Solving Algebraic Riccati Equations with SLICOT

Peter Benner
Institut für Mathematik, MA 4-5
Technische Universität Berlin
Straße des 17. Juni 136
D-10623 Berlin, Germany
eMail: benner@math.tu-berlin.de
phone +49-30-314-28035; fax +49-30-314-79706

Vasile Sima
National Institute for Research & Development in Informatics
Bd. Mareşal Al. Averescu, Nr. 8–10
71316 Bucharest, Romania
eMail: vsima@iciadmin.ici.ro
phone +40-21-224-0765; fax +40-21-224-0539

Abstract— **The numerical solution of algebraic Riccati equations is a central issue in computer-aided control systems design. It is the key step in many computational methods for model reduction, filtering, and controller design for linear control systems. We discuss recent advances in the solvers for continuous-time and discrete-time algebraic Riccati equations available in the SLICOT Library (Subroutine Library In COntrol Theory) and compare their performance with the corresponding solvers in the MATLAB toolboxes.**

Keywords— **algebraic Riccati equations, numerical algorithms, computer-aided control system design, numerical linear algebra, software library**

I. Introduction

Let A, $E \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and Q, R, F be symmetric matrices of suitable dimensions. The *continuous-time algebraic Riccati equation (CARE)*

$$0 = Q + A^T XE + E^T XA - E^T XFXE \qquad (1)$$

and the *discrete-time algebraic Riccati equation (DARE)*

$$E^T XE = Q + A^T XA - A^T XB(R + B^T XB)^{-1} B^T XA \qquad (2)$$

play a fundamental role in many analysis and design procedures for linear and nonlinear control problems. In particular, the methods for computing linear-quadratic regulators and Kalman filters, for solving linear-quadratic Gaussian (H_2-) optimal control problems, as well as the computation of (sub)optimal H_∞ controllers are traditionally based on solving algebraic Riccati equations (AREs). Moreover, model reduction techniques based on stochastic and positive real balancing require the solution of AREs [20], [28]. In these applications, usually $E = I_n$ in (1) and (2). The case $E \neq I_n$ appears, for instance, in factorization procedures for transfer functions, see, e.g., [33], or optimal control problems for second-order systems, see, e.g., [13], [18], [19]. Common to all these applications is that among the possibly infinitely many solutions of the ARE, usually the unique *stabilizing* solution X_* is required; this is the solution for which all eigenvalues of the matrix pencils $\lambda E - (A - FX_*E)$ in the continuous-time case, and $\lambda E - (A - B(R + B^T X_* B)^{-1} B^T X_* A)$ in the discrete-time case, are in the appropriate stability region (the left half plane in the continuous-time case and the open unit disk in the discrete-time case).

There is a huge amount of theoretical results available both in systems and control, as well as in the linear algebra literature devoted to matrix Riccati equations and related topics; see, e.g., the monographs [21], [26]. Due to its importance in computational control, a vast variety of numerical methods has been proposed for solving AREs; see, e.g., [7], [26], [30]. There are also a lot of associated software implementations, both commercial (e.g., in MATLAB[1] [17], [25], [12], [5]), copyrighted freeware (e.g., in the SLICOT Library [9], [27], [32]), or in the public domain (e.g., in Scilab [14]). The reliability, efficiency, and functionality of the different solvers vary significantly from package to package.

This paper presents several solvers for AREs available in the SLICOT Library (**S**ubroutine **L**ibrary **I**n **CO**ntrol **T**heory). SLICOT provides Fortran 77 implementations of many numerical algorithms in systems and control theory, as well as standardized interfaces (gateways) to MATLAB and Scilab. Built around a nucleus of basic numerical linear algebra subroutines from the state-of-the-art software packages LAPACK [4] and BLAS [15], [16], [23] the potential of modern high-performance computer architectures can be exploited.

The paper also presents some performance improvements (concerning efficiency, reliability, and accuracy) offered by

This work was supported in part by the European Community BRITE-EURAM III *Thematic Networks Programme NICONET* (project BRRT–CT97-5040) and the DFG Research Center "Mathematics for key technologies" (FZT 86) in Berlin.

[1] MATLAB is a registered trademark of The MathWorks, Inc.

the SLICOT tools, in comparison with equivalent computations performed by some MATLAB functions included in the MATLAB nucleus or in the Control Systems, Robust Control, μ-Analysis and Synthesis, or LMI Control Toolboxes [17], [25], [12], [5]. The results show that, at comparable or better accuracy, SLICOT computations are several times faster than MATLAB computations; moreover, the underlying problem structure is often fully exploited.

