Low-Rank Cholesky Factor Krylov Subspace Methods for Generalized Projected Lyapunov Equations

Matthias Bollhöfer* André K. Eppler†

Abstract

Large-scale descriptor systems arising from circuit simulation often require model reduction techniques. Among many methods, Balanced Truncation is a popular method for constructing a reduced order model. In the heart of Balanced Truncation methods, a sequence of projected generalized Lyapunov equations has to be solved. In this article we present a general framework for the numerical solution of projected generalized Lyapunov equations using preconditioned Krylov subspace methods based on iterates with a low-rank Cholesky factor representation. This approach can be viewed as alternative to the LRCF-ADI method, a well established method for solving Lyapunov equations. We will show that many well-known Krylov subspace methods such as (F)GMRES, QMR, BICGSTAB and CG can be easily modified to reveal the underlying low-rank structures.

1 Introduction

The numerical simulation of large-scale integrated circuits nowadays approaches system sizes of several hundred million equations. This ever-increasing size has several sources; one of which is the accelerating scale of miniaturization, another reason is the increasing density of the integrated devices. The simulation of the complete system requires many simulation runs with different input signals. These simulation runs would be impossible to compute in acceptable time using the original system. Instead it is necessary to replace the original system by a significantly smaller reduced model which inherits the essential structures and properties of the original system as, e.g., passivity and stability. To deal with this problem model order reduction techniques (MOR) have turned out to be a key technology in order to generate reduced models. Among the most popular methods for MOR are those based on Krylov subspace method or *Balanced Truncation (BT)* [3, 13, 30]. For problems arising from circuit simulation in particular passivity-preserving balanced trucation methods [33, 34, 45] are of particular interest, since beside reducing the circuit to a reduced order model, major important properties like stability and passivity have to be preserved to obtain a physically correct model. Another frequently used method mainly applied to *partial differential-algebraic equations (PDAE)* is the *Proper Orthogonal Decomposition (POD)* method, cf. [16, 26].

This article is organized as follows. In Section 2 we will give a brief introduction to balanced truncation which is the motivation for our methods and requires to solve several sequences of generalized projected Lyapunov equations. This includes existing numerical methods for solving Lyapunov equations. In Section 3 we will present our novel approach for generalized projected Lyapunov equations based on Krylov subspace methods. Finally we will use use several examples from circuit simulation, as well as other examples, to demonstrate our approach in Section 4.

*Institute for Computational Mathematics, TU Braunschweig, D-38106 Braunschweig, Germany (m.bollhoefer@tu-bs.de).
†Institute for Computational Mathematics, TU Braunschweig, D-38106 Braunschweig, Germany (a.eppler@tu-bs.de).

2 Balanced Truncation

The basis for the numerical methods for generalized projected Lyapunov equations presented in this paper are those using Balanced Truncation (BT). In particular passivity-preserving Balanced Truncation methods will be of special interest for model order reduction techniques applied to circuit simulation problems.

2.1 Introduction to Balanced Truncation

To start with the idea of Balanced Truncation we consider a linear time invariant descriptor system

$$\begin{array}{rcl} E\dot{x} & = & Ax+Bu \\ y & = & Cx+Du \end{array} \quad \text{where } A,E \in \mathbb{R}^{n,n}, B \in \mathbb{R}^{n,m}, C \in \mathbb{R}^{p,n}, D \in \mathbb{R}^{p,m}$$

such that $m, p \ll n$. Numerical methods for MOR replace E, A, B, C by smaller matrices $\tilde{E}$, $\tilde{A}$, $\tilde{B}$, $\tilde{C}$ such that for all matrices the initial dimension n is replaced by a suitable $l \ll n$, i.e., $\tilde{A}, \tilde{E} \in \mathbb{R}^{l,l}$, $\tilde{B} \in \mathbb{R}^{l,m}$, $\tilde{C} \in \mathbb{R}^{p,l}$.

When using Balanced Truncation the reduction of the model is done by multiplying with matrices $W \in \mathbb{R}^{l,n}$, $T \in \mathbb{R}^{n,l}$ in order to obtain the reduced descriptor system

$$(E,A,B,C,D) \to (\tilde{E},\tilde{A},\tilde{B},\tilde{C},D) = (WET, WAT, WB, CT, D).$$

The transformation matrices W and T are constructed using the solutions of generalized Lyapunov equations, the so-called proper controllability gramian G_{pc} and the proper observability gramian G_{po}. When E is singular one also has to take into account the improper controllability gramian and the improper observability gramian, for details we refer to [34]. For the computation of a reduced model we have to compute $X = G_{pc}$ and $Y = G_{po}$ by solving the projected generalized Lyapunov equations

$$\begin{array}{rcl} EXA^T + AXE^T + P_l BB^T P_l^T & = & 0, \text{ where } X = P_r X P_r^T, \\ E^T YA + A^T YE + P_r^T C^T C P_r & = & 0, \text{ where } Y = P_l^T Y P_l. \end{array} \tag{1}$$

Here P_l, P_r are obtained from the Weierstrass canonical form for (E,A). To do so we assume that $\det(A - \lambda E) \not\equiv 0$. In this case there exist nonsingular V and Z such that

$$V^{-1}EZ = \begin{pmatrix} I & 0 \\ 0 & N \end{pmatrix}, \; V^{-1}AZ = \begin{pmatrix} J & 0 \\ 0 & I \end{pmatrix}. \tag{2}$$

Here J, N denote matrices in Jordan canonical form where N nilpotent. The left and right projection of (E,A) to

$$(P_l E P_r, P_l A P_r) = \left(V \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} Z^{-1}, V \begin{pmatrix} J & 0 \\ 0 & 0 \end{pmatrix} Z^{-1} \right) \tag{3}$$

yields the projectors P_l and P_r of the matrix pencil $\lambda E - A$ with respect to the subspace of finite eigenvalues. By solving (1) we obtain symmetric, positive semidefinite solutions $X = RR^T$ and $Y = LL^T$, provided that the eigenvalues of J from (2) are located in the open left half plane. In many application problems for MOR in circuit simulation the matrices X, Y are numerically of approximate low rank. Using the singular value decomposition of $L^T ER$ and $L^T AR$ the balanced system is built. This way some general properties such as passivity are not necessarily preserved. To even preserve passivity it is necessary to solve the projected Lur'e equations [33]. In some special cases these in turn can be traced back to algebraic Riccati equations of the form

$$EXA^T + AXE^T + (EXC^T - P_l B)^T R^{-1} (EXC^T - P_l B) = 0, \text{ where } X = P_r X P_r^T \tag{4}$$

and

$$A^T YE + E^T YA + (B^T YE - CP_r)^T R^{-1}(B^T YE - CP_r) = 0, \text{ where } Y = P_l^T Y P_l. \tag{5}$$

For details we refer to [33, 34]. Solving Riccati equations using Newton's method or the Newton-Kleinman method [4, 44] requires solving a sequence of projected, generalized Lyapunov equations of the form

$$\begin{aligned} EX_k A_k^T + A_k X_k E^T + P_l B_k B_k^T P_l^T &= 0, \text{ where } X_k = P_r X_k P_r^T, \\ E^T Y_k A_k + A_k^T Y_k E + P_r^T C_k^T C_k P_r &= 0, \text{ where } Y_k = P_l^T Y_k P_l. \end{aligned} \tag{6}$$

Compared with the original pencil (E,A), the matrix A_k in (E,A_k) is obtained from a low-rank correction of A. For large-scale sparse systems arising from circuit simulation this allows for the computation of sparse approximations (resp. sparse factorizations) of (E,A) and then to transfer these approximations to the pencil (E,A_k) using the Sherman–Morrison–Woodbury formula [14] with respect to A_k.

2.2 Numerical Methods for Projected, Generalized Lyapunov Equations

We will now describe in detail how projected, generalized Lyapunov equations of type

$$EXA^T + AXE^T + P_l BB^T P_l^T = 0, \text{ where } X = P_r X P_r^T \tag{7}$$

are solved numerically. For simplicity we restrict ourselves to solving a single equation of this type which is at the heart of Balanced Truncation methods and in practice such equations have to be solved frequently.

One of the most commonly used methods for solving (projected) generalized Lyapunov equations is the ADI method [28, 31, 44, 47]. The ADI method for solving (7) consists of a sequence $j = 1, 2, 3, \ldots$ of steps, which is decomposed into two half-steps

$$\begin{aligned} (E + \tau_j A) X_{j-\frac{1}{2}} A^T &= -P_l BB^T P_l^T - AX_{j-1}(E - \tau_j A)^T, \\ AX_j (E + \tau_j A)^T &= -P_l BB^T P_l^T - (E - \tau_j A) X_{j-\frac{1}{2}} A^T. \end{aligned}$$

From these two coupled equations we successively compute $(X_j)_j$. Here $\tau_1, \tau_2, \tau_3, \ldots$ refer to shift parameters that have to be chosen appropriately to achieve convergence, see [32, 47]. Starting with $X_0 = 0$ and using that the right hand side $P_l BB^T P_l^T$ is symmetric and positive semidefinite one can easily verify that all iterates $X_j = R_j R_j^T$ are also symmetric and positive semidefinite. This can be used explicitly in the ADI method to represent the iterates by low rank Cholesky factors

$$R_j = \left[\sqrt{-2\mathrm{Re}(\tau_j)} \{(E + \tau_j A)^{-1} P_l B\}, \; \{(E + \tau_j A)^{-1}(E - \bar{\tau}_j A) R_{j-1}\} \right].$$

For the generalized case, the projectors P_l and P_r from (3) ensure that if $R_{j-1} = P_r R_{j-1}$, then we also obtain $R_j = P_r R_j$ and thus $X_j = P_r X_j P_r^T$ holds.

The matrices of type $(E \pm \tau_j A)$, $(E \pm \tau_j A)^{-1}$ commute with each other independent on the choice of τ_j. This observation has been used in [28] to reduce the numerical complexity of the computation of R_j by one order of magnitude. This has lead to the *Low-Rank Cholesky Factor-ADI Method (LRCF-ADI)* and can be described for the case of general and projected Lyapunov equations by Algorithm 2.1.

Algorithm 2.1 LRCF-ADI for generalized, projected Lyapunov equations 7

Compute shift parameters $\tau_1, \ldots, \tau_t$
$z_1 = \sqrt{-2\mathrm{Re}(\tau_1)}(E+\tau_1 A)^{-1} P_l B$
$R = [z_1]$
for $i = 2 \ldots t \ldots$ **do**
    $z_i = P_{i-1} = \frac{\sqrt{-2\tau_i}}{\sqrt{-2\tau_{i-1}}} \left[z_{i-1} - (\tau_i + \bar{\tau}_{i-1})(E+\tau_i A)^{-1} A z_{i-1} \right]$
    $R_i = [R_{i-i}\, z_i]$
end for

For the convergence of the ADI method the choice of the shift parameters $\tau_1, \tau_2, \ldots$ is essential. For the case where $E = I$ and $-A$ is symmetric and positive definite optimal shift parameters are known [47]. In general one often has to work with heuristic parameters as, e.g., in [31, 32] although asymptotically optimal shifts can be determined by Fejér-Walsh points [43] or Leja-Bagby points [27, 42]. Also, recent global optimization strategies to approximate optimal shifts have lead to promising results [38].

3 Low-rank Cholesky Factor Krylov Subspace Methods

The objective of this article is to describe novel numerical solution methods for projected generalized Lyapunov equations based on low-rank Krylov subspace methods. These are frequently used as core part of the model order reduction approach. In principle ADI methods belong to the class of iterative methods for solving the linear system (7). This can be equivalently rewritten as

$$\mathscr{L}\mathscr{X} = \mathscr{B},$$

where $\mathscr{L} = E \otimes A + A \otimes E$ corresponds to the Lyapunov operator in (1), $\mathscr{X} = \mathrm{vec}(X)$ and, $\mathscr{B} = \mathrm{vec}(-P_l B B^T P_l^T)$. Our goal is to preserve the matrix structure as well as the low-rank structure of the Lyapunov equation (7), while at the same time the benefits of structure-preserving preconditioned Krylov subspace methods applied to $\mathscr{L}\mathscr{X} = \mathscr{B}$ will be exploited.

