# Numerical Solution of Matrix Riccati Equations: A Comparison of Six Solvers [1]

Petko Hr. Petkov[2], Mihail M. Konstantinov[3] Da-Wei Gu[4] and V. Mehrmann[5]

July 1999

[1]This paper presents research results of the European Community BRITE-EURAM III Thematic Networks Programme NICONET and is distributed by the Working Group on Software WGS. *WGS secretariat*: Mrs. Ida Tassens, ESAT - Katholieke Universiteit Leuven, K. Mercierlaan 94, 3001-Leuven-Heverlee, BELGIUM. This report is also available by anonymous ftp from *wgs.esat.kuleuven.ac.be/pub/WGS/REPORTS/nic1999-10.ps.Z*
[2]Department of Automatics, Technical University of Sofia, 1756 Sofia, Bulgaria
[3]University of Architecture & Civil Engineering, 1 Hr. Smirnenski Blv., 1421 Sofia, Bulgaria
[4]Control Systems Research, Department of Engineering, University of Leicester, Leicester LE1 7RH, U.K.
[5]Fakultät für Mathematik, Technische Universität Chemnitz, D-09107 Chemnitz, Germany

**Abstract**

We present results from the evaluation of six solvers intended for the numerical solution of continuous-time matrix algebraic Riccati equations. The solvers include four MATLAB functions from different toolboxes and two Fortran 77 solvers developed by the authors. The comparison implements two benchmark problems each comprising of 1600 6-th order Riccati equations with known solutions. For each solver and each equation we compute the relative forward and backward errors and for two of the solvers we investigate the accuracy of condition and error estimates. Some conclusions concerning the numerical behaviour of the solvers are given.

# 1 Introduction

The numerical solution of matrix Riccati equations plays an important role in the design of control systems. While the solutions of these equations are studied well from theoretical point of view, the numerical properties of various computational methods are still not evaluated in depth. This motivates a further study of several available codes for numerical solution of Riccati equations.

In this report we investigate the numerical behaviour of six solvers intended for the numerical solution of continuous-time matrix algebraic Riccati equations. The report does not aim to compare all available routines for solving Riccati equations at the moment; it aims to compare only the numerical behaviour of some widely used solvers with the behaviour of the solvers which will be included in the SLICOT library [3]. That is why the list of solvers is limited at this stage to four MATLAB [19] [1] functions and two solvers developed recently [21]:

1. The function `are` from the Control Systems Toolbox of MATLAB.

2. The function `aresolv` from the Robust Control Toolbox of MATLAB.

3. The function `care` from the Control Systems Toolbox of MATLAB.

4. The function `ric_schr` from mu-toolbox of MATLAB.

5. An mex-function `ricc` implementing the Schur method with a block–scaling increasing the numerical reliability.

6. An mex-function `ricm` implementating the matrix sign function method with the same scaling.

The last two implementations involve computation of condition estimates and forward error estimates and implement LAPACK [1] and BLAS [18, 7, 6] subroutines for solving the corresponding numerical linear algebra subproblems.

The following notation is used in the report.

- $\mathcal{R}$ – 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 $A$;
- $\|A\|_F = (\sum |a_{ij}|^2)^{1/2})$ – the Frobenius norm;
- $I_n$ – the unit $n \times n$ matrix;
- $A \otimes B$ – the Kronecker product of matrices $A$ and $B$;
- $\varepsilon$ – the roundoff unit of the machine arithmetic.

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

## 2 Conditioning, forward and backward errors

The numerical evaluation of the routines for solving Riccati equations requires a knowledge about the conditioning of the problems solved and involves comparison of the forward and backward errors done by the different solvers.

Consider the continuous–time matrix algebraic Riccati equation

$$A^T X + XA + C - XDX = 0, \tag{1}$$

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.

Let the coefficient matrices $A,\ C,\ D$ in (1) be subject to perturbations $\Delta A,\ \Delta C,\ \Delta D$, respectively, so that instead of the initial data we have the matrices $\tilde{A} = A + \Delta A$, $\tilde{C} = C + \Delta C$, $\tilde{D} = D + \Delta D$.

If small perturbations in the data $\Delta A,\ \Delta C,\ \Delta D$ lead to small variations $\Delta X$ in the solution $\tilde{X} = X + \Delta X$, then we say that the Riccati equation is *well-conditioned* and if these perturbations lead to large variations in the solution then 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 $\Delta C$ and $\Delta D$ are symmetric. It should be pointed out that if $\|\Delta A\|$, $\|\Delta C\|$ and $\|\Delta D\|$ are sufficiently small, then the perturbed solution $\tilde{X} = X + \Delta X$ is well defined [14, 8].

The *condition number of the Riccati equation* is defined as (see [4])

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

For sufficiently small $\delta$ we have (within first order terms)

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

Thus, the condition number is a measure of the maximum sensitivity of the solution to perturbations in the data. If the condition number of the Riccati equation is very large, then it is probable that floating point errors will affect the solution very much and this is independent of the method which is used to solve the equation.

Let $A_c = A - DX$ and let

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

Using a first–order perturbation analysis it is possible to show that

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

where

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

is the condition number of (1) using Frobenius norm,

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

and $W$ is the vec–permutation matrix [11]. Note that

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

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 [12]).

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

The computation of $K_F$ requires the construction and manipulation of $n^2 \times n^2$ matrices which requires a lot of space and computations for large $n$. That is why in practice it is justified to use cheap approximations of the condition number which may be computed at the cost of finding the solution. One possible estimate for the condition number is described in [21] and is implemented in the Fortran solvers described below.

The most important characteristic of the accuracy in solving the Riccati equation is the *relative forward error*

$$\psi(\bar{X}) = \frac{\|\bar{X} - X\|_M}{\|X\|_M} \tag{2}$$

of the computed solution $\bar{X}$ where $\|X\|_M = \max_{ij}|x_{ij}|$. It varies between 0 (the solution is exact) and 1 (the solution has no accurate digits). The disadvantage of using the forward error in the evaluation of Riccati solvers is that it depends not only upon the numerical properties of the method used but also on the conditioning of the problem solved so that the knowledge of this error does not reveal in full the numerical behaviour of the solver implemented. That is why we are interested also in the *relative backward error* of the computed solution $\bar{X}$ which is defined by

$$\eta(\bar{X}) = \min\left\{\epsilon : \frac{\|\Delta A\|_F}{\|A\|_F}, \frac{\|\Delta C\|_F}{\|C\|_F}, \frac{\|\Delta D\|_F}{\|D\|_F} \leq \epsilon\right\} \tag{3}$$

subject to

$$(A + \Delta A)^T \bar{X} + \bar{X}(A + \Delta A) + C + \Delta C - \bar{X}(D + \Delta D)\bar{X} = 0.$$

Thus, the backward error is the smallest relative perturbation in the data such that the perturbed solution of the equation coincides with the computed solution. This error characterizes the backward numerical stability and if it is of the order of the machine precision then the method is considered as perfectly backward stable.

In Appendix 2 we give the MATLAB m-file `backricc` which for a given $\bar{X}$ computes the backward error $\eta(\bar{X})$ implementing the formulas given in [10].

A posteriori forward error bounds for the computed solution of Riccati equation may be obtained in several ways, see for instance [10, 24]. 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 [1] and matrix Sylvester equations [13]. Such error bounds are implemented in the Fortran 77 solvers `ricc` and `ricm` used in the comparison done in this report. The solver `ricc` implements the Schur method [16, 17] with block scaling [21] which enhances the numerical stability while `ricm` implements the matrix sign function method [22, 5, 15, 9, 23]. Both solvers make use of LAPACK [1], Release 3.0, and are implemented as mex-files which makes them easily accessible from MATLAB.

Note that the Riccati solvers implemented in MATLAB do not produce condition estimates and forward error estimates.