II. Algebraic Riccati Equations

In a general setting, AREs are defined as follows.

- Continuous-time and discrete-time algebraic Riccati equations: if, for notational convenience, we use the abbreviation

$$\begin{array}{rcll} \mathrm{op}(F) & := & \mathrm{op}(B)\, R^{-1}\, \mathrm{op}(B)^T & \text{(for CARE)}, \\ \mathrm{op}(\hat{R}) & := & R + \mathrm{op}(B)^T X\, \mathrm{op}(B) & \text{(for DARE)}, \end{array}$$

where $\mathrm{op}(M)$ is either M or M^T, then

$$\begin{aligned} 0 &= Q + \mathrm{op}(A)^T X + X\, \mathrm{op}(A) - X\, \mathrm{op}(F)\, X, && (3) \\ X &= Q + \mathrm{op}(A)^T X\, \mathrm{op}(A) && (4) \\ &\quad - \mathrm{op}(A)^T X\, \mathrm{op}(B)\, \mathrm{op}(\hat{R})^{-1}\, \mathrm{op}(B)^T X\, \mathrm{op}(A). \end{aligned}$$

- Generalized CAREs and DAREs: defining

$$\begin{array}{rcll} \mathrm{op}(\mathcal{L}(X)) & := & \mathrm{op}(L) + \mathrm{op}(E)^T X\, \mathrm{op}(B) & \text{(for CARE)}, \\ \mathrm{op}(\hat{\mathcal{L}}(X)) & := & \mathrm{op}(L) + \mathrm{op}(A)^T X\, \mathrm{op}(B) & \text{(for DARE)}, \end{array}$$

the equations are defined by

$$\begin{aligned} 0 &= Q + \mathrm{op}(A)^T X\, \mathrm{op}(E) + \mathrm{op}(E)^T X\, \mathrm{op}(A) && (5) \\ &\quad - \mathrm{op}(\mathcal{L}(X))\, R^{-1}\, \mathrm{op}(\mathcal{L}(X))^T, \\ 0 &= Q - \mathrm{op}(E)^T X\, \mathrm{op}(E) + \mathrm{op}(A)^T X\, \mathrm{op}(A) && (6) \\ &\quad - \mathrm{op}(\hat{\mathcal{L}}(X))\, \mathrm{op}(\hat{R})^{-1}\, \mathrm{op}(\hat{\mathcal{L}}(X))^T. \end{aligned}$$

For AREs, the ability to work with the $\mathrm{op}(\cdot)$ operator is important for notational convenience. For instance, an optimal regulator problem involves the solution of an ARE with $\mathrm{op}(M) = M$, while an optimal estimator problem involves the solution of an ARE with $\mathrm{op}(M) = M^T$. The Riccati solution can be used for computing the gain matrix of the optimal regulator, G, or estimator, $K = G^T$,

$$\begin{aligned} G &= R^{-1}\, \mathrm{op}(\mathcal{L}(X))^T, && (7) \\ G &= \mathrm{op}(\hat{R})^{-1}\, \mathrm{op}(\hat{\mathcal{L}}(X))^T, && (8) \end{aligned}$$

for continuous-time and discrete-time systems, respectively. The basic methods for solving AREs are the Schur vector method [22] and the deflating subspaces method [29], [31] which we will briefly describe here for future reference: for the CARE (3), the associated *Hamiltonian matrix* is

$$H = \begin{bmatrix} \mathrm{op}(A) & -\mathrm{op}(F) \\ -Q & -\mathrm{op}(A)^T \end{bmatrix}, \qquad (9)$$

while the *symplectic matrix* associated to (4) is

$$H = \begin{bmatrix} \mathrm{op}(A) + \mathrm{op}(F)\, \mathrm{op}(A)^{-T} Q & -\mathrm{op}(F)\, \mathrm{op}(A)^{-T} \\ -\mathrm{op}(A)^{-T} Q & \mathrm{op}(A)^{-T} \end{bmatrix} \qquad (10)$$

with F defined as in the continuous-time case. Note that the symplectic matrix can only be formed if A is nonsingular and even in case A is invertible, forming H should be done with care if A is ill-conditioned; in the latter case, it is preferable to work with the symplectic pencil

$$L - \lambda M = \begin{bmatrix} \mathrm{op}(A) & 0 \\ -Q & I_n \end{bmatrix} - \lambda \begin{bmatrix} I_n & \mathrm{op}(F) \\ 0 & \mathrm{op}(A)^T \end{bmatrix}. \qquad (11)$$

The Schur vector method now computes the ordered real Schur form T of the Hamiltonian or symplectic matrix H, i.e., $U \in \mathbb{R}^{2n,2n}$ orthogonal is computed such that with a partitioning according to (9), (10),

$$HU = UT = \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix} \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}$$

and the eigenvalues of T_{11} are the n stable eigenvalues of H. Hence, the columns of $[\, U_{11}^T,\, U_{21}^T \,]^T$ span the stable H-invariant subspace. If the stabilizing solution exists, then U_{11} is invertible and the solution of the CARE or DARE is $X = U_{21} U_{11}^{-1}$.

