factors with different column dimensions and a rectangular D-factor. For instance, if Z- and Y-sequences are stopped after k_Z and, respectively, k_Y iterations, then $Z_{j_Z} \in \mathbb{C}^{n\times k_Z r}$, $Y_{k_Y} \in \mathbb{C}^{m\times k_Y r}$, and $D \in \mathbb{C}^{k_Z r\times k_Y r}$. We leave this for further research since we exclusively used the residual based stopping criterion in this paper.

3 Computing Real Low-Rank Factors in the Presence of Complex Shift Parameters

Here we present an approach for generating real low-rank solutions within the fADI method. Next to the matrices defining (1), (2) being real, the main ingredient for all of these approaches is that both sets of shift parameters are proper, i.e., they are of the form

$$\{\alpha_1, \ldots, \alpha_{K_a}\} = \{\nu_1, \ldots, \nu_{L_a}\} \subset \mathbb{C},$$
$$\{\beta_1, \ldots, \beta_{K_b}\} = \{\mu_1, \ldots, \mu_{L_b}\} \subset \mathbb{C},$$

where ν_k, $k = 1, \ldots, L_a$, is either a real number or a pair of complex conjugate numbers $\{\alpha_k, \overline{\alpha_k}\}$. Similarly for μ_k, $k = 1, \ldots, L_b$. This is no restrictive but a natural assumption since the (approximate) eigenvalues of the real matrix pairs $(A,\ E)$ and $(B,\ C)$ used for generating the shift parameters come in complex conjugate pairs, too. Moreover, we restrict the set of pairs (ν_k, μ_k) to the following cases:

1. Both ν_k and μ_k are real numbers α_k and β_k.
2. Both $\nu_k = \{\alpha_k, \alpha_{k+1} = \overline{\alpha_k}\}$ and $\mu_k = \{\beta_k, \beta_{k+1} = \overline{\beta_k}\}$ are pairs of complex conjugate numbers.
3. A complex pair meets two real shifts:
   a) ν_k, ν_{k+1} are real numbers and μ_k is a complex pair,
   b) ν_k is a complex pair and μ_k, μ_{k+1} are real numbers.

This is not a severe restriction since it can be achieved by a simple and usually slight reordering and rearrangement of the sets of shifts. Moreover, due to the commutativity relation in Lemma 1, the iterates of the (G-)fADI iteration after a number of iterations do not change if the order of the processed shifts is changed. The above restrictions, however, drastically simplify the encountered equations for generating real low-rank factors which will become clear later.

3.1 Interconnections Between Complex Iterates

Our approach for computing real solution factors is motivated by [12] for the (G-)LR-ADI iteration for (generalized) Lyapunov equations. There, the connection between

the iterates V_k, V_{k+1} w.r.t. the consecutive complex shifts α_k, $\overline{\alpha_k}$ is carefully exploited leading to a significant reduction of the required (complex) arithmetic operations. It is shown that

$$V_{k+1} = \overline{V_k} + 2\tfrac{\operatorname{Re}(\alpha_k)}{\operatorname{Im}(\alpha_k)} \operatorname{Im}(V_k),$$

which reveals that the second complex linear system with the coefficient matrix $A+\overline{\alpha_k}E$ is not required anymore. We now investigate the adaption of this technique to the (G-)fADI iteration. Eventually this will lead to a generalization of [12, Theorem 1]. It will turn out that, depending on the current and previous shift parameters, Z- and Y-block-iterates can be constructed from the real and imaginary parts of previous iterates. As in the proof for [12, Theorem 1] we work through different possible subsequences of shift parameters and begin with case 2) since nothing has to be taken care of in case 1).

Theorem 6. Let $\hat{V}_{k-1}$, $\hat{W}_{k-1} \in \mathbb{R}^{n\times r}$, $\{\alpha_k,\ \alpha_{k+1} := \overline{\alpha_k}\}$, and $\{\beta_k,\ \beta_{k+1} := \overline{\beta_k}\}$. Then the iterates at step $k+1$ of Algorithm 3 are given by

$$V_{k+1} = \overline{V_k} + \tfrac{\gamma_k}{\operatorname{Im}(\beta_k)} \operatorname{Im}(V_k), \tag{17a}$$

$$\hat{V}_{k+1} = \hat{V}_{k-1} + 2\operatorname{Re}(\gamma_k) E \operatorname{Re}(V_k) + \left(\tfrac{|\gamma_k|^2}{\operatorname{Im}(\beta_k)} - 2\operatorname{Im}(\gamma_k)\right) E \operatorname{Im}(V_k) \in \mathbb{R}^{n\times r}, \tag{17b}$$

$$W_{k+1} = \overline{W_k} + \tfrac{\overline{\gamma_k}}{\operatorname{Im}(\alpha_k)} \operatorname{Im}(W_k), \tag{17c}$$

$$\begin{aligned} \hat{W}_{k+1} = \hat{W}_{k-1} &- 2\operatorname{Re}(\gamma_k) C^T \operatorname{Re}(W_k) \\ &- \left(\tfrac{|\gamma_k|^2}{\operatorname{Im}(\alpha_k)} + 2\operatorname{Im}(\gamma_k)\right) C^T \operatorname{Im}(W_k) \in \mathbb{R}^{m\times r}. \end{aligned} \tag{17d}$$

Proof. At step 3 of Algorithm 3, V_k is obtained from $(A-\beta_k E)V_k = \hat{V}_{k-1} \in \mathbb{R}^{n\times r}$. Splitting β_k and V_k into their real and imaginary parts gives

$$\begin{aligned} (A - \operatorname{Re}(\beta_k)E) \operatorname{Re}(V_k) &= \hat{V}_k, \\ (A - \operatorname{Re}(\beta_k)E) \operatorname{Im}(V_k) &= \operatorname{Im}(\beta_k) E \operatorname{Re}(V_k). \end{aligned} \tag{18}$$

For V_{k+1} this yields, employing (18),

$$\begin{aligned} V_{k+1} &= (A-\overline{\beta_k}E)^{-1}\hat{V}_k = (A-\overline{\beta_k}E)^{-1}(\hat{V}_{k-1} + \gamma_k E V_k) \\ &= \overline{V_k} + \gamma_k (A-\overline{\beta_k}E)^{-1} \left(\tfrac{1}{\operatorname{Im}(\beta_k)}(A - \operatorname{Re}(\beta_k)E)\operatorname{Im}(V_k) + \jmath E \operatorname{Im}(V_k)\right) \\ &= \overline{V_k} + \tfrac{\gamma_k}{\operatorname{Im}(\beta_k)} (A-\overline{\beta_k}E)^{-1}(A-\overline{\beta_k}E)\operatorname{Im}(V_k) \end{aligned}$$

and (17a) is established. The relation (17b) follows from

$$\begin{aligned} \hat{V}_{k+1} &= \hat{V}_k + \gamma_{k+1} E V_{k+1} \\ &= \hat{V}_{k-1} + \gamma_k E V_k + \overline{\gamma_k} E \left(\overline{V_k} + \tfrac{\gamma_k}{\operatorname{Im}(\beta_k)} E \operatorname{Im}(V_k)\right) \\ &= \hat{V}_{k-1} + 2\operatorname{Re}(\gamma_k) E \operatorname{Re}(V_k) + \left(\tfrac{|\gamma_k|^2}{\operatorname{Im}(\beta_k)} - 2\operatorname{Im}(\gamma_k)\right) E \operatorname{Im}(V_k). \end{aligned}$$

For W_k we have $(B - \operatorname{Re}(\alpha_k)C)^T \operatorname{Im}(W_k) = -\operatorname{Im}(\alpha_k)C^T \operatorname{Re}(W_k)$ and thus

$$\begin{aligned} W_{k+1} &= (B - \overline{\alpha_k}C)^{-H}\left(\hat{W}_{k-1} - \overline{\gamma_k}C^T W_k\right) \\ &= \overline{W_k} - \overline{\gamma_k}(B - \overline{\alpha_k}C)^{-H}\left(\tfrac{-1}{\operatorname{Im}(\alpha_k)}(B - \operatorname{Re}(\alpha_k)C)^T \operatorname{Im}(W_k) + \jmath C^T \operatorname{Im}(W_k)\right) \\ &= \overline{W_k} + \tfrac{\overline{\gamma_k}}{\operatorname{Im}(\alpha_k)}(B - \overline{\alpha_k}C)^{-H}\left((B - \operatorname{Re}(\alpha_k)C)^T \operatorname{Im}(W_k) - \jmath \operatorname{Im}(\alpha_k)C^T \operatorname{Im}(W_k)\right) \end{aligned}$$

giving (17c) and consequently (17d) via

$$\begin{aligned} \hat{W}_{k+1} &= \hat{W}_k - \overline{\gamma_{k+1}}C^T W_{k+1} \\ &= \hat{W}_{k-1} - \overline{\gamma_k}C^T W_k - \gamma_k C^T\left(\overline{W_k} + \tfrac{\overline{\gamma_k}}{\operatorname{Im}(\alpha_k)}C^T \operatorname{Im}(W_k)\right) \\ &= \hat{W}_{k-1} - 2\operatorname{Re}(\gamma_k)C^T \operatorname{Re}(W_k) - \left(\tfrac{|\gamma_k|^2}{\operatorname{Im}(\alpha_k)} + 2\operatorname{Im}(\gamma_k)\right)C^T \operatorname{Im}(W_k). \end{aligned}$$

□

Note that only $\hat{V}_{k+1}, \hat{W}_{k+1}$ are needed to continue the iteration and since both are real, the result also holds if the algorithm is continued with another pair of complex conjugate shifts. Exactly as in the result for the G-LR-ADI for Lyapunov equations, the data corresponding to iteration $k+1$ is constructed from quantities already available after iteration k such that the linear systems w.r.t. the complex conjugate shifts $\overline{\alpha_k}$, $\overline{\beta_k}$ are obsolete which significantly reduces the overall computational costs.

