Technische Universität Chemnitz

Sonderforschungsbereich 393

Numerische Simulation auf massiv parallelen Rechnern

P. Hr. Petkov M. M. Konstantinov V. Mehrmann

DGRSVX and DMSRIC: Fortran 77 subroutines for solving continuous–time matrix algebraic Riccati equations with condition and accuracy estimates

Preprint SFB393/98-16

Preprint-Reihe des Chemnitzer SFB 393

SFB393/98-16 **May 1998**

Authors:

P.Hr. Petkov
Dept. of Automatics
Technical Univ. of Sofia
1756 Sofia, Bulgaria
E-mail: <php@mbox.digsys.bg>

M.M. Konstantinov
Univ. of Arch. & Civil Eng.
1 Hr. Smirnenski Blv.
1421 Sofia, Bulgaria
E-mail: <mmk_fte@uacg.acad.bg>

V. Mehrmann
Fakultät für Mathematik
Technische Universität Chemnitz
D-09107 Chemnitz, Germany
E-mail: <mehrmann@mathematik.tu-chemnitz.de>

DGRSVX and DMSRIC: Fortran 77 subroutines for solving continuous–time matrix algebraic Riccati equations with condition and accuracy estimates [*]

P.Hr. Petkov[†] M.M. Konstantinov[‡] V. Mehrmann[§]

Abstract

We present new Fortran 77 subroutines which implement the Schur method and the matrix sign function method for the solution of the continuous-time matrix algebraic Riccati equation on the basis of LAPACK subroutines. In order to avoid some of the well-known difficulties with these methods due to a loss of accuracy, we combine the implementations with block scalings as well as condition estimates and forward error estimates. Results of numerical experiments comparing the performance of both methods for more than one hundred well- and ill-conditioned Riccati equations of order up to 150 are given. It is demonstrated that there exist several classes of examples for which the matrix sign function approach performs more reliably and more accurately than the Schur method. In all cases the forward error estimates allow to obtain a reliable bound on the accuracy of the computed solution.

1 Introduction

In this report we present a set of Fortran 77 subroutines for the solution of continuous–time matrix algebraic Riccati equations. The following functionalities are provided by these programs:

- An implementation of the Schur method [39, 40] with a block–scaling which increases the numerical reliability.
- An implementation of the matrix sign function method [15, 19, 24, 38, 37] with the same scaling.
- Both implementations are accompanied with the computation of condition estimates and forward error estimates which allow to estimate the sensitivity of the problem and the accuracy of the solution.
- Both implementations use LAPACK [2] and BLAS [41, 22, 21] subroutines for the efficient solution of the corresponding numerical linear algebra subproblems.

[*]This work was supported by the Volkswagen Stiftung Research Grant 5/72 764.

[†]On leave from the Dept. of Automatics, Technical Univ. of Sofia, 1756 Sofia, Bulgaria E-mail: <php@mbox.digsys.bg>.

[‡]On leave from the Univ. of Arch. & Civil Eng., 1 Hr. Smirnenski Blv., 1421 Sofia, Bulgaria E-mail: <mmk_fte@uacg.acad.bg>.

[§]Fakultät für Mathematik, Technische Universität Chemnitz, D-09107 Chemnitz, Germany, E-mail: <mehrmann@mathematik.tu-chemnitz.de>.

The following notation is used in the paper.

- $\mathcal{R}$ – the field of real numbers;
- $\mathcal{R}^{m\times n}$ – the space of $m\times n$ matrices $A=[a_{ij}]$ over $\mathcal{R}$;
- A^T – the transpose of a matrix A;
- $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ – the maximum and minimum singular value of A;
- $\|A\|_1$ – the matrix 1-norm of the matrix A;
- $\|A\|_2=\sigma_{\max}(A)$ – the spectral norm of the matrix A;
- $\|A\|_F=(\sum|a_{ij}|^2)^{1/2})$ – the Frobenius norm;
- I_n – the $n\times n$ identity matrix;
- $A\otimes B$ – the Kronecker product of matrices A and B;
- $\mathrm{Vec}(A)$ – the vector, obtained by stacking the columns of A in one vector;
- ε – the roundoff unit of the machine arithmetic.

Consider the continuous–time matrix algebraic Riccati equation

$$A^TX+XA+C-XDX=0 \tag{1}$$

where $A\in\mathcal{R}^{n\times n}$ and the matrices $C,\ D,\ X\in\mathcal{R}^{n\times n}$ are symmetric. We assume that there exists a non–negative definite solution X which stabilises $A-DX$. This includes, for instance, the case when C and D are non–negative definite with the pair (A,D) stabilizable and the pair (C,A) detectable. The described programs, however, may be used also in the case when the matrices C and D are symmetric indefinite, as for example in the solution of H_∞ optimisation problems [29].

The numerical solution of a matrix Riccati equation as most other numerical problems may face serious difficulties. First of all, the equation may be *ill-conditioned*, i.e., small perturbations in the coefficient matrices A, C, D may lead to large variations in the solution.

As is well known, the conditioning of a problem depends neither on the use method nor on the properties of the computer architecture. So, it is necessary to have a quantitative characterisation of the conditioning in order to estimate the accuracy of solution obtained.

Another difficulty is connected with the stability of the numerical method and the robustness of its implementation. In general we can do a backward error analysis and estimate the backward error. In the case of Riccati equations one of the possible factors that leads to numerical instability is the scaling of the Hamiltonian matrix associated with the Riccati equation.

In the solution of the Riccati equation, the situation is even more complicated, since a mixup between the conditioning and the instability of the method may happen. This is due to the fact that the Riccati solution is determined in essentially two steps, for example in the Schur method first a computation of the Schur form of the Hamiltonian matrix is performed and then from the invariant subspaces the Riccati solution is determined via the solution of a linear system. In such a situation it may happen that, although the solution of the Riccati equation is a well-conditioned problem, one of the intermediate problems may be drastically more ill-conditioned than the other. This can be viewed as an ill-conditioning of the problem or as an instability of the method. In the case of the Schur method this means that, although we use the QR-algorithm to obtain the ordered Schur form and a numerically stable linear system solver to obtain the Riccati solution from the invariant subspace, the solution may be very inaccurate, see the examples below. Thus we may either conclude that the combined numerical method is unstable, although it consists of stable components or we may conclude that the solution of the problem is ill-conditioned, since it is a composition of two mappings one of which may be ill-conditioned. Either point of view is justified, since the ill-conditioning of the problem and the instability of the method is mixed up in this case and a careful analysis is needed. Some of these difficulties are resolved by using a proper scaling of the Riccati equation.

The paper is organised as follows. In Section 2 we discuss different approaches to the condition estimation for continuous-time Riccati equations. We present an efficient method for estimating the condition number which is based on the matrix norm estimator, implemented in LAPACK [2]. In Section 3 we briefly describe the Schur method and the matrix sign function method and discuss their numerical properties.

Based on the results of earlier experiments, two cheap block-scaling schemes are presented in Section 4, which enhance the accuracy of the Riccati solvers [12, 47]. In Section 5 we propose a residual based forward error estimate which is based on the LAPACK norm estimator and may be used in combination with any method for solving the Riccati equation. In Section 6 we describe the software implementation of both methods, which is based entirely on modules from LAPACK and BLAS. Finally, in Section 7 we present the results of several numerical experiments comparing the performance of the Schur method and the sign function methods for more than one hundred well- and ill-conditioned Riccati equations of order 150. We demonstrate the loss of accuracy of the un-scaled Schur and matrix sign function methods in the solution of well-conditioned equations and show the improvement of the accuracy due to both types of scaling. We also show that there exist several classes of examples for which the matrix sign function method performs more reliably than the Schur method. In all cases the forward error estimate allows to obtain a reliable bound on the accuracy of the solution computed.

2 Conditioning and condition estimation

Let the coefficient matrices A, C, D in (1) be subject to perturbations ΔA, ΔC, ΔD, respectively, so that instead of the initial data we have the matrices $\tilde{A} = A + \Delta A$, $\tilde{C} = C + \Delta C$, $\tilde{D} = D + \Delta D$. The aim of the *perturbation analysis* of (1) is to investigate the variation ΔX in the solution $\tilde{X} = X + \Delta X$ due to the perturbations ΔA, ΔC, ΔD. If small perturbations in the data lead to small variations in the solution we say that the Riccati equation is *well-conditioned* and if these perturbations lead to large variations in the solution this equation is *ill-conditioned*. In the perturbation analysis of the Riccati equation it is supposed that the perturbations preserve the symmetric structure of the equation, i.e., the perturbations ΔC and ΔD are symmetric. If $\|\Delta A\|$, $\|\Delta C\|$ and $\|\Delta D\|$ are sufficiently small, then the perturbed solution $\tilde{X} = X + \Delta X$ is well defined [34, 23].

The *condition number of the Riccati equation* is defined as (see [17])

$$K = \lim_{\delta \to 0} \sup \left\{ \frac{\|\Delta X\|}{\delta \|X\|} : \|\Delta A\| \leq \delta \|A\|, \ \|\Delta C\| \leq \delta \|C\|, \ \|\Delta D\| \leq \delta \|D\| \right\}.$$

For sufficiently small δ we have (within first order terms)

$$\frac{\|\Delta X\|}{\|X\|} \leq K\delta.$$

Let $\bar{X}$ be the solution of the Riccati equation computed by a numerical method in finite arithmetic with relative precision ε. If the method is *backward stable*, then we can estimate the error in the solution error

$$\frac{\|\bar{X} - X\|}{\|X\|} \leq p(n)K\varepsilon$$

with some low–order polynomial $p(n)$ of n. This shows the importance of the condition number in the numerical solution of Riccati equation. Consider the perturbed Riccati equation

$$(A+\Delta A)^T(X+\Delta X)+(X+\Delta X)(A+\Delta A)+C+\Delta C-(X+\Delta X)(D+\Delta D)(X+\Delta X) = 0 \quad (2)$$

and set $A_c = A - DX$. Subtracting (1) from (2) and neglecting the second– and higher–order terms in ΔX (i.e., using a first–order perturbation analysis) we obtain a Lyapunov equation in ΔX:

$$A_c^T \Delta X + \Delta X A_c = -\Delta C - (\Delta A^T X + X\Delta A) + X\Delta D X = 0. \quad (3)$$

Using the vectorized version of the equation, we obatin that

$$\|\mathrm{Vec}(M)\|_2 = \|M\|_F$$

and equation (3) can be written as

$$\begin{aligned}(I_n \otimes A_c^T + A_c^T \otimes I_n)\mathrm{Vec}(\Delta X) \quad &= \quad -\mathrm{Vec}(\Delta C) - \\ &\quad (I_n \otimes X + (X \otimes I_n)W)\mathrm{Vec}(\Delta A) + \\ &\quad (X \otimes X)\mathrm{Vec}(\Delta D)),\end{aligned} \quad (4)$$

where we use the representations

$$\begin{aligned}\mathrm{Vec}(\Delta A^T) &= W\mathrm{Vec}(\Delta A) \\ \mathrm{Vec}(MZN) &= (N^T \otimes M)\mathrm{Vec}(Z)\end{aligned}$$

and W is the so called vec–permutation matrix, such that $\mathrm{Vec}(M^T) = W\mathrm{Vec}(M)$, [28].

Since the matrix A_c is stable, the matrix $I_n \otimes A_c^T + A_c^T \otimes I_n$ is nonsingular and we have that

$$\begin{aligned}\mathrm{Vec}(\Delta X) \quad &= \quad (I_n \otimes A_c^T + A_c^T \otimes I_n)^{-1}(-\mathrm{Vec}(\Delta C) - \\ &\quad (I_n \otimes X + (X \otimes I_n)W)\mathrm{Vec}(\Delta A) + \\ &\quad (X \otimes X)\mathrm{Vec}(\Delta D))\end{aligned} \quad (5)$$

Equation (5) can be written as

$$\mathrm{Vec}(\Delta X) = -[P^{-1}, \ \ Q, \ \ -S]\begin{bmatrix}\mathrm{Vec}(\Delta C) \\ \mathrm{Vec}(\Delta A) \\ \mathrm{Vec}(\Delta D)\end{bmatrix}, \quad (6)$$

where

$$\begin{array}{rcl} P & = & I_n \otimes A_c^T + A_c^T \otimes I_n, \\ Q & = & P^{-1}(I_n \otimes X + (X \otimes I_n)W), \\ S & = & P^{-1}(X \otimes X). \end{array}$$

If we set

$$\eta = \max\left\{\|\Delta A\|_F/\|A\|_F,\ \|\Delta C\|_F/\|C\|_F,\ \|\Delta D\|_F/\|D\|_F\right\},$$

then it follows from (6) that

$$\|\Delta X\|_F/\|X\|_F \leq \sqrt{3} K_F \eta,$$

where

$$K_F = \|\left[P^{-1},\ Q,\ S\right]\|_2/\|X\|_F$$

is the condition number of (1) using Frobenius norms. The computation of K_F requires the construction and manipulation of $n^2 \times n^2$ matrices which is not practical for large n. Furthermore, the computation of the condition number of the Riccati equation involves the solution matrix X, so that the condition number can be determined only after solving the equation.

In Appendix 1 we give the MATLAB [1] m-file `cndricc.m` for the computation of the condition number K_F.

Since the computation of the exact condition number is a difficult task, it is useful to derive approximations of K that can be obtained cheaply.

