

# Truncated low-rank methods for solving general linear matrix equations

Daniel Kressner and Petar Sirković<sup>\*,†</sup>

<sup>1</sup>Chair of Numerical Algorithms and HPC, MATHICSE, EPF Lausanne, CH-1015 Lausanne, Switzerland

## SUMMARY

This work is concerned with the numerical solution of large-scale linear matrix equations  $A_1 X B_1^T + \dots + A_K X B_K^T = C$ . The most straightforward approach computes  $X \in \mathbb{R}^{m \times n}$  from the solution of an  $mn \times mn$  linear system, typically limiting the feasible values of  $m, n$  to a few hundreds at most. Our new approach exploits the fact that  $X$  can often be well approximated by a low-rank matrix. It combines greedy low-rank techniques with Galerkin projection and preconditioned gradients. In turn, only linear systems of size  $m \times m$  and  $n \times n$  need to be solved. Moreover, these linear systems inherit the sparsity of the coefficient matrices, which allows to address linear matrix equations as large as  $m = n = O(10^5)$ . Numerical experiments demonstrate that the proposed methods perform well for generalized Lyapunov equations. Even for the case of standard Lyapunov equations, our methods can be advantageous, as we do not need to assume that  $C$  has low rank. Copyright © 2015 John Wiley & Sons, Ltd.

Received 26 February 2014; Revised 19 January 2015; Accepted 30 January 2015

KEY WORDS: general linear matrix equation; Lyapunov equation; greedy low-rank; generalized Lyapunov equation; Galerkin projection

## 1. INTRODUCTION

We consider the numerical solution of large-scale linear matrix equations of the form

$$\sum_{k=1}^K A_k X B_k^T = C, \quad (1)$$

for given coefficient matrices  $A_1, \dots, A_K \in \mathbb{R}^{m \times m}$ ,  $B_1, \dots, B_K \in \mathbb{R}^{n \times n}$ , and  $C \in \mathbb{R}^{m \times n}$ . Equation can also be seen as a linear system

$$\sum_{k=1}^K (B_k \otimes A_k) \text{vec}(X) =: \mathcal{A} \text{vec}(X) = \text{vec}(C). \quad (2)$$

The matrix equation (1) is uniquely solvable if and only if  $\mathcal{A} \in \mathbb{R}^{mn \times mn}$  is invertible, which will be assumed throughout the whole paper.

For  $K = 2$ , the matrix equation (1) reduces to the so called *generalized Sylvester equation*, which includes the standard Sylvester equation  $A_1 X + X B_2^T = C$  and the *Lyapunov equation*

\*Correspondence to: Petar Sirković, Chair of Numerical Algorithms and HPC, MATHICSE, EPF Lausanne, CH-1015 Lausanne, Switzerland.

†E-mail: petar.sirkovic@epfl.ch

$A_1 X + X A_1^T = -C$ , with  $C$  symmetric positive definite, as particularly important special cases. The efficient numerical solution of Lyapunov and Sylvester equations has been studied intensively during the last decades, and significant progress has been made; we refer to the work in [1, 2] for recent surveys. In particular, a number of approaches have been developed for  $K = 2$  that avoid the explicit computation and storage of the  $m \times n$  matrix  $X$ , but attempt to compute a low-rank approximation to  $X$  and store the low-rank factors only. Of course, this requires that  $X$  can be well approximated by a low-rank matrix at all, that is, the singular values of  $X$  have a strong decay. Such a decay has been shown for a low-rank right-hand side  $C$  in [3–5].

None of the established methods for Lyapunov and Sylvester equations generalizes to the case  $K > 2$ . In fact, the recent survey paper by Simoncini [2] states that *The efficient numerical solution to ... [reference to equation (1)] thus represents the next frontier for linear matrix equations ...* Among the existing work addressing  $K > 2$ , particular attention has been paid to the *generalized Lyapunov equation*

$$AX + XA^T + \sum_{k=1}^K N_k X N_k^T = -DD^T. \quad (3)$$

In fact, this appears to be the most frequently encountered instance of (1) for  $K > 2$  and typically arises in connection with bilinear dynamical systems. By extending results for the Lyapunov case, singular value decay bounds for  $X$  have been established in [6, 7], under various conditions on  $A$  and  $N_k$ .

For the special case of Lyapunov equation, the solution can be found numerically using Alternating Direction Implicit (ADI)-preconditioned Krylov subspaces [8]. Similarly, for the numerical solution of (3), Damm [9] proposed a fixed point iteration based on the splitting  $\mathcal{L}(X) + \mathcal{N}(X) = -DD^T$  of (3) with the Lyapunov operator  $\mathcal{L} : X \mapsto AX + XA^T$ . This iteration converges if  $\mathcal{L}$  is the dominant part of (3), that is, the spectral radius of  $\mathcal{L}^{-1}\mathcal{N}$  of smaller than 1.

A rather different approach by Benner and Breiten [6] treats (3) as an  $n^2 \times n^2$  linear system in the entries of  $X$ . Based on ideas from [10, 11], a standard iterative solver, such as CG or Biconjugate Gradient stabilized (BiCGstab), is combined with low-rank truncation of the iterates. This approach requires the availability of a preconditioner to ensure fast convergence. There is evidence [11] that fast convergence is crucial to avoid an excessive growth of the *numerical* ranks during intermediate iterations. Natural candidates for preconditioners are  $\mathcal{L}$  or approximations thereof, such as one iteration of the ADI method, especially if  $\mathcal{L}$  is the dominant part. Numerical experiments reported in [6] demonstrate that this approach performs remarkably well.

In this paper, we develop a framework of low-rank methods for addressing the general linear matrix equation (1). Our approach is very much inspired by a class of methods proposed in [12, 13] for solving Fokker–Planck equations and stochastic partial differential equations; see [14] for a survey of recent developments. The basic idea is to subsequently refine the current approximation to the solution  $X$  by adding a rank-1 correction. This correction is chosen to minimize a certain target functional, which renders the approach a greedy algorithm. As we will see, this basic approach may require further improvement to perform well for a larger range of applications. We will discuss two techniques for improving convergence: adding information from the preconditioned residual, similar to the techniques considered in [15], and performing Galerkin projection.

The rest of this paper is organized as follows. In Section 2, we explain the basic algorithm using greedy rank-1 updates. For the special case of stable symmetric Lyapunov equations, this algorithm is shown to preserve symmetry of the solution. As shown in Section 3, the performance of this basic algorithm is improved by using Galerkin projections. In Section 4, we discuss the incorporation of preconditioners into the method. Finally, a variety of numerical experiments is presented in Section 5.

## 2. BASIC ALGORITHM USING GREEDY RANK-1 UPDATES

In this section, we describe the basic greedy rank-1 strategy for approximating the solution  $X$  of (1). Starting from the zero initial guess  $X_0 = 0$ , a sequence of approximations  $X_1, X_2, X_3, \dots$

with  $\text{rank}(X_j) \leq j$  is constructed as follows. Given the current approximation  $X_j$ , the next approximation takes the form

$$X_{j+1} = X_j + u_{j+1}v_{j+1}^T, \quad (4)$$

where the rank-1 correction  $u_{j+1}v_{j+1}^T$  is chosen to minimize the approximation error. If the system matrix  $\mathcal{A}$  defined in (2) is symmetric positive definite, we may use the energy norm induced by  $\mathcal{A}$  to measure the error. Otherwise, we will use the residual norm. In the following, we will discuss details for these two choices. For notational convenience, we will identify the matrix representation  $\mathcal{A} \in \mathbb{R}^{mn \times mn}$  with the corresponding linear operator

$$\mathcal{A} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}, \quad \mathcal{A} : X \mapsto \sum_{k=1}^K A_k X B_k^T.$$

### 2.1. Symmetric positive definite case

Let us assume that  $\mathcal{A}$  is symmetric positive definite. Then the linear operator  $\mathcal{A}$  induces the scalar product  $\langle Y, Z \rangle_{\mathcal{A}} = \text{tr}(Y^T \mathcal{A}(Z))$  on  $\mathbb{R}^{m \times n}$  along with the corresponding norm  $\|Y\|_{\mathcal{A}} = \sqrt{\langle Y, Y \rangle_{\mathcal{A}}}$ . We choose the correction  $u_{j+1}v_{j+1}^T$  in (4) such that the approximation error measured in this norm is as small as possible. This yields the minimization problem

$$\begin{aligned} \min_{u,v} \|X - X_j - uv^T\|_{\mathcal{A}}^2 &= \min_{u,v} \langle X - X_j - uv^T, X - X_j - uv^T \rangle_{\mathcal{A}} \\ &= \|X - X_j\|_{\mathcal{A}}^2 + \min_{u,v} \langle uv^T, uv^T \rangle_{\mathcal{A}} - 2 \text{tr}(vu^T \mathcal{A}(X - X_j)) \\ &= \|X - X_j\|_{\mathcal{A}}^2 + \min_{u,v} \langle uv^T, uv^T \rangle_{\mathcal{A}} - 2 \text{tr}(vu^T C_j), \end{aligned}$$

where we set  $C_j := \mathcal{A}(X - X_j) = C - \mathcal{A}(X_j)$ . Ignoring the constant term, we thus obtain  $u_{j+1}v_{j+1}^T$  from the minimization of the functional

$$J(u, v) := \langle uv^T, uv^T \rangle_{\mathcal{A}} - 2 \text{tr}(vu^T C_j). \quad (5)$$

Note that  $J$  is convex in each of the two vectors  $u, v$ , but it is not jointly convex. This setting is well suited for the alternating linear scheme (ALS) [16]. Note that a minor complication arises from the nonuniqueness in the representation of  $uv^T$  by the factors  $u, v$ :  $J(u, v) = J(\lambda u, \lambda^{-1}v)$  for any  $\lambda \neq 0$ . In ALS, this can be easily addressed by normalizing the factor that is currently not optimized.