We can now investigate the associated part of the low-rank solution. There, Z_{k-1}, Y_{k-1}, and D_{k-1} are augmented by

$$\begin{aligned} [V_k,\ V_{k+1}] &= [\operatorname{Re}(V_k) + \jmath \operatorname{Im}(V_k),\ \operatorname{Re}(V_{k+1}) + \jmath \operatorname{Im}(V_{k+1})] \\ &= \underbrace{[\operatorname{Re}(V_k),\ \operatorname{Im}(V_k)]}_{=:\hat{Z}_k} \underbrace{\begin{bmatrix} 1 & 1 \\ \jmath & \frac{\operatorname{Re}(\beta_k) - \alpha_k}{\operatorname{Im}(\beta_k)} \end{bmatrix} \otimes I_r}_{=:S_{Z_k}}, \\ [W_k,\ W_{k+1}] &= \underbrace{[\operatorname{Re}(W_k),\ \operatorname{Im}(W_k)]}_{=:\hat{Y}_k} \underbrace{\begin{bmatrix} 1 & 1 \\ \jmath & \frac{\overline{\beta_k} - \operatorname{Re}(\alpha_k)}{\operatorname{Im}(\alpha_k)} \end{bmatrix} \otimes I_r}_{=:S_{Y_k}}, \\ D_{(k,k+1)} &:= \operatorname{diag}(\gamma_k,\ \overline{\gamma_k}) \otimes I_r. \end{aligned}$$

The corresponding part of the solution X_{k+1} is

$$[V_k,\ V_{k+1}]D_{(k,k+1)}[W_k,\ W_{k+1}]^H = \hat{Z}_k \hat{D}_k \hat{Y}_k^T \in \mathbb{R}^{n \times m}$$

with

$$\begin{aligned} \hat{D}_k &:= S_{Z_k} D_{(k,k+1)} S_{Y_k}^H \\ &= \begin{bmatrix} 2\operatorname{Re}(\gamma_k) & \frac{|\gamma_k|^2}{\operatorname{Im}(\alpha_k)} + 2\operatorname{Im}(\gamma_k) \\ \frac{|\gamma_k|^2}{\operatorname{Im}(\beta_k)} - 2\operatorname{Im}(\gamma_k) & \left(\frac{|\gamma_k|^2}{\operatorname{Im}(\beta_k)\operatorname{Im}(\alpha_k)} + 2\right)\operatorname{Re}(\gamma_k) \end{bmatrix} \otimes I_r \in \mathbb{R}^{2r \times 2r}, \end{aligned} \tag{19}$$

such that only real data is added to the existing low-rank factors. This finishes the treatment of case 2).

In case 3), i.e., if only one of the two pairs is complex and the other two involved shifts are real, the situation changes slightly but follows directly from the previous Theorem.

Corollary 7. Let $\hat{V}_{k-1}$, $\hat{W}_{k-1}$ be real. For case 3a), i.e., β_k, $\beta_{k+1} := \overline{\beta_k}$ and $\alpha_k,\ \alpha_{k+1} \in \mathbb{R}$ it holds

$$V_{k+1} = \operatorname{Re}(V_k) + \tfrac{\operatorname{Re}(\gamma_k)}{\operatorname{Im}(\beta_k)} \operatorname{Im}(V_k) \in \mathbb{R}^{n\times r}, \tag{20a}$$

$$\begin{aligned}\hat{V}_{k+1} &= \hat{V}_{k-1} + (\operatorname{Re}(\gamma_k) + \operatorname{Re}(\gamma_{k+1}))E \operatorname{Re}(V_k) \\ &\quad + \tfrac{\operatorname{Re}(\gamma_k)\operatorname{Re}(\gamma_{k+1}) - \operatorname{Im}(\beta_k)^2}{\operatorname{Im}(\beta_k)} E \operatorname{Im}(V_k) \in \mathbb{R}^{n\times r},\end{aligned} \tag{20b}$$

$$W_{k+1} = W_k - \gamma_{k+1}\tilde{W}_{k+1} \in \mathbb{C}^{m\times r}, \quad \tilde{W}_{k+1} := (B - \alpha_{k+1}C)^{-T}C^T W_k \in \mathbb{R}^{n\times r}, \tag{20c}$$

$$\hat{W}_{k+1} = \hat{W}_{k-1} - (\operatorname{Re}(\gamma_k) + \operatorname{Re}(\gamma_{k+1}))C^T W_k + \delta_k C^T \tilde{W}_{k+1} \in \mathbb{R}^{m\times r}, \tag{20d}$$

where $\delta_k := \alpha_{k+1}^2 - 2\operatorname{Re}(\beta_k)\alpha_{k+1} + |\beta_k|^2$. For case 3b), i.e. $\beta_k,\ \beta_{k+1} \in \mathbb{R}$ and $\alpha_k,\ \alpha_{k+1} = \overline{\alpha_k} \in \mathbb{C}$ we have

$$V_{k+1} = V_k + \gamma_{k+1}\tilde{V}_{k+1} \in \mathbb{C}^{n\times r}, \quad \tilde{V}_{k+1} := (A - \beta_{k+1}E)^{-1}EV_k \in \mathbb{R}^{n\times r}, \tag{21a}$$

$$\hat{V}_{k+1} = \hat{V}_{k-1} + (\operatorname{Re}(\gamma_k) + \operatorname{Re}(\gamma_{k+1}))EV_k + \delta_k E\tilde{V}_{k+1} \in \mathbb{R}^{n\times r}, \tag{21b}$$

$$W_{k+1} = \operatorname{Re}(W_k) + \tfrac{\operatorname{Re}(\gamma_k)}{\operatorname{Im}(\alpha_k)} \operatorname{Im}(W_k) \in \mathbb{R}^{m\times r}, \tag{21c}$$

$$\begin{aligned}\hat{W}_{k+1} &= \hat{W}_{k-1} - (\operatorname{Re}(\gamma_k) + \operatorname{Re}(\gamma_{k+1}))C^T \operatorname{Re}(W_k) \\ &\quad + \tfrac{-\operatorname{Re}(\gamma_k)\operatorname{Re}(\gamma_{k+1}) + \operatorname{Im}(\alpha_k)^2}{\operatorname{Im}(\alpha_k)} C^T \operatorname{Im}(W_k) \in \mathbb{R}^{m\times r},\end{aligned} \tag{21d}$$

where $\delta_k := \beta_{k+1}^2 - 2\operatorname{Re}(\alpha_k)\beta_{k+1} + |\alpha_k|^2$.

Proof. In case 3a) the relation (20a) follows directly from (17a) by using $\operatorname{Im}(\alpha_k) = 0$. Plugging this into the constituting equations for $\hat{V}_{k+1}$ leads, after some simplifications, to (20b). The equation (20c) is nothing else than the original constructing formula for $W_{k+1} = W_k - \gamma_{k+1}\left((B - \alpha_{k+1}C)^{-T}(C^T W_k)\right)$ and exploiting that the solution of the occurring linear system is real due to $\alpha_{k,k+1} \in \mathbb{R}$, $W_k \in \mathbb{R}^{m\times r}$. Using this with some simple rearrangements leads then to (20d). The case 3b) is proved using similar steps. □

Along the lines of case 2) it can be revealed that the new parts in the low-rank factors in case 3a) are given by the real blocks $\hat{Z}_k := [\operatorname{Re}(V_k), \operatorname{Im}(V_k)]$, $\hat{Y}_k := [W_k, \tilde{W}_{k+1}]$, and

$$\hat{D}_k = \begin{bmatrix} \operatorname{Re}(\gamma_k) + \operatorname{Re}(\gamma_{k+1}) & -\delta_k \\ \frac{\operatorname{Re}(\gamma_k)\operatorname{Re}(\gamma_{k+1}) - \operatorname{Im}(\beta_k)^2}{\operatorname{Im}(\beta_k)} & \frac{-\operatorname{Re}(\gamma_k)\delta_k}{\operatorname{Im}(\beta_k)} \end{bmatrix} \otimes I_r \in \mathbb{R}^{2r\times 2r}. \tag{22}$$

Similarly, the low-rank factors in case 3b) are augmented by $\hat{Z}_k := [V_k, \tilde{V}_{k+1}]$, $\hat{Y}_k := [\operatorname{Re}(W_k),\ \operatorname{Im}(W_k)]$, and

$$\hat{D}_k := \begin{bmatrix} \operatorname{Re}(\gamma_k)+\operatorname{Re}(\gamma_{k+1}) & \frac{\operatorname{Re}(\gamma_k)\operatorname{Re}(\gamma_{k+1})-\operatorname{Im}(\alpha_k)^2}{\operatorname{Im}(\alpha_k)} \\ \delta_k & \frac{\delta_k \operatorname{Re}(\gamma_k)}{\operatorname{Im}(\alpha_k)} \end{bmatrix} \otimes I_r \in \mathbb{R}^{2r\times 2r} \tag{23}$$

which finishes the discussion regarding case 3).

