Deutsche Forschungsgemeinschaft

Priority Program 1253

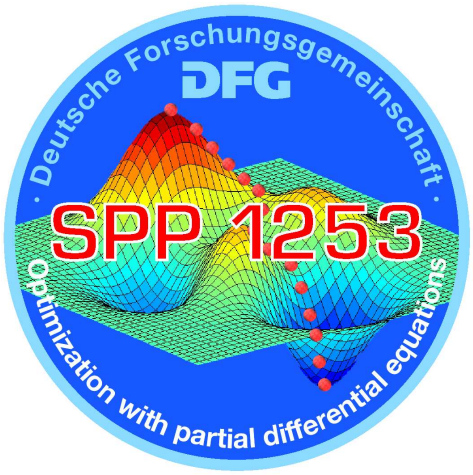
Optimization with Partial Differential Equations

Peter Benner and Jens Saak

A Galerkin-Newton-ADI Method for Solving Large-Scale Algebraic Riccati Equations

January 2010

Preprint-Number SPP1253-090



http://www.am.uni-erlangen.de/home/spp1253

A Galerkin-Newton-ADI Method for Solving Large-Scale Algebraic Riccati Equations*

PETER BENNER AND JENS SAAK†

Abstract. The alternating directions implicit (ADI) iteration has been proven to be a highly efficient method for solving stable large scale Lyapunov equations when applied in order to compute low rank factors of the solution. Employing Newton-ADI or Newton-Kleinman-ADI methods for solving algebraic Riccati equations (AREs), one has to solve a stable large scale Lyapunov equation in every Newton step. It has been shown, that the *sparse plus low rank* structure of the Lyapunov equation in the Newton step can easily be incorporated in the low rank ADI iteration. Still, the ADI iterations convergence speed is strongly depending on certain shift parameters. In this paper we will discuss a hybrid Galerkin-ADI approach that can drastically accelerate the ADI iteration when good shifts are unknown or hard to compute. The same ideas can be applied to accelerate the inexact Newton iteration resulting from the approximative/iterative solution of the Lyapunov equations.

Key words. Lyapunov equation, Riccati equation, Newton's method, ADI, Galerkin projection

AMS subject classifications. 65B99, 65F50, 15A24, 49M15, 65H10

1. Introduction.

The numerical treatment of large scale sparse algebraic matrix equations

$$0 = \mathfrak{R}(X) := CC^T + XA + A^TX - XBB^TX \tag{ARE}$$

and

$$FX + XF^T = -GG^T \tag{LYAP}$$

is one of the key ingredients of many linear-quadratic optimal control problems for parabolic PDEs, as well as the balancing based model order reduction of large linear systems [1, 3, 6, 11, 29, 33]. In those applications, the coefficient matrices $A, F \in \mathbb{R}^{n\times n}$ are normally sparse (or data sparse) such that multiplications and linear system solves with A and F are comparably cheap. The matrices C, B and G are rectangular, i.e., $C \in \mathbb{R}^{n\times r_C}$, $B \in \mathbb{R}^{n\times r_B}$, $G \in \mathbb{R}^{n\times r_G}$ with r_C, r_G, $R_B \ll n$ (usually).

The basic observation on which all methods for solving such kinds of matrix equations are based, is that often the numerical rank of the solution is very small compared to its actual dimension (see [37, 2, 15]) and therefore it allows for a good approximation via low rank solution factors. In this paper we concentrate on the low rank solution of those matrix equations based on alternating directions implicit (ADI) related methods [4, 31, 35]. Other methods for solving these equations are based on Krylov subspace projections [18, 21, 23, 39, 42] or the matrix sign function [8].

In Section 3.1 we start the discussion by an introduction of the basic ADI iteration and its application to Lyapunov equations. This will be followed by a short review of the inexact Newton-Kleinman type iteration for the algebraic Riccati equation (ARE) in Section 4. Section 5 presents acceleration techniques for the low rank ADI iteration, that can drastically reduce the number of iterations steps required in the ADI process. The acceleration technique is not restricted to the Lyapunov operator, but

*Supported by German Science Foundation (DFG) within priority program SPP1253 under project grants BE2174/8-1,2.

†Faculty of Mathematics, Chemnitz University of Technology, D-09107 Chemnitz, Germany, {benner, jens.saak}@mathematik.tu-chemnitz.de

Algorithm 1 Low Rank Cholesky Factor ADI Iteration (LRCF-ADI)

Input: F,G defining $FX + XF^T = -GG^T$ and shift parameters $\{p_1, \ldots, p_{i_{max}}\}$
Output: $Z = Z_{i_{max}} \in \mathbb{C}^{n \times t_{i_{max}}}$, such that $ZZ^H \approx X$
1: $V_1 = \sqrt{-2\operatorname{Re}(p_1)}(F + p_1 I)^{-1} G$
2: $Z_1 = V_1$
3: **for** $i = 2, 3, \ldots, i_{max}$ **do**
4: $\quad V_i = \sqrt{\operatorname{Re}(p_i)/\operatorname{Re}(p_{i-1})}(V_{i-1} - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1} V_{i-1})$
5: $\quad Z_i = [Z_{i-1} \;\; V_i]$
6: **end for**

can as well be applied in the case of the ARE. We will therefore discuss two types of accelerations for the Newton Kleinman ADI iteration, comparing the cases where the acceleration is applied in the inner (ADI) or outer (Newton) loop, respectively. The numerical examples in Section 6 demonstrate the benefits when applying these kinds of techniques in practice.

In the following I always denotes the identity matrix of appropriate size and $\otimes$ represents the Kronecker (tensor) product.

2. LRCF-ADI for Large Scale Lyapunov Equations.

2.1. Classic ADI and Lyapunov Equations. Originally the ADI iteration has been developed to solve finite difference discretizations of Poisson's equation [34]:

$$\begin{aligned} -\Delta u &= f && \text{in } \Omega \subset \mathbb{R}^d, \\ u &= 0 && \text{on } \partial\Omega. \end{aligned}$$

For $d = 1$ using centered differences on an equidistant mesh with nodes $x_i \in \Omega$ and mesh width $h \in \mathbb{R}$ this is well known to form the linear system of equations $A_{1D}u = f$, where $u_i = u(x_i)$, $f_i = f(x_i)$, $i = 1, \ldots, n$, the boundary conditions are reflected by $u_0 = u_{n+1} = 0$ and we have

$$A_{1D} = \frac{1}{h}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}.$$

In 2D, when applying the 5-point differences star, it is equally well known that the resulting linear system of equations looks like $A_{2D}u = f$, where

$$A_{2D} = \frac{1}{h^2}\begin{bmatrix} K & -I & & & \\ -I & K & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & K & -I \\ & & & -I & K \end{bmatrix} \quad \text{and} \quad K = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix}.$$

It is an easy exercise to proof that then in fact we have

$$A_{2D} = \underbrace{(A_{1D} \otimes I)}_{=:H} + \underbrace{(I \otimes A_{1D})}_{=:V}. \tag{2.1}$$

This lead to the idea of an iteration capable of exploiting the tridiagonal structure of A_{1D}, which finally gave rise to the classical two step ADI iteration described by

$$(H + p_i I)\, u_{i+\frac{1}{2}} = (p_i I - V)\, u_i + f \tag{2.2}$$

$$(V + p_i I)\, u_{i+1} = (p_i I - H)\, u_{i+\frac{1}{2}} + f \tag{2.3}$$

for certain shift parameters $p_i \in \mathbb{C}$, where the optimal parameters solve

$$\min_{\{p_1,\dots,p_\ell\}\subset\mathbb{C}} \quad \max_{\lambda\in\Lambda(H),\mu\in\Lambda(V)} \quad \left|\prod_{k=0}^{\ell} \frac{(p_k-\lambda)(p_k-\mu)}{(p_k+\lambda)(p_k+\mu)}\right|. \tag{2.4}$$

This iteration can easily be generalized to any linear system with $A = H + V$, where H, V commute.