In the first half-iteration of ALS, we consider  $v$  with  $\|v\|_2 = 1$  to be fixed and optimize for  $u$ :

$$\begin{aligned} \hat{u} &= \arg \min_u J(u, v) = \arg \min_u \langle uv^T, uv^T \rangle_{\mathcal{A}} - 2 \text{tr}(vu^T C_j) \\ &= \arg \min_u \sum_{k=1}^K \text{tr}(vu^T A_k uv^T B_k^T) - 2 \text{tr}(vu^T C_j) \\ &= \arg \min_u \sum_{k=1}^K (u^T A_k u)(v^T B_k v) - 2u^T C_j v. \end{aligned} \quad (6)$$

The matrix

$$\hat{A} := \sum_{k=1}^K (v^T B_k v) A_k \quad (7)$$

amounts to  $(v^T \otimes I)\mathcal{A}(v \otimes I)$  and thus inherits the positive definiteness from  $\mathcal{A}$ . Therefore, the solution of the unconstrained linear-quadratic optimization problem (6) is given by the solution of the linear system  $\hat{A}\hat{u} = C_j v$ .

In the second half-iteration of ALS, we fix the normalized  $u \leftarrow \hat{u}/\|\hat{u}\|_2$  and optimize for  $v$ . By the same arguments, the minimizer  $\hat{v}$  is given by the solution of the linear system  $\hat{B}\hat{v} = C_j^T u$ , with

$$\hat{B} := \sum_{k=1}^K (u^T A_k u) B_k. \quad (8)$$

The described procedure is summarized in Algorithm 1.

---

**Algorithm 1** ALS for minimizing (5).

---

Choose random vectors  $u, v$  such that  $\|v\|_2 = 1$ .

**while** not converged **do**

Solve linear system  $\hat{A}\hat{u} = C_j v$  with  $\hat{A}$  defined in (7).

Normalize  $u \leftarrow \hat{u}/\|\hat{u}\|_2$ .

Solve linear system  $\hat{B}\hat{v} = C_j^T u$  with  $\hat{B}$  defined in (8).

Normalize  $v \leftarrow \hat{v}/\|\hat{v}\|_2$ .

**end while**

---

We refer to the work in [16] concerning the local convergence of Algorithm 1. Let us emphasize, however, that in our setting there is no need to let Algorithm 1 converge to high accuracy, and we stop it after a few iterations.

*Remark 2.1*

The system matrices  $\hat{A}$  and  $\hat{B}$  in (7)–(8) are linear combinations of the coefficient matrices  $A_1, \dots, A_K$  and  $B_1, \dots, B_K$ , respectively. They therefore inherit the sparsity of these matrices, which allows to use a sparse direct solver [17] for solving the linear systems in Algorithm 1. In the special case of a Lyapunov equation  $AX + XA^T = C$ , we have

$$\hat{A} = A + (v^T A v)I, \quad \hat{B} = A + (u^T A u)I.$$

*Remark 2.2*

Similar to the discussion in [18], the procedure earlier can be extended to work with rank- $r$  corrections  $UV^T$ , where  $U \in \mathbb{R}^{m \times r}$  and  $V \in \mathbb{R}^{n \times r}$ , instead of rank-1 corrections. The first half-step of ALS then consists of fixing  $V$  (normalized to have orthonormal columns) and optimize for  $U$ . The resulting linear system takes the form of a linear operator equation  $\hat{A}(\hat{U}) = C_j V$  with

$$\hat{A} : \mathbb{R}^{m \times r} \rightarrow \mathbb{R}^{m \times r}, \quad \hat{A} : Y \mapsto \sum_{k=1}^K A_k Y (V^T B_k V)^T. \quad (9)$$

For the special case of a Lyapunov equation, we have  $\hat{A} : Y \mapsto AY + Y(V^T AV)^T$ . After computing a Schur decomposition of the  $r \times r$  matrix  $V^T AV$ , the linear operator equation  $\hat{A}(\hat{U}) = C_j V$  decouples into  $r$  linear systems; see for example [2, Sec. 4.3].

For  $K > 2$ , such a decoupling is usually impossible, and one therefore has to solve an  $mr \times mr$  linear system for the matrix representation  $\hat{A} = \sum_{k=1}^K V^T B_k V \otimes A_k$ . The unfavorable sparsity pattern and the size of  $\hat{A}$  make the application of a sparse direct solver to this linear system expensive; see the work in [19] for a related discussion.

*2.1.1. Basic greedy rank-1 algorithm.* Combining Algorithm 1 with the basic iteration (4) for rank-1 updates leads to Algorithm 2.

Assuming that a fixed number  $\text{alsit}$  of inner iterations in Algorithm 1 is used, Algorithm 2 requires the solution of  $2R \times \text{alsit}$  linear systems of size  $m \times m$  or  $n \times n$ . According to Remark 2.1, these linear systems inherit the sparsity from the coefficient matrices. Note that  $X_R$  is not stored explicitly but in terms of its low-rank factors  $[u_1, \dots, u_R] \in \mathbb{R}^{m \times R}$ ,  $[v_1, \dots, v_R] \in \mathbb{R}^{n \times R}$ . Similarly, the updated

**Algorithm 2** Greedy rank-1 updates.

**Input:** Matrices  $A_1, \dots, A_K, B_1, \dots, B_K, C$  defining a symmetric positive definite linear matrix equation (1), number of updates  $R$ .

**Output:** Rank- $R$  approximation  $X_R$  to the solution of (1).

$$X_0 = 0$$

$$C_0 = C$$

**for**  $j = 0, 1, \dots, R - 1$  **do**

    Apply Algorithm 1 with right-hand side  $C_j$  to determine rank-1 correction  $u_{j+1}v_{j+1}^T$ .

$$X_{j+1} \leftarrow X_j + u_{j+1}v_{j+1}^T$$

$$C_{j+1} \leftarrow C_j - \sum_{k=1}^K A_k u_{j+1}v_{j+1}^T B_k^T$$

**end for**

right-hand side  $C_j$  is stored implicitly, as a sum of the matrix  $C$  and  $j$  rank- $K$  correction terms. Note that we only need to perform matrix–vector multiplications with  $C_j$  and  $C_j^T$ . To perform this efficiently, it is sufficient that  $C$  is sparse or has moderate rank. For example, if  $C$  has rank  $R_C \ll \min\{m, n\}$  and is given in factorized form, a matrix–vector multiplication with  $C_j$  can be performed in  $O((m+n)r)$  operations with  $r = R_C + KR$ . However, in contrast to many algorithms for large-scale matrix equations [2], it is not necessary that  $C$  is of (very) low rank; see Section 5.2 for an example.

**2.1.2. Preservation of symmetry in the solution.** In numerical experiments, we sometimes observed that ALS, Algorithm 1, converges to a symmetric solution for symmetric right-hand sides. The following results show this property for the special case of symmetric Lyapunov equations.

*Lemma 2.3*

Let us consider the Lyapunov equation  $AX + XA = C$ , where  $A$  is symmetric positive definite and  $C$  is symmetric positive semidefinite. Then for every local minimum  $(u_*, v_*)$  of the corresponding functional  $J(u, v) := \langle uv^T, uv^T \rangle_A - 2 \operatorname{tr}(vu^T C)$ , the matrix  $u_*(v_*)^T$  is symmetric positive semidefinite.

*Proof*

Let  $(u_*, v_*)$  be a local minimum of  $J(u, v)$ . Because the zero matrix is symmetric, we may assume  $u_* \neq 0$  and  $v_* \neq 0$ . Note that

$$J(u, v) = v^T A v u^T u + u^T A u v^T v - 2u^T C v$$

is invariant under rescaling:  $J(u, v) = J(\lambda u, \frac{1}{\lambda} v)$  for  $\lambda \in \mathbb{R} \setminus \{0\}$ . Hence, we may assume w.l.o.g. that  $\|u_*\|_2 = \|v_*\|_2$ . Under these restrictions,  $u_* v_*^T$  is symmetric positive semidefinite if and only if  $u_* = v_*$ .

In contradiction to the statement of the lemma, let us suppose that  $u_* \neq v_*$ . Because  $f_{u_*}(v) := J(u_*, v)$  is strictly convex, its unique minimum is given by  $v_*$ . In particular, this implies  $J(u_*, v_*) < J(u_*, u_*)$ . Analogously,  $J(u_*, v_*) < J(v_*, v_*)$ . Adding these two inequalities, one gets

$$\begin{aligned} & 2u_*^T u_* v_*^T A v_* + 2v_*^T v_* u_*^T A u_* - 4u_*^T C v_* \\ & < 2u_*^T u_* u_*^T A u_* + 2v_*^T v_* v_*^T A v_* - 2u_*^T C u_* - 2v_*^T C v_*. \end{aligned}$$

Using  $\|u_*\|_2 = \|v_*\|_2$ , this is equivalent to

$$\begin{aligned} -2u_*^T C v_* &< -u_*^T C u_* - v_*^T C v_* \\ \Leftrightarrow 0 &< -(u_* - v_*)^T C (u_* - v_*), \end{aligned}$$

which leads to a contradiction, because  $C$  is positive semidefinite.  $\square$

We can use Lemma (2.3) to prove the following theorem, which proves that in this special case, Algorithm 2 converges monotonically from below to the exact solution, providing always symmetric positive semidefinite approximate solutions. This is important, because in some applications, positive definiteness of the solution is further exploited.