Remark 1. Without the restrictions on the possible shift subsequences (case 1 – case 3) it is possible that there are longer recurrences of the V_k, W_k iterates. Consider, for instance, the sequence β_k, $\beta_{k+1} = \overline{\beta_k}$, β_{k+2}, $\beta_{k+3} = \overline{\beta_{k+2}} \in \mathbb{C}$, $\alpha_k \in \mathbb{R}$, α_{k+1}, $\alpha_{k+2} = \overline{\alpha_{k+1}}$ and $\alpha_{k+3} \in \mathbb{R}$. Using the same techniques as above, it is possible to show, e.g., that

$$\begin{aligned} V_{k+3} = &\tfrac{\operatorname{Im}(\beta_{k+2})-\operatorname{Im}(\alpha_{k+1})}{\operatorname{Im}(\beta_{k+2})} \operatorname{Re}(V_{k+2}) + \tfrac{\operatorname{Re}(\beta_{k+2})-\operatorname{Re}(\alpha_{k+1})}{\operatorname{Im}(\beta_{k+2})} \operatorname{Im}(V_{k+2}) \\ &- \tfrac{\operatorname{Im}(\alpha_{k+1})}{\operatorname{Im}(\beta_{k+2})} \operatorname{Re}(V_k) - \tfrac{\operatorname{Im}(\alpha_{k+1})}{\operatorname{Im}(\beta_{k+2})} \tfrac{\operatorname{Re}(\beta_k)-\alpha_k}{\operatorname{Im}(\beta_k)} \operatorname{Im}(V_k) \in \mathbb{R}^{n\times r} \end{aligned}$$

and similar, equally complicated relations exist for $\hat{V}_{k+3}$, W_{k+3}, and $\hat{W}_{k+3}$. Hence, $4r$ new columns are added to Z_{k-1}, Y_{k-1}, and D_{k-1} is augmented by an $4r \times 4r$ block at its diagonal. However, if α_{k+3} is complex, the expressions get even longer and more complicated. Real extensions for the low-rank factors can only be generated if the shift sequence ends with either an α or β shift being real. However, since ADI type iterations are independent of the order of the shifts, these longer relations are not needed.

Using the relations and real expansions discovered in the previous section, we are now able to provide a modified G-fADI algorithm which takes proper care of complex shifts with respect to the above considered cases. Note that the $\tilde{V}$, $\tilde{W}$ quantities encountered in case 3) do not require additional storage of an $n \times r$ array in a clever implementation. Algorithm 4 illustrates the complete G-fADI iteration for generating real low-rank solution factors (G-fADI-R). Moreover, some of the constants needed in the computation of the $\hat{V}$, $\hat{W}$ matrices can be reused in the $\hat{D}$ factor. It is also reasonable to test for convergence via the residual norm when the $\hat{V}$, $\hat{W}$ variables are real, i.e., after the current case has been processed. The same holds if one wishes to improve the solution X_k via Galerkin projection ideas [15, 36], or decrease the storage requirements by employing column compression techniques on the low-rank factors [40], which can then be carried out in real arithmetic only.

3.2 Avoiding All Complex Operations

Algorithm 4 computes real low-rank solution factors but temporarily employs some complex arithmetic operations and storage although the amount is significantly reduced compared to the original Algorithm 3. In this section we mention two approaches to get rid of the remaining complex arithmetic computations and storage. In particular, these are the solutions of the linear systems when a shift is complex, and

Algorithm 4: G-fADI-R for generating real low-rank solutions of (2)

Input : $A,\ B,\ E,\ C,\ F,\ G$ as in (2) and proper, suitably ordered shift parameters $\{\alpha_1, \ldots, \alpha_{k_{\max}}\}$, $\{\beta_1, \ldots, \beta_{k_{\max}}\}$, tolerance $0 < \tau \ll 1$.

Output: $Z_{k_{\max}} \in \mathbb{R}^{n \times rk_{\max}}$, $Y_{k_{\max}} \in \mathbb{R}^{m \times rk_{\max}}$, $D_{k_{\max}} \in \mathbb{R}^{rk_{\max} \times rk_{\max}}$ such that $Z_{k_{\max}} D_{k_{\max}} Y_{k_{\max}}^T \approx X$.

$\hat{V}_0 = F,\ \hat{W}_0 = G,\ k = 1.$

while $\|\hat{V}_{k-1}\hat{W}_{k-1}^T\| \geq \tau \|FG^T\|$ **do**

  $V_k = (A - \beta_k E)^{-1}\hat{V}_{k-1},\ W_k = (B - \alpha_k C)^{-H}\hat{W}_{k-1}.$

  $\gamma_k = \beta_k - \alpha_k.$

  if $\beta_k \in \mathbb{R} \ \wedge \alpha_k \in \mathbb{R}$ **then**

    $\hat{V}_k = \hat{V}_{k-1} + \gamma_k E V_k,\ \hat{W}_k = \hat{W}_{j-1} - \overline{\gamma_k} C^T W_k.$

    $Z_k = [Z_{k-1}, V_k],\ Y_k = [Y_{k-1}, W_k],\ D_k = \operatorname{diag}(D_{k-1}, \gamma_k I_r),\ k = k + 1.$

  if $\beta_k \in \mathbb{C} \ \wedge \alpha_k \in \mathbb{C}$ **then**

    $\hat{V}_{k+1} = \hat{V}_{k-1} + 2\operatorname{Re}(\gamma_k) E \operatorname{Re}(V_k) + \left(\frac{|\gamma_k|^2}{\operatorname{Im}(\beta_k)} - 2\operatorname{Im}(\gamma_k)\right) E \operatorname{Im}(V_k).$

    $\hat{W}_{k+1} = \hat{W}_{k-1} - 2\operatorname{Re}(\gamma_k) C^T \operatorname{Re}(W_k) - \left(\frac{|\gamma_k|^2}{\operatorname{Im}(\alpha_k)} + 2\operatorname{Im}(\gamma_k)\right) C^T \operatorname{Im}(W_k).$

    $Z_{k+1} = [Z_{k-1}, \operatorname{Re}(V_k), \operatorname{Im}(V_k)],\ Y_{k+1} = [Y_{k-1}, \operatorname{Re}(W_k), \operatorname{Im}(W_k)]$

    $D_k = \operatorname{diag}\left(D_{k-1}, \hat{D}_k\right)$ with $\hat{D}_k$ from (19), $k = k + 2.$

  if $\beta_k \in \mathbb{C} \ \wedge \ \alpha_k \in \mathbb{R} \ \wedge \ \alpha_{k+1} \in \mathbb{R}$ **then**

    $\gamma_{k+1} = \overline{\beta_k} - \alpha_{k+1},\ \delta_k := \alpha_{k+1}^2 - 2\operatorname{Re}(\beta_k)\alpha_{k+1} + |\beta_k|^2.$

    $\hat{V}_{k+1} = \hat{V}_{k-1} + (\operatorname{Re}(\gamma_k + \gamma_{k+1})) E \operatorname{Re}(V_k) + \frac{\operatorname{Re}(\gamma_k)\operatorname{Re}(\gamma_{k+1}) - \operatorname{Im}(\beta_k)^2}{\operatorname{Im}(\beta_k)} E \operatorname{Im}(V_k).$

    $\hat{W}_k = \hat{W}_{k-1} - (\operatorname{Re}(\gamma_k) + \operatorname{Re}(\gamma_{k+1})) C^T W_k,$

    $W_{k+1} = (B - \alpha_{k+1} C)^{-T} C^T W_k.$

    $\hat{W}_{k+1} = \hat{W} + \delta_k C^T W_{k+1}.$

    $Z_{k+1} = [Z_{k-1}, \operatorname{Re}(V_k), \operatorname{Im}(V_k)],\ Y_{k+1} = [Y_{k-1}, W_k, W_{k+1}],$

    $D_k = \operatorname{diag}\left(D_{k-1}, \hat{D}_k\right)$ with $\hat{D}_k$ from (22), $k = k + 2.$

  if $\beta_k \in \mathbb{R} \ \wedge \ \beta_{k+1} \in \mathbb{R} \ \wedge \ \alpha_k \in \mathbb{C}$ **then**

    $\gamma_{k+1} = \beta_{k+1} - \overline{\alpha_k},\ \delta_k := \beta_{k+1}^2 - 2\operatorname{Re}(\alpha_k)\beta_{k+1} + |\alpha_k|^2.$

    $\hat{V}_k = \hat{V}_{k-1} + (\operatorname{Re}(\gamma_k) + \operatorname{Re}(\gamma_{k+1})) E V_k,$

    $V_{k+1} := (A - \beta_{k+1} E)^{-1} E V_k,\ \hat{V}_{k+1} = \hat{V}_k + \delta_k E V_{k+1}.$

    $\hat{W}_{k+1} = \hat{W}_{k-1} - (\operatorname{Re}(\gamma_k) + \operatorname{Re}(\gamma_{k+1})) C^T \operatorname{Re}(W_k) + \frac{-\operatorname{Re}(\gamma_k)\operatorname{Re}(\gamma_{k+1}) + \operatorname{Im}(\alpha_k)^2}{\operatorname{Im}(\alpha_k)} C^T \operatorname{Im}(W_k).$

    $Z_{k+1} = [Z_{k-1}, V_k, V_{k+1}],\ Y_{k+1} = [Y_{k-1}, \operatorname{Re}(W_k), \operatorname{Im}(W_k)],$

    $D_k = \operatorname{diag}\left(D_{k-1}, \hat{D}_k\right)$ with $\hat{D}_k$ from (23), $k = k + 2.$

the corresponding V_k and W_k iterates. One way to get rid of these remaining complex instances is to rewrite the complex linear systems, exemplary $(A-\beta_k E)V_k = \hat{V}_{k-1}$, into an equivalent real representation, for instance:

$$\begin{bmatrix} A-\mathrm{Re}\,(\beta_k)E & -\mathrm{Im}\,(\beta_k)E \\ \mathrm{Im}\,(\beta_k)E & A-\mathrm{Re}\,(\beta_k)E \end{bmatrix} \begin{bmatrix} \mathrm{Re}\,(V_k) \\ \mathrm{Im}\,(V_k) \end{bmatrix} = \begin{bmatrix} \hat{V}_{k-1} \\ 0 \end{bmatrix}, \tag{24}$$

where we exploited that the right hand side $\hat{V}_{k-1}$ (but also $\hat{W}_{k-1}$) is a real $n \times r$ matrix after each case has been processed in G-fADI-R. There are also other equivalent real representations possible [20]. Using this and rewriting Algorithm 4 such that it operates on $\mathrm{Re}\,(\hat{V}_k)$, $\mathrm{Im}\,(\hat{W}_k)$, will give a formulation where all remaining complex operations and storage requirements are discarded. For the G-LR-ADI for generalized Lyapunov equations this has been carried out in [10, 11]. The price one has to pay is that the $2n \times 2n$ system (24) is often much more expensive to solve that the original $n \times n$ complex one since modern computing environments are usually perfectly capable of handling complex linear systems. We do not pursue this further since we do not expect any run time savings from this approach as it can be seen in the numerical examples in [10, 11].