Wachspress [44, 45] first observed that the vectorization

$$[(I \otimes F) + (F \otimes I)]\,\mathrm{vec}X = \mathrm{vec}C, \tag{2.5}$$

of the Lyapunov equation (LYAP) yields exactly the same structure as (2.1) and thus the Lyapunov equation is an ADI model problem. This observation lead to the ADI iteration for the Lyapunov equation:

$$(F + p_i I)\, X_{i+\frac{1}{2}} = C - \left(F^T - p_i I\right) X_i, \tag{2.6}$$

$$(F + p_i I)\, X_{i+1} = C^T - \left(F^T - p_i I\right) X_{i+\frac{1}{2}}. \tag{2.7}$$

Additionally, Wachspress [45, 46] provides a way to compute the optimal (in terms of the global convergence rate) shift parameters such that the ℓ-th iterate meets a prescribed relative error bound, provided the spectrum of F is real. For complex spectra with moderately sized imaginary parts of the eigenvalues, an asymptotically optimal choice is given. A fast heuristic choice of the parameters was introduced by Penzl [36], and Sabino [41] treats a potential theory based selection algorithm. The authors contribution in [7] combines the optimality of the Wachspress parameters and the fast computability of Penzl's parameters.

2.2. The Basic Idea of Low Rank ADI. The key observation [35, 31] towards a low rank version of this iteration is, that after rewriting it into a one step iteration

$$\begin{aligned} Y_j = &-2p_j(F+p_jI)^{-1}GG^T(F+p_jI)^{-T} \\ &+ (F+p_jI)^{-1}(F-p_jI)Y_{j-1}(F-p_jI)^T(F+p_jI)^{-T}, \end{aligned} \tag{2.8}$$

we find that (2.8) is symmetric. Now assuming $Y_j = Z_j Z_j^T$ and $Y_0 = 0$ we can write the iteration in terms of the factors Z_j as

$$\begin{aligned} Z_1 &= \sqrt{-2p_1}(F+p_1I)^{-1}G, \\ Z_j &= \left[\sqrt{-2p_1}(F+p_jI)^{-1}G,\ (F+p_jI)^{-1}(F-p_jI)Z_{j-1}\right]. \end{aligned}$$

Hence, we can write the ADI iteration such that it forms the factors by successively adding a fixed number of columns in every step. In the current formulation, however, all columns have to be processed in every step, which makes the iteration increasingly expensive. Now let $J \in \mathbb{N}$ be the number of shift parameters we have at hand. Then

Algorithm 2 Newton's Method for Algebraic Riccati Equations – Kleinman Iteration

Input: A,B,C as in (ARE) and an initial guess $K^{(0)}$ for the feedback.
Output: X^{k_0} solving (ARE) and the optimal state feedback K^{k_0} (or approximations when stopped before convergence).

1: **for** $k = 1, 2, \ldots, k_0$ **do**
2: Determine the solution $X^{(k)}$ of $(A - BK^{(k-1)^T})^T X^{(k)} + X^{(k)}(A - BK^{(k-1)^T}) = -CC^T - K^{(k-1)}K^{(k-1)^T}$.
3: $K^{(k)} = X^{(k)}B$.
4: **end for**

defining the matrices $T_j := (F - p_j I)$ and inverse matrices $S_j := (F + p_j I)^{-1}$ following [31] we can express the J-th iterate as

$$Z_J = \left[S_J\sqrt{-2p_J}G,\ S_J(T_JS_{J-1})\sqrt{-2p_{J-1}}G,\ \ldots,\ S_JT_J\cdots S_2(T_2S_1)\sqrt{-2p_1}G \right].$$

Now observing that the S_j and T_j commute, we can rearrange these matrices. We note that every block of the dimension of G essentially contains its left neighbor, i.e., predecessor in the iteration. Thus we find that we can rewrite the factor in the form

$$Z_J = [z_J,\ P_{J-1}z_J,\ P_{J-2}(P_{J-1}z_J),\ \ldots,\ P_1(P_2\cdots P_{J-1}z_J)], \tag{2.9}$$

and only need to apply a *step operator*

$$\begin{aligned} P_i &:= \frac{\sqrt{-2\operatorname{Re}(p_i)}}{\sqrt{-2\operatorname{Re}(p_{i+1})}}(F + p_i I)^{-1}(F - p_{i-1}I) \\ &= \frac{\sqrt{-2\operatorname{Re}(p_i)}}{\sqrt{-2\operatorname{Re}(p_{i+1})}}\left[I - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1}\right] \end{aligned} \tag{2.10}$$

to compute the new columns in every step. Note that we have implicitly applied the shifts in reverse order here. That does not matter on the other hand, since there is no natural ordering of the shifts anyway.

Especially note that only the new columns need to be processed after rearrangement. In summary, this gives the presentation in Algorithm 1. Note that Algorithm 1 can easily be adapted to generalized Lyapunov equations

$$FXE^T + EXF^T = -GG^T \tag{2.11}$$

by using

$$P_i = \frac{\sqrt{-2\operatorname{Re}(p_i)}}{\sqrt{-2\operatorname{Re}(p_{i+1})}}\left[I - (p_i + \overline{p_{i-1}})(F + p_i E)^{-1}E\right]$$

in Step 4 [3, 40].

3. Low Rank Newton ADI .

3.1. Newton-Kleinman Iteration for AREs. One classical approach to solve the algebraic Riccati equation is to tackle its nonlinearity with a Newton type method. The basic Newton step then is

$$\mathfrak{R}'|_X(N_\ell) = -\mathfrak{R}(X_\ell), \qquad X_{\ell+1} = X_\ell + N_\ell. \tag{3.1}$$

Taking a closer look at the Frechét derivative of the Riccati operator $\mathfrak{R}$ at X we observe that this is the Lyapunov operator

$$\mathfrak{R}'|_X : \quad N \mapsto (A - BB^T X)^T N + N(A - BB^T X). \tag{3.2}$$

The representation of Newton's method proposed by Kleinman [25] (see Algorithm 2) is mathematically equivalent to the basic Newton iteration and provides advantages in the context of low rank solutions. A detailed discussion of the advantages of Kleinman's formulation can be found in [4]. In the following, we want to apply low rank solution techniques to the Lyapunov equation arising in every Newton step. These techniques solve the Lyapunov equation to a certain tolerance. In that sense we need to interprete the global approach as an inexact Newtons method. Here the Kleinman formulation provides additional advantages, see [13]. In the following subsection we show how the LRCF-ADI can be employed to formulate a factored Newton ADI iteration for (ARE).

3.2. Combining Newton-Kleinman and LRCF-ADI . We will now sketch how the special structure of the closed loop operators in the Lyapunov equations for every Newton step fit into the LRCF-ADI framework of the previous sections. We can then use the low rank framework for the inner ADI iteration to derive the low rank Cholesky factor Newton method (LRCF-NM) for the ARE [4, 36] summarized in Algorithm 3.

DEFINITION 3.1 (splr). *A matrix $F \in \mathbb{R}^{n\times n}$ is called* sparse plus low rank *or simply* splr *if we can find matrices $A \in \mathbb{R}^{n\times n}$ and $U, V \in \mathbb{R}^{n\times p}$ such that*

$$F = A + UV^T.$$

Let $F = A - BK^{(k-1)^T}$ be the current closed loop operator in both the Newton and Newton-Kleinman iteration. Obviously, F is splr. Now recalling which operations are needed in the LRCF-ADI we see that these can be computed in low rank fashion. For the shift parameter computation we need to multiply and solve with F and in the ADI step we need to solve a shifted system with F. First we note, that with F also $F + pI$ for a scalar p is splr, since we can simply replace A by $\tilde{A} = A + pI$ to match the definition. For solving the linear systems with the splr matrices we can now apply the Sherman-Morrison-Woodburry formula[1] (e.g. [14])

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1}U)^{-1}V^T A^{-1}. \tag{3.3}$$

Note that when applied correctly this boils down to two linear system solves with A, since A^{-1} can be factored out to the left or right and $V^T A^{-1}$ and $A^{-1}U$ are thin matrices which can be stored in $\mathcal{O}(n)$ memory requirement and thus either one can be precomputed if $p \ll n$, depending on whether A^{-1} is factored out to the left ($V^T A^{-1}$) or right ($A^{-1}U$).

4. Accelerated ADI-based Solution of Lyapunov Equations. The most criticized property of the ADI iteration for solving Lyapunov equations is its demand for a set of good shift parameters to ensure fast convergence. Although we have investigated several parameter computation techniques [7] that are cheaply computable,

[1] This formula is often referred to as *matrix inversion lemma* in the engineering literature, as well.