## 3 Numerical experiments

In this section we present the results of several numerical experiments which show the behaviour of the corresponding solvers in the solution of several well- and ill-conditioned Riccati equations. For this purpose we use two benchmark collection problems each comprasing of 1600 6-th order Riccati equations with different conditioning. All experiments are done on PC with Pentium II processor on 333 MHz using MATLAB v. 5.3.0.10183 (R11) with relative precision $\varepsilon = 2.22 \times 10^{-16}$.

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

$$\begin{aligned} A &= TA_0T^{-1}, \\ C &= T^{-T}C_0T^{-1}, \\ D &= TD_0T^T, \end{aligned}$$

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

$$X = T^{-T}X_0T^{-1}$$

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 [2]

$$T = H_2SH_1$$

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

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

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

$$\operatorname{cond}_2(T) = s^{n-1}.$$

Taking into account the form of $T$ we obtain that

$$\begin{aligned} A &= H_2 S H_1 A_0 H_1 S^{-1} H_2, \\ C &= H_2 S^{-1} H_1 C_0 H_1 S^{-1} H_2, \\ D &= H_2 S H_1 D_0 H_1 S H_2. \end{aligned}$$

These matrices are computed easily with relative precision of order $\varepsilon$. 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 underlying 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.

**Benchmark problem 1**

The equations are constructed as described above with

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

where

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

The solution $X_0$ consists of two copies of diagonal blocks given by

$$\begin{aligned} X_1 &= \mathrm{diag}(x_1, x_2, x_3), \\ x_i &= (a_{ii} + \sqrt{a_{ii}^2 + c_{ii} d_{ii}})/d_{ii} \end{aligned} \tag{4}$$

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 Riccati equation with matrices $A_0,\ C_0,\ D_0$ was used in [20] to reveal the loss of accuracy of the unscaled Schur method. These equations are well conditioned ($K_F$ is of order 1) for small $s$ 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 Fig. 1 we give the relative residual

$$\|R\|_F = \frac{\|A^T X + XA + C - XDX\|_F}{2\|A\|_F \|X\|_F + \|C\|_F + \|D\|_F \|X\|_F^2}$$

as a function of the free parameters $k$ and $s$. The magnitude of this residual reflects the accuracy of the approximate solution $X$, computed in (4). Since the maximum value of $\|R\|_F$ is close to $\varepsilon$, it is justified to accept $X$ as the exact solution in the floating point arithmetic under consideration.

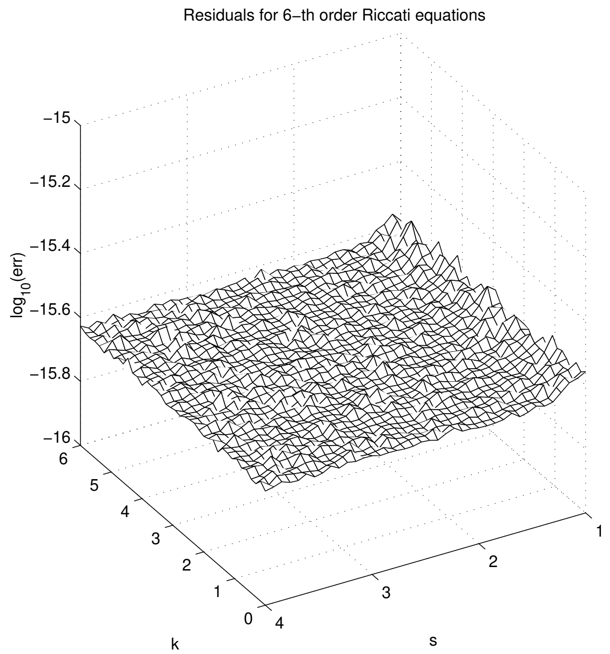


Figure 1: Relative residuals for the Riccati equations from Problem 1

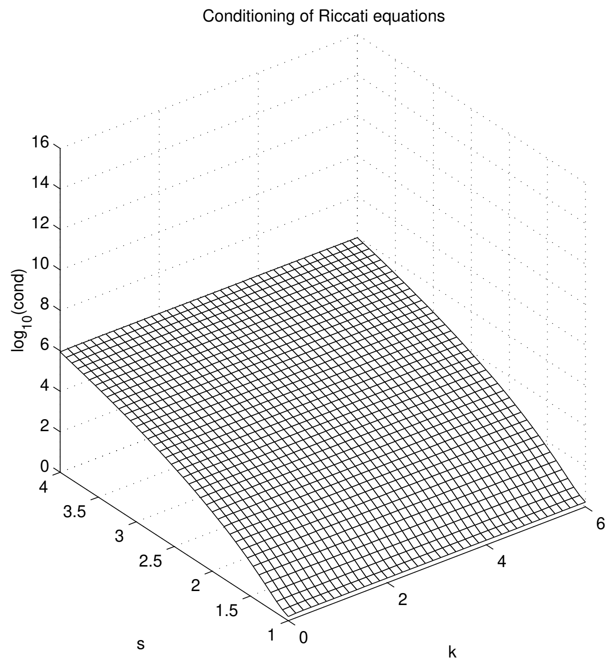


Figure 2: Conditioning of the Riccati equations from Problem 1

In Fig. 2 we show the conditioning of the corresponding Riccati equations for the same values of $k$ and $s$. The condition number is computed by the function `cndricc` given in Appendix 1. Practically, the conditioning does not depend on $k$ and the maximum value of $K_F$ is of order $10^6$.

In Fig. 3 and Fig. 4 we show the forward errors and the backward errors, respectively, for the function `are` from MATLAB. The forward errors increase with $s$ which is explained by the increase in the condition number of the Riccati equation. However, the forward error increases also with the increase of $k$ which is a sign of potential numerical instability. In fact, for $k = 6$ and $s = 4$ the forward error is of order $10^{-3}$, but from the sensitivity analysis we can expect errors of order only $K_F\varepsilon \approx 10^{-10}$. The numerical instability of the method implemented in `are` is obvious from Fig. 4 where we show the corresponding backward errors for the same collection of examples. For large values of $k$ the backward error approaches $10^{-3}$ which shows that the computation of the solution is equivalent to introducing perturbations of the same size in the data. This is a result from the usage of the unscaled Schur method which is potentialy numericaly unstable.

The forward and backward errors for the function `aresolv`, obtained by using the Schur method option, are given in Fig. 5 and Fig. 6. Both errors are slightly smaller than in the case of the function `are` but this function is obviously also numericaly unstable.

The numerical behaviour of the function `care`, as illustrated in Fig. 7 and Fig. 8, is much better. The forward errors are slightly larger than expected from the conditioning, but the backward errors remain less than $10^{-10}$ so that the corresponding implementation could be accepted as conditionally stable. These properties of the `care` are due to the block-scaling of the Hamiltonian matrix which allows to avoid the instability of the original Schur method.

In Fig. 9 and Fig. 10 we present the results of using the function `ric_schr` from the mu-toolbox of MATLAB. They are very close to the results obtained by `aresolv` and demonstrate again the numerical instability of the unscaled Schur method.

The numerical behaviour of the solver `ricc` implementing the Schur method with block scaling is illustrated in Fig. 11 and Fig. 12. As in the case of the function `care` the backward errors are less than $10^{-10}$ for all $k$ and $s$ so that the corresponding implementation may be considered as conditionally stable. This solver, however, produces also estimates of the condition number and a bound on the forward error which increases significantly the reliability of the implementation. In Fig. 13 we show for the same example the ratio of the estimate of the condition number and the true condition number. We see that the difference between these quantities is less than half a decimal digit for all values of $k$ and $s$. Experiments with other examples confirm the high reliability of the condition estimator which makes use of the LAPACK estimator `DLACON`. The behaviour of the forward error estimator implemented in `ricc` is illustrated in Fig. 14 where we show the ratio of the true forward error and the forward error bound as a function of $k$ and $s$. It is seen from the figure that the bound is allways larger than the magnitude of the actual error (as one should expect) and that the bound becomes more pessimistic with the increase of $s$. For this example the pessimism of the estimate does not exceed 3 decimal digits for any $k$ and $s$ which is acceptable.