3.1 Low-Rank Krylov Subspace Methods

Krylov subspace methods without preconditioning consist of series of matrix-vector multiplications, scalar products and linear combinations of vectors. The residuals $\mathscr{R}_k = \mathscr{B} - \mathscr{L}\mathscr{X}_k$ are located in $\mathrm{span}\{\mathscr{R}_0, \mathscr{L}\mathscr{R}_0, \ldots, \mathscr{L}^{k-1}\mathscr{R}_0\}$ and the approximate solutions $\mathscr{X}_{k+1}$, respectively, can be represented by elements of the space $\mathscr{X}_0 + \mathrm{span}\{\mathscr{R}_0, \mathscr{L}\mathscr{R}_0, \ldots, \mathscr{L}^{k-1}\mathscr{R}_0\}$. For two-sided Krylov subspace methods such as BiCG or QMR, multiplications with the transposed matrix also have be taken into account. Here as part of the solution process, both Riccati equations (4), (5) could be treated simultaneously solving both associated linear equations (6) in common. This follows from the property of two-sided Lanczos methods which require a right initial guess such as $P_l B_k B_k^T P_l^T$ and an appropriate left initial guess which could be chosen as $P_r^T C_k^T C_k P_r$. Yet the two-sided methods have to be slightly modified to explicitly compute the additional approximate solution. While the iterates are located in a Krylov subspace on one hand, on the other hand we have that the right hand side $-P_l B B^T P_l^T$ of the Lyapunov equation, as well as the approximate solution $X = RR^T$, can be represented as symmetric low-rank matrices. The obvious approach to migrate both structures for adapted Krylov subspace methods consists of keeping all iterates of the Krylov subspace method in symmetric low-rank format. This in turn yields elementary operations for iterates of type $Z_i = Q_i M_i Q_i^T$, where $M_i = M_i^T$, $i = 1, 2$ are also symmetric but of much smaller size than Z_i. We set $\mathscr{Z}_i = \mathrm{vec}(Z_i)$ and note that elementary operations are translated as follows:

- $\mathscr{L}\mathscr{Z}_1$ is equivalently written as

$$EZ_1A^T + AZ_1E^T = \underbrace{[EQ_1, AQ_1]}_{=:Q_2} \underbrace{\begin{bmatrix} 0 & M_1 \\ M_1 & 0 \end{bmatrix}}_{=:M_2} \underbrace{[EQ_1, AQ_1]^T}_{=:Q_2^T}$$

- analogously, $\mathscr{L}^T\mathscr{Z}_1$ is represented by

$$E^TZ_1A + A^TZ_1E = \underbrace{[E^TQ_1, A^TQ_1]}_{=:Q_2} \underbrace{\begin{bmatrix} 0 & M_1 \\ M_1 & 0 \end{bmatrix}}_{=:M_2} \underbrace{[E^TQ_1, A^TQ_1]^T}_{=:Q_2^T}$$

- linear combinations $\alpha\mathscr{Z}_1 + \beta\mathscr{Z}_2$ can be traced back to

$$\alpha Z_1 + \beta Z_2 = \underbrace{[Q_1, Q_2]}_{=:Q_3} \underbrace{\begin{bmatrix} \alpha M_1 & 0 \\ 0 & \beta M_2 \end{bmatrix}}_{=:M_3} \underbrace{[Q_1, Q_2]^T}_{=:Q_3^T}$$

- finally, scalar products are easily computed using the trace of matrices by

$$\mathscr{Z}_1^T\mathscr{Z}_2 = \operatorname{trace}(Z_1^TZ_2) = \operatorname{trace}(Z_1Z_2).$$

This shows that in principle Krylov subspace methods can be set up such that all iterates are represented by symmetric low-rank matrices.

3.2 Low-Rank Cholesky Factor Preconditioning

If we wish to supplement a Krylov subspace solver with an additional preconditioner, then in the worst case the low-rank structure of the single iterates is lost. This holds even for the simple example of diagonal preconditioning. Instead the preconditioner has to be adapted such that the low-rank structure is inherited. The natural choice for a preconditioner in this case is obtained from the LRCF-ADI method. Given $Z_1 = Q_1M_1Q_1^T$, we can apply t steps of the LRCF-ADI method from Section 2.2 starting with a right hand side Cholesky factor $B := Q_1$. This way we obtain the LRCF-ADI factors $(R_j)_{j=1,\ldots,t}$ which in turn yield a symmetric low-rank matrix

$$Q_1M_1Q_1^T \longrightarrow B := Q_1 \quad \overset{\text{LRCF-ADI}}{\underset{\text{for } B = Q_1}{\longrightarrow}} \quad R_t \longrightarrow R_t\,(I_t \otimes M_1)\,R_t^T.$$

Using ADI we obtain in a canonical way that the composed system

$$R_t\,(I_t \otimes M_1)\,R_t^T \equiv Q_2M_2Q_2^T \tag{8}$$

is again a symmetric low-rank matrix. By construction, $Q_2M_2Q_2^T$ could be equivalently computed by applying t steps of the usual ADI method starting with initial guess $X_0 = 0$ and right hand side $-Q_1M_1Q_1^T$.

There are several structure-preserving Krylov subspace methods for (generalized) Lyapunov equations which are essentially based on the (block-) Krylov subspace

$$\operatorname{span}\{B, AB, A^2B, \ldots, A^{k-1}B\},$$

see, e.g. [19–21, 23, 29, 41]. Krylov-subspace methods in conjunction with ADI preconditioning are frequently used [7, 17, 22], whereas the preservation of the low-rank structure of the iterates is not employed. Structure preservation of the GMRES and FGMRES methods ([36]) with LRCF-ADI preconditioning is further discussed in [9]. In [25] one can find a generalization of low-rank Krylov subspace methods for up to d–dimensional tensors.

3.3 Low-Rank Pseudo Arithmetic

The elementary matrix and vector operations preserve the symmetric low-rank format but numerically concatenation of symmetric low-rank matrices such as the linear combination may significantly increase the numerical rank of the iterates. To bypass this problem we need to introduce a pseudo arithmetic similar to the approach that is used for hierarchical matrices [15]. Let $Z = WMW^T$ with an additional inner small symmetric matrix $M \in \mathbb{R}^{l,l}$ be given. Z may have been obtained from one of the elementary operations described in Section 3.1 . Then Z is compressed as follows:

1. We compute $W = QR\Pi^T$, where $Q \in \mathbb{R}^{n,r}$, $R \in \mathbb{R}^{r,l}$ and $\Pi \in \mathbb{R}^{l,l}$ using the QR decomposition with column pivoting [14]. To determine the rank using this QR decomposition has to be handled with care and should include the recent modifications suggested in [8], which is the case for LAPACK release 3.2 or higher. After truncation we obtain $W \approx Q_1 R_1 \Pi^T$.

2. Next we determine the eigenvalue decomposition $T = U\Sigma U^T$ of $T = R_1 \Pi^T M \Pi R_1^T$ and reduce U, Σ to matrices U_1, Σ_1 of lower rank whenever the diagonal entries of Σ are sufficiently small in modulus.

3. This finally yields the truncated $W \approx (Q_1 U_1)\Sigma_1 (Q_1 U_1)^T$, which is computed after each elementary operation, resp. after a sequence of elementary operations.

With respect to Krylov subspace methods we usually apply the iterative solver for solving $\mathscr{L}\mathscr{X} = \mathscr{B}$ until the norm of the residual $\|\mathscr{B} - \mathscr{L}\mathscr{X}_j\|_2 \leqslant \varepsilon$. Here ε may be an absolute or relative tolerance and may include contributions from $\mathscr{B}$. For generalized Lyapunov equations this condition reads as

$$\|EX_jA^T + AX_jE^T + P_lBB^TP_l^T\|_F \leqslant \varepsilon$$

and certainly any low-rank decomposition of R_j need not be significantly more accurate than ε. Whenever $EX_jA^T + AX_jE^T + P_lBB^TP_l^T \equiv W_jM_jW_j^T$ is compressed to lower rank, it is enough to compute a truncated $QR\Pi$ decomposition. To do so assume that

$$W_j = Q_jR_j\Pi_j^T$$

such that

$$R_j = \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} = \left(\begin{array}{ccc|ccc} r_{11} & \cdots & r_{1p} & r_{1,p+1} & \cdots & r_{1,l} \\ & \ddots & \vdots & \vdots & & \vdots \\ 0 & & r_{pp} & r_{p,p+1} & \cdots & r_{p,l} \\ \hline & & & r_{p+1,p+1} & \cdots & r_{p+1,l} \\ & 0 & & \vdots & & \vdots \\ & & & r_{n,p+1} & \cdots & r_{nl} \end{array}\right).$$

The QR decomposition with column pivoting ensures that

$$|r_{11}| \geqslant \cdots \geqslant |r_{pp}| \geqslant \| \begin{pmatrix} r_{p+1,i} \\ \vdots \\ r_{n,i} \end{pmatrix} \|_2,$$

for all $i = p+1,\ldots,l$. To make sure that the residual is accurate enough we may use a threshold tol_r, which should be chosen one order of magnitude less than ε and terminate the $QR\Pi$ decomposition as soon as

$$\max_{i=p+1,\ldots,l} \| \begin{pmatrix} r_{p+1,i} \\ \vdots \\ r_{n,i} \end{pmatrix} \|_2 \leqslant \mathrm{tol}_r \,. \tag{9}$$

This requires only a minor change to the $QR\Pi$ decomposition which is truncated as soon as the threshold is reached. Q_1, R_1 are then obtained by taking the first p columns of Q_j and the leading $p \times l$ block (R_{11}, R_{12}) of R_j multiplied by Π_j^T. In a similar way all other iterates of the low-rank Krylov subspace solver will be truncated to lower rank. To summarize our truncation strategy we give a small error analysis.

Lemma 3.1 *Let $Z = WMW^T \in \mathbb{R}^{n,n}$ such that $W \in \mathbb{R}^{n,l}$, $M \in \mathbb{R}^{l,l}$ for some $l > 0$. Suppose that the truncated $QR\Pi$ decomposition of $W = QR\Pi^T$ truncates the matrix R in (9) for some $\mathrm{tol}_r = \varepsilon |r_{11}|$. Discarding R_{22}, the approximate factorization*

$$\tilde{Z} = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \Pi^T M \Pi \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}^T Q^T$$

satisfies

$$\|Z - \tilde{Z}\|_2 \leqslant 2\sqrt{l-p}\, \varepsilon \|M\|_2 \|W\|_2^2 + \mathscr{O}(\varepsilon^2).$$

Moreover, suppose that

$$T := \begin{pmatrix} R_{11} & R_{12} \end{pmatrix} \Pi^T M \Pi \begin{pmatrix} R_{11} & R_{12} \end{pmatrix}^T$$

is decomposed as

$$T = U \Sigma U^T = (U_1, U_2) \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} (U_1, U_2)^T$$

such that $U \in \mathbb{R}^{p,p}$ is orthogonal, $\Sigma_1 = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$, $\Sigma_2 = \mathrm{diag}(\sigma_{r+1}, \ldots, \sigma_p)$, $|\sigma_1| \geqslant \cdots \geqslant |\sigma_p|$ and $|\sigma_i| \leqslant \varepsilon |\sigma_1|$ for all $i > r$, then the approximate low rank factorization

$$\hat{Z} = (Q \begin{pmatrix} I_p \\ 0 \end{pmatrix} U_1)\, \Sigma_1\, (Q \begin{pmatrix} I_p \\ 0 \end{pmatrix} U_1)^T$$

satisfies

$$\|Z - \hat{Z}\|_2 \leqslant (2\sqrt{l-p}+1)\varepsilon \|M\|_2 \|W\|_2^2 + \mathscr{O}(\varepsilon^2).$$

We first note that

$$|r_{11}| = \max_{j=1,\ldots,l} \|Re_j| \leqslant \max_{\|x\|_2=1} \|Rx\|_2 = \|R\|_2 = \|W\|_2.$$

Conversely, using (9) we obtain

$$\begin{aligned} \|R_{22}\|_2 &= \max_{\|y\|_2=1} \|R_{22} y\|_2 = \max_{\|y\|_2=1} \| \sum_{i>p} R_{22} e_i y_i \|_2 \\ &\leqslant \max_{\|y\|_2=1} \sum_{i=1}^{l-p} \|R_{22} e_i\|_2 \, |y_i| \\ &\leqslant \max_{\|y\|_2=1} \left(\sum_{i=1}^{l-p} \|R_{22} e_i\|_2^2 \right)^{1/2} \left(\sum_{i=1}^{l-p} |y_i|^2 \right)^{1/2} \\ &\leqslant \left((l-p)\varepsilon^2 |r_{11}|^2 \right)^{1/2} \leqslant \sqrt{l-p}\, \varepsilon \|W\|_2. \end{aligned}$$

It follows that

$$Z-\tilde{Z} \;=\; Q\begin{pmatrix} 0 & 0 \\ 0 & R_{22} \end{pmatrix}\Pi^T M W^T + W M \Pi \begin{pmatrix} 0 & 0 \\ 0 & R_{22} \end{pmatrix}^T Q^T + Q\begin{pmatrix} 0 & 0 \\ 0 & R_{22} \end{pmatrix}\Pi^T M \Pi \begin{pmatrix} 0 & 0 \\ 0 & R_{22} \end{pmatrix}^T Q^T.$$

Thus bounding the norm of $Z-\tilde{Z}$ yields

$$\|Z-\tilde{Z}\|_2 \;\leqslant\; 2\|R_{22}\|_2\|M\|_2\,\|W\|_2 + \|R_{22}\|_2^2\|M\|_2 \leqslant 2\sqrt{l-p}\,\varepsilon\|M\|_2\|W\|_2^2 + \mathscr{O}(\varepsilon^2).$$

Next observe that $\|T\|_2 = |\sigma_1|$ and we can bound $\|T\|_2$ by

$$\|T\|_2 \leqslant \|M\|_2\,\|\begin{pmatrix} R_{11} & R_{12} \end{pmatrix}\|_2^2 \leqslant \|M\|_2\,\|W\|_2^2.$$

If we now further truncate T, then

$$\begin{aligned}
\|Z-\hat{Z}\|_2 &\leqslant \|Z-\tilde{Z}\|_2 + \|\tilde{Z}-\hat{Z}\|_2 \\
&\leqslant 2\sqrt{l-p}\,\varepsilon\|M\|_2\|W\|_2^2 + \mathscr{O}(\varepsilon^2) + \|(Q\begin{pmatrix} I_p \\ 0 \end{pmatrix}U_2)\,\Sigma_2\,(Q\begin{pmatrix} I_p \\ 0 \end{pmatrix}U_2)^T\|_2 \\
&\leqslant 2\sqrt{l-p}\,\varepsilon\|M\|_2\|W\|_2^2 + \|\Sigma_2\|_2 + \mathscr{O}(\varepsilon^2) \\
&\leqslant 2\sqrt{l-p}\,\varepsilon\|M\|_2\|W\|_2^2 + \varepsilon|\sigma_1| + \mathscr{O}(\varepsilon^2) \\
&\leqslant (2\sqrt{l-p}+1)\varepsilon\|M\|_2\|W\|_2^2 + \mathscr{O}(\varepsilon^2),
\end{aligned}$$

which completes the proof.