The deflating subspace approach proceeds in an analogous way. Define the *extended Hamiltonian pencil* associated to (5),

$$\begin{aligned} L - \lambda M = &\begin{bmatrix} \mathrm{op}(A) & 0 & \mathrm{op}(B) \\ Q & \mathrm{op}(A)^T & \mathrm{op}(L) \\ \mathrm{op}(L)^T & \mathrm{op}(B)^T & R \end{bmatrix} \qquad (12) \\ &- \lambda \begin{bmatrix} \mathrm{op}(E) & 0 & 0 \\ 0 & -\mathrm{op}(E)^T & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

and *extended symplectic pencil* corresponding to (6),

$$\begin{aligned} L - \lambda M = &\begin{bmatrix} \mathrm{op}(A) & 0 & \mathrm{op}(B) \\ Q & -\mathrm{op}(E)^T & \mathrm{op}(L) \\ \mathrm{op}(L)^T & 0 & R \end{bmatrix} \qquad (13) \\ &- \lambda \begin{bmatrix} \mathrm{op}(E) & 0 & 0 \\ 0 & -\mathrm{op}(A)^T & 0 \\ 0 & -\mathrm{op}(B)^T & 0 \end{bmatrix} \end{aligned}$$

Then the ordered generalized real Schur form of $L - \lambda M$ is computed, i.e., $U, V \in \mathbb{R}^{2n+m,2n+m}$ orthogonal are computed such that $(L - \lambda M)U = V(T_L - \lambda T_M)$ and the first n columns of U, denoted by $[\, U_{11}^T,\, U_{21}^T,\, U_3^T \,]^T$ span the stable deflating subspace of $L - \lambda M$. Then, the stabilizing solution of the generalized CARE or DARE is given via $XE = U_{21} U_{11}^{-1}$ and the gain matrices in (7), (8) are given via $G = -U_3 U_{11}^{-1}$. The deflating subspace approach is also used to solve the DARE via (11); the deflating subspace is given by the first n Schur vectors of the ordered generalized real Schur form of $L - \lambda M$ in (11), denoted by $[\, U_{11}^T,\, U_{21}^T \,]^T$ and $X = U_{21} U_{11}^{-1}$. It should also be noted that the deflating subspace approach using the extended pencils yields better numerical accuracy if R is ill-conditioned as rounding errors introduced by forming R^{-1} are avoided.

For the standard continuous-time case (3) with $\mathrm{op}(M) = M$, SLICOT also includes an implementation

of the matrix sign function method [11]. It is planned to extend SLICOT by integrating CARE and DARE solvers based on Newton's method (with line search) [6], [8], [30]. This will be particularly useful as it enables users to compute a solution to maximal accuracy by refining any approximate solution by applying Newton's method to it. Moreover, solvers based on structure-preserving methods for the underlying eigenproblems ([3], [10]) are under consideration. These methods can deal with situations where eigenvalues of the corresponding matrix (pencil) are close to or on the imaginary axis (CARE case) or unit circle (DARE case).

III. Solvers for Riccati Equations

The MATLAB Control Toolbox and the LMI Toolbox ([25], [17]) include two solvers for AREs. The commands

```
[X,ev,G,rr] = care(A,B,Q,R,L,E);
[X,ev,G,rr] = dare(A,B,Q,R,L,E);
```

compute the unique symmetric stabilizing solution X of the generalized continuous-time or discrete-time algebraic Riccati equation (5) or (6), respectively, with $\mathrm{op}(M) = M$. When omitted, `R`, `L` and `E` are set to the default values I_m, $0 \in \mathbb{R}^{n\times m}$, and I_n, respectively. The optional output arguments `ev` and `rr` contain the vector of closed-loop eigenvalues (i.e., `eig( A - B*G, E )`), and the Frobenius norm of the relative residual matrix, respectively. An additional last input argument `'report'` turns off the error messages and returns instead a success/failure diagnosis in `rr`, then containing one of the following results

-1	if the associated matrix or matrix pencil (9)–(13) has eigenvalues too close to the $j\omega$ axis (or the unit circle, respectively);
-2	if the linear algebraic system $U_{11}^T X = U_{21}^T$ cannot be solved due to singularity of U_{11};
`rr`	the relative residual when the solver succeeds.

