NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
*Numer. Linear Algebra Appl.* 2008; **15**:755–777
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nla.622

# Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems

Peter Benner[1,*,†,‡], Jing-Rebecca Li[2,§] and Thilo Penzl[3,¶]

[1]*Fakultät für Mathematik, TU Chemnitz, 09107 Chemnitz, Germany*
[2]*INRIA-Rocquencourt, Projet POEMS, Domaine de Voluceau–Rocquencourt–B.P. 105, 78153 Le Chesnay Cedex, France*
[3]*Last address: Department of Mathematics and Statistics, University of Calgary, Calgary, Alta., Canada T2N 1N4*

SUMMARY

We study large-scale, continuous-time linear time-invariant control systems with a sparse or structured state matrix and a relatively small number of inputs and outputs. The main contributions of this paper are numerical algorithms for the solution of large algebraic Lyapunov and Riccati equations and linear-quadratic optimal control problems, which arise from such systems. First, we review an alternating direction implicit iteration-based method to compute approximate low-rank Cholesky factors of the solution matrix of large-scale Lyapunov equations, and we propose a refined version of this algorithm. Second, a combination of this method with a variant of Newton's method (in this context also called Kleinman iteration) results in an algorithm for the solution of large-scale Riccati equations. Third, we describe an implicit version of this algorithm for the solution of linear-quadratic optimal control problems, which computes the feedback directly without solving the underlying algebraic Riccati equation explicitly. Our algorithms are efficient with respect to both memory and computation. In particular, they can be applied to problems of very large scale, where square, dense matrices of the system order cannot be stored in the computer memory. We study the performance of our algorithms in numerical experiments. Copyright © 2008 John Wiley & Sons, Ltd.

Received 5 January 2008; Revised 18 July 2008; Accepted 22 July 2008

KEY WORDS: Lyapunov equation; algebraic Riccati equation; linear-quadratic optimal control; LQR problem; Newton's method; low-rank approximation; control system; sparse matrices

*Correspondence to: Peter Benner, Fakultät für Mathematik, TU Chemnitz, 09107 Chemnitz, Germany.
†E-mail: benner@mathematik.tu-chemnitz.de
‡This author's part of the work was completed while he was with the Department of Mathematics, University of Bremen, Germany.
§This author's part of the work was completed while she was with the Department of Mathematics, MIT, Cambridge, MA, U.S.A.
¶Thilo Penzl (28/06/1968–17/12/1999) was supported by DAAD (*Deutscher Akademischer Austauschdienst*), Bonn, Germany.

## PREAMBLE

This is an unabridged reprint of an unpublished manuscript written by the authors in 1999. The latest version of this work was finalized on December 8, 1999. Thilo Penzl died in a tragic avalanche accident on December 17, 1999, in Canada, and thus, work on the manuscript came to an abrupt end. Nevertheless, this work was often cited and requested by people working on large-scale matrix equations as it forms the basis for parts of the software package LyaPack [1]. This special issue on *Large-Scale Matrix Equations of Special Type* offered the possibility to make this draft available in the open literature. Due to this fact, we did not take into account any development since 2000 except for updated references. Most changes regarding the original manuscript, in large parts prepared by Thilo Penzl shortly before his sudden death, are to be found in some re-wording and corrections of grammar and typos. Sometimes, we add a footnote if a statement in the original draft is no longer (completely) true due to recent developments. We hope that this publication of the manuscript serves the community and helps to bear in remembrance the influential scientific work Thilo Penzl contributed in his by way too short career.

## 1. INTRODUCTION

Continuous-time, finite-dimensional, linear time-invariant (LTI), dynamical systems play an essential role in modern control theory. In this paper we deal with state–space realizations

$$\begin{aligned} \dot{x}(\tau) &= Ax(\tau) + Bu(\tau), \quad x(0) = x_0 \\ y(\tau) &= Cx(\tau) \end{aligned} \tag{1}$$

of such systems. Here, $A \in \mathbb{R}^{n,n}$, $B \in \mathbb{R}^{n,m}$, $C \in \mathbb{R}^{q,n}$, and $\tau \in \mathbb{R}$. (Throughout this paper, $\mathbb{R}$, $\mathbb{R}^{n,m}$, $\mathbb{C}$, $\mathbb{C}^{n,m}$ denote the sets of real numbers, real $n \times m$ matrices, complex numbers, and complex $n \times m$ matrices, respectively.) The vector-valued functions $u$, $x$, and $y$ are called *input*, *state*, and *output* of the LTI system (1), respectively. The *order* of this LTI system is $n$. Typically, $n \gg m, q$.

In the last 2–3 decades, much research has addressed the construction of numerically robust algorithms for a variety of problems that arise in context with linear systems as in (1). Such problems are, for example, optimal control, robust control, system identification, game theory, model reduction, and filtering, see e.g. [2–6]. However, these methods generally have at least a memory complexity $\mathcal{O}(n^2)$ and a computational complexity $\mathcal{O}(n^3)$, regardless whether or not the system matrix $A$ is sparse or otherwise structured. Therefore, the majority of numerical algorithms in linear control theory is restricted to systems of moderate order. Of course, the upper limit for this order depends on the problem to be solved as well as the particular computing environment and may vary between a few hundreds and a few thousands. However, a significant number of applications lead to dynamical systems of larger order. Large systems arise from the semidiscretization of (possibly linearized) partial differential equations (PDEs) by means of finite differences or finite elements, see e.g. [7–10], and many more. Another source for such systems is circuit design and simulation (VLSI computer-aided design, RCL interconnect modeling, etc.) [11, 12]. Further examples include, for instance, large mechanical space structures [13, 14].

Mostly, systems originating from the applications mentioned above possess two interesting properties. First, their order $n$ is large (say, $n > 1000$), but the dimensions of the input and output spaces are relatively small ($m, q \ll n$, often $m, q \leqslant 10$). For example, the order of a system arising from a



DOI: 10.1002/nla

parabolic PDE is about the number of grid points or mesh nodes used for the semidiscretization, which is relatively large. In contrast, $m$ and $q$ are often quite small and independent of the fineness of the discretization. Second, the system matrix $A$ is structured. Often, $A$ is a sparse matrix or it is implicitly represented as a product of sparse matrices and inverses of sparse matrices. In general, this structure allows the numerically inexpensive realization of matrix-vector products and the efficient solution of systems of linear equations with $A$. The smallness of $m$ and $q$ and the structure of $A$ are most important for the usefulness of the algorithms we will present.

Basically, this paper contains three contributions which are algorithms for large continuous-time algebraic Lyapunov equations (CALEs), continuous-time algebraic Riccati equations (CAREs), and linear-quadratic optimal control problems. Among the latter problem class, we will specifically be concerned with the problem commonly known as *linear-quadratic regulator problem* in control theory, see e.g. [3, 9, 15, 16]. We therefore use the common abbreviation *LQR problem* when addressing it. We will solve large CALEs by a method based on the well-known *alternating direction implicit* (ADI) iteration. The basic version of this method has been proposed before, but here we present an improved algorithm and discuss numerical aspects in more detail. There exist plenty of applications for CALEs, one of which is the solution of CAREs. We will combine Newton's method with the ADI-based CALE solver to compute efficiently the stabilizing solution of large CAREs. Among many other applications in control theory, LQR problems can be solved by computing the solution of CAREs. An implicit version of our combined algorithm (similar to an algorithm proposed in [17]) will be used to solve LQR problems in a more memory efficient way without forming CARE solutions.

This paper is organized as follows. For the ease of exposition, we will introduce the problems addressed in it in reverse order. We describe the basics of the LQR problem in Section 2 and the classical Newton method for CAREs in Section 3. In Section 4 ,we discuss the efficient solution of large CALEs by the ADI-based method. The combination of this method with the Newton iteration for CAREs is proposed in Section 5, whereas the implicit formulation of the resulting algorithm is presented in Section 6. The results of numerical experiments with our algorithms are reported in Section 7. Finally, concluding remarks are provided in Section 8.

## 2. THE LINEAR-QUADRATIC OPTIMAL CONTROL PROBLEM

Consider the linear-quadratic optimal control problem (referred to as LQR problem in the following as explained above), where the *cost functional*

$$\mathscr{J}(u, y, x_0) = \frac{1}{2}\int_0^\infty y(\tau)^{\mathrm{T}} Q y(\tau) + u(\tau)^{\mathrm{T}} R u(\tau)\,\mathrm{d}\tau \tag{2}$$

with

$$Q = Q^{\mathrm{T}} \geqslant 0 \quad \text{and} \quad R = R^{\mathrm{T}} > 0 \tag{3}$$

is to be minimized and the LTI system (1) represents the constraints. The solution of this problem is determined by the linear state feedback

$$u(\tau) = -R^{-1} B^{\mathrm{T}} X_* x(\tau) =: -K_*^{\mathrm{T}} x(\tau) \tag{4}$$



DOI: 10.1002/nla

where $X_*$ is the symmetric, positive semidefinite, stabilizing solution of the CARE

$$\mathcal{R}(X) = C^{\mathrm{T}}QC + A^{\mathrm{T}}X + XA - XBR^{-1}B^{\mathrm{T}}X = 0 \tag{5}$$

e.g. [4, 6]. A solution $X$ of this CARE is called *stabilizing* iff the closed-loop matrix $A - BR^{-1}B^{\mathrm{T}}X$ is stable. We call a matrix *stable* iff each of its eigenvalues has a negative real part. Under moderate assumptions a unique stabilizing solution of (5) exists, see e.g. [2, 16]. The transpose of the matrix $K_*$ defined by (4) is called *optimal state feedback*.