Rewrite equation (3) as

$$\Delta X = -\Omega^{-1}(\Delta C) - \Theta(\Delta A) + \Pi(\Delta D), \tag{7}$$

where

$$\begin{array}{rcl} \Omega(Z) & = & A_c^T Z + Z A_c, \\ \Theta(Z) & = & \Omega^{-1}(Z^T X + XZ), \\ \Pi(Z) & = & \Omega^{-1}(XZX) \end{array}$$

are linear operators in the space of $n \times n$ matrices, which determine the sensitivity of X with respect to the perturbations in C, A, D, respectively. Based on (7) it was suggested in [17] to use the approximate condition number

$$K_B := \frac{\|\Omega^{-1}\|\|C\| + \|\Theta\|\|A\| + \|\Pi\|\|D\|}{\|X\|}, \tag{8}$$

where

$$\begin{array}{rcl} \|\Omega^{-1}\| & = & \max_{Z\neq 0} \frac{\|\Omega^{-1}(Z)\|}{\|Z\|} \\ \|\Theta\| & = & \max_{Z\neq 0} \frac{\|\Theta(Z)\|}{\|Z\|} \\ \|\Pi\| & = & \max_{Z\neq 0} \frac{\|\Pi(Z)\|}{\|Z\|} \end{array}$$

are the corresponding induced operator norms. Note that the quantity

$$\|\Omega^{-1}\|_F = \max_{Z\neq 0} \frac{\|Z\|_F}{\|A_c^T Z + Z A_c\|_F} = \frac{1}{\mathrm{sep}_F(A_c^T, -A_c)}$$

[1] MATLAB is a trademark of The Mathworks, Inc

where

$$\mathrm{sep}_F(A_c^T, -A_c) := \min_{Z \neq 0} \frac{\|A_c^T Z + Z A_c\|_F}{\|Z\|_F} = \sigma_{\min}(I_n \otimes A_c^T + A_c^T \otimes I_n)$$

is connected to the sensitivity of the Lyapunov equation

$$A_c^T X + X A_c = -C$$

(see [30]).

Comparing (5) and (7) we obtain that

$$\begin{aligned} \|\Omega^{-1}\|_F &= \|P^{-1}\|_2, \\ \|\Theta\|_F &= \|Q\|_2, \\ \|\Pi\|_F &= \|S\|_2. \end{aligned} \tag{9}$$

Cheap approximations of $\|\Omega^{-1}\|_2$, $\|\Theta\|_2$ and $\|\Pi\|_2$ can be computed in the following way [34]. Let H_k be the solution of the Lyapunov equation

$$A_c^T H_k + H_k A_c = -X^k$$

for $k = 0, 1, 2$. Then

$$\begin{array}{l} \|\Omega^{-1}\|_2 = \|H_0\|_2, \quad \|\Pi\|_2 = \|H_2\|_2, \\ 2\|H_1\|_2 \le \|\Theta\|_2 \le 2\|H_0\|_2^{1/2}\|H_2\|^{1/2}. \end{array}$$

This result allows to estimate K_B by solving three Lyapunov equations with the same matrix A_c but with different right–hand sides. As it is shown, however, in [34], there exist some cases when the quantity $2\|H_0\|_2^{1/2}\|H_2\|^{1/2}$ may be much larger than $\|\Theta\|_2$, which may lead to a large overestimation of K_B.

The quantities $\|\Omega^{-1}\|_1$, $\|\Theta\|_1$, $\|\Pi\|_1$ can be efficiently estimated by using the norm estimator, proposed in [27, 32] which estimates the norm $\|T\|_1$ of a linear operator T, given the ability to compute Tv and $T^T w$ quickly for arbitrary v and w. This estimator is implemented in the LAPACK subroutine `xLACON` [2], which is called via a reverse communication interface, providing the products Tv and $T^T w$. With respect to the computation of

$$\|\Omega^{-1}\|_F = \|P^{-1}\|_2 = \frac{1}{\mathrm{sep}_F(A_c^T, -A_c)}$$

the use of `xLACON` means to solve the linear equations

$$Py = v, \;\; P^T z = v,$$

where

$$P = I_n \otimes A_c^T + A_c^T \otimes I_n, \quad P^T = I_n \otimes A_c + A_c \otimes I_n,$$

v being determined by `xLACON`. This is equivalent to the solution of the Lyapunov equations

$$\begin{array}{c} A_c^T Y + Y A_c = V \\ A_c Z + Z A_c^T = V \\ \mathrm{Vec}(V) = v, \mathrm{Vec}(Y) = y, \mathrm{Vec}(Z) = z, \end{array} \tag{10}$$

see [6] for a similar approach in estimating the sensitivity of invariant subspaces. The solution of Lyapunov equations can be obtained in a numerically reliable way using the Bartels–Stewart algorithm [10], which first reduces the matrix A_c to Schur triangular form via orthogonal similarity transformations and then solves recursively the triangular Lyapunov equation. Note that in (10) the matrix V is symmetric, which allows a reduction in complexity by operating on vectors v of length $n(n+1)/2$ instead of n^2.

An estimate of $\|\Theta\|_1$ can be obtained in a similar way by solving the Lyapunov equations

$$\begin{array}{c} A_c^TY + YA_c = V^TX + XV \\ A_cZ + ZA_c^T = V^TX + XV. \end{array} \tag{11}$$

To estimate $\|\Pi\|_1$ via `xLACON`, it is necessary to solve the equations

$$\begin{array}{c} A_c^TY + YA_c = XVX \\ A_cZ + ZA_c^T = XVX, \end{array} \tag{12}$$

where the matrix V is again symmetric and we can again work with shorter vectors.

The accuracy of the estimates that we obtain via this approach depends on the ability of `xLACON` to find a right–hand side vector v which maximises the ratios

$$\frac{\|y\|}{\|v\|}, \ \frac{\|z\|}{\|v\|}.$$

when solving the equations

$$Py = v, \ P^Tz = v.$$

As in the case of other condition estimators it is always possible to find special examples when the value produced by `xLACON` underestimates the true value of the corresponding norm by an arbitrary factor. Note, however, that this may happen only in rare circumstances.

Consider finally the condition estimation for the dual Riccati equation

$$AX + XA^T + C - XDX = 0, \tag{13}$$

where $A \in \mathcal{R}^{n\times n}$ and the matrices $C \in \mathcal{R}^{n\times n}$, $D \in \mathcal{R}^{n\times n}$ are symmetric. In this case the approximate condition number K_B of the equation is determined again by (8) where the operators Ω, Θ, Π are given by

$$\begin{array}{rcl} \Omega(Z) & = & A_cZ + ZA_c^T, \\ \Theta(Z) & = & \Omega^{-1}(ZX + XZ^T), \\ \Pi(Z) & = & \Omega^{-1}(XZX), \end{array}$$

with

$$A_c = A - XD.$$

The norms of these operators may be estimated by `xLACON` as shown above for equation (1) which allows to use the same software in estimating the conditioning of (1) and (13).

The following table summarizes the Lyapunov equations which we need to solve in estimating $\|\Omega^{-1}\|_1$, $\|\Theta\|_1$ and $\|\Pi\|_1$.

	Equation (1)	Equation (13)
$\|\Omega^{-1}\|_1$	$A_c^TY + YA_c = V$	$A_cY + YA_c^T = V$
	$A_cZ + ZA_c^T = V$	$A_c^TZ + ZA_c = V$
$\|\Theta\|_1$	$A_c^TY + YA_c = V^TX + XV$	$A_cY + YA_c^T = VX + XV^T$
	$A_cZ + ZA_c^T = V^TX + XV$	$A_c^TZ + ZA_c = VX + XV^T$
$\|\Pi\|_1$	$A_c^TY + YA_c = XVX$	$A_cY + YA_c^T = XVX$
	$A_cZ + ZA_c^T = XVX$	$A_c^TZ + ZA_c = XVX$

3 Numerical algorithms

There are different methods for the numerical solution of the continuous–time Riccati equation, see for exmaple [1, 14, 16, 40, 43, 46]. While the new methods in [1, 14] are specifically designed to make use of the Hamiltonian structure and promise to be the most accurate methods, they have not yet been completely implemented as production software. On the other hand, the Schur method and the matrix sign function method have already been used for many years successfully. They do not make use of the specific structure and have been integrated already in several sofware environments, like the MATLAB control toolbox [42] or the SLICOT library [13]. In this report we discuss only the latter two methods, since their current implementations are widely available.

The Schur method [39, 40] is based on the computation of an orthonormal basis of the invariant subspace associated with the stable eigenvalues of the Hamiltonian matrix

$$H = \begin{bmatrix} A & -D \\ -C & -A^T \end{bmatrix}. \tag{14}$$

This may be done without too much extra work using the high–quality routines from LAPACK which implement the QR method for the reduction to Schur form followed by appropriate reordering of this form.

Suppose that H has no eigenvalues on the imaginary axis and let $U \in \mathcal{R}^{2n\times 2n}$ be an orthogonal transformation matrix that reduces H to real Schur form

$$T = U^T H U = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix},$$

where $T_{ij} \in \mathcal{R}^{n\times n}$ and T_{11}, T_{22} are quasi upper-triangular with 1×1 or 2×2 diagonal blocks. It is always possible to find an additional orthogonal transformation which arranges the diagonal blocks of T, such that all eigenvalues of T_{11} have negative real parts [7]. Partitioning U accordingly into four $n\times n$ blocks,

$$U = \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix},$$

we obtain the unique non–negative definite solution of the continuous-time Riccati equation as the solution of the linear system $XU_{11} = U_{21}$, i.e.,

$$X = U_{21}U_{11}^{-1}. \tag{15}$$

The numerical properties of the Schur method are well analysed [45, 36, 12]. The method essentially consists of two parts, the transfomation to Schur form, which via the use of the QR-algorithm can be implemented in a numerically backward stable way and the solution of linear system (15). First of all it should be noted that, since the Schur method does not respect the Hamiltonian structure, it is not strongly backwards stable, i.e., the resulting invariant subspace is only the invariant subspace of a nearby matrix which is not Hamiltonian.

Furthermore, even if the solution of the Riccati equation is well-conditioned, the solution of the Hamiltonian eigenvalue problem or the solution of the linear system may be ill-conditioned. This was demonstrated in [45] by given some well-conditioned, low order examples of Riccati equations, for which the Schur method does not give accurate results.

Specifically, the analysis done in [45] shows that if

$$\delta = \mathrm{sep}_F(T_{11}, T_{22}) - c(n)\varepsilon\|H\|_2 \geq 0$$

and

$$c(n)\varepsilon\|H\|_2^2(1 + c(n)\varepsilon) \leq \frac{1}{4}\delta^2,$$

where $c(n)$ is some low order polynomial in n, then the solution $\bar{X}$ of the Riccati equation computed by the Schur method satisfies

$$\frac{\|\bar{X} - X\|_2}{\|X\|_2} \leq \frac{2c(n)\varepsilon}{\delta}\left(1 + \frac{1}{\|X\|_2}\right)\|H\|_2\|\bar{U}_{11}^{-1}\|_2 + \frac{\|\bar{X} - X^0\|_2}{\|X\|_2}, \tag{16}$$

where

$$X^0 = \bar{U}_{21}\bar{U}_{11}^{-1}$$

is the exact result computed from the computed orthonormal basis of the stable invariant subspace of H. It may happen that the linear system is ill-conditioned or it may happen that the computation of the stable invariant subspace is an ill-conditioned problem, even if the solution is Riccati equation is a well-condtioned problem. The bound (16) shows that the error in $\bar{X}$ will be large if $\|\bar{U}_{11}^{-1}\|$ is large.This may happen, even for well conditioned examples, when, for instance, $\|D\| << \|C\|$ and small perturbations in the Hamiltonian matrix correspond to large perturbations in the original data. We present such examples in Section 7. As was shown in [36] this difficulty can be avoided by an appropriate scaling of the coefficient matrices which ensures that $\|X\|_2 = 1$.

The other widely used method for the solution of (1) is the sign function method, [48, 19, 38, 24, 49, 50]. Suppose again that the matrix Hamiltonian matrix (14) associated with the Riccati equation has no eigenvalues on the imaginary axis and let

$$H = V \begin{bmatrix} J_- & 0 \\ 0 & J_+ \end{bmatrix} V^{-1}$$

be the Jordan decomposition of H, where the eigenvalues of the $n \times n$ submatrix J_- are the stable eigenvalues of H. The matrix sign function of H is then defined as

$$\mathrm{sign}(H) = V \begin{bmatrix} -I_n & 0 \\ 0 & I_n \end{bmatrix} V^{-1}. \tag{17}$$

Then the matrix

$$P_- = \frac{1}{2}(I_{2n} - \mathrm{sign}(H))$$

is the spectral projector on the invariant subspace corresponding to the stable eigenvalues of H and, since we have assumed that H has exactly n eigenvalues with negative real part, it follows that P_- is a real matrix whose rank is equal to n.

Let $P_- = QR\Pi$ be the rank revealing QR decomposition of P_- [26], where Q is orthogonal, R is upper triangular , and Π is a permutation matrix chosen so that the leading n columns of Q span the image of P_-. Then Q yields the spectral decomposition [5]

$$Q^T H Q = \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix} \tag{18}$$

where the eigenvalues of the matrix H_{11} have negative real parts. Also we have that

$$Q^T \mathrm{sign}(H) Q = \left[\begin{array}{cc} -I_n & Y \\ 0 & I_n \end{array} \right],$$

where Y is the (unique) solution of the Sylvester equation

$$H_{11}Y - YH_{22} = -2H_{12}.$$

If the matrix Q is partitioned as

$$Q \equiv \left[\begin{array}{cc} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{array} \right],$$

where each block Q_{ij} is of dimension $n \times n$, then the columns of $\left[\begin{array}{c} Q_{11} \\ Q_{21} \end{array} \right]$ span the stable invariant subspace of H and the non–negative solution of the Riccati equation is given by the solution of the linear system $XQ_{11} = Q_{21}$, i.e.,

$$X = Q_{21}Q_{11}^{-1}.$$

Due to rounding errors, instead of (18) in finite arithmetic one obtains

$$Q^T H Q = \left[\begin{array}{cc} H_{11} & H_{12} \\ E_{21} & H_{22} \end{array} \right], \tag{19}$$

where the quantity $\|E_{21}\|/\|H\|$ measures the backward error in the computed spectral decomposition.

The matrix sign function of H may be computed efficiently using the following simple Newton iteration [19, 35].