The command below, where `*` means either `c` or `d`,

```
[U11,U21,ev,rr] = *are(A,B,Q,R,L,E,'implicit');
```

also turns off the error messages, but each solver returns the matrices U_{11} and U_{21} so that $X = U_{21}U_{11}^{-1}$. The value `rr` $= 0$ indicates success. It should be noted that the MATLAB Robust Control Toolbox and the μ-Analysis and Synthesis Toolbox also provide CARE and DARE solvers (`aresolv`, `daresolv` in [12], `ric_eig` and `ric_schr` in [5], where the latter only apply to CAREs). The functionality of these solvers is reduced compared to `*are` as only a particular form of the equations (3) and (4) can be solved. They are either based on the Schur vector method applied to the associated Hamiltonian or symplectic matrix or on computing the eigenvectors of these matrices corresponding to the stable eigenvalues. These methods typically behave worse than the `*are` solvers, see Table I. Also note that `daresolv` calls `dare` if A has zero eigenvalues (which is not reported in [12])!

The SLICOT Library contains 9 "user-callable" and 3 "programmer-callable" Fortran 77 routines for Riccati equations (3)–(6). Just one solver, based on a refined Schur vector method for standard equations (with $E = I_n$ and $L = 0$), is able to deal with the operator $\mathrm{op}(\cdot)$. All other solvers assume $\mathrm{op}(M) = M$. Currently, there are two MEX-files, `aresol` and `aresolc`, both for equations with $\mathrm{op}(M) = M$, $E = I_n$, but L can be nonzero. The MEX-file `aresolc` can provide estimates for condition numbers and forward error bounds for solutions. (Note that condition number estimates are also provided by the MATLAB Robust Control Toolbox functions `riccond`, `driccond`.) Another MEX-file for general E and L matrices, based on the available SLICOT Fortran routines, will be ready soon. The following M-files calling the corresponding MEX-file are currently available for solving Riccati equations:

`slcaregs`	Solve CARE with generalized Schur method applied to (12).
`slcares`	Solve CARE with Schur method.
`slcaresc`	Solve CARE with refined Schur method and estimate condition.
`sldaregs`	Solve DARE with generalized Schur method applied to (13).
`sldares`	Solve DARE with Schur method.
`sldaresc`	Solve DARE with refined Schur method and estimate condition.
`sldaregsv`	Solve DARE with generalized Schur method applied to (11).

The command

```
[X,G,ev,rcond1] = sl*ares(A,Q,R,B,L,flag);
```

computes the symmetric solution X of a CARE (5) or DARE (6) with $E = I_n$, the feedback gain matrix G in (7) or (8), the closed-loop eigenvalues `ev`, and an estimate `rcond1` of the reciprocal of the condition number of the system of algebraic equations from which the solution X is obtained ($XU_{11} = U_{21}$), by using the Schur vector method applied to (9) or (10). The command

```
[U11,U21,ev,rcond1] = sl*ares(A,Q,R,B,L,flag);
```

computes a basis, $\left[\,U_{11}^T \;\; U_{21}^T\,\right]^T$, of the invariant subspace instead of the solution X. (Clearly, $X = U_{21}U_{11}^{-1}$, when U_{11} is nonsingular.) The commands

```
[X,ev,rcond1] = sl*ares(A,Q,F,flag);
[U11,U21,ev,rcond1] = sl*ares(A,Q,F,flag);
```

compute the same results when $F = BR^{-1}B^T$ is given instead of B and R. The argument `flag` is a vector with 3 elements containing the following options:

`flag(1)` $= 0$	Compute the stabilizing solution; otherwise, compute the anti-stabilizing solution.
`flag(2)` $= 0$	Use a scaling strategy; otherwise, do not use a scaling strategy.
`flag(3)` $= 0$	Compute both the solution X and the feedback gain matrix G;
`flag(3)` > 0	Compute the solution X only;
`flag(3)` < 0	Compute the invariant subspace.

Default value is `flag` $= [0, 0, 1]$.