Another way to work exclusively with real arithmetic operations is based on the observation that

$$(M \pm \xi I)(M \pm \overline{\xi} I) = M^2 \pm 2\,\mathrm{Re}\,(\xi) M + |\xi|^2 I$$

for all $M \in \mathbb{C}^{n\times n}$ and $\xi \in \mathbb{C}$. For Lyapunov equations and the LR-ADI iteration this was used to derive a completely real algorithm [14, Algorithm 4],[35]. A similar algorithm for generalized Lyapunov equations and the G-LR-ADI iterations can be found in [11, Algorithm 1]. It is also possible to exploit the above identity in the G-fADI iteration for generalized Sylvester equations. Consider for instance the first two iterations of G-fADI for two complex conjugate pairs of shift $\{\alpha_1, \overline{\alpha_1}\}$ and $\{\beta_1, \overline{\beta_1}\}$. We have (using the original formulation as in Algorithm 1) $V_1 = (A-\beta_1 E)^{-1}F$ and

$$\begin{aligned} V_2 &= \left(I_n + (\overline{\beta_1} - \alpha_1)(A-\overline{\beta_1}E)^{-1}E\right) V_1 = (A-\overline{\beta_1}E)^{-1}(A-\alpha_1 E)(A-\beta_1 E)^{-1}F \\ &= (E^{-1}A - \alpha_1 I_n)(E^{-1}A - \overline{\beta_1} I_n)^{-1}(A-\beta_1 E)^{-1}F \\ &= (E^{-1}A - \alpha_1 I_n)(AE^{-1}A - 2\,\mathrm{Re}\,(\beta_1)A + |\beta_1|^2 E)^{-1}F = -\alpha_1 \tilde{V}_1 + \tilde{V}_2, \end{aligned}$$

where

$$\tilde{V}_1 := (AE^{-1}A - 2\,\mathrm{Re}\,(\beta_1)A + |\beta_1|^2 E)^{-1}F,\ \tilde{V}_2 := E^{-1}(A\tilde{V}_1). \tag{25}$$

In a similar way one obtains

$$\begin{aligned} W_2 = -\overline{\beta_1}\tilde{W}_1 + \tilde{W}_2,\ \tilde{W}_1 &:= (BC^{-1}B - 2\,\mathrm{Re}\,(\alpha_1)B + |\alpha_1|^2 B)^{-T}G, \\ \tilde{W}_2 &:= C^{-T}(B^T\tilde{W}_1). \end{aligned} \tag{26}$$

The other cases can also be handled in that way which allows to rewrite Algorithm 3 into a completely real form, where no complex computations and storage are used.

The disadvantage, however, is that the properties of the coefficient matrices in the required linear systems in (25),(26) are most likely much worse than those of the original complex linear systems. Due to the squaring ($E = I_n$ or $C = I_m$) or the involved inverses of E, C they can, except when E, C are, e.g., identity or diagonal matrices, become easily large and dense matrices or it might not even be possible to construct them explicitly. Hence, the efficiency of sparse-direct or iterative solvers is severely deteriorated rendering the overall algorithm much less effective. Note that additional solves with E and C are required in the generalized case. Some numerical evidence of this argumentation (for Lyapunov equations) can be found in the numerical examples in [12, 10, 11] and we do not pursue this approach either. If really no complex operations are wanted or possible, the first approach using the augmented linear systems should be preferred.

4 Special Sylvester Equations

It is known that for generalized Lyapunov equations ($B = -A^T$, $C = E^T$, $G = -F$, $\alpha_k = -\overline{\beta_k}$) G-fADI reduces to the G-LR-ADI iteration. Consequently, case 3) cannot occur and G-fADI-R simplifies to [11, Algorithm 2] and to [12, Algorithm 3] if $E = I_n$. In this section we discuss some more special cases of the general Sylvester equation (2) which also lead to simpler versions of G-fADI-R.

4.1 Cross-Gramian Sylvester Equation

Choosing $B = -A$, $C = E$ leads to Sylvester equations of the form

$$AXE + EXA = FG^T \tag{27}$$

which occur, e.g., in cross-Gramian model order reduction [44]. There, the defining matrices A, E, F, G represent an linear, time-invariant control system. With the reasonable choice $\beta_k = -\alpha_k$ these changes lead to

$$\begin{aligned} V_k &= (A + \alpha_k E)^{-1}\hat{V}_{k-1}, \quad W_k = -(A + \alpha_k E)^{-H}\hat{W}_{k-1}, \\ \hat{V}_k &= \hat{V}_{k-1} - 2\alpha_k E V_k, \quad \hat{V}_k = \hat{W}_{k-1} + 2\overline{\alpha_k} E^T W_k. \end{aligned}$$

If the linear system for V_k is solved using a sparse LU decomposition $LU = (A+\alpha_k E)$, the LU factors can be reused for solving the second linear system for W_k since $U^H L^H = (A+\alpha_k E)^H$. It is also clear that case 3) cannot happen. If α_k is a complex shift followed by its complex conjugate, using (17a)–(17d) gives

$$\begin{aligned} V_{k+1} &= \overline{V_k} + 2\tfrac{\alpha_k}{\operatorname{Im}(\alpha_k)} \operatorname{Im}(V_k), \\ \hat{V}_{k+1} &= \hat{V}_{k-1} - 4\operatorname{Re}(\alpha_k) E \operatorname{Re}(V_k) - 4\tfrac{\operatorname{Re}(\alpha_k)^2}{\operatorname{Im}(\alpha_k)} E \operatorname{Im}(V_k), \\ W_{k+1} &= \overline{W_k} + 2\tfrac{\overline{\alpha_k}}{\operatorname{Im}(\alpha_k)} \operatorname{Im}(W_k), \\ \hat{W}_{k+1} &= \hat{W}_{k-1} + 4\operatorname{Re}(\alpha_k) E^T \operatorname{Re}(W_k) - 4\tfrac{\operatorname{Re}(\alpha_k)^2}{\operatorname{Im}(\alpha_k)} E^T \operatorname{Im}(W_k). \end{aligned}$$

Algorithm 5: G-fADI-R for generating real low-rank solutions of (27)

Input : $A,\ E,\ F,\ G$ as in (27) and proper shift parameters $\{\alpha_1, \ldots, \alpha_{k_{\max}}\} \subset \mathbb{C}_-$, tolerance $0 < \tau \ll 1$.

Output: $Z_{k_{\max}} \in \mathbb{R}^{n \times rk_{\max}}$, $Y_{k_{\max}} \in \mathbb{R}^{n \times rk_{\max}}$ such that $Z_{k_{\max}} Y_{k_{\max}}^T \approx X$.

$\hat{V}_0 = F,\ \hat{W}_0 = G,\ k = 1.$

while $\|\hat{V}_{k-1}\hat{W}_{k-1}^T\| \geq \tau \|FG^T\|$ **do**

  $V_k = (A + \alpha_k E)^{-1}\hat{V}_{k-1},\ W_k = -(A + \alpha_k E)^{-H}\hat{W}_{k-1}.$

  if $\alpha_k \in \mathbb{R}$ **then**

    $\hat{V}_k = \hat{V}_{k-1} - 2\alpha_k E V_k,\ \hat{W}_k = \hat{W}_{j-1} + 2\alpha_k E^T W_k.$

    $Z_k = [Z_{k-1}, \sqrt{-2\alpha_k} V_k],\ Y_k = [Y_{k-1}, \sqrt{-2\alpha_k} W_k],\ k = k + 1.$

  else

    $\delta_k := \frac{\operatorname{Re}(\alpha_k)}{\operatorname{Im}(\alpha_k)},\ \phi_k := 2\sqrt{-\operatorname{Re}(\alpha_k)}.$

    $\hat{V}_{k+1} = \hat{V}_{k-1} + \phi_k^2 E\,(\operatorname{Re}(V_k) - \delta_k \operatorname{Im}(V_k)).$

    $\hat{W}_{k+1} = \hat{W}_{k-1} - \phi_k^2 E^T\,(\operatorname{Re}(W_k) + \delta_k \operatorname{Im}(W_k)).$

    $Z_{k+1} = \left[Z_{k-1}, \phi_k(\operatorname{Re}(V_k) + \delta_k \operatorname{Im}(V_k)), \phi_k\sqrt{\delta_k^2 + 1} \cdot \operatorname{Im}(V_k)\right].$

    $Y_{k+1} = \left[Y_{k-1}, \phi_k(\operatorname{Re}(W_k) - \delta_k \operatorname{Im}(V_k)), \phi_k\sqrt{\delta_k^2 + 1} \cdot \operatorname{Im}(W_k)\right].$

    $k = k + 2.$

The $2r \times 2r$ augmentation of D_{k-1} can also be deduced from (19):

$$\hat{D}_k = 4 \begin{bmatrix} -\operatorname{Re}(\alpha_k) & \frac{\operatorname{Re}(\alpha_k)^2}{\operatorname{Im}(\alpha_k)} \\ -\frac{\operatorname{Re}(\alpha_k)^2}{\operatorname{Im}(\alpha_k)} & \frac{\operatorname{Re}(\alpha_k)(2|\alpha_k|^2 - \operatorname{Im}(\alpha_k)^2)}{\operatorname{Im}(\alpha_k)^2} \end{bmatrix} \otimes I_r.$$

Usually, the underlying system defining (27) is assumed to be asymptotically stable, i.e., $\Lambda(A, E) \subset \mathbb{C}_-$. If we restrict all α_k also to $\mathbb{C}_-$, then there exists the factorization

$$\hat{D}_k = -4\operatorname{Re}(\alpha_k) \begin{bmatrix} 1 & 0 \\ \delta_k & 1 \end{bmatrix} \begin{bmatrix} 1 & \\ & \delta_k^2 + 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta_k \\ 0 & 1 \end{bmatrix} \otimes I_r$$

with $\delta_k := \frac{\operatorname{Re}(\alpha_k)}{\operatorname{Im}(\alpha_k)}$ and the middle diagonal matrix has only positive entries. Hence, the above factorization of $\hat{D}_k$ can implicitly be accumulated into the augmentations of Z_{k-1} and Y_{k-1} such that $\hat{D}_k$ is not required. The resulting modification of G-fADI-R for solving the cross-Gramian equation (27) is given in Algorithm 5.