We also note that `ricc` is considerably faster than `care`.

Finally, in Fig. 15 and Fig. 16 we give the corresponding results for the solver `ricm` implementing the matrix sign function method plus the same scaling of the Hamiltonian matrix, implemented in `ricc`. Both forward and backward errors produced by this solver are the smallest

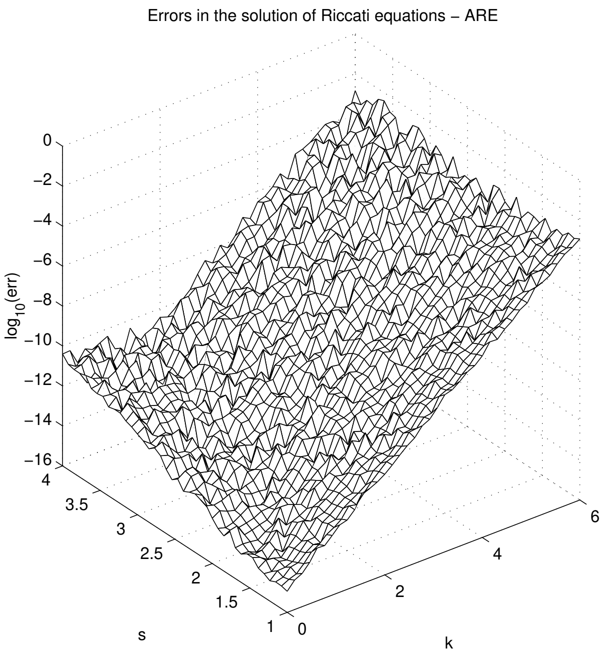


Figure 3: Forward errors for `are`

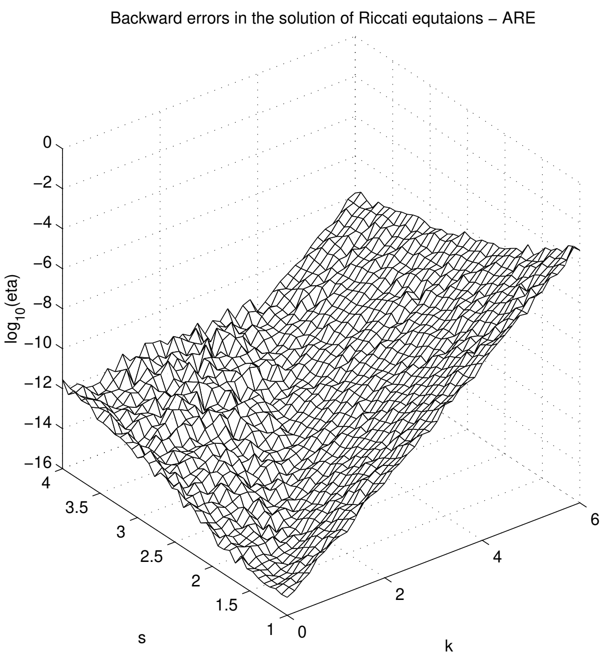


Figure 4: Backward errors for `are`

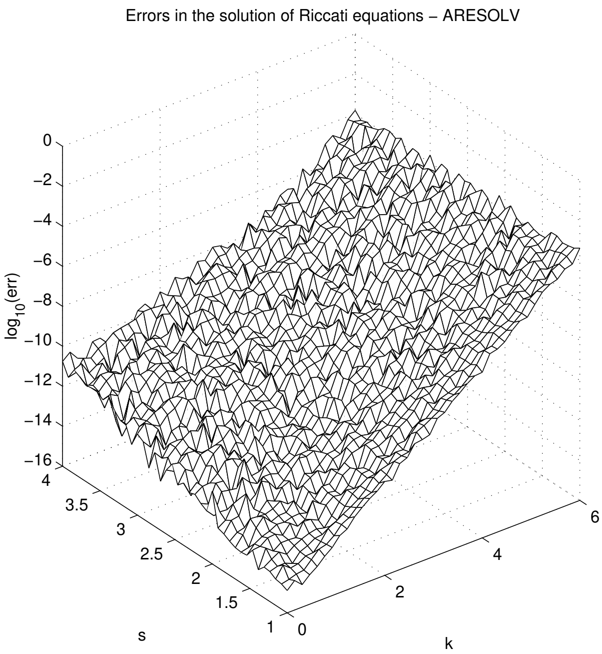


Figure 5: Forward errors for `aresolv`

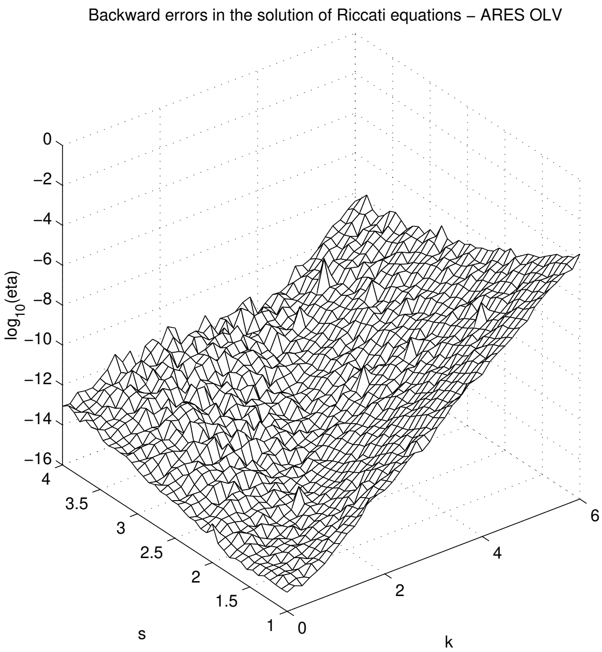


Figure 6: Backward errors for `aresolv`

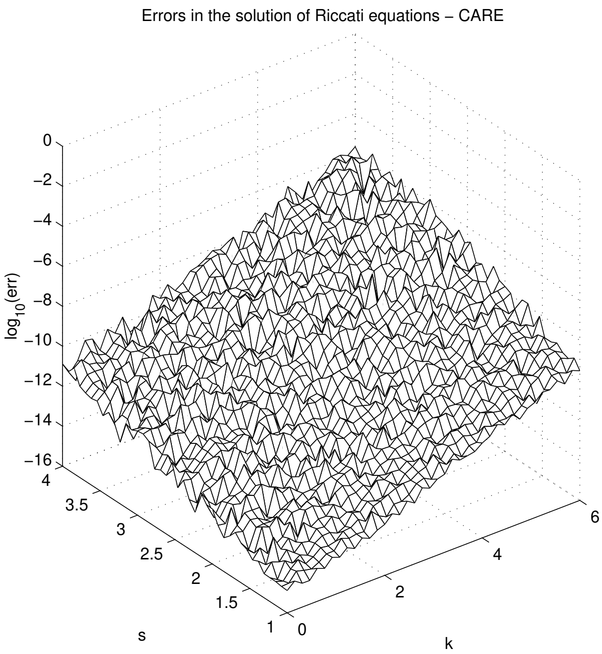


Figure 7: Forward errors for `care`

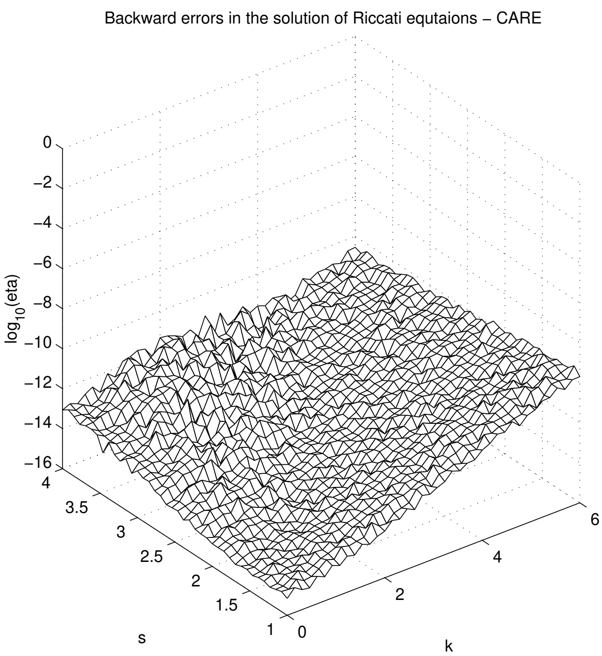


Figure 8: Backward errors for `care`

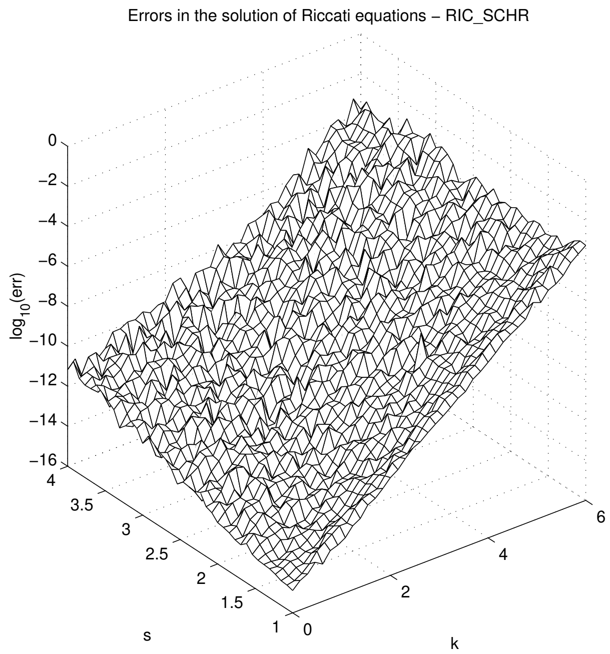


Figure 9: Forward errors for `ric_schr`

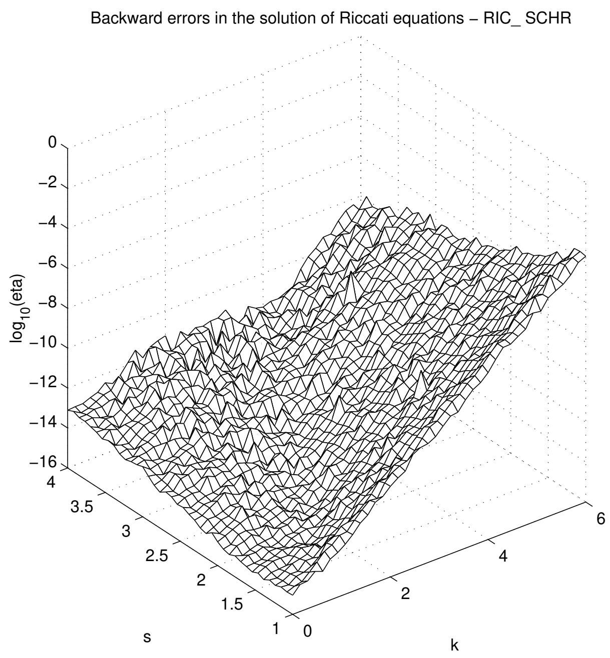


Figure 10: Backward errors for `ric_schr`

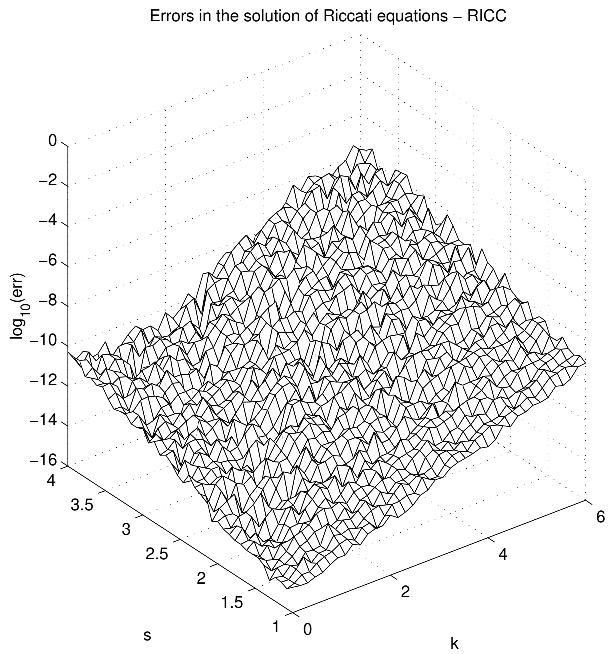


Figure 11: Forward errors for `ricc`

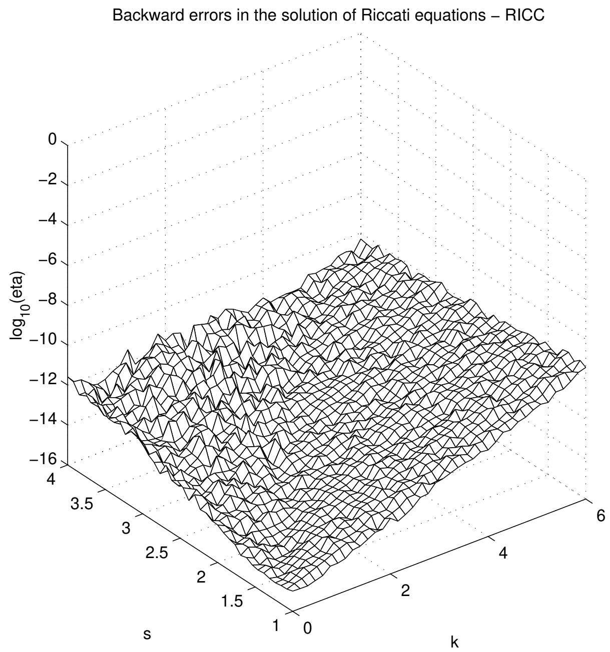


Figure 12: Backward errors for `ricc`

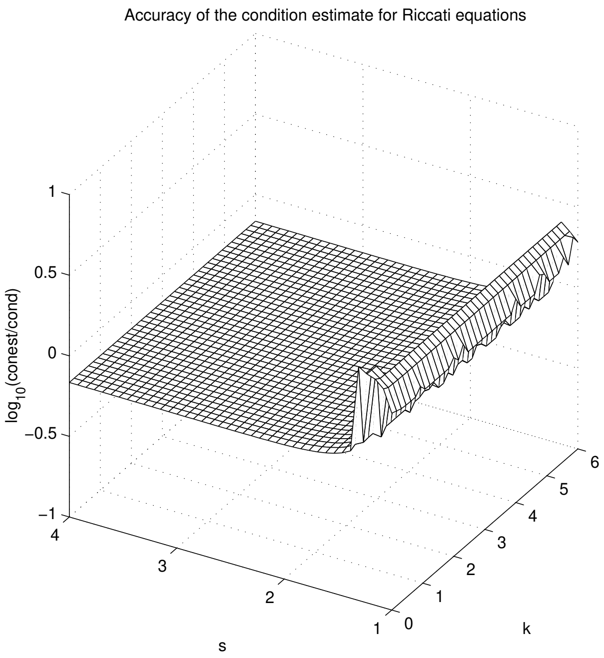


Figure 13: Accuracy of the condition estimate for `ricc`

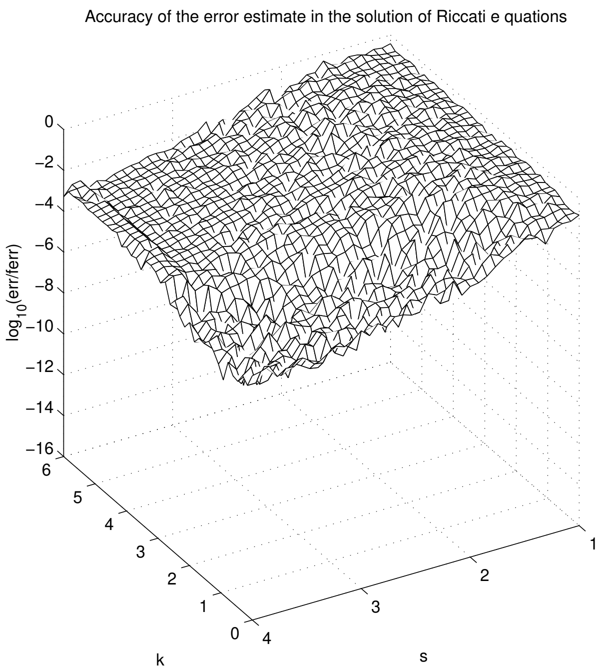


Figure 14: Accuracy of the forward error estimate for `ricc`

between all solvers under comparison. This confirms the good numerical properties of this method for solving the Riccati equation.

**Benchmark problem 2**

Consider a family of Riccati equations with $n = 6$ for which the diagonal blocks are chosen as in Problem 1 with

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

These equations become ill-conditioned with increasing $k$, the ill-conditioning being due to the decrease of the quantity $\mathrm{sep}_F(A_c^T, -A_c)$.

In Fig. 17 we show the relative residual for all values of $k$ and $s$ under interest which justify the acceptance of the computed $X$ as an "exact" solution. The values of the condition number shown in Figure 18 indicate that for large $k$ and $s$ the Riccati equation can be ill-conditioned and for $k = 3,\ s = 4$ we may expect a loss of 11 decimal digits in the computed solution.