The CARE offers a possibility to solve the LQR problem analytically using (4). Of course, this is only feasible for very small dimension $n$. Therefore, usually the computation of the optimal control involves the numerical solution of the CARE (5). Many methods for this purpose have been devised since the occurrence of the CARE in the solution of the LQR problem in the early 60s [18]. We will give a brief survey on the usual approaches in the next section. But of course, there are other possibilities to solve the LQR problem:

1. Solve it as *constrained optimization* problem. This requires to discretize the integral expression as well as the differential equation and to form a quadratic program, which can then be solved by any method feasible for quadratic programming.
2. Using either Pontryagin's maximum principle or a direct proof based on the calculus of variations (see e.g. [4, 19] and references therein), it can be shown that the optimal control and the associated trajectory can be obtained from the two-point boundary value problem

$$\begin{bmatrix} \dot{x}(t) \\ \dot{\mu}(t) \end{bmatrix} = \begin{bmatrix} A & BR^{-1}B^{\mathrm{T}} \\ C^{\mathrm{T}}QC & -A^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} x(t) \\ \mu(t) \end{bmatrix}, \quad \begin{matrix} x(0) = x^0 \\ \lim_{t\to\infty} \mu(t) = 0 \end{matrix} \tag{6}$$

   where $\mu(t)$ is called the co-state of the system and the optimal control is obtained from $\mu$ as $u(t) = R^{-1}B^{\mathrm{T}}\mu(t)$. This offers the possibility to solve the LQR problem using numerical methods for linear boundary value problems.

Both these approaches involve a discretization error. Moreover, the infinite time interval needs to be truncated and it is difficult to define a suitable interval $[0, T]$ so that an accurate solution is obtained in a reasonable time. Another difficulty with the Approach 1 is the enormous size of the resulting quadratic program due to the discretization of the cost functional (2) and the differential equation (1), see also [20] for more details on this approach. Another advantage of the solution via the CARE is the feedback form (4) of the optimal control. In contrast to the solutions obtained by the other two approaches, this allows the direct implementation as a controller in a closed-loop control system.

## 3. NEWTON'S METHOD FOR ALGEBRAIC RICCATI EQUATIONS

The numerical solution of CAREs of the form (5) is the central task in solving optimal control or optimization problems for linear continuous-time systems, such as stabilization, LQR problems, Kalman filtering, linear-quadratic Gaussian control, $\mathcal{H}_2$ optimal control, etc.; see e.g. [4, 6, 15, 16, 21] and the references given therein. We will assume here that $Q$ is positive semidefinite and $R$ is positive definite and that a unique stabilizing solution of (5) exists. Note that the



DOI: 10.1002/nla

methods to be considered here rely on these assumptions as we make use of the fact that the right-hand side in the equivalent formulation of (5),

$$(A-BR^{-1}B^{\mathrm{T}}X)^{\mathrm{T}}X+X(A-BR^{-1}B^{\mathrm{T}}X)=-C^{\mathrm{T}}QC-XBR^{-1}B^{\mathrm{T}}X \tag{7}$$

is positive semidefinite. CAREs as they occur in model reduction [22, 23] or $\mathscr{H}_\infty$ (sub-) optimal control [24–26] differ from (5) in that the right-hand side in (7) is in general indefinite. Therefore, these CAREs cannot be treated directly by the methods presented in this paper.

We are interested in solving (5) for large and structured coefficient matrices $A$ and low-rank matrices $S$ and $G$. The usual solution methods for CAREs such as the Schur vector method [27], the sign function method [28–30], or symplectic methods [31–33] do not make (full) use of these structures and require $O(n^3)$ flops and workspace of size $O(n^2)$ even for sparse problems, and are therefore not suitable for our purpose. (For surveys of the most popular approaches with cubic complexity, see [3–6, 34].)

All the methods mentioned above use the corresponding Hamiltonian eigenvalue problem to solve (5). Methods that exploit the sparsity in the Hamiltonian matrix are proposed in [35, 36]. There the Hamiltonian matrix is projected onto a reduced-order Hamiltonian for which the corresponding CARE of reduced size is solved. This solution is then prolongated to a full-size Riccati solution matrix. Similarly, low-rank methods that are based on the projection of the underlying LTI system onto a reduced-order model can be found in [37, 38]. Usually, these projections are computed using a Krylov subspace method. No convergence theory for these methods is known and often, small residuals can only be obtained with these approaches for relatively large 'reduced-order' systems.

Another approach to solve CAREs is to treat them as systems of nonlinear equations. It is then straightforward to use Newton's method or its relatives [39–42] for its solution. As the Newton iteration (see Algorithms 1 and 2 below) consists of solving a sequence of Lyapunov equations, it can be used to solve large-scale problems if efficient methods for solving large Lyapunov equations are available. This idea has been used in [17, 43, 44]. The methods there rely on the implementation of an accelerated Smith iteration or a block SOR method for the Lyapunov equation. Although the methods in [43, 44] explicitly rely on the specific structure of the underlying control problem, the method in [17] can be considered as a general-purpose solver for computing the optimal state feedback in large and sparse LQR problems. This latter method has some close relations to the method that will be developed in this paper. We discuss these relations and the differences in Section 6.

Our methods make use of a factorization $TT^{\mathrm{T}}=C^{\mathrm{T}}QC+XBR^{-1}B^{\mathrm{T}}X$, where $T\in\mathbb{R}^{n\times t}$ with $t\ll n$ and the fact that in this case, the solution of the CARE can be approximated by a product $ZZ^{\mathrm{T}}$ of low rank.[‖]

Similar approaches have been used to compute a factored solution of the CARE in [23, 45]. Both approaches use Hammarling's method [46] for solving the Lyapunov equations during the iteration steps of Newton's method. The method for solving Lyapunov equations suggested in [46] works with the $n\times n$ Cholesky factors of the solution and never forms the full solution matrix. Nevertheless, the use of the QR algorithm, as initial step in this approach, requires dense

[‖]It turns out that in some situations, we will have to use complex arithmetic so that the approximate CARE solution will have the form $ZZ^H$, where $Z^H$ denotes the conjugate complex transpose of $Z$.



DOI: 10.1002/nla

**Algorithm 1.** (Newton's method for the CARE (5))

INPUT: $A$, $B$, $C$, $Q$, $R$ as in (5) and a starting guess $X^{(0)}$.

OUTPUT: An approximate solution $X^{(k)}$ of (5).

FOR $k=1,2,\ldots,$

$K^{(i-1)}=X^{(k-1)}BR^{-1}$.

Solve

$$(A^{\mathrm{T}}-K^{(k-1)}B^{\mathrm{T}})N^{(k)}+N^{(k)}(A-BK^{(k-1)^{\mathrm{T}}})=-\mathscr{R}(X^{(k-1)}) \tag{8}$$

for $N^{(k)}$.

$X^{(k)}=X^{(k-1)}+N^{(k)}$.

END

**Algorithm 2.** (Newton's method for the CARE (5)—Kleinman iteration)

INPUT: $A$, $B$, $C$, $Q$, $R$ as in (5) and a starting guess $K^{(0)}$ for the feedback matrix.

OUTPUT: An approximate solution $X^{(k)}$ of (5) and an approximation $K^{(k)}$ of the optimal state feedback.

FOR $k=1,2,\ldots,$

Solve

$$(A^{\mathrm{T}}-K^{(k-1)}B^{\mathrm{T}})X^{(k)}+X^{(k)}(A-BK^{(k-1)^{\mathrm{T}}})=-C^{\mathrm{T}}QC-K^{(k-1)}RK^{(k-1)^{\mathrm{T}}} \tag{9}$$

for $X^{(k)}$.

$K^{(k)}=X^{(k)}BR^{-1}$.

END

matrix algebra at a cost of $O(n^3)$ flops and $O(n^2)$ memory and is therefore not suitable for our purpose.

In the remainder of this section we review Newton's method for the special form of the CARE given in (5). Throughout this paper we use superscripts in parentheses to label variables within the Newton iteration.

There are two possible formulations of Newton's method for CAREs. The first one (Algorithm 1) can be considered as the standard formulation of Newton's method applied to a system of nonlinear equations. The formulation of Newton's method proposed by Kleinman [41] (see Algorithm 2) is one of the standard methods for computing the stabilizing solution of the CARE (5). It is based on (7).

Both formulations are mathematically equivalent. If $K^{(0)}$ is identical in both implementations, i.e. if in Algorithm 2, $K^{(0)}=X^{(0)}BR^{-1}$, then they deliver the same sequences of $X^{(k)}$'s and $K^{(k)}$'s. The following theorem summarizes the convergence theory for Algorithms 1 and 2.



DOI: 10.1002/nla

*Theorem 1*
If $(A, B)$ is stabilizable, i.e. $\operatorname{rank}[A-\lambda I_n, B]=n$ for all $\lambda\in\mathbb{C}$ with non-negative real part, then choosing $X^{(0)}=X^{(0)^{\mathrm{T}}}\in\mathbb{R}^{n\times n}$ in Algorithm 1 or $K^{(0)}\in\mathbb{R}^{n\times m}$ in Algorithm 2, such that $A-BK^{(0)^{\mathrm{T}}}$ is stable, the iterates $X^{(k)}$ and $K^{(k)}$ satisfy the following assertions:

(a) For all $k\geqslant 0$, the matrix $A-BK^{(k)^{\mathrm{T}}}$ is stable and the Lyapunov equations (8) and (9) admit unique solutions, which are positive semidefinite.
(b) $\{X^{(k)}\}_{k=1}^{\infty}$ is a non-increasing sequence, satisfying $X^{(k)}\geqslant X^{(k+1)}\geqslant 0$ for all $k\geqslant 1$. Moreover, $X_*=\lim_{k\to\infty}X^{(k)}$ exists and is the unique stabilizing solution of the CARE (5).
(c) There exists a constant $\gamma>0$, such that

$$\|X^{(k+1)}-X_*\|\leqslant\gamma\|X^{(k)}-X_*\|^2,\quad k\geqslant 1$$

i.e. the $X^{(k)}$ converge globally quadratic to $X_*$ from any stabilizing initial guess.

A complete proof of the above result can be found, e.g. in [16].

