Technische Universität Chemnitz

Sonderforschungsbereich 393

Numerische Simulation auf massiv parallelen Rechnern

Peter Benner Enrique S. Quintana-Ortí

Solving Stable Generalized Lyapunov Equations with the Matrix Sign Function

Preprint SFB393/97-23

Preprint-Reihe des Chemnitzer SFB 393

SFB393/97-23 **October 1997**

Solving Stable Generalized Lyapunov Equations with the Matrix Sign Function*

Peter Benner† Enrique S. Quintana-Ortí‡

Abstract

We investigate the numerical solution of the stable generalized Lyapunov equation via the sign function method. This approach has already been proposed to solve standard Lyapunov equations in several publications. The extension to the generalized case is straightforward. We consider some modifications and discuss how to solve generalized Lyapunov equations with semidefinite constant term for the Cholesky factor. The basic computational tools of the method are basic linear algebra operations that can be implemented efficiently on modern computer architectures and in particular on parallel computers. Hence, a considerable speed-up as compared to the Bartels-Stewart and Hammarling's methods is to be expected. We compare the algorithms by performing a variety of numerical tests.

Key words: Generalized Lyapunov matrix equations, mathematical software, matrix sign function, condition estimation.
AMS(MOS) subject classifications: 65F10, 93B40, 93B51.

1 Introduction

The *generalized Lyapunov equation*

$$0 = Q + A^T X E + E^T X A, \tag{1.1}$$

where $A, E, X, Q \in \mathbb{R}^{n\times n}$, $Q = Q^T$, and $X = X^T$ is the sought-after solution, plays a fundamental role in control theory and the stability analysis of linear systems; see, e.g., [2, 28, 31, 35] and the references given therein. (Note that everything in this paper also

*The work on this paper was partially supported by the *Sonderforschungsbereich 393 "Numerische Simulation auf massiv parallelen Rechnern"* of the *Technische Universität Chemnitz,* Germany, and by the *Deutsche Forschungsgemeinschaft,* research grant *Me 790/7-1 "Singuläre Steuerungsprobleme"*. Enrique S. Quintana-Ortí was partially supported by the CICYT project No. TIC96-1062-C03-C03.

†Zentrum für Technomathematik, Fachbereich 3/Mathematik und Informatik, Universität Bremen, D–28334 Bremen, Germany. E-mail: benner@numerik.uni-bremen.de

‡Departamento de Informática, Universidad Jaime I, Campus Penyeta Roja, 12.071–Castellón, Spain. E-mail: quintana@inf.uji.es

holds for the complex case, i.e., $A, E, X, Q \in \mathbb{C}^{n\times n}$.) In linear control problems governed by first-order ordinary differential equations (ODE), usually $E = I_n$, where I_n denotes the identity of order n. The case $E \neq I_n$ appears if the control problem is governed by a second-order ODE (e.g., [12]) or a descriptor system (e.g., [28]), or if the underlying first-order ODE comes from the finite-element discretization of a partial differential equation (PDE) (e.g., [7, 33]).

Here, we assume that E is nonsingular and hence, $A-\lambda E$ is a regular matrix pencil, i.e., $\det(A-\mu E) \not\equiv 0$ for $\mu \in \mathbb{C}$. Additionally, we assume $\lambda_i + \lambda_j \neq 0$ for all $\lambda_i, \lambda_j \in \sigma(A, E)$, where

$$\sigma(A,E) \;:=\; \{\lambda \in \mathbb{C} \cup \{\infty\} \;:\; \lambda = \frac{\alpha}{\beta},\ \det(\beta A - \alpha E) = 0,\ \textit{with}\ \lambda = \infty \text{ if } \beta = 0\}$$

denotes the generalized spectrum of a regular matrix pencil. These assumptions guarantee (and are necessary) that (1.1) has a unique solution (see, e.g., [26, 30], and the references given therein). The nonsingularity of E implies that all eigenvalues of the matrix pencil $A - \lambda E$ are finite while the condition $\lambda_i + \lambda_j \neq 0$ implies that $\lambda_j \neq 0$ for all $\lambda_j \in \sigma(A, E)$ and hence the nonsingularity of A. Note that the roles of A and E can be swapped as they appear in a symmetric way in (1.1).

The matrix pencil $A-\lambda E$ is called *stable* if all its eigenvalues are contained in the open left half plane, denoted by $\sigma(A, E) \subset \mathbb{C}^-$. This property will be assumed throughout this paper and the associated Lyapunov equation will be called *stable* Lyapunov equation. We will provide reasons why the proposed sign function approach is not feasible for matrix pencils $A-\lambda E$ having eigenvalues on or on both sides of the imaginary axis. But note that everything derived in this paper also holds if $A-\lambda E$ is *antistable*, i.e., has all its eigenvalues in the open right half plane. Either one (stable/antistable) of these two assumptions guarantees that $\lambda_i + \lambda_j \neq 0$ for all $\lambda_i, \lambda_j \in \sigma(A, E)$.

Furthermore, if Q is positive/negative (semi-)definite, then the solution X of (1.1) is also positive/negative (semi-)definite. See, e.g., [26, 30] and the references given therein. We will call a generalized Lyapunov equation *(semi-)definite* if $A-\lambda E$ is stable (or antistable) and the constant term Q is (semi-)definite.

As E and A are both invertible, (1.1) is equivalent to either one of the standard Lyapunov equations

$$\begin{aligned} 0 &= \tilde{Q} + \tilde{A}^T X + X\tilde{A}, \qquad \tilde{A} := AE^{-1}, \quad \tilde{Q} := E^{-T}QE^{-1}, \\ 0 &= \hat{Q} + \hat{A}^T X + X\hat{A}, \qquad \hat{A} := EA^{-1}, \quad \hat{Q} := A^{-T}QA^{-1}. \end{aligned}$$

But if any of these transformations is performed, the subsequent solution of the standard Lyapunov equations is affected by the errors made which are essentially determined by the condition numbers of E or A, respectively. Also, the condition number of $\tilde{A}$ or $\hat{A}$ can be significantly worse than that of A which may affect the iterative schemes employed in our algorithms. Moreover, in many of the above-mentioned applications, E is a sparse matrix while its inverse may be full. As we will see in Section 2, the additional cost for our algorithm caused by using the generalized form (1.1) rather than the standard form given above basically comes from matrix multiplications with E, this additional cost is negligible if E has only $O(n)$ nonzero entries.

Numerical solution methods for generalized Lyapunov equations are studied in [14, 15, 30]. The methods investigated there are generalizations of the Bartels-Stewart method [5] and Hammarling's algorithm [18, 36] introduced for standard Lyapunov equations ($E = I_n$). The initial step in all these methods is the application of the QZ algorithm (see, e.g., [16] and the references given therein) to the matrix pencil $A - \lambda E$. This is followed by a back substitution process. Note that Hammarling's algorithm is only applicable for (semi-)definite Lyapunov equations.

The condition of the Lyapunov equation (1.1) is given by the condition number of the *Lyapunov operator* (see, e.g., [14, 15, 30])

$$\Omega \;:\; \mathcal{S}_n \;\longrightarrow\; \mathcal{S}_n \;:\; Z \;\longrightarrow\; A^T Z E + E^T Z A,$$

where $\mathcal{S}_n$ denotes the set of symmetric matrices in $\mathbb{R}^{n \times n}$. This condition number can easily be derived observing that (1.1) is equivalent to the linear system

$$\Big((E^T \otimes A^T) + (A^T \otimes E^T) \Big) \operatorname{vec}(X) \;=\; -\operatorname{vec}(Q).$$

Here, $\otimes$ denotes the *Kronecker product* (see, e.g., [26]) and vec (.) denotes the operation of stacking the columns of an $n \times m$ matrix into an $n \cdot m$ vector. Consequently, (1.1) has a unique solution if and only if the matrix representation of Ω, given by $W := (E^T \otimes A^T) + (A^T \otimes E^T)$, is nonsingular which in turn is guaranteed by the above assumptions on the coefficients of (1.1). The condition number of the Lyapunov operator is therefore the standard condition number for linear systems, given by

$$\operatorname{cond}_2(\Omega) \;:=\; \operatorname{cond}_2(W) \;=\; \|W\|_2 \|W^{-1}\|_2. \tag{1.2}$$

The *separation* of the Lyapunov operator is given by

$$\operatorname{sep}_F(\Omega) \;:=\; \min_{\|Z\|_F = 1} \|A^T Z E \;+\; E^T Z A\|_F \tag{1.3}$$

and it can be shown that $\operatorname{sep}_F(\Omega) \;=\; 1/\|W^{-1}\|_2$ (see, e.g., [30]). Hence,

$$\operatorname{cond}_2(\Omega) \;=\; \frac{\|W\|_2}{\operatorname{sep}_F(\Omega)} \;\approx\; \frac{2\|A\|_2 \|E\|_2}{\operatorname{sep}_F(\Omega)}.$$

The condition number and separation of the Lyapunov operator will be used in Section 5 to judge the quality of the approximate solutions computed by the tested numerical methods.

In Section 2 we will show how to solve the generalized Lyapunov equation (1.1) using the sign function iteration. This part is a specialization and simplification of the method derived in [13] for the numerical solution of generalized algebraic Riccati equations using that (1.1) is a special case of such an equation. The application to semidefinite Lyapunov equations will be analyzed in Section 3. An algorithmic presentation of the proposed methods and implementation details will be discussed in Section 4. The Lyapunov solvers based on the sign function iteration will then be compared to the Bartels-Stewart algorithm and Hammarling's algorithm with respect to accuracy of computed solutions and computation time in Section 5.

2 The Matrix Sign Function and Lyapunov Equations

The sign function method was first introduced in 1971 by Roberts [32] to solve algebraic Riccati equations of the form

$$0 \;=\; Q + A^T X + XA - XGX \tag{2.4}$$

where $A, G, Q, X \in \mathbb{R}^{n\times n}$, $G = G^T$, $Q = Q^T$, and $X = X^T$ is the unknown solution matrix. Roberts also shows how to solve stable Sylvester and Lyapunov equations via the matrix sign function. These ideas will be used and extended in the sequel.