Algorithm 3 Low Rank Cholesky Factor Newton Method (LRCF-NM)

Input: A, B, C, $K^{(0)}$ for which $A - BK^{(0)^T}$ is stable (e.g., $K^{(0)} = 0$ if A is stable)
Output: $Z = Z^{(k_{max})}$, such that ZZ^H approximates the solution X of (ARE).

1: **for** $k = 1, 2, \ldots, k_{max}$ **do**
2: Determine (sub)optimal ADI shift parameters $p_1^{(k)}, p_2^{(k)}, \ldots$ with respect to the matrix $F^{(k)} = A^T - K^{(k-1)}B^T$ (e.g., [35, Algorithm 1]).
3: $G^{(k)} = \begin{bmatrix} C & K^{(k-1)} \end{bmatrix}$.
4: Compute matrix $Z^{(k)}$ by Algorithm 1, such that the product $Z^{(k)}Z^{(k)^H}$ approximates the solution of $F^{(k)}X^{(k)} + X^{(k)}F^{(k)^T} = -G^{(k)}G^{(k)^T}$.
5: $K^{(k)} = Z^{(k)}(Z^{(k)^H}B)$.
6: **end for**

most of these are suboptimal in many cases and thus fast convergence can indeed not be guaranteed. Additionally, if the convergence is slow, the LRCFs may grow without adding essential information in subsequent iteration steps.

Krylov subspace methods for solving large Lyapunov equations are based on the equally named paper [21] by Jaimoukha and Kasenally. There, the basic idea is to consider the Schur decomposition $X = U\Sigma U^T$ of the solution X, where $U \in \mathbb{R}^{n\times n}$ is an orthogonal matrix and $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$ is diagonal due to the symmetry of X. Further the eigenvalues are considered to be ordered such that $|\sigma_1| \geq |\sigma_2| \geq \cdots \geq |\sigma_n|$. Therefore it coincides with the SVD of X. The best rank-m Frobenius-norm approximation is thus given by

$$X_m := U \begin{bmatrix} \Sigma_m & 0 \\ 0 & 0 \end{bmatrix} U^T = U_m \Sigma_m U_m^T. \tag{4.1}$$

Here, $\Sigma_m = \operatorname{diag}(\sigma_1, \ldots, \sigma_m)$, and $U_m \in \mathbb{R}^{n\times m}$ consists of the first m columns of U. The basic idea now is to compute X_m via the solution of a projected version of the Lyapunov equation (LYAP)

$$(U_m^T F U_m)Y_m + Y_m(U_m^T F^T U_m) = -U_m^T G G^T U_m, \tag{4.2}$$

on $\mathcal{U}_m$, the column span of U_m, and define X_m as

$$X_m = U_m Y_m U_m^T. \tag{4.3}$$

The basic projection idea described in the above has already been considered by Saad [39] 4 years earlier for general subspaces $\mathcal{U}_m$ to project onto. The main concern in [21] and related Krylov subspace methods (e.g., [20, 42, 24, 23, 19]) is then to find or compute an orthogonal basis of a good (Krylov) subspace approximating U_m. The most promising among these methods seems to be the recently proposed Krylov-plus-inverse-Krylov (KPIK) Method by Simoncini [42], which uses a rational Krylov subspace for the projection. A convergence analysis for the projection based solvers is carried out in [43].

We employ the same projection idea here, but replace the critical Krylov subspace by the column span of the current ADI iterate Z_i in the i-th iteration step. This idea is motivated by the observation [30] that the column span of the ADI solution is in fact a certain rational Krylov subspace.

The main drawback of our approach is the necessity for an orthogonal basis of the subspace spanned by Z_i. We compute a QR decomposition

$$Z_i =: U_i R_i \tag{4.4}$$

and use U_i for the projection in (4.2):

$$(U_i^T F U_i) Y_i + Y_i (U_i^T F^T U_i) = -U_i^T B B^T U_i. \tag{4.5}$$

Clearly, the computation of U_i will in general be much more expensive than the actual iteration step. Thus we need to save many steps to make it worth the effort.

Computing a Cholesky factorization of $Y_i = \tilde{R}_i^T \tilde{R}_i$, we define the optimization $\tilde{Z}_i$ of Z_i on the current subspace via

$$\tilde{Z}_i := U_i \tilde{R}_i. \tag{4.6}$$

We emphasize, that Hammarling's method [17], as well as the sign function method [8] can directly compute the Cholesky factor $\tilde{R}_i$. Note that dropping the original R_i completely is no problem at all, since our primary concern is the subspace information contained in the orthogonal columns of U_i only. Note further that in cases where Z_i does not have full rank, the column space of a standard QR-decomposition does not necessarily coincide with the column space of Z_i (see [14, Section 5.4]). In those cases it is crucial to use QR with column pivoting to ensure the equality of the column spaces. Finally note that the procedure is not restricted to the standard Lyapunov equation case and when applying to generalized equations we need to solve the generalized projected equation

$$(U_i^T F U_i) Y_i (U_i^T E^T U_i) + (U_i^T E U_i) Y_i (U_i^T F^T U_i) = -U_i^T B B^T U_i, \tag{4.7}$$

for a Cholesky factor of Y_i.

A similar method has also been proposed for the more general Sylvester equation case in [5]. We have to keep in mind that the Krylov projection based methods do not work for all Lyapunov equations, where the basic ADI iteration is applicable. They are only valid for those equations where the projected equation (e.g. (4.2)) remains stable, i.e., the matrix $U_m^T F U_m$ is asymptotically stable. It can be shown using Bendixson's theorem [32], that $F + F^T < 0$ is sufficient to get $\mathrm{Re}\,(\lambda) < 0$ for all $\lambda \in \Lambda(U^T F U)$ if $U \in \mathbb{R}^{n \times r}$. This is clearly a restriction, but it holds for any dissipative operator. Dissipativity of the operator is a rather basic assumption, that has to be made, e.g., in many control problems for parabolic PDEs anyway. The integration of the above acceleration into LRCF-ADI gives the Galerkin projection accelerated LRCF-ADI (LRCF-ADI-GP) presented in Algorithm 4.

The condition $F + F^T < 0$ is only sufficient by construction, but it can be crucial. For example let $F = \mathrm{diag}(A_1, A_2, A_3, A_4, -\mathrm{diag}(1:400))$ [36, Section C.3] with

$$A_1 = \begin{bmatrix} -0.01 & -200 \\ 200 & -0.001 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} -0.2 & -300 \\ 300 & -0.1 \end{bmatrix},$$
$$A_3 = \begin{bmatrix} -0.02 & -500 \\ 500 & 0 \end{bmatrix}, \qquad A_4 = \begin{bmatrix} -0.01 & 520 \\ 520 & -0.01 \end{bmatrix},$$

then we find

$$\begin{aligned} F + F^T = \mathrm{diag}(&-0.02,\ 0.002,\ -0.4,\ -0.2,\\ &-0.04,\ 0,\ -0.02,\ -0.02,\ -2:-2:-400), \end{aligned}$$

is indefinite and the LRCF-ADI-GP stagnates for this example.