Set	$S_0 = H$;
For	$j = 0, 1, \ldots$ until convergence or $j > j_{max}$ do
	$S_{j+1} = \frac{1}{2}(\gamma S_j + \frac{1}{\gamma} S_j^{-1})$
	if $\|S_{j+1} - S_j\|_1 \leq tol \|S_j\|_1$, exit
End	

Here tol is the stopping criterion for the iteration (say, $tol = 100n\varepsilon$), and j_{max} limits the maximum number of iterations (say $j_{max} = 50$). The scaling factor γ is chosen to accelerate the convergence of the iteration [9, 8, 35]. The computation of the matrix S_j^{-1} may be reduced to the inversion of a symmetric matrix using the properties of the Hamiltonian matrix [15, 19, 37].

The Newton iteration is globally and ultimately quadratic convergent [37] but the initial convergence may be slow. Different types of scalings which accelerate the convergence are investigated in [35].

The matrix sign function method has the advantage that it is easily constructed from a small set of highly parallelizable matrix building blocks, including matrix multiplication, QR decomposition and matrix inversion. This makes it preferable to the other methods in the parallel solution of high order Riccati equations. The numerical properties of the matrix sign function are studied in [18, 20, 4]. As was shown in [20], the matrix sign function and the projector P_- may be significantly more ill-conditioned than the stable invariant subspace of H and the stable invariant subspace of $\mathrm{sign}(H)$ is never significantly more ill-conditioned than the corresponding invariant subspace of H. This is confirmed by the numerical experiments described in Section 7.

The analysis of [4] is based on the expectation that the computed matrix sign function is of half the possible precision so that the backward error $\|E_{21}\|/\|H\|$ is of order $\sqrt{\varepsilon}\|\mathrm{sign}(H)\|$. We note that in all our experiments involving computation of the matrix sign function of matrices of order 300, the relative backward error did not exceed the value 50ε which does not confirm the expectations on the backward error. The error analysis for the Newton iteration suggests [18, 20] that this iteration may be inaccurate when the matrix S_j is ill-conditioned. To overcome this difficulty the Newton iteration can be carried out with a shift along the imaginary line [20].

To summarize the discussion on the two most widely used methods for the solution of the continuous-time algebraic Riccati equation, we have seen that both methods may face some numerical difficulties and a possible loss in accuracy in specific examples even when the solution of the Riccati equation is a well-conditioned problem. It is therefore necessary to develop new methods which overcome these difficulties, and although significant progress towards this goal has been made in recent years [1, 14] the problem is not completely solved yet. Another possibility is to combine the implementations of the Schur and matrix sign function method with condition and accuracy estimates and appropriate scalings to guarantee that possible failures like a significant loss of accuracy is minimized and it is detected when it happens. In the next section we discuss these scaling techniques.

4 Block scaling

Consider a similarity transformation of the Hamiltonian matrix (14) with a matrix

$$T = \begin{bmatrix} pI_n & 0 \\ 0 & qI_n \end{bmatrix}; \;\; p \neq 0, \; q \neq 0.$$

As a result one obtains

$$\bar{H} = T^{-1}HT = \begin{bmatrix} A & -\rho D \\ -C/\rho & -A^T \end{bmatrix}$$

where $\rho = q/p$.

In terms of the Riccati equation (1) this transformation leads to the scaled equation

$$A^T\bar{X} + \bar{X}A + C - \bar{X}D\bar{X} = 0 \tag{20}$$

where $\bar{X} = X/\rho$. It has been shown in [36] that this scaling does not change the conditioning of Riccati equation. Depending on the choice of ρ we may scale the solution of (20) in different ways. If, for instance, $\rho = \|X\|$, then $\|\bar{X}\| = 1$ and it has been shown in [36] that this is the “optimal scaling” which ensures backward stability of the Schur method, since it guarantees that none of the two subproblems is significantly more ill-conditioned than the other and than the original problem. A disadvantage of this scaling, however, is the necessity to know $\|X\|$ which requires to solve first the unscaled equation.

Another approach to the scaling of Riccati equation is taken in [44]. It is based on the observation that if $\|C\|$ is large and $\|D\|$ is small, then the implementation of the QR-method for finding the Schur form of the Hamiltonian matrix leads to large errors in $\bar{U}_{11}$ and $\bar{U}_{12}$ which may yield to a significant loss in accuracy in the solution. For this reason it was proposed in [44] to use

$$\rho = \frac{\|C\|}{\|D\|} \tag{21}$$

i.e., $p = \|C\|$, $q = \|D\|$, which means that $\|C/\rho\| = \|D\|$ and $\|\rho D\| = \|C\|$.

The advantage of the "norm-ratio scaling" (21) is that it is cheap to compute, but this scaling will obviously be meaningful only if $\|C\| > \|D\|$. If, however, $\|C\| < \|D\|$ then this scaling leads to even larger errors in comparison with the unscaled equation. For this reason the following modification of this scaling was proposed in [47].

$$\begin{aligned} \rho &= 1 \quad \text{if} \quad \|C\| \leq \|D\|, \\ \rho &= \frac{\|C\|}{\|D\|} \quad \text{if} \quad \|C\| > \|D\|. \end{aligned} \tag{22}$$

This scaling guarantees that $\|C/\rho\| \leq \|\rho D\|$, which improves the numerical properties of both Schur and matrix sign function method. This scaling, of course, may give worse results than the optimal scaling, but in many cases it gives satisfactory results with much less computational effort. On the other hand, several experiments have shown that the scaling (22) may lead to large norms of the scaled Hamiltonian matrix and failure of the QR-algorithm in computing the Schur decomposition. For this reason it was suggested in [12] to use another modification of this scaling as

$$\begin{aligned} \rho &= 1 \quad \text{if} \quad \|C\| \leq \|D\|, \\ \rho &= \sqrt{\frac{\|C\|}{\|D\|}} \quad \text{if} \quad \|C\| > \|D\|. \end{aligned} \tag{23}$$

Other types of scaling may be found in [12, 24].

5 Error estimation

A posteriori error bounds for the computed solution of Riccati equation may be obtained in several ways, see for instance [25, 51]. One of the most efficient and reliable ways to get an estimate of the solution error is to use practical error bounds, similar to the case of solving linear systems of equations [3, 2] and matrix Sylvester equations [31].

Let

$$R = A^T \bar{X} + \bar{X} A + C - \bar{X} D \bar{X}$$

be the exact residual matrix associated with the computed solution $\bar{X}$. Setting $\bar{X} := X + \Delta X$, where X is the exact solution and ΔX is the absolute error in the solution, one obtains

$$R = (A - D\bar{X})^T \Delta X + \Delta X (A - D\bar{X}) + \Delta X D \Delta X.$$

If we neglect the second order term in ΔX, we obtain the linear system of equations

$$\bar{P} \mathrm{Vec}(\Delta X) = \mathrm{Vec}(R),$$

where $\bar{P} = I_n \otimes \bar{A}_c^T + \bar{A}_c^T \otimes I_n$, $\bar{A}_c = A - D\bar{X}$. In this way we have

$$\|\mathrm{Vec}(X - \bar{X})\|_\infty = \|\bar{P}^{-1} \mathrm{Vec}(R)\|_\infty \leq \|\, |\bar{P}^{-1}|\, |\mathrm{Vec}(R)|\, \|_\infty.$$

As it is known [3] this bound is optimal if we ignore the signs in the elements of $\bar{P}^{-1}$ and $\mathrm{Vec}(R)$.

In order to take into account the rounding errors in forming the residual matrix, instead of R we use

$$\bar{R} = fl(C - A^T\bar{X} - \bar{X}A - \bar{X}D\bar{X}) = R + \Delta R,$$

where

$$|\Delta R| \leq \varepsilon(4|C| + (n+4)(|A^T|\,|\bar{X}| + |\bar{X}|\,|A|) + 2(n+1)|\bar{X}|\,|D|\,|\bar{X}|) =: R_\varepsilon.$$

Here we made use of the well known error bounds for matrix addition and matrix multiplication [33].

In this way we have obtained the overall bound

$$\frac{\|X - \bar{X}\|_M}{\|\bar{X}\|_M} \leq \frac{\|\,|P^{-1}|\,(|\mathrm{Vec}(\bar{R})| + \mathrm{Vec}(R_\varepsilon))\|_\infty}{\|\bar{X})\|_M}, \tag{24}$$

where $\|X\|_M = \max_{i,j}|x_{ij}|$.

The numerator in the right hand side of (24) is of the form $\|\,|P^{-1}|r\|_\infty$, and as in [3, 31] we have

$$\begin{aligned} \|\,|\bar{P}^{-1}|r\,\|_\infty &= \|\,|\bar{P}^{-1}|D_R e\|_\infty = \|\,|\bar{P}^{-1}D_R|e\|_\infty \\ &= \|\,|\bar{P}^{-1}D_R|\,\|_\infty = \|\bar{P}^{-1}D_R\|_\infty \end{aligned}$$

where $D_R = \mathrm{diag}(r)$ and $e = (1,1,...,1)^T$. This shows that $\|\,|P^{-1}|r\|_\infty$ can be efficiently estimated using the norm estimator **xLACON** in LAPACK, which estimates $\|Z\|_1$ at the cost of computing a few matrix-vector products involving Z and Z^T. This means that for $Z = \bar{P}^{-1}D_R$ we have to solve a few linear systems involving $\bar{P} = I_n \otimes \bar{A}_c^T + \bar{A}_c^T \otimes I_n$ and $\bar{P}^T = I_n \otimes \bar{A}_c + \bar{A}_c \otimes I_n$ or, in other words, we have to solve several Lyapunov equations $\bar{A}_c^T X + X\bar{A}_c = V$ and $\bar{A}_c X + \bar{A}_c^T = W$. Note that the Schur form of $\bar{A}_c$ is already available from the condition estimation of the Riccati equation, so that the solution of the Lyapunov equations can be obtained efficiently via the Bartels-Stewart algorithm. Also, due to the symmetry of the matrices $\bar{R}$ and R_ε, we only need the upper (or lower) part of the solution of this Lyapunov equations which allows to reduce the complexity by manipulating only vectors of length $n(n+1)/2$ instead of n^2.

The error estimation in the solution of (13) is obtained in a similar way.

6 Software implementation

In this section we discuss some of the implementation issues in the solution of the Riccati equation

$$\mathrm{op}(A^T)X + X\mathrm{op}(A) + C - XDX = 0$$

where $\mathrm{op}(A) = A$ or A^T.

The implementation of the Schur method with condition and accuracy estimates is done by the Fortran 77 double precision driver subroutine `DGRSVX` whose calling sequence is given in Appendix 2. It makes use of the LAPACK subroutines `DGEESX` to reduce the Hamiltonian matrix to upper Schur form and `DGESVX` to solve the linear system $XU_{11} = U_{21}$.

The transformed Lyapunov equation $\mathrm{op}(A_c^T)X + X\mathrm{op}(A_c) = C$ arising in the condition estimation of the Riccati equation in which A_c is in Schur form is solved by the subroutine `DTRLYP` and the auxiliary routine `DLALY2`.

To avoid overflows, instead of estimating the condition number K_B an estimate of the reciprocal condition number

$$\frac{1}{\tilde{K}_B} = \frac{\widetilde{\mathrm{sep}}_1(\bar{A}_c^T, -\bar{A}_c)\|\bar{X}\|_1}{\|C\|_1 + \widetilde{\mathrm{sep}}_1(\bar{A}_c^T, -\bar{A}_c)(\|\tilde{\Theta}\|_1\|A\|_1 + \|\tilde{\Pi}\|_1\|D\|_1)}$$

is determined. Here $\bar{A}_c$ is the computed matrix A_c and the estimated quantities are denoted by $\tilde{}$.

The forward error bound is obtained as described in Section 5, where the corresponding triangular Lyapunov equations are solved by the subroutine `DTRLYP`.

The subroutine `DGRSVX` requires storage space proportional to $9n^2 + 10n$.

The solution of the Riccati equation by the matrix sign function method is carried out by the Fortran 77 double precision driver subroutine `DMSRIC` whose calling sequence is given in Appendix 2. Instead of the nonsymmetric Newton iteration described in Section 3, `DMSRIC` implements the equivalent symmetric iteration [37]

$$\begin{array}{rcl} Z_0 & = & JH \\ Z_{j+1} & = & \frac{1}{2}(\gamma Z_j + \frac{1}{\gamma} J Z_j^{-1} J) \end{array}$$

where

$$J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}.$$

In this case it suffices to compute the upper triangular part of Z_{j+1}. The sign function is recovered at the end from $\mathrm{sign}(H) = -JZ$. The inversion of the symmetric matrix Z_j is performed by the LAPACK subroutines `DSYTRF` and `DSYTRI`. The scaling factor is chosen as

$$\gamma = \sqrt{\|Z_j^{-1}\|_F / \|Z_j\|_F}$$

which gives nearly optimal performance on a wide range of problems [37].

The condition and forward error estimates are determined as in the subroutine `DGRSVX` which requires an additional Schur decomposition of the $n \times n$ matrix A_c. This part of `DMSRIC` is difficult to parallelize and reduces to some extent the efficiency in using the matrix sign function method. In principle, the solution of the n–th order Lyapunov equations related to condition and forward error estimation can be performed also by using a sign function of order $2n$, but this is inefficient in comparison to the using of triangular Lyapunov equations, obtained via the Schur decomposition.

The subroutine `DMSRIC` requires storage space proportional to $9n^2 + 7n$, which is slightly less than the space used by `DGRSVX`.

7 Numerical experiments

In this section we present the results of several numerical experiments which show the behaviour of the Schur method and the sign function method in the solution of several well- and ill-conditioned Riccati equations up to order $n = 150$. All experiments were carried out on an HP 715/33 workstation with relative machine precision $\varepsilon = 2.22 \times 10^{-16}$. In all cases the matrix sign function is computed with tolerance $tol = n\varepsilon$, the maximum number of iterations being set to 60.