*Theorem 2.4*

Let us consider the Lyapunov equation

$$AX + XA = C, \quad (10)$$

where  $A$  is symmetric positive definite and  $C$  is symmetric positive semidefinite. Assuming that Algorithm 1 always converges to a local minimum, the application of Algorithm 2 to (10) results in a monotonically increasing (in the Löwner ordering; see [20]) sequence of approximations

$$0 = X_0 \leq X_1 \leq \dots \leq X_R \leq \dots \leq X. \quad (11)$$

*Proof*

We will prove (11) by induction. Initially, we have that  $X_0 = 0$  and  $C_0 = C$  are both symmetric positive semidefinite. Suppose that after  $j$  iterations of Algorithm 2 the approximate solution  $X_j$  and the corresponding updated right-hand side  $C_j = C - AX_j - X_j A$  are both symmetric positive semidefinite. The next greedy rank-1 update  $u_{j+1}v_{j+1}^T$  is a local minimizer of (5) for the updated equation  $A(X - X_j) + (X - X_j)A = C_j$ . Lemma 2.3 yields  $u_{j+1} = v_{j+1}$  because of the positive (semi)definiteness of both  $A$  and  $C_j$ . In turn, the new approximate solution  $X_{j+1} = X_j + u_{j+1}u_{j+1}^T \geq X_j$  is also symmetric positive semidefinite. This also implies that in an ALS half-iteration with  $v = u_{j+1}$  fixed,  $u_{j+1}$  is the minimizer of (6), providing the following equivalent expressions it satisfies:

$$(u_{j+1}^T u_{j+1} A + u_{j+1}^T A u_{j+1} I) u_{j+1} = C_j u_{j+1} \quad (12)$$

$\Updownarrow$

$$\frac{C_j u_{j+1} - u_{j+1}^T A u_{j+1} u_{j+1}}{u_{j+1}^T u_{j+1}} = A u_{j+1}, \quad (13)$$

which also implies

$$2u_{j+1}^T A u_{j+1} u_{j+1}^T u_{j+1} = u_{j+1}^T C_j u_{j+1}. \quad (14)$$

The updated right-hand side now has the form

$$C_{j+1} = C_j - A u_{j+1} u_{j+1}^T - u_{j+1} u_{j+1}^T A.$$

The positive semidefiniteness of  $C_{j+1}$  follows from

$$\begin{aligned}
 y^T C_{j+1} y &= y^T C_j y - y^T \frac{C_j u_{j+1} - u_{j+1} A u_{j+1} u_{j+1}^T}{u_{j+1}^T u_{j+1}} u_{j+1}^T y - y^T u_{j+1} \\
 &\quad \times \left( \frac{C_j u_{j+1} - u_{j+1} A u_{j+1} u_{j+1}^T}{u_{j+1}^T u_{j+1}} \right)^T y \\
 &= y^T C_j y - y^T C_j u_{j+1} \frac{u_{j+1}^T y}{u_{j+1}^T u_{j+1}} - u_{j+1}^T C_j y \frac{u_{j+1}^T y}{u_{j+1}^T u_{j+1}} \\
 &\quad + u_{j+1}^T A u_{j+1} \frac{y^T u_{j+1} u_{j+1}^T y}{u_{j+1}^T u_{j+1}} + u_{j+1}^T A u_{j+1} \frac{y^T u_{j+1} u_{j+1}^T y}{u_{j+1}^T u_{j+1}} \\
 &= y^T C_j y - y^T C_j u_{j+1} \frac{u_{j+1}^T y}{u_{j+1}^T u_{j+1}} - u_{j+1}^T C_j y \frac{u_{j+1}^T y}{u_{j+1}^T u_{j+1}} \\
 &\quad + \frac{u_{j+1}^T C_j u_{j+1} y^T u_{j+1} u_{j+1}^T y}{u_{j+1}^T u_{j+1} u_{j+1}^T u_{j+1}} \\
 &= (y - u_{j+1} \frac{u_{j+1}^T y}{u_{j+1}^T u_{j+1}})^T C_j (y - u_{j+1} \frac{u_{j+1}^T y}{u_{j+1}^T u_{j+1}}) \geq 0,
 \end{aligned}$$

where we have used (13) in the first equality and (14) in the third equality. This proves the induction step and finishes the proof.  $\square$

Theorem 2.4 itself is of limited practical relevance, as it requires the availability of exact local minima. In practice, we stop Algorithm 1 (very) early and only obtain approximate local minima. The result of Theorem 2.4 may then still be used as a theoretical justification for choosing the subspaces  $U$  and  $V$  equal, resulting in computational savings in our main algorithm, Algorithm 4.

## 2.2. Symmetric indefinite and nonsymmetric cases

In the case when  $\mathcal{A}$  is not symmetric positive definite, we use the residual norm to measure the error. Applying the derivation of Section 2.1 to the normal equation leads to the minimization of the functional

$$J(u, v) := \langle uv^T, uv^T \rangle_{\mathcal{A}^T \mathcal{A}} - 2 \operatorname{tr}(vu^T \mathcal{A}^T(C_j)) \quad (15)$$

for determining the best rank-1 correction. The symmetric positive definite linear operator  $\mathcal{A}^T \mathcal{A}$  has the form

$$\mathcal{A}^T \mathcal{A} : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}, \quad \mathcal{A}^T \mathcal{A} : X \mapsto \sum_{k,l=1}^K A_k^T A_l X B_k^T B_l.$$

As before, we use ALS to address the minimization of (15). The first half-iteration takes the form

$$\begin{aligned}
 \tilde{u} &= \arg \min_u J(u, v) = \langle uv^T, uv^T \rangle_{\mathcal{A}^T \mathcal{A}} - 2 \operatorname{tr}(vu^T \mathcal{A}^T(C_j)) \\
 &= \arg \min_u \sum_{k=1}^K \sum_{l=1}^K (u^T A_k^T A_l u) (v^T B_l^T B_k v) - 2 \sum_{k=1}^K (u^T A_k^T C_j B_k v).
 \end{aligned} \quad (16)$$

The matrix

$$\tilde{A} := \sum_{k=1}^K \sum_{l=1}^K (v^T B_l^T B_k v) A_k^T A_l \quad (17)$$

amounts to  $(v^T \otimes I) \mathcal{A}^T \mathcal{A} (v \otimes I)$  and thus inherits the positive definiteness from  $\mathcal{A}^T \mathcal{A}$ . Therefore, the solution of the unconstrained linear-quadratic optimization problem (16) is given by the solution of the linear system  $\tilde{A} \tilde{u} = \sum_{i=k}^K A_k^T C_j B_k v$ .

In the second half-iteration of ALS, we fix the normalized  $u \leftarrow \tilde{u} / \|\tilde{u}\|_2$  and optimize for  $v$ . By the same arguments, the minimizer  $\hat{v}$  is given by the solution of the linear system  $\tilde{B} \tilde{v} = \sum_{k=1}^K (A_k^T C_j B_k)^T u$ , with

$$\tilde{B} := \sum_{k=1}^K \sum_{l=1}^K (v^T A_l^T A_k v) B_k^T B_l. \quad (18)$$

Using the described procedure instead of Algorithm 1 in Algorithm 2 then yields the basic greedy rank-1 algorithm for indefinite and nonsymmetric  $\mathcal{A}$ .

### 2.3. Numerical example

The approach described in Section 2.2 considers  $\mathcal{A}^T \mathcal{A}$  instead of  $\mathcal{A}$ . This squares the condition number, which is well known to slow down convergence of classical iterative methods for solving linear systems. Our greedy low-rank methods are no exception.

To illustrate this point, we consider a generalized Lyapunov equation

$$AX + XA^T + N_1 X N_1^T = -DD^T \quad (19)$$

from the discretization of a 2D heat equation with bilinear boundary control; see Example 5.1 later for more details. We have used 50 discretization points in each direction, resulting in matrices of size  $n = 2500$ . The corresponding  $n^2 \times n^2$  system matrix  $\mathcal{A}$  is symmetric, but not positive definite; it has one negative eigenvalue.

The bottom curves in the plots of Figure 1 show the singular values of the exact solution  $X$  for (19). Because the  $(j+1)$ th singular value is the 2-norm error of the *best* rank- $j$  approximation to  $X$ , the singular values represent a lower bound for the error of the iterates obtained from any greedy rank-1 algorithm. As can be seen in Figure 1(a), Algorithm 2 based on the residual norm converges quite slowly or may even stagnate. We now modify (19), by dividing the matrices  $N_i$  by 2. In turn, the matrix  $\mathcal{A}$  becomes definite. As seen in Figure 1(b), the convergence of Algorithm 2 based on the residual norm does not benefit from this modification. However, the positive definiteness allows us to use the energy norm, which significantly speeds up convergence (Figure 1(c)). Although the error curve is still not close to the best possible convergence predicted by the singular values, this clearly shows that it is preferable to use the energy norm formulation whenever possible. However, in the indefinite case, further improvements are needed to attain satisfactory convergence.

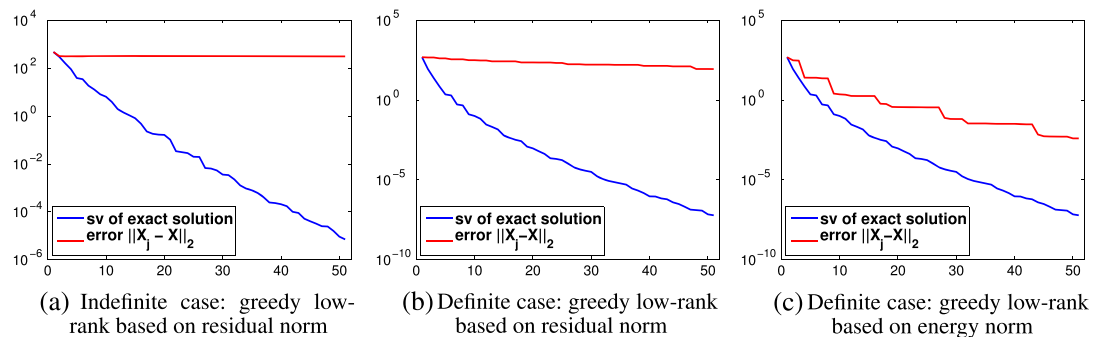


Figure 1. Convergence of basic greedy rank-1 algorithm for the generalized Lyapunov equation (19) arising from the discretization of 2D heat equation with bilinear boundary control.