Replacing the names `slcares` by `slcaregs` and `sldares` by `sldaregs`, the same results can be computed using the generalized Schur method applied to (12) or (13). These M-files can be also used when R is singular. For this reason, the calling sequences using the matrix F above are not valid for `slcaregs` and `sldaregs`. The argument `flag` is again a vector with 3 elements: `flag(1)` has the same meaning, `flag(2)` corresponds to `flag(3)` above, and `flag(3)` is this time a tolerance to check the singularity of the matrix pencil. If `flag`$(3) \leq 0$, the used tolerance is ε_M, the machine precision. Default value is `flag` $= [0, 1, 0.0]$.

The M-file `sldaregsv` has the same function and calling sequences as `sldares`, but it uses the generalized Schur method applied to (11). The argument `flag` has the same meaning as for `slcaregs` and `sldaregs`.

The M-files `slcaresc` and `sldaresc` are similar to `slcares` and `sldares`, respectively, but for positive values of `flag(3)` we have

`flag`$(3) = 1$	Compute the solution X only;
`flag`$(3) = 2$	Compute the solution X and estimates of the separation, reciprocal condition number and forward error bound.

These M-files use a refined Schur method and, for `flag`$(3) = 2$, the calling sequences are:

```
[X,ev,rcond1,acc] = sl*aresc(A,Q,R,B,L,flag);
[X,ev,rcond1,acc] = sl*aresc(A,Q,F,flag);
```

where `acc` is a vector of length 3 containing estimates of the separation, reciprocal condition number, and forward error bound, respectively.

IV. Numerical results

This section presents typical performance results for some ARE solvers contained in the SLICOT Library, called via the associated gateways. The calculations were performed on an IBM PC computer at 500 MHz, with 128 Mb memory, using Compaq Visual Fortran V6.5, non-optimized BLAS, and MATLAB 6.1 (R12). These results show that SLICOT routines often outperform MATLAB calculations. While the accuracy is comparable, and sometimes better, the gain in efficiency by calling SLICOT routines can be significant. Note that the results have been obtained by timing in MATLAB the equivalent computations. Even better efficiency is to be expected by calling the SLICOT Fortran routines directly (not through gateways), and similar accuracy/efficiency improvements are possible for other SLICOT computations. Better results can be expected using optimized BLAS implementations.

Figure 1 shows the execution times and relative residuals for solving randomly generated CAREs with $n = 30 : 30 : 300$ and $m = n/5$.

Figure 2 shows the execution times and speed-up factors for solving the CAREs for the benchmark collection [1]. (All examples are included, except for Example 4.4 ($n = 421$, $m = 211$), which could not be solved satisfactorily by the Schur vector based solvers.) Default values for the parameter(s) have been used.

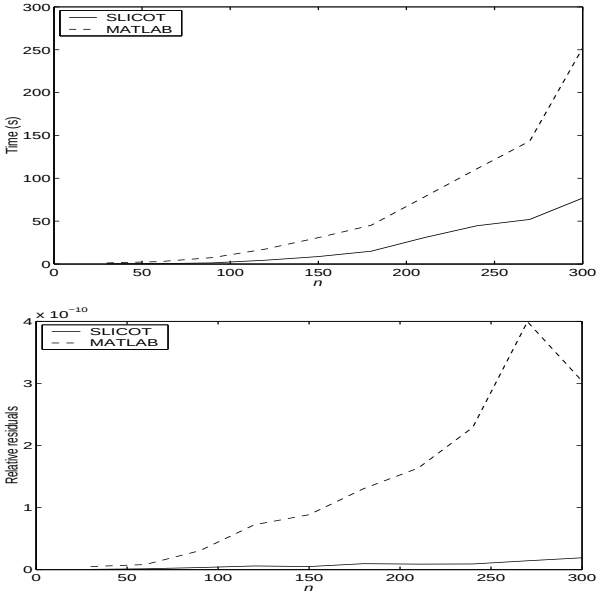

Fig. 1. SLICOT `slcares` versus MATLAB `care` for random CAREs, $n = 30 : 30 : 300$, $m = n/5$. Timing (top) and relative residuals (bottom) comparisons.