In Figures 19 - 32 we show the numerical behaviour of the different solvers for this bencmark problem. As for the previous problem, the best results are obtained by the functions `care` and `ricm` and the worst results are obtained by `are` and `aresolv`. It is interesting that in this case the function `ric_schr` is producing the maximum possible accuracy and it performs actually in the same way as `care` and `ricm`.

In Figure 29 and Fig. 30 we illustrate the accuracy of the condition estimate and forward error bound produced by the solver `ricc`. (The same estimators are implemented also in the solver `ricm`.) As for the previous benccmark problem the results confirm the reliability of these estimators.

## 4 Conclusions

The limited numerical experiments with six Riccati solvers lead to the following conclusions.

- Three of the MATLAB solvers (`are, aresolv` and `ric_schr`) for matrix Riccati equations implement the unscaled Schur method which is known to be numerically unstable. The results confirm that for some well conditioned Riccati equations this method may produce results with inacceptably high forward and backward errors.

- The function `care` is the only MATLAB solver for the Riccati equation which performs well in all test examples due to the use of scaling of the Hamiltonian matrix. This function, however, is considerably slower than the corresponding Fortran solvers.

- The two Fortran solvers (`ricc` and `ricm`) which will be included in the SLICOT library always perform very well and produce reliable estimates of the condition number and forward error. It is clear from the presented results that similarly to the solution of linear systems of equations the solution of matrix Riccati equation should be accompanied with the computation of condition and accuracy estimates which increases significantly the reliability in solving several control systems design problems.