Though Algorithms 1 and 2 look very similar, there are some subtle differences, which play a fundamental role in their implementation for large-scale structured systems.

The main disadvantage of Algorithm 1 is that the right-hand side of the CALE (8) is the residual of the CARE, which is in general an indefinite, full-rank matrix. As the performance of the CALE solver that will be used in this paper strongly depends on the low-rank structure of the right-hand side, the formulation of Algorithm 2 is preferable as the right-hand side of the CALE (9) has rank at most $m+q$. Moreover, if the solution of (9) is computed by a low-rank method like the method presented in Section 4, yielding a low-rank solution of the CALE, the approximate solutions of the CARE computed by Algorithm 2 themselves will be of low rank. This cannot be guaranteed for Algorithm 1 as there, the low-rank solution of the CALE is added to the last iterate such that in general, the rank of the iterates computed this way will be increasing. Moreover, for initialization, a stabilizing symmetric matrix $X^{(0)}\in\mathbb{R}^{n,n}$ is needed in Algorithm 1, whereas only a stabilizing feedback $K^{(0)}\in\mathbb{R}^{n,m}$ is needed in Algorithm 2. Starting from $K^{(0)}$ is in particular favorable for problems, where $m\ll n$ and the implicit version of Newton's method, computing an approximation to $K_*$ as presented in Section 6, is to be used. For these reasons, in this paper we exclusively use implementations of Newton's method for CAREs based on the formulation given in Algorithm 2.

*Remark 2*
The computation of a stabilizing initial guess $X^{(0)}$ or feedback $K^{(0)}$, respectively, is a computational challenge by itself if $A$ is not stable—otherwise, i.e. if all eigenvalues of $A$ have negative real part, one can simply use $X^{(0)}=0$ or $K^{(0)}=0$. Stability of $A$ can often be assumed in applications arising from parabolic PDEs, so that for a large class of practical applications, we do not need to compute stabilizing initial guesses or feedbacks. In case we have to find a stabilizing $X^{(0)}$ or $K^{(0)}$, one can follow the approach taken in [17] based on Chandrasekhar's method. For small, dense systems (partial) stabilization techniques based on solving a Lyapunov equation or pole placement have been suggested, see e.g. [3, 6] and references therein. It is an open problem to extend these methods to large and sparse systems. We will not discuss this any further in this paper.

In the following section, we discuss methods for solving the Lyapunov equations arising in each Newton step of Algorithms 1 and 2.



DOI: 10.1002/nla

## 4. SOLUTION OF LARGE LYAPUNOV EQUATIONS

Besides Newton's method for CAREs, several topics in control theory, such as stability analysis, stabilization, balancing, and model reduction involve CALEs; see e.g. [47–50]. These linear matrix equations usually have the structure

$$FX + XF^{\mathrm{T}} = -GG^{\mathrm{T}} \tag{10}$$

where $G \in \mathbb{R}^{n,t}$ is a rectangular matrix with $t \leqslant n$ and the matrix $F \in \mathbb{R}^{n,n}$ is stable. The stability of $F$ is sufficient for the existence of a solution matrix $X \in \mathbb{R}^{n,n}$, which is unique, symmetric, and positive semidefinite. If the pair $(F, G)$ is controllable, then $X$ is even positive definite. These and other theoretical results on CALEs can be found in [51, 52], for example. Note that (10) is mathematically equivalent to a system of linear equations with $\mathcal{O}(n^2)$ unknowns. For this reason, CALEs of order $n > 1000$ are said to be of large scale.

The Bartels–Stewart method [53] and Hammarling's method [45] are the numerical standard methods for CALEs. While the first is applicable to the more general Sylvester equation, the second tends to deliver more accurate results in the presence of round-off errors. Both methods require the computation of the Schur form of $F$. As a consequence, they generally cannot profit from sparsity or other structures in the equation. The squared Smith method [54] and the sign function method [30] are iterative methods, which cannot exploit sparsity or structures as well.** However, they are of particular interest when dense CALEs are to be solved on parallel computers [57, 58].

ADI [59, 60] is an iterative method, which often delivers good results for sparse or structured CALEs. The solution methods mentioned so far have the computational complexity $\mathcal{O}(n^3)$, except for the ADI method. Its complexity strongly depends on the structure of $F$ and can be much lower. All methods have the memory complexity $\mathcal{O}(n^2)$ because they generate the dense solution matrix $X$ explicitly. It should be stressed that often the memory complexity, rather than the amount of computation, is the limiting factor for the applicability of solution methods for large CALEs.

In many cases, large CALEs have right-hand sides of very low rank $t \ll n$. If this is true, the non-negative eigenvalues of the solution $X$ tend to decay very fast, which is discussed in [61–63]. Thus, the solution matrix can be approximated very accurately by a positive semidefinite matrix of relatively low rank. This property is important for what we call *low-rank methods* for CALEs. Low-rank methods are the only existing methods, which can solve very large CALEs. (Here, in particular, problems of order $n$ are considered to be 'very large', if it is impossible to store dense $n$-by-$n$ matrices in the computer memory.) Low-rank methods avoid forming the solution $X$ explicitly. Instead, they generate low-rank Cholesky factors (LRCFs) $Z$, such that the LRCF product $ZZ^H$ approximates $X$. Note that throughout this paper the attribute 'low-rank' is used, when the rank of the corresponding matrix is much smaller than the system order $n$. For instance, the matrix $Z \in \mathbb{C}^{n,r}$ with $r \ll n$ is called a LRCF, although its (column) rank is generally full. Moreover, the term Cholesky factor will not imply triangular structure—usually, the LRCFs $Z$ are dense, rectangular matrices. Most low-rank methods [38, 64–66] are Krylov subspace methods, which are based either on the Lanczos process or the Arnoldi process (see e.g. [67, Section 9]).

**Recent results show that data-sparse structures like the hierarchical matrix format can be exploited, see [55, 56], so that these methods have become alternatives for certain classes of large-scale problems.



DOI: 10.1002/nla

Furthermore, there are low-rank methods [66, 68] based on the explicit representation of the CALE solution in integral form [51].

In this paper, we use low-rank methods based on the ADI iteration [60]

$$\begin{aligned}(F+p_iI_n)X_{i-1/2} &= -GG^{\mathrm{T}} - X_{i-1}(F^{\mathrm{T}} - p_iI_n)\\ (F+\bar{p}_iI_n)X_i{}^{\mathrm{T}} &= -GG^{\mathrm{T}} - X_{i-1/2}{}^{\mathrm{T}}(F^{\mathrm{T}} - \bar{p}_iI_n)\end{aligned} \tag{11}$$

where $X_0=0$, $X_{i-1/2}\in\mathbb{C}^{n,n}$ are auxiliary matrices, and $p_i\in\mathbb{C}_-$ (where $\mathbb{C}_-$ denotes the open left complex half-plane) are certain shift parameters. If optimal parameters are used, the sequence $\{X_i\}_{i=0}^{\infty}$ converges superlinearly toward the solution $X$ of (10). See, for example, [69–71] and references given therein for a more detailed discussion on the convergence of the ADI iteration.

Different low-rank methods based on the ADI iteration were independently derived in [72] and [73] and later published in a more complete form in [74]. In this paper we make use of a slight modification of the iteration proposed in [72]. This method, which we refer to as LRCF-ADI, is derived as follows. We first replace the ADI iterates $X_i$ by the product $Z_iZ_i^H$ and rewrite the iteration in terms of the LRCFs $Z_i$. LRCF-ADI, as described in Algorithm 3 below, is obtained by a rearrangement of the resulting iteration. This rearrangement lowers the computational cost significantly. Note that, unlike in the original ADI iteration, the parameters $p_{i-1}$ and $p_i$ are involved in the $i$th iteration step. See [72] for details.

**Algorithm 3.** (Low-rank Cholesky factor ADI iteration (LRCF-ADI))

INPUT: $F$, $G$ as in (10), $\{p_1, p_2, \ldots, p_{i_{\max}}\}$.

OUTPUT: $Z=Z_{i_{\max}}\in\mathbb{C}^{n,ti_{\max}}$, such that $ZZ^H\approx X$, where $X$ solves (10).

1. $V_1=\sqrt{-2\operatorname{Re}p_1}(F+p_1I_n)^{-1}G$.
2. $Z_1=V_1$.

FOR $i=2,3,\ldots,i_{\max}$

3. $V_i=\sqrt{\operatorname{Re}p_i/\operatorname{Re}p_{i-1}}(V_{i-1}-(p_i+\bar{p}_{i-1})(F+p_iI_n)^{-1}V_{i-1})$.
4. $Z_i=[Z_{i-1}\ \ V_i]$.

END

Let $\mathscr{P}_j$ be either a negative real number or a pair of complex conjugate numbers with negative real part and non-zero imaginary part. We call a parameter set of type $\{p_1, p_2, \ldots, p_i\}=\{\mathscr{P}_1, \mathscr{P}_2, \ldots, \mathscr{P}_j\}$ a *proper* parameter set. Throughout this paper, we assume that proper parameter sets $\{p_1, p_2, \ldots, p_{i_{\max}}\}$ are used in the LRCF-ADI iterations.

If $X_i=Z_iZ_i^H$ is generated by a proper parameter set $\{p_1, \ldots, p_i\}$, then $X_i$ is real, which follows from (11). However, if there are non-real parameters in this subsequence, $Z_i$ is not real. As we consider real CALEs, one is often interested in real LRCFs. Moreover, it is often desired to implement numerical algorithms using exclusively real arithmetics. Both objectives are attained by the following reformulation of LRCF-ADI.



DOI: 10.1002/nla

Let $\{p_1, \ldots, p_{i_{\max}}\}$ be a proper parameter (sub)set. Then, the subsequences of LRCFs $Z_i$, which are formed in Steps 5 and 8 of Algorithm 4 on page 765 are real and their products $Z_i Z_i^{\mathrm{T}}$ are equal to the corresponding products $Z_i Z_i^H$ in Algorithm 3 as well as to the matrix $X_i$ in (11) when exact arithmetic is used.