4.2 Symmetric Left and Unsymmetric Right Hand Side

A similar special case is the matrix equation ($B = -A^T$, $C = E^T$)

$$AXE^T + EXA^T = FG^T \tag{28}$$

which might be considered as generalized Lyapunov equation with an unsymmetric right hand side. A sufficient condition for a unique solution is again $\Lambda(A,E) \subset \mathbb{C}_-$. Now we might set $\beta_k = -\overline{\alpha_k}$ which yields

$$V_k = (A + \overline{\alpha_k}E)^{-1}\hat{V}_{k-1}, \quad W_k = -(A + \overline{\alpha_k}E)^{-1}\hat{W}_{k-1},$$
$$\hat{V}_k = \hat{W}_{k-1} - 2\,\mathrm{Re}\,(\alpha_k)EV_k, \quad \hat{V}_k = \hat{W}_{k-1} + 2\,\mathrm{Re}\,(\alpha_k)EW_k.$$

Since the coefficient matrices in both linear systems are identical, we can solve for V_k, W_k simultaneously. Case 3) cannot happen and if α_k is a complex shift followed by its complex conjugate, we find using (17a),(17c) that

$$\hat{V}_{k+1} = \hat{V}_{k-1} - 4\,\mathrm{Re}\,(\alpha_k)E\,\mathrm{Re}\,(V_k) + 4\tfrac{\mathrm{Re}\,(\alpha_k)^2}{\mathrm{Im}\,(\alpha_k)}E\,\mathrm{Im}\,(V_k),$$
$$\hat{W}_{k+1} = \hat{W}_{k-1} + 4\,\mathrm{Re}\,(\alpha_k)E\,\mathrm{Re}\,(W_k) - 4\tfrac{\mathrm{Re}\,(\alpha_k)^2}{\mathrm{Im}\,(\alpha_k)}E\,\mathrm{Im}\,(W_k).$$

as well as a similar $2r \times 2r$ augmentation of D_{k-1} by the negative definite matrix

$$\hat{D}_k = -4\,\mathrm{Re}\,(\alpha_k)\begin{bmatrix} 1 & \delta_k \\ \delta_k & 2\delta_k^2+1 \end{bmatrix} \otimes I_r = -4\,\mathrm{Re}\,(\alpha_k)\hat{L}_k\hat{M}_k\hat{L}_k^T,$$
$$\hat{L}_k = \begin{bmatrix} 1 & 0 \\ \delta_k & 1 \end{bmatrix} \otimes I_r, \quad \hat{M}_k = \mathrm{diag}\left(1, \delta_k^2+1\right) \otimes I_r$$

which is the same LDL^T factorization as used in the LR-ADI for generalized Lyapunov equations in [12]. The factors $\hat{L}_k$, $\hat{M}_k$ can again implicitly be multiplied to $[\mathrm{Re}\,(V_k),\ \mathrm{Im}\,(V_k)]$ and $[\mathrm{Re}\,(W_k),\ \mathrm{Im}\,(W_k)]$. The resulting G-fADI-R for solving (28) is given in Algorithm 6. Note that the conjugation of α_k in the linear system has been revoked since it is not important in which order the shifts of a complex conjugated pair are processed.

4.3 Discrete-time Lyapunov Equations

It is obvious that (2) can be seen as discrete-time Sylvester equation such that also those equations can be solved by G-fADI and its real version. Choosing $C = A^T$, $B = E^T$ and $G = -F$ leads to a generalized discrete-time Lyapunov or Stein equation

$$AXA^T - EXE^T = -FF^T, \tag{29}$$

which has a unique symmetric positive (semi)definite solution if $|\lambda_j| < 1$ for all $\lambda_j \in \Lambda(A,E)$. There is already some work on solving large-scale Stein equations, e.g., by Krylov subspace [42], Smith [8, 37], or ADI type methods [7],[42, Section 6.3]. Here we follow a rather unconventional approach by solving (29) with the G-fADI iteration for generalized Sylvester equations. There, some simplifications occur as follows where we assume that $0 \notin \Lambda(A,E)$ and $\alpha_k \neq 0\ \forall k$. Since $\Lambda(B,C) = \Lambda(E^T,A^T) = 1/\overline{\Lambda(A,E)}$, setting $\beta_k = 1/\overline{\alpha_k}$ appears to be the natural choice. Then the V_k, $\hat{V}_k$ are given by

$$V_k = (A - \beta_k E)^{-1}\hat{V}_{k-1} = (A - \tfrac{\alpha_k}{|\alpha_k|^2}E)^{-1}\hat{V}_{k-1},$$
$$\hat{V}_k = \hat{V}_{k-1} + \alpha_k\tfrac{1-|\alpha_k|^2}{|\alpha_k|^2}EV_k.$$

Algorithm 6: G-fADI-R for generating real low-rank solutions of (28)

Input : $A,\ E,\ F,\ G$ as in (28) and proper shift parameters $\{\alpha_1, \ldots, \alpha_{k_{\max}}\} \subset \mathbb{C}_-$, tolerance $0 < \tau \ll 1$.

Output: $Z_{k_{\max}} \in \mathbb{R}^{n \times rk_{\max}}$, $Y_{k_{\max}} \in \mathbb{R}^{n \times rk_{\max}}$ such that $Z_{k_{\max}} Y_{k_{\max}}^T \approx X$.

$\hat{V}_0 = F,\ \hat{W}_0 = G,\ k = 1.$

while $\|\hat{V}_{k-1}\hat{W}_{k-1}^T\| \geq \tau \|FG^T\|$ **do**

  $[V_k, W_k] = (A + \alpha_k E)^{-1}[\hat{V}_{k-1}, -\hat{W}_{k-1}].$

  if $\alpha_k \in \mathbb{R}$ **then**

    $[\hat{V}_k, \hat{W}_k] = [\hat{V}_{k-1}, \hat{W}_{j-1}] + 2\alpha_k E[-V_k, W_k].$

    $Z_k = [Z_{k-1}, \sqrt{-2\alpha_k} V_k],\ Y_k = [Y_{k-1}, \sqrt{-2\alpha_k} W_k],\ k = k + 1.$

  else

    $\delta_k := \frac{\operatorname{Re}(\alpha_k)}{\operatorname{Im}(\alpha_k)},\ \phi_k = 2\sqrt{-\operatorname{Re}(\alpha_k)}$

    $\hat{V}_{k+1} = \hat{V}_{k-1} + \phi_k^2 E\left(\operatorname{Re}(V_k) - \delta_k \operatorname{Im}(V_k)\right)$

    $\hat{W}_{k+1} = \hat{W}_{k-1} - \phi_k^2 E\left(\operatorname{Re}(W_k) - \delta_k \operatorname{Im}(W_k)\right).$

    $Z_{k+1} = \left[Z_{k-1}, \phi_k\left(\operatorname{Re}(V_k) + \delta_k \operatorname{Im}(V_k)\right), \phi_k\sqrt{1+\delta_k^2}\operatorname{Im}(V_k)\right].$

    $Y_{k+1} = \left[Y_{k-1}, \phi_k\left(\operatorname{Re}(W_k) + \delta_k \operatorname{Im}(W_k)\right), \phi_k\sqrt{1+\delta_k^2}\operatorname{Im}(W_k)\right].$

    $k = k + 2.$

For $W_k,\ \hat{W}_k$ we have at first

$$W_1 = (B - \alpha_1 C)^{-H} G = (\overline{\alpha_1} A - E)^{-1} F = -\tfrac{1}{\overline{\alpha_1}} V_1,$$
$$\hat{W}_1 = -F + \tfrac{1-|\alpha_1|^2}{\alpha_1} A W_1 = -F + \tfrac{1-|\alpha_1|^2}{|\alpha_1|^2} A V_1 = -\tfrac{1}{|\alpha_1|^2}\hat{V}_1,$$

where we used that $AV_k = \frac{\alpha_k}{|\alpha_k|^2} E V_k + \hat{V}_{k-1} = \frac{1}{1-|\alpha_k|^2}\hat{V}_k - \frac{|\alpha_k|^2}{1-|\alpha_k|^2}\hat{V}_{k-1}$. Subsequently applying this procedure for $k = 2, 3, \ldots$ eventually yields

$$W_k = \tfrac{\theta_{k-1}}{\overline{\alpha_k}} V_k, \quad \hat{W}_k = -\theta_k \hat{V}_k$$

with $\theta_k := \left(|\alpha_1|^2 \cdot \ldots \cdot |\alpha_k|^2\right)^{-1}$, $\theta_0 = 1$. Hence, $W_k,\ \hat{W}_k$ as well as the solution factor Y_k are not required. Accumulating the constants in front of W_k in the above expressions into the D-factor leads to the iteration scheme

$$V_k = (A - \tfrac{\alpha_k}{|\alpha_k|^2} E)^{-1}\hat{V}_{k-1}, \quad \hat{V}_k = \hat{V}_{k-1} + \alpha_k \tfrac{1-|\alpha_k|^2}{|\alpha_k|^2} E V_k,$$
$$Z_k = [Z_{k-1}, V_k], \quad D_k = \operatorname{diag}\left(D_{k-1}, (1 - |\alpha_k|^2)\theta_k I_r\right),\ \theta_k = \tfrac{\theta_{k-1}}{|\alpha_k|^2}.$$