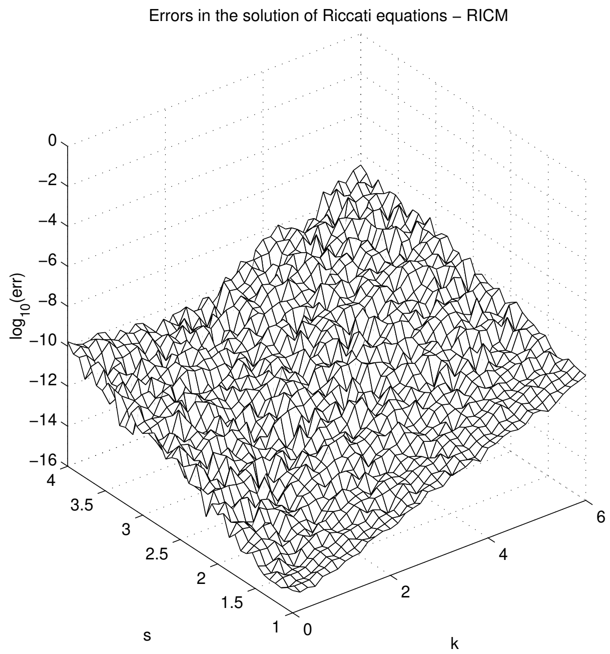


Figure 15: Forward errors for `ricm`

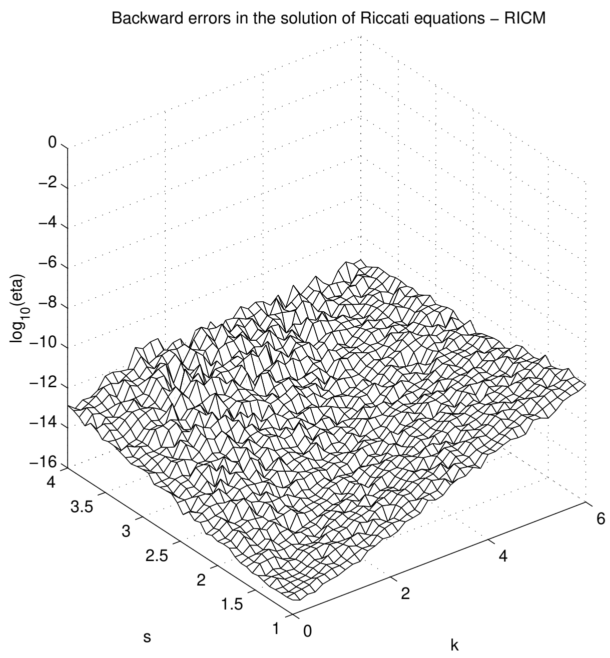


Figure 16: Bakward errors for `ricm`

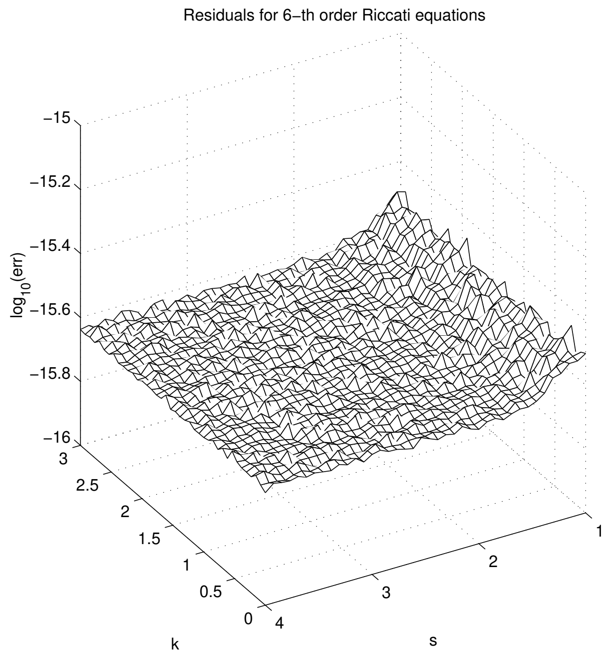


Figure 17: Relative residuals for the Riccati equations from Problem 2

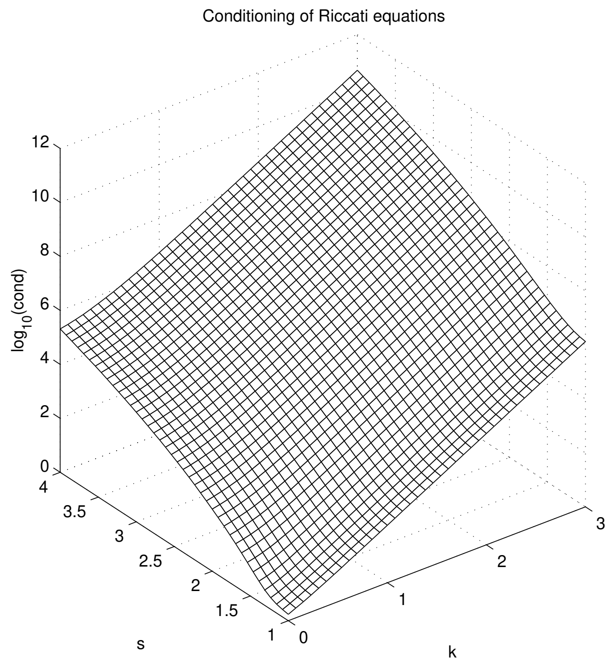


Figure 18: Conditioning of the Riccati equations from Problem 2

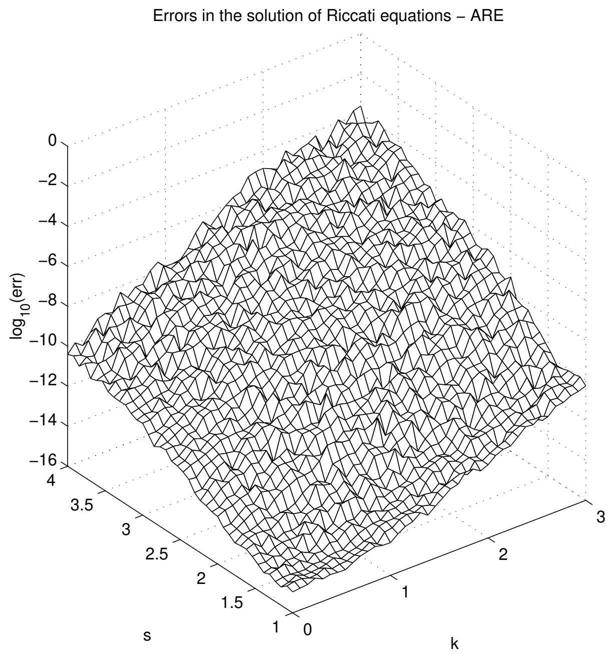


Figure 19: Forward errors for `are`