In order to have a closed form solution, the test matrices in the Riccati equation are chosen as

$$
\begin{aligned}
A &= ZA_0Z^{-1}, \\
C &= Z^{-T}C_0Z^{-1}, \\
D &= ZD_0Z^T,
\end{aligned}
$$

where A_0, C_0, D_0 are diagonal matrices and Z is a nonsingular transformation matrix. The solution of (1) is then given by

$$X = Z^{-T}X_0Z^{-1}$$

where X_0 is a diagonal matrix whose elements are determined simply from the elements of A_0, C_0, D_0. To avoid large rounding errors in constructing and inverting T this matrix is chosen as [11]

$$Z = H_2SH_1$$

where H_1 and H_2 are elementary reflectors and S is a diagonal matrix,

$$
\begin{aligned}
H_1 &= I_n - 2ee^T/n, \quad e = [1, 1, ..., 1]^T \\
H_2 &= I_n - 2ff^T/n, \quad f = [1, -1, 1, ..., (-1)^{n-1}]^T, \\
S &= \mathrm{diag}(1, s, s^2, ..., s^{n-1}), \quad s > 1.
\end{aligned}
$$

Using different values of the scalar s, it is possible to change the condition number of the matrix Z with respect to inversion,

$$\mathrm{cond}_2(Z) = s^{n-1}.$$

Taking into account the form of Z we obtain that

$$
\begin{aligned}
A &= H_2SH_1A_0H_1S^{-1}H_2, \\
C &= H_2S^{-1}H_1C_0H_1S^{-1}H_2, \\
D &= H_2SH_1D_0H_1SH_2.
\end{aligned}
$$

These matrices are computed easily with relative precision of order ε. Apart from the simplicity of these Riccati equations, their numerical solution presents a difficult task for both the Schur and the sign function method, since the diagonal structure of the equations is not recognized by both methods. On the other hand, the use of such equations in testing the corresponding numerical methods allows to check easily the solution accuracy.

Example 1 The aim of this example is to illustrate the accuracy of condition estimates. For this purpose we computed the quantities related to the exact condition number K_F, by using the m–file `cndricc.m` in MATLAB and compared them with the estimates obtained by the subroutine `DGRSVX`. Since the computation of the exact quantities $\|\Omega^{-1}\|_2$, $\|\Theta\|_2$, $\|\Pi\|_2$ requires large amount of space, we used in this example equations of order 15. (The corresponding Kronecker matrix products have dimensions 225×225 in this case.) The equations are constructed as described above with

$$
\begin{aligned}
A_0 &= \mathrm{diag}(A_1, A_1, A_1, A_1, A_1), \\
C_0 &= \mathrm{diag}(C_1, C_1, C_1, C_1, C_1), \\
D_0 &= \mathrm{diag}(D_1, D_1, D_1, D_1, D_1),
\end{aligned}
$$

where

$$
\begin{aligned}
A_1 &= \operatorname{diag}(-1 \times 10^{-k}, -2, -3 \times 10^{k}), \\
C_1 &= \operatorname{diag}(3 \times 10^{-k}, 5, 7 \times 10^{k}), \\
D_1 &= \operatorname{diag}(10^{-k}, 1, 10^{-k}), \\
X_1 &= \operatorname{diag}(1, 1, 1).
\end{aligned}
$$

The solution is given by

$$X_0 = \operatorname{diag}(X_1, X_1, X_1, X_1, X_1).$$

The conditioning of these equations deteriorates with the increase of k.

The results for different k and $s = 1$ are shown in Table 1. It is seen that the exact and the estimated quantities are of the same order for each k.

Table 1
Accuracy of condition estimates for Example 1

k	$1/\|\Omega\|_1$	$1/\|\tilde{\Omega}\|_1$	$\|\Theta\|_1$	$\|\tilde{\Theta}\|_1$
0	2.76×10^{0}	1.65×10^{0}	7.26×10^{-1}	5.92×10^{-1}
1	1.44×10^{-1}	2.41×10^{-1}	1.39×10^{1}	7.14×10^{0}
2	1.36×10^{-2}	2.83×10^{-2}	1.47×10^{2}	6.24×10^{1}
3	1.35×10^{-3}	2.84×10^{-3}	1.48×10^{3}	6.24×10^{2}
4	1.35×10^{-4}	2.84×10^{-4}	1.48×10^{4}	6.24×10^{3}
5	1.35×10^{-5}	2.84×10^{-5}	1.48×10^{5}	6.24×10^{4}
6	1.35×10^{-6}	2.84×10^{-6}	1.48×10^{6}	6.23×10^{5}
k	$\|\Pi\|_1$	$\|\tilde{\Pi}\|_1$	K_F	$\tilde{K}_B$
0	3.63×10^{-1}	4.70×10^{-1}	1.72×10^{0}	8.52×10^{0}
1	6.95×10^{0}	4.14×10^{0}	1.34×10^{2}	9.21×10^{2}
2	7.37×10^{1}	3.54×10^{1}	1.34×10^{4}	8.08×10^{4}
3	7.41×10^{2}	3.53×10^{2}	1.34×10^{6}	8.08×10^{6}
4	7.42×10^{3}	3.53×10^{3}	1.34×10^{8}	8.08×10^{8}
5	7.42×10^{4}	3.53×10^{4}	1.34×10^{10}	8.08×10^{10}
6	7.42×10^{5}	3.52×10^{5}	1.34×10^{12}	8.07×10^{12}

Example 2 Consider the solution of well-conditioned Riccati equations of order $n = 150$ constructed, such that the matrices $A_0,\ C_0,\ D_0$ consist of 50 copies of diagonal blocks

$$\begin{aligned} A_1 &= \mathrm{diag}(1 \times 10^k, 2 \times 10^k, 3 \times 10^k), \\ C_1 &= \mathrm{diag}(1 \times 10^{-k}, 1, 1 \times 10^k), \\ D_1 &= \mathrm{diag}(10^{-k}, 10^{-k}, 10^{-k}). \end{aligned}$$

The solution X_0 also consists also of 50 copies of diagonal blocks given by

$$\begin{aligned} X_1 &= \mathrm{diag}(x_1, x_2, x_3), \\ x_i &= (a_{ii} + \sqrt{a_{ii}^2 + c_{ii}d_{ii}})/d_{ii} \end{aligned}$$

where $a_{ii}, c_{ii}, d_{ii},\ i = 1,2,3$ are the corresponding diagonal elements of A_1, C_1, D_1, respectively. Note that the corresponding third order Riccati equation with matrices obtained by using $A_0,\ C_0,\ D_0$ was used in [45] to reveal the loss of accuracy of the unscaled version of the Schur method. This equation is very well conditioned (K_B is of order 1) but in the unscaled version of the Schur method the difference between the norms of the blocks of Hamiltonian matrix increases quickly with k which introduces large errors in the solution.

In Table 2 we show the estimate $\tilde{K}_B$ of K_B, the estimate $ferr$ of the relative forward error and the actual relative error err in the solution for the unscaled Schur method for $s = 1$ and different values of k. In Table 3 we show the corresponding values produced by the matrix sign function method (*iter* is the number of iterations performed). The accuracy of both methods is compared in Figure 1.

Table 2

Accuracy of unscaled Schur method for Example 2

k	K_B	$ferr$	err
0	3.87×10^{0}	1.11×10^{-13}	3.52×10^{-15}
1	3.70×10^{0}	9.75×10^{-13}	3.86×10^{-13}
2	3.80×10^{0}	1.11×10^{-13}	8.23×10^{-11}
3	4.00×10^{0}	5.94×10^{-8}	2.56×10^{-9}
4	4.15×10^{0}	7.07×10^{-7}	3.16×10^{-7}
5	3.00×10^{0}	1.65×10^{-4}	7.63×10^{-5}
6	3.81×10^{0}	1.21×10^{-2}	5.36×10^{-3}

Table 3

Accuracy of unscaled matrix sign function method for Example 2

k	K_B	$iter$	$ferr$	err
0	4.60×10^{0}	5	2.21×10^{-13}	7.18×10^{-15}
1	3.01×10^{0}	6	2.49×10^{-13}	4.03×10^{-14}
2	3.80×10^{0}	6	9.63×10^{-12}	4.57×10^{-12}
3	3.80×10^{0}	6	9.79×10^{-10}	4.19×10^{-10}
4	3.61×10^{0}	6	3.18×10^{-8}	1.38×10^{-8}
5	4.40×10^{0}	6	7.52×10^{-6}	3.67×10^{-6}
6	3.80×10^{0}	6	1.21×10^{-3}	5.77×10^{-4}

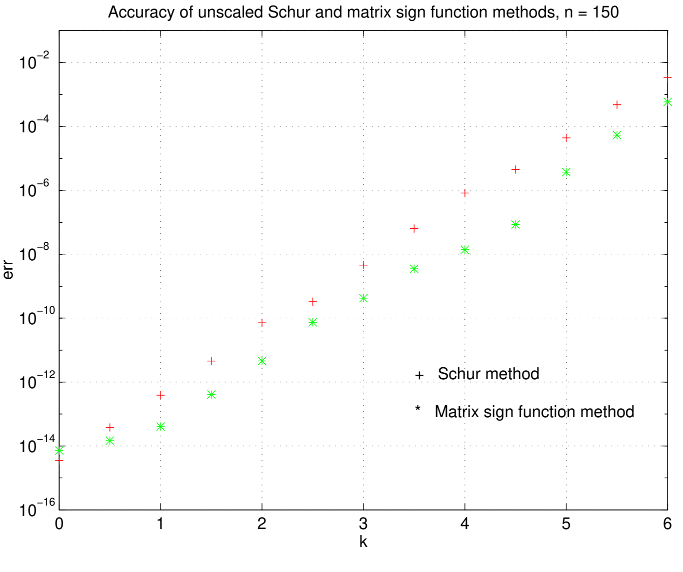

Figure 1: Accuracy of un–scaled Schur and matrix sign function methods

We see from these examples which have condition numbers of order 1 for all k, that the Schur method for the Riccati equation (as a combination of two methods for two subproblems) as well as the sign function method are unstable, or in other words, the splitting of the problem into the computation of the invariant subspace follwed by the solution of a linear system, rewrites a well-conditioned problem as a composition of two subproblems, one of which is ill-conditioned. It is interesting to note that the same inaccuracy occurs if one uses the m–files from [35], which implement the optimal scaling (with respect to the convergence speed) of the matrix sign function method. We note that in this case the matrix sign function converges rapidly for all k but the accuracy of the solution is low for large values of k. The next experiments confirm the observation that the convergence of the matrix sign function method and the accuracy of the obtained solution are not directly connected.

The accuracy of the Schur method for the same example in cases of scaling with $\rho = \sqrt{\|C\|_1/\|D\|_1}$ and $\rho = \|C\|_1/\|D\|_1$ is shown in Table 4 and the corresponding results for the scaled matrix sign function method are given in Table 5.

Table 4

Accuracy of scaled Schur method for Example 2

	$\rho = \sqrt{\|C\|_1/\|D\|_1}$		$\rho = \|C\|_1/\|D\|_1$	
k	$ferr$	err	$ferr$	err
0	1.11×10^{-13}	3.52×10^{-15}	1.11×10^{-13}	3.52×10^{-15}
1	1.45×10^{-13}	1.72×10^{-14}	1.19×10^{-13}	4.44×10^{-15}
2	5.39×10^{-13}	1.92×10^{-13}	1.28×10^{-13}	7.53×10^{-15}
3	3.69×10^{-12}	1.47×10^{-12}	1.21×10^{-13}	6.01×10^{-15}
4	3.64×10^{-11}	1.54×10^{-11}	1.24×10^{-13}	6.88×10^{-15}
5	3.55×10^{-10}	1.32×10^{-10}	1.21×10^{-13}	5.57×10^{-15}
6	3.79×10^{-9}	1.53×10^{-9}	1.22×10^{-13}	5.80×10^{-15}

Table 5

Accuracy of scaled matrix sign function method for Example 2

	$\rho = \sqrt{\|C\|_1/\|D\|_1}$			$\rho = \|C\|_1/\|D\|_1$		
k	$iter$	$ferr$	err	$iter$	$ferr$	err
0	5	2.21×10^{-13}	7.18×10^{-15}	5	2.21×10^{-13}	7.18×10^{-15}
1	6	2.37×10^{-13}	7.91×10^{-15}	6	2.42×10^{-13}	1.08×10^{-14}
2	6	2.48×10^{-13}	3.60×10^{-14}	6	2.37×10^{-13}	1.21×10^{-14}
3	6	6.13×10^{-13}	2.72×10^{-13}	6	2.26×10^{-13}	5.37×10^{-15}
4	6	4.60×10^{-12}	2.19×10^{-12}	6	2.30×10^{-13}	7.69×10^{-15}
5	6	5.66×10^{-11}	2.73×10^{-11}	6	2.28×10^{-13}	5.44×10^{-15}
6	6	4.58×10^{-10}	2.18×10^{-10}	6	2.33×10^{-13}	7.46×10^{-15}

The accuracy of both methods for both types of scaling is presented in Figures 2 and 3. For both methods the full accuracy is obtained by scaling with $\rho = \|C\|_1/\|D\|_1$, which ensures the norm of the scaled solution to be near to one. As it is well-known [36], such scaling removes the ill-conditioning of one of the subproblems and leads to numerical stability in the combined Schur method. It is interesting that the same scaling also leads to better numerical stability in matrix sign function method, which demonstrates again the close connection between both methods. For these example the scaling with $\rho = \sqrt{\|C\|_1/\|D\|_1}$ does not lead to the full possible accuracy. Note that for both the unscaled and scaled