The spectral norm of the residual can be computed via $\|R_k\| = |\theta_k| \|\hat{V}_k\|^2$. Note that this iteration is different from the ADI iteration for (29) presented in [7, 42]. Now let $\hat{V}_{k-1} \in \mathbb{R}^{n \times r}$ and $\alpha_k,\ \alpha_{k+1} = \overline{\alpha_k} \in \mathbb{C}$. By using similar techniques as in Section 3 we

Algorithm 7: LR-ADI-R for Stein equations (29)

Input : $A,\ E,\ F$ as in (29) and proper shift parameters $\{\alpha_1, \ldots, \alpha_{k_{\max}}\}$ with $0 < |\alpha_k| < 1$, tolerance $0 < \tau \ll 1$.

Output: $Z_{k_{\max}} \in \mathbb{R}^{n \times r k_{\max}}$ such that $Z_{k_{\max}} Z_{k_{\max}}^T \approx X$.

1 $\hat{V}_0 = F,\ \theta_0 = 1,\ k = 1.$

2 **while** $|\theta_{k-1}|^2 \|\hat{V}_{k-1}\|^2 \geq \tau \|F\|^2$ **do**

3 $\quad V_k = (A - \frac{\alpha_k}{|\alpha_k|^2} E)^{-1} \hat{V}_{k-1}.$

4 $\quad$ **if** $\alpha_k \in \mathbb{R}$ **then**

5 $\quad\quad \hat{V}_k = \hat{V}_{k-1} + \frac{1-\alpha_k^2}{\alpha_k} E V_k,\ \ \theta_k = \frac{\theta_{k-1}}{\alpha_k^2}.$

6 $\quad\quad Z_k = [Z_{k-1}, \sqrt{(1-\alpha_k^2)\theta_k} V_k],\ k = k+1.$

7 $\quad$ **else**

8 $\quad\quad \hat{V}_{k+1} = \hat{V}_{k-1} + \frac{2\operatorname{Re}(\alpha_k)(1-|\alpha_k|^2)}{|\alpha_k|^2} E \operatorname{Re}(V_k)$

9 $\quad\quad\quad + \frac{(1-|\alpha_k|^2)(|\alpha_k|^2 - |\alpha_k|^4 - 2\operatorname{Im}(\alpha_k)^2)}{|\alpha_k|^2 \operatorname{Im}(\alpha_k)} E \operatorname{Im}(V_k).$

10 $\quad\quad \theta_{k+1} = \frac{\theta_{k-1}}{|\alpha_k|^4},\ \delta_k := \frac{\operatorname{Re}(\alpha_k)}{\operatorname{Im}(\alpha_k)},\ \ell_1 := \sqrt{1-|\alpha_k|^4},\ \ell_2 := \ell_1^{-1} \delta_k (1-|\alpha_k|^2)^2,$
$\quad\quad \ell_3 := \sqrt{(1-|\alpha_k|^2)(|\alpha_k|^2 + \delta_k^2(1-2|\alpha_k|^2) + |\alpha_k|^4(1+\delta_k^2)) - \ell_2^2}.$

11 $\quad\quad Z_{k+1} = \left[Z_{k-1}, \sqrt{\theta_{k+1}} \left(\ell_1 \operatorname{Re}(V_k) + \ell_2 \operatorname{Im}(V_k)\right), \sqrt{\theta_{k+1}} \ell_3 \operatorname{Im}(V_k)\right].$

12 $\quad\quad k = k+2.$

obtain

$$V_{k+1} = \overline{V_k} + \frac{\alpha_k(1-|\alpha_k|^2)}{\operatorname{Im}(\alpha_k)} \operatorname{Im}(V_k),$$
$$\hat{V}_{k+1} = \hat{V}_{k-1} + \frac{2\operatorname{Re}(\alpha_k)(1-|\alpha_k|^2)}{|\alpha_k|^2} E \operatorname{Re}(V_k) + \frac{(1-|\alpha_k|^2)(|\alpha_k|^2 - |\alpha_k|^4 - 2\operatorname{Im}(\alpha_k)^2)}{|\alpha_k|^2 \operatorname{Im}(\alpha_k)} E \operatorname{Im}(V_k),$$

and

$$\hat{D}_k := \sqrt{(1-|\alpha_k|^2)\theta_{k+1}} \begin{bmatrix} 1+|\alpha_k|^2 & \delta_k(1-|\alpha_k|^2) \\ \delta_k(1-|\alpha_k|^2) & |\alpha_k|^2 + \delta_k^2(1-2|\alpha_k|^2) + |\alpha_k|^4(1+\delta_k^2) \end{bmatrix} \otimes I_r \in \mathbb{R}^{2r \times 2r}$$

with $\theta_{k+1} = \frac{\theta_{k-1}}{|\alpha_k|^4},\ \delta_k := \frac{\operatorname{Re}(\alpha_k)}{\operatorname{Im}(\alpha_k)}$. It can be easily shown that $\hat{D}_k$ is positive definite, provided $|\alpha_k| < 1$, and its Cholesky factor can implicitly be multiplied to $[\operatorname{Re}(V_k),\ \operatorname{Im}(V_k)]$ such that $\hat{D}_k$ is also not required. The resulting algorithm for computing real solutions factors for (29) in given in Algorithm 7. The constants $\ell_{1,2,3}$ in step 10 are the entries of the Cholesky factors of $\hat{D}_k(\sqrt{\theta_{k+1}})^{-1}$.

Note that Algorithm 7 can be rewritten such that the coefficient matrices in the linear systems are $|\alpha_k|^2 E - \alpha_k A$ or $\alpha_k E - A$. This will essentially introduce other constants in the defining equations for V_k, $\hat{V}_k$ leading also to slightly other entries in $\hat{D}_k$. Which choice will be the best, e.g. w.r.t. numerical robustness, might a topic for further research.

The construction of θ_k constitutes a weakness of Algorithm 7 since it can easily become extremely large if $|\alpha_k| \ll 1$ for some shifts. This was also observed in some

Table 1: The parameters for the fADI method and its modifications.

Parameter	Meaning
$k_+^A,\ k_-^A$	number of Arnoldi steps w.r.t. $E^{-1}A$ and $A^{-1}E$
$k_+^B,\ k_-^B$	number of Arnoldi steps w.r.t. $C^{-1}B$ and $B^{-1}C$
$J = J_{\mathbb{R}} + 2J_{\mathbb{C}}$	number of shifts (real ones plus complex pairs) w.r.t. A, E
$L = L_{\mathbb{R}} + 2L_{\mathbb{C}}$	number of shifts (real ones plus complex pairs) w.r.t. B, C
$\epsilon_{\text{res}} = 10^{-10}$	tolerance for normalized residual

numerical examples. Moreover, the case $|\alpha_k| \approx 1$ can lead to cancellation. We plan to pursue the circumvention of these issues in future work regarding ADI methods for Stein equations.

If the right hand side of (29) is of the form FG^T, similar techniques as for (28) in Section 4.2 can be applied to construct a modified version of Algorithm 7.

5 Numerical Examples

In this section we test the considered algorithms in their complex and real implementation for different types of Sylvester equations. All experiments have been carried out in MATLAB® 7.11.0 on an Intel®Xeon®W3503 CPU with 2.40 GHz and 6 GB RAM. The occurring linear systems were solved with the MATLAB backslash operator which employs sparse direct techniques. The algorithms were terminated when $\|R_k\|/\|FG^T\| < \epsilon_{\text{res}}$, $\epsilon_{\text{res}} = 10^{-10}$, where the residual norm was computed using the novel relation (16). For the shift parameters we ran k_+^A and k_-^A steps of an Arnoldi process w.r.t. $E^{-1}A$ and $A^{-1}E$, respectively. From the resulting $k_+^A + k_-^A$ Ritz values J were selected consisting of $J_{\mathbb{R}}$ real and $J_{\mathbb{C}}$ pairs of complex conjugated shifts. The generation of shifts corresponding to B, C was carried out similarly. Table 1 summarizes these and all other parameters required for the methods. If the number of required iterations exceeded the number of shifts J or K, the respective shifts were used in a cyclical manner. Note that we did not apply the heuristic shift selection strategy via the approximate solution of (15) since taking the generated Ritz values alone works surprisingly well for our examples. In some cases this approach worked even better than selecting shifts via (15). Moreover, it is adequate enough for showing efficiency improvements of the algorithms which compute real low-rank solution factors. We used the following test examples, where the majority of them were constructed entirely for testing purposes without any background in applications.