## 3. GALERKIN PROJECTION

In this section, we combine greedy rank-1 updates with Galerkin projection, similar to an existing idea [21] to accelerate convergence of the ADI method for Sylvester equations. After  $R$  iterations of Algorithm 2, the approximate solution takes the form

$$X_R = \sum_{j=1}^R u_j v_j^T. \quad (20)$$

Considering the subspaces  $\mathcal{U} = \text{span}(\{u_1, \dots, u_R\})$  and  $\mathcal{V} = \text{span}(\{v_1, \dots, v_R\})$ , this means that  $X_R$  is contained in the tensor product  $\mathcal{V} \otimes \mathcal{U}$ . It is not unreasonable to expect that one can obtain an improved approximation to  $X$  by choosing the best approximation from  $\mathcal{V} \otimes \mathcal{U}$ . For this purpose, let the columns of  $U, V \in \mathbb{R}^{n \times R}$  form orthonormal bases of  $\mathcal{U}$  and  $\mathcal{V}$ , respectively. Then every element in  $\mathcal{V} \otimes \mathcal{U}$  takes the form  $UYV^T$  for some  $R \times R$  matrix  $Y$ .

If  $\mathcal{A}$  is symmetric positive definite, we arrive at the minimization problem

$$\begin{aligned} & \min_{Z \in \mathcal{V} \otimes \mathcal{U}} \|X - Z\|_{\mathcal{A}}^2 \\ &= \min_{Z \in \mathcal{V} \otimes \mathcal{U}} \text{tr}(X^T C) + \langle Z, Z \rangle_{\mathcal{A}} - 2 \text{tr}(Z^T C) \\ &= \min_{Y \in \mathbb{R}^{R \times R}} \text{tr}(X^T C) + \langle UYV^T, UYV^T \rangle_{\mathcal{A}} - 2 \text{tr}(VY^T U^T C), \\ &= \min_{Y \in \mathbb{R}^{R \times R}} \text{tr}(X^T C) + \text{vec}(Y)^T (V \otimes U)^T \mathcal{A} (V \otimes U) \text{vec}(Y) - 2 \text{vec}(Y)^T (V \otimes U)^T \text{vec}(C). \end{aligned}$$

This minimization problem is strictly convex and has the unique solution  $Y_R$  given by the solution of the linear system

$$\sum_{k=1}^K (V^T \otimes U^T)(B_k \otimes A_k)(V \otimes U) \text{vec}(Y_R) = (V^T \otimes U^T) \text{vec}(C). \quad (21)$$

This can be viewed as a Galerkin projection of the original equation (1) onto the subspace  $\mathcal{V} \otimes \mathcal{U}$ .

If  $\mathcal{A}$  is not symmetric positive definite, minimizing the residual yields  $Y_R$  as the solution of the linear system

$$\sum_{k=1}^K \sum_{l=1}^K (V^T \otimes U^T)(B_k \otimes A_k)^T (B_l \otimes A_l)(V \otimes U) \text{vec}(Y_R) = \sum_{k=1}^K (V^T \otimes U^T)(B_k \otimes A_k)^T \text{vec}(C). \quad (22)$$

Combining greedy rank-1 updates, Algorithm 2, with Galerkin projection yields Algorithm 3.

---

**Algorithm 3** Greedy rank-1 updates with Galerkin projection.

---

**Input:** Matrices  $A_1, \dots, A_K, B_1, \dots, B_K, C$  defining a linear matrix equation (1), number of updates  $R$ .

**Output:** Rank- $R$  approximation  $X_R$  to the solution of (1).

$X_0 = 0$

$C_0 = C$

**for**  $j = 0, 1, \dots, R - 1$  **do**

    Apply Algorithm 1 with right-hand side  $C_j$  to determine rank-1 correction  $u_{j+1}v_{j+1}^T$ .

    Orthonormalize  $u_{j+1}$  wrt  $U$  and append to  $U$ .

    Orthonormalize  $v_{j+1}$  wrt  $V$  and append to  $V$ .

$Y_{j+1} \leftarrow$  solution of the Galerkin equation (21) or (22)

$X_{j+1} \leftarrow UY_{j+1}V^T$

$C_{j+1} \leftarrow C - \sum_{k=1}^K A_k X_{j+1} B_k^T$

**end for**

---

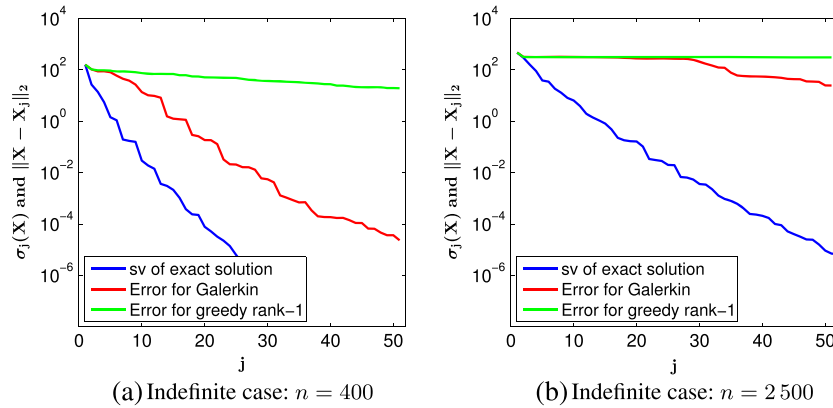


Figure 2. Convergence of error  $\|X_j - X\|_2$  for Algorithm 3 versus the basic greedy rank-1 algorithm applied to the generalized Lyapunov equation (19).

### Remark 3.1

Both (21) and (22) amount to solving a dense linear system of size  $R^2 \times R^2$ . This is performed by a LU decomposition, which requires  $\mathcal{O}(R^6)$  operations and thus limits  $R$  to moderate values, say  $R \leq 100$ . A notable exception occurs for (21) when  $K = 2$ . Then (21) is a generalized Sylvester equation and can be solved with  $\mathcal{O}(R^3)$  operations [22]. For the general case, one may be able to exploit the Kronecker structure (21) and (22) by using the preconditioned conjugate gradient method. This, however, requires the availability of a good preconditioner.

### 3.1. Numerical example

We reconsider the example from Section 2.3, with  $n = 400$  (20 discretization points in each direction) and  $n = 2500$  (50 discretization points in each direction). In both cases, the corresponding operator  $\mathcal{A}$  is indefinite, and therefore, the residual-based formulation needs to be used. Figure 2 shows the convergence improvement obtained from the use of Galerkin projection. Clearly, a significant improvement sets in much earlier for  $n = 400$  than for  $n = 2500$ .

## 4. INJECTING PRECONDITIONED RESIDUALS

The example from Section 3.1 shows that the use of greedy low-rank techniques and Galerkin projection is not sufficient to attain quick convergence for ill-conditioned problems. It is sometimes possible to construct an efficient preconditioner  $\mathcal{P}$  for a general linear matrix equation  $\mathcal{A}(X) = C$ . For example, suitable preconditioners for the generalized Lyapunov equation (3) can often be obtained from preconditioners for the Lyapunov operator  $X \mapsto AX + XA^T$ . The usual way of using preconditioners when solving linear systems consists of replacing  $\mathcal{A}(X) = C$  by the preconditioned equation  $\mathcal{P}^{-1}(\mathcal{A}(X)) = \mathcal{P}^{-1}(C)$ . This, however, bears a major disadvantage: Assuming that  $\mathcal{P}^{-1}$  can be represented by a sum of  $L$  Kronecker products, the composition  $\mathcal{P}^{-1} \circ \mathcal{A}$  is a sum of  $K \cdot L$  (instead of  $K$ ) Kronecker products. This significantly increases the cost of Algorithms 2 and 3.

In this section, we therefore suggest a different way of incorporating preconditioners, inspired by the alternating minimal energy method (AMEn) from [15]. In AMEn, a low-rank approximation of the residual is used to enrich the subspaces in the Galerkin projection. Our approach follows the same idea, but uses a preconditioned residual instead of the residual. In turn, information from one step of the preconditioned Richardson iteration is injected into the subspaces.

The preconditioned residual in step  $j + 1$  of Algorithm 3 is given by  $\mathcal{P}^{-1}(C_j)$ , with  $C_j = C - \sum_{k=1}^K A_k X_j B_k^T$ . Of course, this matrix is not computed explicitly but represented in terms of its low-rank factors, exploiting the fact that  $C_j$  itself is given in terms of low-rank factors and  $\mathcal{P}^{-1}$  is a short sum of Kronecker products. Still, the rank of  $\mathcal{P}^{-1}(C_j)$  is usually quite high and needs to be truncated further. As we will discuss in Remark 4.1 later, from a theoretical point of view, it would be desirable to truncate  $\mathcal{P}^{-1}(C_j)$  within a (small) prescribed accuracy. However, this may

require a large rank and, thus, quickly lead to impractically large dimensions of the subspaces  $U$  and  $V$ . Following [15], we therefore truncate  $\mathcal{P}^{-1}(C_j)$  to fixed rank, say rank 5. The matrices containing the corresponding dominant left and right singular vectors are denoted by  $U_{\text{res}}$  and  $V_{\text{res}}$ , respectively. These vectors are added to  $U$  and  $V$  before performing the Galerkin projection. In effect, the dimension of the subspaces spanned by  $U$  and  $V$  grows more quickly compared with Algorithm 3. In particular, the solution of the linear systems (21) or (22) becomes rapidly expensive (Remark 3.1). To diminish this effect, we perform another low-rank truncation after every Galerkin projection. This requires the computation of an SVD of the (small) matrix  $Y_{j+1}$ . If possible, the tolerance for performing this truncation should be kept small, say  $\text{tol} = 10^{-10}$ , as it ultimately determines the accuracy of the approximate solution.

