MATHICSE Technical Report
Nr. 18.2012
April 2012

A preconditioned low-rank CG method for parameter-dependent Lyapunov matrix equations

D. Kressner, M. Plesinger, C. Tobler

A PRECONDITIONED LOW-RANK CG METHOD FOR PARAMETER-DEPENDENT LYAPUNOV MATRIX EQUATIONS[*]

DANIEL KRESSNER[†], MARTIN PLEŠINGER[‡], AND CHRISTINE TOBLER[§]

Abstract. This paper is concerned with the numerical solution of symmetric large-scale Lyapunov equations with low-rank right-hand sides and coefficient matrices depending on one or several parameters. Specifically, we consider the situation when the parameter dependence is sufficiently smooth and the aim is to compute solutions for many different parameter samples. Based on existing results for Lyapunov equations and parameter-dependent linear systems, we prove that the tensor containing all solution samples typically allows for an excellent low multilinear rank approximation. Stacking all sampled equations into one huge linear system, this fact can be exploited by combining the preconditioned CG method with low-rank truncation. Our approach is flexible enough to allow for a variety of preconditioners based, for example, on the sign function iteration or the ADI method.

1. Introduction.

Let us consider a Lyapunov matrix equation

$$A(\alpha)X(\alpha)M(\alpha)^T + M(\alpha)X(\alpha)A(\alpha)^T = B(\alpha)B(\alpha)^T, \tag{1.1}$$

where the coefficient matrices $A(\alpha), M(\alpha) \in \mathbb{R}^{n\times n}$, $B(\alpha) \in \mathbb{R}^{n\times t}$, and consequently also the solution matrix $X(\alpha) \in \mathbb{R}^{n\times n}$, depend on a vector of parameters $\alpha \in \Omega \subset \mathbb{R}^p$.

We are concerned with the problem of solving (1.1) for possibly many parameter samples. This is needed in interpolatory model reduction techniques for parametrized linear control systems, see [4, 6]. For the rest of this paper, we will assume that $A(\alpha)$ and $M(\alpha)$ are both symmetric positive definite for every $\alpha \in \Omega$. In particular, this implies that (1.1) has a unique symmetric positive definite solution. Our assumption is satisfied, for example, for Lyapunov equations (1.1) arising from the finite element discretization of an infinite-dimensional linear control system governed by a parameter-dependent parabolic PDE. In this case, $A(\alpha)$ and $M(\alpha)$ correspond to the stiffness and mass matrices, respectively. We will comment on extensions to the nonsymmetric case in Section 5.

For the rest of this paper, we suppose that the right-hand side of (1.1) has low rank, i.e., $t \ll n$. Under our assumptions, this implies that $X(\alpha)$ admits an excellent low-rank approximation for fixed α; see, e.g., [17, 28, 34, 37]. Virtually all existing algorithms for large-scale Lyapunov equations exploit this observation. This includes the low-rank Smith iteration and ADI method [31, 33], subspace methods [9, 25, 36, 39, 40], and low-rank variants of the sign function iteration [5, 11]. All these methods deal efficiently with a single Lyapunov equation evaluated at an individual parameter sample, but none of them can be extended in an obvious way to deal efficiently with many parameter samples.

In this paper, we propose new Krylov subspace based techniques to solve (1.1) for many parameter samples simultaneously. For this purpose, we combine our recently developed low-rank techniques for solving parameter-dependent linear systems [28] with low-rank techniques for Lyapunov equations. For this purpose, we proceed as follows.

[*]The work of the second author has been supported by CRUS through the Scientific Exchange program SciEx–NMS, project No. 09.071. The work of the third author has been supported by SNF Grant PDFMP2 124898.

[†]ANCHP, MATHICSE, EPF Lausanne, Switzerland. daniel.kressner@epfl.ch

[‡]Department of Mathematics and Didactics of Mathematics, TU Liberec, Czech Republic. martin.plesinger@tul.cz

[§]ANCHP, MATHICSE, EPF Lausanne, Switzerland. christine.tobler@epfl.ch

In Section 2, we consider (1.1) for a fixed parameter sample and treat it as a Kronecker-structured $n^2 \times n^2$ linear system. This view, which has already been promoted in [24], allows more flexibility in the choice of the solver and the preconditioner. More specifically, we combine the standard conjugate gradient (CG) method with preconditioners inspired by existing methods for solving Lyapunov equations. One obvious disadvantage of this approach is that each iterate in the CG method is a vector of length n^2, which is infeasible for large-scale applications. This can be avoided by applying low-rank truncations to the iterates, an idea that has been successfully used in [7, 16].

In Section 3, the approach from Section 2 is extended to $m > 1$ parameter samples by considering all m Lyapunov equations simultaneously in one huge block diagonal linear system of size $mn^2 \times mn^2$. Again, a CG method is applied to solve this system. However, instead of low-rank matrix truncation, we now consider the iterates of length mn^2 as 3rd order tensors of size $n \times n \times m$ and apply multilinear low-rank approximation [15]. The success of this approach crucially depends on the approximability of the solution tensor. In the case of smooth dependence on a single parameter, we prove rapid decay of the approximation error for increasing multilinear ranks. This approach is also well suited for several parameters, provided that the number of samples does not grow too large. This can be achieved for several parameters by sparse collocation techniques, see for example [1].

In Section 4, an alternative approach is suggested for $p > 1$ parameters, in the case that the samples are arranged in a tensor grid. The solutions of (1.1) are collected into a tensor of order $2+p$, where the first two modes correspond to the rows/columns of the solutions and each of the remaining p modes corresponds to a parameter. The associated linear system is solved by combining CG with low-rank truncation in the so called hierarchical Tucker format [18, 23].

REMARK 1.1. *Most numerical experiments in this paper have been performed in* MATLAB *version 6.5 (R13) on an Intel Core2 Duo (T8300)* 2.40 GHz *processor. The experiments in Section 4 have been performed in* MATLAB *version 7.8 (R2009a).*

2. No parameter. We first consider (1.1) for a fixed parameter sample. For simplicity, we omit the dependence on the parameter:

$$AXM^T + MXA^T = BB^T, \tag{2.1}$$

where A and M are both symmetric positive definite. It is well known that (2.1) can be cast as a Kronecker product linear system

$$\big(M \otimes A + A \otimes M\big)x = b, \tag{2.2}$$

with $x = \mathrm{vec}(X)$ and $b = \mathrm{vec}(BB^T)$, where $\mathrm{vec}(\cdot)$ stacks the columns of an $n \times n$ matrix into a vector of length n^2.

2.1. The basic form of preconditioned CG. As the matrix in the linear system (2.2) is symmetric positive definite, we can apply the preconditioned CG method to (2.2). We will base our preconditioner on existing methods for solving a standard Lyapunov equation of the form

$$\bar{A}\bar{X} + \bar{X}\bar{A}^T = \bar{B}\bar{B}^T. \tag{2.3}$$

Note that (2.1) and (2.3) are equivalent via the relations $\bar{A} = L_M^{-1} A L_M^{-T}$, $\bar{B} = L_M^{-1} B$, and $X = L_M^{-T} \bar{X} L_M^{-1}$, with the Cholesky factorization $M = L_M L_M^T$. Given a preconditioner $\mathcal{P}^{-1}$ for

$$\bar{\mathcal{A}} := I \otimes \bar{A} + \bar{A} \otimes I \tag{2.4}$$

the Kronecker product formulation of (2.3), a preconditioner for (2.2) is obtained as

$$\big(L_M^{-1} \otimes L_M^{-1}\big)\mathcal{P}^{-1}\big(L_M^{-T} \otimes L_M^{-T}\big). \tag{2.5}$$

Algorithm 1 is the standard CG method [3] applied to (2.2) with the preconditioner (2.5). The only difference to the standard formulation is that we recast all operations in terms of $n \times n$ matrices instead of vectors of length n^2. In particular, the inner product $\langle \cdot, \cdot \rangle$ should be understood as the matrix inner product and the preconditioner $\mathcal{P}^{-1}$ is considered as a linear operator on $\mathbb{R}^{n\times n}$.

Algorithm 1 Conjugate gradient method for solving $AXM^T + MXA^T = BB^T$.

Input: Symmetric positive definite matrices $A, M \in \mathbb{R}^{n\times n}$ and right-hand side matrix $B \in \mathbb{R}^{n\times t}$, tolerance $\texttt{tol} > 0$.

Output: Approximation X_k to solution of Lyapunov equation (2.1).

```
1: $L_M \leftarrow \texttt{chol}(M)$    {(sparse) Cholesky decomposition}
2: $k \leftarrow 0$,
3: $R_0 \leftarrow BB^T$    {initial residual corresponding to (2.1)}
4: Compute $\bar{R}_0 \leftarrow L_M^{-1} R_0 L_M^{-T}$.
5: Compute $\bar{Z}_0 = \mathcal{P}^{-1}(\bar{R}_0)$.    {apply preconditioner for (2.3)}
6: Compute $Z_0 \leftarrow L_M^{-1} \bar{Z}_0 L_M^{-T}$.
7: $\rho_0^{\text{CG}} \leftarrow \langle R_0, Z_0 \rangle$
8: $P_0 \leftarrow Z_0$
9: repeat
10:     $k \leftarrow k + 1$
11:     $W_k \leftarrow AP_{k-1}M^T + MP_{k-1}A^T$    {apply Lyapunov operator}
12:     $\alpha_k^{\text{CG}} \leftarrow \rho_{k-1}^{\text{CG}} / \langle W_k, P_{k-1} \rangle$
13:     $X_k \leftarrow X_{k-1} + \alpha_k^{\text{CG}} P_{k-1}$
14:     $R_k \leftarrow R_{k-1} - \alpha_k^{\text{CG}} W_k$
15:     Compute $\bar{R}_k \leftarrow L_M^{-1} R_k L_M^{-T}$.
16:     Compute $\bar{Z}_k = \mathcal{P}^{-1}(\bar{R}_k)$.    {apply preconditioner for (2.3)}
17:     Compute $Z_k \leftarrow L_M^{-1} \bar{Z}_k L_M^{-T}$.
18:     $\rho_k^{\text{CG}} \leftarrow \langle R_k, Z_k \rangle$
19:     $\beta_k^{\text{CG}} \leftarrow \rho_k^{\text{CG}} / \rho_{k-1}^{\text{CG}}$
20:     $P_k \leftarrow Z_k + \beta_k^{\text{CG}} P_{k-1}$
21: until $\|BB^T - (AX_kM^T + MX_kA^T)\|_F < \texttt{tol}$    {test true residual for convergence}
```

It is well known that the solution X of the Lyapunov equation (2.1) is symmetric. It turns out that Algorithm 1 automatically preserves this symmetry. More specifically, if the preconditioner $\mathcal{P}^{-1}$ applied to a symmetric matrix again results in a symmetric matrix, it can be easily seen that all iterates P_k, R_k, X_k, W_k, Z_k generated by Algorithm 1 are symmetric $n \times n$ matrices.