In the remaining part of this section we shall discuss implementational details of Algorithm 3, most of which can easily be extended to Algorithm 4.

### *4.1. Basic matrix operations*

An efficient implementation of LRCF-ADI relies on the following two basic matrix operations:

$$Y \longleftarrow FW \tag{12}$$

$$Y \longleftarrow (F + pI_n)^{-1} W \quad (\mathrm{Re}\, p < 0) \tag{13}$$

Here, $W \in \mathbb{C}^{n,t}$ with $t \ll n$. For structured matrices $F$, a large variety of numerical techniques exist to compute (12) and (13) very efficiently. For example, if $F$ is a circulant matrix, both operations can be realized inexpensively by means of fast Fourier transformations; e.g. [67, 75]. If $F$ is a sparse matrix, then the computation (12) is trivial and highly efficient, whereas the solution of the system of linear equations (13) can be realized by direct or iterative methods. In the first case, sparse direct solvers (e.g. [76]) can be applied in combination with bandwidth reduction algorithms; e.g. [77]. In the second case, a large number of iterative methods are available, such as preconditioned Krylov subspace methods for systems of linear equations (for instance, GMRES or QMR combined with ILU; e.g. [78]), geometric multigrid methods (e.g. [79]), or algebraic multilevel methods; e.g. [80]. Note that in contrast to LRCF-ADI, where systems with right-hand sides containing only few vectors must be solved, the applicability of iterative methods for the solution of the systems of linear equations in the conventional ADI formulation (11) is severely restricted, because those systems have right-hand sides containing as many as $n$ vectors.

### *4.2. Stopping criteria*

Only in rare cases (e.g. when $F$ is symmetric and its spectral bounds are known), the number of ADI (or LRCF-ADI) steps needed to attain a certain accuracy can be determined *a priori*. Instead, the decision when to terminate the iteration must generally be made *ad hoc*. For example, this issue is quite straightforward in context with iterative methods for systems of linear equations, such as GMRES or QMR. There, the (normalized) residual norms, which are often generated as a by-product in these algorithms, are generally used for this purpose. In contrast, the construction of inexpensive stopping criteria for the LRCF-ADI iteration is a non-trivial issue because the generation of the $n$-by-$n$ residual matrix and the computation of its (Frobenius) norm would require a prohibitive amount of memory and computation. A much less expensive way to compute this norm without forming the residual matrix explicitly is described in [73, Equation (20)]. However, its computational cost is still relatively high and often larger than that of the actual iteration itself.

We propose an alternative stopping criterion that avoids computing the residual norm. Here, the key observation is the following. Unlike, for example, Krylov subspace methods for systems of linear equations, where the convergence curves (in terms of residual norms) can be quite erratic (e.g. spikes in CGS, plateaus in GMRES or QMR), (LRCF-)ADI with a suitably ordered sequence of (sub-)optimal shift parameters tends to converge smoothly and (super)linearly.



DOI: 10.1002/nla

**Algorithm 4.** (LRCF-ADI iteration - real version (LRCF-ADI-R))

INPUT: $F$, $G$ as in (10), $\{p_1, p_2, \ldots, p_{i_{\max}}\}$.

OUTPUT: $Z = Z_{i_{\max}} \in \mathbb{R}^{n, t i_{\max}}$, such that $ZZ^{\mathrm{T}} \approx X$, where $X$ solves (10).

IF $p_1$ is real

1. $\tilde{V}_1 = (F + p_1 I)^{-1} G$, $\quad V_1 = \sqrt{-2p_1}\,\tilde{V}_1$,
   $Z_1 = V_1$, $\quad i = 2$.

ELSE

2. $\tilde{V}_1 = (F^2 + 2\mathrm{Re}\, p_1 F + |p_1|^2 I_n)^{-1} G$, $\quad \tilde{V}_2 = F\tilde{V}_1$,
   $V_1 = 2\sqrt{-\mathrm{Re}\, p_1}\,|p_1|\,\tilde{V}_1$, $\quad V_2 = 2\sqrt{-\mathrm{Re}\, p_1}\,\tilde{V}_2$,
   $Z_2 = [V_1 \;\; V_2]$, $\quad i = 3$.

END

WHILE $i \leqslant i_{\max}$

IF $p_i$ is real

IF $p_{i-1}$ is real

3. $\tilde{V}_i = \tilde{V}_{i-1} - (p_{i-1} + p_i)(F + p_i I_n)^{-1} \tilde{V}_{i-1}$,

ELSE

4. $\tilde{V}_i = \tilde{V}_{i-1} \; - 2\mathrm{Re}(p_{i-1} + p_i)\tilde{V}_{i-2}$
   $+ (|p_{i-1}|^2 + 2p_i \mathrm{Re}\, p_{i-1} + p_i^2)(F + p_i I_n)^{-1} \tilde{V}_{i-2}$.

END

5. $V_i = \sqrt{-2p_i}\,\tilde{V}_i$, $\quad Z_i = [Z_{i-1} \;\; V_i]$, $\quad i \leftarrow i + 1$.

ELSE

IF $p_{i-1}$ is real

6. $\tilde{V}_i = (F^2 + 2\mathrm{Re}\, p_i F + |p_i|^2 I_n)^{-1} (F\tilde{V}_{i-1} - p_{i-1}\tilde{V}_{i-1})$,

ELSE

7. $\tilde{V}_i = \tilde{V}_{i-2} \; + (F^2 + 2\mathrm{Re}\, p_i F + |p_i|^2 I_n)^{-1} \times$
   $((|p_{i-1}|^2 - |p_i|^2)\tilde{V}_{i-2} - 2\mathrm{Re}(p_{i-1} + p_i)\tilde{V}_{i-1})$.

END

8. $V_i = 2\sqrt{-\mathrm{Re}\, p_i}\,|p_i|\,\tilde{V}_i$, $\quad \tilde{V}_{i+1} = F\tilde{V}_i$,
   $V_{i+1} = 2\sqrt{-\mathrm{Re}\, p_i}\,\tilde{V}_{i+1}$, $\quad Z_{i+1} = [Z_{i-1} \;\; V_i \;\; V_{i+1}]$, $\quad i \leftarrow i + 2$.

END

END

Moreover, the norms $\|V_i\|_F$ tend to decay quite evenly. Thus, it seems natural to terminate the iteration when

$$\frac{\|V_i\|_F}{\|Z_i\|_F} \leqslant \varepsilon \tag{14}$$



DOI: 10.1002/nla

where $\varepsilon$ is a tiny, positive constant (for example, $\varepsilon = n\varepsilon_{\text{mach}}$, where $\varepsilon_{\text{mach}}$ is the machine precision). To do this we need not compute $\|Z_i\|_F$ in each iteration step. Instead, $\|Z_i\|_F^2$ can be accumulated in the course of the iteration as $\|Z_i\|_F^2 = \|Z_{i-1}\|_F^2 + \|V_i\|_F^2$, which is inexpensive because $V_i$ contains only very few columns.

### *4.3. Shift selection*

The choice of the shift parameters $p_i$ is related to a rational approximation problem. Several algorithms for the computation of suboptimal parameter sets or asymptotically optimal parameter sequences have been proposed; see e.g. [69–71, 81] and references therein. These algorithms generally rely on the availability of certain bounds for the spectrum of $F$. A quite simple, numerically inexpensive, heuristic algorithm that circumvents this problem can be found in [73]. This algorithm delivers suboptimal, proper parameter sets.

### *4.4. Comparison of LRCF-ADI and Krylov subspace methods*

In the remainder of this section we would like to explain why we prefer ADI-based low-rank methods to the aforementioned Krylov subspace methods ([38, 65, 66] etc.) to compute low-rank solutions to CALEs. As a first aspect, we compare the convergence and the robustness of both types of low-rank methods. ADI-based low-rank methods converge linearly or superlinearly. In contrast, Krylov subspace methods for CALEs generally exhibit a sublinear convergence; e.g. [38, Figures 1 and 2]. Often a stagnation of the convergence curves can be observed; e.g. [73, Table 5]. Note that this even happens for many symmetric problems, for which the performance of ADI is very fast and reliable. The slower convergence of Krylov subspace methods is not surprising because they are related to polynomials in $F$, whereas ADI is based upon a rational approximation problem. As a consequence, LRCFs delivered by Krylov subspace methods generally have a higher rank and, hence, demand more memory for storage than LRCFs of the same accuracy computed by LRCF-ADI. This is indicated by the results of comparing numerical tests reported in [73]. Furthermore, the smallness of this rank is often crucial for the complexity of 'outer' algorithms, such as the model reduction algorithms in [72, 82]. Undoubtedly, when combined with preconditioners (e.g. ILU), Krylov subspace methods, such as GMRES or QMR, are powerful tools for the solution of very large systems of linear equations. Nevertheless, in our opinion, the applicability of Krylov subspace methods for CALEs is much more limited because no way has been found to involve preconditioning in these algorithms. (A Krylov subspace method for CALEs with preconditioning has been proposed in [83]. However, this algorithm does not deliver LRCFs.) In view of robustness, it should also be noted that Krylov subspace methods for CALEs can fail, if $F + F^{\mathrm{T}}$ is not negative definite. In contrast, only the stability of $F$ is needed for the convergence of ADI-based methods. The second important aspect is the complexity w.r.t. memory and computation. Here, the basic difference is that Krylov subspace methods for CALEs are based on matrix products (12), whereas shifted systems of linear equations (13) need to be solved in LRCF-ADI, which is sometimes—but not always—a serious drawback. First, in a number of cases (e.g. if $F$ has a thin band structure or if the underlying system arises from a finite element semidiscretization [82, Example 1]), the operations (12) and (13) require about the same amount of computation. In many other situations, Krylov subspace methods for systems of linear equations with preconditioning can be applied to solve (13) efficiently. Finally, we would like to stress that the computational cost per iteration



DOI: 10.1002/nla

step is raising in Krylov subspace methods for CALEs based on the Arnoldi process, whereas it is constant for Algorithms 3 and 4. Taking all these considerations into account, we believe that LRCF-ADI is in many situations the method of choice for solving large CALEs of type (10), in particular, when the smallness of the rank of the computed LRCF products plays a crucial role and a high accuracy of this approximate solution is desired.††

In the following section, we discuss how to incorporate LRCF-ADI into Newton's method for CAREs.