Example 1. To get a standard Sylvester equation we use, similar as in [32, Example 2], 5-point discretizations of the operator

$$L(x) := \Delta x - f_1(\xi_1, \xi_2)\frac{\partial x}{\partial \xi_1} - f_2(\xi_1, \xi_2)\frac{\partial x}{\partial \xi_2} - f_3(\xi_1, \xi_2)x \quad \text{on} \quad \Omega = (0,1)^2,$$

where $x = x(\xi_1,\ \xi_2)$ and with imposed homogeneous Dirichlet boundary conditions. The matrix A is obtained from using 80 equidistant grid points for each spatial di-

mension and $f_1(\xi_1,\xi_2) = \mathrm{e}^{\xi_1+\xi_2}$, $f_2(\xi_1,\xi_2) = 1000\xi_2$, $f_3(\xi_1,\xi_2) = \xi_1$. Likewise, B is obtained from discretizing $-L(x)$ using 60 grid points and $f_1(\xi_1,\xi_2) = \sin(\xi_1 + 2\xi_2)$, $f_2(\xi_1,\xi_2) = 20\mathrm{e}^{\xi_1+\xi_2}$, $f_3(\xi_1,\xi_2) = \xi_1\xi_2$. The right hand side factors F, G are random matrices with $r = 4$ columns.

Example 2. Similar as Example 1, but now with 110 grid points and $f_1(\xi_1,\xi_2) = \mathrm{e}^{\xi_1\xi_2}$, $f_2(\xi_1,\xi_2) = \sin(\xi_1\xi_2)$, $f_3(\xi_1,\xi_2) = \xi_2^2 - \xi_1^2$ for A, as well as 30 grid points and $f_1(\xi_1,\xi_2) = 100\mathrm{e}^{\xi_1}$, $f_2(\xi_1,\xi_2) = 10\xi_1\xi_2$, $f_3(\xi_1,\xi_2) = \sqrt{\xi_2^2 + \xi_1^2}$ to define B. This is essentially [17, Example 1].

Example 3. We use the IFISS 3.2 package [43] to discretize $\dot{x} = L(x)$ on $(0,1)^2$ by Q1 finite elements on a uniform grid. The same settings as in the IFISS example T-CD3 are used to set the functions f_1, f_2 and the boundary conditions. Motivated by the Sylvester example in [6], two different grid sizes were used to generate A, E and B, C: a 128×128 and a 64×64 grid, respectively, and F, G are random matrices with $r = 5$ columns.

The next four examples represent the special Sylvester equations considered in Section 4. They were treated with both G-fADI(-R) and the adapted methods proposed there to show additional performance gains from exploiting their special structure.

Example 4. We take the same settings as in Example 1 to generate A but now with 100 grid points and $r = 1$ and set $B = -A$. The right hand side factors F, G are random vectors. Since the resulting equation is a cross-Gramian Sylvester equation of the form (27), we also apply the specially tailored modification of Section 4.1 (Algorithm 5).

Example 5. The same data as in Example 4 but now $B = -A^T$ which leads to an equation of the form (5) such that Algorithm 6 is applied as well.

Example 6. As in [42, Example 1] we set

$$A = C^T = \begin{bmatrix} 0 & \vartheta & & & \\ -\vartheta & 0 & \vartheta & & \\ & -\vartheta & 0 & & \\ & & & \ddots & \vartheta \\ & & & -\vartheta & 0 \end{bmatrix}, \quad F = -G = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}, \quad B = E = I_n$$

with $n = 50000$ and $\vartheta = 0.49$. This leads to a Stein equation (29) which is also handled by Algorithm 7.

Example 7. Let A, F from Example 4 define a Lyapunov equation $AX + XA^T = -FF^T$. Then X is also the solution of a generalized Stein equation defined by $\tilde{A} = \eta I_n - A$, $\tilde{E} = \eta I_n + A$ and $\tilde{F} = \sqrt{2\eta}F$ with $\eta = 10^3$. As in the previous example, we apply both G-fADI(-R) and Algorithm 7. See [42, Example 2] for a similar experiment.

The dimensions of the considered Sylvester equations, the settings for the shift parameter computation, the number of the required fADI iterations k^{it} until termination, and the computation times $t_{\mathbb{C}}$, $t_{\mathbb{R}}$ for the complex as well as real fADI algorithms are summarized in Table 2. It is evident that for all examples the methods computing real

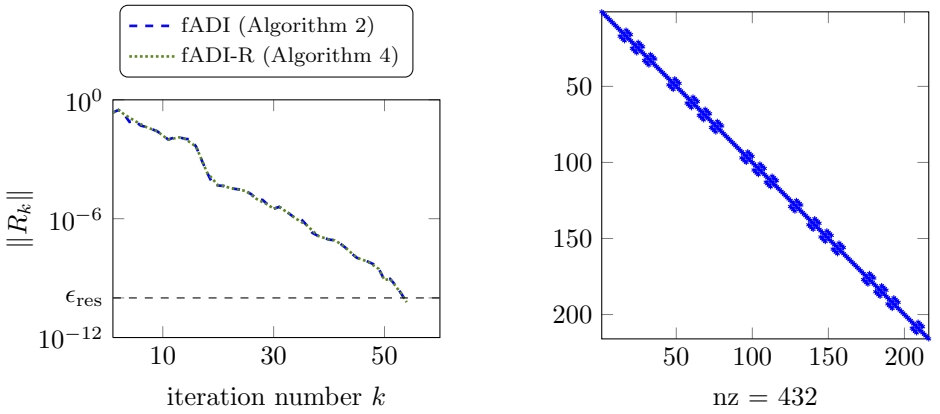
Table 2: Parameters, required fADI iterations k^{it} and computation times $t_{\mathbb{C}}$, $t_{\mathbb{R}}$ in seconds of basic and real implementations.

Ex.	n, m, r	$k_+^A, k_-^A, k_+^B, k_-^B$	$J(J_{\mathbb{R}}, J_{\mathbb{C}}), L(L_{\mathbb{R}}, L_{\mathbb{C}})$	k^{it}	$t_{\mathbb{C}}$	$t_{\mathbb{R}}$
1	6400,3600,4	10,10,10,10	20 (8,6), 20 (12,4)	54	8.24	4.72
2	12100,900,4	10,10,5,5	20 (18,1), 10 (2,4)	43	12.4	6.12
3	16641,4225,5	10,20,10,20	30 (12,9), 30 (12,9)	84	54.59	29.15
4	10000,-,1	10,10,-	20 (8,6), -	140	33.52	20.27
5	10000,-,1	10,10,-	20 (8,6), -	133	31.58	19.73
6	50000,-,2	10,0,-	10 (0,5), -	68	23.59	10.89
7	10000,-,1	10,0,-	10 (2,4),-	262	66.09	44.05
4	G-fADI iteration for (27), Algorithm 5			140	33.68	20.10
5	G-fADI iteration for (28), Algorithm 6			133	19.29	12.00
6	G-fADI iteration for (29), Algorithm 7			68	11.62	6.12
7	G-fADI iteration for (29), Algorithm 7			262	32.08	22.08

solution factors required, depending on the number of processed complex shifts, substantially less time. The obtained solutions were almost identical, except for possible rounding errors.

This can also be seen in the residual history for Example 1 shown in Figure 1(a). Both the complex (Algorithm 2) and real version (Algorithm 4) have almost identical residual norms throughout the iteration. Only minor deviations occurred exactly when the complex method is in between processing a pair of complex conjugated shifts, i.e., cases 2) or 3). This is a similar observation as in [12, Example 1, Figure 1]. From the sparsity pattern of the matrix D_k in Figure 1(b), one can see that the cases 2) and 3) were encountered 18 times which is revealed by the respective number of $2r \times 2r$ blocks along the diagonal.

Before we proceed we investigate, using Example 3, the performance of the computation of the (normalized) Sylvester residual norm in G-fADI(-R) (Algorithms 3 and 4) exploiting the low-rank structure given bin Theorem 5 by using (16a),(16b). For comparison we also compute the residual norm with a Lanczos process applied to $R_k^H R_k$ using the `eigs` command in MATLAB. Figure 2 illustrates the history of the required runtime of both approaches as G-fADI and G-fADI-R proceed. Obviously, using the Lanczos process is drastically more expensive (177.8 / 65.42 seconds) than our novel approach via the low-rank structure of R_k (0.41 / 0.13 seconds). It even considerably exceeds the runtime of the main computations (54.59 / 29.15 seconds) of G-fADI(-R). One can also clearly see that the Lanczos approach gets increasingly expensive as the iterations proceed due to the increasing dimensions of the low-rank solution factors. Obviously, the total runtimes are smaller in G-fADI-R because the residual norm is only computed once after a complex pair of shifts has been processed and not twice as in G-fADI. Furthermore, Figure 2 reveals that the times for computing $\|R_k\|$ at each iteration k are smaller in G-fADI-R since only real data is involved there whereas $\hat{V}_k$, $\hat{W}_k$ in G-fADI might have (rounding error induced) non-zero imaginary parts, although



(a) Residual norm history.

(b) Sparsity pattern of D_k generated by fADI-R.

Figure 1: Results for Example 1. (a) Residuals norms against iteration number k for fADI and fADI-R. (b) Sparsity pattern of the matrix D_k in fADI-R.

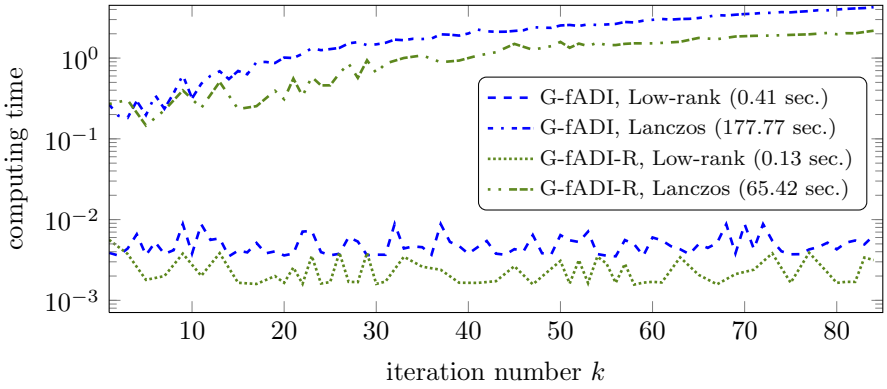


Figure 2: Time (in seconds) needed for computing the (normalized) Sylvester residual norm R_k for Example 3 in G-fADI-R against the iteration number k, where $\|R_k\|$ was computed with a Lanczos process for $R_k^H R_k$ as well as the novel approach exploiting the low-rank representation (8),(13) of R_k. The total times are given in brackets behind the respective entry in the legend.

both quantities are real (cf. Theorem 6, Corollary 7). Similar observations, regarding both the efficiency of the real (G-)fADI iteration and the residual norm computation, were also made for the other examples. Similar results concerning the LR-ADI for generalized Lyapunov equations can be found in [13].