Although we may have $\|Z\|_2 < \|M\|_2\|W\|_2^2$ we consider this situation as rare in practice. Furthermore, the factor $\sqrt{l-p}$ is more of technical nature. Therefore using some $\tilde{\varepsilon}$ of one order of magnitude less than ε, we expect the truncation strategy to be in practice satisfactory in order to obtain $\|Z-\hat{Z}\|_2 \leq \varepsilon\|Z\|_2$. In Section 4 we will demonstrate the effectiveness of our approach.

To accommodate the preservation of symmetric low-rank matrices during elementary operations with the truncation to lower rank, a library LR-BLAS (**L**ow **R**ank-**B**asic **L**inear **A**lgebra **S**ubroutines) is designed which is summarized in Table 1.

Operation	Function Reference
$\mathscr{Y} \leftarrow \mathscr{Y} + \alpha\mathscr{X}$	`lraxpy`
$\mathscr{Y} \leftarrow \alpha\mathscr{L}\mathscr{X} + \beta\mathscr{Y}$	`lrgemv`
$\mathscr{Y} \leftarrow \alpha\mathscr{Y}$	`lrscal`
$\alpha \leftarrow \|\mathscr{Y}\|$	`lrnorm`
$\alpha \leftarrow (\mathscr{Y}, \mathscr{X})$	`lrdot`

Table 1: Overview LR-BLAS library

The introduction of low-rank BLAS allows for the easy truncation to lower rank after an elementary operation is performed. We indicate and control whether only a concatenation of matrices is built or if rank compression is required. Even when the rank is to be reduced we can internally distinguish between only using the truncated $QR\Pi$ decomposition or reducing the rank further with the help of an eigenvalue decomposition. Also, we can handle the case when one of the symmetric low-rank input matrices ($\mathscr{X}$ or

$\mathscr{Y}$) already consists of orthonormal factors $X = QMQ^T$ such that $Q^TQ = I$. In this case one can simplify the amount of work when applying the QR decomposition. Internally, it is more convenient to represent a low-rank matrix $X = QRMR^TQ^T$ rather than $X = QMQ^T$. For the sequel of this article we will skip this detail.

The introduction of a low-rank pseudo arithmetic has immediate consequences when being used for generalized projected Lyapunov equations. While concatenation of symmetric low-rank matrices does not require any additional safe guard strategy, the situation changes as soon as the rank is compressed. After each rank compression with thresholds larger than the machine precision, the projectors P_l and P_r have to be applied again. In particular iterates such as the approximate solution $X_k \approx R_kM_kR_k^T$ require a projection step $X_k \to P_rR_kM_kR_k^TP_r^T = \hat{X}_k$ while iterates like the residual have to be treated differently. Recall that we have

$$\begin{aligned} E\hat{X}_kA^T + A\hat{X}_kE^T + P_lBB^TP_l^T &= P_l(E\hat{X}_kA^T + A\hat{X}_kE^T + BB^T)P_l^T \\ &\approx S_kN_kS_k^T, \end{aligned}$$

thus here we obviously need to project with P_l to ensure that the iterates are mapped back to the correct invariant subspace associated with the finite eigenvalues of (E,A).

3.4 Approximate LRCF-ADI Preconditioning

Independent of the use of a low-rank pseudo arithmetic in Section 3.3, the explicit projection of the preconditioned iterate R_t from (8) gives the opportunity to replace the explicit inverses $(E+\tau_jA)^{-1}$ by an approximate inverse, e.g., using incomplete LU factorizations. Recall that when t steps of LRCF-ADI preconditioning are applied to a right hand side $B = P_lB$, then each iterate R_j, $j = 1,2,\ldots,t$ satisfies $R_j = P_rR_j$. This is certainly not longer fulfilled when $(E+\tau_jA)^{-1}$ is replaced by an approximation. If in doubt, in any LRCF-ADI preconditioning step substitutes

$$(E+\tau_jA)^{-1} \to P_r(\widetilde{E+\tau_jA})^{-1}$$

and explicitly projects the approximate solution back. In Section 4 we will demonstrate the effect of replacing the exact LU factorization of $E+\tau_jA$ by an ILU. At this point we like to stress that (low-rank) Krylov subspace methods are much less sensitive to the use of an ILU for $E+\tau_jA$ while the usual ADI method is much more affected.

3.5 Selected Low-Rank Krylov Subspace Methods

We now give some examples of preconditioned Krylov subspace methods adapted for generalized, projected Lyapunov equations using CFADI preconditioning. The most popular method, at least when E and A are symmetric and positive definite, is the conjugate gradient method. We will demonstrate the changes for this method first.

Suppose we wish to solve a system $\mathscr{L}\mathscr{X} = \mathscr{B}$ with a symmetric positive definite matrix $\mathscr{L}$ and a symmetric positive definite preconditioner $\tilde{\mathscr{L}} \approx \mathscr{L}$. Then the preconditioned CG method reads as given in Algorithm 3.1.

Now for symmetric and positive definite E and A we have $P_l = P_r$ and the generalized projected Lyapunov equation

$$EXA + AXE + P_lBBP_l^T = 0 \text{ where } X = P_r^TXP_r$$

induces the following preconditioned low-rank version Algorithm 3.2 with CFADI preconditioning and given shifts $\tau_1,\ldots,\tau_t$.

We will formally assume that each iterate Y is represented as $Y = Q_YM_YQ_Y^T$ for suitable matrices Q_Y and symmetric M_Y.

Algorithm 3.1 Preconditioned CG Method

Let $\mathscr{X}_0 \in \mathbb{R}^n$ be initial guess
$\mathscr{R}_0 = -\mathscr{B} - \mathscr{L}\,\mathscr{X}_0$
$\mathscr{P} = \tilde{\mathscr{L}}^{-1}\mathscr{R}_0$
for $k = 1,2,3\ldots$ **do**
  $\rho_{old} = \rho$
  $\mathscr{Z} = \mathscr{L}\,\mathscr{P}$
  $\alpha = (\mathscr{R}^T\mathscr{R})/(\mathscr{P}^T\mathscr{Z})$
  $\mathscr{X} = \mathscr{X} + \alpha\mathscr{P}$
  $\mathscr{R} = \mathscr{R} - \alpha\mathscr{Z}$
  $\mathscr{Z} = \tilde{\mathscr{L}}^{-1}\mathscr{R}$
  $\rho = \mathscr{R}^T\mathscr{Z}$
  $\beta = \rho/\rho_{old}$
  $\mathscr{P} = \mathscr{Z} + \beta\mathscr{P}$
end for

While the LR-BLAS internally apply rank compression and projection with P_l, for the preconditioning step one has to mention this explicitly to be consistent. A compression and projection step of P looks as follows.

$$P = R_t(I_t \otimes M_R)R_t^T \equiv Q_P M_P Q_P^T$$

by simple concatenation. Next the rank compression as described in Section 3.3 is performed and we obtain

$$(Q_P, M_P) \to (Q_P^{(new)}, M_P^{(new)}).$$

Eventually P_l is applied, which yields

$$Q_P \to P_l Q_P \equiv Q_P^{(new)}.$$

One may or may not add another rank compression step to Q_P as a result of the projection. But this would have to be done accurately with respect to the machine precision.

The conjugate gradient method is designed for symmetric positive definite problems. This in turn only requires P_l. In general one has to distinguish which projection has to be applied. We demonstrate that in Algorithm 3.3 for the preconditioned GMRES method [37].

We point out that the use of LR-BLAS allows to only concatenate matrices or to compress the rank. Similarly, the projection need not always be applied. We have formulated the algorithms in this more general form to indicate which projection P_l or P_r is used. The basic operation $V^{(1)} = R/\rho$ usually does neither require rank compression nor projection. But if B would not have been projected before, a projection would be required at this point. Similarly, rank compression would usually not be used as long as B does not have a rank much less than the number of columns. For the preconditioning step using t steps of LRCF-ADI, formally there is no need to project W at the end, except if the rank were compressed. Numerically however, applying the projection may reduce the influence of rounding errors from previous preconditioning steps j, $j = 1,\ldots,t$.

The GMRES method can be slightly modified to obtain the flexible GMRES method (FGMRES, [35]). In this case, W would be replaced by $W^{(l)}$ and be kept. Then X is directly computed from $W^{(1)},\ldots,W^{(m)}$ via

$$X = X + W^{(1)}y_1 + \cdots + W^{(m)}y_m \text{ using } \texttt{lraxpy}(P_r).$$

FGMRES allows for variable preconditioning. This implies that the rank in $W^{(1)},\ldots,W^{(m)}$ can be truncated with a larger tolerance tol_p than for the other iterates.

Algorithm 3.2 LR-CG for Lyapunov Equations with CFADI Preconditioning

$X_0 = 0$, $R_0 = -(P_l B)(P_l B)^T$
Compute $P = R_t (I_t \otimes M_{R_0}) R_t^T$ using t steps of LRCF-ADI applied to $B = Q_{R_0}$
Compress and project P
$\rho = \text{trace}(RP)$ using `lrdot`
for $k = 1, 2, 3 \ldots$ **do**
  $\rho_{old} = \rho$
  $Z = EPA + APE$ using `lrgemv`
  $\alpha = \|R\|_F / \text{trace}(PZ)$ using `lrnorm` and `lrdot`
  $X = X + \alpha P$ using `lraxpy`
  $R = R - \alpha Z$ using `lraxpy`
  Compute $Z = R_t (I_t \otimes M_R) R_t^T$ using t steps of LRCF-ADI applied to $B = Q_R$
  Compress and project Z
  $\rho = \text{trace}(RZ)$ using `lrdot`
  $\beta = \rho / \rho_{old}$
  $P = Z + \beta P$ using `lrscal` and `lraxpy`
end for

3.6 Reduced Lyapunov Equation

Several Arnoldi- and GMRES-like methods for Lyapunov equations essentially rely on the (block-) Krylov subspace span$\{B, AB, A^2B, \ldots, A^{k-1}B\}$ (see, e.g., [19–23]). These methods compute subspaces which replace the generalized Lyapunov equation (7) by a reduced equation

$$(WET)\,\tilde{X}\,(WAT)^T + (WAT)\,\tilde{X}\,(WET)^T + W P_l B B^T P_l^T W^T = 0.$$

The resulting approximate solution could be obtained from $X_k = P_r T \tilde{X} T^T P_r^T$. A similar approach would be possible as by product of the FGMRES method in order to obtain an alternative approximate solution. Suppose that the Arnoldi method applied to the Lyapunov operator $\mathscr{L}$ leads to the following equation

$$\mathscr{L}\mathscr{W}_m = \mathscr{V}_{m+1} \underline{\mathscr{H}}_m,$$

where $\mathscr{V}_m \in \mathbb{R}^{n^2,m}$ has orthonormal columns, $\underline{\mathscr{H}}_m \in \mathbb{R}^{m+1,m}$ is upper Hessenberg and the approximate FGMRES solution is given by $\mathscr{X}_m = \mathscr{X}_0 + \mathscr{W}_m s$ for $\mathscr{W}_m \in \mathbb{R}^{n^2,m}$. For the flexible GMRES method the columns of $\mathscr{W}_m$ are usually preconditioned counter parts of $\mathscr{V}_m$, except that the preconditioner may vary from step to step. Minimizing the norm of the residual $\mathscr{B} - \mathscr{L}\mathscr{X}_m$ for the standard GMRES method is equivalent to the minimization of

$$\|\underline{\mathscr{H}}_m y - \|\mathscr{R}_0\|_2 \cdot e_1\|_2 = \min! \tag{10}$$

Here one uses the property that the first column of $\mathscr{V}_m$ is chosen as a scalar multiple of the initial residual $\mathscr{R}_0 = \mathscr{B} - \mathscr{L}\mathscr{X}_0$. The Arnoldi vectors $\mathscr{V}_m e_k$ are rewritten in terms of symmetric low-rank matrices $V^{(k)} = Q_V^{(k)} M_V^{(k)} (Q_V^{(k)})^T$, $k = 1, \ldots, m$. Similarly, during the FGMRES method approximations to column k of $\mathscr{W}_k$ are represented by $W^{(k)} = Q_W^{(k)} M_W^{(k)} (Q_W^{(k)})^T$ from the CFADI preconditioning step. Then the numerical solution in low-rank format is a linear combination

$$X_k = X_0 + \sum_{k=1}^{m} y_k \, Q_W^{(k)} M_W^{(k)} (Q_W^{(k)})^T,$$

Algorithm 3.3 LR-GMRES for Lyapunov Equations with CFADI Preconditioning

$X_0 = 0$, $R_0 = (P_l B)(P_l B)^T$
$\rho = \|R\|_F$ using `lrnorm`
$V^{(1)} = R/\rho$ using `lrscal`(P_l)
for $k = 1, 2, 3 \ldots, m$ **do**
  Compute $W = R_t(I_t \otimes M_V^{(k)})R_t^T$ using t steps of LRCF-ADI applied to $B = Q_V^{(k)}$
  Compress and project W by P_r
  $Z = EWA^T + AWE^T$ using `lrgemv`(P_l)
  for $l = 1, 2, 3 \ldots, k$ **do**
    $h_{lk} = \text{trace}(V^{(l)}Z)$ using `lrdot`
    $Z = Z + h_{lk}V^{(l)}$ using `lraxpy`(P_l)
  end for
  $h_{k+1,k} = \|Z\|_F$ using `lrnorm`
  $V^{(k+1)} = Z/h_{k+1,k}$ using `lrscal`(P_l)
end for
Solve $\|\rho e_1 - \underline{H}_m y\|_2 = \min!$, where $\underline{H}_m = (h_{ij})_{\substack{i=1,\ldots,m+1 \\ j=1,\ldots,m}}$
$Z = V^{(1)}y_1 + \cdots + V^{(m)}y_m$ using `lraxpy`(P_l)
Compute $W = R_t(I_t \otimes M_Z)R_t^T$ using t steps of LRCF-ADI applied to $B = Q_Z$
Compress and project W by P_r
$X = X + W$ using `lraxpy`(P_r)

where the parameters $y = (y_1, \ldots, y_m)^T$ are taken from the minimization of the least squares problem (10). Alternatively the computed matrices $\left(Q_W^{(k)}\right)_k$ and $\left(Q_V^{(k)}\right)_k$ could be used to compute an alternative approximate solution $\hat{X}_k$.