The *matrix sign function* of a matrix $Z \in \mathbb{R}^{n\times n}$ can be defined as follows. Let $\sigma(Z) \cap \imath\mathbb{R} = \emptyset$ ($\imath := \sqrt{-1}$) and denote the Jordan decomposition of Z by

$$Z = S \begin{bmatrix} J^- & 0 \\ 0 & J^+ \end{bmatrix} S^{-1},$$

where the Jordan blocks corresponding to the, say, k eigenvalues in the open left half plane are collected in J^- and the Jordan blocks corresponding to the remaining $n-k$ eigenvalues in the open right half plane are collected in J^+. Then

$$\operatorname{sign}(Z) \;:=\; S \begin{bmatrix} -I_k & 0 \\ 0 & I_{n-k} \end{bmatrix} S^{-1}. \tag{2.5}$$

Note that $\operatorname{sign}(Z)$ is unique and independent of the order of the eigenvalues in the Jordan decomposition of Z (see, e.g., [25, Section 22.1]). Many other equivalent definitions for $\operatorname{sign}(Z)$ can be given; see, e.g., the recent survey papers [24, 23].

It is well-known (see, e.g., [32, 23]) that $(I_n - \operatorname{sign}(Z))/2$ defines the skew projection onto the stable Z-invariant subspace parallel to the antistable Z-invariant subspace whereas $(I_n + \operatorname{sign}(Z))/2$ defines the skew projection onto the antistable Z-invariant subspace parallel to the stable Z-invariant subspace.

The sign function can be computed via the Newton iteration for the equation $Z^2 = I$ where the starting point is chosen as Z, i.e.,

$$Z_0 \;\leftarrow\; Z, \qquad Z_{k+1} \;\leftarrow\; \left(Z_k + Z_k^{-1}\right)/2, \quad \text{for } k = 0, 1, 2, \ldots. \tag{2.6}$$

It is shown in [32] that $\operatorname{sign}(Z) = \lim_{k\to\infty} Z_k$.

Although the convergence of the Newton iteration is globally quadratic, the initial convergence may be slow. There have been several proposals to accelerate this iteration by scaling, e.g., in [8], the *determinantal scaling*

$$Z_k \leftarrow \frac{1}{|\det(Z_k)|^{1/n}} Z_k$$

is introduced. Other scalings are given in [32, 4, 19, 22] and a comparison of these strategies for accelerating the convergence of the Newton iteration can be found in [3]. Several other schemes have been developed for computing the sign function of a matrix. For a summary see [23].

A generalization of the matrix sign function method to a matrix pencil $Z - \lambda Y$ was given by Gardiner and Laub [13] in case Z and Y are nonsingular. They consider the iteration

$$\begin{array}{rcl} Z_0 & \leftarrow & Z, \\ c_k & \leftarrow & \left(\dfrac{|\det(Z_k)|}{|\det(Y)|}\right)^{\frac{1}{n}} \\ Z_{k+1} & \leftarrow & \dfrac{1}{2c_k}(Z_k + c_k^2 Y Z_k^{-1} Y). \end{array} \qquad \text{for } k = 0, 1, 2, \ldots \tag{2.7}$$

It is easy to see that this iteration is equivalent to computing the sign function of the matrix $Y^{-1}Z$ via the standard Newton iteration as given in (2.6). If $Z_\infty := \lim_{j\to\infty} Z_j$, then $Z_\infty - Y$ defines the skew projection onto the stable right deflating subspace of $Z - \lambda Y$ parallel to the antistable deflating subspace and $Z_\infty + Y$ defines the skew projection onto the antistable right deflating subspace of $Z - \lambda Y$ parallel to the stable deflating subspace. In [13] the iteration (2.7) is used to compute the stabilizing solution of the generalized continuous-time algebraic Riccati equation

$$0 \;=\; Q + A^T X E + E^T X A - E^T X G X E, \tag{2.8}$$

where A, G, Q, X are as in (2.4) and $E \in \mathbb{R}^{n\times n}$ is nonsingular. Here, X is *stabilizing* in the sense that $\sigma(A - GXE, E) \subset \mathbb{C}^-$. It is known [13, 25, 28] that if such a stabilizing solution exists, then it is unique and the columns of $[\, I_n,\ E^T X\,]^T$ span the stable deflating subspace of the matrix pencil

$$H - \lambda K \;:=\; \left[\begin{array}{cc} A & G \\ Q & -A^T \end{array}\right] - \lambda \left[\begin{array}{cc} E & 0 \\ 0 & E^T \end{array}\right]. \tag{2.9}$$

Therefore, (2.8) can be solved by applying (2.7) to $H - \lambda K$ and then forming the resulting projector $H_\infty - K$ onto the stable deflating subspace of $H - \lambda K$. A basis of this subspace is then given by the range of that projector. If this basis is given by the columns of $[\, U^T,\ V^T\,]^T$, $U, V \in \mathbb{R}^{n\times n}$, then U is invertible and the solution of (2.8) is given by the solution of the linear matrix equation

$$XEU = -V. \tag{2.10}$$

It can also be shown that it is not necessary to compute explicitly a basis of the stable deflating subspace of $H - \lambda K$ but X can be obtained from the overdetermined but consistent set of linear equations

$$(H_\infty + K) \left[\begin{array}{c} I_n \\ XE \end{array}\right] \;=\; 0. \tag{2.11}$$

As the columns of $[\, I_n,\ E^T X\,]^T$ span the range of $H_\infty - K$, they are contained in the null space of any projector onto the antistable deflating subspace of $H - \lambda K$ from (2.9). As

$H_\infty + K$ defines such a projector, equation (2.11) follows immediately. If

$$H_\infty \;:=\; \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix},$$

then (2.11) is equivalent to

$$\begin{bmatrix} W_{12} \\ W_{22} + E^T \end{bmatrix} XE = -\begin{bmatrix} W_{11} + E \\ W_{21} \end{bmatrix}. \tag{2.12}$$

Using different motivations, (2.12) is derived in [32] for $E = I_n$ and for $E \neq I_n$ in [13]. We also have the following result:

Proposition 2.1 *The generalized continuous-time algebraic Riccati equation (2.8) has a stabilizing solution if and only if $H - \lambda K$ has no eigenvalues on the imaginary axis and*

$$\begin{bmatrix} W_{12} \\ W_{22} + E^T \end{bmatrix}$$

has full column rank.

Proof By the equivalence transformation

$$\begin{bmatrix} E^{-1} & 0 \\ 0 & I_n \end{bmatrix} (H - \lambda K) \begin{bmatrix} I_n & 0 \\ 0 & E^{-T} \end{bmatrix} \;=:\; \tilde{H} - \lambda I_{2n},$$

the generalized equation (2.8) is equivalent to

$$0 \;=\; Q + \tilde{A}^T \tilde{X} + \tilde{X}\tilde{A} - \tilde{X}\tilde{G}\tilde{X}, \tag{2.13}$$

where $\tilde{A} = E^{-1}A$, $\tilde{G} = E^{-1}GE^{-T}$, and $\tilde{X} = E^T XE$. Equation (2.13) has a stabilizing solution $\tilde{X}$ if and only if (2.8) has a stabilizing solution X. Furthermore, $\sigma(H, K) = \sigma(\tilde{H})$. Defining

$$\begin{bmatrix} \tilde{W}_{11} & \tilde{W}_{12} \\ \tilde{W}_{21} & \tilde{W}_{22} \end{bmatrix} \;:=\; \operatorname{sign}\left(\tilde{H}\right),$$

it follows (see [13])

$$\begin{bmatrix} E & 0 \\ 0 & I_n \end{bmatrix} \operatorname{sign}\left(\tilde{H}\right) \begin{bmatrix} I_n & 0 \\ 0 & E^T \end{bmatrix} \;=\; H_\infty.$$

Applying Theorem 22.4.1 of [25] to $\tilde{H}$ we obtain that

$$\operatorname{rank}\left(\begin{bmatrix} \tilde{W}_{12} \\ \tilde{W}_{22} + I_n \end{bmatrix}\right) \;=\; n$$

if and only if (2.13) has a stabilizing solution and $\tilde{H}$ has no eigenvalues on the imaginary axis, i.e., by the above considerations, if and only if (2.8) has a stabilizing solution and

$H - \lambda K$ has no eigenvalues on the imaginary axis. Now the theorem follows by noting that E is nonsingular and hence

$$\begin{aligned} n \;=\; \operatorname{rank}\left(\begin{bmatrix} \tilde{W}_{12} \\ \tilde{W}_{22} + I_n \end{bmatrix}\right) &= \operatorname{rank}\left(\begin{bmatrix} E & 0 \\ 0 & I_n \end{bmatrix}\begin{bmatrix} \tilde{W}_{12} \\ \tilde{W}_{22} + I_n \end{bmatrix} E^T\right) \\ &= \operatorname{rank}\left(\begin{bmatrix} W_{12} \\ W_{22} + E^T \end{bmatrix}\right). \end{aligned}$$

□

The Lyapunov equation (1.1) is a special case of the generalized continuous-time algebraic Riccati equation (2.8). This implies that one can solve (1.1) by means of the sign function method applied to the matrix pencil in (2.9) which then takes the form

$$H - \lambda K \;=\; \begin{bmatrix} A & 0 \\ Q & -A^T \end{bmatrix} - \lambda \begin{bmatrix} E & 0 \\ 0 & E^T \end{bmatrix}. \tag{2.14}$$

Under the given assumptions, this matrix pencil is regular and $H - \lambda K$ has an n-dimensional stable deflating subspace. If this subspace is spanned by the columns of $[\, U^T,\ V^T \,]^T$, then U is nonsingular and $X = -VU^{-1}E^{-1}$ is the solution of the generalized Lyapunov equation (1.1). Note that X can be computed via (2.12) rather than by forming the deflating subspace explicitly. Theorem 2.1 implies that for Lyapunov equations where $\sigma(A, E)$ is not contained in either the open left or open right half plane, the linear system (2.12) is not consistent as it does not have full column rank and the Lyapunov equation can in general not be solved via this approach. In order to solve (1.1) via the sign function applied to $H - \lambda K$ it would be necessary to find an expression for the projector onto the deflating subspace corresponding to $\sigma(A, E) \cap \sigma(H, K)$ in terms of the sign function. At this writing, we are not aware of such an expression.