Algorithm 4 Galerkin Projection Accelerated LRCF-ADI (LRCF-ADI-GP)

Input: F,G defining $FX + XF^T = -GG^T$ and shift parameters $\{p_1, \ldots, p_{i_{max}}\}$
Output: $Z = Z_{i_{max}} \in \mathbb{C}^{n \times t_{i_{max}}}$, such that $ZZ^H \approx X$

$V_1 = \sqrt{-2\operatorname{Re}(p_1)}(F + p_1 I)^{-1} G$
$Z_1 = V_1$
for $i = 2, 3, \ldots, i_{max}$ **do**
  $V_i = \sqrt{\operatorname{Re}(p_i)/\operatorname{Re}(p_{i-1})}(V_{i-1} - (p_i + \overline{p_{i-1}})(F + p_i I)^{-1} V_{i-1})$
  $Z_i = [Z_{i-1} \;\; V_i]$
  Orthogonalize the columns of Z_i,
  e.g., $U_i =$`mgs`(Z_i) (modified Gram-Schmidt),
  or $[U_i, R_i, \Pi_i] =$`qr`$(Z_i, 0)$ (QR decomposition)
  and deflate if rank-deficiency is detected in Z_i.
  $F_i = U_i^T F U_i$, $G_i = U_i^T G$
  Solve $F_i Y_i + Y_i F_i^T = -G_i G_i^T$ exactly for R_i with $Y_i = R_i^T R_i$.
  $Z_i = U_i R_i$
  Update V_i from the last column block of Z_i according to (4.8).
end for

Restarted ADI Interpretation. Note that the last step in the loop is rather arbitrary, since we can not tell which columns are the ones "belonging" to $V_i \in \mathbb{R}^{n \times m_v}$ after computing the updated $Z_i \in \mathbb{R}^{n \times m_z}$. That means that we are more or less free to choose any appropriate number of columns in Z_i. Here we decided to take the last columns, i.e.,

$$V_i = Z(:, m_z - m_v + 1 : mz), \tag{4.8}$$

since they will in general contain linear combinations of the largest number of orthogonal columns from U_i (compare (4.6)) and thus have most subspace information saved. In this sense this should be seen as a restarted ADI iteration, since we can no longer guarantee the full structure of the factor as in (2.9), but keep as much information as possible for the restart. However compared to, e.g., `gmres`$(\bar{k})$, this technique implements a partial restart only, since we keep the full factor Z_i as the initial factor after the restart. A "fully restarted" variant could additionally compute a column pivoted QR decomposition of Z_i and only keep the leading $\bar{k}$ columns as in `gmres`$(\bar{k})$.

Krylov Subspace Interpretation. From the viewpoint of the LRCF-ADI the Galerkin projection is an acceleration technique trying to improve the quality of the iterate on the current column space. That means it can be interpreted as some kind of subspace optimization method applied to the ADI iteration. On the other hand, in her thesis Jing-Rebecca Li showed [30, Corollary 1] that the column span of the Lyapunov solution is itself a special type of rational Krylov subspace. The span of its factor is then the same Krylov subspace. Combining that knowledge with the idea of taking rational Krylov subspaces (as in KPIK [42]) for the projection in [21] and noting that the projection technique proposed here does not change the subspace but rather performs a change of bases to optimize the factor, we immediately see that Algorithm 4 can also be interpreted as a certain Krylov subspace projection method. It is thus in fact a hybrid Krylov-ADI-Iteration.

Rank Deficiency and Combination With Column Compression. In cases where Z_i is rank deficient, we need to perform column pivoting while computing the QR

decomposition in (4.4). We will now discuss how we can combine this approach with the column compression technique discussed, e.g., in [40, Section 4.4.1]. The basic idea is to replace Step 6 in Algorithm 4 by an orthogonalize-and-compress step. The naive approach would be to apply the RRQR to compress the columns first and then use the new factor to compute the orthogonal basis for the projection. Obviously this would require the relatively expensive orthogonalization twice. Since this is unacceptable especially for larger systems, we propose an alternative way, that will not avoid the two orthogonalizations, but apply one of them to a much smaller matrix.

Let $Z \in \mathbb{R}^{n\times r_c}$ and assume $\operatorname{rank}(Z,\tau) = r < r_c$ as in [40, Section 4.4.1] for a given $\tau \in \mathbb{R}$. We can now compute the "economy size" QR decomposition with column pivoting

$$Z =: U_1 R_1 \Pi_1, \tag{4.9}$$

where $U_1 \in \mathbb{R}^{n\times r_c}$, $R_1 \in \mathbb{R}^{r_c\times r_c}$ and Π_1 a permutation matrix. Note that this can be done efficiently using level 3 BLAS [38] via `xGEQP3` routines in recent LAPACK implementations. Also MATLAB uses these routines when called with second input parameter 0 or three output parameters. Now assuming that $r_c \ll n$, R_1 is much smaller than Z. The numerical rank decision can therefore be performed a lot cheaper on R_1 than on Z^T. Hence computing

$$R_1 =: U_2 R_2 \Pi_2, \tag{4.10}$$

using the RRQR, we have a cheap way to perform the rank decision. The final factorization then is

$$Z = U_1 U_2 R_2 \Pi_2 \Pi_1. \tag{4.11}$$

Note that the rank decision can also be performed based on a spectral decomposition or singular value decomposition of R_1, which leads to a similar framework as in [12]. The scaling step suggested there can also be used here to possibly improve the numerical accuracy of the QRP factorization in (4.11). Due to the small size of R_1 the higher accuracy of the rank decision can outweigh the higher cost of these approaches compared to the RRQR.

Defining U as the first r columns of the product U_1U_2 we can now proceed as in (4.5) and (4.6), resulting in a compressed and corrected new LRCF iterate $\tilde{Z}_i$ as in (4.6). Note that solving (4.5) now is even cheaper, since the subspace is smaller and so are the matrices. Also, the numerical stability of the computation for (4.5) will be better, since due to the truncation, the condition numbers of the projected matrices should have decreased.

5. Accelerating the Inexact Newton's Method. For the ARE, the straightforward way to exploit the acceleration features presented above is to use LRCF-ADI-GP instead of LRCF-ADI to perform the inner loop (Step 4) in Algorithm 3. On the other hand, the operator $\mathfrak{R}(X)$ in (ARE) has the same symmetry properties as the Lyapunov operator in equations (LYAP) and (4.2). Therefore we can apply the same ideas here and project the ARE to the column space of the current Newton iterate and solve the small projected ARE exactly to obtain the optimal solution on the current subspace. This again corresponds to projection based solvers for the ARE as discussed in [18, 21, 22].

In analogy to equations (4.4)–(4.6), we compute the QR decomposition

$$Z^{(i)} =: \hat{U}_i \hat{R}_i. \tag{5.1}$$

Algorithm 5 Low Rank Cholesky Factor Newton Galerkin Method (LRCF-NM-GP)

Input: A, B, C, $K^{(0)}$ for which $A - BK^{(0)^T}$ is stable (e.g., $K^{(0)} = 0$ if A is stable)
Output: $Z = Z_{(k_{max})}$, such that ZZ^H approximates the solution X of (ARE).

1: **for** $k = 1, 2, \ldots, k_{max}$ **do**
2: Determine (sub)optimal ADI shift parameters $p_1^{(k)}, p_2^{(k)}, \ldots$ with respect to the matrix $F^{(k)} = A^T - K^{(k-1)}B^T$ (e.g., [35, Algorithm 1]).
3: $G^{(k)} = \begin{bmatrix} C & K^{(k-1)} \end{bmatrix}$.
4: Compute matrix $Z_{(k)}$ by Algorithm 1, or Algorithm 4, such that the product $Z_{(k)}{Z_{(k)}}^H$ approximates the solution of $F^{(k)}X^{(k)} + X^{(k)}{F^{(k)}}^T = -G^{(k)}{G^{(k)}}^T$.
5: Orthogonalize the columns of $Z_{(k)}$,
e.g., $\hat{U}_k =$`mgs`$(Z_{(k)})$ (modified Gram-Schmidt),
or $[\hat{U}_k, R_k, \Pi_k] =$`qr`$(Z, 0)$(QR decomposition)
and deflate if rank-deficiency is detected in $Z_{(k)}$.
6: $A_k = \hat{U}_k^H A\hat{U}_k$, $B_k = \hat{U}_k^H B$, $C_k = \hat{U}_k^H C$
7: Find solution factor $\tilde{R}_k$ of solution $Y^{(k)}$ of $C_kC_k^T + A_k^TY^{(k)} + Y^{(k)}A_k - Y^{(k)}B_kB_k^TY^{(k)} = 0$.
8: $Z_{(k)} = \hat{U}_k^H\tilde{R}_k$.
9: $K^{(k)} = Z_{(k)}({Z_{(k)}}^H B)$.
10: **end for**

Here again we apply the rank decision and deflation technique described in (4.9)-(4.11). We then use $\hat{U}_i$ to solve the projected ARE

$$\begin{aligned} 0 = (\hat{U}_i^TC)(C^T\hat{U}_i) + Y^{(i)}(\hat{U}_i^TA\hat{U}_i) + (\hat{U}_i^TA^T\hat{U}_i)Y^{(i)} \\ - Y^{(i)}(\hat{U}_i^TB)(B^T\hat{U}_i)Y^{(i)}, \end{aligned} \tag{5.2}$$

on the current subspace and reestablish the solution factor from

$$Y^{(i)} = \hat{U}_i^T\tilde{Z}_{(i)}\tilde{Z}_{(i)}^H\hat{U}_i,$$

or from a Cholesky factorization $\tilde{R}_i\tilde{R}_i^H$ of $Y^{(i)}$ as $\tilde{Z}_{(i)} = \hat{U}^{(i)}\tilde{R}_i$. Again it is due to Hammarling [16] that the Cholesky factorization can be computed directly. Note that the test implementation used in the numerical experiments computes the solution of the projected ARE (5.2) using `care` from the MATLAB Control Systems Toolbox and factorizes the positive (semi)definite solution matrix via `cholp` from the Matrix Computation Toolbox[2]. The latter is pure MATLAB code with cascaded loops, which is far from optimal in MATLAB, i.e., we might get even better performance with a mex-file based Lapack-style implementation, or by computation of the factors directly using a Newton-Hammarling type iteration.