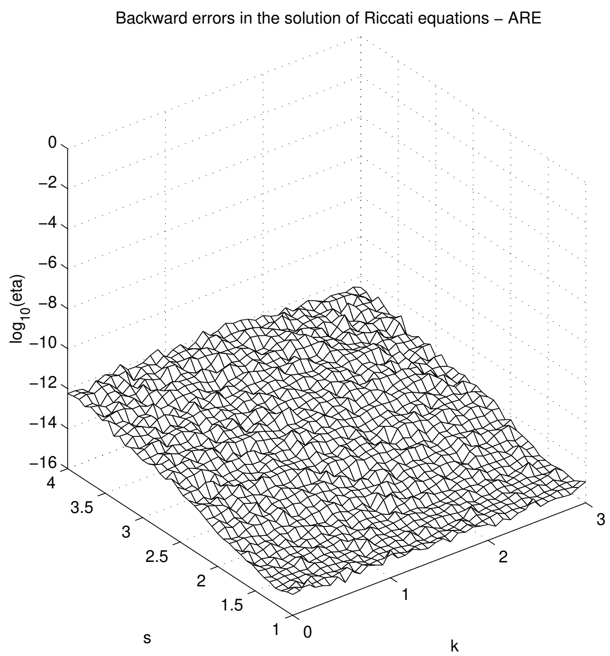


Figure 20: Backward errors for `are`

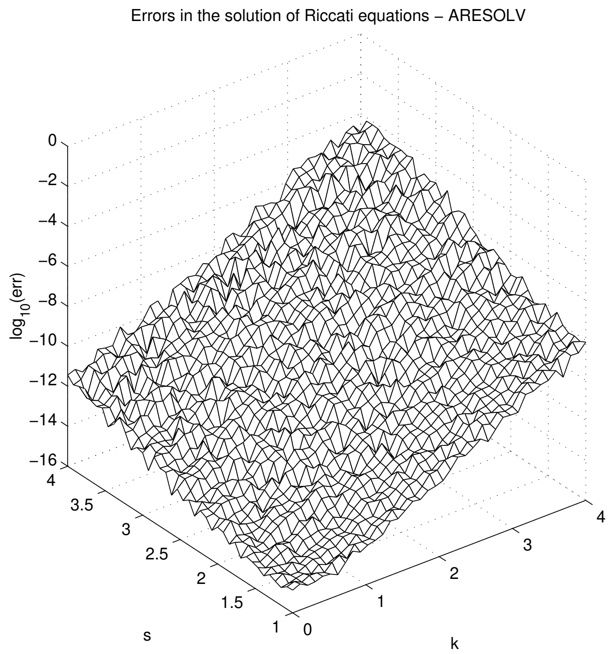


Figure 21: Forward errors for `aresolv`

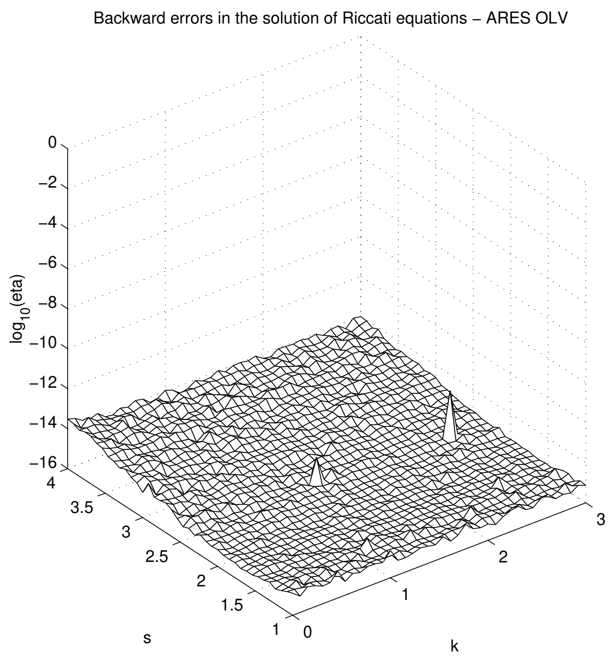


Figure 22: Backward errors for `aresolv`

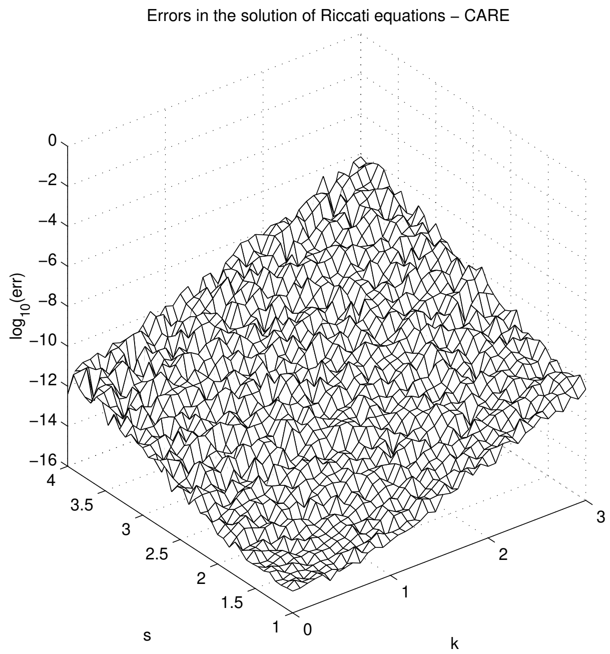


Figure 23: Forward errors for `care`

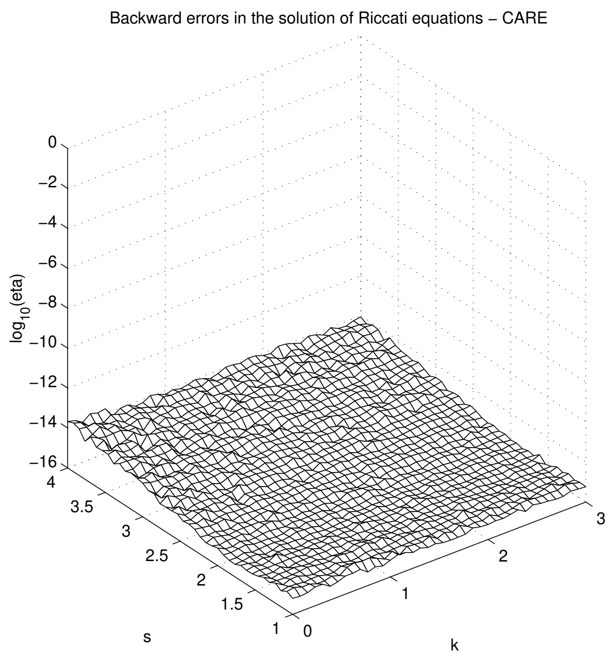


Figure 24: Backward errors for `care`