The solution of (1.1) in case $E = I_n$ and A stable by means of the sign function method was already suggested by Roberts [32] using the identity

$$\frac{1}{2}\left(I + \operatorname{sign}\left(\begin{bmatrix} A & 0 \\ Q & -A^T \end{bmatrix}\right)\right) \;=\; \begin{bmatrix} 0 & 0 \\ X & I \end{bmatrix} \tag{2.15}$$

(Note that our notation slightly differs from [32].) This shows that it is neither necessary to compute $[\, U^T,\ V^T \,]^T$ explicitly nor to solve (2.12). The solution of the Lyapunov equation can be read off from (2.15) directly.

The computation of the sign function is rather expensive compared to solving the Lyapunov equation by the Bartels-Stewart algorithm [5] or its generalizations [14, 15, 30]. We will therefore show how to simplify these computations.

Let us first consider the solution of the Lyapunov equation (1.1) in case $E = I_n$ using the Newton iteration (2.6) to compute the sign function of H as given in (2.14). Roberts shows in [32] that the Newton iteration boils down to

$$\begin{aligned} A_0 &\leftarrow A, & A_{k+1} &\leftarrow \frac{1}{2}\left(A_k + A_k^{-1}\right), \\ Q_0 &\leftarrow Q, & Q_{k+1} &\leftarrow \frac{1}{2}\left(Q_k + A_k^{-T} Q_k A_k^{-1}\right), \end{aligned} \qquad \text{for } k = 0, 1, 2, \ldots \tag{2.16}$$

and from (2.15) it is easy to see that $X = (\lim_{k\to\infty} Q_k)/2$. The same procedure is re-derived a couple of times in the literature; see [6, 21, 11].

If $E \neq I_n$ and $A - \lambda E$ is stable we can use the generalized Newton iteration (2.7) applied to the matrix pencil $H - \lambda K$ to obtain

$$\begin{array}{lll} A_0 \leftarrow A, & A_{k+1} \leftarrow \frac{1}{2}\left(A_k + E A_k^{-1} E\right), & \\ & & \text{for } k = 0, 1, 2, \ldots \\ Q_0 \leftarrow Q, & Q_{k+1} \leftarrow \frac{1}{2}\left(Q_k + E^T A_k^{-T} Q_k A_k^{-1} E\right), & \end{array} \tag{2.17}$$

and

$$X = \frac{1}{2} E^{-T} \left(\lim_{k\to\infty} Q_k\right) E^{-1}.$$

Although (2.17) is a straightforward generalization of (2.16), we were not able to locate a reference for this approach to solve generalized Lyapunov equations. In the following, we will denote the limits in iterations (2.16) and (2.17) by $A_\infty := \lim_{k\to\infty} A_k$ and $Q_\infty := \lim_{k\to\infty} Q_k$, respectively. Note that for (2.16), $A_\infty = -I_n$ while for (2.17) we have $A_\infty = -E$. These relations can be used to design simple stopping criteria; see Section 4.

The iterations (2.16) and (2.17) save a lot of workspace and computational cost compared to applying the (generalized) Newton iteration to H (or $H - \lambda K$). A complete account of the computational cost of the resulting algorithm and a comparison to the Bartels-Stewart method as presented in [15, 30] are given in Section 4. Roughly speaking, we will see there that ten steps of (2.17) are about as expensive as solving (1.1) by the generalized Bartels-Stewart algorithm. It can be observed that convergence of (2.17) often requires 7–10 iterations such that the computational cost of both methods is usually similar. Note also that both methods require the same amount of work space.

Some of the computational work can be saved by computing an initial QR or QL factorization of E, $E = UR_E$. We may then set

$$A_0 - \lambda E_0 \;:=\; U^T(A - \lambda E) \;=\; U^T A - \lambda R_E$$

and apply the iteration (2.17) to $A_0 - \lambda E_0$. This only requires an additional update of the computed solution X by $X \leftarrow UXU^T$. With this transformation, the matrix multiplications with E in (2.17) require only multiplication by a triangular matrix and the linear systems to be solved in order to obtain X from Q_∞ are triangular systems. Also note that in case determinantal scaling is used, $\det(E)$ is needed which also requires a factorization of E so that an initial QR or QL factorization does not cause (significant) extra work. Moreover, quite frequently E is the result of an a priori transformation of the underlying system. It is then often the case that E is already upper or lower triangular; see, e.g., [7, 28, 29, 37].

A complete description of the resulting algorithms and their computational cost will be given in Section 4.

The following remark does not replace a thorough error analysis but provides an explanation why our method usually provides Lyapunov solutions with an accuracy as predicted by the condition number of the corresponding Lyapunov operator.

Remark 2.2 *In [9] it is shown that invariant subspaces computed via the sign function iteration are essentially as accurate as those computed by the QR algorithm, provided the sign function of a matrix can be computed with sufficient accuracy. The same arguments also show that a deflating subspace of a matrix pencil obtained via the generalized sign function method will be essentially as accurate as computed by the QZ algorithm. In Section 2 we saw that XE is determined by the stable deflating subspace of $H - \lambda K$. Moreover, the condition number of this subspace is in our case given by $1/\mathrm{sep}_F(\Omega)$.*

Remark 2.2 shows that the accuracy to which X is computed by our algorithm is basically determined by $\mathrm{sep}_F(\Omega)$ and the condition of E. If E is well-conditioned, this is basically equivalent to $\mathrm{cond}_2(\Omega)$. Moreover, in some applications, $E^T XE = Q_\infty/2$ or $XE = E^{-T}Q_\infty/2$ are required such that the final inversions of E can (partially) be circumvented. For instance, the observability Gramian of a linear time-invariant system in generalized state-space form,

$$\begin{aligned} E\dot{x}(t) &= Ax(t) + Bu(t), \qquad t \geq t_0, \qquad x(t_0) = x_0, \\ y(t) &= Cx(t), \qquad t \geq t_0, \end{aligned}$$

is given by the solution Y of the standard Lyapunov equation

$$(E^{-1}A)^T Y + Y(E^{-1}A) + CC^T \;=\; 0. \tag{2.18}$$

Equation (2.18) is equivalent to (1.1) with $X = E^{-T}YE^{-1}$ and $Q = CC^T$. Solving (1.1) using (2.17), the observability Gramian is given by $Y = E^T XE = Q_\infty/2$. The matrix $XE = E^{-T}Q_\infty/2$ is required, e.g., when computing J–inner–outer factorizations of rational matrices by the algorithm given in [38].

3 The Semidefinite Lyapunov Equation

The semidefinite generalized Lyapunov equation can be written as

$$A^T XE + E^T XA \pm C^T C \;=\; 0, \tag{3.19}$$

where $C \in \mathbb{R}^{p\times n}$. In this case, the solution matrix X can also be written in factored form as $X = \pm Y^T Y$ as X is semidefinite. In the following we will only consider the "+"-case, the other case follows analogously. Also, if $A - \lambda E$ is antistable, everything remains the same, just if the constant term is positive (negative) semidefinite, then X is negative (positive) semidefinite such that $C^T C$ and $Y^T Y$ have opposite signs.

In many applications, the Cholesky factor Y of X is required rather than the solution X itself, e.g., [17, 34]. Hammarling's method [18, 36, 30] computes this factor without forming the product $C^T C$ and the solution X explicitly. The advantage of this approach is that the condition number of X can be up to the square of that of its Cholesky factor Y. Hence, a significant increase in accuracy can be observed using this approach if X is ill-conditioned.

We will see that the method presented in the previous section can be modified to compute the Cholesky factor of X directly similar to Hammarling's algorithm.

Setting $Q = C^T C$, the iteration for Q in (2.17) can be re-written as

$$\begin{aligned} C_0 &\leftarrow C, && (3.20)\\ C_{k+1}^T C_{k+1} &\leftarrow \frac{1}{2}\left(C_k^T C_k + E^T A_k^{-T}(C_k^T C_k)A_k^{-1}E\right) && (3.21)\\ &= \frac{1}{2}\begin{bmatrix} C_k \\ C_k A_k^{-1}E \end{bmatrix}^T \begin{bmatrix} C_k \\ C_k A_k^{-1}E \end{bmatrix}, \quad \text{for } k = 0,1,2,\ldots. \end{aligned}$$

The resulting algorithm would in each step augment the current iterate C_k by the product $C_k A_k^{-1} E$ such that

$$C_{k+1} := \frac{1}{\sqrt{2}}\begin{bmatrix} C_k \\ C_k A_k^{-1}E \end{bmatrix}.$$

The computational cost for the k-th iteration step of such a procedure is $2(2^k p)n^2$, where p is the number of rows of C. This compares to $3n^3$ flops for each iteration step for the Q_k in (2.17). (The product $A_k^{-1}E$ is already formed for the iteration on A_k!)

The above approach requires to double in each iteration step the workspace needed for the iterates C_k. As the rank of the solution X of (3.19) and hence of its Cholesky factor Y can not be predicted by the rank of C, the implementation of Hammarling's algorithm in [30] requires a work array of dimension at least $n \times n$ for C if it is supposed to be overwritten by Y. This suggests to use (3.20) only as long as $2^k p$ is less than $n/2$ which is also the bound for which the original iteration (2.17) becomes cheaper than (3.20). This bound is given by

$$k > \left\lfloor \log_2 \frac{n}{p} \right\rfloor, \tag{3.22}$$

where $\lfloor\, x \,\rfloor$ denotes the integer part of x.