Suppose that we compute a QR decomposition with column pivoting [14] to obtain

$$[Q_V^{(1)}, \ldots, Q_V^{(m)}] = Q_V R_V \Pi_V^T, \; [Q_W^{(1)}, \ldots, Q_W^{(m)}] = Q_W R_W \Pi_W^T,$$

where $\text{rank}R_V = r_V$, $\text{rank}R_W = r_W$. Similar to the compression to lower rank at other parts of the Krylov subspace method here one could work with lower accuracy as well. Let $r = \max\{r_V, r_W\}$, then the numerical solution X_k can be rewritten as

$$X_k = X_0 + Q_W S Q_W^T, \text{ where } S = R_W \Pi_W^T \begin{pmatrix} s_1 M_W^{(1)} & & 0 \\ & \ddots & \\ 0 & & s_m M_W^{(m)} \end{pmatrix} \Pi_W R_W^T.$$

Q_V and Q_W can be alternatively used to construct a reduced r-dimensional Lyapunov equation. Let

$$E_Q = Q_V^T E Q_W, A_Q = Q_V^T A Q_W$$

and compute S as numerical solution of the reduced equation

$$E_Q \, S \, A_Q^T + A_Q \, S \, E_Q^T + Q_K^T R_0 Q_K = 0,$$

where $R_0 = EX_0A^T + AX_0E^T + BB^T$. For small r this could be computed with standard methods [2]. We obtain

$$\hat{X}_m = X_0 + Q_W S Q_W^T$$

as approximate solution of a reduced Lyapunov equation. In Section 4 we will demonstrate the effectiveness of this approach.

In summary the low-rank Krylov subspace methods introduced in Section 3 allow for structured iterative methods. If (P_lEP_r, P_lAP_r) is already symmetric and P_lEP_r positive semidefinite, one could use a low-rank version of the simplified QMR (SQMR) method [11] for symmetric indefinite problems. If even P_lAP_r is positive definite, then the low-rank CG method can be applied. Low-rank CG and low-rank SQMR can make use of the CFADI preconditioning approach while at the same time low-rank structures and symmetry of the Lyapunov operator is preserved. In the general case we could easily introduce low-rank Krylov subspace methods such as low-rank BiCGStab, low-rank QMR and other methods (cf. [36]).

4 Numerical Results

In this section we will demonstrate the effectiveness of our approach. We will start with the sensitivity of low-rank Krylov subspace methods with respect to the shifts used for the CFADI preconditioning step and compare them with the usual LRCF-ADI method. Next we will demonstrate different low-rank Krylov subspace methods such as (F)GMRES, QMR and BICGSTAB for projected, generalized Lyapunov equations to evaluate their strengths and their weaknesses. We will further investigate replacing the direct solver for the single iterates $(E+\tau_j A)^{-1}$ by an approximate factorization to compare the sensitivity of ADI and Krylov subspace methods with respect to incomplete factorizations. Here we use as approximate factorization the multilevel ILU factorization from the software package [1]ILUPACK which is described in detail in [5]. Further numerical results will discuss the use of the reduced equation from Section 3.6 for the numerical solution. We will finally demonstrate how parallel direct solvers can accelerate the process of solving large-scale projected Lyapunov equations.

Some of our experiments use the software package PABTEC, see [33], which has been designed for the model order reduction of descriptor systems arising from circuit simulation. Here we replaced the default LRCF-ADI method by preconditioned low-rank Krylov subspace methods such as (F)GMRES, QMR and BICGSTAB and adapted the interfaces to allow for complete simulation runs based on Krylov subspace techniques.

4.1 Model Problems

In the following part we like to introduce three model problems which we will use for demonstration. The first two are examples arise from descriptor systems modeling circuit-equations while the third one is a more academic parabolic partial differential equation. All these examples illustrate the applicability of low-rank Krylov subspace methods.

As our first two examples we discuss linear RLC networks of the following type, modeled using the modified nodal analysis (MNA). Let e be the vector of node potentials, v_V, v_I be the voltages of the voltage sources, respectively of the current sources. Denote by i_L, i_V, i_I the currents through the inductors, voltage sources and current sources. We define the state vector x, the vector of inputs u and the output vector y via

$$x = \begin{pmatrix} e \\ i_L \\ i_V \end{pmatrix}, \; u = \begin{pmatrix} i_I \\ v_V \end{pmatrix}, \; y = \begin{pmatrix} v_I \\ i_V \end{pmatrix}.$$

[1]Matthias Bollhöfer and Yousef Saad. ILUPACK - preconditioning software package. Available online at http://ilupack.tu-bs.de/. Release V2.4, June 2011

Then the circuit equations can be written as

$$\begin{aligned} E\dot{x} &= Ax + Bu \\ y &= -B^T x, \end{aligned}$$

where E, A and B are given by

$$E = \begin{pmatrix} A_C C A_C^T & 0 & 0 \\ 0 & L & 0 \\ 0 & 0 & 0 \end{pmatrix}, A = \begin{pmatrix} -A_R G A_R^T & -A_L & -A_V \\ A_L^T & 0 & 0 \\ A_V^T & 0 & 0 \end{pmatrix}, B = \begin{pmatrix} -A_I & 0 \\ 0 & 0 \\ 0 & -I \end{pmatrix}.$$

Here A_C, A_R, A_L, A_V, A_I refer to the incidence matrices with respect to the capacitors, resistors, inductors, as well as with respect to the voltage sources and current sources. C, L, G denote the capacitance matrix, the inductance matrix and the conductivity matrix. The differential-algebraic equations which we discuss here are of differentiation index 1 (cf. [6]).

Example 4.1 *As a first example we consider a RC high pass circuit provided by NEC Laboratories Europe. It consists of* 2002 *conductors,* 2003 *resistors and* 3 *voltage sources. Using the MNA this leads to a system of dimension* 2007 *with* 3 *inputs and* 3 *outputs.*

Example 4.2 *We consider further test* [2] *examples of several RC circuits. For some details we refer to [18]. Here we restrict ourselves to examples of following sizes, reported in Table 2.*

Acronym	Capacitors	Resistors	Voltage Sources	System Size
RC1	2353	1393	109	974
RC2	3065	5892	21	3272
RC3	9999	9999	3	10002
RC4	12025	53285	78	29961

Table 2: Large-Scale RC circuits

The circuits in Table 2 are of differentiation index 2*. Since we like to demonstrate the applicability of low-rank Krylov subspace methods for index-*1 *systems we remove several voltage sources which are responsible for the higher index. After removing these voltage sources we have for circuit RC1,* 6 *voltage sources and for each circuit RC2, RC3 and RC4,* 1 *voltage source. Furthermore, we artificially add resistors with average conductivity to $A_R G A_R^T$ to make this matrix positive definite. We are aware of changing the original shape of these circuits. However, our main goal is the demonstration of low-rank Krylov subspace methods using the* ***PABTEC*** *software as framework.*

For both problem classes of RC circuits in Example 4.1 and Example 4.2 we use the technology as provided by the software package **PABTEC** (see [33]) to demonstrate solving an associated projected algebraic Riccati equation with the help of Newton's method. Here in every Newton iteration step a projected, generalized Lyapunov equation has to be solved.

Example 4.3 *The final example we will use in our numerical experiments is the parabolic partial differential equation*

$$v_t = \Delta v + \mathscr{B}u \equiv v_{xx} + v_{yy} + v_{zz} + \mathscr{B}u,$$

[2] http://sites.google.com/site/rionutiu2/research/software

where $v = v(x,y,z,t)$, $(x,y,z) \in \Omega = [0,1]^3$ and $t \geq 0$. We assume that we have some initial value $v(x,y,z,0)$ and homogeneous Dirichlet boundary conditions. To keep the discussion simple, we consider an academic control $\mathscr{B}$ such that after discretization in space using a 7-point discretization stencil, the control reduces to the vector with all ones. Suppose that we have an equidistant mesh with mesh size $h = \frac{1}{N+1}$. This leads to a total system size of $n = N^3$ unknowns. The semi-discretized ordinary differential equation is of type

$$\dot{w} = -Aw + Bu,$$

where A is the discretized Laplacian operator in three spatial dimensions. We apply model order reduction to these semi-discretized equations using balanced truncation. For symmetry reasons we simply compute the associated Gramian as the solution of the Lyapunov equation

$$XA + AX = BB^T,$$

meaning N^6 unknowns for the referring Lyapunov operator. Since A is symmetric and positive definite, the Lyapunov equation $X(-A) + (-A)X + BB^T = 0$ is stable and therefore balanced truncation can be applied. We know that the spectrum of A lies inside the interval $(3\pi^2, \frac{12}{h^2})$. This allows for a simple computation of the optimal ADI shift-parameters introduced by Wachspress [47].

We use this example in order to illustrate a low-rank version of the conjugate gradient method. Furthermore, a parallel sparse direct solver for solving the shifted systems $(A + \tau_i I)x = b$ is used to examine the scalability. Finally, this example demonstrates the advantages of using multilevel incomplete factorizations rather than direct solvers within the CFADI method.

In the sequel all computations were conducted on a 64GB Linux workstation with four Intel Xeon E7440 Quadcore processors using Matlab Release R2008b.

4.2 Different Krylov Subspace Methods and their Efficiency with Respect to the Selection of Shifts

In the following experiments we will compare how flexible GMRES [35], GMRES [37], QMR [12] and BICGSTAB [46] can be used to solve projected generalized Lyapunov equations. We will describe how different choices of shifts affect the LRCF-ADI method and low-rank Krylov subspace methods. For this purpose we consider examples 4.1 and 4.2. Here it is necessary to use the heuristic approach (referred to as *"Algorithm 1"* in [32]) for calculating the shift parameters. As part of the passivity-preserving balanced truncation we will solve the projected Riccati equations from (4), (5) up to a tolerance of 10^{-4}. The same accuracy is used for truncating the Hankel singular values for Balanced Truncation. As a heuristic approach we decided to solve each Lyapunov equation up a relative residual norm of 10^{-6}. One benefit of our class of Krylov subspace methods is that we can use the norm provided by our Krylov-subspace method and do not need to explicitly evaluate the residual-norm within the LRCF-ADI algorithm. We vary the number t of calculated shift parameters from 4, 5, 10, 20 finally to 30. For the low-rank Krylov methods we use a tolerance of 10^{-8} for truncating the ranks which is two orders of magnitude smaller than the desired residual. The number of ADI steps we display in Figures 1, 2, 3, 4 and 5 refer to the accumulated sum of all shifted systems that were solved using Newton's method.

As can be seen from Figures 1 - 5, there is neither a method that is always fastest nor is there a method always requiring the smallest number of ADI solving steps. Comparing flexible GMRES with standard GMRES, the difference in the number of ADI iterations can be explained by the different nature of these approaches. While the number of Krylov subspace iteration steps is the same, standard GMRES requires one additional solving step at the end of each restart. In contrast to this, flexible GMRES stores the preconditioned residuals explicitly and does not require an additional preconditioning step. The slightly improved