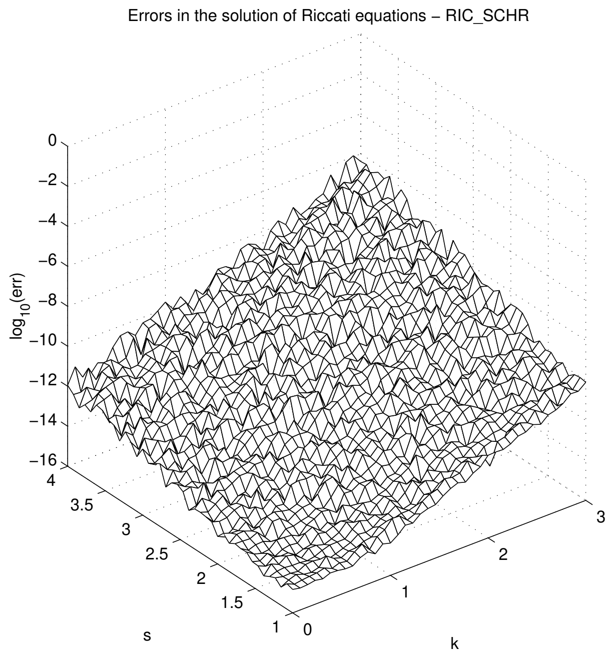


Figure 25: Forward errors for `ric_schr`

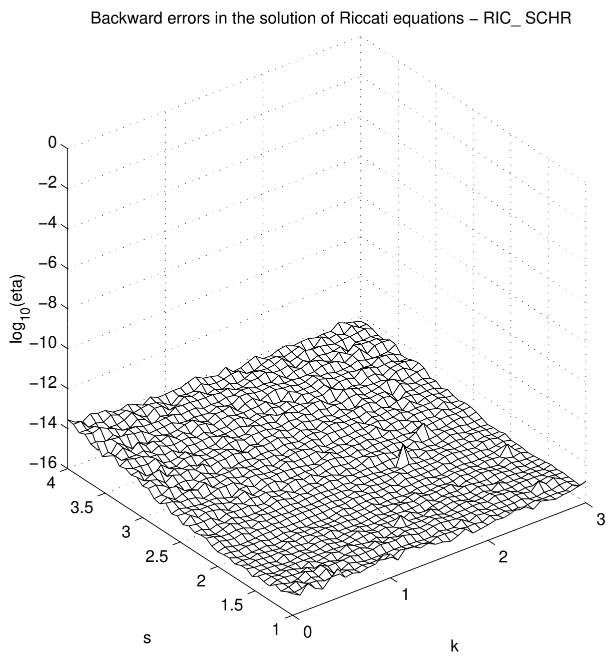


Figure 26: Backward errors for `ric_schr`

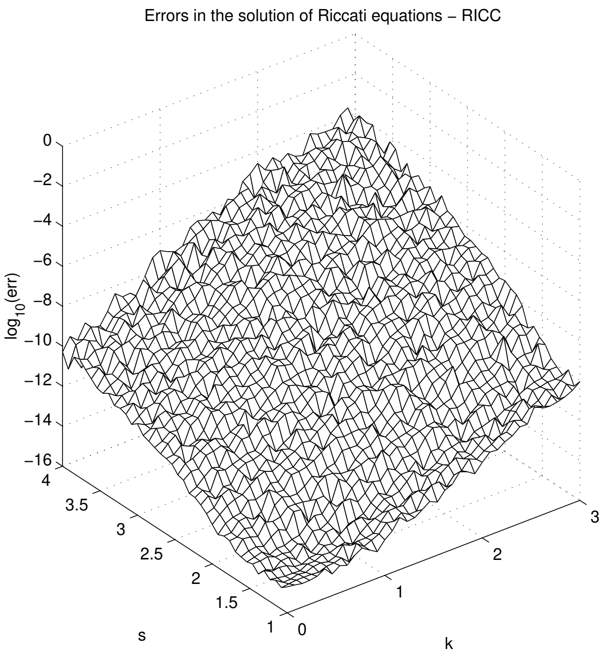


Figure 27: Forward errors for `ricc`

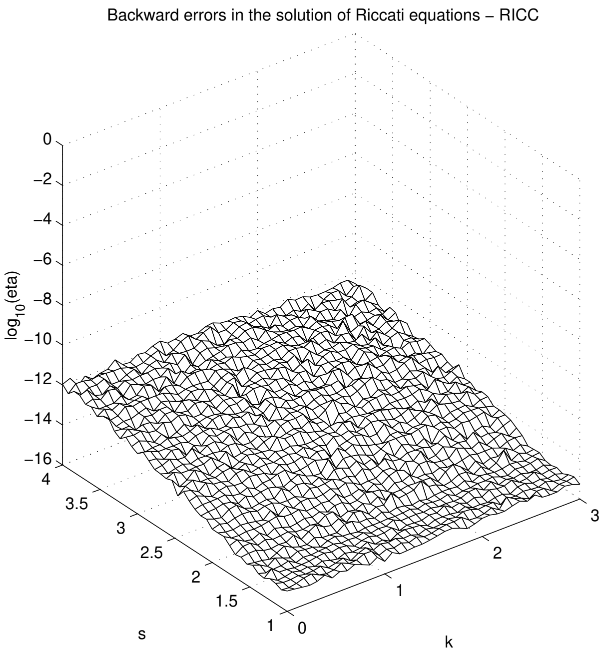


Figure 28: Backward errors for `ricc`

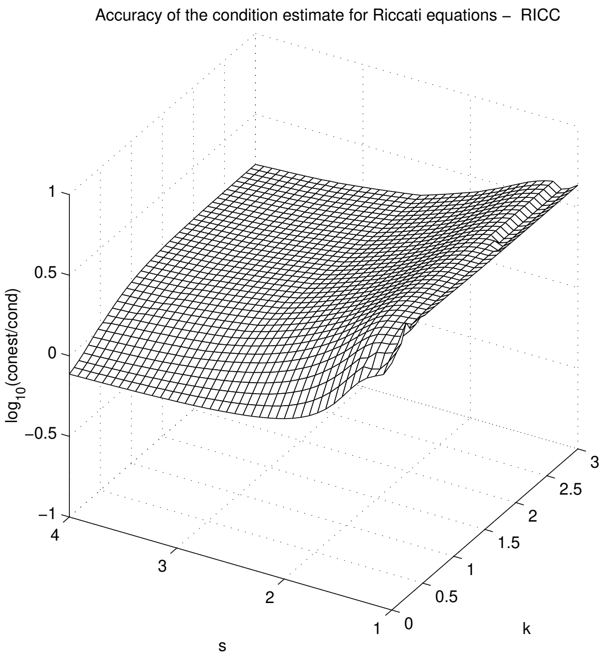


Figure 29: Accuracy of the condition estimate for the Riccati equations from Example 2