The lines 4–7 of Table 2 belong to the special Sylvester equations of the Examples

4–7. They are solved directly by (G-)fADI and its real modification (G-)fADI-R. The last four lines of Table 2 give the timings for the structure exploiting modifications and their real implementations given in Algorithms 5–7 in Section 4.

For the cross-Gramian Sylvester equation of Example 4 there is no significant difference in the timings of fADI(-R) and Algorithm 5. We did not reuse the LU factors of $A - \alpha E$ in the solution of both adjoint linear systems since computing those with the `lu` command in MATLAB was more costly than solving both systems separately with the backslash command[1].

For Example 6 the application of Algorithm 6 leads to a significant reduction of the computation time. This is mainly because the factorization of the shifted linear system has to be computed only once for constructing both V_k and W_k.

A decrease of the run times is also apparent for Examples 6 and 7 when applying Algorithm 7 instead of G-fADI(-R). However, in some cases the algorithm broke down when some of the α_k were too small in magnitude which resulted in an overflow when computing the value θ_k. Hence, we set $k_-^A = 0$ and used only the Ritz values approximating the eigenvalues with the largest magnitude. Note that for Example 7 the approximate solution can be obtained more easily by applying LR-ADI-R [12, Algorithm 3] to the equivalent Lyapunov equation which required 230 iterations and about 16 seconds.

6 Conclusions and Outlook

In this work we investigated the factored ADI method [15, 36] for solving large-scale generalized Sylvester equations. We extended some results for Lyapunov equations [31, 23, 13] which resulted in a new low-rank representation of the residual matrix R_k. This enables a cheap way to compute the residual norm. The main part of this paper is devoted to the handling of complex shift parameters. This lead to extensions of the result [12, Theorem 1] which allows to formulate an fADI iteration (Algorithm 4) that constructs real low-rank solution factors by employing significantly reduced amounts of complex arithmetic operations and storage. The occurring formulas are more complicated compared to the low-rank ADI for Lyapunov equations. Numerical experiments show a clear reduction of the computation time if complex shifts are handled properly. The treatment of special generalized Sylvester equations such as cross-Gramian equations, equations with a symmetric left but unsymmetric right hand side, and even Stein equations was also discussed. Specially tailored modifications of G-fADI have been proposed which also incorporate a proper handling of complex shifts. These structure exploiting variants can lead to an additional efficiency gain as shown in the numerical examples.

In this paper we did not extensively focus on the efficient generation of shift parameters of high quality. This might be regarded as very pressing matter in context to ADI type methods. The adaptive computation of shifts during the ADI iteration is

[1] The reason is, that to out knowledge, `lu` uses a different factorization algorithm then \.

currently investigated. The use of the derived low-rank ADI iteration for Stein equations within the Newton's method for discrete-time algebraic Riccati equations [7] will be investigated soon. Another obvious future research perspective is the application of the proposed ADI methods to solve the Sylvester equations which occur in Newton methods for nonsymmetric algebraic Riccati equations

$$AXC + EXB - EXDXC + H = 0,$$

see, e.g., [19, Listing 3.11]. This might lead to generalizations of the low-rank Newton type ADI methods [14, 31, 23] available for algebraic Riccati equations.

References

[1] L. Bao, Y. Lin, and Y. Wei, *Krylov subspace methods for the generalized Sylvester equation*, Applied Mathematics and Computation, 175 (2006), pp. 557–573.

[2] ——, *A new projection method for solving large Sylvester equations*, Applied Numerical Mathematics, 57 (2007), pp. 521–532. DOI:10.1016/j.apnum.2006.07.005.

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

[4] P. Benner, *Solving large-scale control problems*, IEEE Control Systems Magazine, 14 (2004), pp. 44–59.

[5] ——, *The matrix factorization paradigm in solving matrix equations.* Householder Symposium XVI, Seven Springs Mountain Resort, Champion, Pennsylvania, USA, see http://www.mpi-magdeburg.mpg.de/mpcsc/benner/talks/hh05.pdf., 2005.

[6] P. Benner and T. Breiten, *On optimality of interpolation-based low rank approximations of large-scale matrix equations*, Tech. Rep. MPIMD/11-10, Max Planck Institute Magdeburg Preprints, December 2011. available from http://www.mpi-magdeburg.mpg.de/preprints/2011/10/.

[7] 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.

[8] P. Benner, G. E. Khoury, and M. Sadkane, *On the Squared Smith Method for Large-Scale Stein Equations*, Tech. Rep. MPIMD/12-15, Max Planck Institute Magdeburg Preprints, September 2012. Available from http://www.mpi-magdeburg.mpg.de/preprints/.

[9] P. Benner, M. Köhler, and J. Saak, *Sparse-Dense Sylvester equations in $\mathcal{H}_2$-model order reduction*, Tech. Rep. MPIMD/11-11, Max Planck Institute Magdeburg Preprints, December 2011. available from http://www.mpi-magdeburg.mpg.de/preprints/2011/11/.

[10] P. Benner, P. Kürschner, and J. Saak, *Avoiding complex arithmetic in the low-rank ADI method efficiently*, Proc. Appl. Math. Mech., 12 (2012), pp. 639–640.

[11] ——, *Real versions of low-rank ADI methods with complex shifts*, Tech. Rep. MPIMD/12-11, Max Planck Institute Magdeburg Preprints, 2012. Available from http://www.mpi-magdeburg.mpg.de/preprints/.

[12] ——, *Efficient Handling of Complex Shift Parameters in the Low-Rank Cholesky Factor ADI Method*, Numerical Algorithms, 62 (2013), pp. 225–251.

[13] ——, *An improved numerical method for balanced truncation for symmetric second order systems*, Mathematical and Computer Modelling of Dynamical Systems , (2013).

[14] 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.

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

[16] 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.

[17] 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.

[18] D. Calvetti and L. Reichel, *Application of ADI iterative methods to the restoration of noisy images*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 165–186.

[19] B. A. Dario, I. Bruno, and B. Meini, *Numerical Solution of Algebraic Riccati Equations*, Society for Industrial and Applied Mathematics, 2011.

[20] D. Day and M. A. Heroux, *Solving complex-valued linear systems via equivalent real formulations*, SIAM J. Sci. Comput., 23 (2001), pp. 480–498.

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

[22] M. Epton, *Methods for the solution of $AXD - BXC = E$ and its application in the numerical solution of implicit ordinary differential equations*, BIT, 20 (1980), pp. 341–345.

[23] 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.

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

[25] J. Gardiner, A. J. Laub, J. Amato, and C. Moler, *Solution of the Sylvester matrix equation $AXB + CXD = E$*, ACM Trans. Math. Software, 18 (1992), pp. 223–231.

[26] 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.

[27] L. Grasedyck, *Existence of a low rank or H-matrix approximant to the solution of a Sylvester equation*, Numer. Lin. Alg. Appl., 11 (2004), pp. 371–389.

[28] S. Hammarling, *Numerical solution of the stable, non-negative definite Lyapunov equation*, IMA J. Numer. Anal., 2 (1982), pp. 303–323.

[29] R. Horn and C. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1991.

[30] D. Hu and L. Reichel, *Krylov-subspace methods for the Sylvester equation*, Linear Algebra Appl., 172 (1992), pp. 283–313.

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

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

[33] P. Lancaster and M. Tismenetsky, *The Theory of Matrices*, Academic Press, Orlando, 2nd ed., 1985.

[34] J.-R. Li, *Model Reduction of Large Linear Systems via Low Rank System Gramians*, PhD thesis, Massachusettes Institute of Technology, September 2000.

[35] J.-R. Li and J. White, *Low rank solution of Lyapunov equations*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 260–280.

[36] 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.

[37] T. Li, P. C.-Y. Weng, E. K. wah Chu, and W.-W. Lin, *Large-scale Stein and Lyapunov equations, Smith method, and applications*, Numerical Algorithms, (2012).

[38] T. Penzl, *A cyclic low rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 21 (2000), pp. 1401–1418.

[39] 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).

[40] 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.

[41] 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.

[42] M. Sadkane, *A low-rank Krylov squared Smith method for large-scale discrete-time Lyapunov equations*, Linear Algebra and its Applications, 436 (2012), pp. 2807–282.

[43] D. Silvester, H. Elman, and A. Ramage, *Incompressible Flow and Iterative Solver Software (IFISS) version 3.2*, May 2012.

[44] D. Sorensen and A. Antoulas, *The Sylvester equation and approximate balanced reduction*, Linear Algebra and its Applications, 351–352 (2002), pp. 671–700.

[45] D. Sorensen and Y. Zhou, *Bounds on eigenvalue decay rates and sensitivity of solutions to Lyapunov equations*, Tech. Rep. TR02-07, Dept. of Comp. Appl. Math., Rice University, Houston, TX, June 2002. Available online from http://www.caam.rice.edu/caam/trs/tr02.html.

[46] ——, *Direct methods for matrix Sylvester and Lyapunov equations*, J. Appl. Math, 2003 (2003), pp. 277–303.

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