---

**Algorithm 4** Greedy rank-1 updates with Galerkin projection and preconditioned residuals.

---

**Input:** Matrices  $A_1, \dots, A_K, B_1, \dots, B_K, C$  defining a linear matrix equation (1), number of updates  $R$ .

**Output:** Low-rank approximation  $X_R$  to the solution of (1).

$X_0 = 0$

$C_0 = C$

**for**  $j = 0, 1, \dots, R - 1$  **do**

    Apply Algorithm 1 with right-hand side  $C_j$  to determine rank-1 correction  $u_{j+1}v_{j+1}^T$ .

    Compute approximate left/right dominant singular vectors  $U_{\text{res}}, V_{\text{res}}$  of  $\mathcal{P}^{-1}(C_j)$ .

    Orthonormalize  $[u_{j+1}, U_{\text{res}}]$  wrt  $U$  and append to  $U$ .

    Orthonormalize  $[v_{j+1}, V_{\text{res}}]$  wrt  $V$  and append to  $V$ .

$Y_{j+1} \leftarrow$  solution of the Galerkin equation (21) or (22).

    Truncate  $Y_{j+1}$  to lower rank.

$X_{j+1} \leftarrow UY_{j+1}V^T$

$C_{j+1} \leftarrow C - \sum_{k=1}^K A_k X_{j+1} B_k^T$

**end for**

---

*Remark 4.1*

Assuming that the truncation of the preconditioned residual  $\mathcal{P}^{-1}(C_j)$  is performed within a prescribed accuracy, the optimality properties of the Galerkin projection imply that Algorithm 4 converges at least as fast as the inexact steepest descent method applied to the preconditioned linear system  $\mathcal{P}^{-1}(\mathcal{A}(X)) = \mathcal{P}^{-1}(C)$ . As explained in more detail in [15, Sec 4.2], this implies linear convergence with a rate determined by the condition number of  $\mathcal{P}^{-1} \circ \mathcal{A}$  and the truncation level.

#### 4.1. Preconditioners

It remains to discuss examples of effective preconditioners for which  $\mathcal{P}^{-1}$  is represented as a short sum of Kronecker products. As mentioned earlier, we can use preconditioners for the Lyapunov operator  $X \mapsto AX + XA^T$  in the case of a generalized Lyapunov equation (3). As discussed in Reference [23], such preconditioners can be derived from iterative methods for solving Lyapunov equations:

1. One step of the ADI method with a single shift  $\sigma$  gives rise to the preconditioner

$$\mathcal{P}_{\text{ADI}}^{-1} = (A - \sigma I)^{-1} \otimes (A - \sigma I)^{-1}.$$

Suitable choices for  $\sigma$  are discussed in, for example, [1].

2. One step of the sign function iteration for Lyapunov equations gives rise to the preconditioner

$$\mathcal{P}_{\text{sign}}^{-1} = \frac{1}{2c}(I \otimes I + c^2 A^{-1} \otimes A^{-1}), \quad (23)$$

with the scaling factor  $c = \sqrt{\frac{\|A\|_2}{\|A^{-1}\|_2}}$ . The matrix 2-norm can be approximated using  $\|M\|_2 \approx \sqrt{\|M\|_1 \|M\|_\infty}$  [24].

The application of  $\mathcal{P}_{\text{ADI}}^{-1}$  and  $\mathcal{P}_{\text{sign}}^{-1}$  to a matrix of rank  $\ell$  requires the solution of  $2\ell$  linear systems with the (shifted) matrix  $A$ . To optimize this step, the LU factors are computed only once and reused in every iteration.

#### 4.2. Numerical example

Figure 3 shows the convergence of Algorithm 4 for the example from Sections 2.3 and 3.1 for  $n = 2500$ . We used the preconditioner  $\mathcal{P}_{\text{sign}}^{-1}$  from (23). The convergence, compared with Algorithm 3, clearly improves, to the extent that the method becomes practical for this example. This comes at the expense of a faster increase of the rank, which makes the Galerkin projection more expensive. To limit this increase, we apply a more aggressive truncation strategy and cap the rank at 50. This procedure is explained in more detail in Section 5.

### 5. NUMERICAL EXPERIMENTS

In this section, we first report on the performance of Algorithm 4 for a number of large-scale examples available in the literature, and then we perform a more detailed study of the impact of the individual parts of our algorithm on its performance. Algorithm 4 has been implemented in MATLAB Version 7.14.0.739 (R2012a), and all experiments have been performed on an Intel Xeon CPU E31225 with 4 cores, 3.1 GHz, and 8 GB RAM (Intel Corp., Santa Clara, CA, USA).

Unless stated otherwise, we have made the following choices in the implementation of Algorithm 4:

*ALS iterations.* The number of ALS iterations (Algorithm 1) in the greedy rank-1 procedure is fixed to five.

*Preconditioner.* The sign function-based preconditioner  $\mathcal{P}_{\text{sign}}^{-1}$  from (23) is used.

*Truncation of residual.* The preconditioned residual  $\mathcal{P}^{-1}(C_j)$  is replaced by its best rank-5 approximation. This truncation is performed by combining QR decompositions with an SVD, exploiting the fact that the rank of  $\mathcal{P}^{-1}(C_j)$  is not full but given by the product of  $\text{rank}(C_j)$  with the Kronecker rank of  $\mathcal{P}^{-1}$  (which is 2 for  $\mathcal{P}_{\text{sign}}^{-1}$ ).

*Truncation of iterates.* As explained in Section 4, we truncate  $Y_{j+1}$  to lower rank such that all singular values below the relative tolerance  $\text{tol} = 10^{-10}$  are neglected and the maximal rank  $\text{maxrank}$  is never exceeded. This strategy bears the risk that little new information can be added once  $\text{maxrank}$  is reached. To avoid this, we have implemented a restart strategy when this

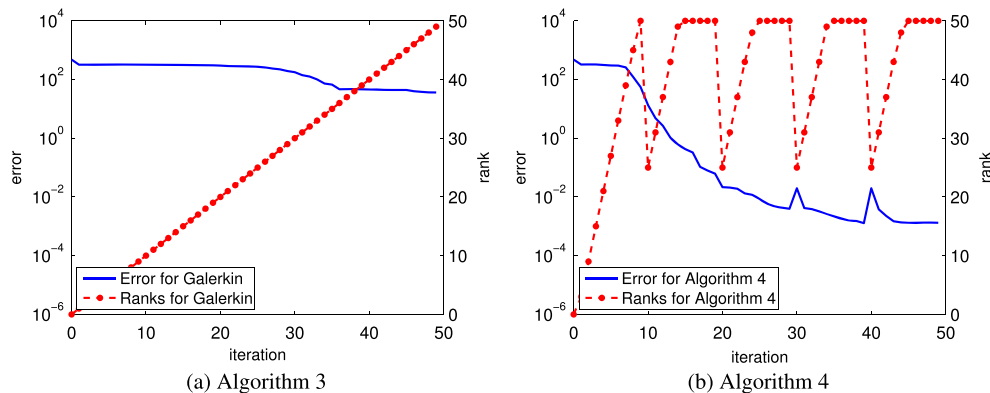


Figure 3. Convergence of error  $\|X_j - X\|_2$  and ranks of  $X_j$  for Algorithms 3 and 4 applied to the generalized Lyapunov equation (19).

happens: In every 10 iterations, the current approximation is truncated more aggressively to rank  $0.6 \times \text{maxrank}$ .

In all experiments later, we measure the convergence of Algorithm 4 by computing the relative residual norm

$$\|C - \mathcal{A}(X_j)\|_F / \|C\|_F.$$

### 5.1. Generalized Lyapunov equations

Generalized Lyapunov equations typically arise from bilinear control problems of the form

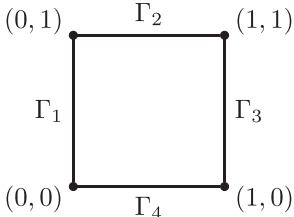
$$\dot{x}(t) = Ax(t) + \sum_{k=1}^K N_k x(t) u_k(t) + Du(t), \quad x(0) = x_0, \quad (24)$$

with the state vector  $x(t) \in \mathbb{R}^n$  and the control  $u(t) \in \mathbb{R}^\ell$ . The controllability Gramian [25] for (24) plays an important role in model reduction of bilinear systems and is given by the solution of the generalized Lyapunov equation (3).

In the following, we consider two examples of bilinear control systems, a bilinear boundary control problem and the Carleman bilinearization of an RC circuit.

#### Example 5.1

Following [6, 9], we consider the heat equation on the unit square with bilinear boundary control: where  $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$  are the boundaries of  $]0, 1[^2$ . After a standard finite difference discretization,

$$\begin{aligned} \frac{\partial}{\partial t} z &= \Delta z && \text{in } ]0, 1[^2, \\ \vec{n} \cdot \nabla z &= 0.5 \cdot u \cdot (z - 1) && \text{on } \Gamma_1, \\ z &= 0 && \text{on } \Gamma_2, \Gamma_3, \Gamma_4, \end{aligned}$$


the controllability Gramian is obtained as the solution of the generalized Lyapunov equation

$$AX + XA^T + N_1 X N_1^T = -DD^T, \quad (25)$$

where  $A \in \mathbb{R}^{n \times n}$  is the discretization of the 2D Laplace operator. The matrices  $N_1, D$  arise from the Neumann boundary control on  $\Gamma_1$  and therefore have  $O(\sqrt{n})$  nonzero columns. The corresponding  $n^2 \times n^2$  system matrix  $\mathcal{A} = I \otimes A + A \otimes I + N_1 \otimes N_1$  turns out to be symmetric, but indefinite; most of its eigenvalues are negative and only a few are positive.