2.2. Incorporating low-rank truncation into the preconditioned CG method. A serious drawback of Algorithm 1, the storage requirements are $O(n^2)$ as the iterates are generally dense $n \times n$ matrices. Motivated by the facts that the right-hand side BB^T has low rank (as we assumed $t \ll n$) and that the solution X can be well approximated by a low-rank matrix, we expect the iterates to be also approximable by low-rank matrices. We will explicitly enforce low rank by truncating the iterates repeatedly.

Any symmetric matrix $S \in \mathbb{R}^{n\times n}$ of rank $r \ll n$ can be stored in $O(nr)$ memory by means of a decomposition

$$S = U_S \Lambda_S U_S^T, \tag{2.6}$$

where $\Lambda_S \in \mathbb{R}^{r\times r}$ is symmetric and $U_S \in \mathbb{R}^{n\times r}$. For example, this can be achieved by the spectral decomposition of S. All iterates of Algorithm 1 will be represented in the factored form (2.6). This allows all operations needed in Algorithm 1 to be performed efficiently:

Matrix multiplication. An operation of the form $\bar{S} = L_M^{-1} S L_M^{-T}$ is performed as

$$\bar{S} = L_M^{-1}\big(U_S \Lambda_S U_S^T\big)L_M^{-T} = \big(L_M^{-1}U_S\big)\Lambda_S\big(U_S^T L_M^{-T}\big) =: U_{\bar{S}}\Lambda_S U_{\bar{S}}^T,$$

not increasing the rank and requiring $O(\mathsf{nnz}(L_M)r)$ instead of $O(\mathsf{nnz}(L_M)n)$ operations, where nnz denotes the number of nonzero entries of a matrix.

Matrix addition. $\bar{S} = S + T$ is performed as

$$\begin{aligned} S + T &= U_S\Lambda_S U_S^T + U_T\Lambda_T U_T^T = \big[U_S, U_T\big]\begin{bmatrix} \Lambda_S & 0 \\ 0 & \Lambda_T \end{bmatrix}\big[U_S, U_T\big]^T \\ &=: U_{\bar{S}}\Lambda_{\bar{S}}U_{\bar{S}}^T. \end{aligned}$$

While this operation has zero cost, the rank generally increases to $r_S + r_T$, where r_S, r_T are the ranks of S, T.

Application of Lyapunov operator. $\bar{S} = ASM^T + MSA^T$ is performed in an analogous way:

$$\begin{aligned} ASM^T + MSA^T &= AU_S\Lambda_S U_S^T M^T + MU_S\Lambda_S U_S^T A^T \\ &= \big[AU_S, MU_S\big]\begin{bmatrix} 0 & \Lambda_S \\ \Lambda_S & 0 \end{bmatrix}\big[AU_S, MU_S\big]^T \\ &=: U_{\bar{S}}\Lambda_{\bar{S}}U_{\bar{S}}^T, \end{aligned}$$

doubling the rank and requiring $O\big(\mathsf{nnz}(M)r + \mathsf{nnz}(A)r\big)$ instead of $O\big(\mathsf{nnz}(M)n + \mathsf{nnz}(A)n\big)$ operations.

Matrix inner product. $\langle S, T\rangle$ is performed as

$$\langle S, T\rangle = \mathsf{tr}\big(ST\big) = \mathsf{tr}\big(U_T^T U_S\Lambda_S U_S^T U_T\Lambda_T\big),$$

where the last matrix product can be evaluated in $O(nr_Sr_T)$ instead of $O(n^2)$ operations, provided that Λ_S, Λ_T are diagonal.

The only operation not covered in the list above is the application of the preconditioner $\mathcal{P}^{-1}$; this will be discussed in Section 2.3.

When applying Algorithm 1, the ranks of the iterates and consequently also the storage requirements, will grow dramatically. This rank growth can be limited as follows. Given a factored matrix $S = U_S\Lambda_S U_S^T$ of rank r, we first perform a QR decomposition $U_S = QR$ with $Q \in \mathbb{R}^{n\times r}$ having orthonormal columns and $R \in \mathbb{R}^{r\times r}$ being upper triangular. Then

$$S = U_S\Lambda_S U_S^T = Q\big(R\Lambda_S R^T\big)Q^T.$$

We then compute a spectral decomposition

$$R\Lambda_S R^T = \big[V_1, V_2\big]\begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix}\big[V_1, V_2\big]^T,$$

where $\Lambda_1 \in \mathbb{R}^{\bar{r}\times\bar{r}}$ is a diagonal matrix containing the $\bar{r}$ eigenvalues of largest absolute magnitude. The truncated matrix $\bar{S}$ of rank $\bar{r}$ is obtained as

$$\bar{S} = U_{\bar{S}}\Lambda_{\bar{S}}U_{\bar{S}}^T, \quad \text{with} \quad U_{\bar{S}} = QV_1, \quad \Lambda_{\bar{S}} = \Lambda_1.$$

Note that $\|S - \bar{S}\|_2 = \|\Lambda_2\|_2 = \sigma_{\bar{r}+1}(S)$, where $\sigma_j(\cdot)$ denotes the jth largest singular value of a matrix. It remains to discuss the choice of the parameter $\bar{r}$, the numerical rank to which the different iterates of Algorithm 1 are truncated.

Numerical rank for X_k. For a desired user-specified accuracy $\varepsilon > 0$ and a safety factor $C_1 \leq 1$, we let the *numerical rank of* X_k denote the smallest j such that

$$\sigma_{j+1}(X_k) \leq C_1\, \varepsilon\, \|X_k\|_2. \tag{2.7}$$

In our experiments, we have observed that $C_1 = 0.05$ gives good performance.

Numerical rank for R_k. As the residuals can be expected to become small with increasing k, a relative criterion of the form (2.7) would lead to unnecessarily high ranks in latter stages of the CG method. Instead, we let the numerical rank *numerical rank of* R_k denote the smallest j such that

$$\sigma_{j+1}(R_k) \leq \max\big\{C_1\, \varepsilon\, \|R_k\|_2, C_2\, \varepsilon\, \|R_0\|_2, \big\}, \tag{2.8}$$

for safety factors $C_1 \leq 1$ and $C_2 \leq 1$. Analogous criteria are used for the iterates P_k and W_k. In our experiments, we have observed that choosing $C_1 = 0, C_2 = 0.1$ for R_k, $C_1 = C_2 = 0.25$ for P_k, and $C_1 = 0$, $C_2 = 1$ for W_k gives good performance.

Summarizing the discussion, all iterates X_k, P_k, R_k, W_k, Z_k of Algorithm 1 are represented in the factored form (2.6). We apply truncation with the parameters described above after every operation that increases the rank of the iterates, i.e., in lines 5, 11, 13, 14, 16, and 20.

REMARK 2.1. *In exact arithmetic, the matrix R_k corresponds to the residual of the approximation X_k produced in the kth step of CG. In finite-precision arithmetic, $\|R_k\|_F$ remains a faithful convergence indicator (up to machine precision) [41]. To a certain extent, this remains valid when introducing low-rank truncations. However, it is safer to use the explicitly recomputed residual for the stopping criterion in line 21 of Algorithm 1.*

2.3. Preconditioners for Lyapunov equations.

The convergence of Algorithm 1 is governed by classical results for the CG method. To attain fast convergence, the use of an effective preconditioner $\mathcal{P}^{-1}$ for $\bar{\mathcal{A}}$ defined in (2.4) is therefore mandatory. In the context of low-rank truncation, there is another important reason for preconditioning. As illustrated in Figure 2.1, Algorithm 1 without any preconditioner not only suffers from slow convergence but also from a significant growth of the numerical ranks in the initial phase. As we will see below, this transient growth is diminished when using effective preconditioners.

In the following, we will discuss various possibilities for the preconditioner. On the one hand, the preconditioner should preserve the symmetry and (approximately) low rank. The latter requirement is satisfied when $\mathcal{P}^{-1}$ can be written as a short sum of Kronecker products. This rules out, for example, the use of classical preconditioners such as Jacobi and SSOR [24]. On the other hand, the preconditioner should reduce the condition number of $\bar{\mathcal{A}}$ significantly. Fully structure-preserving preconditioners, such as $\mathcal{P}^{-1} = (P \otimes P)^{-1}$, may not offer enough flexibility to achieve this goal, see [42] for a related discussion. We consider two preconditioners inspired by the ADI iteration and the sign function iteration.

2.3.1. ADI preconditioner.

Preconditioning a Krylov subspace method with a few iterates of ADI was already proposed in [24] for Lyapunov equations, see also [9,

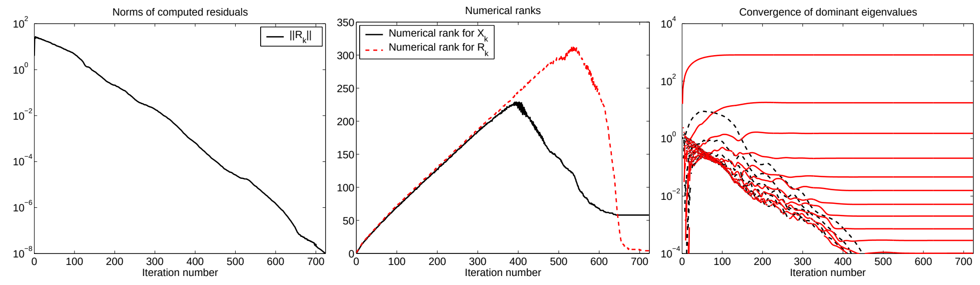


FIG. 2.1. *Algorithm 1 applied to the example from [29, Sec. 2.3.1]. The matrices A and M are the stiffness and mass matrices, respectively, from the piecewise linear FE discretization of the Poisson equation on the unit square with zero Dirichlet boundary conditions* ($n = 11\,036$)*. The matrix $B \in \mathbb{R}^{n\times 1}$ is chosen randomly.* Left: *Convergence of the residual norms.* Middle: *Numerical ranks for X_k and R_k for $\varepsilon = 10^{-8}$.* Right: *Evolution of 15 largest singular values of X_k as k increases. Solid lines correspond to positive eigenvalues, dashed lines to absolute values of negative eigenvalues. The* CG *method stops after* 723 *iterations* (8 h, 25 min, 31 s) *with residual norm* $9.685 \cdot 10^{-10}$.

10]. Algorithm 2 describes one cycle of ADI with ℓ real negative shifts $\varphi_1, \dots, \varphi_\ell$. The optimal choice of shifts is discussed in [44, 45, 35, 37]. In each iteration of ADI, linear systems with the matrix $\bar{A} - \varphi_j I_n = L_M^{-1}(A - \varphi_j M)L_M^{-T}$ need to be solved. Typically, this is done by a sparse direct solver, computing the sparse Cholesky factorization of $A - \varphi_j M$ only once in a preprocessing step.