Convergence Towards the Stabilizing Solution. In general the stabilizing solution is not the only solution of the ARE. For the low rank Newton-Kleinman framework, [4] provides assumptions under which convergence towards the stabilizing solution can be guaranteed if the Lyapunov equations in every Newton step are solved accurately enough. An inexact Newton interpretation of the framework is presented in [13]. The authors prove monotone convergence to the stabilizing solution from above, even if

[2] `http://www.ma.man.ac.uk/~highham/mctoolbox`

the Lyapunov equations are only solved approximately, provided a certain operator valued estimate is met. The open question now is whether, or when our new approach computes the stabilizing solution, as well. The following theorem answers this question for an important case often present in the context of LQR control systems with parabolic PDE constraints.

THEOREM 5.1 (Convergence towards the stabilizing solution). *Let $A \in \mathbb{R}^{n\times n}$ with $A + A^T \leq 0$, B, C as in (ARE). If*

1. *(A, B) is stabilizable, i.e., $\operatorname{rank}[A - \lambda I, B] = n$ for all $\lambda \in \mathbb{C}$ with $\operatorname{Re}(\lambda) \leq 0$,*
2. *(C, A) is detectable, i.e., (A^T, C^T) stabilizable,*
3. *$BB^T \geq 0$ and $C^TC \geq 0$,*

then choosing $K^{(0)} \in \mathbb{R}^{n\times r_B}$ such that $A - BK^{(0)^T}$ is stable, Algorithm 5 computes the unique positive semidefinite stabilizing solution of (ARE) if it converges.

Proof. First we note that the assumptions are such that a unique stabilizing solution of (ARE) exists, e.g., by [27, Theorem 9.1.2]. Further the condition $A + A^T \leq 0$ ensures that the projected Lyapunov equations in the LRCF-ADI-GP remain solvable and allow for symmetric positive semidefinite solutions. Further we observe that with (A, B) stabilizable this holds for the projected pairs (A_k, B_k) as well, since if K stabilizes (A, B), then $\hat{U}_k^H K$ (with $\hat{U}_k$ from Step 5) stabilizes (A_k, B_k). Analogously, we realize that (C_k, A_k) remains detectable. Also we find $B_kB_k^T \geq 0$ and $C_kC_k^T \geq 0$, and thus the projected AREs (5.2) allow for unique stabilizing (on the current subspace defined by the column span of $Z_{(k)}$) symmetric positive semidefinite solutions. Thus, by construction all iterates are positive semidefinite, since they are positive semidefinite on the corresponding subspace and zero on its orthogonal complement. From the stabilizability and detectability assumptions it follows [27, Theorem 7.2.8] that the closed loop matrix $A - BB^TX$ cannot have purely imaginary eigenvalues, for any solution X of (ARE). We rewrite (5.2) in the form

$$\left(-(A - BB^TX)\right)X + X\left(-(A - BB^TX)\right)^T = CC^T + XBB^TX. \tag{5.3}$$

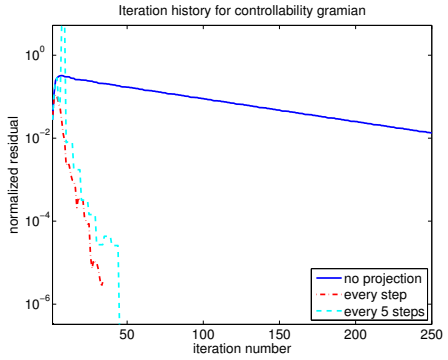
Here we now insert $X = X_\infty = \lim_{k\to\infty} Z_kZ_k^T$, the limit of the Algorithm, i.e., the computed solution, which exists by assumption. Note that the right hand side of (5.3) is positive semidefinite. Now employing [28, Section 13.1 Proposition 1] we can conclude that the number of eigenvalues of $-(A-BB^TX_\infty)$ in the open left half plane is limited from above by the number of eigenvalues of X_∞ with negative real part. Thus $A - BB^TX_\infty$ is Hurwitz and X_∞ is in fact the unique stabilizing solution. □

We verify this observation by numerical tests in Section 6.2, Table 6.4.

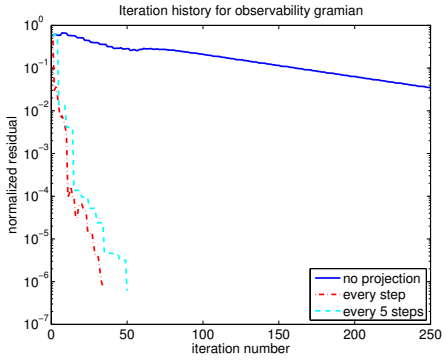
An Extended Line Search Interpretation. The basic idea of the Newton method with line search is to optimize the step length in the current search direction defined by the Newton update, i.e.,

$$X_{k+1} = X_k + \alpha_kN_k, \quad \text{where} \quad \alpha_k = \operatorname*{argmin}_{\alpha\in\mathbb{R}} \mathfrak{R}(X_k + \alpha N_k).$$

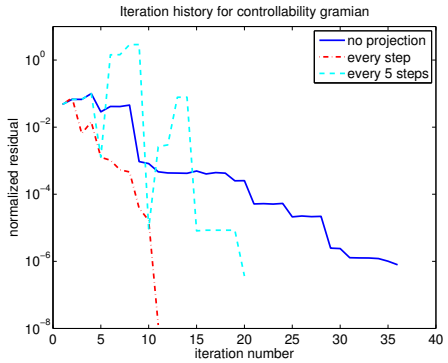
Thus one optimizes with respect to a one dimensional subspace. Now in our approach the search space is defined by the column span of the new iterate $X_k + N_k$. Computation of the Galerkin projected solution results in an optimization of the solution on the current column span. It is therefore an extension of the line search idea to a multidimensional search space.



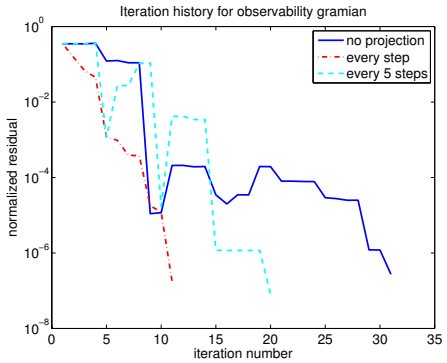
(a) Residual histories, controllability LE, bad shift parameter choice.



(b) Residual histories, observability LE, bad shift parameter choice.



(c) Residual histories, controllability LE, good shift parameter choice.



(d) Residual histories, observability LE, good shift parameter choice.

Figure 6.1: Galerkin projected solution of controllability and observability Lyapunov equations for the steel profile example (dimension 20209).

6. Numerical Experiments. We present results for both, the Lyapunov equation and Riccati equation cases. We will see that in many cases we are unable to save very much computation times for the Lyapunov equations. On the other hand, the small savings accumulate in a Newton-ADI method and the combination with the higher accuracy can even accelerate the convergence of the outer Newton method.

Note that the numerical results for the Galerkin projection acceleration in Section 6 have been acquired by an experimental MATLAB implementation using `orth` to compute the orthogonal basis of span(Z), which uses an SVD approach for the rank decision and truncation. This, although more reliable in terms of the rank approximation, will in general be more time consuming than the RRQR based approach. In the present context it is more favorable to have faster execution, thus we propose the RRQR based approach for efficient implementations. The efficient RRQR based codes and tests are part of the implementation in the upcoming C-based version of the Matrix Equation Sparse Solver (M.E.S.S. MATLAB-toolbox) called C.M.E.S.S. and corresponding timings will be given in [26].