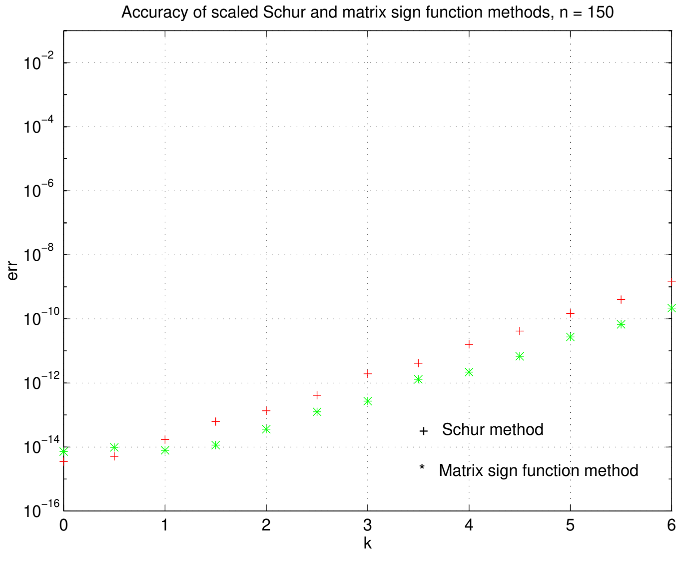

Figure 2: Accuracy of scaled Schur and matrix sign function methods, $\rho = \sqrt{\|C\|_1/\|D\|_1}$

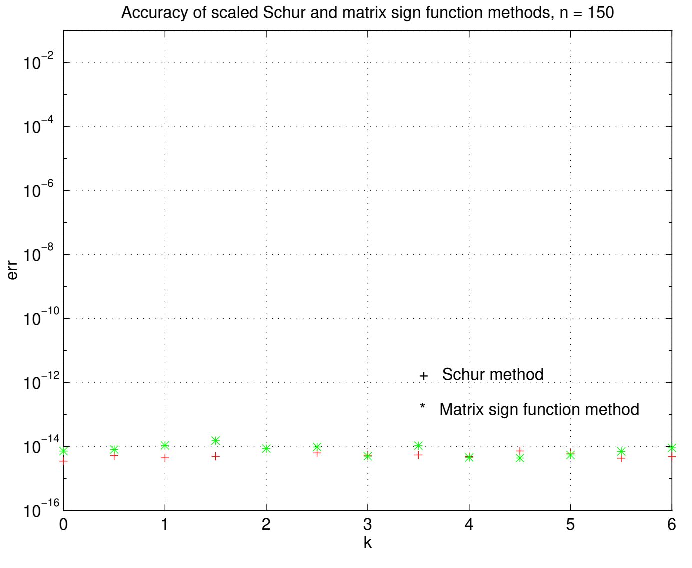

Figure 3: Accuracy of scaled Schur and matrix sign function methods, $\rho = \|C\|_1/\|D\|_1$

versions of the methods the accuracy estimate, as given by the corresponding value of the quantity $ferr$, is close to the actual value of the solution error.

In Figure 4 we show of the error in matrix sign function versus the number of iterations in the case of scaling with $\rho = \|C\|_1/\|D\|_1$. It is necessary to point out that for the same number of iterations as in the unscaled matrix sign function method, this scaling allows to obtain solutions whose error for $k = 6$ is 10^{10} times smaller.

Example 3 Consider a family of Riccati equations of order $n = 150$ obtained for matrices A_0, C_0, D_0 whose diagonal blocks are chosen as

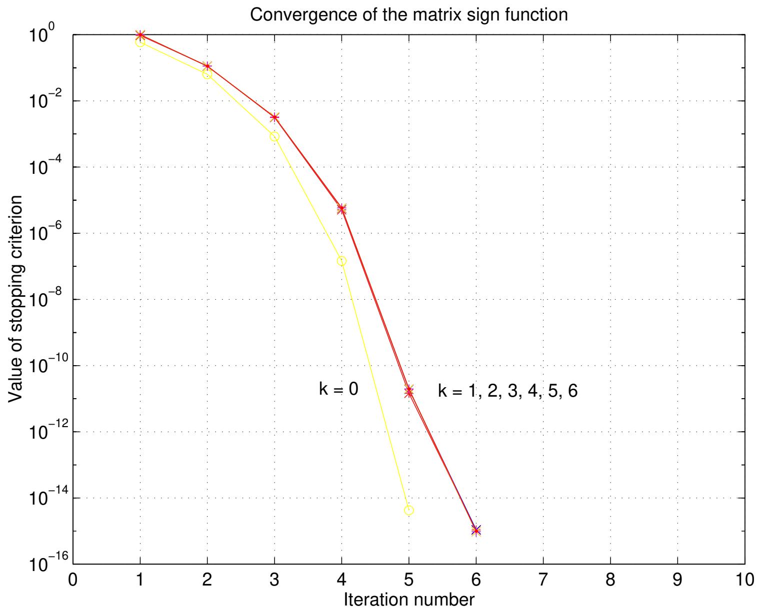

Figure 4: Convergence of the matrix sign function for Example 2, $\rho = \|C\|_1/\|D\|_1$

$$
\begin{aligned}
A_1 &= \operatorname{diag}(1 \times 10^{-k}, 2, 3 \times 10^{k}),\\
C_1 &= \operatorname{diag}(1 \times 10^{k}, 4 \times 10^{2k}, 8 \times 10^{-k}),\\
D_1 &= \operatorname{diag}(10^{-k}, 1, 10^{-k}).
\end{aligned}
$$

These equations become ill-conditioned with the increase of k due to an increase of $\|X\|$ resulting in large values of $\|\Theta\|$ and especially $\|\Pi\|$.

The accuracy of the scaled Schur and matrix sign function method is shown in Tables 6 and 7, respectively, for $s = 1$. It is seen that for both scalings the Schur method failed to produce a solution for $k = 6$ and it failed also for $k = 5$ in the case of the first type of scaling. In the case of the first scaling the failure of the Schur method is due to the non-convergence QR-method, while in the case of the second scaling some of the leading eigenvalues in the Schur decomposition of the Hamiltonian matrix changed their signs after reordering the decomposition. In all these cases the matrix sign function method produced solutions whose accuracy was close to the accuracy predicted by the sensitivity analysis.

Table 6

Accuracy of Schur method for Example 3

	$\rho = \sqrt{\|C\|_1/\|D\|_1}$			$\rho = \|C\|_1/\|D\|_1$		
k	$\hat{K}_B$	$ferr$	err	$\hat{K}_B$	$ferr$	err
0	3.28×10^{0}	1.04×10^{-13}	3.17×10^{-15}	4.43×10^{0}	1.06×10^{-13}	4.57×10^{-15}
1	6.54×10^{1}	8.01×10^{-13}	1.81×10^{-14}	6.14×10^{1}	7.68×10^{-13}	6.48×10^{-15}
2	4.05×10^{2}	6.07×10^{-11}	1.73×10^{-13}	1.00×10^{3}	1.89×10^{-10}	3.82×10^{-13}
3	4.04×10^{3}	1.92×10^{-8}	1.93×10^{-12}	1.26×10^{3}	3.20×10^{-8}	3.46×10^{-11}
4	4.04×10^{4}	1.18×10^{-6}	1.74×10^{-11}	1.25×10^{4}	1.14×10^{-6}	6.82×10^{-9}
5	*	*	*	1.25×10^{5}	1.09×10^{-3}	4.27×10^{-7}
6	*	*	*	**	**	**

* QR-algorithm failure
** Reordering error

Table 7

Accuracy of matrix sign function method for Example 3

	$\rho = \sqrt{\|C\|_1/\|D\|_1}$			
k	$iter$	$\hat{K}_B$	$ferr$	err
0	6	3.71×10^{0}	2.11×10^{-13}	7.11×10^{-15}
1	6	6.64×10^{1}	3.49×10^{-12}	1.83×10^{-14}
2	6	4.05×10^{2}	3.81×10^{-10}	1.39×10^{-13}
3	6	6.70×10^{3}	1.15×10^{-8}	4.22×10^{-13}
4	6	4.04×10^{4}	3.83×10^{-6}	5.34×10^{-12}
5	6	4.04×10^{5}	2.29×10^{-4}	4.39×10^{-11}
6	6	4.04×10^{6}	3.83×10^{-2}	7.54×10^{-10}
	$\rho = \|C\|_1/\|D\|_1$			
k	$iter$	$\hat{K}_B$	$ferr$	err
0	6	5.34×10^{0}	2.23×10^{-13}	1.28×10^{-14}
1	7	9.68×10^{1}	1.56×10^{-12}	3.65×10^{-14}
2	7	1.00×10^{3}	3.82×10^{-10}	7.36×10^{-14}
3	7	1.01×10^{4}	2.28×10^{-8}	8.04×10^{-13}
4	60	6.08×10^{4}	1.16×10^{-6}	5.73×10^{-12}
5	60	1.01×10^{6}	2.29×10^{-4}	4.80×10^{-11}
6	60	1.01×10^{7}	6.48×10^{-2}	3.38×10^{-10}

The convergence of the matrix sign function for $\rho = \|C\|_1/\|D\|_1$ is shown in Figure 5. It is shown that the the stopping criterion is not satisfied for $k = 4, 5, 6$ but nevertheless the obtained accuracy is the maximum possible.

In Figures 6 and 7 we compare the accuracy of Schur and matrix sign function method for both types of scalings. Clearly, for the given example the scaling with $\rho = \sqrt{\|C\|_1/\|D\|_1}$ gives better results, in contrast to Example 2. For all k the matrix sign function method performed better than the Schur method. The detailed comparison of both methods shows that for the scaling $\rho = \|C\|_1/\|D\|_1$ that for the scaling $\rho = \|C\|_1/\|D\|_1$ the quantity $\mathrm{sep}_F(T_{11}, T_{22})$, characterizing the separation between the blocks T_{11} and T_{22} of the Schur form of H, decreases with the increase of k, while the corresponding quantity for the Schur form of the matrix sign function remains approximately equal to 2. This leads to the

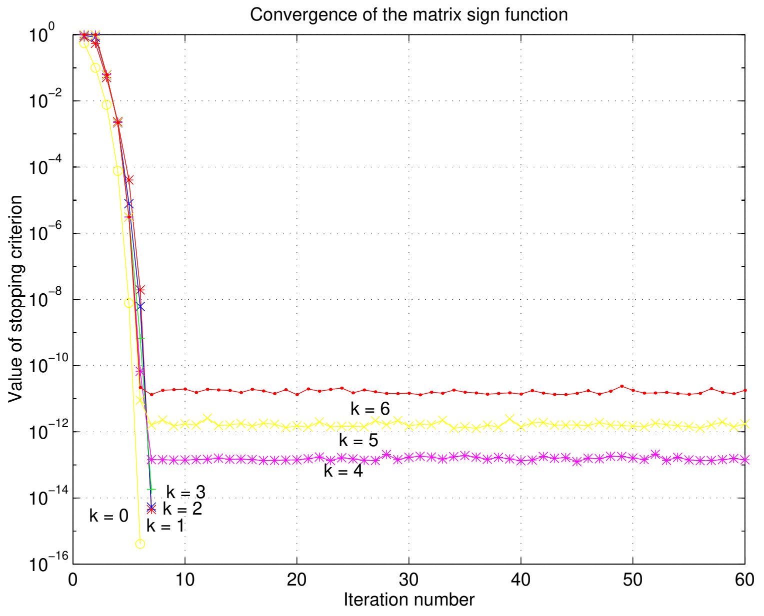

Figure 5: Convergence of the matrix sign function for Example 3

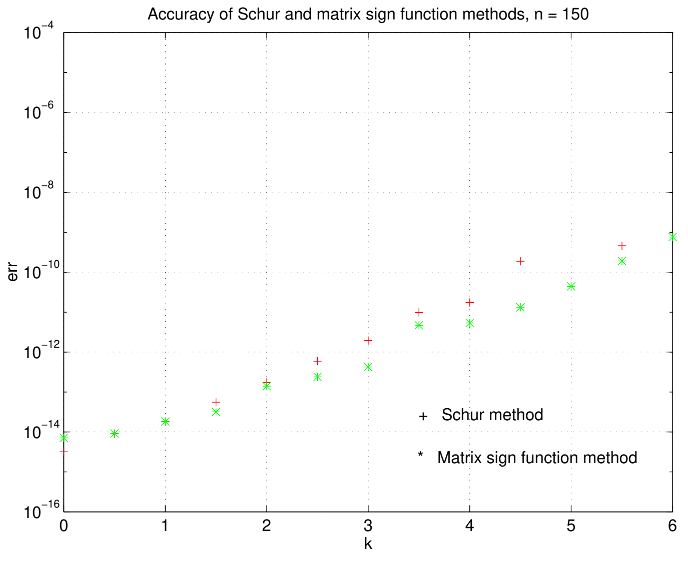

Figure 6: Accuracy of Schur and matrix sign function methods for Example 3, $\rho = \sqrt{\|C\|_1/\|D\|_1}$

conjecture that the orthonormal basis for the stable invariant subspace is computed more accurately by the sign function method in the given case, which was confirmed experimentaly by computing the gap between the subspaces spaned by the corresponding columns of the orthogonal matrices involved in both methods. In Table 8 we show the values of $\text{sep}_F(T_{11}, T_{22})$ and $\text{sep}_F(S_{11}, S_{22})$ (S_{ij} being the corresponding blocks of the Schur form $Q^T\text{sign}(H)Q$ of the matrix sign function) for some larger k for which the Schur method

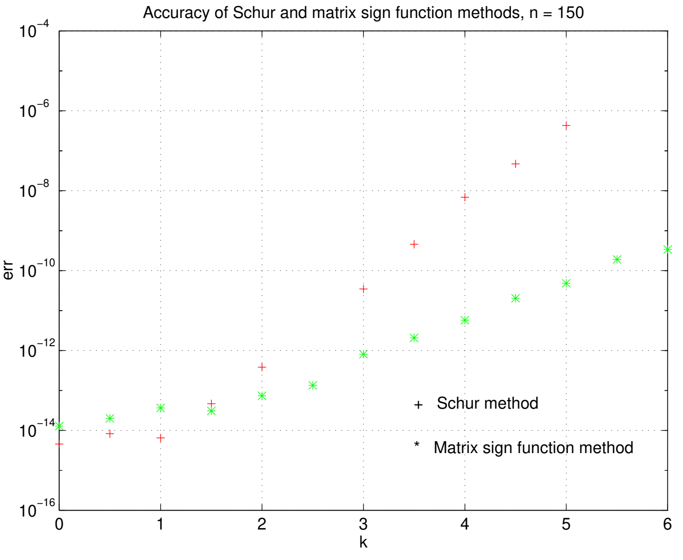

Figure 7: Accuracy of Schur and matrix sign function methods for Example 3, $\rho = \|C\|_1/\|D\|_1$

was able to produce a result, along with the gap $\|P_1 - P_2\|_2$ between the computed stable invariant subspaces by both methods. Here P_1, P_2 are the spectral projectors on the invariant subspaces computed by the Schur and matrix sign function methods, respectively. In all cases when the Schur method produced a solution with large error the gap is large, which means that the QR-method failed to produce an accurate orthonormal basis, in contrast to the matrix sign function. It should be pointed out that in all cases both Schur decompositions of H arising from the Schur method and the matrix sign function method were computed with relative backward errors which were of order ε. Note that for $k = 5.76$ and $k = 6.05$ the matrix sign function method produced a solution for which the error was 10^{10} times smaller than the error in the solution produced by the Schur method.