Algorithm 2 ADI(ℓ) applied to $\bar{A}\bar{Z} + \bar{Z}\bar{A}^T = \bar{R}$.

```
1: $\bar{Z}^{(0)} \leftarrow 0$
2: for $j = 1, \dots, \ell$
3:     solve :  $(\bar{A} - \varphi_j I_n)\bar{Z}^{(j-\frac{1}{2})} = \bar{R} - \bar{Z}^{(j-1)}(\bar{A} + \varphi_j I_n)^T$
4:     solve :  $\bar{Z}^{(j)}(\bar{A} - \varphi_j I_n)^T = \bar{R} - (\bar{A} + \varphi_j I_n)\bar{Z}^{(j-\frac{1}{2})}$
5: end
```

If the right-hand side $\bar{R}$ has low rank then Algorithm 2 can be implemented efficiently in low-rank arithmetic. As each iteration of Algorithm 2 increases the rank, it is necessary to truncate after each iteration. While using $\ell > 1$ yields more effective preconditioners, our numerical experiments revealed that the computational time spent on low-rank repeated truncation offsets this benefit. We have therefore restricted ourselves to $\ell = 1$. In this case, Algorithm 2 reduces to

$$\bar{Z} \;\leftarrow\; -2\varphi_1(\bar{A} - \varphi_1 I_n)^{-1}\bar{R}(\bar{A} - \varphi_1 I_n)^{-T}, \tag{2.9}$$

which preserves the rank of $\bar{R}$. The optimal value of the parameter φ_1 in ADI(1) is given by

$$\varphi_1 = -\sqrt{\lambda_{\max}(\bar{A})\lambda_{\min}(\bar{A})}\,, \tag{2.10}$$

where $\lambda_{\max}, \lambda_{\min}$ denote the largest/smallest eigenvalues of $\bar{A}$ and can be easily estimated by applying a few steps of the Lanczos method [35].

Figure 2.2 shows the performance of Algorithm 1 with the preconditioner (2.9). Compared to Figure 2.1 (no preconditioner), the convergence is dramatically improved. The numerical ranks of the iterates do not grow larger than twice the nu-

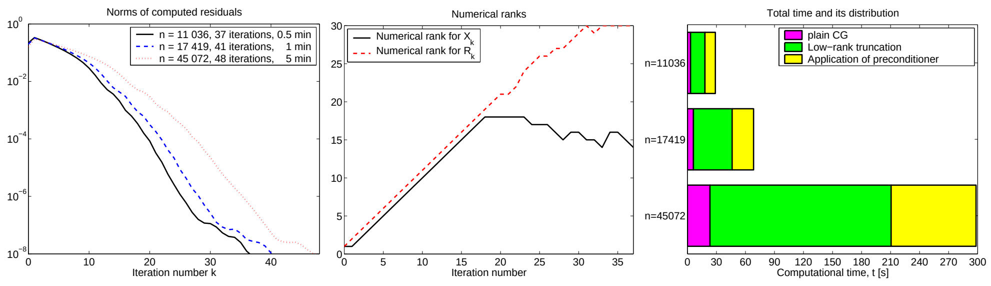


FIG. 2.2. *Algorithm 1 applied to the example from Figure 2.1 with the ADI preconditioner (2.9) for three different mesh sizes n.* Left: *Convergence of the residual norms.* Middle: *Numerical ranks for X_k and R_k for $\varepsilon = 10^{-8}$ and $n = 11\,036$.* Right: *Breakdown of the total execution time. The preprocessing step for constructing the preconditioners is not included in the graph and takes* 0.265 s, 0.532 s, *and* 2.094 s, *respectively.*

merical rank of the solution. Both improvements result into a dramatically reduced execution time: 30 seconds instead of more than 8 hours.

2.3.2. Sign function preconditioner. Instead of ADI, one can also use a few iterations of the sign function method for solving Lyapunov equations as a preconditioner. A similar idea has been successfully used for iterative refinement in the context of a hybrid CPU-GPU implementation [8].

Algorithm 3 performs the first ℓ iterations of the sign function method. A discussion on the choice of appropriate scaling parameters $\omega_j > 0$ can be found, e.g., in [38]. These parameters can be estimated during the computation of the matrices $A^{(j)}$. In particular, for $\ell = 1$, the choice $\omega_1 = \sqrt{\lambda_{\max}(\bar{A})\lambda_{\min}(\bar{A})}$ is recommended, which coincides with (2.10).

Algorithm 3 Sign(ℓ) applied to $\bar{A}\bar{Z} + \bar{Z}\bar{A}^T = \bar{R}$.

```
1: $\bar{Z}^{(0)} \leftarrow \bar{R}$,    $A^{(0)} \leftarrow \bar{A}$
2: **for** $j = 1, \dots, \ell - 1$
3:     $\bar{Z}^{(j)} \leftarrow \frac{1}{2\omega_j}(\bar{Z}^{(j-1)} + \omega_j^2 (A^{(j-1)})^{-1} \bar{Z}^{(j-1)} (A^{(j-1)})^{-T})$
4:     $A^{(j)} \leftarrow \frac{1}{2\omega_j}(A^{(j-1)} + \omega_j^2 (A^{(j-1)})^{-1})$
5: **end**
6: $\bar{Z} \leftarrow \frac{1}{2\omega_\ell}(\bar{Z}^{(\ell-1)} + \omega_\ell^2 (A^{(\ell-1)})^{-1} \bar{Z}^{(\ell-1)} (A^{(\ell-1)})^{-T})$
```

Since the iterates $A^{(j)}$, $j = 0, \dots, \ell - 1$, in Algorithm 3 are independent of the right-hand side $\bar{R}$, they can be precomputed once in a preprocessing step. A major obstacle is that the matrices $A^{(j)}$ for $j \geq 1$ cannot be represented in terms of sparse matrices and must be stored as dense matrices. This can be avoided when using a data-sparse matrix format that allows for storage-efficient (approximate) inversion and addition. Examples for such formats include hierarchical matrices [22] and hierarchically semi-separable (HSS) matrices [12, 46, 43]. Implementations of the sign function method in these formats are discussed in [5, 19, 32]. For $\ell = 1$, this is not needed. In this case, Algorithm 3 reduces to

$$\bar{Z} \leftarrow \frac{1}{2\omega_1}(\bar{R} + \omega_1^2 \bar{A}^{-1} \bar{R} \bar{A}^{-T}), \tag{2.11}$$

which can be performed efficiently via Cholesky factorizations of M and A.

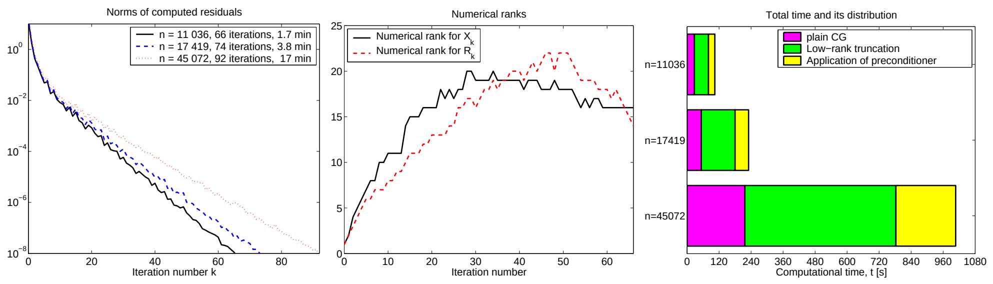


FIG. 2.3. *Algorithm 1 applied to the example from Figure 2.1 with the sign function preconditioner (2.11) for three different mesh sizes n.* Left: *Convergence of the residual norms.* Middle: *Numerical ranks for X_k and R_k for $\varepsilon = 10^{-8}$ and $n = 11\,036$.* Right: *Breakdown of the total execution time. The preprocessing step for constructing the preconditioners is not included in the graph and takes* 0.125 s, 0.250 s, *and* 0.984 s, *respectively.*

Figure 2.3 summarizes a numerical experiment with the sign function preconditioner for $\ell = 1$. For the example under consideration, the performance is worse than for the ADI-based preconditioner, see Figure 2.3, due to slower convergence of the preconditioned CG method. To test whether we could gain advantage from using $\ell > 1$, we have implemented Algorithm 3 in the hierarchical matrix format using the `HLib` library [21]. The iterates $\bar{Z}^{(j)}$ are stored in the low-rank format (2.6) and repeatedly truncated, see [5]. For the sake of simplicity, we have only tested $n = 11\,036$ and set $M = I_n$. Algorithm 1 with this preconditioner for $\ell = 2$ takes 375 s and another 318 s are needed for setting up the matrix $A^{(1)}$. This compares poorly with $\ell = 1$, which leads to a total execution time of 68 s.

3. One parameter.

In this section, we extend the preconditioned CG method discussed above to a Lyapunov equation depending on one parameter:

$$A(\alpha)X(\alpha)M(\alpha)^T + M(\alpha)X(\alpha)A(\alpha)^T = B(\alpha)B(\alpha)^T, \qquad \alpha \in \mathbb{R}. \tag{3.1}$$

More specifically, we consider the solution of (3.1) for m parameter samples $\alpha_1, \ldots, \alpha_m$ with $\alpha_{\min} \equiv \alpha_1 < \ldots < \alpha_m \equiv \alpha_{\max}$.

For each sample α_l, we consider the corresponding linear system

$$\big(M(\alpha_l) \otimes A(\alpha_l) + A(\alpha_l) \otimes M(\alpha_l)\big)\operatorname{vec}\big(X(\alpha_l)\big) = \operatorname{vec}\big(B(\alpha_l)B(\alpha_l)^T\big). \tag{3.2}$$

Similar to the approach in [29], we collect these m linear systems into one huge $mn^2 \times mn^2$ block diagonal system

$$\mathcal{A}x = b \tag{3.3}$$

with

$$\mathcal{A} = \operatorname{diag}\big((M(\alpha_1) \otimes A(\alpha_1) + A(\alpha_1) \otimes M(\alpha_1)), \ldots, \\ (M(\alpha_m) \otimes A(\alpha_m) + A(\alpha_m) \otimes M(\alpha_m))\big).$$

and

$$x = \begin{bmatrix} \operatorname{vec}\big(X(\alpha_1)\big) \\ \vdots \\ \operatorname{vec}\big(X(\alpha_m)\big) \end{bmatrix}, \qquad b = \begin{bmatrix} \operatorname{vec}\big(B(\alpha_1)B(\alpha_1)^T\big) \\ \vdots \\ \operatorname{vec}\big(B(\alpha_m)B(\alpha_m)^T\big) \end{bmatrix}.$$

We now rewrite (3.3) in terms of tensors in the sense of multidimensional arrays. For this purpose, we collect the entries of the solution and the right-hand side into two tensors $\mathcal{X}, \mathcal{B} \in \mathbb{R}^{n\times n\times m}$ with the entries

$$\mathcal{X}_{i,j,l} = \big(X(\alpha_l)\big)_{i,j}, \qquad \mathcal{B}_{i,j,l} = \big(B(\alpha_l)B(\alpha_l)^T\big)_{i,j}.$$

Then the matrix $\mathcal{A}$ can be reinterpreted as a linear operator on $\mathbb{R}^{n\times n\times m}$ and (3.3) becomes

$$\mathcal{A}(\mathcal{X}) = \mathcal{B}. \tag{3.4}$$

3.1. The Tucker format. To develop an efficient algorithm for solving (3.4), the low-rank matrix format (2.6) needs to be replaced by a low-rank format for third-order tensors. For our purposes, a suitable low-rank format is given by the Tucker format [27], which we will briefly introduce.