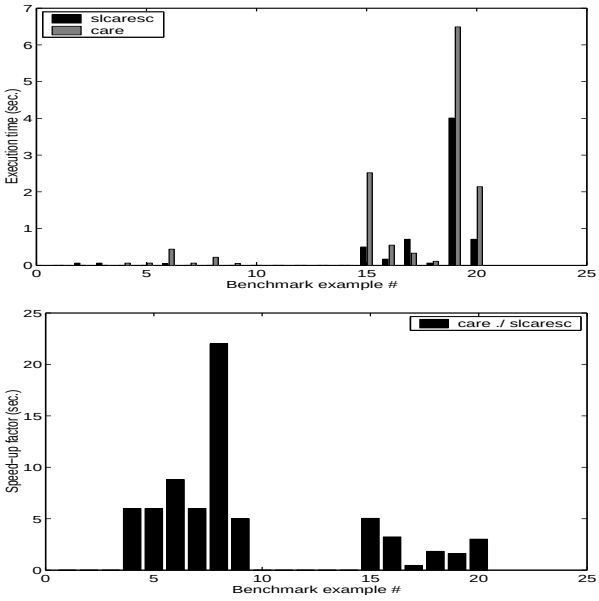

Fig. 2. SLICOT `slcaresc` versus MATLAB `care` for the CAREs in the benchmark collection. Timing (top) and speed-up factors (bottom) comparisons.

Figure 3 shows the decimal logarithms of the relative residuals and relative errors (when available) for the same examples.

Table I shows the cumulative execution times, and the Euclidean norms of the relative residuals and relative errors for 34 experiments performed as described in [1] with the 20 CAREs in the benchmark collection. Besides `slcaresc` and `care`, the functions `ric_eig` and `ric_schr` from the Robust Control Toolbox, as well as `aresolv` from μ-Analysis and Synthesis Toolbox have been used. These

Performance	slcaresc	care	ric_eig	ric_schr	aresolv 'eigen'	aresolv 'Schur'
Time (sec.)	6.71	13.84	2.9	15.75	4.07	15.21
Rel. residuals	2.98e-4	4.90e-4	1.33e+3	1.31e+3	1.33e+3	3.18e+3
Rel. errors	8.88e-5	4.44e-5	2.32e-4	2.45e-4	2.32e-4	5.72e-4

TABLE I

CUMULATIVE PERFORMANCE FOR CONTINUOUS-TIME ARE SOLVERS FOR BENCHMARK COLLECTION EXAMPLES

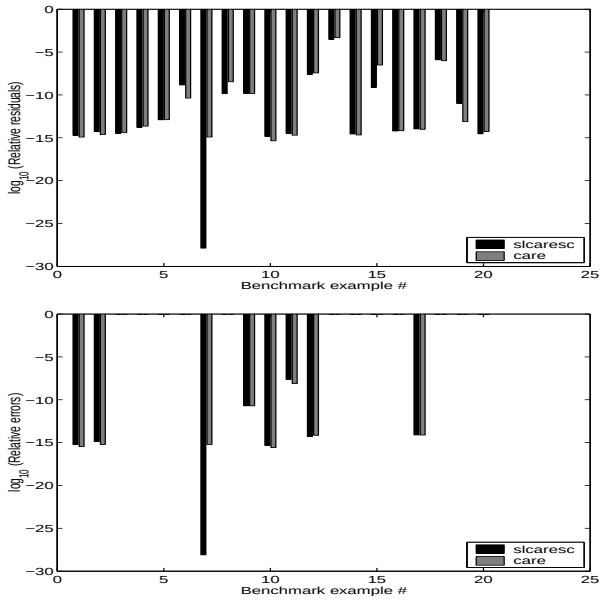

Fig. 3. SLICOT `slcaresc` versus MATLAB `care` for the CAREs in the benchmark collection. Relative residuals (top) and relative errors (bottom) comparisons.

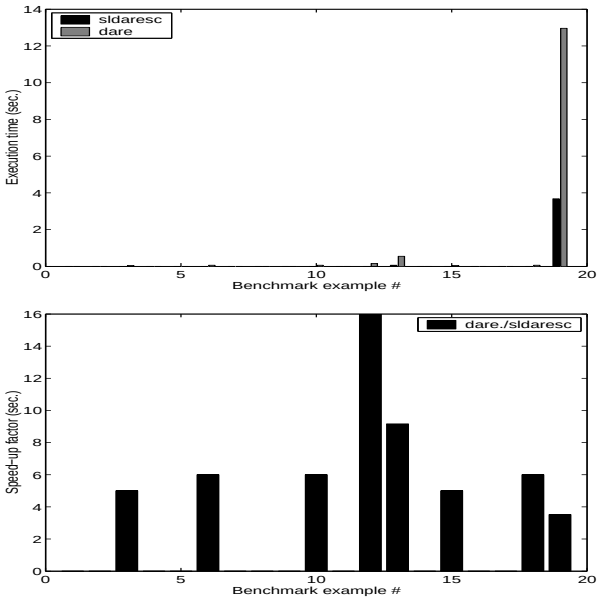

Fig. 4. SLICOT `sldaresc`/`sldaregs` versus MATLAB `dare` for the DAREs in the benchmark collection. Timing (top) and speed-up factors (bottom) comparisons.

additional solvers worked worse than `slcaresc` and `care`; the worst results were obtained for example 2.6 (renumbered 12 in the figures), for which the relative residuals and relative errors corresponding to `ric_eig`, `ric_schr`, and `aresolv` were $\mathcal{O}(10^3)$ and $\mathcal{O}(10^{-4})$, respectively, in comparison with $\mathcal{O}(10^{-8})$ and $\mathcal{O}(10^{-15})$, respectively, for `slcaresc` and `care`.