6.1. Lyapunov Equations. We concentrate on the accelerated solution of Lyapunov equations first. The main purpose of the test computations shown with respect to this equation is to show, that the acceleration technique we presented, is especially helpful when good ADI parameters are unknown or difficult to compute. As the fist test case we consider the optimal cooling of steel profiles test example discussed in [9, 10][3]. For the test presented here we use the semi-discretization of dimension $n = 20209$. We compare computations with 2 sets of shift parameters. One of them providing good performance by itself and the other giving essentially no convergence for the ADI process. For both sets of shift parameters we compute the solutions to the controllability Lyapunov equation and observability Lyapunov equation

$$AXM^T + MXA^T = -BB^T, \qquad A^TXM + M^TXA = -C^TC,$$

corresponding to equation (2.11) with mass matrix M, stiffness matrix A, input matrix B and output matrix C. We compare the LRCF-ADI and LRCF-ADI-GP with projections in every step and in every fifth step.

shift type	LRCF-ADI	LRCF-ADI-GP (every step)	LRCF-ADI-GP (every 5^{th} step)
bad shifts	–	2075.02 s	368.40 s
good shifts	94.30 s	79.29 s	80.32 s

Table 6.1: Execution Times for LRCF-ADI and LRCF-ADI-GP (with projection in every step or every 5^{th} step respectively).

We mainly observe two things in the results shown in Figure 6.1 and Table 6.1. On the one hand, we see that in cases where we know good shifts although we use only a third to half the number of shifts we do not win too much with respect to computation times. On the other hand, if the shifts are giving bad performance of the ADI iteration, we can drastically reduce the number of required shifts and the computation time.The – in Table 6.1 means that the corresponding LRCF-ADI did not converge within 250 iteration steps.

Note that an optimal implementation in, e.g., C or Fortran will have to combine the basis computation with column compression techniques [40] and the evaluation of the stopping criteria (Frobenius norm residual via updated QR factorizations [37]). We have not done this here since it is hardly possible to achieve this efficiently in a MATLAB implementation of the algorithms. Note further that saving iteration steps in the ADI process means that we save large amounts of memory – especially in the case of multiple input and multiple output systems (i.e., $r_G > 1$) where the factors are growing by r_G columns in every iteration step.

6.2. Riccati Equations. For the Newton-Kleinman-ADI tests we restrict ourselves to the standard state space system case, i.e., the ARE as in (ARE). As one test example we use the 3D convective heat flow example introduced in [42, Example 5.2]. Here we choose the larger of the two systems discussed there, i.e., $n = 10648$

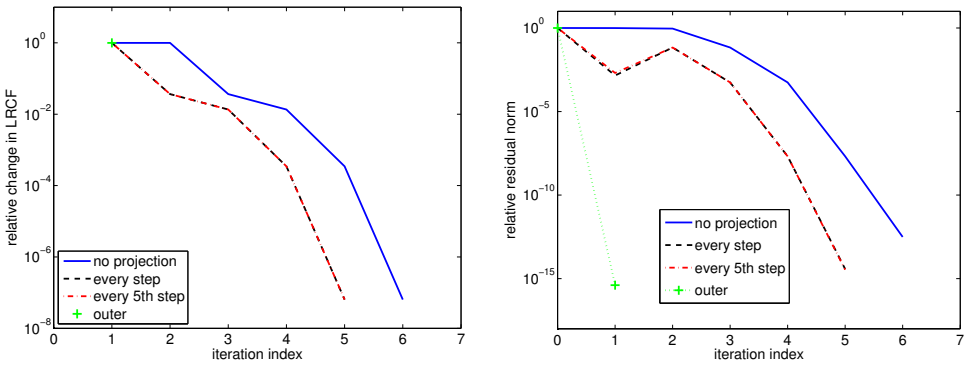
[3]The test matrices can be downloaded from `http://www.imtek.de/simulation/benchmark/?li=30&nr=38881`

and 71632 non-zero entries in A. Note that due to the convection the matrix here is non-symmetric. Together with the strong connectivity of the entries resulting from the 3D discretization, this makes it especially tough for sparse direct solvers. The matrix B is $n \times 10$ dimensional and densely populated and $C = B$. We kindly thank V. Simoncini for providing her test matrices. As second test example we use the well known LyaPack benchmark `demo_l1` – another convection diffusion example. This example resides in 2D on the unit square with 150 discretization points in each direction, i.e., 22500 degrees of freedom in the state space system.

All computations (except for Table 6.4[4]) have been carried out on a 64Bit workstation with a Core2Quad Q9400 CPU running at 2.66GHz and equipped with 4GB of RAM. The MATLAB environment has been used in release R2009b with threaded BLAS capabilities activated, i.e., in dense computations (norms, solvers for the projected equations) all four cores have been used to accelerate the computation. Column compression has been performed with a truncation tolerance equal to the square root of the machine precision. Both the inner and outer loop have been stopped with a relative residual criterion at tolerance 10^{-10}.

test	$\|\Re(ZZ^T)\|_2$	$\frac{\|\Re(ZZ^T)\|_2}{\|CC^T\|_2}$	#NM	$\sum$#ADI	runtime
no proj.	$3.49 \cdot 10^{-09}$	$3.12 \cdot 10^{-13}$	6	586	6667 s
ev. inner step	$4.18 \cdot 10^{-11}$	$3.74 \cdot 10^{-15}$	5	173	1721 s
5^{th} inner step	$3.82 \cdot 10^{-11}$	$3.42 \cdot 10^{-15}$	5	195	2279 s
ev. outer step	$4.55 \cdot 10^{-12}$	$4.07 \cdot 10^{-16}$	1	100	687 s

Table 6.2: LRCF-NM with Galerkin projection in either the inner or outer loops for the 3D example.(#NM: number of Newton steps, Σ#ADI: total number of ADI steps summed up over all Newton step.)



(a) Relative Change in Low Rank Factor Norms. (b) Normalized Residual Histories.

Figure 6.2: LRCF-NM with Galerkin projection in either the inner or outer loops for the 3D example.

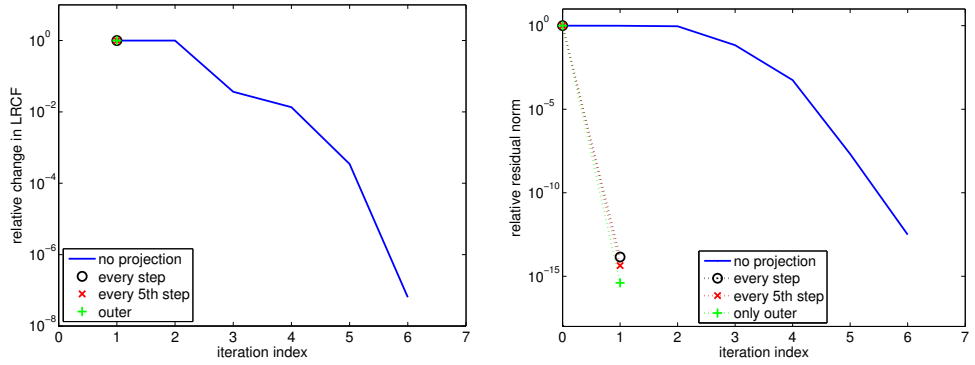
[4]Here we needed a 64GB RAM compute server to compute the `care` solution.

Simoncini's 3D Example. The first test series picks up the computations undertaken in [40, Section 8.2.2]. There we have shown that application of the LRCF-ADI-GP in the inner loop can drastically accelerate the Newton iteration for the FDM discretized equations in 2D both with and without convection for a slightly smaller model with 100 grid points per direction ($n = 10000$). Here, the first series performs the same computations for the two models above. This means that we compare the classical low rank Newton ADI iteration and the variants applying LRCF-ADI-GP instead of LRCF-ADI in the inner loop with Galerkin projections computed in every or every fifth step respectively. Additionally, these results are compared with a variant of the Newton method applying the projection in the outer loop as described in Section 5.