The Tucker format of a third-order tensor $\mathcal{S} \in \mathbb{R}^{n_1\times n_2\times n_3}$ takes the form

$$\mathrm{vec}(\mathcal{S}) = \big(U_3 \otimes U_2 \otimes U_1\big)\mathrm{vec}(\mathcal{C}), \tag{3.5}$$

where $U_i \in \mathbb{R}^{n_i\times r_i}$ for $i = 1, 2, 3$ are the *basis matrices* and $\mathcal{C} \in \mathbb{R}^{r_1\times r_2\times r_3}$ is the *core tensor*. If all U_k have full column rank then the triple (r_1, r_2, r_3) corresponds to the multilinear rank of $\mathcal{S}$.

The Tucker format (3.5) is closely linked to the three different matricizations of $\mathcal{S}$. The mode-1 matricization $S^{(1)} \in \mathbb{R}^{n_1\times n_2 n_3}$ is obtained by arranging the 1-mode fibers $\mathcal{S}(:, i_2, i_3) \in \mathbb{R}^{n_1}$ for $i_2 = 1, \ldots, n_2$, $i_3 = 1, \ldots, n_3$ into the columns of $S^{(1)}$. Similarly, $S^{(2)} \in \mathbb{R}^{n_2\times n_1 n_3}$ and $S^{(3)} \in \mathbb{R}^{n_3\times n_1 n_2}$ are obtained from the 2-mode fibers $\mathcal{S}(i_1, :, i_3)$ and the 3-mode fibers $\mathcal{S}(i_1, i_2, :)$, respectively. Then the multilinear rank satisfies

$$r = (r_1, r_2, r_3) = \big(\mathrm{rank}(S^{(1)}), \mathrm{rank}(S^{(2)}), \mathrm{rank}(S^{(3)})\big).$$

Moreover, the truncation of an explicitly given tensor to a given lower multilinear rank $\bar{r} = (\bar{r}_1, \bar{r}_2, \bar{r}_3)$ can be performed by means of singular value decompositions of $S^{(1)}, S^{(2)}, S^{(3)}$, the so called *higher-order singular value decomposition* (HOSVD) [15]. The obtained quasi-best approximation $\bar{\mathcal{S}}$ satisfies the error bound

$$\|\mathcal{S} - \bar{\mathcal{S}}\|^2 := \|\mathrm{vec}(\mathcal{S}) - \mathrm{vec}(\bar{\mathcal{S}})\|_2^2 \le \sum_{i=1}^{3} \sum_{j=\bar{r}_i+1}^{n_i} \sigma_j^2\big(S^{(i)}\big). \tag{3.6}$$

This shows that the neglected singular values determine the error, just as in the matrix case.

The right-hand side and solution tensors for the parameter-dependent Lyapunov equation (3.1) are symmetric in the first two indices. This property,

$$\mathcal{S}_{i_1,i_2,i_3} = \mathcal{S}_{i_2,i_1,i_3} \quad \text{for all} \quad i_1, i_2, i_3,$$

is equivalent to $S^{(1)} = S^{(2)}$. It can be easily seen that the HOSVD can be modified to preserve this symmetry in the sense that $U_1 = U_2$ holds for the basis matrices and $C^{(1)} = C^{(2)}$ holds for the core tensor. The corresponding symmetric variant of the Tucker format takes the form

$$\mathrm{vec}(\mathcal{S}) = \big(U_3 \otimes U_1 \otimes U_1\big)\mathrm{vec}(\mathcal{C}), \quad \text{with} \quad C^{(1)} = C^{(2)}. \tag{3.7}$$

3.2. Approximability of $\mathcal{X}$ in the Tucker format. In this section, we show that the solution tensor $\mathcal{X}$ of (3.1) can be well approximated in the Tucker format, provided that $t \ll n$ and the parameter dependence is sufficiently smooth. According to the error bound (3.6), this can be shown by bounding the singular values of the matricizations

$$X^{(1)} = X^{(2)} = [X(\alpha_1), \dots, X(\alpha_m)], \quad X^{(3)} = [\mathrm{vec}(X(\alpha_1)), \dots, \mathrm{vec}(X(\alpha_m))]^T.$$

For theoretical purposes, we may assume $M(\alpha) \equiv I_n$ without loss of generality, by a suitable transformation of the Lyapunov equation. Moreover, by a suitable parameter transformation, we may also assume that $\alpha \in [-1, 1]$.

The error bound for $X^{(3)}$ can be immediately obtained by applying an existing result [29, Thm 2.4] on parameter-dependent linear systems, as the columns of $X^{(3)}$ are solutions to the linear system (3.2) with the system matrix $\mathcal{A}(\alpha) = A(\alpha) \otimes I + I \otimes A(\alpha)$.

LEMMA 3.1. *Let $B(\alpha) : [-1, 1] \to \mathbb{R}^{n \times t}$ and $A(\alpha) : [-1, 1] \to \mathbb{R}^{n \times n}$ both have analytic extensions to the Bernstein ellipse $\mathcal{E}_{\rho_0}$ for some $\rho_0 > 1$, and assume that $\mathcal{A}(\alpha)$ is invertible for all $\alpha \in \mathcal{E}_{\rho_0}$. Then*

$$\sigma_k(X^{(3)}) \le \frac{2\rho\sqrt{m}}{1 - \rho^{-1}} \max_{\eta \in \partial\mathcal{E}_\rho} \|\mathcal{A}(\eta)^{-1}\|_2 \, C_B \, \rho^{-k}, \tag{3.8}$$

for any $1 < \rho < \rho_0$, where $C_B := \max_{\eta \in \partial\mathcal{E}_\rho} \|B(\eta)\|_F^2$.

In the case that $A(\alpha)$ is symmetric positive definite for all $\alpha \in [-1, 1]$, the bound (3.8) can be simplified. In particular, if $A(\alpha) = A_0 + \alpha A_1$, a perturbation argument can be used to show that

$$\max_{\eta \in \partial\mathcal{E}_\rho} \|\mathcal{A}(\eta)^{-1}\|_2 \lesssim \frac{1}{\mu_{\min} - (\rho - 1)^2 \|A_1\|_2}$$

for $\rho - 1$ sufficiently small, where $\mu_{\min}$ is the minimal value of $\lambda_{\min}(A_0 + \alpha A_1)$ for $\alpha \in [-1, 1]$.

The following theorem gives bounds for the singular values of $X^{(1)} = X^{(2)}$. The result is only shown for $t = 1$; bounds for general t can be obtained by superposition.

THEOREM 3.2. *Let $B(\alpha) : [-1, 1] \to \mathbb{R}^{n \times 1}$, $A(\alpha) : [-1, 1] \to \mathbb{R}^{n \times n}$ have analytic extensions to $\mathcal{E}_{\rho_0}$ for some $\rho_0 > 1$. Moreover, we assume that $A(\alpha)$ is symmetric positive definite on $[-1, 1]$ and remains positive definite*[1] *on $\mathcal{E}_{\rho_0}$. Then there exists a constant $C > 0$ not depending on k and B such that*

$$\sigma_{k+1}(X^{(1)}) \le C \, C_B \exp\left(-\pi \sqrt{\frac{\log(\rho)}{\log(8\kappa)}} \sqrt{k} \right),$$

for any $1 < \rho < \rho_0$, where $C_B := \max_{\eta \in \partial\mathcal{E}_\rho} \|B(\eta)\|_2^2$ and

$$\mu_{\min} = \frac{1}{2} \inf_{\alpha \in \mathcal{E}_\rho} \lambda_{\min}\big(A(\alpha) + A(\alpha)^*\big), \quad \mu_{\max} = \max_{\alpha \in [-1,1]} \lambda_{\max}\big(A(\alpha)\big), \quad \kappa = \frac{\mu_{\max}}{\mu_{\min}}.$$

[1]A general matrix $B \in \mathbb{C}^{n \times n}$ is called positive definite if its Hermitian part $(B^* + B)/2$ is positive definite.

Proof. We start by recalling existing results [20, 28, 37] on the singular value decay of $X(\alpha)$ for fixed $\alpha \in [-1,1]$. Based on the approximation of $1/x$ on the interval $x \in [2\mu_{\min}, 2\mu_{\max}]$ by sums of exponentials, one obtains a low-rank approximation

$$X(\alpha) \approx \widetilde{X}(\alpha) = \sum_{j=1}^{\widetilde{k}} g_j(\alpha) g_j(\alpha)^T, \qquad g_j(\alpha) := \sqrt{\gamma_j}\, \exp\bigl(-\omega_j A(\alpha)\bigr) B(\alpha),$$

for certain parameters $\gamma_j > 0$ and $\omega_j > 0$ independent of α. The approximation error satisfies

$$\|X(\alpha) - \widetilde{X}(\alpha)\|_F \le \frac{8\,\|B(\alpha)\|_2^2}{\mu_{\min}} \exp\left(-\frac{\pi^2}{\log(8\kappa)}\,\widetilde{k}\right).$$

For the 1-mode matricization of the corresponding tensor $\widetilde{\mathcal{X}}$, this directly implies the error bound

$$\|X^{(1)} - \widetilde{X}^{(1)}\|_F \le \frac{8\sqrt{m}}{\mu_{\min}}\, C_B \exp\left(-\frac{\pi^2}{\log(8\kappa)}\,\widetilde{k}\right). \tag{3.9}$$

By rearranging the terms contributing to each $\widetilde{X}(\alpha_l)$, we can write

$$\widetilde{X}^{(1)} = \widetilde{U}_1 \widetilde{V}_1^T + \cdots + \widetilde{U}_{\widetilde{k}} \widetilde{V}_{\widetilde{k}}^T$$

with

$$\begin{aligned}\widetilde{U}_j &= \Bigl[\|g_j(\alpha_1)\|_2 \cdot g_j(\alpha_1),\ \ldots,\ \|g_j(\alpha_m)\|_2 \cdot g_j(\alpha_m)\Bigr] \in \mathbb{R}^{n\times m}\\ \widetilde{V}_j &= \operatorname{diag}\left(\tfrac{1}{\|g_j(\alpha_1)\|_2} g_j(\alpha_1),\ \ldots,\ \tfrac{1}{\|g_j(\alpha_m)\|_2} g_j(\alpha_m)\right) \in \mathbb{R}^{nm\times m}\end{aligned}$$

for $j = 1, \ldots, \widetilde{k}$.