Table 8

Accuracy of stable invariant subspace computation for Example 3, $\rho = \|C\|_1/\|D\|_1$

	Schur method		Matrix sign function method		
k	$\tilde{\text{sep}}_1(T_{11}, T_{22})$	err	$\tilde{\text{sep}}_1(S_{11}, S_{22})$	err	$\|P_1 - P_2\|_2$
5.70	1.63×10^{-6}	2.52×10^{-4}	2.00×10^{0}	2.76×10^{-10}	1.76×10^{-4}
5.73	1.86×10^{-10}	4.45×10^{-1}	2.00×10^{0}	2.67×10^{-10}	9.75×10^{-1}
5.76	1.98×10^{-7}	7.70×10^{-5}	2.00×10^{0}	2.18×10^{-10}	5.83×10^{-4}
5.85	1.36×10^{-10}	3.06×10^{-1}	1.99×10^{0}	3.04×10^{-10}	9.15×10^{-1}
6.05	1.31×10^{-9}	1.00×10^{0}	1.99×10^{0}	3.74×10^{-10}	9.95×10^{-1}
6.40	1.65×10^{-8}	1.30×10^{-1}	1.86×10^{0}	1.38×10^{-9}	7.81×10^{-1}
6.50	6.94×10^{-9}	1.01×10^{0}	1.90×10^{0}	1.40×10^{-9}	9.97×10^{-1}

Example 4 Consider finally a family of Riccati equations with $n = 150$ for which the diagonal blocks are chosen as in Example 1 as

$$A_1 = \text{diag}(-1 \times 10^{-k}, -2, -3 \times 10^{k}),$$

$$
\begin{aligned}
C_1 &= \mathrm{diag}(3 \times 10^{-k}, 5, 7 \times 10^{k}), \\
D_1 &= \mathrm{diag}(10^{-k}, 1, 10^{k}).
\end{aligned}
$$

As in Example 3 these equations become ill-conditioned with increasing k, but this time the ill-conditioning is due to the decrease of the quantity $\mathrm{sep}_F(A_c^T, -A_c)$.

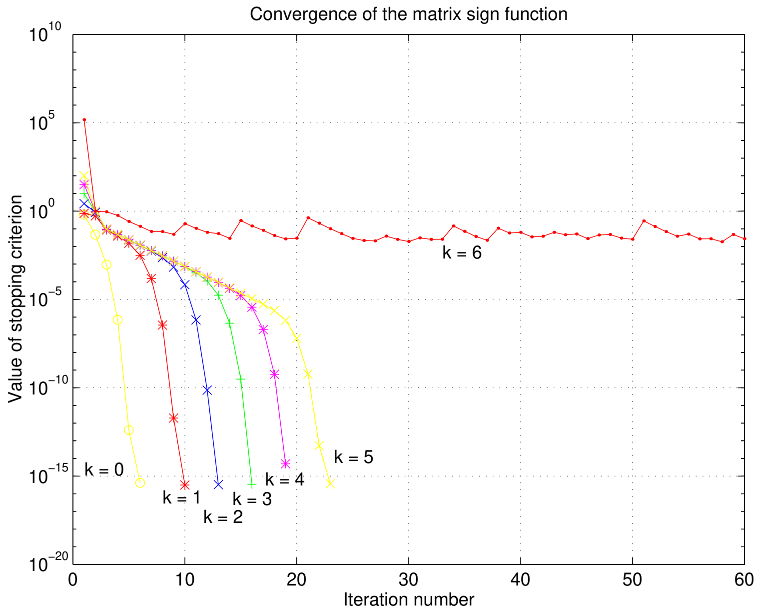

Figure 8: Convergence of the matrix sign function for Example 4

In Tables 9 and 10 we show the results obtained by both methods and both type of scalings for different values of k and $s = 1$. The Schur method failed to produce solutions for $k = 2$ and $k = 6$ in case of scaling with $\rho = \|C\|_1/\|D\|_1$. Note that the condition number of the Riccati equation for $k = 2$ is of order only 10^5. The matrix sign function method produced again reliable solutions for all k, although it did not converge for $k = 6$ (see Figure 8). The accuracy of both methods is compared in Figures 9 and 10. For this example the scaling with $\rho = \sqrt{\|C\|_1/\|D\|_1}$ gives better results, similarly to the case of Example 3.

Table 9

Accuracy of Schur method for Example 4

	$\rho = \sqrt{\|C\|_1/\|D\|_1}$		
k	$\hat{K}_B$	$ferr$	err
0	8.49×10^0	6.54×10^{-14}	6.43×10^{-15}
1	1.30×10^3	2.56×10^{-13}	8.91×10^{-14}
2	1.32×10^5	1.34×10^{-10}	3.41×10^{-11}
3	1.27×10^7	1.41×10^{-8}	2.91×10^{-9}
4	1.27×10^9	4.04×10^{-6}	7.17×10^{-7}
5	1.26×10^{11}	1.47×10^{-3}	3.15×10^{-4}
6	6.41×10^{12}	5.13×10^{-1}	9.97×10^{-2}
	$\rho = \|C\|_1/\|D\|_1$		
k	$\hat{K}_B$	$ferr$	err
0	1.54×10^1	9.31×10^{-14}	2.31×10^{-14}
1	2.81×10^3	8.09×10^{-13}	5.62×10^{-13}
2	$*$	$*$	$*$
3	3.01×10^7	1.98×10^{-6}	4.47×10^{-7}
4	3.01×10^9	2.87×10^{-3}	6.25×10^{-4}
5	1.50×10^{13}	1.00×10^0	1.00×10^0
6	$**$	$**$	$**$

$*$ QR-algorithm failure

$**$ Reordering error

Table 10

Accuracy of matrix sign function method for Example 4

	$\rho = \sqrt{\\|C\\|_1/\\|D\\|_1}$			
k	$iter$	$\tilde{K}_B$	$ferr$	err
0	5	7.56×10^{0}	1.39×10^{-13}	2.31×10^{-14}
1	8	1.31×10^{3}	1.09×10^{-12}	1.76×10^{-14}
2	10	1.33×10^{5}	1.24×10^{-10}	1.84×10^{-12}
3	12	1.28×10^{7}	9.44×10^{-9}	1.42×10^{-10}
4	13	1.27×10^{9}	2.61×10^{-7}	2.49×10^{-9}
5	15	1.26×10^{11}	5.05×10^{-4}	1.01×10^{-6}
6	16	1.26×10^{13}	1.40×10^{-1}	1.52×10^{-4}
	$\rho = \\|C\\|_1/\\|D\\|_1$			
k	$iter$	$\tilde{K}_B$	$ferr$	err
0	6	1.28×10^{1}	1.35×10^{-13}	1.69×10^{-14}
1	10	3.79×10^{3}	3.76×10^{-13}	2.22×10^{-14}
2	13	2.95×10^{5}	2.80×10^{-10}	1.94×10^{-11}
3	16	3.01×10^{7}	1.12×10^{-6}	1.23×10^{-8}
4	19	2.05×10^{9}	5.49×10^{-4}	1.16×10^{-5}
5	23	2.94×10^{11}	3.82×10^{-1}	9.96×10^{-3}
6	60	1.95×10^{13}	1.00×10^{0}	9.96×10^{-1}

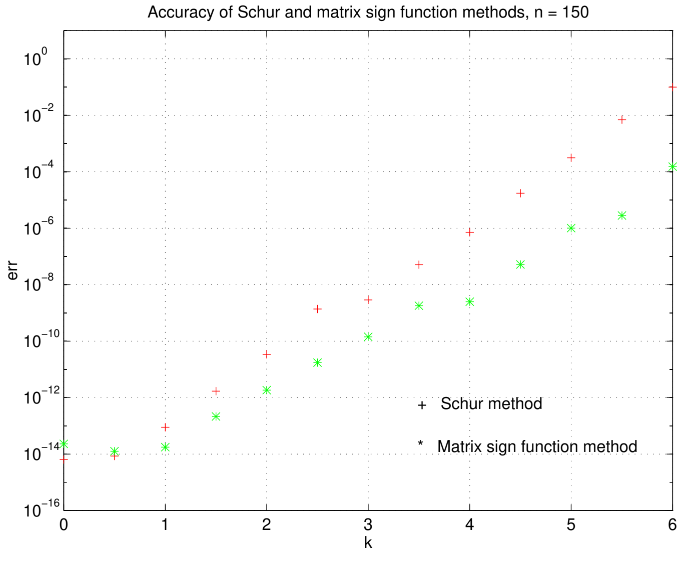

Figure 9: Accuracy of Schur and matrix sign function methods for Example 4, $\rho = \sqrt{\|C\|_1/\|D\|_1}$

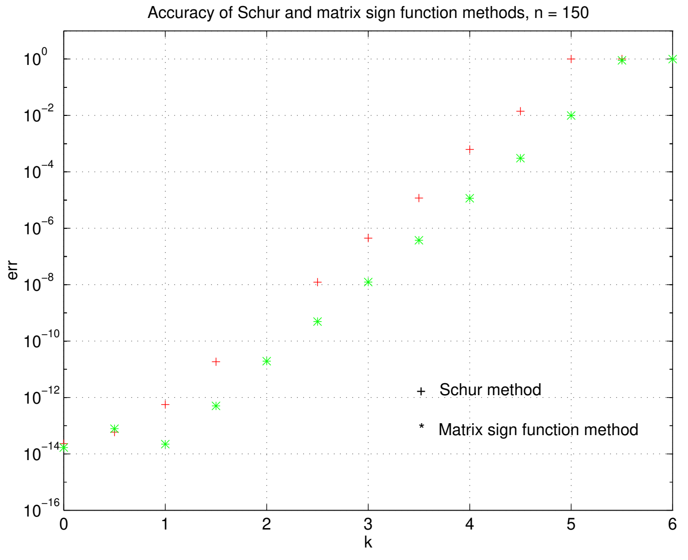

Figure 10: Accuracy of Schur and matrix sign function methods for Example 4, $\rho = \|C\|_1/\|D\|_1$

8 Conclusions

The analysis that we have presented and also the numerical experiments lead to the following conclusions.

- The convergence of the matrix sign function method is not related directly to the accuracy of the Riccati equation solution. There exist well-conditioned examples for which the matrix sign function method converges rapidly but the solution of the Riccati equation is far from the exact solution and there are ill-conditioned examples for which the matrix sign function method converges with an error much larger than ε and nevertheless the solution of the Riccati equation is found with accuracy predicted by the sensitivity analysis. However, the conditioning of the Riccati equation clearly affects the convergence of the matrix sign function.

- In some cases the Schur method did not produce an answer even for some moderately ill-conditioned Riccati equations due to convergence problems with the QR-method and difficulties related to the ordering of the Schur form, independently on the used scaling scheme used. None of the scaling techniques always produces the best answer, so that the best scaling of the Riccati equation in the Schur method remains an open problem. In contrast to the Schur method, the matrix sign function method always produced reliable solution for all examples studied in this report and in many cases this solution is more accurate than the solution obtained by the Schur method. This confirms the conjecture in [20] that the use of matrix sign function does not produce worse results in computing the invariant subspaces than the QR-method. The very accurate results obtained by using the matrix sign function in several cases do not confirm the proposition stated in [4] that the sign function always yields solutions with an accuracy of half the machine precision.

On the basis of these conclusions it is difficult to recommend one of the used scaling schemes. In our code we implemented the scaling with $\rho = \sqrt{\|C\|_1/\|D\|_1}$, since it tends to produce Hamiltonian matrices with smaller norm and performs better in some ill-conditioned cases.

ACKNOWLEDGMENT

The first two authors wish to express their gratitude to the Fakultät für Mathematik, Technische Universität Chemnitz for the hospitality and fine working environment provided during their stay in Chemnitz supported by Volkswagen Stiftung.

References

[1] G.S. Ammar, P. Benner, and V. Mehrmann. A multishift algorithm for the numerical solution of algebraic Riccati equations. *Electr. Trans. Num. Anal.*, 1:33–48, 1993.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, second edition, 1995.

[3] M. Arioli, J.W. Demmel, and I.S. Duff. Solving sparse linear systems with sparse backward error. *SIAM J. Matrix Anal. Appl.*, 10:165–190, 1989.

[4] Z. Bai and J. Demmel. Design of a parallel nonsymmetric eigenroutine toolbox, Part II. Technical Report 95-11, Department of Mathematics, University of Kentucky, Lexington, KY, 1995.

[5] Z. Bai, J. Demmel, J. Dongarra, A. Petitet, H. Robinson, and K. Stanley. The spectral decomposition of nonsymmetric matrices on distributed memory parallel computers. *SIAM J. Sci. Statist. Comput.*, 18:1446–1461, 1997.

[6] Z. Bai, J. Demmel, and A. Mckenney. On computing condition numbers for the nonsymmetric eigenproblem. *ACM Trans. Math. Software*, 19:202–223, 1993.

[7] Z. Bai and J.W. Demmel. On swapping diagonal blocks in real Schur form. *Linear Algebra Appl.*, 186:73–95, 1993.