The results for Simoncini's 3D model are displayed in Figure 6.2 and Table 6.2. Table 6.2 compares the absolute and normalized final residuals, i.e., $\|\mathfrak{R}(ZZ^T)\|_2$ and $\frac{\|\mathfrak{R}(ZZ^T)\|_2}{\|CC^T\|_2}$, the total iteration numbers and the global runtimes.

test	$\|\mathfrak{R}(ZZ^T)\|_2$	$\frac{\|\mathfrak{R}(ZZ^T)\|_2}{\|CC^T\|_2}$	#NM	$\sum$#ADI	runtime
no proj.	$3.49 \cdot 10^{-09}$	$3.12 \cdot 10^{-13}$	6	586	6732 s
outer + inner	$1.59 \cdot 10^{-10}$	$1.42 \cdot 10^{-14}$	1	71	963 s
outer + 5^{th} inner	$4.91 \cdot 10^{-11}$	$4.39 \cdot 10^{-15}$	1	75	609 s
only outer	$4.55 \cdot 10^{-12}$	$4.07 \cdot 10^{-16}$	1	100	689 s

Table 6.3: LRCF-NM with Galerkin projection in booth loops for the 3D example.



(a) Relative Change in Low Rank Factor Norms. (b) Normalized Residual Histories.

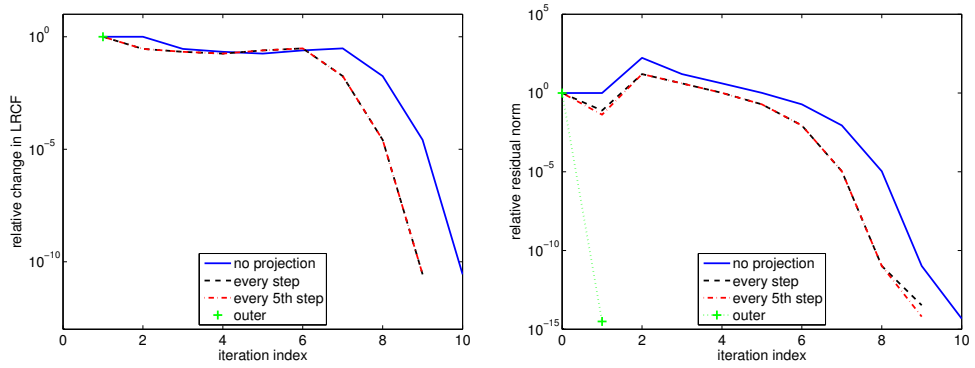
Figure 6.3: LRCF-NM with Galerkin projection in both loops for the 3D example.

In Figure 6.3 and Table 6.3 we find the results of a similar test series. In contrast to the case above, here we apply the Galerkin projection in the outer Newton loop in all cases except for the first, i.e., we combine the projections in the inner and outer loops and compare the results to the classic projection free approach.

In Table 6.4 we compare the computed solutions with the sought-after solution. As reference for the comparison we use the result X of a dense solve using `care` from the

$\|Z_0 Z_0^T - X\|_2$	$\|Z_1 Z_1^T - X\|_2$	$\|Z_5 Z_5^T - X\|_2$	$\|ZZ^T - X\|_2$
$1.56 \cdot 10^{-10}$	$7.99 \cdot 10^{-11}$	$2.56 \cdot 10^{-11}$	$6.60 \cdot 10^{-13}$

Table 6.4: Deviation of computed factorized solutions from the stabilizing solution X (computed via `care` from MATLAB's Control Systems Toolbox).



(a) Relative Change in Low Rank Factor Norms. (b) Normalized Residual Histories.

Figure 6.4: LRCF-NM with Galerkin projection in either the inner or outer loops for the 2D example.

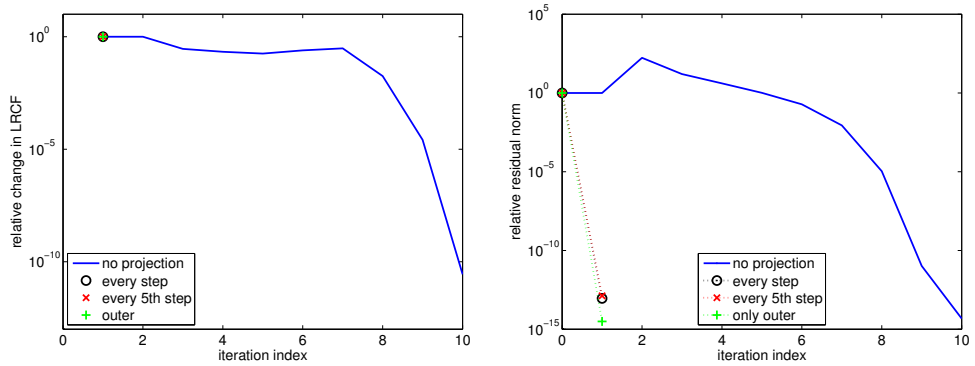
MATLAB Control Systems Toolbox. The factors in Table 6.4 are the final iterate of the LRCF-NM without projection Z_0, the solution Z where only projection in the outer loop as been applied, the result Z_1 of the computation where additionally the LRCF-ADI-GP uses projection in every step and the result Z_5 for Galerkin projection in every fifth step of the LRCF-ADI-GP. Note that the 2-norm differences can be approximated by a power iteration without forming the LRCF products due to symmetry.

2D LyaPack Demo. We have repeated the computations carried out on the 3D example above for the `demo_l1` example. The corresponding results can be found in Tables 6.5, 6.6 and Figures 6.4, 6.5. They show essentially the same behavior as above. Note, however, that here we even find acceleration factors larger than 20. In contrast to the smaller 3D example here we could not compute the deviations from the `care`-solution, since solving the 22500 dimensional ARE in dense arithmetics went even beyond the capabilities of our compute server equipped with 64GB RAM.

7. Conclusions. We have presented hybrid Galerkin-ADI based solvers for Lyapunov and Riccati equations. These can drastically reduce the number of ADI steps and Newton steps taken. In turn the number of shifted linear system solves is reduced and significant savings in computation time can be achieved. In the numerical tests the proposed techniques have lead to a reduction factor upto 10 or 20 for the execution times in MATLAB. The reduced number of ADI steps especially leads to smaller low rank factors and thus additionally helps to minimize the memory demands of the final iterates. The interpretations of our framework in the terminology of different classes of methods give rise to new ideas such as the "fully restarted LRCF-ADI" concept, leading to a whole number of future research perspectives in this context.

test	$\|\mathfrak{R}(ZZ^T)\|_2$	$\frac{\|\mathfrak{R}(ZZ^T)\|_2}{\|CC^T\|_2}$	#NM	$\sum$#ADI	runtime
no proj.	$7.46 \cdot 10^{-11}$	$4.74 \cdot 10^{-15}$	10	534	590 s
ev. inner step	$5.26 \cdot 10^{-10}$	$3.34 \cdot 10^{-14}$	9	168	296 s
5^{th} inner step	$1.00 \cdot 10^{-10}$	$6.35 \cdot 10^{-15}$	9	187	287 s
ev. outer step	$4.91 \cdot 10^{-11}$	$3.12 \cdot 10^{-15}$	1	100	47 s

Table 6.5: LRCF-NM with Galerkin projection in either the inner or outer loops for the 2D example.



(a) Relative Change in Low Rank Factor Norms. (b) Normalized Residual Histories.

Figure 6.5: LRCF-NM with Galerkin projection in both loops for the 2D example.

test	$\|\mathfrak{R}(ZZ^T)\|_2$	$\frac{\|\mathfrak{R}(ZZ^T)\|_2}{\|CC^T\|_2}$	#NM	$\sum$#ADI	runtime
no proj.	$7.46 \cdot 10^{-11}$	$4.74 \cdot 10^{-15}$	10	534	580 s
outer + inner	$1.42 \cdot 10^{-09}$	$9.02 \cdot 10^{-14}$	1	34	28 s
outer + 5^{th} inner	$2.07 \cdot 10^{-09}$	$1.32 \cdot 10^{-13}$	1	35	24 s
only outer	$4.91 \cdot 10^{-11}$	$3.12 \cdot 10^{-15}$	1	100	45 s

Table 6.6: LRCF-NM with Galerkin projection in both loops for the 2D example.

Acknowledgments. The authors whish to express their gratefulness to René Schneider and Thomas Mach for the help- and insightful discussions on the topic and numerical realization that increased the quality of this publication a lot.