Note that the columns of $\widetilde{U}_j$ are evaluations of the vector-valued function

$$\|g_j(\alpha)\|_2 \cdot g_j(\alpha),$$

which has an analytic extension to $\mathcal{E}_{\rho_0}$. By [29, Corollary 2.3], there is, for every $\widehat{k} \le \min\{m,n\}$, a matrix $\widehat{U}_j \in \mathbb{R}^{n\times m}$ of rank $\widehat{k}$ such that

$$\|\widetilde{U}_j - \widehat{U}_j\|_F \le \frac{2\rho\sqrt{m}}{1-\rho^{-1}} \max_{\eta\in\partial\mathcal{E}_\rho} \|g_j(\eta)\|_2^2\, \rho^{-\widehat{k}}.$$

A classical result by Dahlquist [14] implies

$$\|g_j(\eta)\|_2 \le \sqrt{\gamma_j}\bigl\|\exp\bigl(-\omega_j A(\eta)\bigr)\bigr\|_2 \bigl\|B(\eta)\bigr\|_2 \le \sqrt{\gamma_j}\exp(-\omega_j\mu_{\min}) C_B$$

for all $\eta \in \mathcal{E}_\rho$. Then the approximation $\widehat{X}^{(1)} = \sum_{j=1}^{\widetilde{k}} \widehat{U}_j \widetilde{V}_j^T$ has rank $\widetilde{k}\cdot\widehat{k}$ and satisfies

$$\|\widetilde{X}^{(1)} - \widehat{X}^{(1)}\|_F \le \sum_{j=1}^{\widetilde{k}} \|\widetilde{U}_j - \widehat{U}_j\|_F \le \frac{2\rho\sqrt{m}}{1-\rho^{-1}} C_B \left(\sum_{j=1}^{\widetilde{k}} \gamma_j \exp(-2\omega_j\mu_{\min})\right) \rho^{-\widehat{k}}.$$

The remaining exponential sum turns out to be the approximation of $1/x$ at $x = 2\mu_{\min}$ from [13] and can be bounded by

$$\sum_{j=1}^{\widetilde{k}} \gamma_j \exp(-2\omega_j\mu_{\min}) \le \frac{1}{2\mu_{\min}} + \frac{8}{\mu_{\min}} \exp\left(-\frac{\pi^2}{\log(8\kappa)}\,\widetilde{k}\right) \le \frac{8.5}{\mu_{\min}}.$$

Hence, with $\widehat{C} = \frac{17\rho\sqrt{m}}{\mu_{\min}(1-\rho^{-1})}$ we obtain

$$\|\widetilde{X}^{(1)} - \widehat{X}^{(1)}\|_F \le \widehat{C} C_B \rho^{-\widehat{k}}. \tag{3.10}$$

Combining (3.9) with (3.10) yields

$$\begin{aligned}\|X^{(1)} - \widehat{X}^{(1)}\|_F &\le \|X^{(1)} - \widetilde{X}^{(1)}\|_F + \|\widetilde{X}^{(1)} - \widehat{X}^{(1)}\|_F \\ &\le C_1\, C_B \left(\exp\left(-\frac{\pi^2 \widetilde{k}}{\log(8\kappa)} \right) + \rho^{-\widehat{k}} \right) \\ &= C_1\, C_B \Big(\exp\big(-\tau_1 \widetilde{k}\big) + \exp\big(-\tau_2 \widehat{k}\big) \Big), \end{aligned} \tag{3.11}$$

where $C_1 = \max\{8\sqrt{m}/\mu_{\min}, \widehat{C}\}$ and $\tau_1 = \pi^2/(\log(8\kappa))$, $\tau_2 = \log(\rho)$. Recall that $\widehat{X}^{(1)}$ has rank $\bar{k} := \widetilde{k} \cdot \widehat{k}$.

For a given integer k, we now balance the influence of the two terms in (3.11) by choosing

$$\widetilde{k} := \left\lfloor \sqrt{k \frac{\tau_2}{\tau_1}} \right\rfloor, \qquad \widehat{k} := \left\lfloor \sqrt{k \frac{\tau_1}{\tau_2}} \right\rfloor.$$

Then

$$\begin{aligned}\sigma_{k+1}(X^{(1)}) &\le \sigma_{\bar{k}+1}(X^{(1)}) \le \|X^{(1)} - \widehat{X}^{(1)}\|_F \\ &\le C_1\, C_B \Big(\exp\big(-\tau_1 \widetilde{k}\big) + \exp\big(-\tau_2 \widehat{k}\big) \Big) \\ &\le C_1\, C_B \Big(\exp\big(-\tau_1 \big(\sqrt{k\tau_2/\tau_1} - 1\big)\big) + \exp\big(-\tau_2 \big(\sqrt{k\tau_1/\tau_2} - 1\big)\big) \Big) \\ &= C_1\, C_B \big(\exp(\tau_1) + \exp(\tau_2)\big) \exp\big(-\sqrt{\tau_1\tau_2 k}\big), \end{aligned}$$

which completes the proof by setting $C := C_1\big(\exp(\tau_1) + \exp(\tau_2)\big)$. □

The bounds of Lemma 3.1 and Theorem 3.2 predict a pronounced difference between the singular value decays of $X^{(3)}$ and of $X^{(1)}, X^{(2)}$. Such a significantly slower decay for $X^{(1)}, X^{(2)}$ does not appear to be an artifact of the proof, but it also shows up numerically, see Figure 3.1 below.

3.3. The CG method with low-rank truncation in Tucker format. The basic idea from Section 2 carries over in a relatively straightforward manner to the solution of the parameter-dependent Lyapunov equation (3.1). Formally, we apply the CG method to the $mn^2 \times mn^2$ linear system (3.3) and apply repeated low-rank truncation to keep the computational cost low. For this purpose, we view all iterates as tensors $\mathcal{X}_k, \mathcal{R}_k, \mathcal{P}_k, \mathcal{W}_k, \mathcal{Z}_k \in \mathbb{R}^{n\times n\times m}$ and store them in the Tucker format (3.7). Although the formal algorithmic description of this CG method is virtually identical with Algorithm 1, the efficient implementation of the required operations demands a more detailed discussion.

3.3.1. Matrix multiplication. Lines 4, 6, 15, and 17 of the tensorized variant of Algorithm 1 require the multiplication of a tensor $\mathcal{S} \in \mathbb{R}^{n\times n\times m}$ with L_M^{-1} in modes 1 and 2. In the Tucker format (3.7), this can be easily performed; the resulting tensor $\bar{\mathcal{S}}$ takes the form

$$\operatorname{vec}(\bar{\mathcal{S}}) = \big(U_3 \otimes L_M^{-1} U_1 \otimes L_M^{-1} U_1\big) \operatorname{vec}(\mathcal{C}).$$

The ADI(1) preconditioner (2.9) from Section 2.3.1 can be applied in a similar fashion, by multiplying modes 1 and 2 with the matrix $(A(\bar{\alpha}) - \varphi_1 I_n)^{-1}$, and the core tensor with the scalar $-2\varphi_1$. Here, $\bar{\alpha}$ corresponds to an average of the parameter samples and $\varphi_1 = -\sqrt{\lambda_{\max}\big(A(\bar{\alpha})\big)\lambda_{\min}\big(A(\bar{\alpha})\big)}$.

3.3.2. Addition of tensors. Given two tensors in the Tucker format,

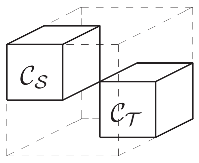
$$\operatorname{vec}(\mathcal{S}) = \big(U_3 \otimes U_1 \otimes U_1\big)\operatorname{vec}(\mathcal{C}_{\mathcal{S}}), \quad \operatorname{vec}(\mathcal{T}) = \big(V_3 \otimes V_1 \otimes V_1\big)\operatorname{vec}(\mathcal{C}_{\mathcal{T}}), \tag{3.12}$$

the sum $\bar{\mathcal{S}} = \mathcal{S} + \mathcal{T}$ can be represented by concatenating the factors:

$$\operatorname{vec}(\bar{\mathcal{S}}) = \big([U_3, V_3] \otimes [U_1, V_1] \otimes [U_1, V_1]\big)\operatorname{vec}(\mathcal{C}_{\bar{\mathcal{S}}}),$$

with

$$\mathcal{C}_{\bar{\mathcal{S}}} = \operatorname{diag}_{3\mathsf{D}}(\mathcal{C}_{\mathcal{S}}, \mathcal{C}_{\mathcal{T}}) = \tag{3.13}$$



3.3.3. Application of the linear operator $\mathcal{A}$. To discuss the efficient application of $\mathcal{A}$, we first assume that $A(\alpha) = A_0 + \alpha A_1$ is *affine linear* in α, and $M(\alpha) \equiv M$ is constant. In this case, the matrix representation of $\mathcal{A}$ takes the form

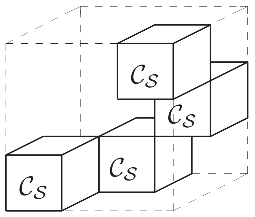
$$\mathcal{A} = I \otimes M \otimes A_0 + I \otimes A_0 \otimes M + D \otimes M \otimes A_1 + D \otimes A_1 \otimes M,$$

where $D = \operatorname{diag}(\alpha_1, \dots, \alpha_m) \in \mathbb{R}^{m \times m}$. Applied to a tensor $\mathcal{S}$, the operator $\mathcal{A}$ is a combination of matrix multiplication and addition. For $\mathcal{S}$ given in the Tucker format (3.7), the tensor $\bar{\mathcal{S}} = \mathcal{A}(\mathcal{S})$ takes the form

$$\begin{aligned}\operatorname{vec}(\bar{\mathcal{S}}) &= \big(I \otimes M \otimes A_0 + I \otimes A_0 \otimes M + D \otimes M \otimes A_1 + D \otimes A_1 \otimes M\big) \cdots \\ &\quad \big(U_3 \otimes U_1 \otimes U_1\big)\operatorname{vec}(\mathcal{C}_{\mathcal{S}}) \\ &= \big([U_3, DU_3] \otimes [A_0U_1, A_1U_1, MU_1] \otimes [A_0U_1, A_1U_1, MU_1]\big)\operatorname{vec}(\mathcal{C}_{\bar{\mathcal{S}}}),\end{aligned}$$

with

$$\mathcal{C}_{\bar{\mathcal{S}}} = \quad . \tag{3.14}$$


$\mathcal{C}_{\mathcal{S}}$
$\mathcal{C}_{\mathcal{S}}$
$\mathcal{C}_{\mathcal{S}}$
$\mathcal{C}_{\mathcal{S}}$

For general $A(\alpha), M(\alpha)$ depending nonlinearly on α, it is often also possible to apply $\mathcal{A}$ efficiently, in particular when $A(\alpha), M(\alpha)$ are low-degree matrix polynomials in α or can be reparametrized to take this form. A more detailed discussion can be found in [29].

3.3.4. Inner product. The inner product between two tensors $\mathcal{S}$ and $\mathcal{T}$ in the Tucker format (3.12) can be written as

$$\begin{aligned}\langle \mathcal{S}, \mathcal{T} \rangle &:= \langle \mathrm{vec}(\mathcal{S}), \mathrm{vec}(\mathcal{T}) \rangle \\ &= \langle (U_3 \otimes U_1 \otimes U_1)\mathrm{vec}(\mathcal{C}_\mathcal{S}), (V_3 \otimes V_1 \otimes V_1)\mathrm{vec}(\mathcal{C}_\mathcal{T}) \rangle \\ &= \langle \mathrm{vec}(\mathcal{C}_\mathcal{S}), (U_3^T V_3 \otimes U_1^T V_1 \otimes U_1^T V_1)\mathrm{vec}(\mathcal{C}_\mathcal{T}) \rangle.\end{aligned}$$

In other words, we only need to form the two small matrices $U_1^T V_1$ and $U_3^T V_3$, apply them to one of the Tucker cores, and then compute the inner product between the Tucker cores.