Similarly, Figure 4 shows the execution times and speed-up factors for solving the DAREs for the benchmark collection [2], and Figure 5 shows the decimal logarithms of the relative residuals and relative errors (when available) for the same examples. The SLICOT function `sldaregs` is used instead of `sldaresc` when the matrices R and/or A are singular.

Table II shows the cumulative execution times, and the Euclidean norms of the relative residuals and relative errors for 25 experiments performed as described in [2] with the 19 DAREs in the benchmark collection. Besides `sldaresc`/`sldaregs` and `dare`, `daresolv` from the μ-Analysis and Synthesis Toolbox has been used. This additional solver calls `dare` when matrix A is singular, or when used with option `aretype` = `'dare'`; it did not work (for `aretype` $\neq$ `'dare'`) for examples 1.1, 1.2, and 1.4 (renumbered 1, 2, and 4, respectively, in the figures),

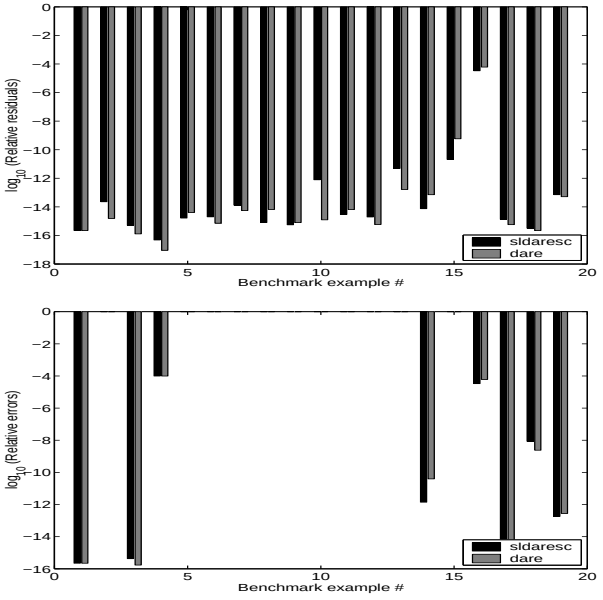

Fig. 5. SLICOT `sldaresc`/`sldaregs` versus MATLAB `dare` for the DAREs in the benchmark collection. Relative residuals (top) and relative errors (bottom) comparisons.

Performance	sldaresc/ sldaregs	dare	daresolv 'eigen'	daresolv 'Schur'
Time (sec.)	3.74	14.12	12.64	13.38
Rel. residuals	3.34e-5	6.11e-5	6.11e-5	6.11e-5
Rel. errors	3.34e-5	6.11e-5	6.11e-5	6.11e-5

TABLE II

CUMULATIVE PERFORMANCE FOR DARE SOLVERS FOR BENCHMARK COLLECTION EXAMPLES

which have a singular matrix R. The worst relative errors, of order $\mathcal{O}(10^{-4})$, have been obtained for Example 1.4, but the associated relative residuals were very small. All solvers gave $\mathcal{O}(10^{-5})$ relative residuals and errors for Example 2.3 (renumbered 16). The other examples have been solved significantly more accurately.

V. CONCLUSIONS

Easy-to-use solvers for algebraic Riccati equations, available in the SLICOT Library have been discussed. Based on Fortran 77 codes implementing state-of-the-art algorithms, the high-level MATLAB or Scilab interfaces offer extended functionality, and improved reliability and efficiency over the existing software tools. This is illustrated by the included summary of the numerical results.

REFERENCES

[1] J. Abels and P. Benner, "CAREX – a collection of benchmark examples for continuous-time algebraic Riccati equations (version 2.0)," SLICOT Working Note 1999-14, Nov. 1999,

[2] J. Abels and P. Benner, "DAREX – a collection of benchmark examples for discrete-time algebraic Riccati equations (version 2.0)," SLICOT Working Note 1999-15, Nov. 1999,

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

[4] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, PA, third edition, 1999.