The convergence of Algorithm 4 for  $n = 10,000$  and the maximal rank  $\text{maxrank} = 90$  is shown in Figure 4. The execution time spent per iteration significantly increases as the size of the subspaces  $\mathcal{U}$  and  $\mathcal{V}$  grows, mainly due to the increased cost of constructing and solving the Galerkin system (22) and partly due to the orthogonalization that has to be performed. When increasing  $n$  further, we would need to work with even larger values of  $\text{maxrank}$  to attain reasonable convergence.

Inspired by the experiments in [6], we consider a slight modification of this example, dividing the matrices  $N_i$  by 2. In turn, the matrix  $\mathcal{A}$  becomes definite, and Algorithm 4 can be based on the energy norm. Also, the singular value decay of  $X$  appears to improve. Figure 5 shows the obtained results for  $n = 250,000$ . Even though  $n$  is larger than in Figure 4, Algorithm 4 converges significantly faster and attains a higher accuracy with the same maximal rank.

For both examples, the convergence of Algorithm 4 is clearly sublinear. This appears to be typical for algorithms based on greedy low-rank strategies; see, for example, [26].

Compared with the results for  $n = 562,500$  reported in [6] for the preconditioned CG with low-rank truncation, our algorithm seems to perform slightly worse in terms of attainable accuracy versus the rank of the approximate solution.  $\diamond$

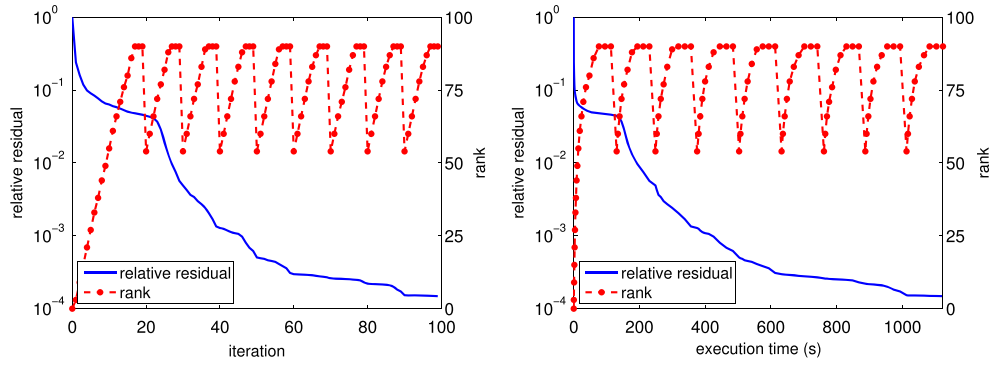


Figure 4. Convergence of relative residual norm for Algorithm 4 applied to Example 5.1 (indefinite case).

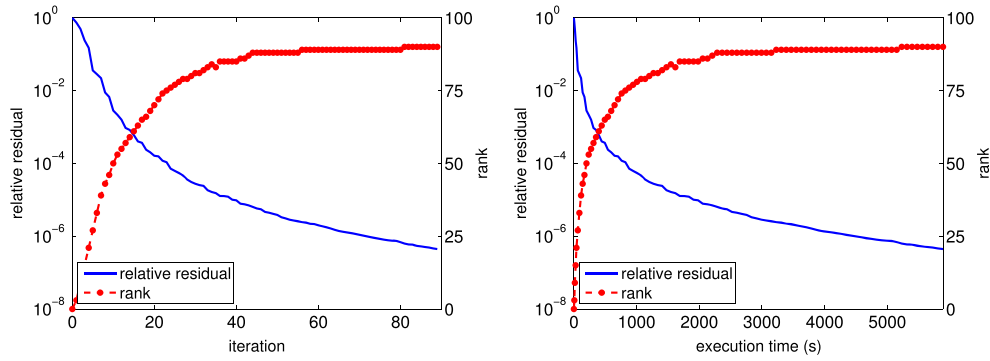


Figure 5. Convergence of relative residual norm for Algorithm 4 applied to Example 5.1 (definite case).

#### Example 5.2

This example is taken from [27] and concerns a scalable RC ladder with  $n_0$  resistors described by

$$v_t = f(v) + bu(t), \quad (26)$$

where

$$f(v) = \begin{pmatrix} -g(v_1) - g(v_1 - v_2) \\ g(v_1 - v_2) - g(v_2 - v_3) \\ \vdots \\ g(v_{n_0-1} - v_{n_0}) \end{pmatrix}, \quad g(v) = \exp(40v) + v - 1.$$

Using Carleman bilinearization, the nonlinear control problem (26) can be approximated by a bilinear control problem of the form (24). In turn, we obtain a generalized Lyapunov equation

$$AX + XA^T + NXN^T = -DD^T,$$

with  $X \in \mathbb{R}^{(n_0+n_0^2) \times (n_0+n_0^2)}$  and

$$A = \begin{bmatrix} A_0 & A_1 \\ 0 & I \otimes A_0 + A_0 \otimes I \end{bmatrix},$$

and  $A_0$  is a tridiagonal matrix, and  $A_1$  arises from the coupling of first-order and second-order terms.

According to our experiments, it is beneficial for this example to skip the greedy rank-1 procedure entirely and only include information from the preconditioned residual in  $U$  and  $V$ . The resulting convergence for  $n_0 = 500$ , that is  $n = 250, 500$ , and  $\text{maxrank} = 70$  is displayed in Figure 6.

The algorithm converges quickly to an accuracy below  $10^{-3}$ , after which the convergence slows down because of imposed limit on the subspace size.

For reference, we also include the results for a modification discussed in [6], where the matrix  $N$  is divided by 2. Figure 7 shows nearly the same convergence behavior. Compared to the results reported in [6], the convergence of our algorithm is significantly faster until the imposed limit on the subspace size is reached.  $\diamond$

The following example is concerned with a stochastic control problem.

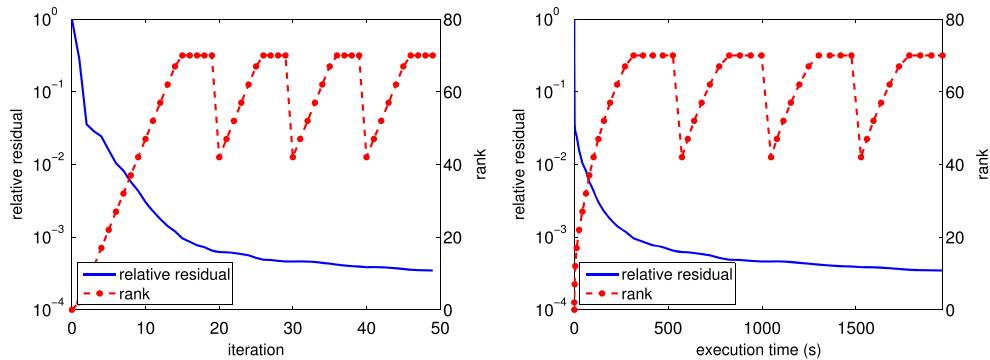


Figure 6. Convergence of relative residual norm for Algorithm 4 (without greedy rank-1) applied to Example 5.2.

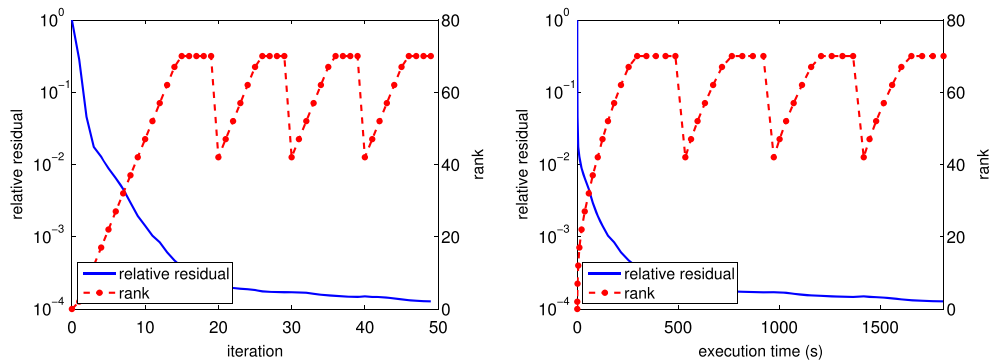


Figure 7. Convergence of relative residual norm for Algorithm 4 (without greedy rank-1) applied to Example 5.2 with  $N$  replaced by  $N/2$ .

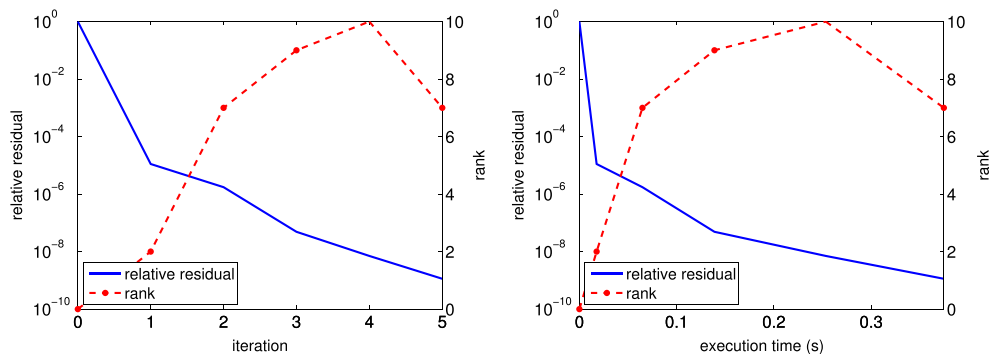


Figure 8. Convergence of relative residual norm for Algorithm 4 applied to Example 5.3.

*Example 5.3*

This example is taken from [6] and arises from the control of a dragged Brownian particle, whose motion is described by the Fokker–Planck equation. We refer to the work in [28] for a detailed explanation of this example. After discretization, the resulting generalized Lyapunov equation has size  $n = 10,000$  and is of the form (25). The matrix  $N_1$  is sparse and has full rank 10,000, while the right-hand side has rank 1.