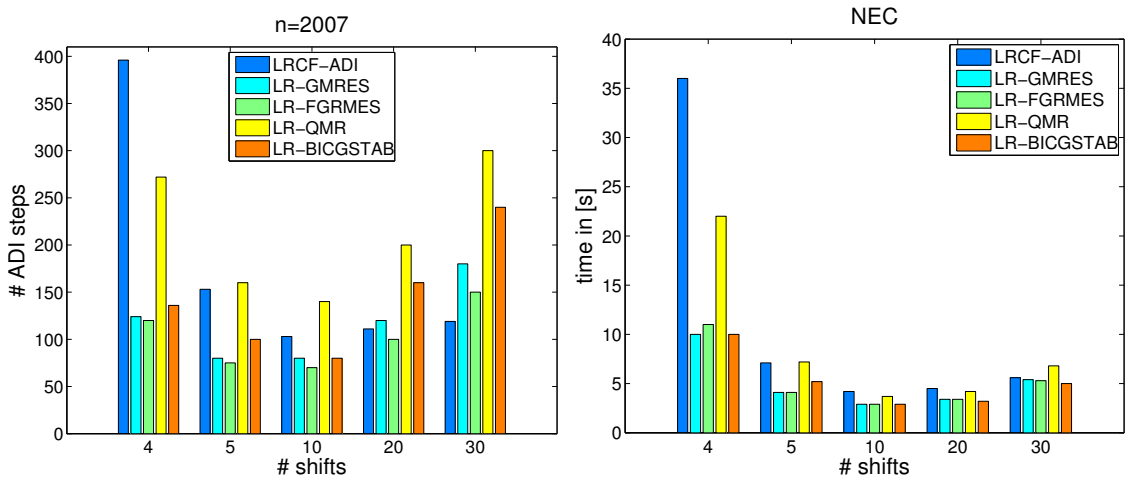


Figure 1: Number of ADI steps and runtime for Example 4.1

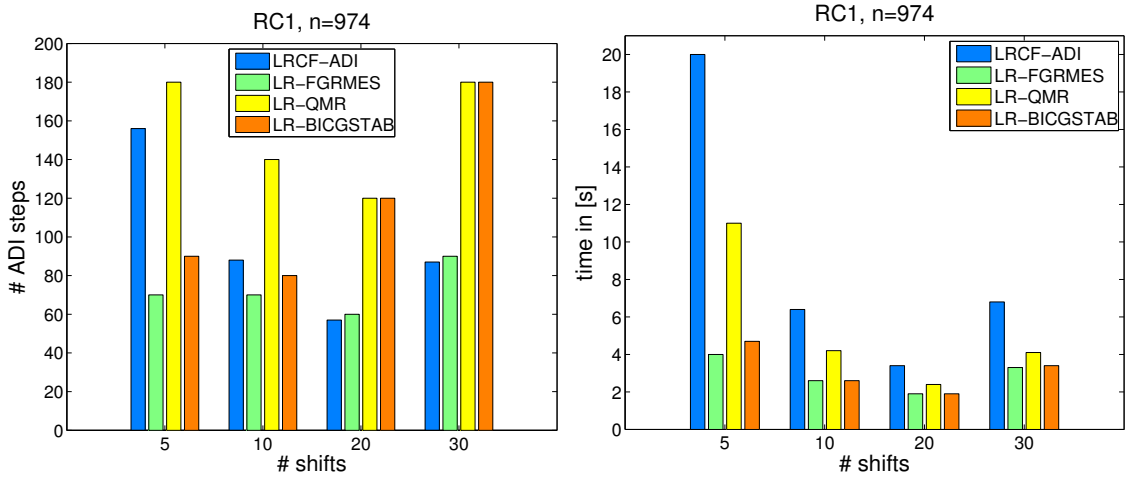


Figure 2: Number of ADI steps and runtime for circuit RC1 from Example 4.2

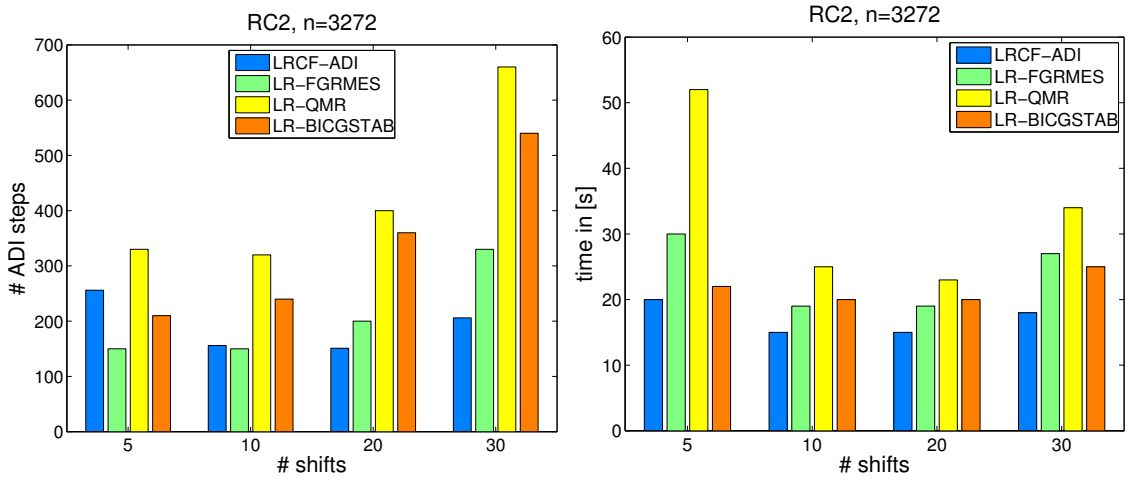


Figure 3: Number of ADI steps and runtime for circuit RC2 from Example 4.2

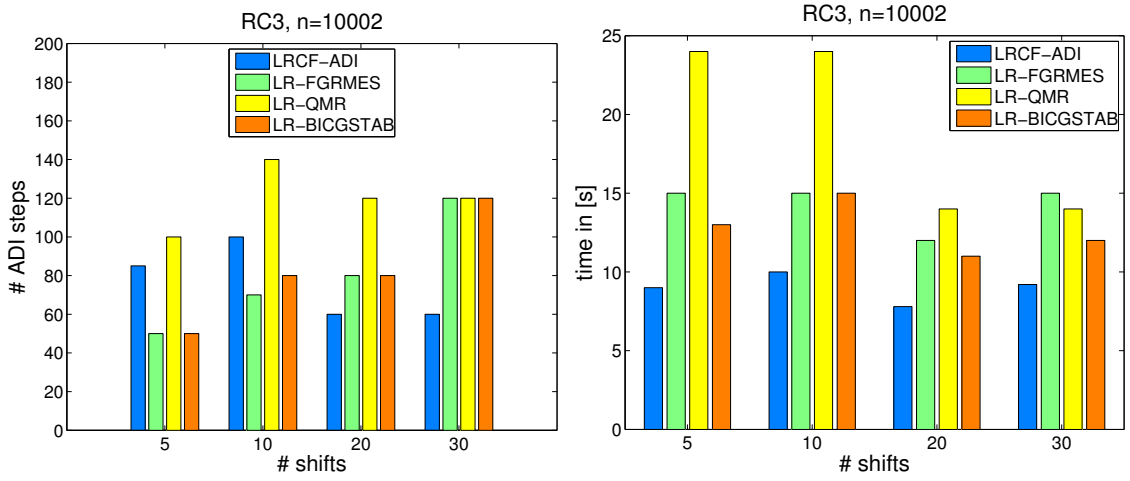


Figure 4: Number of ADI steps and runtime for circuit RC3 from Example 4.2

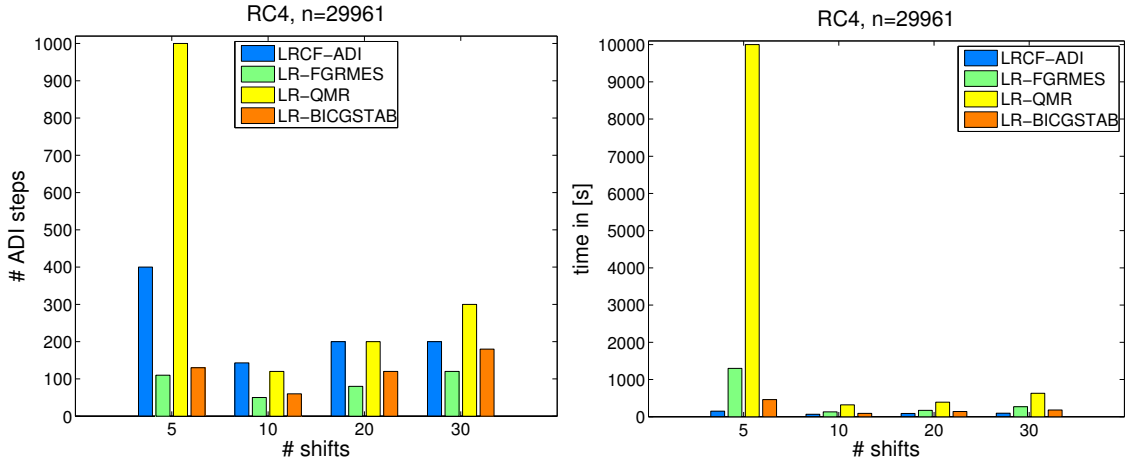


Figure 5: Number of ADI steps and runtime for circuit RC4 from Example 4.2

computation time of flexible GMRES with respect to GMRES is obtained by using twice as many vectors in low-rank format. When working with restarts this is an acceptable tradeoff so we prefer to use flexible GMRES over standard GMRES in low-rank arithmetic.

BICGSTAB and QMR require in each iteration step that either the matrix is applied twice (BICGSTAB) or the transposed matrix is used in addition (QMR). The same holds for the application of the preconditioner. Often BICGSTAB is comparable to GMRES with respect to time while QMR is typically the slowest method.

We emphasize that the number of inner iteration steps for the projected Lyapunov equations is small, when a larger number of shifts is used. When using $t = 20$ or $t = 30$ shifts, the number of inner iteration steps is typically less than 10 steps. We illustrate the relation between inner ADI solving steps and outer Newton steps in Figure 6 for the case of the LRCF-ADI method and LR-FGMRES and different numbers of shifts for Example 4.1.

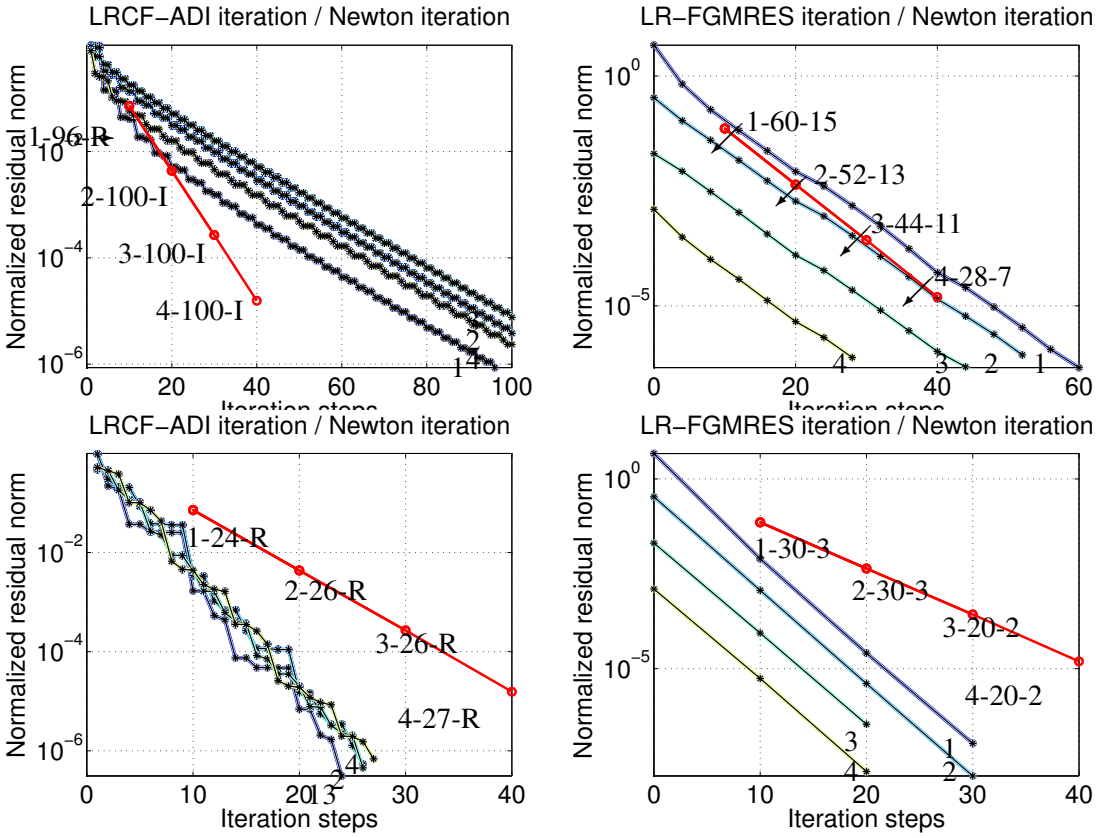
The two graphics at the top of Figure 6 refer to the use of 4 shifts while the two graphics at the bottom of Figure 6 refer to the use of 10 shifts. On the left of Figure 6 we find the LRCF-ADI method, on the right LR-FGMRES is displayed. The meaning of the three-coded numbers of type *a*-*b*-*c* in Figure 6 is explained in the legend just below the graphics.

The solid red line in Figure 6 reveals the norm of the nonlinear residual in Newton's method. The other lines display the convergence history of the residuals during the inner solves. In particular we observe that both methods, LRCF-ADI and LR-FGMRES reach the threshold 10^{-4} of the nonlinear residual after four outer steps. It can also be observed that LRCF-ADI using 4 shifts exceeds the limit 100 of inner iteration steps for solving the projected Lyapunov equation without converging. In spite of misconvergence, the outer Newton method in this case still converged to the desired accuracy.

4.3 Truncated $QR\Pi$ Decomposition

In this section we will demonstrate the difference in using the regular QR decomposition with column pivoting as implemented in LAPACK (also used inside MATLAB) with a truncated version that stops the decomposition as soon as the desired accuracy for the truncation is reached (for details cf. Section 3.3).

In Example 4.1 the main time for performing the Balanced Truncation algorithm is consumed when solving the Riccati equation. In Table 3 the computation time of the LR-FGMRES method using **PABTEC** for different numbers of shifts using the full $QR\Pi$ decomposition versus the truncated $QR\Pi$ is stated.



legend for the symbols of type *a-b-c*	
a	number of outer Newton steps
b	number of inner ADI solving steps
c	if *c* is a number, then *c* denotes the number of FGMRES steps if $c = I$, then the inner solver terminated after the number of iteration steps is exceeded if $c = R$, then ADI converged with a sufficiently small residual

Figure 6: Comparison of LRCF-ADI and LR-FGMRES using 4 (top line) and 10 (bottom line) shifts

# shifts	standard $QR\Pi$ [sec]	truncated $QR\Pi$ [sec]
4	19.25	18.43
5	7.33	6.90
10	4.03	3.90
20	3.95	3.79
30	4.24	4.20

Table 3: Comparison of standard $QR\Pi$ and truncated $QR\Pi$ within LR-FGMRES, Example 4.1

As solver for the Lyapunov equation we use LR-FGMRES. As in Section 4.2 both relative rank tolerances were set to 10^{-8} whereas we are solving the Lyapunov equations with accuracy 10^{-6}. The gain observed for using the truncated $QR\Pi$ was approximately in the range of about $5-8\%$ in overall runtime of the LR-FGMRES method.