## 5. THE LOW-RANK CHOLESKY FACTOR NEWTON METHOD

Due to (3) the matrices $Q$ and $R$ can be factored, e.g. by a Cholesky factorization, as

$$Q = \tilde{Q}\tilde{Q}^{\mathrm{T}} \quad \text{and} \quad R = \tilde{R}\tilde{R}^{\mathrm{T}}$$

where the matrices $\tilde{Q} \in \mathbb{R}^{q,h}$ $(h \leqslant q)$ and $\tilde{R} \in \mathbb{R}^{m,m}$ have full rank. Thus, the CALEs to be solved in (9) have the structure

$$F^{(k)}X^{(k)} + X^{(k)}F^{(k)^{\mathrm{T}}} = -G^{(k)}G^{(k)^{\mathrm{T}}}$$

where $F^{(k)} = A^{\mathrm{T}} - K^{(k-1)}B^{\mathrm{T}}$ and $G^{(k)} = [C^{\mathrm{T}}\tilde{Q} \;\; K^{(k-1)}\tilde{R}]$. Note that $G^{(k)}$ contains only $t = m + h \ll n$ columns. Hence, these CALEs can be solved efficiently by LRCF-ADI. The CALE solutions form a sequence of approximate solutions to the CARE (5). Therefore, employing Algorithm 3 in Algorithm 2 for solving (9) yields a method to determine LRCF products $Z^{(k)}Z^{(k)^H}$, which approximate the solutions of CAREs. The resulting algorithm, which we refer to as LRCF-NM, allows the efficient solution of a class of large-scale CAREs. Unlike conventional methods as discussed in Section 3, it can be applied to CAREs, which are so large that the dense $n$-by-$n$ solution matrix cannot be stored in the computer memory.

The remaining part of this section addresses implementational issues and numerical aspects with respect to this algorithm.

### 5.1. Convergence toward the stabilizing solution

In general, the CARE (5) has further solutions besides the stabilizing one, which is computed by Algorithms 1 and 2 according to Theorem 1. In theory, the standard Newton method can be proved to converge to the stabilizing solution provided that the initial feedback or the initial iterate is stabilizing. Since the CALEs in Step 3 of Algorithm 5 are solved only approximately, convergence toward the stabilizing solution cannot be taken for granted. No theoretical results are known on how accurate these CALEs must be solved to ensure convergence toward the stabilizing solution. At least, convergence toward a non-stabilizing CARE solution can be detected in Algorithm 5, because LRCF-ADI in the $k$th iteration step diverges (i.e. $\limsup_{i\to\infty} \| V_i^{(k)} \| = \infty$), if the feedback iterate $K^{(k-1)}$ is not stabilizing. That means, taking an additional iteration step in Algorithm 5 can be used to verify that the computed LRCF product is an approximation to the stabilizing solution

††In [84], a generalized Krylov subspace method is suggested that overcomes some of the drawbacks of Krylov subspace methods discussed here. This approach yields a viable alternative to LRCF-ADI.



DOI: 10.1002/nla

of (5). It should also be noted that in practice, convergence to a non-stabilizing solution has not been observed so far.

**Algorithm 5.** (Low-rank Cholesky factor Newton method (LRCF-NM))

INPUT: $A$, $B$, $C$, $Q$, $R$, $K^{(0)}$ for which $A-BK^{(0)\mathrm{T}}$ 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 the CARE (5).

FOR $k=1,2,\ldots,k_{\max}$

1. Determine (sub)optimal ADI shift parameters $p_1^{(k)}, p_2^{(k)},\ldots$ with respect to the matrix $F^{(k)}=A^{\mathrm{T}}-K^{(k-1)}B^{\mathrm{T}}$ (e.g. [73, Algorithm 1]).
2. $G^{(k)}=[C^{\mathrm{T}}\tilde{Q}\ \ K^{(k-1)}\tilde{R}]$.
3. Compute matrix $Z^{(k)}$ by Algorithm 3 or 4, such that the LRCF product $Z^{(k)}Z^{(k)H}$ approximates the solution of $F^{(k)}X^{(k)}+X^{(k)}F^{(k)\mathrm{T}}=-G^{(k)}G^{(k)\mathrm{T}}$.
4. $K^{(k)}=Z^{(k)}(Z^{(k)H}BR^{-1})$ .

END

### 5.2. Basic matrix operations

In this paragraph a few comments on the realization of the basic matrix operations (12) and (13) will be made. The coefficient matrix of the CALE to be solved in Step 3 is $F^{(k)}=A^{\mathrm{T}}-K^{(k-1)}B^{\mathrm{T}}$. Obviously, the products (12) can be computed efficiently as $Y \longleftarrow A^{\mathrm{T}}W-K^{(k-1)}(B^{\mathrm{T}}W)$, if $A^{\mathrm{T}}W$ can be formed inexpensively. Solving the shifted systems of linear equations (13), whose coefficient matrix is $A^{\mathrm{T}}+pI_n-K^{(k-1)}B^{\mathrm{T}}$, is a bit more complicated. Provided that $(A^{\mathrm{T}}+pI_n)^{-1}W$ can be computed efficiently, we can use the Sherman–Morrison–Woodbury formula (e.g. [67, Section 2.1.3]) to solve (13) as

$$
\begin{aligned}
M_K &\longleftarrow (A^{\mathrm{T}}+pI)^{-1}K^{(k-1)}\\
M_W &\longleftarrow (A^{\mathrm{T}}+pI)^{-1}W\\
Y &\longleftarrow M_W+M_K((I_m-B^{\mathrm{T}}M_K)^{-1}B^{\mathrm{T}}M_W)
\end{aligned}
$$

Finally, we propose two stopping criteria that are related to the residual norm and the 'smallness of changes' in $K^{(k)}$. While the first is a reliable but numerically very expensive criterion, the second is computationally much cheaper, but also less reliable.

### 5.3. Stopping criterion based on the residual norm

In the course of iterative methods, one is usually interested in monitoring the residual norm, which can serve as a stopping criterion for the iteration. Unfortunately, the straightforward computation of $\|\mathscr{R}(Z^{(k)}Z^{(k)H})\|_F$ in the Newton iteration by forming the residual matrix $\mathscr{R}(Z^{(k)}Z^{(k)H})$ (cf. (5)) requires an extensive amount of memory. If $Z^{(k)}$ contains much less columns than rows, the residual norm can be determined efficiently by computing the right-hand term in (15), which is the Frobenius norm of a small matrix. This technique is similar to that in [73, Equation (20)]



DOI: 10.1002/nla

and yields

$$\|\mathscr{R}(Z^{(k)}Z^{(k)H})\|_F = \left\| [C^{\mathrm{T}}\tilde{Q} \;\; A^{\mathrm{T}}Z^{(k)} \;\; Z^{(k)}] \begin{bmatrix} I_h & 0 & 0 \\ 0 & 0 & I_{z_k} \\ 0 & I_{z_k} & -Z^{(k)H}BR^{-1}B^{\mathrm{T}}Z^{(k)} \end{bmatrix} \begin{bmatrix} \tilde{Q}^{\mathrm{T}}C \\ Z^{(k)H}A \\ Z^{(k)H} \end{bmatrix} \right\|_F$$

$$= \left\| R^{(k)} \begin{bmatrix} I_h & 0 & 0 \\ 0 & 0 & I_{z_k} \\ 0 & I_{z_k} & -Z^{(k)H}BR^{-1}B^{\mathrm{T}}Z^{(k)} \end{bmatrix} R^{(k)\mathrm{T}} \right\|_F \tag{15}$$

Here, $I_h, I_{z_k}$ denote identity matrices of proper dimensions, where $z_k$ is the number of columns of $Z^{(k)}$. The triangular matrix $R^{(k)}$ is computed by an 'economy size' QR factorization $U^{(k)}R^{(k)} = [C^{\mathrm{T}}\tilde{Q} \;\; A^{\mathrm{T}}Z^{(k)} \;\; Z^{(k)}]$. Note that $U^{(k)}$ is not accumulated as it is not needed in the evaluation of (15). This way of evaluating the residual norm is much cheaper than computing the explicit residual matrix, but it can still be much more expensive than computing $Z^{(k)}$ itself.

### *5.4. Stopping criterion based on $K^{(k)}$*

If the LQR problem is to be solved, one is interested in the state-feedback matrix $K$ rather than the LRCF $Z$. Therefore, it seems to be reasonable to stop the iteration as soon as the changes in the matrices $K^{(k)}$ become small or more precisely,

$$\frac{\|K^{(k)} - K^{(k-1)}\|_F}{\|K^{(k)}\|_F} \leqslant \varepsilon \tag{16}$$

Here, $\varepsilon$ is a tiny, positive constant (e.g. $\varepsilon = n\varepsilon_{\mathrm{mach}}$). Apparently, this criterion is very inexpensive, because $K \in \mathbb{R}^{n,m}$ and $m \ll n$.

## 6. THE IMPLICIT LOW-RANK CHOLESKY FACTOR NEWTON METHOD

The solution of the LQR problem (1) and (2) is described by the optimal state-feedback matrix $K$, whereas the solution of the CARE or its low-rank approximations only play the role of auxiliary quantities. This provides the motivation for the LRCF-NM-I, which is a mathematically equivalent, implicit version of LRCF-NM. It computes an approximation to $K$ without forming LRCF-NM iterates $Z^{(k)}$ and LRCF-ADI iterates $Z_i^{(k)}$ at all.