The computation simplifies for $\mathcal{S} = \mathcal{T}$. In particular, when the columns of U_1 and U_3 are orthonormal, we have $\|\mathcal{S}\|^2 = \langle \mathcal{S}, \mathcal{S} \rangle = \langle \mathcal{C}_\mathcal{S}, \mathcal{C}_\mathcal{S} \rangle = \|\mathcal{C}_\mathcal{S}\|^2$.

3.3.5. Low-rank truncation. Repeated addition and application of $\mathcal{A}$ lets the multilinear ranks of the iterates of the CG method quickly grow. As in Section 2, this rank growth can be limited by repeatedly truncating the iterates to lower multilinear ranks.

Given $\mathrm{vec}(\mathcal{S}) = \big(U_3 \otimes U_1 \otimes U_1\big)\mathrm{vec}(\mathcal{C})$ with $\mathcal{C} \in \mathbb{R}^{r_1 \times r_1 \times r_3}$, the first step of low-rank truncation consists of computing QR decompositions $U_1 = Q_1 R_1$ and $U_3 = Q_3 R_3$, where $Q_j \in \mathbb{R}^{n_j \times r_j}$ has orthonormal columns and $R_j \in \mathbb{R}^{r_j \times r_j}$ is upper triangular. Then

$$\mathrm{vec}(\mathcal{S}) = \big(Q_3 \otimes Q_1 \otimes Q_1\big)\mathrm{vec}(\mathcal{C}_Q), \quad \text{with} \quad \mathrm{vec}(\mathcal{C}_Q) := \big(R_3 \otimes R_1 \otimes R_1\big)\mathrm{vec}(\mathcal{C}).$$

Forming $\mathcal{C}_Q$ becomes expensive for larger (r_1, r_1, r_3). This cost can be reduced by exploiting the block structures (3.13) and (3.14) of the core tensor $\mathcal{C}$.

Upon completion of the orthogonalization step, we compress the Tucker core $\mathcal{C}_Q$ by computing SVDs of its matricizations $C_Q^{(1)}$ and $C_Q^{(3)}$. More specifically, for $\bar{r} = (\bar{r}_1, \bar{r}_1, \bar{r}_3)$ with $\bar{r}_1 \le r_1$ and $\bar{r}_3 \le r_3$, we compute the dominant left singular vectors $W_1 \in \mathbb{R}^{r_1 \times \bar{r}_1}$ and $W_3 \in \mathbb{R}^{r_3 \times \bar{r}_3}$ of $C_Q^{(1)}$ and $C_Q^{(3)}$, respectively. The truncated tensor $\bar{\mathcal{S}}$ is then obtained by projection:

$$\mathrm{vec}(\bar{\mathcal{S}}) = \big(\bar{U}_3 \otimes \bar{U}_1 \otimes \bar{U}_1\big)\mathrm{vec}(\bar{\mathcal{C}}),$$

where

$$\bar{U}_1 := U_1 W_1, \quad \bar{U}_3 := U_3 W_3, \quad \mathrm{vec}(\bar{\mathcal{C}}) := \big(W_3^T \otimes W_1^T \otimes W_1^T\big)\mathrm{vec}(\mathcal{C}_Q).$$

Using (3.6), the truncation error can be bounded by

$$\|\mathcal{S} - \bar{\mathcal{S}}\|^2 \le \sum_{i=1}^{3} \sum_{j=\bar{r}_i+1}^{r_i} \sigma_j^2\big(S^{(i)}\big).$$

The choice of $\bar{r}_i$ is based on the singular values of $S^{(i)}$. For truncating the iterates $\mathcal{X}_k$ of the CG method, we use a relative criterion of the form (2.7). For truncating $\mathcal{R}_k, \mathcal{P}_k, \mathcal{W}_k$, we use a mixed relative/absolute criterion of the form (2.8).

3.3.6. Summary of computational cost. Table 3.1 summarizes the complexity of the operations discussed above. Note that the cost for the orthogonalization step needed for low-rank truncation (see Section 3.3.5) is included in the cost for addition and application of $\mathcal{A}$. Moreover, it is assumed that $A(\alpha) = A_0 + \alpha A_1$ and $M(\alpha) \equiv M$. The cost for computing the Cholesky factorization $A(\bar{\alpha}) = LL^T$ needed for the ADI(1) preconditioner depends on the sparsity pattern of $A(\bar{\alpha})$ and is not included.

TABLE 3.1
Computational cost of individual operations in low-rank tensor variant of the CG *method applied to the $n \times n$ parameter-dependent Lyapunov equation* (3.1) *with m parameter samples. It is assumed that $n \gg r_1$ and $m \gg r_3$.*

Operation	Computational cost
Addition	$O\big(nr_1^2 + mr_3^2 + r_1^3 r_3 + r_1^2 r_3^2\big)$
Application of $\mathcal{A}(\cdot)$	$O\big(\mathsf{nnz}(A_0 + A_1 + M)r_1 + nr_1^2 + mr_3^2 + r_1^3 r_3 + r_1^2 r_3^2\big)$
Truncation	$O\big(r_1^3 r_3 + r_1^2 r_3^2 + r_3^3\big)$
Inner product	$O\big(nr_1^2 + mr_3^2 + r_1^3 r_3 + r_1^2 r_3^2\big)$
ADI(1) precond.	$O\big(\mathsf{nnz}(L) r_1\big)$

3.4. Numerical experiments. The described CG method with low-rank truncation in Tucker format has been applied to the example from [29, Sec. 2.3.1], see also Figure 2.1. Here, the parameter-dependence of the stiffness matrix $A(\alpha) = A_0 + \alpha A_1$ arises from a parametrization of the material coefficient in parts of the domain. The right-hand side is a random rank-1 matrix independent of α.

We use the ADI(1) preconditioner with the optimal shift φ_1 with respect to $\bar{\alpha} = \sqrt{\alpha_{\min}\alpha_{\max}}$. The initial guess $\mathcal{X}_0$ is set to $(\mathcal{X}_0)_{i,j,l} = (X(\bar{\alpha}))_{i,j}$, where $X(\bar{\alpha})$ is the low-rank solution of (3.1) computed by Algorithm 1.

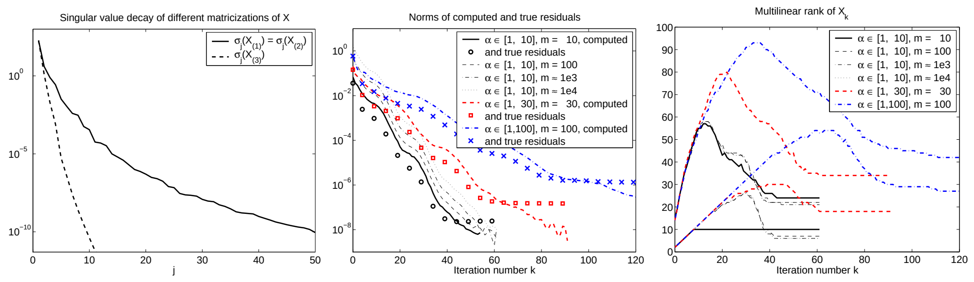


FIG. 3.1. *Solution of parameter-dependent Lyapunov equation* $(A_0+\alpha A_1)X(\alpha)M^T + MX(A_0+\alpha A_1)^T = BB^T$ *with* $n = 371$. Left: *Singular values for matricizations of* $\mathcal{X}$ *when using samples* $\alpha_j = j$, $j = 1, 2, \ldots, 100$. Middle: *Convergence of low-rank tensor* CG *measured by decay of computed (and true) residuals, and its dependence on number of samples of α, and on interval* $[\alpha_{\min}, \alpha_{\max}]$. Right: *Multilinear ranks* (r_1, r_2, r_3) *of* $\mathcal{X}_k$. *The upper graphs correspond to* $r_1 = r_2$; *the lower graphs correspond to* r_3.

Figure 3.1 and Table 3.2 display the obtained results. Compared to the corresponding results without parameter dependence (see Figure 2.2), it becomes evident that the convergence is somewhat slower and the ranks grow significantly larger. These effects become more pronounced as the parameter variation increases. This is certainly due to the fact that the average-based ADI(1) preconditioner becomes less effective.

4. Several parameters. This section introduces an extension of the low-rank tensor CG method from one to several parameters. More specifically, we consider

$$A(\alpha)X(\alpha)M(\alpha)^T + M(\alpha)X(\alpha)A(\alpha)^T = B(\alpha)B(\alpha)^T, \qquad \alpha \in \mathbb{R}^p, \tag{4.1}$$

with the parameter vector $\alpha = \big(\alpha^{(1)}, \ldots, \alpha^{(p)}\big) \in \mathbb{R}^p$. Suppose that we want to solve (4.1) for the samples

$$\alpha_{j_1,\ldots,j_p} := (\alpha_{j_1}^{(1)}, \ldots, \alpha_{j_p}^{(p)}) \quad \text{with} \quad \alpha_1^{(\ell)} < \alpha_2^{(\ell)} < \cdots < \alpha_{m_\ell}^{(\ell)}, \quad \ell = 1, \ldots, p.$$

TABLE 3.2

Detailed results for low-rank tensor CG applied to parameter-dependent Lyapunov equation. The individual columns contain: (i) parameter interval, (ii) number of samples m, (iii) number of CG iterations $k_{\max}$, (iv) multilinear rank of approximate solution after $k_{\max}$ iterations, (v) residual norm $\mathsf{res} := \frac{1}{\sqrt{m}}\|\mathcal{B}-\mathcal{A}(\mathcal{X}_{k_{\max}})\|_F$*, and (vi) total computational time.*

$[\alpha_{\min}, \alpha_{\max}]$	m	$k_{\max}$	$\mathrm{rank}(\mathcal{X}_{k_{\max}})$	res	Computational time	
$[1, 10]$	10	60	$(24, 24, 10)$	$7.21 \cdot 10^{-9}$	46 s	(0.76 s/it)
$[1, 10]$	30	60	$(22, 22, 9)$	$5.86 \cdot 10^{-9}$	2 min, 15 s	(2.25 s/it)
$[1, 10]$	100	60	$(22, 22, 7)$	$5.49 \cdot 10^{-9}$	3 min, 47 s	(3.78 s/it)
$[1, 10]$	1 000	60	$(21, 21, 6)$	$5.08 \cdot 10^{-9}$	6 min, 21 s	(6.35 s/it)
$[1, 10]$	10 000	60	$(22, 22, 6)$	$5.15 \cdot 10^{-9}$	10 min, 19 s	(10.3 s/it)
$[1, 30]$	30	90	$(34, 34, 18)$	$2.71 \cdot 10^{-8}$	8 min, 17 s	(5.53 s/it)
$[1, 100]$	100	140	$(42, 42, 27)$	$1.31 \cdot 10^{-7}$	66 min, 12 s	(28.4 s/it)
$[1, 100]$	330	140	$(38, 38, 24)$	$1.47 \cdot 10^{-7}$	95 min, 17 s	(40.8 s/it)
$[1, 100]$	1 000	140	$(33, 33, 19)$	$1.13 \cdot 10^{-7}$	114 min, 23 s	(49 s/it)