[5] G.J. Balas, J.C. Doyle, K. Glover, A. Packard, and R. Smith, *μ-Analysis and Synthesis Toolbox. For Use with* MATLAB, *Version 4*, The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, Mass, 01760, 2001.

[6] P. Benner, "Accelerating Newton's method for discrete-time algebraic Riccati equations," in *Mathematical Theory of Networks and Systems*, A. Beghi, L. Finesso, and G. Picci, Eds., Il Poligrafo, Padova, Italy, 1998, pp. 569–572.

[7] P. Benner, "Computational methods for linear-quadratic optimization," *Supplemento ai Rendiconti del Circolo Matematico di Palermo, Serie II*, No. 58, pp. 21–56, 1999.

[8] P. Benner and R. Byers, "An exact line search method for solving generalized continuous-time algebraic Riccati equations," *IEEE Trans. Automat. Control*, vol. 43, pp. 101–107, 1998.

[9] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, and A. Varga, "SLICOT - a subroutine library in systems and control theory," in *Applied and Computational Control, Signals, and Circuits*, B.N. Datta, Ed., vol. 1, chapter 10, pp. 499–539. Birkhäuser, Boston, MA, 1999.

[10] 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.*, vol. 86, pp. 17–43, 1997.

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

[12] R.Y. Chiang and M.G. Safonov, *Robust Control Toolbox. For Use with* MATLAB. *Version 2,* The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, Mass, 01760, 2000.

[13] B.N. Datta, "Linear and numerical linear algebra in control theory: Some research problems," *Linear Algebra Appl.*, vol. 197/198, pp. 755–790, 1994.

[14] F. Delebecque and S. Steer, *Integrated Scientific Computing with Scilab*, Birkhäuser, Boston, 1997.

[15] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, "Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms," *ACM Trans. Math. Soft.*, vol. 16, pp. 1–17, 1990.

[16] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, "Algorithm 656: An extended set of Fortran Basic Linear Algebra Subprograms," *ACM Trans. Math. Softw.*, vol. 14, pp. 1–17, 18–32, 1988.

[17] P. Gahinet, A. Laub, and A. Nemirovski, "The LMI Control Toolbox," The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760, 1995.

[18] M. Gawronski, *Balanced control of flexible structures*, Number 211 of *Lecture Notes in Control and Information Sciences*, Springer-Verlag, London, UK, 1996.

[19] M. Gawronski, *Dynamics and control of structures: A modal approach*, Springer-Verlag, Berlin, FRG, 1998.

[20] M. Green, "Balanced stochastic realization," *Linear Algebra Appl.*, vol. 98, pp. 211–247, 1988.

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

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

[23] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, "Basic linear algebra subprograms for FORTRAN usage," *ACM Trans. Math. Software*, vol. 5, pp. 303–323, 1979.

[24] The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, Mass, 01760, MATLAB *Version 6.0.0.88*, 2001.

[25] The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, Mass, 01760, *The* MATLAB *Control Toolbox, Version 5*, 2000.

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

[27] V. Mehrmann, V. Sima, A. Varga, and H. Xu, "A MATLAB MEX-file environment of SLICOT," SLICOT Working Note 1999-11, Aug. 1999,

[28] R. Ober. Balanced parametrizations of classes of linear systems. *SIAM J. Control & Optimiz.*, 29:1251–1287, 1991.

[29] T. Pappas, A.J. Laub, and N.R. Sandell, "On the numerical solution of the discrete-time algebraic Riccati equation," *IEEE Trans. Automat. Control*, vol. AC-25, pp. 631–641, 1980.

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

[31] P. Van Dooren, "A generalized eigenvalue approach for solving Riccati equations," *SIAM J. Sci. Statist. Comput.*, vol. 2, pp. 121–135, 1981.

[32] S. Van Huffel and V. Sima, "SLICOT and control systems numerical software packages," in *Proc. 2002 IEEE Int. Conf. Control Appl. and IEEE Int. Symp. Comp. Aided Cont. Syst. Design, CCA/CACSD 2002, Sep. 18–20, 2002, Glasgow, UK*, P.R. Kalata, Ed., 2002, pp. 39–44. Omnipress, Madison, WI.

[33] A. Varga, "Computation of normalized coprime factorizations of rational matrices," *Sys. Control Lett.*, vol. 271, pp. 37–45, 1998.

Electronic versions of [1], [2], [27] are available from `http://www.win.tue.nl/niconet/NIC2/reports.html`.