The basic idea behind LRCF-NM-I is to generate the matrix $K^{(k)}$ itself in Step 3 of Algorithm 5 instead of solving the CALE for $Z^{(k)}$ and computing the product $K^{(k)} = Z^{(k)}Z^{(k)H}BR^{-1}$ in Step 4. Note that the matrix $K^{(k)}$ can be accumulated in the course of the 'inner' LRCF-ADI iteration as

$$K^{(k)} = \lim_{i \to \infty} K_i^{(k)}$$



DOI: 10.1002/nla

where

$$K_i^{(k)} := Z_i^{(k)} Z_i^{(k)H} B R^{-1} = \sum_{j=1}^{i} V_j^{(k)} (V_j^{(k)H} B R^{-1}) \tag{17}$$

Eventually, the desired matrix $K$ is the limit of the matrices $K_i^{(k)}$ for $k, i \to \infty$. This consideration motivates Algorithm 6, which is best understood as a version of LRNM with an inner loop (Steps 4 and 5), consisting of interlaced sequences based on Step 3 in Algorithm 3 and the partial sums given by the right-hand term in (17). An analogous algorithm based on the real version of LRCF-ADI (Algorithm 4) can be derived in the same way.

---

**Algorithm 6.** (Implicit low-rank Cholesky factor Newton method (LRCF-NM-I))

INPUT: $A$, $B$, $C$, $Q$, $R$, $K^{(0)}$ for which $A - BK^{(0)\mathrm{T}}$ is stable (e.g. $K^{(0)} = 0$, if $A$ is stable).

OUTPUT: $K^{(k_{\max})}$, which approximates $K$ given by (4).

FOR $k = 1, 2, \ldots, k_{\max}$

1. Determine (sub-)optimal ADI shift parameters $p_1^{(k)}, p_2^{(k)}, \ldots$ with respect to the matrix $F^{(k)} = A^{\mathrm{T}} - K^{(k-1)} B^{\mathrm{T}}$ (e.g. [73, Algorithm 1]).
2. $G^{(k)} = [C^{\mathrm{T}} \tilde{Q} \;\; K^{(k-1)} \tilde{R}]$.
3. $V_1^{(k)} = \sqrt{-2 \operatorname{Re} p_1^{(k)}} (F^{(k)} + p_1^{(k)} I_n)^{-1} G^{(k)}$.

FOR $i = 2, 3, \ldots, i_{\max}^{(k)}$

4. $V_i^{(k)} = \sqrt{\operatorname{Re} p_i^{(k)} / \operatorname{Re} p_{i-1}^{(k)}} (V_{i-1}^{(k)} - (p_i^{(k)} + \bar{p}_{i-1}^{(k)})(F^{(k)} + p_i^{(k)} I_n)^{-1} V_{i-1}^{(k)})$.
5. $K_i^{(k)} = K_{i-1}^{(k)} + V_i^{(k)} (V_i^{(k)H} B R^{-1})$.

END

6. $K^{(k)} = K_{i_{\max}^{(k)}}^{(k)}$.

END

---

Most remarks on numerical aspects of LRCF-NM made in Section 5 also apply to LRCF-NM-I. Here, we only point out the differences between both methods.

It is easy to see that the computational costs of Algorithms 5 and 6 are identical. However, the memory demand of the latter is smaller. In fact, it is as low as $\mathcal{O}(n)$ in many situations. For example, this is the case when $A$ is a band matrix with bandwidth $\mathcal{O}(1)$, where the shifted systems of linear equations (13) are solved directly. Another such scenario is sparse matrices with $\mathcal{O}(n)$ non-zero entries, where these systems of linear equations are solved by a Lanczos- or Arnoldi-based Krylov subspace method combined with a preconditioner, whose memory demand is $\mathcal{O}(n)$. In such cases, an improvement by one order of magnitude compared with standard methods is achieved. Note further that, unlike LRCF-NM, the application of LRCF-NM-I can still make sense when no accurate low-rank approximations to the CALE solutions exist at all.



DOI: 10.1002/nla

The essential drawback of LRCF-NM-I is that the safe stopping criterion (15) cannot be applied, because $Z^{(k)}$ is not stored. That means one has to rely on criterion (16). Likewise, (14) must be used as stopping criterion in the inner loop, since computing the CALE residual would require forming the iterates $Z_i^{(k)}$.

Basically, the same concept of a Newton–Kleinman-based 'feedback iteration' is used for the numerical solution of the LQR problem in [17], although the method suggested there differs from LRCF-NM-I in the way it is implemented. The formulation of Newton's method in [17] is based on the difference of two consecutive Newton steps from Algorithm 2. With the notation employed here, this leads to

$$(A^{\mathrm{T}}-K^{(k-1)}B^{\mathrm{T}})N^{(k-1)}+N^{(k-1)}(A-BK^{(k-1)^{\mathrm{T}}})=N^{(k-2)}BR^{-1}B^{\mathrm{T}}N^{(k-2)^{\mathrm{T}}},\quad k=2,\ldots \tag{18}$$

where $X^{(k)}=X^{(k-1)}+N^{(k-1)}$. The advantage of this formulation is that the constant term $C^{\mathrm{T}}QC$ disappears starting with the 2nd Newton step—in the first step, $X^{(1)}$ has to be computed as in Algorithm 2 or needs to be known—so that the right-hand side of the Lyapunov equation has a lower rank than the Lyapunov equation in (9). Unfortunately, numerical experience shows that this iteration is less robust with respect to the accuracy of the computed solutions of the Lyapunov equations. Often, the limiting accuracy of this version of the Newton iteration is inferior to the accuracy obtained by Algorithm 2. Also note that in [17], (18) is only used for deriving an iteration directly defined for the feedback matrices similar to (17), while the computation of approximate low-rank solutions to CALEs or CAREs, which are of interest in many other applications besides the LQR problem, is not discussed there at all.

## 7. NUMERICAL EXPERIMENTS

In this section we report the results of numerical experiments with LRCF-NM and LRCF-NM-I applied to two large-scale test examples. These experiments were conducted at the Department of Mathematics and Statistics of the University of Calgary. The computations were performed using MATLAB 5.3 on a SUN Ultra 450 workstation with IEEE double precision arithmetics (machine epsilon $\varepsilon_{\mathrm{mach}}=2^{-52}\approx 2.2\cdot 10^{-16}$). In our tests, ADI shift parameters are computed by the heuristic algorithm proposed in [73, Algorithm 1].

The following test example is used to show the performance of LRCF-NM. That means, our goal is to compute a LRCF $Z=Z^{(k_{\max})}$, such that $ZZ^H$ approximates the solution of (5).

*Example 1*

Here, the dynamical system (1) arises from the model of a cooling process, which is part of the production of steel rails; see [82, 85]. The cooling process is described by an instationary two-dimensional heat equation, which is semidiscretized by the finite element method. The problem dimensions are $n=3113$, $m=6$, and $q=6$. The matrix $A$ has the structure $A=-U_M^{-1}NU_M^{-T}$, where $N$ is the stiffness matrix and $U_M$ is the Cholesky factor of the mass matrix $M$. The factorization of $M$ and the solution of shifted systems of linear equations are realized by sparse direct methods. We consider the following three choices of $R$ and $Q$:

*Example 1a*: $R=I_6$, $Q=I_6$,
*Example 1b*: $R=10^{-2}I_6$, $Q=I_6$,
*Example 1c*: $R=10^{-4}I_6$, $Q=I_6$.



DOI: 10.1002/nla

Table I. Test results for LRCF-NM applied to Example 1.

| Example | | | 1a | 1b | 1c |
|---|---|---|---|---|---|
| CARE | Iterations steps | | 5 | 8 | 12 |
| | No. of columns of $Z$ | | 540 | 492 | 522 |
| | $r(Z)$ | | $7.3\times10^{-14}$ | $4.2\times10^{-14}$ | $1.0\times10^{-14}$ |
| CALEs | Iterations steps | Minimum | 45 | 40 | 42 |
| | | Average | 45.2 | 41.4 | 44.6 |
| | | Maximum | 46 | 45 | 46 |

In the three test runs, we use the inexpensive stopping criteria (16) for the Newton iteration and (14) for the LRCF-ADI iterations. We require that the latter is fulfilled in 10 consecutive iteration steps, so that the risk of a premature termination of these inner iterations is tiny. To verify the results of LRCF-ADI, we subsequently compute the normalized residual norm

$$r(Z) := \frac{\|C^{\mathrm{T}}QC + A^{\mathrm{T}}ZZ^H + ZZ^HA - ZZ^HBR^{-1}B^{\mathrm{T}}ZZ^H\|_F}{\|C^{\mathrm{T}}QC\|_F} \tag{19}$$

using (15). Table I shows the test results. Besides $r(Z)$, we also display the number of Newton steps (i.e. $k_{\max}$) and the number of columns of the LRCF $Z$. We also provide information about the number of iteration steps in the inner LRCF-ADI iterations.

Obviously, the number of Newton steps strongly depends on the choice of $R$. However, the number of columns in the delivered LRCFs $Z$ is about the same in each test run. Note that this quantity is much smaller than the system order $n$, although the LRCF products $ZZ^H$ are very accurate approximations to the CARE solution $X$ in terms of $r(Z)$. It should be noted that solving the CARE in this example with standard solvers like the CARE solvers in MATLAB and its toolboxes are impossible using current desktop computers.[‡‡]

We now investigate LRCF-NM-I using a second, more theoretical test example of scalable size.