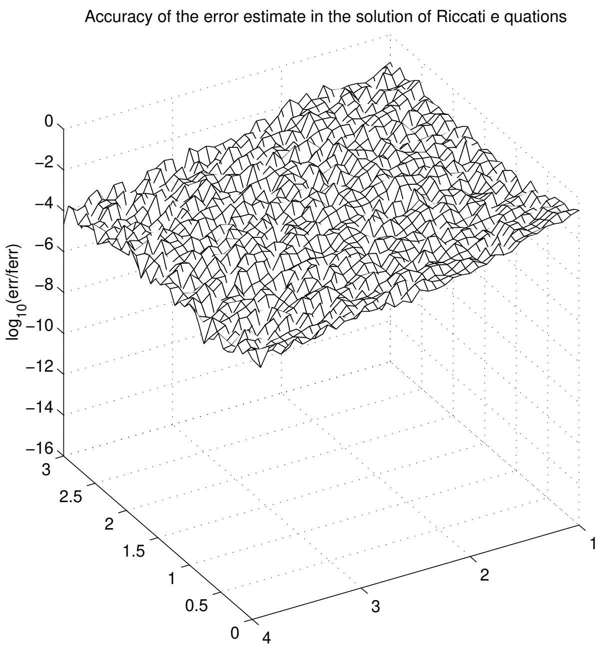


Figure 30: Accuracy of the forward estimate for the Riccati equations from Example 2

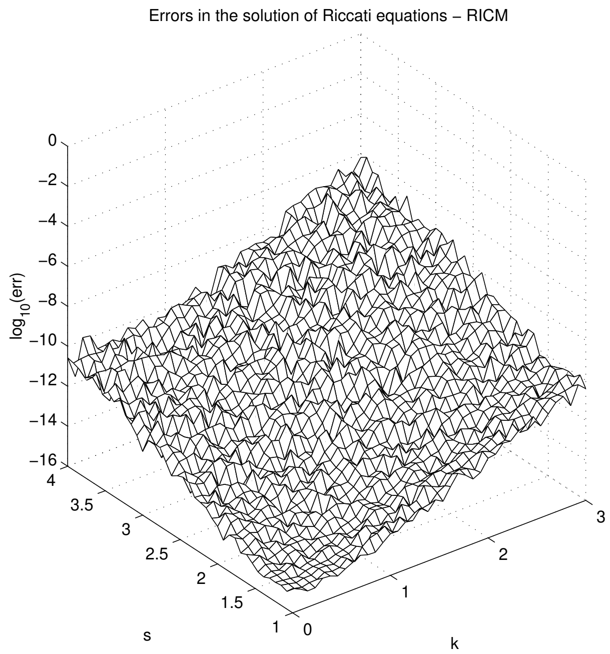


Figure 31: Forward errors for `ricm`

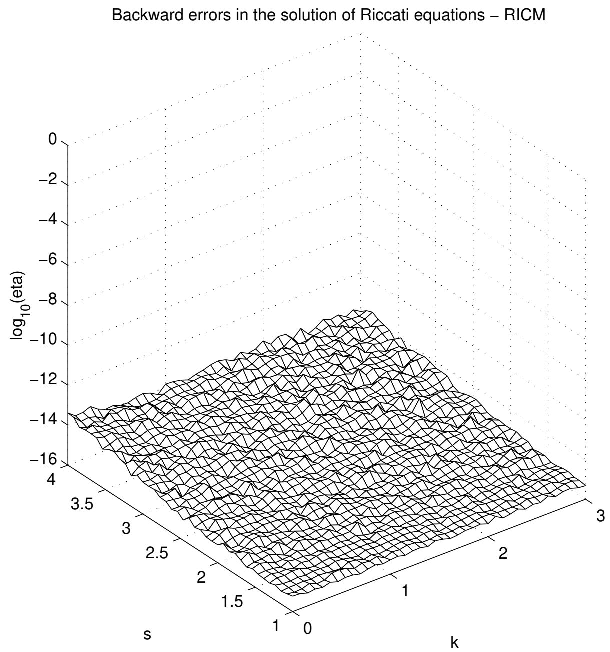


Figure 32: Backward errors for `ricm`

**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.

## References

[1] 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.

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

[3] 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).

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

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

[6] 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.

[7] 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.

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

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

[10] 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.

[11] 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.

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

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

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

[15] 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.

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

[17] 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.

[18] 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.

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

[20] 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.

[21] P.Hr. Petkov, M.M. Konstantinov, and V. Mehrmann. DGRSVX and DMSRIC: Fortran 77 subroutines for solving continuous-time matrix algebraic Riccati equations with condition and accuracy estimates. Technical Report SFB393/98-16, Fakultät für Mathematik, TU Chemnitz, 09107 Chemnitz, FRG, May 1998.

[22] 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).

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

[24] 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 a 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(eye(n),Ac') + kron(Ac',eye(n))),
%        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.
%

%        01.01.1999
%

function [cond,Omega,Theta,Pi] = cndricc(A,C,D,X)
n = max(size(A));
nora = norm(A,'fro');
norc = norm(C,'fro');
nord = norm(D,'fro');

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

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;
```

```
norx = norm(X,'fro');
if norx > 0
   cond = norm([D1, D2, -D3]) / norx;
else
   cond = 0;
end
```

## Appendix 2. MATLAB m-file for determining the backward error for a continuos–time Riccati equation

```
%function [dA,dC,dD,eta] = backricc(A,C,D,X,alpha,gamma,delta)
%BACKRICC Compute the backward error estimates for the continuous-time
%         matrix algebraic Riccati equation A'X + XA + C - XDX = 0
%

%         23/06/1999
%

function [dA,dC,dD,eta] = backricc(A,C,D,X,alpha,gamma,delta)
n = max(size(A));

% Compute the residual matrix
Res = A'*X + X*A + C - X*D*X;
[U,S] = eig(X);
R = U'*Res*U;
R = (R + R')/2;

% Find the perturbation matrices
for i = 1:n
    for j = 1:n
        dA(i,j) = -2*alpha*S(i,i)*R(i,j)/...
                   (2*alpha^2*(S(i,i)^2 + S(j,j)^2) + gamma^2+...
                   delta^2*S(i,i)^2*S(j,j)^2);
        dC(i,j) = -gamma*R(i,j)/...
                   (2*alpha^2*(S(i,i)^2 + S(j,j)^2) + gamma^2+...
                   delta^2*S(i,i)^2*S(j,j)^2);
        dD(i,j) = delta*S(i,i)*S(j,j)*R(i,j)/...
                   (2*alpha^2*(S(i,i)^2 + S(j,j)^2) + gamma^2+...
                   delta^2*S(i,i)^2*S(j,j)^2);
    end
end

dA = U*dA*U';
dC = U*dC*U';
dD = U*dD*U';

% Compute the backward error estimate
eta = max([norm(dA,'fro'),norm(dC,'fro'),norm(dD,'fro')]);
dA = alpha*dA;
dC = gamma*dC;
dD = delta*dD;
```