Assembling the corresponding linear systems for all samples $\alpha_{j_1,\ldots,j_p}$ into one huge linear system results in

$$\mathcal{A}(\mathcal{X}) = \mathcal{B}, \tag{4.2}$$

with $\mathcal{A} : \mathbb{R}^{n\times n\times m_1\times\cdots\times m_p} \to \mathbb{R}^{n\times n\times m_1\times\cdots\times m_p}$ and

$$\mathcal{X}_{i_1,i_2,j_1,\ldots,j_p} = \big(X(\alpha_{j_1,\ldots,j_p})\big)_{i_1,i_2}, \qquad \mathcal{B}_{i_1,i_2,j_1,\ldots,j_p} = \big(B(\alpha_{j_1,\ldots,j_p})B(\alpha_{j_1,\ldots,j_p})^T\big)_{i_1,i_2}.$$

For simplicity, we will assume affine linear parameter dependence for the rest of this section, that is, $A(\alpha) = A_0 + \sum_{\ell=1}^{p} \alpha_\ell A_\ell$, and constant $M(\alpha) \equiv M$, $B(\alpha) \equiv B$. The matrix representation of $\mathcal{A}$ then takes the form

$$\begin{aligned}\mathcal{A} &= I \otimes \cdots \otimes I \otimes I \otimes M \otimes A_0 + I \otimes \cdots \otimes I \otimes I \otimes A_0 \otimes M \\ &\quad + I \otimes \cdots \otimes I \otimes D_1 \otimes M \otimes A_1 + I \otimes \cdots \otimes I \otimes D_1 \otimes A_1 \otimes M \\ &\quad + I \otimes \cdots \otimes D_2 \otimes I \otimes M \otimes A_2 + I \otimes \cdots \otimes D_2 \otimes I \otimes A_2 \otimes M \\ &\quad + \ \cdots \\ &\quad + D_p \otimes \cdots \otimes I \otimes I \otimes M \otimes A_p + D_p \otimes \cdots \otimes I \otimes I \otimes A_p \otimes M.\end{aligned}$$

with $D_\ell = \mathrm{diag}(\alpha_1^{(\ell)}, \ldots, \alpha_{m_\ell}^{(\ell)})$, for $\ell = 1, \ldots, p$.

The Tucker format, which was used for the one-parameter case, will be replaced by the so called hierarchical Tucker format[18, 23] for the tensors of order $d = p + 2$ arising in the solution of (4.2). In the following, we will only describe some key features of the hierarchical Tucker format and refer to [18, 23, 30] for more detailed descriptions.

To understand the hierarchical Tucker format, it is necessary to extend the concept of *matricization* of a tensor from Section 3.1. Consider a tensor $\mathcal{X} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$ with modes $1, \ldots, d$, and a splitting of these modes into two disjoint sets: $\{1, \ldots, d\} = t \cup s$ with $t = \{t_1, \ldots, t_k\}$ and $s = \{s_1, \ldots, s_{d-k}\}$. Then the corresponding *matricization* of $\mathcal{X}$ is given by

$$X^{(t)} \in \mathbb{R}^{(n_{t_1}\cdots n_{t_k})\times(n_{s_1}\cdots n_{s_{d-k}})} \quad \text{with} \quad \Big(X^{(t)}\Big)_{(i_{t_1},\ldots,i_{t_k}),(i_{s_1},\ldots,i_{s_{d-k}})} := \mathcal{X}_{i_1,\ldots,i_d}$$

for any indices $i_1, \ldots, i_d$ in the multi-index set $\{1, \ldots, n_1\} \times \cdots \times \{1, \ldots, n_d\}$.

The splitting of the modes into subsets is done recursively, until each subset is a singleton. This recursive splitting is represented by a binary tree $\mathcal{T}$, where each node t is a subset of $\{1, \ldots, d\}$. The root node is given by $\{1, \ldots, d\}$ and each leaf node is a singleton, that is, $\{1\}, \ldots, \{d\}$. Each parent node in $\mathcal{T}$ is the disjoint union of its two children.

Having prescribed a maximal rank k_t for each node $t \in \mathcal{T}$, the set of *hierarchical Tucker tensors of hierarchical rank at most* $(k_t)_{t\in\mathcal{T}}$ is defined as

$$\mathcal{H}\text{-Tucker}\big((k_t)_{t\in\mathcal{T}}\big) = \Big\{\mathcal{X} \in \mathbb{R}^{n_1\times\cdots\times n_d} : \operatorname{rank}\big(X^{(t)}\big) \le k_t \text{ for all } t \in \mathcal{T}\Big\}.$$

These constraints on the rank of the matricizations $X^{(t)}, t \in \mathcal{T}$, allow $\mathcal{X}$ to be stored in the hierarchical Tucker format, with storage requirements of

$$O(dnr + dr^3),$$

where $r = \max\{r_t : t \in \mathcal{T}\}$ and $n = \max\{n_1, \ldots, n_d\}$. This is a significant improvement compared to the storage requirements of $O(dnr + r^d)$ for the Tucker format.

The low-rank variants of the CG method described in the last sections can be directly extended to work with the hierarchical Tucker format. All necessary operations can be computed efficiently in this format, e.g., the addition and inner product of two tensors. In our implementation, we have used the `htucker` toolbox [30] for MATLAB, which provides this functionality. The symmetry of $\mathcal{X}$, in the sense of

$$\mathcal{X}_{i_1,i_2,j_1,\ldots,j_p} = \mathcal{X}_{i_2,i_1,j_1,\ldots,j_p},$$

is not exploited in our implementation.

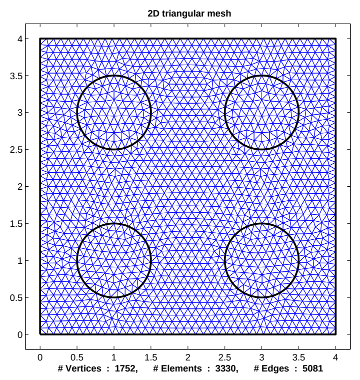
4.1. Numerical experiment. Our variant of the CG method with tensors in the hierarchical Tucker format is applied to a parameter-dependent Lyapunov equation related to [29, Sec. 4]. The matrix $A(\alpha) \in \mathbb{R}^{n\times n}$ results from the finite element discretization of the stationary heat equation on a domain $[0, 4]^2$ containing four disjoint discs (see Figure 4.1). The heat conductivity coefficient in each of these discs is governed by a parameter $\alpha^{(\ell)}$, thus $A(\alpha)$ depends on 4 parameters, and the tensor $\mathcal{X}$ is of order 6. Each parameter is sampled from the interval $[1, 10]$ in a uniform way. We have chosen sizes $n = 1\,580$ and $m_\ell = 1000$, $\ell = 1, \ldots, 4$, thus storing the tensor $\mathcal{X}$ explicitly would require $2.4964 \cdot 10^{18}$ floating point numbers. An ADI(1) preconditioner based on $A(\bar{\alpha})$ for a parameter sample average $\bar{\alpha}$ is used.

In the CG method, the following truncations are used: each iterate $\mathcal{X}_k$ is truncated such that $\|\mathcal{X}_k - \widetilde{\mathcal{X}}_k\| \le 10^{-5}\|\mathcal{X}_k\|$, with an additional requirement that the ranks may not exceed 200. The residuals $\mathcal{R}_k$ are truncated such that $\|\mathcal{R}_k - \widetilde{\mathcal{R}}_k\| \le 10^{-5}\|\mathcal{R}_0\|$, and the maximal rank is 170. The iterates $\mathcal{P}_k$ and $\mathcal{W}_k$ are truncated in an analogous manner.

Figure 4.2 displays the obtained results. The norm of the final residual satisfies

$$\frac{1}{\sqrt{m_1 m_2 m_3 m_4}}\|\mathcal{B} - \mathcal{A}(\mathcal{X}_k)\| = 3.202 \cdot 10^{-5}.$$

The total computational time for all iterations was 3 h 56 min, with 887 MB memory requirements.



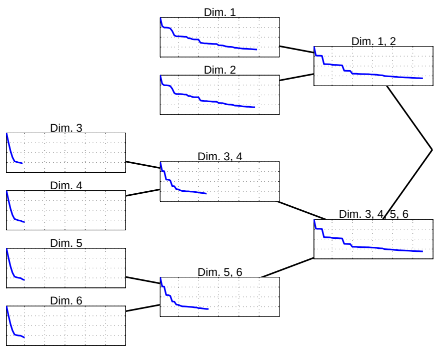
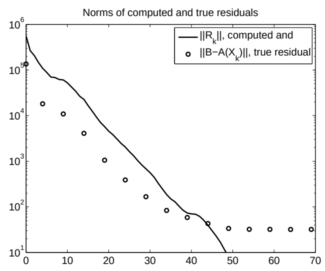


FIG. 4.1. Left: *FE mesh used in the numerical experiment. The heat conductivity coefficient in each disc depends on one of the parameters* $\alpha^{(\ell)}$, $\ell = 1,\dots,4$. Right: *Singular value decay for different matricizations of the computed tensor* $\mathcal{X}$. *The final ranks* $\mathrm{rank}(X^{(1,2)}) = \mathrm{rank}(X^{(3,4,5,6)}) = 55$, $\mathrm{rank}(X^{(3,4)}) = 24$, $\mathrm{rank}(X^{(5,6)}) = 25$, $\mathrm{rank}(X^{(1)}) = 49$, $\mathrm{rank}(X^{(2)}) = 48$, $\mathrm{rank}(X^{(3)}) = 9$, $\mathrm{rank}(X^{(4)}) = 10$, $\mathrm{rank}(X^{(5)}) = 10$, $\mathrm{rank}(X^{(6)}) = 10$.



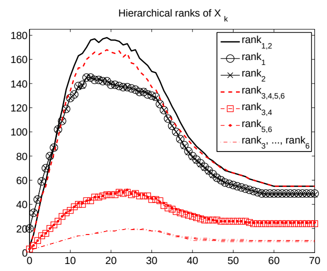


FIG. 4.2. Left: *Convergence of residual norms.* Right: *Convergence behavior for hierarchical ranks of* $\mathcal{X}_k$.

5. Conclusions and future work.

We have presented methods for solving Lyapunov matrix equations depending on no, one, or several parameters. Our methods consist of applying the preconditioned CG method combined with low-rank matrix or tensor truncation to a (huge) linear system formulation. While this point of view was mainly taken to have enough flexibility for dealing with the parameter-dependent case, the method appears to be quite competitive even for standard Lyapunov matrix equations. A more detailed comparison to existing methods based on an optimized implementation of our method remains to be done. In the parameter-dependent case, the results are promising but also indicate that the rank growth in the transient phase of the method may pose a bottleneck for more complex problems, which can only be overcome by the development of more effective preconditioners, see [26] for an example.