If k has reached the bound given above (which is the case for $k = 0$ if $p > n/2$), we propose to form the augmented matrix $\tilde{C}_{k+1} = [\, C_k^T,\, (C_k A_k^{-1}E)^T\,]^T \in \mathbb{R}^{2s_k \times n}$, where $C_k \in \mathbb{R}^{s_k \times n}$ with $s_0 = p$. Then compute its QR factorization,

$$\tilde{C}_{k+1} = U_{k+1}\tilde{R}_{k+1} = U_{k+1}\begin{bmatrix} R_{k+1} \\ 0 \end{bmatrix}\begin{matrix} \}\, r_{k+1} \\ \}\, 2s_k - r_{k+1} \end{matrix},$$

where $r_{k+1} := \operatorname{rank}(\tilde{C}_{k+1})$. It follows that $C_{k+1}^T C_{k+1} := \tilde{C}_{k+1}^T \tilde{C}_{k+1} = R_{k+1}^T R_{k+1}/2$ and hence we can set $C_{k+1} := R_{k+1}/\sqrt{2}$ and $s_{k+1} := r_{k+1}$. Note that in order to obtain the Cholesky factor of X, a QR factorization of C_{k+1} has to be computed at convergence even if k does not reach the bound in (3.22). In order to determine the rank of $\tilde{C}_{k+1}$ correctly, it may be more reasonable to employ a QR factorization with column pivoting (see, e.g., [16]) or even a rank-revealing QR factorization (see, e.g., [10]). In that case R_{k+1} is obtained as the upper $r_{k+1} \times n$ part of the product of the upper triangular matrix $\tilde{R}_{k+1}$ and a permutation matrix Π_{k+1}, i.e.,

$$\tilde{R}_{k+1}\Pi_{k+1} = \begin{bmatrix} \hat{R}_{k+1} & T_{k+1} \\ 0 & 0 \end{bmatrix}\begin{bmatrix} (\Pi_{k+1})_{11} & (\Pi_{k+1})_{12} \\ (\Pi_{k+1})_{21} & (\Pi_{k+1})_{22} \end{bmatrix} = \begin{bmatrix} R_{k+1} \\ 0 \end{bmatrix}.$$

Another alternative is to compute the QR factorization of the augmented matrix in each iteration step. As the rank of X may be up to n, this approach requires a work space of size $2n \times n$. This approach was outlined for the case $E = I_n$ (without discussing any implementation details) in [27].

The approaches to the solution of semidefinite Lyapunov equations described above can of course be combined with an initial QR or QL factorization of E as proposed at the end of Section 2.

4 Algorithms and Implementation Issues

Before presenting the methods described so far in an algorithmic fashion and discussing their computational cost we make a few remarks on stopping criteria. In Section 2 it was observed that $A_\infty = \lim_{k\to\infty} A_k = -E$; see (2.11). This suggests the stopping criterion

$$\|A_k + E\| \;\leq\; tol \cdot \|E\| \tag{4.23}$$

for a suitable norm and a user-defined tolerance *tol*. There are many suggestions for choosing *tol* in algorithms based on (2.6) where usually stopping criteria like $\|A_{k+1} - A_k\| \leq tol\|A_{k+1}\|$ or $\|A_k^2 - I_n\| \leq tol$ are used. It has been observed (see, e.g., [7]) that often a choice like $tol = c \cdot n \cdot \varepsilon$, where ε denotes the machine precision, and usually the constant c is chosen as 10 or 100, may lead to a stagnation in the iteration because the criterion can not be satisfied, e.g., due to ill-conditioning of the sign function matrix. This can usually be overcome by using $tol = c \cdot n \cdot \sqrt{\varepsilon}$ and performing *one* (or two) additional iteration steps after the stopping criterion is satisfied. Due to the quadratic convergence of the Newton iteration, this is usually enough to reach the attainable accuracy.

For our implementations we choose (4.23) using the 1–norm for ease of computation and $tol = 10 \cdot n \cdot \sqrt{\varepsilon}$ and performed two additional iteration steps after reaching this tolerance. All other stopping criteria suggested for (2.6) are also possible here. Our criterion (4.23) has the advantage of being very easy to check and does not require additional computations (like the ones based on $\|A_k^2 - I\|$) or additional workspace (like the ones based on $\|A_{k+1} - A_k\|$). In case $A - \lambda E$ is antistable, (4.23) has to be modified to

$$\|A_k - E\| \;\leq\; tol \cdot \|E\|. \tag{4.24}$$

If an initial QR or QL factorization is performed, then in the stopping criteria (4.23) and (4.24), E has to be replaced by its triangular factor.

We will start our description by the basic algorithm obtained by the iterative scheme given in (2.17) and determinantal scaling.

Algorithm 4.1 *[*SIGE*]*

Input: $A, E, Q \in \mathbb{R}^{n\times n}$ *with* $Q = Q^T$, $\sigma(A, E) \subset \mathbb{C}^-$.

Output: *Solution* $X \in \mathbb{R}^{n\times n}$ *of (1.1).*

1. Compute $\gamma_E = |det(E)|^{\frac{1}{n}}$ by LU factorization of E.
   $V = A, \qquad X = Q.$

2. `WHILE` $\|V+E\|_1 > tol \cdot \|E\|_1$

   2.1 $A = LUP$ by LU factorization with partial pivoting.
   2.2 $\gamma_A = |\det(A)|^{\frac{1}{n}} = \prod_{k=1}^{n} |u_{kk}|^{\frac{1}{n}}$.
   2.3 $W = P^T U^{-1} L^{-1} E$ by forward and backward substitution.
   2.4 $\gamma = \gamma_A / \gamma_E$.
   2.5 $A = \frac{1}{2}\left(\frac{1}{\gamma} V + \gamma E W\right)$.
   2.6 $X = \frac{1}{2}\left(\frac{1}{\gamma} X + \gamma W^T X W\right)$.
   2.7 $V = A$.

   `END WHILE`

3. $E = LUP$ by LU factorization with partial pivoting.
   $X = \frac{1}{2} L^{-T} U^{-T} P X P^T U^{-1} L^{-1}$ by forward and backward substitution.

`END`

In Step 1, E is not overwritten by its LU decomposition and in the iteration, the original E is used to avoid the introduction of rounding errors resulting from Gaussian elimination. Therefore, the LU factorization has to be repeated in Step 3 unless there is an additional workspace of order n^2 available. As the computational cost of one LU factorization is usually cheap compared to the rest of computations we choose to repeat it in Step 3.

In each iteration step, A is overwritten by its LU decomposition and a workspace of dimension n^2 is thus required to store the current iterate A_k. Another workspace of size n^2 is required to store $A^{-1}E$ in order to re-use this product during the update of Q_k. In Step 2.6, the computation of $W^T X W$ can use V as workspace such that no additional workspace is required. Altogether Algorithm 4.1 requires a workspace of size $5n^2$. This is the same workspace required by the implementation of the generalized Bartels-Stewart method described in [30].

The next algorithm employs an initial QL factorization of E as described at the end of Section 2.

Algorithm 4.2 *[SITR]*
Input: $A, E, Q \in \mathbb{R}^{n\times n}$ *with* $Q = Q^T$, $\sigma(A, E) \subset \mathbb{C}^-$.
Output: *Solution* $X \in \mathbb{R}^{n\times n}$ *of (1.1).*

1. $E = U_E L_E$ by QL factorization.
   $\gamma_E = |det(E)|^{\frac{1}{n}} = \prod_{k=1}^{n} |(l_E)_{kk}|^{\frac{1}{n}}$.
   $A = U_E^T A, \qquad V = A, \qquad X = Q.$

2. `WHILE` $\|V + L_E\|_1 \; > \; tol \cdot \|L_E\|_1$

   2.1 $A \;=\; LUP$ by LU factorization with partial pivoting.

   2.2 $\gamma_A \;=\; |\det(A)|^{\frac{1}{n}} \;=\; \prod_{k=1}^{n} |u_{kk}|^{\frac{1}{n}}$.

   2.3 $W \;=\; P^T U^{-1} L^{-1} L_E$ by forward and backward substitution.

   2.4 $\gamma \;=\; \gamma_A / \gamma_E$.

   2.5 $A \;=\; \frac{1}{2}\left(\frac{1}{\gamma} V + \gamma L_E W\right)$.

   2.6 $X \;=\; \frac{1}{2}\left(\frac{1}{\gamma} X + \gamma W^T X W\right)$.

   2.7 $V \;=\; A$.

   `END WHILE`

3. $X \;=\; \frac{1}{2} L_E^{-T} X L_E^{-1}$ by forward substitution.
   $X \;=\; U_E X U_E^T$.

`END`

Note that the forward substitution process in Step 2.3, expressed by the product $L^{-1}L_E$, yields again a lower triangular matrix which saves about $2n^3/3$ flops as compared to a forward substitution process with a full (or upper triangular) matrix R_E. For this reason we choose the QL factorization in Step 1 rather than a QR factorization. The matrix E is overwritten by its QL decomposition such that the matrix U is kept in factored form.

Since the constant term Q in (1.1) is symmetric, all iterates Q_k in (2.17) are symmetric, too. This symmetry can be exploited in the above algorithms by storing only the upper or lower triangular part of X, thus saving workspace of size $n^2/2$. The equivalence transformation $W^T X W$ can then be computed using only $3n^3$ flops rather than $4n^3$ flops.

Taking into account all these considerations, we obtain the computational costs given in Table 4.1 for the above algorithms (counting only the dominant terms).

	SIGE	SITR
Step 1	$\frac{2}{3}n^3$ flops	$\frac{10}{3}n^3$ flops
Step 2 (one iteration)	$\frac{23}{3}n^3$ flops	$\frac{17}{3}n^3$ flops
Step 3	$\frac{11}{3}n^3$ flops	$\frac{9}{2}n^3$ flops

Table 4.1: Flop counts for SIGE and SITR.

These flop counts are based on the ones given in [16] for the LU and Householder QR/QL decompositions and the exploitation of all the available structure, i.e., symmetry and triangularity.

It can be seen that the overhead in Steps 1 and 3 of Algorithm SITR is higher than for SIGE. This is compensated by the savings during two iteration steps such that usually, i.e., if the iteration requires more than two iterations, Algorithm SITR is slightly cheaper than Algorithm SIGE.

Based on the flop counts given in [30] we can conclude that Algorithm SIGE (SITR) is cheaper than the Bartels-Stewart method for up to 9 (11) iterations required in Step 2. We will see that the savings in computation time are usually higher than predicted by these flop counts due to the efficient implementation of the necessary basic linear algebra tools.

Next, we will present the analogues to Algorithms 4.1 and 4.2 for positive semidefinite Lyapunov equations based on Section 3.

Algorithm 4.3 [SIGS]

Input: $A, E \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{p\times n}$, $\sigma(A, E) \subset \mathbb{C}^-$.

Output: *Cholesky factor* Y *of the solution* $X \in \mathbb{R}^{n\times n}$ *of (1.1).*

1. Compute $\gamma_E = |det(E)|^{\frac{1}{n}}$ by LU factorization of E.
   $V = A, \quad Y = C, \quad k = 0.$

2. `WHILE` $\|V + E\|_1 > tol \cdot \|E\|_1$

   2.1 $k = k + 1.$

   2.2 $A = LUP$ by LU factorization with partial pivoting.