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

[9] A.Y. Barraud. Investigation autour de la fonction signe d'une matrice - Application à l'équation de Riccati. *R.A.I.R.O. Automatique/Systems Analysis and Control*, 13:335–368, 1979.

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

[11] C.A. Bavely and G.W. Stewart. An algorithm for computing reducing subspaces by block diagonalization. *SIAM J. Numer. Anal.*, 16:359–367, 1979.

[12] P. Benner. *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*. PhD thesis, Fak. f. Mathematik, TU Chemnitz-Zwickau, Chemnitz, FRG, February 1997.

[13] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga. SLICOT-a subroutine library in systems and control theory. *Applied and Computational Control, Signals and Circuits*, 1998. (to appear).

[14] P. Benner, V. Mehrmann, and H. Xu. A new method for computing the stable invariant subspace of a real Hamiltonian matrix. *J. Comput. Appl. Math.*, 86:17–43, 1997.

[15] G.J. Bierman. Computational aspects of the matrix sign function solution to the ARE. In *Proc. 23rd IEEE Conference on Decision and Control*, pages 514–519, Las Vegas, NV, December 1984.

[16] A. Bunse-Gerstner, R. Byers, and V. Mehrmann. Numerical methods for algebraic Riccati equations. In S. Bittanti, editor, *Lecture Notes of the Workshop on "The Riccati Equation in Control, Systems, and Signals", Como, Italy, June 26-28, 1989*, pages 107–116. Pitagora Editrice, Bologna, Italy, 1989.

[17] R. Byers. Numerical condition of the algebraic Riccati equation. *Contemp. Math.*, 47:35–49, 1985.

[18] R. Byers. Numerical stability and instability in matrix sign function based algorithms. In C.I. Byrnes and A. Lindquist, editors, *Computational and Combinatorial Methods in Systems Theory*, pages 185–200. Elsevier (North-Holland), New York, 1986.

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

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

[21] J.J. Dongarra, J. Du Croz, I. Duff, and S. Hammarling. A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 16:1–17, 1990.

[22] J.J. Dongarra, J. Du Croz, S. Hammarling, and R.J Hanson. An extended set of FORTRAN Basic Linear Algebra Subprograms. *ACM Trans. Math. Software*, 14:1–17, 1988.

[23] P. Gahinet and A.J. Laub. Computable bounds for the sensitivity of the algebraic Riccati equation. *SIAM J. Cont. Optim.*, 28:1461–1480, 1990.

[24] J.D. Gardiner. A stabilized matrix sign function algorithm for solving algebraic Riccati equations. *SIAM J. Sci. Statist. Comput.*, 18:1393–1411, 1997.

[25] A.R. Ghavimi and A.J. Laub. Backward error, sensitivity and refinement of computed solutions of algebraic Riccati equations. *Numer. Alg. Appl.*, 2:29–49, 1995.

[26] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, third edition, 1996.

[27] W.W. Hager. Condition estimates. *SIAM J. Sci. Statist. Comput.*, 5:311–316, 1984.

[28] H.V. Henderson and S.R. Searle. The vec-permutation matrix, the vec operator and Kronecker products: A review. *Lin. Multilin. Alg.*, 9:271–288, 1981.

[29] G. Hewer. Existence theorems for positive semidefinite and sign indefinite stabilizing solutions of H_∞ Riccati equations. *SIAM J. Cont. Optim.*, 31:16–29, 1993.

[30] G. Hewer and C. Kenney. The sensitivity of the stable Lyapunov equation. *SIAM J. Cont. Optim.*, 26:321–344, 1988.

[31] N. J. Higham. Perturbation theory and backward error for $AX - XB = C$. *BIT*, 33:124–136, 1993.

[32] N.J. Higham. FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation (Algorithm 674). *ACM Trans. Math. Software*, 14:381–396, 1988.

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

[34] C. Kenney and G. Hewer. The sensitivity of the algebraic and differential Riccati equations. *SIAM J. Cont. Optim.*, 28:50–69, 1990.

[35] 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.

[36] C. Kenney, A.J. Laub, and M. Wette. A stability-enhancing scaling procedure for Schur-Riccati solvers. *Syst. Contr. Lett.*, 12:241–250, 1989.

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

[38] C.S. Kenney, A.J. Laub, and P.M. Papadopoulos. Matrix-sign algorithms for Riccati equations. *IMA J. Math. Contr. Inform.*, 9:331–344, 1992.

[39] A.J. Laub. A Schur method for solving algebraic Riccati equations. *IEEE Trans. Automat. Control*, AC-24:913–921, 1979.

[40] A.J. Laub. Invariant subspace methods for the numerical solution of Riccati equations. In S. Bittanti, A.J. Laub, and J.C. Willems, editors, *The Riccati Equation*, pages 163–196. Springer-Verlag, Berlin, 1991.

[41] C.L. Lawson, R.J. Hanson, D.R. Kincaid, and F.T. Krogh. Basic Linear Algebra Subprograms for FORTRAN usage. *ACM Trans. Math. Software*, 5:308–323, 1979.

[42] The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, Mass, 01760. *The MATLAB Control Toolbox*, 1990.

[43] 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, 1991.

[44] P. Pandey. On scaling an algebraic Riccati equation. In *Proc. 1993 American Control Conference*, pages 1583–1587, San Francisco, CA, June 1993.

[45] P.Hr. Petkov, N.D. Christov, and M.M. Konstantinov. On the numerical properties of the Schur approach for solving the matrix Riccati equation. *Syst. Contr. Lett.*, 9:197–201, 1987.

[46] P.Hr. Petkov, N.D. Christov, and M.M. Konstantinov. *Computational Methods for Linear Control Systems.* Prentice-Hall, Hemel Hempstead, Herts, UK, 1991.

[47] P.Hr. Petkov, M.M. Konstantinov, D.W. Gu, and I. Postlethwaite. Numerical issues in the solution of continuous-time matrix algebraic Riccati equations. Technical Report 96–13, Control Systems Research, Department of Engineering, Leicester University, Leicester, UK, May 1996.

[48] 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).

[49] V. Sima. Algorithm SIGNM Solving continuous-time algebraic Riccati equations using matrix sign function method. *Stud. Res. Comp. Inf.*, 1:335–355, 1992.

[50] V. Sima. *Algorithms for Linear-Quadratic Optimization.* Marcel Dekker, Inc., New York, 1996.

[51] J.-G. Sun. Residual bounds of approximate solutions of the algebraic Riccati equation. *Numer. Math.*, 76:249–263, 1997.

Appendix 1. MATLAB m-file for determining the exact condition number of continuos–time Riccati equation

```
function [cond,Omega,Theta,Pi] = cndricc(A,C,D,X);
%CNDRICC Quantities related to the conditioning of the
%        continuous-time matrix algebraic Riccati equation
%                   A'*X + X*A + C - X*D*X = 0.
%        The condition number of Riccati equation is given by
%        cond = norm([Theta*norm(A,'fro'), Omega*norm(C,'fro'),
%                  Pi*norm(D,'fro')])/norm(X,'fro')
%        where Omega, Theta and Pi are defined by
%        Omega = inv(kron(Ac',eye(n)) + kron(eye(n),Ac')),
%        Theta = Omega*(kron(eye(n),X) + kron(X,eye(n))*W),
%        Pi = Omega*kron(X,X), Ac = A-D*X
%        and W is the vec-permutation matrix.
%
%        31.03.1998
%

n = max(size(A));
nora = norm(A,'fro');
norc = norm(C,'fro');
nord = norm(D,'fro');

Ac = A - D*X;
M = kron(Ac',eye(n)) + kron(eye(n),Ac');

Omega = inv(M);

W = 0*eye(n*n);
for i = 1:n,
    for j = 1:n,
    W(j+(i-1)*n,i+(j-1)*n) = 1.;
    end
end

Theta = M\(kron(eye(n),X) + kron(X,eye(n))*W);
Pi = M\kron(X,X);

D1 = norc*Omega;
D2 = nora*Theta;
D3 = nord*Pi;
cond = norm([D1, D2, -D3]) / norm(X,'fro');
```