The improvement using the truncated $QR\Pi$ decomposition can not only be used in low-rank Krylov subspace methods, but it can also have a beneficial impact of the LRCF-ADI method. When solving the projected Riccati equations using LRCF-ADI, at each Newton step we have to concatenate the current approximate low-rank solution $Z = Q_Z Q_Z^T$ of the Riccati equation and the recent low-rank update $P = Q_P Q_P^T$ from solving the projected Lyapunov equation to obtain

$$Z+P = \begin{bmatrix} Q_Z & Q_P \end{bmatrix} \begin{bmatrix} Q_Z & Q_P \end{bmatrix}^T \quad \underset{\text{compression}}{\overset{\text{rank}}{\longrightarrow}} \quad Q_Z^{(new)} (Q_Z^{(new)})^T.$$

Usually we would apply a slim QR decomposition

$$\begin{bmatrix} Q_Z & Q_P \end{bmatrix} \overset{!}{=} QR$$

such that Q has as many columns as $\begin{bmatrix} Q_Z & Q_P \end{bmatrix}$. After that we would apply a singular value decomposition

$$R \overset{!}{=} U_r \Sigma_r V_r^T$$

to truncate the rank of R to some r and obtain

$$Q_Z^{(new)} = Q U_r \Sigma_r.$$

When we use the truncated $QR\Pi$ decomposition instead, we can already compute approximately

$$\begin{bmatrix} Q_Z & Q_P \end{bmatrix} \overset{!}{=} Q_s R_s \Pi^T + E$$

such that $\|E\|$ is small and Q_s and R_s^T may already have significantly less columns s than $\begin{bmatrix} Q_Z & Q_P \end{bmatrix}$. Next a singular value decomposition only needs to be applied to the already reduced system

$$R_s \overset{!}{=} U_r \Sigma_r V_r^T.$$

Thus, the truncated $QR\Pi$ decomposition may not only save time during the $QR\Pi$ decomposition of $\begin{bmatrix} Q_Z & Q_P \end{bmatrix}$, but the singular value decomposition is also applied to system of smaller size and may lead to additional improvements. To illustrate this effect we compare the LRCF-ADI method for Examples 4.1 and 4.2. Although the total computation time is not drastically improved, at least the time of the rank compression is moderately improved. In Figures 7 and 8 we illustrate the computation times of both rank compression techniques, accumulated over all Newton steps.

We can observe a moderate to significant gain in particular when using a smaller number of shifts. When only using a smaller number of shifts, the total number of ADI steps significantly increases since the LRCF-ADI method needs more steps to converge. This in turn results in a higher pseudo rank caused by simple concatenation. Here the gain is most significant.

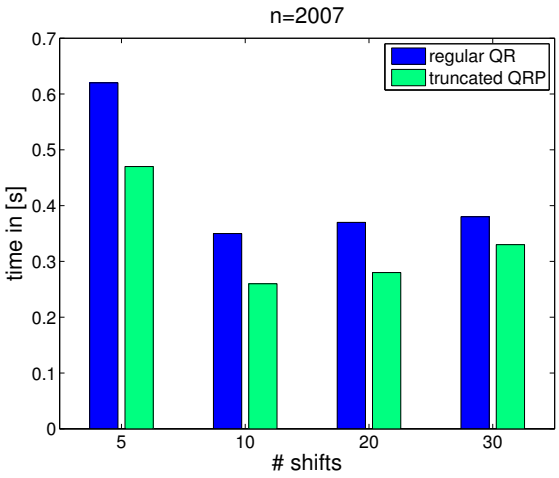


Figure 7: Computation time QR plus SVD version truncated $QR\Pi$ plus SVD. for Example 4.1

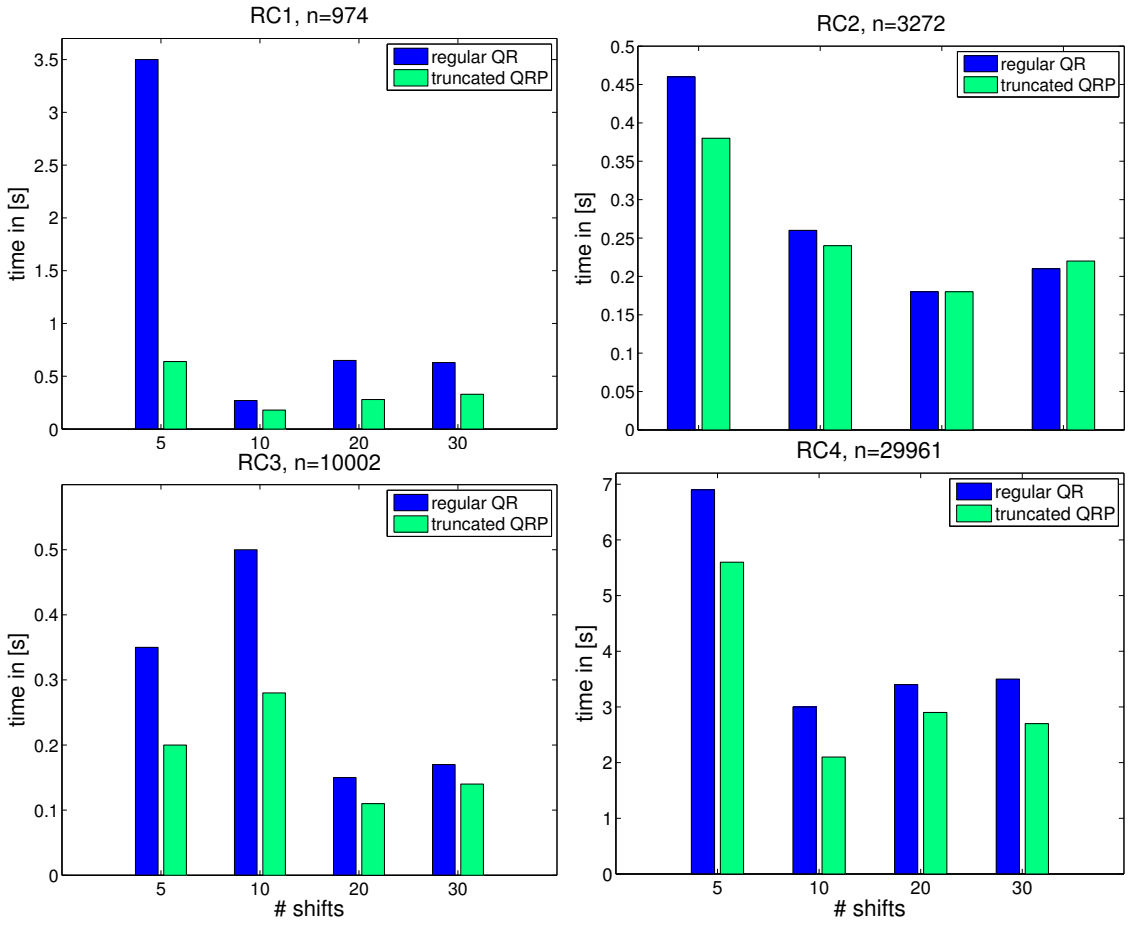


Figure 8: Computation time regular QR plus SVD implementation versus truncated $QR\Pi$ plus SVD for Example 4.2.

4.4 Evolution of the Rank Representations in the Low-Rank CG method

We will now report for the preconditioned LR-CG method from Algorithm 3.2 how ranks of the symmetric low-rank matrices X, R and P behave during the iterative process. To illustrate their behaviour we select Example 4.3 since we believe that the LR-CG method is the easiest low-rank Krylov subspace method and this example allows for the use of the preconditioned LR-CG method. We select as discretization parameter $N = 60$ which lead to a sparse symmetric positive definite matrix A of size $n = N^3 = 216'000$. The associated Lyapunov equation $X(-A) + (-A)X + BB^T = 0$ is numerically solved to obtain a low-rank symmetric positive semidefinite solution $X \in \mathbb{R}^{n,n}$. In the experiment we use a residual norm of 10^{-6} as termination criterion for the preconditioned LR-CG method. Since A is symmetric and positive definite we are able to use the optimal Wachspress shifts [47] for CFADI preconditioning. We demonstrate the behaviour of the ranks of X, R and P when using $t = 4, 6, 8$ and $t = 10$ shifts. For any of these shift values the LR-CG method only requires a few steps to converge (see Table 4).

number of shifts	4	6	8	10
number of LR-CG steps	7	5	4	3

Table 4: Number of shifts and number of preconditioned LR-CG steps for Example 4.3 and $N = 60$

In Figure 9 we illustrate the behaviour of the ranks of X, R and P in the LR-CG method, when we use a truncation tolerance of 10^{-8}.

The solid lines in Figure 9 refer to the situation where X, R and P are updated and truncated to lower rank in the LR-CG method, i.e., whenever the operations

$$
\begin{aligned}
&X = X + \alpha P \text{ using } \texttt{lraxpy} \\
&R = R - \alpha Z \text{ using } \texttt{lraxpy} \\
&\dots \\
&P = Z + \beta P \text{ using } \texttt{lrscal} \text{ and } \texttt{lraxpy}
\end{aligned}
$$

are completed within Algorithm 3.2. For X the dashed lines indicate the intermediate rank before the `lraxpy` routine compresses the rank. Similarly, for R the dashed line indicates the pseudo rank before and after the rank truncation of Z in the `lrgemv` routine that computes $Z = XA + AX$ and the situation before and after `lraxpy` compresses the rank for $R = R - \alpha Z$. Finally, the dashed line that is used for P includes the pseudo rank from the CFADI preconditioning step followed by its rank compression, as well as the additional rank compression, when $P = Z + \beta P$ is computed. We can observe for X, R and P that the intermediate ranks can be significantly higher than the rank that is obtained when `lraxpy` is completed. As we would expect, at the end of each rank compression step, the rank of X and P tends towards a constant rank, while the R the rank of the residual becomes small or even 0 when the LR-CG method converges. The general behaviour of the ranks, in particular that ranks first increase and then decrease again has also been observed in other low-rank Krylov subspace methods and applications [24]. The intermediate increase of the rank can be interpreted as another justification for using the truncated $QR\Pi$ decomposition to improve the performance of low-rank Krylov subspace methods as already illustrated in Section 4.3.

4.5 Numerical Solution based on Reduced Lyapunov Equations

The LR-FGMRES method computes orthonormal Arnoldi vectors that can be used to define a reduced projected Lyapunov equation (see Section 3.6). However, although having this reduced Lyapunov equation

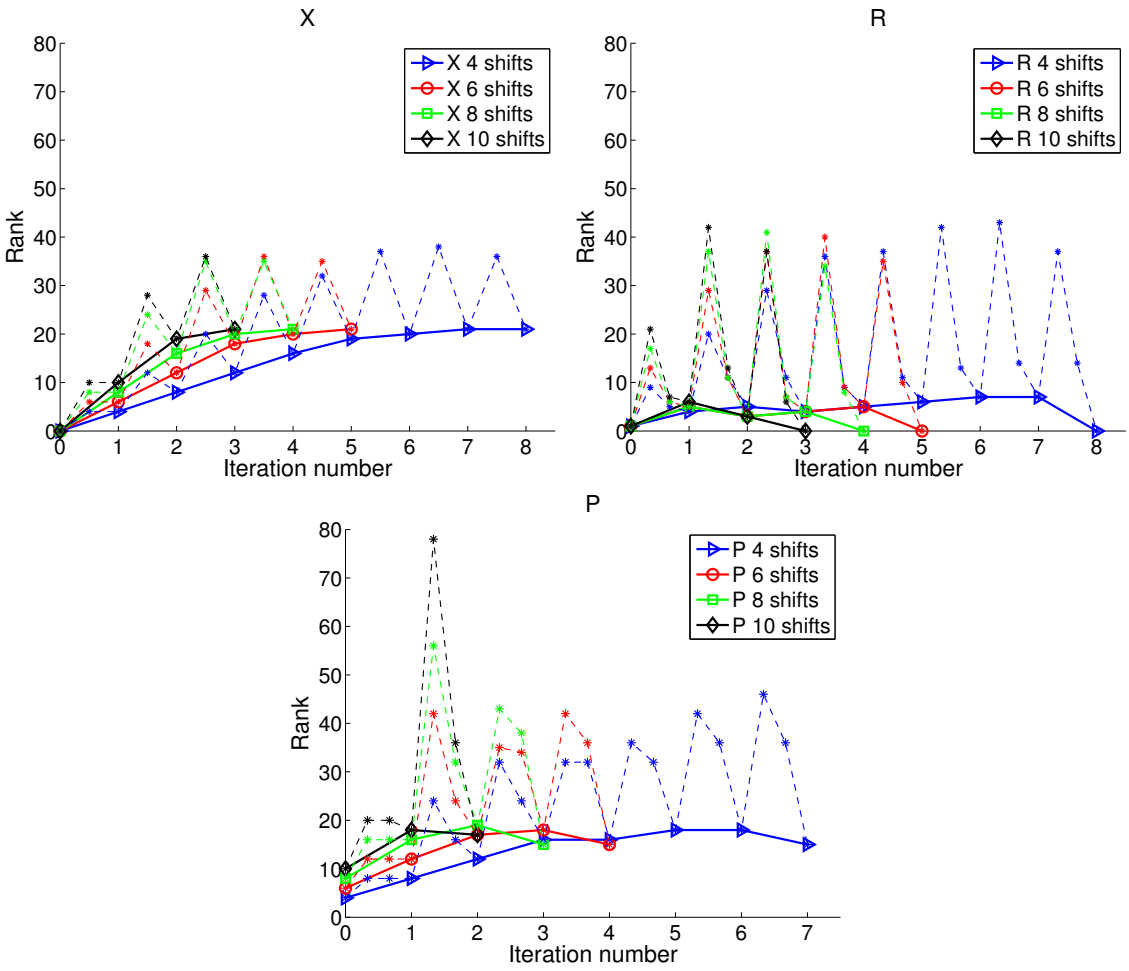


Figure 9: Evolution of ranks for different selected vectors in LR-CG for $N = 60$

available, the additional information we can extract from solving this reduced equation does not necessarily improve the low-rank solution computed via LR-GMRES. To illustrate this effect we will consider Example 4.1 using different number of shift parameters. Here we simply examine solving a simple Lyapunov equation using a tolerance of 10^{-10} for the residual and a truncation threshold for the rank of 10^{-12}.