   2.3 $\gamma_A = |\det(A)|^{\frac{1}{n}} = \prod_{k=1}^{n} |u_{kk}|^{\frac{1}{n}}.$

   2.4 $W = P^T U^{-1} L^{-1} E$ by forward and backward substitution.

   2.5 $\gamma = \gamma_A/\gamma_E.$

   2.6 $A = \frac{1}{2}\left(\frac{1}{\gamma}V + \gamma EW\right).$

   2.7 $\gamma = \sqrt{\gamma}.$

   2.8 `IF` $k \leq \left\lfloor \log_2 \frac{n}{p} \right\rfloor$ `THEN`

   $$Y = \frac{1}{\sqrt{2}} \begin{bmatrix} \frac{1}{\gamma}Y \\ \gamma YW \end{bmatrix}.$$

   `ELSE`

   $$\begin{bmatrix} \frac{1}{\gamma}Y \\ \gamma YW \end{bmatrix} = U \begin{bmatrix} Y \\ 0 \end{bmatrix} \text{ by QR factorization.}$$

   $$Y = \frac{1}{\sqrt{2}}Y.$$

   `END IF`

   2.9 $V = A.$

   `END WHILE`

3. $E = LUP$ by LU factorization with partial pivoting.
$Y = \frac{1}{\sqrt{2}} Y P^T U^{-1} L^{-1}$ by forward and backward substitution.

END

As outlined in Section 3, the QR factorization in Step 2.8 can be replaced by a QR factorization with column pivoting or even a rank-revealing QR factorization.

By SITS we denote a version of Algorithm 4.3 employing the initial QL factorization of E as in Algorithm SITR. We avoid reproducing this algorithm here as it is a straightforward combination of Algorithms 4.2 and 4.3.

The computational costs for Algorithms SIGS and SITS are given in Table 4.2 where $r = \text{rank}(X) = \text{rank}(Y)$. The only difference in the flop counts compared to Algorithms SIGE and SITR comes from the different updating of the approximate solution X or its Cholesky factor Y.

	SIGS	SITS
Step 1	$\frac{2}{3}n^3$ flops	$\frac{10}{3}n^3$ flops
Step 2 (kth iteration, $k \leq \lfloor \log_2 \frac{n}{p} \rfloor$)	$\frac{14}{3}n^3 + 2^k pn^2$ flops	$\frac{8}{3}n^3 + 2^k pn^2$ flops
Step 2 (kth iteration, $k > \lfloor \log_2 \frac{n}{p} \rfloor$)	$\frac{24}{3}n^3$ flops	$\frac{18}{3}n^3$ flops
Step 3	$\frac{2}{3}n^3 + 2rn^2$ flops	$3rn^2$ flops

Table 4.2: Flop counts for SIGS and SITS.

The figures given in Table 4.2 are based on the assumption that after k has reached the value $\lfloor \log_2 \frac{n}{p} \rfloor$, the Cholesky factor Y is a square $n \times n$ matrix for all subsequent iteration steps. If this is not the case (in particular, if $\text{rank}(X) < n$), then these iteration steps are slightly cheaper. It is difficult to compare the costs for Algorithms SIGS and SITS to that of Algorithms SIGE and SITR or to the cost for the generalization of Hammarling's algorithm presented in [30].

If we only compare those parts of the WHILE-loops different in SIGE/SITR and SIGS/SITS, then we have the following result: if $t = \lfloor \log_2 \frac{n}{p} \rfloor$, then the updates of X for the first t iterations of SIGE and SITR cost about $3tn^3$ flops while the corresponding cost for the updates of Y in SIGS and SITS is only $n^3 - 2pn^2$. Therefore, the algorithms computing the Cholesky factor of X save a large amount of floating point operations during the first iterations if the ratio n/p is large. This may be compensated by the slightly more expensive final steps (i.e., Steps $t+1, t+2, \ldots$).

The main reason for using the Cholesky factor approach is the expected higher accuracy in case that X is ill-conditioned. Therefore, we propose to use Algorithms SIGS and SITR even in case they are more expensive than their correspondents SIGE and SITR, in particular also if $p > n/2$. (Note that in case $p > n$, workspace of dimension $2p \times n$ is required for the updates of Y!)

Remark 4.4 *In all algorithms presented in this section, instead of determinantal scaling we may use any of the scaling schemes suggested in [32, 4, 19, 22].*

5 Numerical Results

In this section we compare the methods for solving generalized Lyapunov equations based on the matrix sign function. We include in the comparison two direct methods: the Bartels-Stewart algorithm [5] and Hammarling's algorithm [18], hereafter BTST and HAMM, respectively. The codes for these algorithms are those described in [30].

We have employed the iterative scheme for the matrix sign function described in Section 4 in Algorithm SIGE, and its triangular version in Algorithm SITR. We have also used the iterative schemes for computing the Cholesky factor in Algorithms SIGS and SITS (triangular version). In all iterative schemes we have used the stopping criterion proposed in Section 4, i.e.,

$$\|A_{k+1} + E\|_1 \;\leq\; 10 \cdot n \cdot \sqrt{\varepsilon} \cdot \|E\|_1,$$

plus two additional iterations once the stopping criterion is satisfied.

In our experimental analysis, the backward accuracy of the algorithms is estimated by means of the normalized residual:

$$\|Q + A^T X E + E^T X A\|_1 / \|X\|_1. \tag{5.25}$$

Moreover, the numerical results are compared to the separation of the Lyapunov operator given in (1.3) and denoted by $\mathrm{sep}_F(\Omega)$, and the condition number of the problem, i.e., $\mathrm{cond}_2(\Omega)$ as given in (1.2). These values are estimated using algorithm BTST from [30].

All experiments were performed using Fortran 77 and IEEE double precision arithmetic ($\varepsilon \approx 2.2 \times 10^{-16}$), on a SUN UltraSparc-167MHZ platform. The appropriate compiler optimization options were used in the algorithms to optimize performance. We also made extensive use of vendor supplied BLAS (SUN *performance library*) and the computational kernels in LAPACK [1].

We have performed two sets of experiments. The first set is designed to evaluate the numerical reliability of the solvers, while in the second set we evaluate the performance.

5.1 Numerical reliability

Example 5.1 *[14]* The coefficient matrices in this example are defined as:

$$A \;=\; -((2^{-\tau} - 1)I_n + \mathrm{diag}(1, 2, \ldots, n) + U_n^T), \qquad E \;=\; I_n + 2^{-\tau} U_n,$$

where U_n is an $n \times n$ strictly lower triangular matrix with all nonzero entries equal to 1. The right-hand side matrix Q is computed from

$$Q := -(A^T X E + E^T X A), \tag{5.26}$$

with X set to a matrix with all unit entries.

The condition number $\text{cond}_2(\Omega)$ is controlled by the parameter τ. When τ is increased, one of the generalized eigenvalues of the matrix pencil $A - \lambda E$ approaches zero and the Lyapunov equation becomes more ill-conditioned. Table 5.1 reports the separation and the reciprocal of the condition number for Example 5.1 and matrices of dimension $n = 100$. As expected, the table shows a constant increase in the ill-conditioning of the problem as τ is increased.

τ	$\text{sep}_F(\Omega)$	$1/\text{cond}_2(\Omega)$
10	4.9×10^{-4}	4.2×10^{-8}
20	4.3×10^{-7}	3.7×10^{-11}
30	4.2×10^{-10}	3.6×10^{-14}
40	4.1×10^{-13}	3.6×10^{-17}

Table 5.1: Separation and reciprocal of the condition number for Example 5.1 ($n = 100$).

Table 5.2 reports the normalized residuals (5.25) obtained by the solvers for Example 5.1, $n = 100$, and increasing values of τ.

τ	BTST	SIGE (iter.)	SITR (iter.)
10	3.1×10^{-12}	1.1×10^{-10} (19)	7.6×10^{-11} (19)
20	6.3×10^{-12}	5.4×10^{-8} (27)	4.5×10^{-8} (27)
30	1.3×10^{-12}	5.8×10^{-5} (34)	4.4×10^{-5} (34)
40	7.7×10^{-13}	2.6×10^{-2} (41)	3.1×10^{-2} (41)

Table 5.2: Normalized residuals and number of matrix sign function iterations (inside parentheses) for Example 5.1 ($n = 100$).

The matrix sign function solvers obtain a normalized residual which agrees with the condition number of the problem. Both, SIGE and SITR solvers obtain similar results.

On the other hand, the solver based on the Bartels-Stewart method achieves a highly remarkable accuracy. Actually, the results are much better than expected from the conditioning of the corresponding problems, as measured by $\text{cond}_2(\Omega)$. This can be explained as follows: in the first stage of the Bartels-Stewart algorithm, $A - \lambda E$ is reduced to generalized real Schur form (quasi-triangular form) by means of the QZ algorithm. Since all generalized eigenvalues of the matrix pencil $A - \lambda E$ are real, both matrices are reduced to triangular form. Then, the triangular Lyapunov equation is solved by back substitution. The first stage reveals a small eigenvalue of $A - \lambda E$ which appears as a 1×1 block on the top diagonal entry of the generalized real Schur form. Although the back substitution stage is ill-conditioned due to the existence of this small element, the triangular Lyapunov equation is solved with high accuracy. Thus, it seems that this back substitution process shares the high accuracy of the solution of triangular linear systems [20, Chapter 8], [39].

Example 5.2 Consider the following modification of the matrix A from Example 5.1,

$$A \quad = \quad -((2^{-\tau} - 1)I_n + \mathrm{diag}(n, n-1, \ldots, 1) + U_n^T).$$

Table 5.3 reports the normalized residuals for this variant of Example 5.1. The table shows that a simple reordering of the diagonal elements of A produced remarkable differences in the results for the matrix sign function solvers. Here, similar results are obtained both with the Bartels-Stewart method and the solvers based on the matrix sign function although $\mathrm{sep}_F(\Omega)$ and $\mathrm{cond}_2(\Omega)$ are the same as in Example 5.1.