While this paper has focused on Lyapunov equations with symmetric coefficients, the algorithmic extension to the nonsymmetric case is relatively straightforward, by replacing the CG method with BiCGstab or restarted GMRES, as proposed in [2, 7, 16, 29]. On the other hand, it is not clear how to extend the theoretical results, in particular the bounds on the singular value decay from Theorem 3.2, to the nonsymmetric case.

As mentioned in the introduction, the solution of parameter-dependent Lyapunov equations plays an important role in interpolatory approaches to model reduction of parametrized linear control systems. The methods developed in this paper represent a first step towards efficient algorithms for such model reduction techniques.

Acknowledgments. We thank Zdeněk Strakoš, the home mentor of the second author in the SciEx project that led to this paper.

REFERENCES

[1] I. Babuška, F. Nobile, and R. Tempone, *A stochastic collocation method for elliptic partial differential equations with random input data*, SIAM J. Numer. Anal., 45 (2007), pp. 1005–1034 (electronic).
[2] J. Ballani and L. Grasedyck, *A projection method to solve linear systems in tensor format*, preprint 46, DFG-Schwerpunktprogramm 1324, May 2010. To appear at Numer. Linear Algebra Appl.
[3] R. Barrett, M. Berry, T. F. Chan, J. W. Demmel, J. Donato, J. J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, PA, 1994.
[4] U. Baur, C. Beattie, P. Benner, and S. Gugercin, *Interpolatory projection methods for parameterized model reduction*, SIAM J. Sci. Comput., 33 (2011), pp. 2489–2518.
[5] U. Baur and P. Benner, *Factorized solution of Lyapunov equations based on hierarchical matrix arithmetic*, Computing, 78 (2006), pp. 211–234.
[6] U. Baur, P. Benner, A. Greiner, J. G. Korvink, J. Lienemann, and C. Moosmann, *Parameter preserving model order reduction for MEMS applications*, Math. Comput. Model. Dyn. Syst., 17 (2011), pp. 297–317.
[7] P. Benner and T. Breiten, *Low rank methods for a class of generalized Lyapunov equations and related issues*, MPI Magdeburg Preprints MPIMD/12-03, 2012.
[8] P. Benner, P. Ezzatti, D. Kressner, E. S. Quintana-Ortí, and A. Remón, *A mixed-precision algorithm for the solution of Lyapunov equations on hybrid CPU-GPU platforms*, Parallel Comput., 37 (2011), pp. 439–450.
[9] P. Benner, J. R. Li, and N. Truhar, *On the ADI method for Sylvester equations*, Journal of Computational and Applied Mathematics, 233 (2009), pp. 1035–1045.
[10] P. Benner, H. Mena, and J. Saak, *On the parameter selection problem in the Newton-ADI iteration for large-scale Riccati equations*, Electronic Transitions on Numerical Analysis, 29 (2008), pp. 136–149.
[11] P. Benner and E. S. Quintana-Ortí, *Solving stable generalized Lyapunov equations with the matrix sign function*, Numer. Algorithms, 20 (1999), pp. 75–100.
[12] S. Börm, *$\mathcal{H}_2$-matrices – an efficient tool for the treatment of dense matrices.* Habilitationsschrift, Christian-Albrechts-Universität zu Kiel, 2006.
[13] D. Braess and W. Hackbusch, *Approximation of $1/x$ by exponential sums in* $[1, \infty)$, IMA J. Numer. Anal., 25 (2005), pp. 685–697.
[14] G. Dahlquist, *Stability and error bounds in the numerical integration of ordinary differential equations*, Kungl. Tekn. Högsk. Handl. Stockholm. No., 130 (1959), p. 87.
[15] L. De Lathauwer, B. De Moor, and J. Vandewalle, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.
[16] A. K. Eppler and M. Bollhöfer, *An alternative way of solving large Lyapunov equations*, PAMM, 10 (2010), pp. 547–548.
[17] L. Grasedyck, *Existence of a low rank or $\mathcal{H}$-matrix approximant to the solution of a Sylvester equation*, Numer. Linear Algebra Appl., 11 (2004), pp. 371–389.
[18] ———, *Hierarchical singular value decomposition of tensors*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2029–2054.
[19] L. Grasedyck, W. Hackbusch, and B. N. Khoromskij, *Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices*, Computing, 70 (2003), pp. 121–165.
[20] W. Hackbusch, *Approximation of $1/x$ by exponential sums.* Available from `http://www.mis.mpg.de/scicomp/EXP_SUM/1_x/tabelle`. Retrieved August 2008.
[21] ———, *HLib library.* `http://www.hlib.org`.
[22] ———, *Hierarchische Matrizen: Algorithmen und Analysis*, Springer, 2009.
[23] W. Hackbusch and S. Kühn, *A new scheme for the tensor representation*, J. Fourier Anal. Appl., 15 (2009), pp. 706–722.

[24] M. Hochbruck and G. Starke, *Preconditioned Krylov subspace methods for Lyapunov matrix equations*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 156–171.
[25] I. M. Jaimoukha and E. M. Kasenally, *Krylov subspace methods for solving large Lyapunov equations*, SIAM J. Numer. Anal., 31 (1994), pp. 227–251.
[26] B. N. Khoromskij and C. Schwab, *Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs*, SIAM J. Sci. Comput., 33 (2011), pp. 364–385.
[27] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500.
[28] D. Kressner and C. Tobler, *Krylov subspace methods for linear systems with tensor product structure*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1688–1714.
[29] ———, *Low-rank tensor Krylov subspace methods for parametrized linear systems*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1288–1316.
[30] ———, `htucker` *– a* MATLAB *toolbox for tensors in hierarchical Tucker format*, Tech. Rep. 2012-02, Seminar for Applied Mathematics, ETH Zurich, 2012. Available from `http://anchp.epfl.ch`.
[31] J.-R. Li and J. White, *Low-rank solution of Lyapunov equations*, SIAM Rev., 46 (2004), pp. 693–713.
[32] S. Pauli, *A numericcal solver for Lyapunov equations based on the matrix sign function iteration in HSS arithmetic*, 2010. Semester thesis, SAM, ETH Zurich.
[33] T. Penzl, *A cyclic low-rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 21 (1999), pp. 1401–1418.
[34] ———, *Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case*, Systems Control Lett., 40 (2000), pp. 139–144.
[35] T. Penzl, *Lyapack users guide*, Technical Report SFB393/00-33, Sonderforschungsbereich 393 Numerische Simulation auf massiv parallelen Rechnern, TU Chemnitz, 09107 Chemnitz, FRG, 2000.
[36] Y. Saad, *Numerical solution of large Lyapunov equations*, in Signal processing, scattering and operator theory, and numerical methods (Amsterdam, 1989), vol. 5 of Progr. Systems Control Theory, Birkhäuser Boston, Boston, MA, 1990, pp. 503–511.
[37] J. Sabino, *Solution of Large-Scale Lyapunov Equations via the Block Modified Smith Methods*, PhD thesis, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 2006.
[38] V. Sima and P. Benner, *Experimental evaluation of the new SLICOT solvers for linear matrix equations based on the matrix sign function*, in Proc. of 2008 IEEE Multi-conference on Systems and Control, 9th IEEE Int. Symp. on Computer-Aided Systems Design (CACSD), 2008, pp. 601–606.
[39] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288.
[40] V. Simoncini and V. Druskin, *Convergence analysis of projection methods for the numerical solution of large Lyapunov equations*, SIAM J. Numer. Anal., 47 (2009), pp. 828–843.
[41] Z. Strakoš and P. Tichý, *On error estimation in the conjugate gradient method and why it works in finite precision computations*, Electron. Trans. Numer. Anal., 13 (2002), pp. 56–80.
[42] E. Ullmann, *A Kronecker product preconditioner for stochastic Galerkin finite element discretization*, SIAM J. Sci. Comput., 32 (2010), pp. 923–946.
[43] R. Vandebril, M. Van Barel, and N. Mastronardi, *Matrix computations and semiseparable matrices. Vol. 1*, Johns Hopkins University Press, Baltimore, MD, 2008.
[44] E. L. Wachspress, *Optimum alternating-direction-implicit iteration parameters for a model problem*, Journal of the Society for Industrial and Applied Mathematics, 10 (1962), pp. 339–350.
[45] ———, *Extended application of alternating direction implicit iteration model problem theory*, Journal of the Society for Industrial and Applied Mathematics, 11 (1963), pp. 994–1016.
[46] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), pp. 953–976.

Recent publications :

MATHEMATICS INSTITUTE OF COMPUTATIONAL SCIENCE AND ENGINEERING
Section of Mathematics
Ecole Polytechnique Fédérale
CH-1015 Lausanne

05.2012	A. ABDULLE, G. VILLMART, KONSTANTINOS C. ZYGALAKIS: *Second weak order explicit stabilized methods for stiff stochastic differential equations*
06.2012	A. CABOUSSAT, S. BOYAVAL, A. MASSEREY: *Three-dimensional simulation of dam break flows*
07.2012	J BONNEMAIN, S. DEPARIS, A. QUARTERONI: *Connecting ventricular assist devices to the aorta: a numerical model*
08.2012	J BONNEMAIN, ELENA FAGGIANO, A. QUARTERONI, S. DEPARIS: *A framework for the analysis of the haemodynamics in patient with ventricular assist device*
09.2012	T. LASSILA, A. MANZONI, G. ROZZA: *Reduction strategies for shape dependent inverse problems in haemodynamics*
10.2012	C. MALOSSI, P. BLANCO, P. CROSETTO, S. DEPARIS, A. QUARTERONI: *Implicit coupling of one-dimensional and three-dimensional blood flow models with compliant vessels*
11.2012	S. FLOTRON J. RAPPAZ: *Conservation schemes for convection-diffusion equations with Robin's boundary conditions*
12.2012	A. UMSCHMAJEW, B. VANDEREYCKEN: *The geometry of algorithms using hierarchical tensors*
13.2012	D. KRESSNER, B. VANDEREYCKEN: *Subspace methods for computing the pseudospectral abscissa and the stability radius*
14.2012	B. JEURIS, R. VANDEBRIL, B. VANDEREYCKEN: *A survey and comparison of contemporary algorithms for computing the matrix geometric mean*
15.2012	A. MANZONI, A. QUARTERONI, G. ROZZA: *Computational reduction for parametrized PDEs: strategies and applications*
16.2012	A.C.I. MALOSSI, J. BONNEMAIN: *Numerical comparison and calibration of geometrical multiscale models for the simulation of arterial flows*
17.2012	A. ABDULLE, W. E, B. ENGQUIST, E. VANDEN-EIJNDEN: *The heterogeneous multiscale method*
18.2012	D. KRESSNER, M. PLESINGER, C. TOBLER: *A preconditioned low-rank CG method for parameter-dependent Lyapunov matrix equations*