As can be seen in Figure 8, Algorithm 4 converges quickly for this example and requires less than 0.5 s to attain an accuracy below  $10^{-8}$ . According to the authors in [6, Table 1], the preconditioned BiCG with low-rank truncation requires around 10 s for the same example in a computing environment that is comparable to ours.  $\diamond$

*5.2. Lyapunov equation with right-hand sides having a singular value decay*

As mentioned in the introduction, one of the most important special cases of (1) is the Lyapunov equation

$$AX + XA^T = C, \quad (27)$$

where  $A, C \in \mathbb{R}^{n \times n}$ . There are numerous numerical methods that specifically target (27) [1, 2]. For large-scale equations, most existing strategies crucially depend on a low-rank right-hand side, that is,

$$C = -DD^T, \quad \text{with } D \in \mathbb{R}^{n \times \ell}, \quad \ell \ll n.$$

In particular, this is the case for methods that make use of Krylov subspaces based on  $A$  and  $D$ . The dimension of these subspaces grows proportionally with  $\ell$ , rendering these techniques impractical for larger values of  $\ell$ .

In contrast, Algorithm 3 does not require such a low-rank assumption on the right-hand side to perform efficiently; we only need to be able to perform fast matrix–vector multiplications with  $C$ . Of course, Algorithm 3 can only attain reasonable convergence if the solution  $X$  has a strong singular value decay. For this purpose, it is not necessary that  $C$  has low rank. As the following example demonstrates, it sometimes suffices that  $C$  has a (possibly weaker) singular value decay.

*Example 5.4*

Consider the 2D Poisson equation on the unit square:

$$\begin{aligned} \Delta u(\xi) &= f(\xi), \quad \xi \in \Omega = ]-1, 1[^2, \\ u(\xi) &= 0 \quad \xi \in \partial\Omega. \end{aligned} \quad (28)$$

The standard finite difference discretization with  $n$  grid points in each coordinate yields an  $n^2 \times n^2$  linear system of the form

$$(L \otimes I + I \otimes L) \text{vec}(X) = \text{vec}(F),$$

where  $L$  is the discretization of the 1D Laplace operator and  $F$  contains the values of  $f$  at the grid points. This is equivalent to the Lyapunov equation

$$LX + XL^T = F. \quad (29)$$

In our experiments, we have used  $f(\xi_1, \xi_2) = \exp((\xi_1^p + \xi_2^p)^{\frac{1}{p}})$  with  $p = 10$  and  $n = 40,000$ . This results in a matrix  $F$  with a relatively slow singular value decay. There are several established techniques to multiply with such a matrix  $F$  implicitly and efficiently. For simplicity, we have used adaptive cross approximation (ACA) [29] to replace  $F$  with a matrix of rank  $\ell = 92$ , which corresponds to an error indicator of  $9.7 \times 10^{-8}$  in ACA. The resulting convergence of Algorithm 3 (with 3 ALS iterations in Algorithm 1) is shown in Figure 9.

The observed execution times are very small compared with other examples, because each iteration only requires the solution of  $n \times n$  tridiagonal linear systems and a small-scale Sylvester equation.  $\diamond$



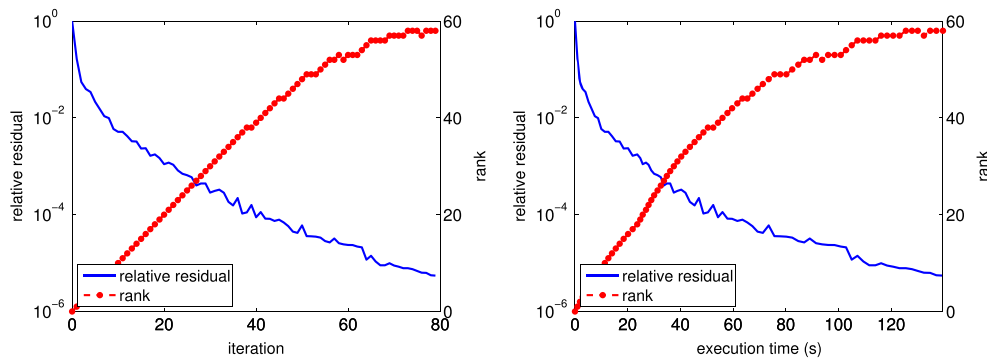


Figure 9. Convergence behavior for the Lyapunov equation arising from 2D Poisson equation with non-low-rank right-hand side.

### 5.3. Detailed numerical study of components of Algorithm 4

The purpose of the following numerical experiments is to consider the different parts of Algorithm 4 separately and assess their impact on its performance.

**5.3.1. Greedy rank-1 updates versus preconditioned residuals.** In Algorithm 4, the bases  $U$  and  $V$  are enriched by adding information from the greedy rank-1 update and the preconditioned residual. The purpose of the following three experiments is to assess the impact of these two enrichment steps separately. Except for the third experiment, we always truncate the preconditioned residuals to rank 1, so that only one vector is added to each basis, equilibrating with the enrichment gained from the greedy rank-1 update. Truncating to rank 1 instead of a higher rank also has the advantage that it enables us to essentially turn off low-rank truncation (we only truncate singular values below  $10^{-14}$ ).

#### Example 5.5

We first consider a synthetic, well-conditioned example of (1) for  $K = 3$  and  $n = 3000$ . The coefficient matrices are given by

$$A_i = \frac{R_i + R_i^T}{2} + \frac{n}{8}I_n, \quad B_i = \frac{S_i + S_i^T}{2} + \frac{n}{8}I_n, \quad i = 1, \dots, 3,$$

and  $C = e_1 e_1^T$ , where the matrices  $R_i$  and  $S_i$  are random matrices generated with the MATLAB function `randn`. No preconditioner is used;  $\mathcal{P} = I$ . From the numerical results shown in Figure 10, it can be concluded, for this particular example, that the enrichment gained from greedy rank-1 updates is much more significant compared with the residuals. However, the approach using just the residuals is significantly faster because the residual does not require the solution of (dense) linear systems unlike for the greedy rank-1 updates.  $\diamond$

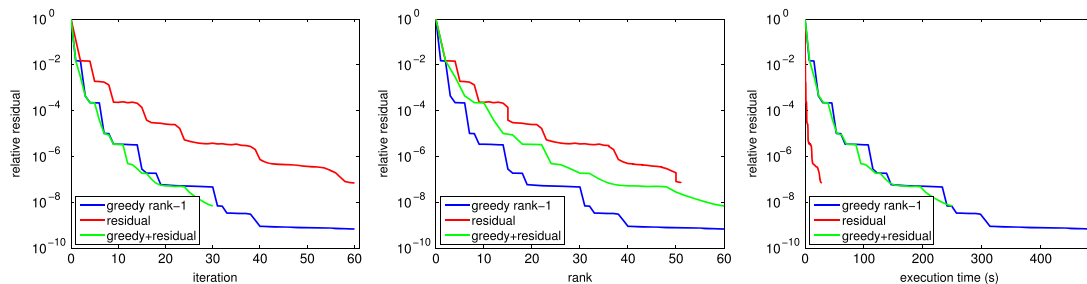


Figure 10. Convergence of Algorithm 4 with different enrichment strategies for Example 5.5.

*Example 5.6*

We consider the generalized Lyapunov equation from Example 5.1 for a modest size,  $n = 2500$ , and use the standard sign-function-based preconditioner. As can be seen in Figure 11, greedy rank-1 updates are still more important than preconditioned residuals, but in contrast to Example 5.5, combining both approaches yields a significant convergence improvement.  $\diamond$

*Example 5.7*

We consider the generalized Lyapunov equation from Example 5.2 with  $n = 2550$ , and the standard preconditioner. In contrast to the previous example, we have observed that truncating the preconditioned residuals to rank 2 instead of rank 1 has a non-negligible impact on the convergence. To illustrate this effect, we have used rank-2 truncation when only the preconditioned residuals are included in the enrichment (red curves in Figure 12). It turns out that this yields better convergence and requires less time compared with combining greedy rank-1 updates and rank-1 truncated preconditioned residuals (green curves in Figure 12), let alone using only greedy rank-1 updates (blue curves in Figure 12).  $\diamond$

From the three experiments earlier, no clear conclusion can be drawn. For some examples, the greedy rank-1 updates constitute the most important parts of the subspaces, while for others, the preconditioned residuals become more important.

**5.3.2. Truncation analysis.** As explained in the beginning of this section, the size of the bases  $U$  and  $V$  is kept under control by a low-rank truncation of the current approximation. All singular values below the relative tolerance  $\text{tol} = 10^{-10}$  are neglected, and the rank is limited to the maximal rank  $\text{maxrank}$ . The purpose of the following experiment is to assess the impact of the latter truncation criterion on the overall performance. To limit the overall rank growth, we always truncate the preconditioned residuals to rank 1. Two vectors are added to each basis in each iteration, one from the greedy rank-1 update and one from the (preconditioned) residual. We compare the implementation of Algorithm 4 with  $\text{maxrank} = \infty$  (we only truncate singular values below  $10^{-12}$ ) and with  $\text{maxrank} = 45$ , combined with restarting every 16 iterations.

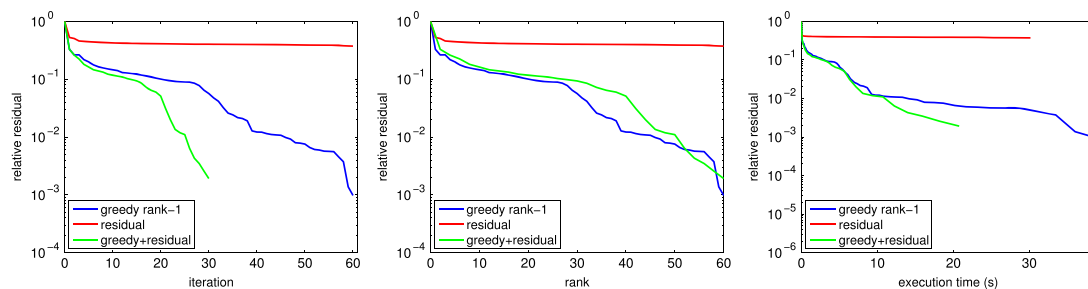


Figure 11. Convergence of Algorithm 4 with different enrichment strategies for Example 5.6.