τ	BTST	SIGE (iter.)	SITR (iter.)
10	1.6×10^{-12}	2.8×10^{-12} (19)	2.8×10^{-11} (19)
20	1.7×10^{-12}	1.0×10^{-12} (27)	2.0×10^{-12} (27)
30	4.9×10^{-12}	9.8×10^{-13} (34)	1.4×10^{-12} (34)
40	3.6×10^{-12}	1.1×10^{-12} (41)	4.9×10^{-13} (41)

Table 5.3: Normalized residuals and number of matrix sign function iterations (inside parentheses) for Example 5.2 ($n = 100$).

Example 5.3 *[30]* The coefficient matrices in this example are defined for $n = 3q$ as follows:

$$A = V_n \,\mathrm{diag}(A_1, \ldots, A_q) W_n, \quad A_i = \begin{pmatrix} \sigma_i & 0 & 0 \\ 0 & \tau_i & \tau_i \\ 0 & -\tau_i & \tau_i \end{pmatrix}, \quad E = V_n W_n,$$

where W_n is an $n \times n$ lower triangular matrix with all unit entries, and V_n is an $n \times n$ matrix with unit entries on and below the anti-diagonal, and all other entries equal to zero. The positive semidefinite right-hand side Q is defined by

$$Q := C^T C, \quad C = [\,1, 2, \ldots, n\,]. \tag{5.27}$$

Here, $A - \lambda E$ has real and complex eigenvalues σ_i and $\tau_i \pm \tau_i\imath$, respectively, where $\imath := \sqrt{-1}$. Table 5.4 reports the separation and the reciprocal of the condition number for Example 5.3, with $\sigma_i = \tau_i := \tau^i$. As the table shows, the separation is not altered much as τ is increased. However, the problem becomes more ill-conditioned due to the increase in $\|A\|_F$.

Table 5.5 reports the normalized residuals achieved by the compared solvers for Example 5.3, $n = 99$, and various values of τ.

This table shows very similar results for all the solvers, which agree with the condition number of the problem.

τ	$\mathrm{sep}_F(\Omega)$	$1/\mathrm{cond}_2(\Omega)$
1.0	6.0×10^{-1}	1.8×10^{-9}
1.2	6.2×10^{-2}	1.4×10^{-11}
1.4	1.0×10^{-1}	2.0×10^{-13}
1.6	1.2×10^{-1}	3.6×10^{-15}
1.8	1.3×10^{-1}	9.9×10^{-17}

Table 5.4: Separation and reciprocal of the condition number for Example 5.3 ($n = 99$).

τ	BTST	SIGE (iter.)	SITR (iter.)
1.0	2.5×10^{-11}	5.9×10^{-12} (6)	2.8×10^{-10} (6)
1.2	9.2×10^{-9}	1.7×10^{-9} (8)	1.3×10^{-8} (8)
1.4	1.7×10^{-6}	3.1×10^{-7} (9)	4.5×10^{-6} (8)
1.6	7.0×10^{-5}	2.8×10^{-5} (9)	1.2×10^{-4} (9)
1.8	3.9×10^{-3}	6.4×10^{-4} (10)	2.0×10^{-3} (9)

τ	HAMM	SIGS (iter.)	SITS (iter.)
1.0	3.4×10^{-11}	2.9×10^{-12} (6)	3.6×10^{-11} (6)
1.2	1.2×10^{-8}	5.0×10^{-9} (8)	1.2×10^{-8} (8)
1.4	9.6×10^{-7}	6.9×10^{-7} (9)	7.5×10^{-7} (8)
1.6	2.8×10^{-5}	5.7×10^{-5} (9)	5.4×10^{-5} (9)
1.8	2.9×10^{-3}	8.1×10^{-4} (10)	3.4×10^{-4} (9)

Table 5.5: Normalized residuals and number of matrix sign function iterations (inside parentheses) for Example 5.3 ($n = 99$).

Example 5.4 We have also generated coefficient matrices A, E with random entries using uniform distribution, and random stable generalized eigenvalues, uniformly distributed in $[-1, 0)$. X was set to a matrix with all unit entries, and the right-hand side was then generated as in (5.27). The numerical results, though not reported, show closely similar accuracies for all solvers.

5.2 Computing performance

Example 5.5 In this example we generated the following coefficient matrices:

$$A - \lambda E \;=\; V_n\,(\mathrm{diag}(\alpha_1, \ldots, \alpha_n) - \lambda I_n)\,W_n,$$

where the scalars $\alpha_1, \ldots, \alpha_n$ are uniformly distributed in $[-10, 0)$ and V_n, W_n are defined as in Example 5.3. Then, C was constructed as a random $p \times n$ matrix and Q was set to $C^T C$. We only used integer entries for C to avoid the loss of numerical accuracy when Q is constructed. In all figures we report the execution time (in seconds), averaged for five executions on different matrices.

Our first experiment is designed to analyze the performance of the solvers based on the matrix sign function when the number of rows of C is increased from $p = 1$ to $p = n$. Thus, we keep n constant and carry out a fixed number of iterations in all solvers based on the matrix sign function, say 10. We also included two direct methods, specifically Bartels-Stewart method and Hammarling's algorithm, in the experiment. The results are given in Figures 5.1–5.3.

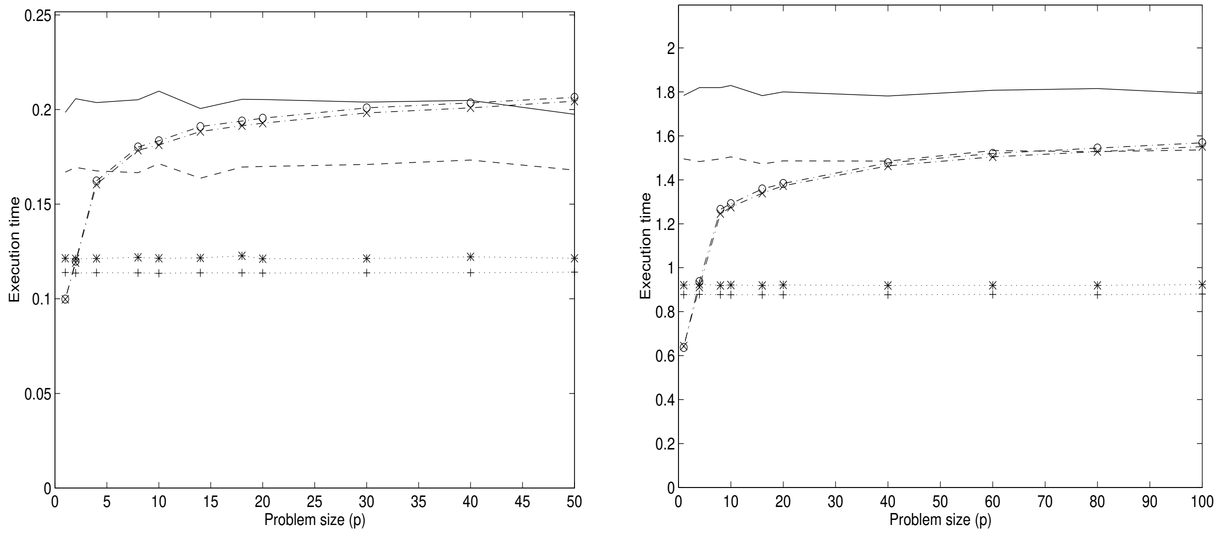

Figure 5.1: Execution times of the tested solvers on the SUN UltraSparc for $n = 50$ (left), $n = 100$ (right), and varying sizes of p. Legend: "—" = BTST, "- - -" = HAMM, "$\cdots + \cdots$" = SIGE, "$\cdots * \cdots$" = SITR, "$- \cdot - \times - \cdot -$" = SIGS, "$- \cdot - \circ - \cdot -$" = SITS.

As expected, the performances of the Bartels-Stewart method and Algorithms SIGE and SITR are independent of p, since all these methods make explicit use of the $n \times n$ matrix Q. It is also worth noting that the cost of Hammarling's algorithm is always smaller than the cost of Bartels-Stewart method, and as p is increased the cost of Hammarling's algorithm only becomes slightly larger. On the other hand, the performance of those methods based on the matrix sign function that compute the Cholesky factor strongly depend on p. The computational cost of these algorithms rapidly increases as p becomes larger and, for large-scale problems ($n \geq 300$), 10 iterations of these methods are even more expensive than direct methods. Comparing the flop counts given in Table 4.2 and [30], Algorithms SIGS and SITS are faster than Hammarling's algorithm for $p \leq n/8$. This can also be observed in Figures 5.1–5.3. As in many control applications, $p \ll n$, Algorithms SIGS and SITS will show good performance for such problems. From the point of view of performance,

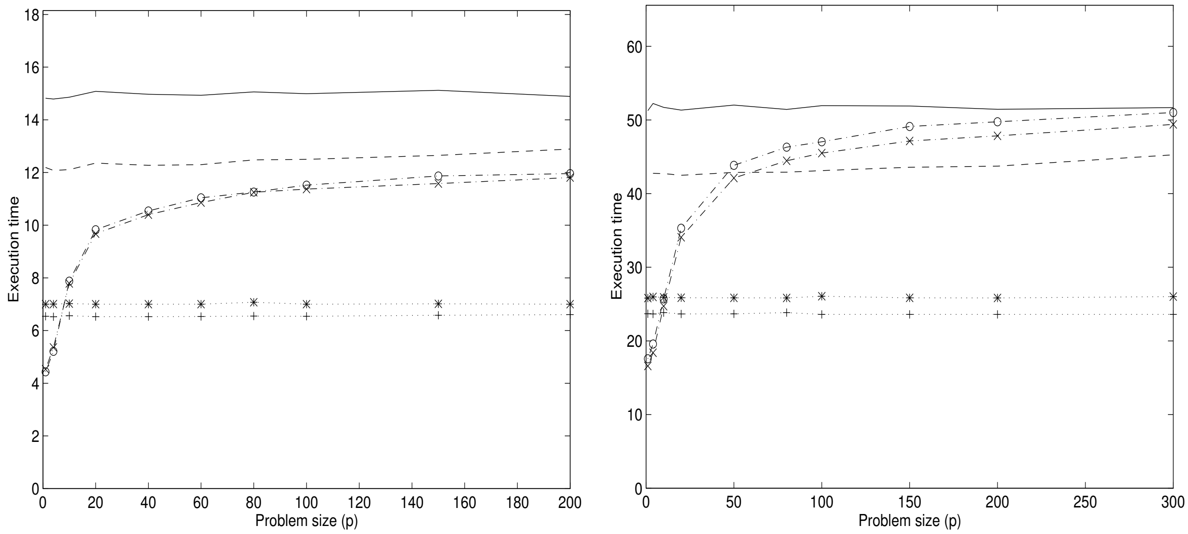

Figure 5.2: Execution times of the tested solvers on the SUN UltraSparc for $n = 200$ (left), $n = 300$ (right), and various sizes for p. Same legend as in Figure 5.1.