REFERENCES

[1] A. Antoulas, *Approximation of Large-Scale Dynamical Systems*, SIAM Publications, Philadelphia, PA, 2005.

[2] A. Antoulas, D. Sorensen, and Y. Zhou, *On the decay rate of Hankel singular values and related issues*, Sys. Control Lett., 46 (2002), pp. 323–342.

[3] P. Benner, *Solving large-scale control problems*, IEEE Control Systems Magazine, 14 (2004), pp. 44–59.
[4] P. Benner, J.-R. Li, and T. Penzl, *Numerical solution of large Lyapunov equations, Riccati equations, and linear-quadratic control problems*, Numer. Linear Algebra Appl., 15 (2008), pp. 755–777.
[5] P. Benner, R.-C. Li, and N. Truhar, *On the ADI method for Sylvester equations*, J. Comput. Appl. Math., 233 (2009), pp. 1035–1045.
[6] P. Benner, V. Mehrmann, and D. Sorensen, eds., *Dimension Reduction of Large-Scale Systems*, vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005.
[7] P. Benner, H. Mena, and J. Saak, *On the parameter selection problem in the Newton-ADI iteration for large-scale Riccati equations*, Electr. Trans. Num. Anal., 29 (2008).
[8] P. Benner and E. Quintana-Ortí, *Solving stable generalized Lyapunov equations with the matrix sign function*, Numer. Algorithms, 20 (1999), pp. 75–100.
[9] P. Benner and J. Saak, *Linear-quadratic regulator design for optimal cooling of steel profiles*, Tech. Rep. SFB393/05-05, Sonderforschungsbereich 393 *Parallele Numerische Simulation für Physik und Kontinuumsmechanik*, TU Chemnitz, D-09107 Chemnitz (Germany), 2005. Available from `http://www.tu-chemnitz.de/sfb393/sfb05pr.html`.
[10] ———, *A semi-discretized heat transfer model for optimal cooling of steel profiles*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D. Sorensen, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 353–356.
[11] A. Bensoussan, G. Da Prato, M. C. Delfour, and S. K. Mitter, *Representation and control of infinite dimensional systems*, Systems & Control: Foundations & Applications, Birkhäuser Boston Inc., Boston, MA, second ed., 2007.
[12] Z. Drmac, *Accurate computation of the product-induced singular value decomposition with applications.*, SIAM J. Numer. Anal., 35 (1998), pp. 1969–1994.
[13] F. Feitzinger, T. Hylla, and E. W. Sachs, *Inexact Kleinman-Newton Method for Riccati Equations*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 272–288.
[14] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, third ed., 1996.
[15] L. Grasedyck, *Existence of a low rank or H-matrix approximant to the solution of a Sylvester equation*, Numer. Lin. Alg. Appl., 11 (2004), pp. 371–389.
[16] S. Hammarling, *Newton's method for solving the algebraic Riccati equation*, NPL Report DITC 12/82, National Physical Laboratory, Teddington, Middlesex TW11 OLW, U.K., 1982.
[17] ———, *Numerical solution of the stable, non-negative definite Lyapunov equation*, IMA J. Numer. Anal., 2 (1982), pp. 303–323.
[18] M. Heyouni and K. Jbilou, *An extended block Arnoldi algorithm for large-scale solutions of the continuous-time algebraic Riccati equation*, Electr. Trans. Num. Anal., 33 (2009), pp. 53–62.
[19] M. Hochbruck and G. Starke, *Preconditioned Krylov subspace methods for Lyapunov matrix equations*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 156–171.
[20] D. Hu and L. Reichel, *Krylov-subspace methods for the Sylvester equation*, Linear Algebra Appl., 172 (1992), pp. 283–313.
[21] I. Jaimoukha and E. Kasenally, *Krylov subspace methods for solving large Lyapunov equations*, SIAM J. Numer. Anal., 31 (1994), pp. 227–251.
[22] K. Jbilou, *An arnoldi based algorithm for large algebraic Riccati equations*, tech. rep., L.M.P.A., Jan. 2006.
[23] ———, *ADI preconditioned Krylov methods for large Lyapunov matrix equations.*, tech. rep., L.M.P.A., Nov. 2008.
[24] K. Jbilou and A. Riquet, *Projection methods for large Lyapunov matrix equations*, Linear Algebra Appl., 415 (2006), pp. 344–358.
[25] D. Kleinman, *On an iterative technique for Riccati equation computations*, IEEE Trans. Automat. Control, AC-13 (1968), pp. 114–115.
[26] M. Köhler and J. Saak, *Efficiency improving implementation techniques for large scale matrix equation solvers*, Chemnitz Scientific Computing Prep., TU Chemnitz, 2009. In preparation.
[27] P. Lancaster and L. Rodman, *The Algebraic Riccati Equation*, Oxford University Press, Oxford, 1995.
[28] P. Lancaster and M. Tismenetsky, *The Theory of Matrices*, Academic Press, Orlando, 2nd ed., 1985.

[29] I. Lasiecka and R. Triggiani, *Control Theory for Partial Differential Equations: Continuous and Approximation Theories I. Abstract Parabolic Systems*, Cambridge University Press, Cambridge, UK, 2000.
[30] J.-R. Li, *Model Reduction of Large Linear Systems via Low Rank System Gramians*, PhD thesis, Massachusettes Institute of Technology, September 2000.
[31] J.-R. Li and J. White, *Low rank solution of Lyapunov equations*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 260–280.
[32] M. Marcus and H. Minc, *A Survey of Matrix Theory and Matrix Inequalities*, Allyn and Bacon, Boston, 1964.
[33] K. Morris and C. Navasca, *Solution of algebraic Riccati equatons arising in control of partial differential equations.*, in Control and Boundary Analysis., J. P. Zolesio and J. Cagnol, eds., vol. 240 of Lecture Notes in Pure Appl. Math., CRC Press, 2005.
[34] D. Peaceman and H. Rachford, *The numerical solution of elliptic and parabolic differential equations*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 28–41.
[35] T. Penzl, *A cyclic low rank Smith method for large sparse Lyapunov equations*, SIAM J. Sci. Comput., 21 (2000), pp. 1401–1418.
[36] ———, Lyapack *Users Guide*, Tech. Rep. SFB393/00-33, Sonderforschungsbereich 393 *Numerische Simulation auf massiv parallelen Rechnern*, TU Chemnitz, 09107 Chemnitz, Germany, 2000. Available from `http://www.tu-chemnitz.de/sfb393/sfb00pr.html`.
[37] ———, *Algorithms for model reduction of large dynamical systems*, Linear Algebra Appl., 415 (2006), pp. 322–343. (Reprint of Technical Report SFB393/99-40, TU Chemnitz, 1999.).
[38] G. Quintana-Ortí, X. Sun, and C. H. Bischof, *A BLAS-3 version of the QR factorization with column pivoting.*, SIAM J. Sci. Comput., 19 (1998), pp. 1486–1494.
[39] Y. Saad, *Numerical solution of large Lyapunov equation*, in Signal Processing, Scattering, Operator Theory and Numerical Methods, M. A. Kaashoek, J. H. van Schuppen, and A. C. M. Ran, eds., Birkhäuser, 1990, pp. 503–511.
[40] J. Saak, *Efficient Numerical Solution of Large Scale Algebraic Matrix Equations in PDE Control and Model Order Reduction*, PhD thesis, TU Chemnitz, July 2009. Available from `http://www.tu-chemnitz.de/~saak/Data/Diss-web.pdf`.
[41] J. Sabino, *Solution of Large-Scale Lyapunov Equations via the Block Modified Smith Method*, PhD thesis, Rice University, Houston, Texas, June 2007. available from: `http://www.caam.rice.edu/tech_reports/2006/TR06-08.pdf`.
[42] V. Simoncini, *A new iterative method for solving large-scale Lyapunov matrix equations*, SIAM J. Sci. Comput., 29 (2007), pp. 1268–1288.
[43] 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.
[44] E. Wachspress, *Iterative solution of the Lyapunov matrix equation*, Appl. Math. Letters, 107 (1988), pp. 87–90.
[45] ———, *The ADI model problem*, 1995. Available from the author.
[46] ———, *ADI iteration parameters for the Sylvester equation*, 2000. Available from the author.