*Example 2*

In this example, the system (1) is related to a three-dimensional convection–diffusion problem

$$\frac{\partial}{\partial\tau}r = \Delta_\xi r - 1000\xi_1\frac{\partial}{\partial\xi_1}r - 100\xi_2\frac{\partial}{\partial\xi_2}r - 10\xi_3\frac{\partial}{\partial\xi_3}r + b(\xi)u(\tau)$$

$$y(\tau) = \int_\Omega c(\xi)r\,\mathrm{d}\xi$$

where $\Omega=(0,1)^3$, $\xi=[\xi_1\ \ \xi_2\ \ \xi_3]^{\mathrm{T}}$, $r=r(\xi,\tau)$, and $r=0$ for $\xi\in\partial\Omega$, i.e. $R$ satisfies homogeneous Dirichlet boundary conditions. $b$ and $c$ are the characteristic functions of the two smaller cubes $(.7,.9)^3$ and $(.1,.3)^3$, respectively, contained in the unit cube $\Omega$. The finite-dimensional model (1) of order $n=n_0^3$ is gained from a finite difference semidiscretization of the problem. Here, $n_0$ is the number of inner grid points in each space dimension. We consider three discretizations

[‡‡]With nowadays 64-Bit MATLAB, this has become feasible, but it still requires a much higher execution time due to the cubic order of complexity of these solvers.



DOI: 10.1002/nla

Table II. Test results for LRCF-NM-I applied to Example 2.

| Example | | | 2a | 2b | 2c |
|---|---|---|---|---|---|
| $(n,m,q)$ | | | (1000,1,1) | (5832,1,1) | (27000,1,1) |
| CARE | Iterations | | 4 | 4 | 3 |
| | $e_K$ | | $1.3\times10^{-8}$ | $8.8\times10^{-8}$ | — |
| CALEs | Iiterations steps | Minimum | 103 | 143 | 96 |
| | | Average | 109.5 | 143 | 96 |
| | | Maximum | 129 | 143 | 96 |

of different mesh size:

*Example 2a*: $n_0=10$, $n=1000$,
*Example 2b*: $n_0=18$, $n=5832$,
*Example 2c*: $n_0=30$, $n=27000$.

The resulting matrices $A$ are sparse and have a relatively large bandwidth. We use QMR with ILU preconditioning to solve the shifted systems of linear equations (13). In this example, we choose $R=10^{-8}$ and $Q=10^{8}$.[§§]

We solve the LQR problem (2) subject to (1) for Example 2 by computing an approximation $K^{(I)}$ to $K$ in (4) via LRCF-NM-I. Again, we use the stopping criteria (16) and (14). To verify the results, we also apply the (explicit) LRCF-NM with stopping criteria based on residual norms, determine the corresponding feedback $K^{(E)}$, and compare the results by evaluating the normalized deviation

$$e_K := \frac{\|K^{(I)}-K^{(E)}\|_F}{\max\{\|K^{(I)}\|_F, \|K^{(E)}\|_F\}}$$

This is only done for Example 2a and b, because Example 2c is too large for an application of LRCF-NM. The results of the three test runs with LRCF-NM-I are shown in Table II.

Note that the underlying PDE has a quite strong convection term, which results in the dominance of the skew-symmetric part of the stiffness matrix $A$, if a rather coarse grid is used in the discretization. Such a dominance has usually a negative effect on the convergence of iterative methods. In this sense, the algebraic properties of $A$ are better for larger values of $n_0$, which explains why in the third test run smaller numbers of Newton and LRCF-ADI steps are needed.

## 8. CONCLUSIONS

In this paper we have presented algorithms for the solution of large-scale Lyapunov equations, Riccati equations, and related linear-quadratic optimal control problems. Basically, the paper contains three contributions. First, the LRCF-ADI iteration for the computation of approximate solutions to Lyapunov equations is described. This method had been proposed before, but in this paper we discuss numerical aspects, which arise in context with this method, in detail. In

[§§] Extreme values for $Q, R$ had to be chosen as otherwise, the convergence of Newton's method (1–2 steps) is too fast to show any effects.



DOI: 10.1002/nla

particular, we have presented a novel modification of LRCF-ADI that delivers real LRCFs and avoids complex computation. Second, we have proposed an efficient variant of Newton's method for Riccati equations, which we call LRCF-NM. The third contribution is an implicit variant of LRCF-NM, which we refer to as LRCF-NM-I. LRCF-ADI has been used as a basis for these two variants of Newton's method. LRCF-NM delivers low-rank approximations to the solution of the Riccati equations. It can be applied to problems that are so large that the dense solution matrix cannot be stored in the computer memory. LRCF-NM-I can be applied to problems of even higher dimension, because this method directly forms approximations to the optimal state feedback, which describes the solution of the linear-quadratic optimal control problem, rather than computing approximations to the solution of the corresponding Riccati equation. Note that LRCF-NM can be combined with any other solver yielding low-rank approximate solutions of Lyapunov equations, but LRCF-NM-I makes explicit use of LRCF-ADI so that the latter cannot be replaced easily in LRCF-NM-I by other Lyapunov solvers, like, e.g. Krylov subspace methods. The results of numerical experiments show the efficiency of LRCF-NM and LRCF-NM-I.

## REFERENCES


1. Penzl T. Lyapack users guide. *Technical Report SFB393/00-33*, Sonderforschungsbereich 393 Numerische Simulation auf massiv parallelen Rechnern, TU Chemnitz, FRG 2000. Available from: http//www.tu-chemnitz.de/sfb393/sfb00pr.html.
2. Abou-Kandil H, Freiling G, Ionescu V, Jank G. *Matrix Riccati Equations in Control and Systems Theory*. Birkhäuser: Basel, Switzerland, 2003.
3. Datta B. *Numerical Methods for Linear Control Systems*. Elsevier Academic Press: San Diego, CA, U.S.A., London, U.K., 2004.
4. Mehrmann V. *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution*. Lecture Notes in Control and Information Sciences, vol. 163. Springer: Berlin, Heidelberg, 1991.
5. Petkov P, Christov N, Konstantinov M. *Computational Methods for Linear Control Systems*. Prentice-Hall: Hertfordshire, U.K., 1991.
6. Sima V. *Algorithms for Linear-Quadratic Optimization, Pure and Applied Mathematics*, vol. 200. Marcel Dekker, Inc.: New York, NY, 1996.
7. Banks H, Kunisch K. The linear regulator problem for parabolic systems. *SIAM Journal on Control and Optimization* 1984; **22**:684–698.
8. Ito K. Finite-dimensional compensators for infinite-dimensional systems via Galerkin-type approximation. *SIAM Journal on Control and Optimization* 1990; **28**:1251–1269.
9. Lasiecka I, Triggiani R. *Differential and Algebraic Riccati Equations with Application to Boundary/Point Control Problems: Continuous Theory and Approximation Theory*. Lecture Notes in Control and Information Sciences, vol. 164. Springer: Berlin, 1991.
10. Morris K. Design of finite-dimensional controllers for infinite-dimensional systems by approximation. *Journal of Mathematical Systems, Estimation, and Control* 1994; **4**:1–30.
11. Freund R. Reduced-order modeling techniques based on Krylov subspaces and their use in circuit simulation. In *Applied and Computational Control, Signals, and Circuits*, Datta B (ed.). vol. 1, Chapter 9. Birkhäuser: Boston, MA, 1999; 435–498.
12. Freund R. Krylov-subspace methods for reduced-order modeling in circuit simulation. *Journal of Computational and Applied Mathematics* 2000; **123**(1–2):395–421.
13. Gawronski M. *Balanced Control of Flexible Structures*. Lecture Notes in Control and Information Sciences, vol. 211. Springer: Berlin, London, U.K., 1996.
14. Papadopoulos P, Laub A, Ianculescu G, Ly J, Kenney C, Pandey P. Optimal control study for the space station solar dynamic power module. *Proceedings of the Conference on Decision and Control CDC-1991*, Brighton, 1991; 2224–2229.
15. Anderson B, Moore J. *Optimal Control–Linear Quadratic Methods*. Prentice-Hall: Englewood Cliffs, NJ, 1990.
16. Lancaster P, Rodman L. *The Algebraic Riccati Equation*. Oxford University Press: Oxford, 1995.