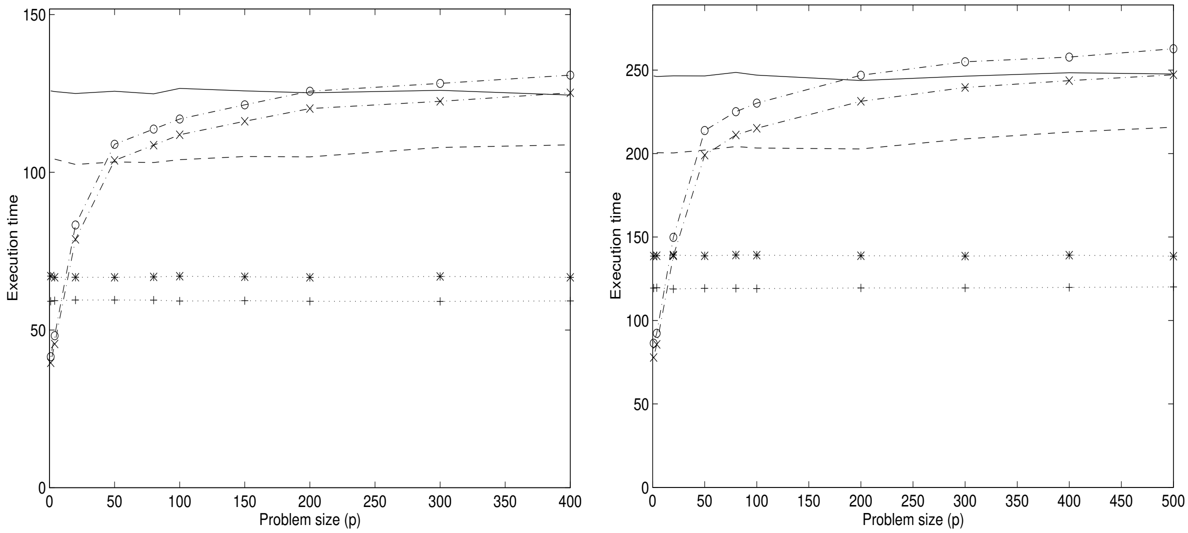

Figure 5.3: Execution times of the tested solvers on the SUN UltraSparc for $n = 400$ (left), $n = 500$ (right), and various sizes for p Same legend as in Figure 5.1.

Algorithms SIGE and SITR are more efficient than Algorithms SIGS and SITS unless p is very small. But if the Cholesky factor of the solution is required, one may sacrifice some execution time for higher accuracy.

A somewhat surprising observation is that solvers that initially transform the matrix pair (A, E) to reduce the cost of the iteration (Algorithms SITR and SITS) obtain higher execution times. Better performances of algorithms SIGE and SIGS are basically due to the highly efficient implementation of the dense matrix product kernels in most current computer architectures. Algorithms SITR and SITS rely on cheaper matrix products (one matrix is triangular), but these matrix kernels are not as efficient as the ones for the "full times full" matrix product.

Our next experiment is designed to evaluate the performances of the iterative solvers compared to the direct methods. Thus, we only report the execution time of the best iterative solver and the best direct method. Since the number of iterations of the iterative solvers depends on the problem, we show the results for a fixed number of iterations: 5, 10, 15, and 20. In practice it can be observed that 8–12 iterations are often enough to achieve convergence if the problem is fairly well-conditioned.

Figures 5.4 and 5.5 report two different extreme cases, $p = 1$ and $p = n$. Figure 5.4 shows that, when $p = 1$, even with 20 iterations, the iterative solvers require smaller execution times than the direct solvers. Actually, the difference between both methods gets larger as n increases. Hence, for $p = 1$, the matrix sign function solvers present much better results.

When $p = n$ the situation is quite different (see Figure 5.5). In this case, the direct method requires similar execution time as 20 iterations of the iterative algorithms. The tendency holds as n is increased since the direct method only obtains slightly better results than 20 iterations of the matrix sign function.

Note that from the flop counts, roughly speaking, already 10 iterations of the sign function based solvers should be as expensive as the direct methods. The better performance of the sign function based methods is caused by the highly efficient computational kernels they are composed of.

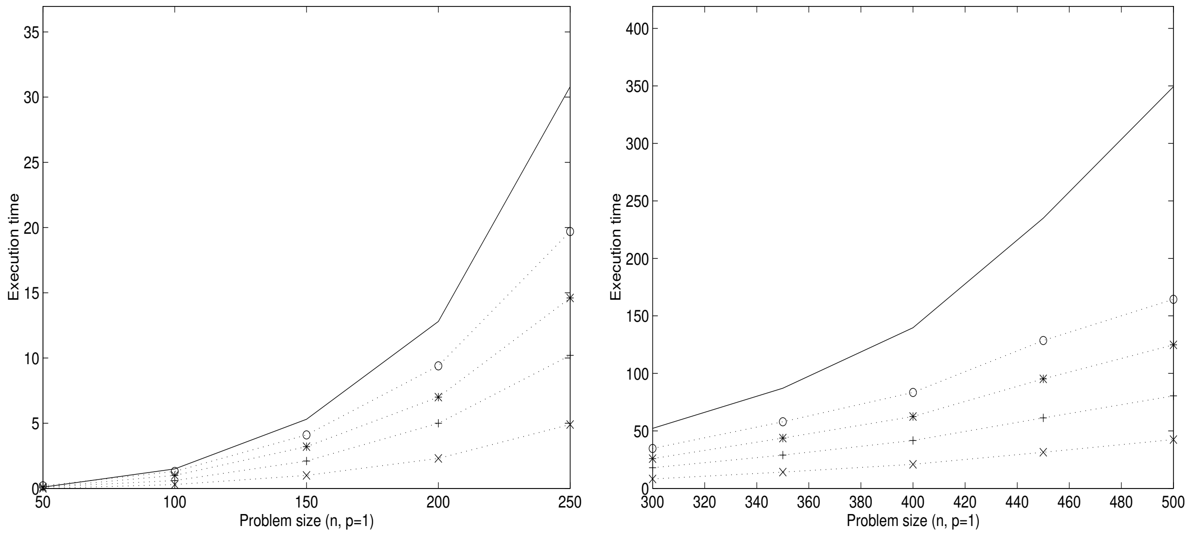

Figure 5.4: Execution times of the tested solvers on the SUN UltraSparc for small (left) and large (right) problems ($p = 1$). Legend: "—" = best direct solver, "$\cdots$" = best iterative solver, where "$\cdots \times \cdots$" = 1 iteration, "$\cdots + \cdots$" = 10 iterations, "$\cdots * \cdots$" = 15 iterations, "$\cdots \circ \cdots$" = 20 iterations.

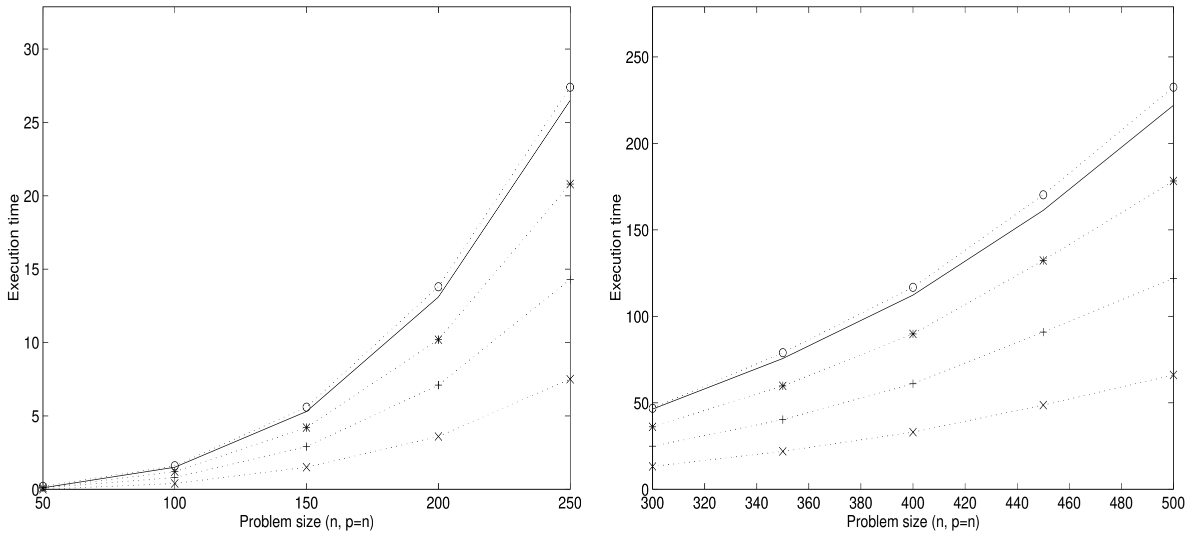

Figure 5.5: Execution times of the tested solvers on the SUN UltraSparc for small (left) and large (right) problems ($p = n$). Same legend as in Figure 5.3.

6 Concluding Remarks

We have presented methods for the numerical solution of stable generalized Lyapunov equations, based on the iterative schemes for the matrix sign function. Several results from the standard case are extended to the generalized case. The derived methods can be split in two classes: methods for computing the solution matrix explicitly and methods for computing its Cholesky factor directly in case of semidefinite Lyapunov equations.

We have also discussed several implementation issues: a new stopping criterion is introduced which saves computational cost/work space compared to existing stopping criteria for sign function iterations, and an initial reduction of the matrix pair results in cheaper iterations (in terms of floating point operations).