The results are shown in Figure 10, where the norm of the residual at the end of m steps LR-FGMRES is compared with the version that uses the information of the reduced system instead.

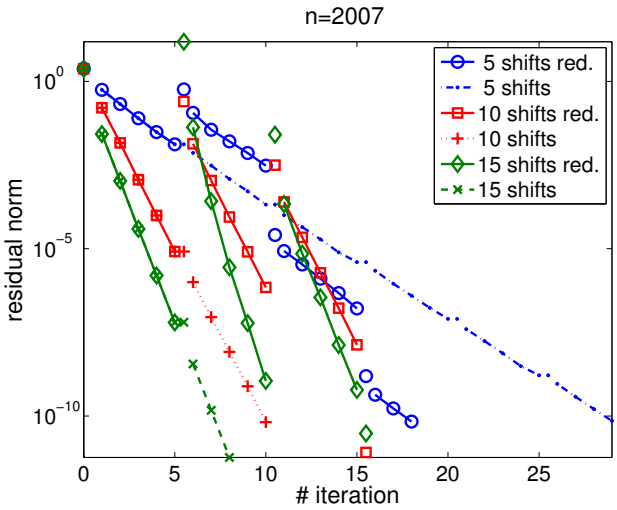


Figure 10: Norm of the residuals for LR-FGMRES using different number of shift parameters. Comparison of usual LR-FGMRES versus approximation via the reduced system.

As we can see from Figure 10 using the approximate solution from the reduced system does not necessarily improve the residual. Moreover, the computational overhead should not be overlooked. Solving the reduced system requires to solve a small projected generalized Lyapunov equation using a method such as the Bartels-Stewart algorithm. This increases the computational amount of work. For further details we refer to [9].

4.6 Incomplete LU versus LU

We now examine numerically how replacing the direct solver for $(E+\tau_j A)^{-1}$ by the multilevel ILU from ILUPACK influences the LRCF-ADI method and the LR-FGMRES method with LRCF-ADI preconditioning. First we use Example 4.1 to compare both methods inside the model order reduction software package PABTEC. Both iterative methods replace the direct solver by the ILU with a default threshold of 10^{-2} for discarding small entries. In addition, in our experiments the iterative solver inside ILUPACK (which is by default GMRES(30)) uses as termination criterion a relative residual of 10^{-4}, 10^{-8} and 10^{-12} to illustrate different accuracy of the multilevel ILU solver.

The results in Figure 11 demonstrate that in principle low-rank Krylov subspace methods can use approximate factorizations rather than direct factorization methods while the usual LRCF-ADI method encounters convergence problems which are caused by solving $(E+\tau_i A)x = b$ with lower relative accuracy.

The convergence for the results in Figure 11 is slightly delayed for LR-FGMRES while LRCF-ADI does not converge anymore. A drawback of the use of approximate factorizations that we observed in the numerical experiments is that the rank of the single iterates significantly increases [10]. This reduces the

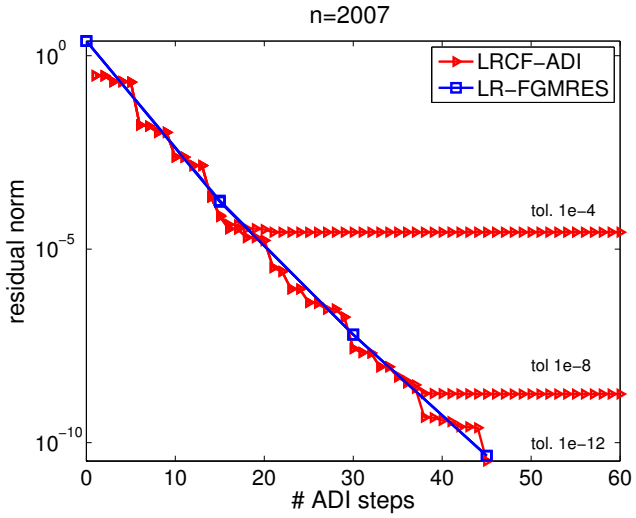


Figure 11: Norm of the residuals for LRCF-ADI and LR-FGMRES, both using incomplete CFADI preconditioning with 15 shifts.

advantages of incomplete factorizations at least for these kind of examples where direct solvers are a natural alternative. The source of this increase will be subject to future research.

As second example we consider Example 4.3 where direct solvers quickly reach their limit because of the complexity and the spatial dimension. Besides, the Lyapunov equations in this case can be numerically solved using the preconditioned LR-CG method. Firstly we will compare the memory consumption. For the comparison we will use MATLAB's `chol` function that computes a Cholesky decomposition in combination with `symamd` which initially reorders the system using the symmetric approximate minimum degree algorithm [1] in order to save fill-in. In the sequel we will refer to this version as "MATLAB". Next we use for comparison the software package[3] PARDISO [39, 40] and its Cholesky decomposition. For the incomplete factorization we will again use ILUPACK and its inverse-based multilevel incomplete Cholesky factorization with the additional option to preserve the vector with all entries equal to 1 exactly. The latter is recommended since the underlying matrix refers to a discretized elliptic partial differential equation. Since the matrix is symmetric positive definite we again use the Wachspress shifts, similar to Section 4.4. Depending on the discretization parameter N these shifts are computed with respect to a given tolerance tol_w for the desired accuracy of CFADI approximation (for details we refer to the parameter ε_1 in [28]). In Table 5 we give the explicit relation between the number of shifts depending on N and tol_w .

N	20			40			60			80		100
tol_w	10^{-1}	10^{-2}	10^{-4}	10^{-1}	10^{-2}	10^{-4}	10^{-1}	10^{-2}	10^{-4}	10^{-1}	10^{-2}	10^{-1}
shifts	3	4	8	3	5	9	4	6	10	4	6	4

Table 5: Number of ADI shifts depending on N and tol_w

In Figure 12 we display how the relative fill-in $\frac{nnz(L+L^T)}{nnz(A)}$ of the nonzero entries of the Cholesky factor L relative to the nonzero entries of A behaves with respect to the discretization size N for $\mathrm{tol}_w = 10^{-1}$ and

[3] http://www.pardiso-project.org

$\text{tol}_w = 10^{-2}$.

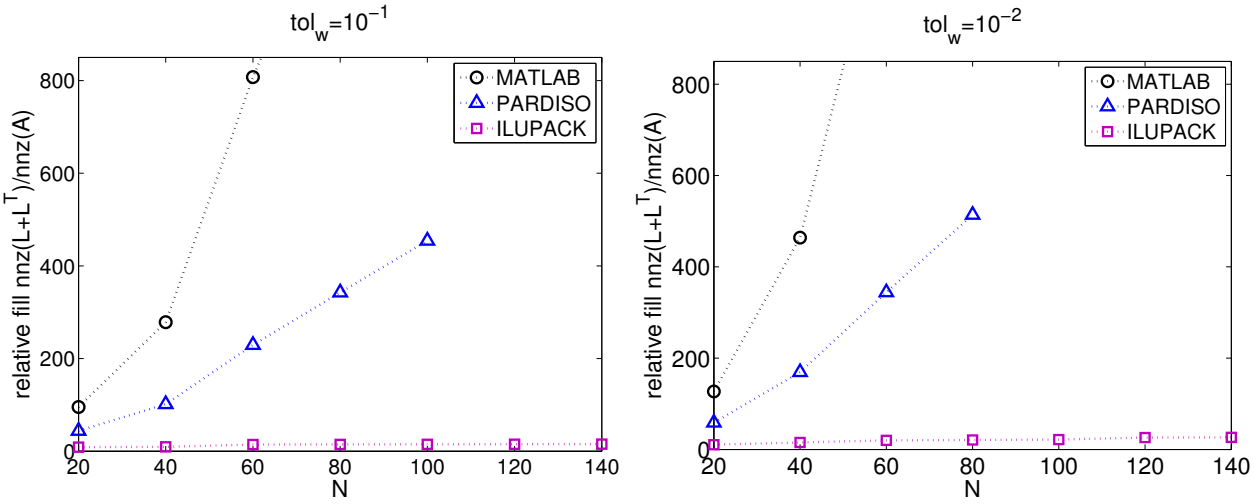


Figure 12: Memory requirement illustrated by the relative fill-in of the Cholesky factor with respect to the given matrix.

As is well-known for problems in three spatial dimensions, the relative fill-in of direct solvers drastically increases when the size of the problem increases with PARDISO being significantly better than MATLAB. In contrast to that ILUPACK yields an almost constant relative fill-in for each tol_w and also only mildly increases when tol_w is decreased (i.e., when the number of shifts is increased). The increase in the amount of fill-in is significantly sublinear! We illustrate this effect for $N = 60$. Since we need to factorize $F_i = A + \tau_i I$, for $i = 1, 2, \ldots, t$ for each shift τ_i, the system F_i is almost equivalent to A, as long as a relatively small shift τ_i is chosen. Increasing the shift τ_i in magnitude, as it is happening in the computation of the optimal ADI shift parameters, makes F_i more and more diagonal dominant. When F_i is almost equivalent to A, the multilevel ILU requires more fill-in and more levels, since in this case a multigrid-like approximation is required. With increasing diagonal dominance of F_i, the multilevel ILU gets sparser and requires less fill-in, adapting automatically to the underlying system. This explains why even increasing the number of shifts does not necessarily result in a linear increase of memory or computation time. In Table 6 we state the computation time for computing the multilevel ILU for a system of size $N^3 = 216'000$ depending on the value of the shift.

shift value	t_{factor} [sec]	levels	fill-in	t_{solve} [sec]	steps
-26999.996	2.0	1	2.2	0.8	8
-3406.818	2.5	2	3.6	1.5	10
-387.730	5.2	3	3.9	1.6	13
-48.923	6.2	5	4.7	2.1	18

Table 6: Performance of ILUPACK's multilevel ILU when four optimal shifts are prescribed, $N = 60$, $\text{tol}_w = 10^{-1}$.

We have chosen $\text{tol}_w = 10^{-1}$, which gives 4 shifts $\tau_1, \ldots, \tau_4$. For a large absolute value of $\tau_1 = -26999.996$ the system is strictly diagonal dominant. Thus only 1 level is required, the computation time is small and the relative fill-in is approximately twice as much as that of the original system. With such a large shift, solving a single system with the multilevel ILU is not only fast because of the sparse approximation, but it also re-

quires the fewest number of iteration steps (in this case 8 steps of preconditioned CG for a single right hand side). When the shift decreases in magnitude, the diagonal dominance becomes less, the number of levels increases and ILUPACK's multilevel ILU behaves more and more like an algebraic multilevel method. This can be verified by the increasing number of levels, the increasing fill-in and the slightly increasing number of CG steps.

The sublinear behaviour of the multilevel ILU is also a significant advantage with respect to the computation time when solving the Lyapunov equations using LR-CG with CFADI preconditioning. We state the computation time in Table 7.

dimension N	# shifts	MATLAB	PARDISO(1)	ILUPACK
20	3	8.5	3.5	3.5
	4	9.3	3.8	3.1
	8	12.8	4.7	2.8
40	3	596.6	144.9	75.8
	5	575.1	137.0	59.3
	9	673.8	156.6	48.4
60	4	9'236.6	1'564.4	375.8
	6	10'847.2	1'717.4	284.3
	10	11'273.8	1'879.9	271.2
80	4	78'870.4	10'562.7	1'312.7
	6	—	10'475.9	1'137.9
100	4	—	43'255.3	3'653.7
100	6	—	—	2'647.5
120	4	—	—	7'551.8
120	7	—	—	6'232.5
140	4	—	—	15'311.4
140	7	—	—	10'201.0

Table 7: Computation time LR-CG in [s] using CFADI preconditioning with different inner solvers (MATLAB / PARDISO 1 cpu / ILUPACK).

As we can see from Table 7, the computation time behaves differently for different solvers when increasing the number shifts. Using more shifts result frequently in working with higher ranks als already seen in Figure 9. This is because increasing the number of shifts only mildly increases the fill-in while at the same time the convergence speed is improved. Here ILUPACK is by far the fastest numerical solver for computing systems with $F_i = A + \tau_i I$. Looking at Table 7 we can also see that the computation time of the direct solvers scales significantly better than their memory requirement which is cause by sparse elimination technologies, such as the elimination tree, super nodes, Level-3-BLAS and cache optimization. These are techniques that are hardly applicable to incomplete factorization techniques.