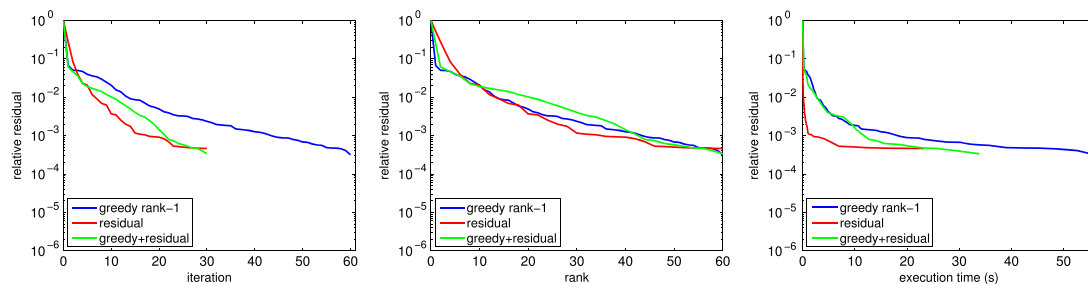


Figure 12. Convergence of Algorithm 4 with different enrichment strategies for Example 5.7.

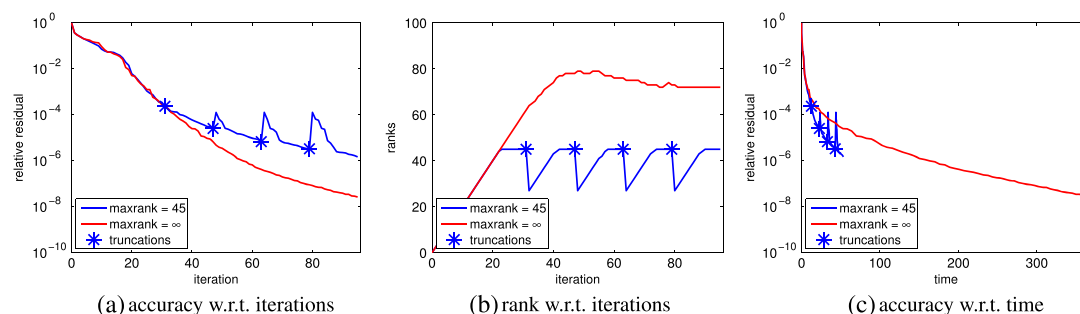


Figure 13. Convergence of Algorithm 4 with truncation based on  $\text{maxrank} = 45$  versus  $\text{maxrank} = \infty$  for Example 5.8.

#### Example 5.8

We consider the generalized Lyapunov equation from Example 5.1 with  $n = 1600$  and the standard preconditioner. From the numerical results shown in Figure 13, we observe the expected effect that truncation slows down convergence. On the other hand, it can be clearly seen from Figure 13(c) that the implementation with truncation produces good results in significantly smaller amounts of time.

◇

## 6. CONCLUSIONS

In principle, greedy low-rank methods are applicable to any linear matrix equation whose solution admits good low-rank approximations. In practice, however, it turns out that these methods need to be combined with Galerkin projection and preconditioning strategies. As discussed in Sections 3 and 5.3, each of these individual components is important to attain reasonable convergence for a wider range of applications. The resulting solver, Algorithm 4, is demonstrated to perform quite well for problems that have been discussed earlier in the literature. For more challenging problems that feature larger ranks, the need for constructing and solving the Galerkin systems (21)–(22) may become a bottleneck. One way to overcome this is to stop our method when a certain rank is reached and use the approximate result as an initial guess for the iterative methods discussed in [6].

## ACKNOWLEDGEMENTS

We thank Tobias Breiten for sending us a MATLAB implementation of Example 5.3. This work is supported by the SNF research module *A Reduced Basis Approach to Large-Scale Pseudospectra Computations* within the SNF ProDoc *Efficient Numerical Methods for Partial Differential Equations*.

## REFERENCES

1. Benner P, Saak J. Numerical solution of large and sparse continuous time algebraic matrix Riccati and Lyapunov equations: a state of the art survey. *GAMM-Mitteilungen* 2013; **36**(1):32–52.
2. Simoncini V. *Computational methods for linear matrix equations 2013*. Available from: <http://www.dm.unibo.it/simoncin/list.html> [Accessed on 12 February 2015].
3. Grasedyck L, Hackbusch W, Khoromskij BN. Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices. *Computing* 2003; **70**(2):121–165.
4. Penzl T. Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. *Systems & Control Letters* 2000; **40**(2):139–144.
5. Antoulas AC, Sorensen DC, Zhou Y. On the decay rate of Hankel singular values and related issues. *Systems & Control Letters* 2002; **46**(5):323–342.
6. Benner P, Breiten T. Low rank methods for a class of generalized Lyapunov equations and related issues. *Numerische Mathematik* 2013; **124**(3):441–470.
7. Merz A. Computation of generalized Gramians for model reduction of bilinear control systems and time-delay systems. *PhD Thesis*, TU Kaiserslautern, 2012.
8. Hochbruck M, Starke G. Preconditioned Krylov subspace methods for Lyapunov matrix equations. *SIAM Journal on Matrix Analysis and Applications* 1995; **16**(1):156–171.

9. Damm T. Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations. *Numerical Linear Algebra with Applications* 2008; **15**(9):853–871.
10. Eppler AK, Bollhöfer M. An alternative way of solving large Lyapunov equations. *Proceedings in Applied Mathematics and Mechanics* 2010; **10**(1):547–548.
11. Kressner D, Tobler C. Low-rank tensor Krylov subspace methods for parametrized linear systems. *SIAM Journal on Matrix Analysis and Applications* 2011; **32**(4):1288–1316.
12. Ammar A, Mokdad B, Chinesta F, Keunings R. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. *Journal of Non-Newtonian Fluid Mechanics* 2006; **139**(3):153–176.
13. Nouy A. Proper generalized decompositions and separated representations for the numerical solution of high dimensional stochastic problems. *Archives of Computational Methods in Engineering* 2010; **17**:403–434.
14. Chinesta F, Ammar A, Cueto E. Recent advances and new challenges in the use of the proper generalized decomposition for solving multidimensional models. *Archives of Computational Methods in Engineering. State of the Art Reviews* 2010; **17**(4):327–350.
15. Dolgov SV, Savostyanov DV. Alternating minimal energy methods for linear systems in higher dimensions. *SIAM Journal on Scientific Computing* 2014; **36**(5):A2248–A2271.
16. Ortega JM, Rheinboldt WC. Iterative solution of nonlinear equations in several variables. In *Classics in Applied Mathematics*, Vol. 30. Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, 2000. DOI: 10.1137/1.9780898719468, Available from: <http://dx.doi.org/10.1137/1.9780898719468>, reprint of the 1970 original.
17. Davis TA. *Direct Methods for Sparse Linear Systems, Fundamentals of Algorithms*, Vol. 2. Society for Industrial and Applied Mathematics (SIAM): Philadelphia, PA, 2006. DOI: 10.1137/1.9780898718881, Available from: <http://dx.doi.org/10.1137/1.9780898718881>.
18. Nouy A. Generalized spectral decomposition method for solving stochastic finite element equations: invariant subspace problem and dedicated algorithms. *Computer Methods in Applied Mechanics and Engineering* 2008; **197**(51–52):4718–4736.
19. Benner P, Breiten T. Interpolation-based  $H_2$ -model reduction of bilinear control systems. *SIAM Journal on Matrix Analysis and Applications* 2012; **33**(3):859–885.
20. Siotani M. Cancellation of section 5 of “Some applications of Loewner’s ordering on symmetric matrices”. *Annals of the Institute of Statistical Mathematics* 1968; **20**:168.
21. Benner P, Li RC, Truhar N. On the ADI method for Sylvester equations. *Journal of Computational and Applied Mathematics* 2009; **233**(4):1035–1045.
22. Gardiner JD, Laub AJ, Amato JJ, Moler CB. Solution of the Sylvester matrix equation  $AXB^T + CXD^T = ESS$ . *Association for Computing Machinery. Transactions on Mathematical Software* 1992; **18**(2):223–231.
23. Kressner D, Plesinger M, Tobler C. A preconditioned low-rank CG method for parameter-dependent Lyapunov matrix equations. *Numerical Linear Algebra with Applications* 2014; **21**(5):666–684.
24. Sima V, Benner P. Experimental evaluation of new SLICOT solvers for linear matrix equations based on the matrix sign function. Computer-aided Control Systems. *IEEE International Conference on IEEE CACSD 2008*, San Antonio, TX, USA, 2008; 601–606.
25. Benner P, Damm T. Lyapunov equations, energy functionals, and model order reduction of bilinear and stochastic systems. *SIAM Journal on Control and Optimization* 2011; **49**(2):686–711.
26. Cancès E, Ehrlicher V, Lelièvre T. Convergence of a greedy algorithm for high-dimensional convex nonlinear problems. *Mathematical Models and Methods in Applied Sciences* 2011; **21**(12):2433–2467.
27. Bai Z, Skoogh D. A projection method for model reduction of bilinear dynamical systems. *Linear Algebra and Its Applications* 2006; **415**(2–3):406–425.
28. Hartmann C, Schäfer-Bung B, Thöns-Zueva A. Balanced averaging of bilinear systems with applications to stochastic control. *SIAM Journal on Control and Optimization* 2013; **51**(3):2356–2378.
29. Bebendorf M, Rjasanow S. Adaptive low-rank approximation of collocation matrices. *Computing* 2003; **70**(1):1–24.