The numerical results show that the matrix sign function solvers obtain results which are as accurate as could be expected from the conditioning of the problem. Furthermore, the new stopping criteria is revealed as an appropriate and efficient tool for our solvers. From the point of view of execution time, the matrix sign function solvers consistently outperform those methods based on the QZ algorithm. Although not reported, similar results were obtained on other platforms (e.g., SGI R10000-200MHz, IBM RS/6000, HP9000/715, etc.). These results will be enhanced on parallel distributed architectures, where the computational kernels involved in the matrix sign function solvers have already shown their high performance.

Acknowledgments

We thank T. Penzl for some helpful discussions and for providing the codes implementing the generalized Bartels-Stewart method and Hammarling's algorithm. We also thank Volker Mehrmann for providing some helpful suggestions.

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J.Dongarra, J. DuCroz, A. Greenbaum, S. Hammerling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide Release 2.0.* SIAM, Philadelphia, 1994.

[2] B. D. O. Anderson and J. B. Moore. *Linear Optimal Control.* Prentice-Hall, Englewood Cliffs, NJ, 1971.

[3] Z. Bai and J. Demmel. Design of a parallel nonsymmetric eigenroutine toolbox, Part I. In R.F. Sincovec et al, editor, *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, 1993.

[4] L. Balzer. Accelerated convergence of the matrix sign function. *Internat. J. Control*, 32:1057–1078, 1980.

[5] R.H. Bartels and G.W. Stewart. Solution of the matrix equation $AX + XB = C$: Algorithm 432. *Comm. ACM*, 15:820–826, 1972.

[6] A. N. Beavers and E. D. Denman. A new solution method for the Lyapunov matrix equations. *SIAM J. Appl. Math.*, 29:416–421, 1975.

[7] P. Benner. *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems.* Dissertation, Fakultät für Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, Germany, February 1997.

[8] R. Byers. Solving the algebraic Riccati equation with the matrix sign function. *Linear Algebra Appl.*, 85:267–279, 1987.

[9] R. Byers, C. He, and V. Mehrmann. The matrix sign function method and the computation of invariant subspaces. *SIAM J. Matrix Anal. Appl.*, 18(3):615–632, 1997.

[10] T. Chan. Rank revealing QR factorizations. *Linear Algebra Appl.*, 88/89:67–82, 1987.

[11] E.D. Denman and A.N. Beavers. The matrix sign function and computations in systems. *Appl. Math. Comput.*, 2:63–94, 1976.

[12] J. D. Gardiner. Stabilizing control for second-order models and positive real systems. *AIAA J. Guidance, Dynamics and Control*, 15(1):280–282, 1992.

[13] J.D. Gardiner and A.J. Laub. A generalization of the matrix-sign-function solution for algebraic Riccati equations. *Internat. J. Control*, 44:823–832, 1986.

[14] J.D. Gardiner, A.J. Laub, J.J. Amato, and C.B. Moler. Solution of the Sylvester matrix equation $AXB + CXD = E$. *ACM Trans. Math. Software*, 18:223–231, 1992.

[15] J.D. Gardiner, M.R. Wette, A.J. Laub, J.J. Amato, and C.B. Moler. Algorithm 705: A Fortran-77 software package for solving the Sylvester matrix equation $AXB^T + CXD^T = E$. *ACM Trans. Math. Software*, 18:232–238, 1992.

[16] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, second edition, 1989.

[17] S. J. 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.

[18] S.J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA J. Numer. Anal.*, 2:303–323, 1982.

[19] N. J. Higham. Newton's method for the matrix square root. *Math. Comp.*, 46:537–549, 1986.

[20] N. J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, PA, 1996.

[21] W. D. Hoskins, D. S. Meek, and D. J. Walton. The numerical solution of $A'Q+QA = -C$. *IEEE Trans. Automat. Control*, AC-22:882–883, 1977.

[22] C. Kenney and A.J. Laub. On scaling Newton's method for polar decomposition and the matrix sign function. *SIAM J. Matrix Anal. Appl.*, 13:688–706, 1992.

[23] C. Kenney and A.J. Laub. The matrix sign function. *IEEE Trans. Automat. Control*, 40(8):1330–1348, 1995.

[24] C. Kenney, A.J. Laub, and P.M. Papadopoulos. Matrix-sign algorithms for Riccati equations. *IMA J. Math. Contr. Info.*, 3:331–344, 1992.

[25] P. Lancaster and L. Rodman. *The Algebraic Riccati Equation.* Oxford University Press, Oxford, 1995.

[26] P. Lancaster and M. Tismenetsky. *The Theory of Matrices.* Academic Press, Orlando, 2nd edition, 1985.

[27] V.B. Larin and F.A. Aliev. Construction of square root factor for solution of the Lyapunov matrix equation. *Sys. Control Lett.*, 20:109–112, 1993.

[28] V. Mehrmann. *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution.* Number 163 in Lecture Notes in Control and Information Sciences. Springer-Verlag, Heidelberg, July 1991.

[29] P. Misra, P. Van Dooren, and A. Varga. Computation of structural invariants of generalized state-space systems. *Automatica*, 30(12):1921–1936, 1994.

[30] T. Penzl. Numerical solution of generalized Lyapunov equations. Technical Report SFB393/96-02, Fak. f. Mathematik, TU Chemnitz-Zwickau, 09107 Chemnitz, FRG, 1996.

[31] P.H. Petkov, N.D. Christov, and M.M. Konstantinov. *Computational Methods for Linear Control Systems.* Prentice-Hall, Hertfordshire, UK, 1991.

[32] J.D. Roberts. Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *Internat. J. Control*, 32:677–687, 1980. (Reprint of Technical Report No. TR-13, CUED/B-Control, Cambridge University, Engineering Department, 1971).

[33] I.G. Rosen and C. Wang. A multi–level technique for the approximate solution of operator Lyapunov and algebraic Riccati equations. *SIAM J. Numer. Anal.*, 32(2):514–541, 1995.

[34] M. G. Safonov and R. Y. Chiang. Model reduction for robust control: A Schur relative error method. *Int. J. Adapt. Cont. and Sign. Proc.*, 2:259–272, 1988.

[35] V. Sima. *Algorithms for Linear-Quadratic Optimization*, volume 200 of *Pure and Applied Mathematics*. Marcel Dekker, Inc., New York, NY, 1996.

[36] A. Varga. A note on Hammarling's algorithm for the discrete Lyapunov equation. *Sys. Control Lett.*, 15(3):273–275, 1990.

[37] A. Varga. Computation of Kronecker-like forms of a system pencil: Applications, algorithms and software. In *Proc. CACSD'96 Symposium, Dearborn, MI*, pages 77–82, 1996.

[38] A. Varga and T. Katayama. Computation of j–inner–outer factorizations of rational matrices. *Internat. J. Robust and Nonlinear Cont.*, 7, 1997.

[39] J.H. Wilkinson. *Rounding Errors in Algebraic Processes.* Prentice-Hall, Englewood Cliffs, NJ, 1963.

Other titles in the SFB393 series:

97-01 P. Benner, V. Mehrmann, H. Xu. A new method for computing the stable invariant subspace of a real Hamiltonian matrix or Breaking Van Loan's curse? January 1997.

97-02 B. Benhammouda. Rank-revealing 'top-down' ULV factorizations. January 1997.

97-03 U. Schrader. Convergence of Asynchronous Jacobi-Newton-Iterations. January 1997.

97-04 U.-J. Görke, R. Kreißig. Einflußfaktoren bei der Identifikation von Materialparametern elastisch-plastischer Deformationsgesetze aus inhomogenen Verschiebungsfeldern. March 1997.

97-05 U. Groh. FEM auf irregulären hierarchischen Dreiecksnetzen. March 1997.

97-06 Th. Apel. Interpolation of non-smooth functions on anisotropic finite element meshes. March 1997

97-07 Th. Apel, S. Nicaise. The finite element method with anisotropic mesh grading for elliptic problems in domains with corners and edges.

97-08 L. Grabowsky, Th. Ermer, J. Werner. Nutzung von MPI für parallele FEM-Systeme. March 1997.

97-09 T. Wappler, Th. Vojta, M. Schreiber. Monte-Carlo simulations of the dynamical behavior of the Coulomb glass. March 1997.

97-10 M. Pester. Behandlung gekrümmter Oberflächen in einem 3D-FEM-Programm für Parallelrechner. April 1997.

97-11 G. Globisch, S. V. Nepomnyaschikh. The hierarchical preconditioning having unstructured grids. April 1997.

97-12 R. V. Pai, A. Punnoose, R. A. Römer. The Mott-Anderson transition in the disordered one-dimensional Hubbard model. April 1997.

97-13 M. Thess. Parallel Multilevel Preconditioners for Problems of Thin Smooth Shells. May 1997.

97-14 A. Eilmes, R. A. Römer, M. Schreiber. The two-dimensional Anderson model of localization with random hopping. June 1997.

97-15 M. Jung, J. F. Maitre. Some remarks on the constant in the strengthened C.B.S. inequality: Application to h- and p-hierarchical finite element discretizations of elasticity problems. July 1997.

97-16 G. Kunert. Error estimation for anisotropic tetrahedral and triangular finite element meshes. August 1997.

97-17 L. Grabowsky. MPI-basierte Koppelrandkommunikation und Einfluß der Partitionierung im 3D-Fall. August 1997.

97-18 R. A. Römer, M. Schreiber. Weak delocalization dueto long-range interaction for two electrons in a random potential chain. August 1997.

97-19 A. Eilmes, R. A. Römer, M. Schreiber. Critical behavior in the two-dimensional Anderson model of localization with random hopping. August 1997.

97-20 M. Meisel, A. Meyer. Hierarchically preconditioned parallel CG-solvers with and without coarse-matrix-solvers inside FEAP. September 1997.

97-21 J. X. Zhong, U. Grimm, R. A. Römer, M. Schreiber. Level-Spacing Distributions of Planar Quasiperiodic Tight-Binding Models. October 1997.

97-22 W. Rehm (Ed.). Ausgewählte Beiträge zum 1. Workshop Cluster-Computing. TU Chemnitz, 6./7. November 1997.

97-23 P. Benner, E. S. Quintana-Ortí. Solving Stable Generalized Lyapunov Equations with the Matrix Sign Function. October 1997

The complete list of current and former preprints is available via
`http://www.tu-chemnitz.de/sfb393/sfb97pr.html`.