Appendix 2. Fortran 77 suboutines

DGRSVX	-	Driver subroutine for solving the continous–time matrix algebraic Riccati equation by the Schur method with condition and accuracy estimates
SELNEG	-	Logical function used to select eigenvalues with negative real parts
DMSRIC	-	Driver subroutine for solving the continuous-time matrix algebraic Riccati
	-	equation by the matrix sign function method with condition and accuracy
	-	estimates
DTRLYP	-	Subroutines for solving the triangular continuous–time matrix Lyapunov
DLALY2	-	equation used in the condition and accuracy estimation

```
      SUBROUTINE DGRSVX( TRANA, N, A, LDA, UPLO, C, LDC, D, LDD, X,
     $                   LDX, WR, WI, RCOND, FERR, WORK, LWORK,
     $                   IWORK, BWORK, INFO )
*
*     Tech. University of Sofia and Tech. University of Chemnitz
*     March 31, 1998
*
*     .. Scalar Arguments ..
      CHARACTER          TRANA, UPLO
      INTEGER            INFO, LDA, LDC, LDD, LDX, LWORK, N
      DOUBLE PRECISION   FERR, RCOND
*     ..
*     .. Array Arguments ..
      LOGICAL            BWORK( * )
      INTEGER            IWORK( * )
      DOUBLE PRECISION   A( LDA, * ), C( LDC, * ), D( LDD, * ),
     $                   X( LDX, * ), WI( * ), WORK( * ), WR( * )
*     ..
*
*  Purpose
*  =======
*
*  DGRSVX solves the real matrix algebraic Riccati equation
*
*     transpose(op(A))*X + X*op(A) + C - X*D*X = 0
*
*  where op(A) = A or A**T and C, D are symmetric (C = C**T, D = D**T) .
*  The matrices A, C and D are N-by-N and the solution X is N-by-N .
*
*  Error bound on the solution and a condition estimate are also
*  provided.
*
*  It is assumed that the matrices A, C and D are such that the
*  corresponding Hamiltonian matrix has N eigenvalues with negative
*  real parts.
*
*  Arguments
*  =========
*
*  TRANA   (input) CHARACTER*1
*          Specifies the option op(A):
*          = 'N': op(A) = A    (No transpose)
*          = 'T': op(A) = A**T (Transpose)
*          = 'C': op(A) = A**T (Conjugate transpose = Transpose)
*
*  N       (input) INTEGER
*          The order of the matrix A, and the order of the
*          matrices C, D and X. N >= 0.
```

```
*
*  A       (input) DOUBLE PRECISION array, dimension (LDA,N)
*          The N-by-N matrix A.
*
*  LDA     (input) INTEGER
*          The leading dimension of the array A. LDA >= max(1,N).
*
*  UPLO    (input) CHARACTER*1
*          = 'U':  Upper triangles of C and D are stored;
*          = 'L':  Lower triangles of C and D are stored.
*
*  C       (input) DOUBLE PRECISION array, dimension (LDC,N)
*          If UPLO = 'U', the leading N-by-N upper triangular part of C
*          contains the upper triangular part of the matrix C.
*          If UPLO = 'L', the leading N-by-N lower triangular part of C
*          contains the lower triangular part of the matrix C.
*
*  LDC     (input) INTEGER
*          The leading dimension of the array C. LDC >= max(1,N)
*
*  D       (input) DOUBLE PRECISION array, dimension (LDD,N)
*          If UPLO = 'U', the leading N-by-N upper triangular part of D
*          contains the upper triangular part of the matrix D.
*          If UPLO = 'L', the leading N-by-N lower triangular part of D
*          contains the lower triangular part of the matrix D.
*
*  LDD     (input) INTEGER
*          The leading dimension of the array D. LDD >= max(1,N)
*
*  WR      (output) DOUBLE PRECISION array, dimension (N)
*  WI      (output) DOUBLE PRECISION array, dimension (N)
*          If TRANA = 'N', WR and WI contain the real and imaginary
*          parts, respectively, of the eigenvalues of A - D*X.
*          If TRANA = 'T' or 'C', WR and WI contain the real and
*          imaginary parts, respectively, of the eigenvalues of A - X*D.
*
*  RCOND   (output) DOUBLE PRECISION
*          The estimate of the reciprocal condition number of the
*          Riccati equation.
*
*  FERR    (output) DOUBLE PRECISION
*          The estimated forward error bound for the solution X.
*          If XTRUE is the true solution, FERR bounds the magnitude
*          of the largest entry in (X - XTRUE) divided by the magnitude
*          of the largest entry in X.
*
*  WORK    (workspace) DOUBLE PRECISION array, dimension (LWORK)
*          On exit, if INFO = 0, WORK(1) contains the optimal LWORK.
```

```
*
*  LWORK   (input) INTEGER
*          The dimension of the array WORK. LWORK >= 9*N*N + 4*N +
*          max(1,6*N).
*          For good performance, LWORK must generally be larger.
*
*  IWORK   (workspace) INTEGER array, dimension max(2*N,N*N)
*
*  BWORK   (workspace) LOGICAL array, dimension (2*N)
*
*  INFO    (output) INTEGER
*          = 0: successful exit
*          < 0: if INFO = -i, the i-th argument had an illegal value
*          = 1: the QR algorithm failed to compute the eigenvalues of
*               the Hamiltonian matrix
*          = 2: the eigenvalues of the Hamiltonian matrix could not be
*               reordered because some eigenvalues were too close to
*               separate
*          = 3: after reordering, roundoff changed values of some
*               complex eigenvalues so that leading eigenvalues in
*               the Schur form have no longer negative real parts
*          = 4: the system of linear equations for the solution is
*               singular to working precision
*          = 5: the matrix A-D*X (or A-X*D) can not be reduced to Schur
*               canonical form and condition number estimate and
*               forward error estimate are not computed
*
*  Further Details
*  ===============
*
*  The Riccati equation is solved by the Schur approach [1] implementing
*  a scaling which enhances the numerical stability [4].
*
*  The condition number of the Riccati equation is estimated as
*
*  cond = ( norm(Theta)*norm(A) + norm(inv(Omega))*norm(C) +
*               norm(Pi)*norm(D) ) / norm(X)
*
*  where Omega, Theta and Pi are linear operators defined by
*
*  Omega(Z) = transpose(op(A))*Z + Z*op(A),
*  Theta(Z) = inv(Omega(transpose(op(Z))*X + X*op(Z))),
*  Pi(Z) = inv(Omega(X*Z*X)).
*
*  The program estimates the quantities
*
*  sep(op(A),-transpose(op(A)) = 1 / norm(inv(Omega)),
*
```

```
*  norm(Theta) and norm(Pi) using 1-norm condition estimator.
*
*  The forward error bound is estimated using a practical error bound
*  similar to the one proposed in [3].
*
*  References
*  ==========
*
*  [1] A.J. Laub. A Schur method for solving algebraic Riccati
*      equations. IEEE Trans. Autom. Control, vol. 24, pp. 913-921,
*      1979.
*  [2] A.R. Ghavimi and A.J. Laub. Backward error, sensitivity, and
*      refinment of computed solutions of algebraic Riccati equations.
*      Numerical Linear Algebra with Applications, vol. 2, pp. 29-49,
*      1995.
*  [3] N.J. Higham. Perturbation theory and backward error for AX-XB=C,
*      BIT, vol. 33, pp. 124-136, 1993.
*  [4] P. Petkov, M. Konstantinov, D. Gu and I. Postlethwaite. Numerical
*      issues in the solution of continuous-time matrix algebraic
*      Riccati equations. Rep. 96-13, Dept. of Engineering, Leicester
*      Univ., 1996.
*
*  ======================================================================


      LOGICAL FUNCTION SELNEG( WR, WI )
*
*     Tech. University of Sofia and Tech. University of Chemnitz
*     March 31, 1998
*
*     .. Scalar Arguments ..
      DOUBLE PRECISION WR, WI
*     ..
*
*     Purpose
*     =======
*
*     SELNEG is used to select eigenvalues with negative real parts
*     to sort to the top left of the Schur form of the Hamiltonian
*     matrix in solving matrix algebraic Riccati equations
*
```

```
      SUBROUTINE DMSRIC( TRANA, N, A, LDA, UPLO, C, LDC, D, LDD, X,
     $                   LDX, WR, WI, RCOND, FERR, WORK, LWORK, IWORK,
     $                   INFO )
*
*     Tech. University of Sofia and Tech. University of Chemnitz
*     March 31, 1998
*
*     .. Scalar Arguments ..
      CHARACTER          TRANA, UPLO
      INTEGER            INFO, LDA, LDC, LDD, LDX, LWORK, N
      DOUBLE PRECISION   FERR, RCOND
*     ..
*     .. Array Arguments ..
      INTEGER            IWORK( * )
      DOUBLE PRECISION   A( LDA, * ), C( LDC, * ), D( LDD, * ),
     $                   X( LDX, * ), WI( * ), WORK( * ), WR( * )
*     ..
*
*  Purpose
*  =======
*
*  DMSRIC solves the real matrix algebraic Riccati equation
*
*     transpose(op(A))*X + X*op(A) + C - X*D*X = 0
*
*  where op(A) = A or A**T and C, D are symmetric (C = C**T, D = D**T) .
*  The matrices A, C and D are N-by-N and the solution X is N-by-N .
*
*  Error bound on the solution and a condition estimate are also
*  provided.
*
*  It is assumed that the matrices A, C and D are such that the
*  corresponding Hamiltonian matrix has N eigenvalues with negative
*  real parts.
*
*  Arguments
*  =========
*
*  TRANA   (input) CHARACTER*1
*          Specifies the option op(A):
*          = 'N': op(A) = A    (No transpose)
*          = 'T': op(A) = A**T (Transpose)
*          = 'C': op(A) = A**T (Conjugate transpose = Transpose)
*
*  N       (input) INTEGER
*          The order of the matrix A, and the order of the
*          matrices C, D and X. N >= 0.
*
```

```
*  A        (input) DOUBLE PRECISION array, dimension (LDA,N)
*           The N-by-N matrix A.
*
*  LDA      (input) INTEGER
*           The leading dimension of the array A. LDA >= max(1,N).
*
*  UPLO     (input) CHARACTER*1
*           = 'U':  Upper triangles of C and D are stored;
*           = 'L':  Lower triangles of C and D are stored.
*
*  C        (input) DOUBLE PRECISION array, dimension (LDC,N)
*           If UPLO = 'U', the leading N-by-N upper triangular part of C
*           contains the upper triangular part of the matrix C.
*           If UPLO = 'L', the leading N-by-N lower triangular part of C
*           contains the lower triangular part of the matrix C.
*
*  LDC      (input) INTEGER
*           The leading dimension of the array C. LDC >= max(1,N)
*
*  D        (input) DOUBLE PRECISION array, dimension (LDD,N)
*           If UPLO = 'U', the leading N-by-N upper triangular part of D
*           contains the upper triangular part of the matrix D.
*           If UPLO = 'L', the leading N-by-N lower triangular part of D
*           contains the lower triangular part of the matrix D.
*
*  LDD      (input) INTEGER
*           The leading dimension of the array D. LDD >= max(1,N)
*
*  WR       (output) DOUBLE PRECISION array, dimension (N)
*  WI       (output) DOUBLE PRECISION array, dimension (N)
*           If TRANA = 'N', WR and WI contain the real and imaginary
*           parts, respectively, of the eigenvalues of A - D*X ;
*           if TRANA = 'T' or 'C', WR and WI contain the real and
*           imaginery parts, respectively, of the eigenvalues of A - X*D.
*
*  RCOND    (output) DOUBLE PRECISION
*           The estimate of the reciprocal condition number of the
*           Riccati equation.
*
*  FERR     (output) DOUBLE PRECISION
*           The estimated forward error bound for the solution X.
*           If XTRUE is the true solution, FERR bounds the magnitude
*           of the largest entry in (X - XTRUE) divided by the magnitude
*           of the largest entry in X.
*
*  WORK     (workspace) DOUBLE PRECISION array, dimension (LWORK)
*           On exit, if INFO = 0, WORK(1) contains the optimal LWORK.
*
```

```
*  LWORK   (input) INTEGER
*          The dimension of the array WORK. LWORK >= 9*N*N + 7*N.
*          For good performance, LWORK must generally be larger.
*
*  IWORK   (workspace) INTEGER array, dimension max(2*N,N*N)
*
*  INFO    (output) INTEGER
*          = 0: successful exit
*          < 0: if INFO = -i, the i-th argument had an illegal value
*          = 1: the Hamiltonian matrix has eigenvalues on the imaginary
*               axis, so the solution and error bounds could not be
*               computed
*          = 2: the iteration for the matrix sign function failed to
*               converge after 60 iterations, but an approximate
*               solution and error bounds have been computed
*          = 3: the system of linear equations for the solution is
*               singular to working precision, so the solution and
*               error bounds could not be computed
*          = 4: the matrix A-D*X (or A-X*D) can not be reduced to Schur
*               canonical form and condition number estimate and
*               forward error estimate have not been computed.
*
*  Further Details
*  ===============
*
*  The Riccati equation is solved by the matrix sign function approach
*  [1], [2] implementing a scaling which enhances the numerical
*  stability [4].
*
*  The condition number of the Riccati equation is estimated as
*
*  cond = ( norm(Theta)*norm(A) + norm(inv(Omega))*norm(C) +
*               norm(Pi)*norm(D) ) / norm(X)
*
*  where Omega, Theta and Pi are linear operators defined by
*
*  Omega(Z) = transpose(op(A))*Z + Z*op(A),
*  Theta(Z) = inv(Omega(transpose(op(Z))*X + X*op(Z))),
*  Pi(Z) = inv(Omega(X*Z*X)).
*
*  The program estimates the quantities
*
*  sep(op(A),-transpose(op(A)) = 1 / norm(inv(Omega)),
*
*  norm(Theta) and norm(Pi) using 1-norm condition estimator.
*
*  The forward error bound is estimated using a practical error bound
*  similar to the one proposed in [3].
```

```
*
*  References
*  ==========
*
*  [1] Z. Bai, J. Demmel, J. Dongarra, A. Petitet, H. Robinson, and
*      K. Stanley. The spectral decomposition of nonsymmetric matrices
*      on distributed memory parallel computers. SIAM J. Sci. Comput.,
*      vol. 18, pp. 1446-1461, 1997.
*  [2] R. Byers, C. He, and V. Mehrmann. The matrix sign function method
*      and the computation of invariant subspaces. SIAM J. Matrix Anal.
*      Appl., vol. 18, pp. 615-632, 1997.
*  [3] N.J. Higham. Perturbation theory and backward error for AX-XB=C,
*      BIT, vol. 33, pp. 124-136, 1993.
*  [4] P. Petkov, M. Konstantinov, D. Gu and I. Postlethwaite. Numerical
*      issues in the solution of continuous-time matrix algebraic
*      Riccati equations. Rep. 96-13, Dept. of Engineering, Leicester
*      Univ., UK, 1996.
*
*  ======================================================================
```

```
      SUBROUTINE DTRLYP( TRANA, N, A, LDA, C, LDC, SCALE, INFO )
*
*     Tech. University of Sofia and Tech. University of Chemnitz
*     March 31, 1998
*
*     .. Scalar Arguments ..
      CHARACTER          TRANA
      INTEGER            INFO, LDA, LDC, N
      DOUBLE PRECISION   SCALE
*     ..
*     .. Array Arguments ..
      DOUBLE PRECISION   A( LDA, * ), C( LDC, * )
*     ..
*
*  Purpose
*  =======
*
*  DTRLYP solves the real Lyapunov matrix equation:
*
*         transpose(op(A))*X + X*op(A) = scale*C
*
*  where op(A) = A or A**T,  A is upper quasi-triangular and C is
*  symmetric (C = C**T). A is N-by-N, the right hand side C and the
*  solution X are N-by-N, and scale is an output scale factor,
*  set <= 1 to avoid overflow in X.
*
*  A must be in Schur canonical form (as returned by DHSEQR), that is,
*  block upper triangular with 1-by-1 and 2-by-2 diagonal blocks;
*  each 2-by-2 diagonal block has its diagonal elements equal and its
*  off-diagonal elements of opposite sign.
*
*  Arguments
*  =========
*
*  TRANA   (input) CHARACTER*1
*          Specifies the option op(A):
*          = 'N': op(A) = A    (No transpose)
*          = 'T': op(A) = A**T (Transpose)
*          = 'C': op(A) = A**H (Conjugate transpose = Transpose)
*
*  N       (input) INTEGER
*          The order of the matrix A, and the order of the
*          matrices X and C. N >= 0.
*
*  A       (input) DOUBLE PRECISION array, dimension (LDA,N)
*          The upper quasi-triangular matrix A, in Schur canonical form.
*
*  LDA     (input) INTEGER
```

```
*          The leading dimension of the array A. LDA >= max(1,N).
*
*  C       (input/output) DOUBLE PRECISION array, dimension (LDC,N)
*          On entry, the symmetric N-by-N right hand side matrix C.
*          On exit, C is overwritten by the solution matrix X.
*
*  LDC     (input) INTEGER
*          The leading dimension of the array C. LDC >= max(1,N)
*
*  SCALE   (output) DOUBLE PRECISION
*          The scale factor, scale, set <= 1 to avoid overflow in X.
*
*  INFO    (output) INTEGER
*          = 0: successful exit
*          < 0: if INFO = -i, the i-th argument had an illegal value
*          = 1: A and -A have common or very close eigenvalues;
*                perturbed values were used to solve the equation
*                (but the matrix A is unchanged).
*
*  =====================================================================



      SUBROUTINE DLALY2( LTRAN, T, LDT, B, LDB, SCALE,
     $                   X, LDX, XNORM, INFO )
*
*     Tech. University of Sofia and Tech. University of Chemnitz
*     March 31, 1998
*
*     .. Scalar Arguments ..
      LOGICAL            LTRAN
      INTEGER            INFO, LDB, LDT, LDX
      DOUBLE PRECISION   SCALE, XNORM
*     ..
*     .. Array Arguments ..
      DOUBLE PRECISION   B( LDB, * ), T( LDT, * ), X( LDX, * )
*     ..
*
*  Purpose
*  =======
*
*  DLALY2 solves for the 2 by 2 symmetric matrix X in
*
*         op(T')*X + X*op(T) = SCALE*B,
*
*  where T is 2 by 2, B is symmetric 2 by 2, and op(T) = T or T',
*  where T' denotes the transpose of T.
```

```
*
*  Arguments
*  =========
*
*  LTRAN   (input) LOGICAL
*          On entry, LTRAN specifies the op(T):
*             = .FALSE., op(T) = T,
*             = .TRUE., op(T) = T'.
*
*  T       (input) DOUBLE PRECISION array, dimension (LDT,2)
*          On entry, T contains an 2 by 2 matrix.
*
*  LDT     (input) INTEGER
*          The leading dimension of the matrix T. LDT >= 2.
*
*  B       (input) DOUBLE PRECISION array, dimension (LDB,2)
*          On entry, the 2 by 2 matrix B contains the symmetric
*          right-hand side of the equation.
*
*  LDB     (input) INTEGER
*          The leading dimension of the matrix B. LDB >= 2.
*
*  SCALE   (output) DOUBLE PRECISION
*          On exit, SCALE contains the scale factor. SCALE is chosen
*          less than or equal to 1 to prevent the solution overflowing.
*
*  X       (output) DOUBLE PRECISION array, dimension (LDX,2)
*          On exit, X contains the 2 by 2 symmetric solution.
*
*  LDX     (input) INTEGER
*          The leading dimension of the matrix X. LDX >= 2.
*
*  XNORM   (output) DOUBLE PRECISION
*          On exit, XNORM is the infinity-norm of the solution.
*
*  INFO    (output) INTEGER
*          On exit, INFO is set to
*             0: successful exit.
*             1: T and -T have too close eigenvalues, so T
*                is perturbed to get a nonsingular equation.
*          NOTE: In the interests of speed, this routine does not
*                check the inputs for errors.
*
* =====================================================================
```