DOI: 10.1002/nla

17. Banks H, Ito K. A numerical algorithm for optimal feedback gains in high dimensional linear quadratic regulator problems. *SIAM Journal on Control and Optimization* 1991; **29**(3):499–515.
18. Kalman R. Contributions to the theory of optimal control. *Boletin Sociedad Matematica Mexicana* 1960; **5**: 102–119.
19. Pinch E. *Optimal Control and the Calculus of Variations*. Oxford University Press: Oxford, U.K., 1993.
20. Durrazi C. Numerical solution of discrete quadratic optimal control problems by Hestenes' method. *Supplemento ai Rendiconti del Circolo Matematico di Palermo*, *Series II* 1999; **58**:133–154.
21. Saberi A, Sannuti P, Chen B. $H_2$ *Optimal Control*. Prentice-Hall: Hertfordshire, U.K., 1995.
22. Green M. Balanced stochastic realization. *Linear Algebra and its Applications* 1988; **98**:211–247.
23. Varga A. On computing high accuracy solutions of a class of Riccati equations. *Control-Theory and Advanced Technology* 1995; **10**(4):2005–2016.
24. Doyle J, Glover K, Khargonekar P, Francis B. State–space solutions to standard $H_2$ and $H_\infty$ control problems. *IEEE Transactions on Automatic Control* 1989; **34**:831–847.
25. Green M, Limebeer D. *Linear Robust Control*. Prentice-Hall: Englewood Cliffs, NJ, 1995.
26. Zhou K, Doyle J, Glover K. *Robust and Optimal Control*. Prentice-Hall: Upper Saddle River, NJ, 1996.
27. Laub A. A Schur method for solving algebraic Riccati equations. *IEEE Transactions on Automatic Control* 1979; **AC-24**:913–921.
28. Byers R. Solving the algebraic Riccati equation with the matrix sign function. *Linear Algebra and its Applications* 1987; **85**:267–279.
29. Gardiner J, Laub A. A generalization of the matrix-sign-function solution for algebraic Riccati equations. *International Journal of Control* 1986; **44**:823–832.
30. Roberts J. Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *International Journal of Control* 1980; **32**:677–687. (Reprint of *Technical Report No. TR-13*, CUED/B-Control, Cambridge University, Engineering Department, 1971.)
31. Ammar G, Benner P, Mehrmann V. A multishift algorithm for the numerical solution of algebraic Riccati equations. *Electronic Transactions on Numerical Analysis* 1993; **1**:33–48.
32. Bunse-Gerstner A, Mehrmann V. A symplectic QR-like algorithm for the solution of the real algebraic Riccati equation. *IEEE Transactions on Automatic Control* 1986; **AC-31**:1104–1113.
33. Byers R. A Hamiltonian QR-algorithm. *SIAM Journal on Scientific and Statistical Computing* 1986; **7**:212–229.
34. Benner P. Computational methods for linear-quadratic optimization. *Supplemento ai Rendiconti del Circolo Matematico di Palermo*, *Series II* 1999; **58**:21–56.
35. Benner P, Faßbender H. An implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem. *Linear Algebra and its Applications* 1997; **263**:75–111.
36. Ferng W, Lin WW, Wang CS. The shift-inverted $J$-Lanczos algorithm for the numerical solutions of large sparse algebraic Riccati equations. *Computers and Mathematics with Applications* 1997; **33**(10):23–40.
37. Hodel A, Poolla K. Heuristic approaches to the solution of very large sparse Lyapunov and algebraic Riccati equations. *Proceedings of the 27th IEEE Conference on Decision and Control*, Austin, TX, 1988; 2217–2222.
38. Jaimoukha I, Kasenally E. Krylov subspace methods for solving large Lyapunov equations. *SIAM Journal on Numerical Analysis* 1994; **31**:227–251.
39. Benner P, Byers R. An exact line search method for solving generalized continuous-time algebraic Riccati equations. *IEEE Transactions on Automatic Control* 1998; **43**(1):101–107.
40. Blackburn T. Solution of the algebraic matrix Riccati equation via Newton-Raphson iteration. *AIAA Journal* 1968; **6**:951–953. See also: *Proceedings of the Joint Automatic Control Conference*, University of Michigan, Ann Arbor, 1968.
41. Kleinman D. On an iterative technique for Riccati equation computations. *IEEE Transactions on Automatic Control* 1968; **AC-13**:114–115.
42. Pandey P. Quasi-Newton methods for solving algebraic Riccati equations. *Proceedings of the American Control Conference*, Chicago, IL, 1992; 654–658.
43. Rosen I, Wang C. A multi-level technique for the approximate solution of operator Lyapunov and algebraic Riccati equations. *SIAM Journal on Numerical Analysis* 1995; **32**(2):514–541.
44. Zečević A, Šiljak D. Solution of Lyapunov and Riccati equations in a multiprocessor environment. *Nonlinear Analysis*, *Theory*, *Methods and Applications* 1997; **30**(5):2815–2825.
45. Hammarling S. Newton's method for solving the algebraic Riccati equation. *NPL Report DITC 12/82*, National Physical Laboratory, Teddington, Middlesex, U.K., 1982.
46. Hammarling S. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA Journal on Numerical Analysis* 1982; **2**:303–323.



DOI: 10.1002/nla

47. Gajić Z, Qureshi M. *Lyapunov Matrix Equation in System Stability and Control*. Mathematics in Science and Engineering. Academic Press: San Diego, CA, 1995.
48. Mullis C, Roberts R. Synthesis of minimum roundoff noise fixed point digital filters. *IEEE Transactions on Circuits and Systems* 1976; **CAS-23**(9):551–562.
49. Moore B. Principal component analysis in linear systems: controllability, observability, and model reduction. *IEEE Transactions on Automatic Control* 1981; **AC-26**:17–32.
50. Glover K. All optimal Hankel-norm approximations of linear multivariable systems and their $L^\infty$ norms. *International Journal on Control* 1984; **39**:1115–1193.
51. Lancaster P. Explicit solutions of linear matrix equation. *SIAM Review* 1970; **12**:544–566.
52. Lancaster P, Tismenetsky M. *The Theory of Matrices* (2nd edn). Academic Press: Orlando, 1985.
53. Bartels R, Stewart G. Solution of the matrix equation $AX+XB=C$: Algorithm 432. *Communications of the ACM* 1972; **15**:820–826.
54. Smith R. Matrix equation $XA+BX=C$. *SIAM Journal on Applied Mathematics* 1968; **16**(1):198–201.
55. Baur U, Benner P. Factorized solution of Lyapunov equations based on hierarchical matrix arithmetic. *Computing* 2006; **78**(3):211–234.
56. Grasedyck L, Hackbusch W, Khoromskij B. Solution of large scale algebraic matrix Riccati equations by use of hierarchical matrices. *Computing* 2003; **70**:121–165.
57. Benner P, Quintana-Ortí E. Solving stable generalized Lyapunov equations with the matrix sign function. *Numerical Algorithms* 1999; **20**(1):75–100.
58. Gardiner J, Laub A. Parallel algorithms for algebraic Riccati equations. *International Journal on Control* 1991; **54**(6):1317–1333.
59. Peaceman D, Rachford H. The numerical solution of elliptic and parabolic differential equations. *Journal of the Society for Industrial and Applied Mathematics* 1955; **3**:28–41.
60. Wachspress E. Iterative solution of the Lyapunov matrix equation. *Applied Mathematics Letters* 1988; **107**:87–90.
61. Antoulas A, Sorensen D, Zhou Y. On the decay rate of Hankel singular values and related issues. *Systems and Control Letters* 2002; **46**(5):323–342.
62. Grasedyck L. Existence of a low rank or $H$-matrix approximant to the solution of a Sylvester equation. *Numerical Linear Algebra with Applications* 2004; **11**:371–389.
63. Penzl T. Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. *Systems and Control Letters* 2000; **40**:139–144.
64. Hodel A, Tenison B, Poolla K. Numerical solution of the Lyapunov equation by approximate power iteration. *Linear Algebra and its Applications* 1996; **236**:205–230.
65. Hu D, Reichel L. Krylov-subspace methods for the Sylvester equation. *Linear Algebra and its Applications* 1992; **172**:283–313.
66. Saad Y. Numerical solution of large Lyapunov equation. In *Signal Processing, Scattering, Operator Theory and Numerical Methods*, Kaashoek MA, van Schuppen JH, Ran ACM (eds). Birkhäuser: Basel, 1990; 503–511.
67. Golub G, Van Loan C. *Matrix Computations* (3rd edn). Johns Hopkins University Press: Baltimore, 1996.
68. Gudmundsson T, Laub A. Approximate solution of large sparse Lyapunov equations. *IEEE Transactions on Automatic Control* 1994; **39**:1110–1114.
69. Ellner NS, Wachspress EL. Alternating direction implicit iteration for systems with complex spectra. *SIAM Journal on Numerical Analysis* 1991; **28**(3):859–870.
70. Starke G. Rationale Minimierungsprobleme in der komplexen Ebene im Zusammenhang mit der Bestimmmung optimaler ADI-Parameter. Dissertation, Fakultät für Mathematik, Universität Karlsruhe, Germany, 1993 (in German).
71. Wachspress E. The ADI model problem 1995. Available from the author.
72. Li JR, Wang F, White J. An efficient Lyapunov equation-based approach for generating reduced-order models of interconnect. *Proceedings of the 36th IEEE/ACM Design Automation Conference*, New Orleans, LA, 1999.
73. Penzl T. A cyclic low rank Smith method for large sparse Lyapunov equations. *SIAM Journal on Scientific Computing* 2000; **21**(4):1401–1418.
74. Li JR, White J. Low rank solution of Lyapunov equations. *SIAM Journal on Matrix Analysis and Applications* 2002; **24**(1):260–280.
75. Davis P. *Circulant Matrices*. Wiley: New York, NY, 1979.
76. Duff I, Erisman A, Reid J. *Direct Methods for Sparse Matrices*. Clarendon Press: Oxford, U.K., 1989.
77. Cuthill E. Several strategies for reducing the bandwidth of matrices. In *Sparse Matrices and their Applications*, Rose D, Willoughby R (eds). Plenum Press: New York, NY, 1972.



DOI: 10.1002/nla

78. Saak J. Effiziente numerische Lösung eines Optimalsteuerungsproblems für die Abkühlung von Stahlprofilen. Diplomarbeit, Fachbereich 3/Mathematik und Informatik, Universität Bremen, Bremen, September 2003.
79. Hackbusch W. *Multigrid Methods and Applications*. Springer: Berlin, Heidelberg, FRG, 1985.
80. Stüben K. Algebraic multigrid (AMG): an introduction with applications. In *Multigrid*, Trottenberg U, Osterlee C, Schüller A (eds). Academic Press: New York, 1999.
81. Calvetti D, Reichel L. Application of ADI iterative methods to the restoration of noisy images. *SIAM Journal on Matrix Analysis and Applications* 1996; **17**:165–186.
82. Penzl T. Algorithms for model reduction of large dynamical systems. *Linear Algebra with Applications* 2006; **415**(2–3):322–343. (Reprint of *Technical Report SFB393/99-40*, TU Chemnitz, 1999.)
83. Hochbruck M, Starke G. Preconditioned Krylov subspace methods for Lyapunov matrix equations. *SIAM Journal on Matrix Analysis and Applications* 1995; **16**(1):156–171.
84. Simoncini V. A new iterative method for solving large-scale Lyapunov matrix equations. *SIAM Journal on Scientific Computing* 2007; **29**:1268–1288.
85. Tröltzsch F, Unger A. Fast solution of optimal control problems in the selective cooling of steel. *Zeitschrift fur Angewandte Mathematik und Mechanik* 2001; **81**:447–456.



DOI: 10.1002/nla