4.7 Parallel Approach

We finally illustrate how the computation can be reduced for large-scale examples when the direct solver is replaced by a multi-threaded direct solver which can make use of several cores during the factorization

and the solution phase. Here we use the direct solver PARDISO [39, 40] and demonstrate the different computation times when using several threads. For this purpose we again chose Example 4.3 since here we are able to adjust/increase the dimension of the equation. As solver we use the LR-CG method since we know in this case the equivalent linear system would be symmetric and positive definite. We increase the size of the matrix A from $20^3 = 8'000$ to $100^3 = 1'000'000$. Remember that the corresponding Lyapunov equation would even have squared size. We will solve the Lyapunov equation up to a residual norm of 10^{-6}. For this example optimal shift parameters can be computed [47]. The number of shifts are computed according to a tolerance tol_w which refers to the convergence speed of the ADI method. Here we choose $\text{tol}_w = 10^{-1}$, $\text{tol}_w = 10^{-2}$ and $\text{tol}_w = 10^{-4}$ as tolerances. The number of shifts can be seen in the second column of Table 8.

dimension N	# shifts	cpu=1	cpu=2	cpu=4	cpu=8
20	3	3.5	3.2	3.2	8.9
	4	3.8	3.4	3.4	10.2
	8	4.7	4.1	4.1	11.8
40	3	144.9	87.5	77.2	124.0
	5	137.0	79.4	66.3	118.6
	9	156.6	88.0	73.5	131.4
60	4	1'564.4	704.2	464.4	983.5
	6	1'717.4	735.4	504.2	1'064.3
	10	1'879.9	794.8	622.8	1'160.1
80	4	10'562.7	4'121.1	2'585.0	6'448.1
	6	10'475.9	4'032.2	2'702.0	6'432.6
100	4	43'255.3	15'363.7	9'767.2	24'577.4

Table 8: Computation time LR-CG in [s] using CFADI preconditioning with a multithreaded version of PARDISO.

The values are always ordered from $\text{tol}_w = 10^{-1}$ down to $\text{tol}_w = 10^{-4}$ (cf. also Table 5). For $N \geqslant 80$ we skipped $\text{tol}_w = 10^{-4}$ and for $N \geqslant 100$ we skipped $\text{tol}_w = 10^{-2}$ additionally for reasons of memory consumption.

Beside the computation time in Table 8 we point out that the number of LR-CG steps only depends on the size of tol_w. Numerically it is advantageous to have a larger value of tol_w and to use more LR-CG steps since this significantly saves memory and occasionally is even the fastest version as can be seen from Table 8.

Using the multithreaded parallel solver PARDISO we observe a significant speedup which is close to linear for larger N. It can also be seen that using 4 threads or 8 threads leads to an optimal performance on our machine. We observed that for maximum possible number of 16 threads the amount of computational time increased drastically. We blame this issue to problems of the dense linear algebra kernels with the multicore architecture. In a multicore processor the processes have to share the cache if more than one thread is assigned to a socket. We believe that this might be an explanation for the numerical observations. Although the multithreaded parallel direct solver PARDISO improves the numerical solution of the LR-CG method with CFADI preconditioning, for larger sizes N the multilevel ILU is still superior although not yet being parallelized. This can be concluded from the comparison in Figure 13 for $\text{tol}_w = 10^{-1}$ and $\text{tol}_w = 10^{-2}$.

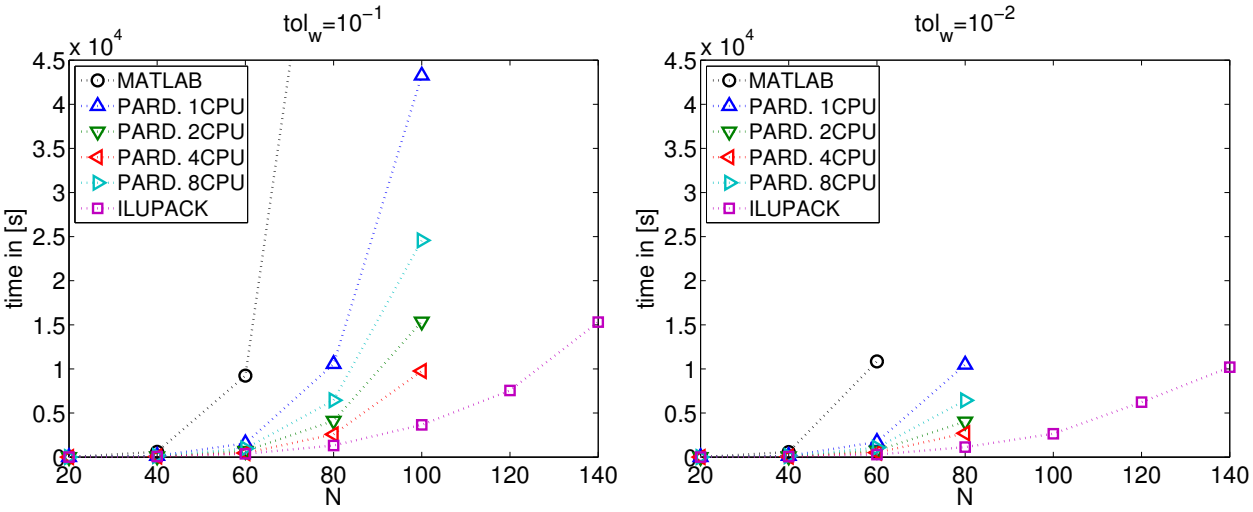


Figure 13: Computation time LR-CG in [s] versus problem size N for various inner solvers of shifted linear systems within CFADI preconditioning.

5 Conclusions

In this article we have demonstrated the benefits of low-rank Krylov subspace methods. When computing the approximate solution of generalized, projected Lyapunov equations, these novel low-rank Krylov subspace comprise the benefits of Krylov subspace methods and the low-rank Cholesky factor representation similar to LRCF-ADI methods. While the superiority of low-rank Krylov subspace methods is not always confirmed in the numerical experiments, their high potential has been illustrated. We have also shown that techniques of early compressing the rank to the desired accuracy is beneficial for low-rank Krylov subspace methods. The results have demonstrated the applicability in model order reduction techniques, in particular for those problems arising from circuit simulation. We have further outlined the wide range of their usage for other problems such as parabolic partial differential equations. We believe that this numerical case study helps understanding when and how low-rank Krylov subspace methods can be used as a technique for model order reduction.

The work reported in this paper was supported by the German Federal Ministry of Education and Research (BMBF), grant no. 03BOPAE4. Responsibility for the contents of this publication rests with the authors.

References

[1] P. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM J. Matrix Anal. Appl.*, 17(4):886–905, 1996.

[2] R. Bartels and G. Stewart. Solution of the matrix equation $AX + XB = C$. *Comm. ACM*, 15(9):820–826, 1972.

[3] P. Benner. Advances in balancing-related model reduction for circuit simulation. In J. R. und L.R.J. Costa, editor, *Scientific Computing in Electrical Engineering SCEE 2008*, volume 14 of *Mathematics in Industry*, pages 469–482, Berlin/Heidelberg, 2010. Springer.

[4] P. Benner, J.-R. Li, and T. Penzl. Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems. *Numerical Linear Algebra with Applications*, 15(9):755–777, 2008.

[5] M. Bollhöfer and Y. Saad. Multilevel preconditioners constructed from inverse–based ILUs. *SIAM J. Sci. Comput.*, 27(5):1627–1650, 2006.

[6] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *The Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, volume 14 of *Classics in Applied Mathematics*. SIAM Publications, 1996.

[7] T. Damm. Direct methods and ADI preconditioned Krylov subspace methods for generalized Lyapunov equations. *Numer. Linear Algebra Appl.*, 15(9):853–871, 2008.

[8] Z. Drmač and Z. Bujanović. On the failure of rank-revealing QR factorization software – a case study. *ACM Trans. Math. Softw.*, 35(2):12:1–12:28, 2008.

[9] A. K. Eppler and M. Bollhöfer. An alternative way of solving large Lyapunov equations. *Proc. Appl. Math. Mech.*, 10(1):547–548, 2010.

[10] A. K. Eppler and M. Bollhöfer. Structure-preserving GMRES methods for solving large Lyapunov equations. In M. Günther, A. Bartel, M. Brunk, S. Schoeps, and M. Striebel, editors, *Progress in Industrial Mathematics at ECMI 2010*, volume 17 of *Mathematics in Industry*, pages 131–136. Springer, 2012.

[11] R. Freund and F. Jarre. A QMR–based interior–point algorithm for solving linear programs. *Mathematical Programming, Series B*, 76(1):183–210, 1997.

[12] R. Freund and N. Nachtigal. QMR: A quasi-minimal residual method for non-hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[13] R. W. Freund. SPRIM: Structure-preserving reduced-order interconnect macromodeling. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 80–87. IEEE Computer Society Press, 2004.

[14] G. H. Golub and C. F. Van Loan. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)(3rd Edition)*. The Johns Hopkins University Press, 3rd edition, October 1996.

[15] W. Hackbusch. *Hierarchische Matrizen*. Springer Berlin/Heidelberg, 2009.

[16] M. Hinze and S. Volkwein. Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: Error estimates and suboptimal control. In *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lecture Notes in Computational Sience and Engineering*, pages Chapter 10 (pages 261–306). Springer-Verlag, Berlin/Heidelberg, Germany, 2005.

[17] M. Hochbruck and G. Starke. Preconditioned krylov subspace methods for lyapunov matrix equations. *SIAM J. Matrix Anal. Appl.*, 16(1):156–171, 1995.

[18] R. Ionuţiu, J. Rommes, and W. H. A. Schilders. SparseRC: Sparsity preserving model reduction for RC circuits with many terminals. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(12):1828–1841, 2011.

[19] I. Jaimoukha and E. Kasenally. Krylov subspace methods for solving large Lyapunov equations. *SIAM J. Numer. Anal.*, 31(1):227–251, 1994.

[20] K. Jbilou. Block Krylov subspace methods for large continuous-time algebraic Riccati equations. *Numer. Algorithms*, 34:339–353, 2003.

[21] K. Jbilou. An Arnoldi based algorithm for large algebraic Riccati equations. *Appl. Math. Lett.*, 19(5):437–444, 2006.

[22] K. Jbilou. ADI preconditioned Krylov methods for large Lyapunov matrix equations. *Linear Algebra Appl.*, 432(10):2473–2485, 2010.

[23] K. Jbilou and A. Riquet. Projection methods for large Lyapunov matrix equations. *Linear Algebra Appl.*, 415(2–3):344–358, 2006.

[24] D. Kressner, M. Plešinger, and C. Tobler. A preconditioned low-rank CG method for parameter-dependent Lyapunov matrix equations. Technical report, EPFL, April 2012.

[25] D. Kressner and C. Tobler. Krylov subspace methods for linear systems with tensor product structure. *SIAM J. Matrix Anal. Appl.*, 31(4):1688–1714, 2010.

[26] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for parabolic systems. *Numer. Math.*, 90:117–148, 2001.

[27] N. Levenberg and L. Reichel. A generalized ADI iterative method. *Numer. Math.*, 66:215–233, 1993.

[28] J.-R. Li and J. White. Low rank solution of lyapunov equations. *SIAM J. Matrix Anal. Appl.*, 24(1):260–280, 2002.

[29] C. C. K. Mikkelsen. *Numerical methods for large Lyapunov equations*. PhD thesis, Purdue University, 2009.

[30] A. Odabasioglu, M. Celik, and L. T. Pileggi. PRIMA: passive reduced-order interconnect macromodeling algorithm. *IEEE Trans. Circuits Syst.*, 17(8):645–654, 1998.

[31] T. Penzl. Lyapack — a MATLAB toolbox for large Lyapunov and Riccati equations, model reduction problems, and linear-quadratic optimal control problems. release 1.8 available at http://www.tu-chemnitz.de/mathematik/industrie_technik/downloads/lyapack-1.8.tar.gz.

[32] T. Penzl. A cyclic low-rank smith method for large sparse Lyapunov equations. *SIAM J. Sci. Comput.*, 21(4):1401–1418, 2000.

[33] T. Reis and T. Stykel. PABTEC: Passivity-preserving balanced truncation for electrical circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(9):1354–1367, 2010.

[34] T. Reis and T. Stykel. Positive real and bounded real balancing for model reduction of descriptor systems. *Internat. J. Control*, 83(1):74–88, 2010.

[35] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, 1993.

[36] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM Publications, second edition, 2003.

[37] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, 1986.

[38] J. Sabino. *Solution of Large-Scale Lyapunov Equations via the Block Modified Smith Method*. PhD thesis, Rice University, Houston, Texas, 2006.

[39] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. *Journal of Future Generation Computer Systems*, 20(3):475–487, 2004.

[40] O. Schenk and K. Gärtner. On fast factorization pivoting methods for symmetric indefinite systems. *Electr. Trans. Num. Anal.*, 23(1):158–179, 2006.

[41] V. Simoncini. A new iterative method for solving large-scale lyapunov matrix equations. *SIAM J. Sci. Comput.*, 29(3):1268–1288, 2007.

[42] G. Starke. Optimal alternating direction implicit parameters for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.*, 28(5):1432–1445, 1991.

[43] G. Starke. Fejér-Walsh points for rational functions and their use in the ADI iterative method. *J. Comput. Appl. Math.*, 46:129–141, 1993.

[44] T. Stykel. Low-rank iterative methods for projected generalized Lyapunov equations. *Electron. Trans. Numer. Anal.*, 30:187–202, 2008.

[45] T. Stykel and T. Reis. Passivity-preserving balanced truncation model reduction of circuit equations. In J. Roos and L. Costa, editors, *Scientific Computing in Electrical Engineering SCEE 2008*, volume 14 of *Mathematics in Industry*, pages 483–490. Springer-Verlag, Berlin/Heidelberg, 2010.

[46] H. A. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13(2):631–644, 1992.

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