Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Computational Experience in Solving Algebraic Riccati Equations

Vasile Sima

Abstract— **State-of-the-art, uni-processor algebraic Riccati equation solvers for automatic control computations are investigated and compared for various problem sizes. General-purpose SLICOT solvers are very efficient for small-size problems, but they cannot compete for larger problems with specialized solvers designed for certain problem classes, such as Newton solvers using low rank Cholesky factors of the solutions of Lyapunov equations built at each iteration.**

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

I. INTRODUCTION

Systems and control algorithms are widely used to model, simulate, and/or optimize industrial, economical, and biological processes. Systems analysis and design procedures often require the solution of general or special linear or quadratic matrix equations. Most high-level algorithms are based on these low-level kernels. The numerical solution of algebraic Riccati equations (AREs) is a cornerstone in computer-aided control systems analysis and design.

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, with $F = BR^{-1}B^T$. The *continuous-time algebraic Riccati equation (CARE)*

$$0 = Q + A^TXE + E^TXA - E^TXFXE \qquad (1)$$

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

$$E^TXE = Q + A^TXA - A^TXB(R + B^TXB)^{-1}B^TXA \qquad (2)$$

play a fundamental role in many computational procedures for linear and nonlinear control problems. Applications include computing linear-quadratic regulators and Kalman filters, solving linear-quadratic Gaussian (H_2-) optimal control problems, finding (sub)optimal H_∞ controllers, and model and controller order reduction based on stochastic and positive real balancing. 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 matrices, or optimal control problems for second-order systems. Common to all these applications is that among the possibly infinitely many solutions of an 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 $\lambda E - (A - B(R + B^TX_*B)^{-1}B^TX_*A)$ in the discrete-time case, are in the appropriate stability region (the left half plane, and the open unit disk, respectively). The $*$ subscript will be dropped in the sequel.

V. Sima is with National Institute for Research & Development in Informatics, 011455 Bucharest, Romania `vsima@ici.ro`

Theoretical results devoted to AREs and related topics abound both in systems and control, as well as in the linear algebra literature; see, e.g., the monographs [1], [2]. Due to their paramount importance, many numerical methods have been proposed for solving AREs; see, e.g., [2], [3]. There are also a lot of associated software implementations, both commercial (e.g., in MATLAB[1] [4]), copyrighted freeware (e.g., in the SLICOT Library [5], [6]), or in the public domain (e.g., in Scilab [7]).

The capabilities and limitations of the general-purpose solvers available in the SLICOT Library and MATLAB are studied, in comparison with some specialized solvers. SLICOT Library (**S**ubroutine **L**ibrary **I**n **CO**ntrol **T**heory) provides Fortran 77 implementations of many numerical algorithms in systems and control theory, as well as standardized interfaces to MATLAB and Scilab. Built around a nucleus of basic numerical linear algebra subroutines from the state-of-the-art software packages LAPACK and BLAS (see [8] and the references therein), the potential of modern high-performance computer architectures can be exploited. While very efficient for small-size problems, the SLICOT and MATLAB solvers cannot compete for larger problems with specialized solvers designed for certain problem classes.

II. ALGEBRAIC RICCATI EQUATIONS AND STANDARD SOLVERS

Let $L \in \mathbb{R}^{n\times m}$. To simplify the notation, define

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

where $\mathrm{op}(M)$ is either M or M^T. Then, *generalized* (descriptor) CAREs and DAREs can be written as

$$\begin{array}{rcll}
0 & = & Q + \mathrm{op}(A)^{\,T}X\,\mathrm{op}(E) + \mathrm{op}(E)^{\,T}X\,\mathrm{op}(A) & (3) \\
 & & - \mathrm{op}(\mathcal{L}(X))\,R^{-1}\,\mathrm{op}(\mathcal{L}(X))^{\,T}, & \\
0 & = & Q - \mathrm{op}(E)^{\,T}X\,\mathrm{op}(E) + \mathrm{op}(A)^{\,T}X\,\mathrm{op}(A) & (4) \\
 & & - \mathrm{op}(\hat{\mathcal{L}}(X))\,\mathrm{op}(\hat{R}(X))^{\,-1}\,\mathrm{op}(\hat{\mathcal{L}}(X))^{\,T}. &
\end{array}$$

These forms are more general than (1) and (2). Standard CAREs and DAREs follow using $E = I_n$, and $L = 0$,

$$\begin{array}{rcll}
0 & = & Q + \mathrm{op}(A)^{\,T}X + X\,\mathrm{op}(A) - X\,\mathrm{op}(F)\,X, & (5) \\
X & = & Q + \mathrm{op}(A)^{\,T}X\,\mathrm{op}(A) & (6) \\
 & & - \mathrm{op}(A)^{\,T}X\,\mathrm{op}(B)\,\mathrm{op}(\hat{R}(X))^{\,-1}\,\mathrm{op}(B)^{\,T}X\,\mathrm{op}(A)\,. &
\end{array}$$

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

The ability to work with the op$(\cdot)$ operator is important for notational convenience. For instance, an optimal regulator problem involves the solution of an ARE with op$(M) = M$, while an optimal estimator problem involves the solution of an ARE with op$(M) = M^T$, B replaced by C, where $C \in \mathbb{R}^{p\times n}$ is the system output matrix, and m replaced by p. The Riccati solution can be used for computing the gain matrix of the optimal regulator, G, or estimator, $K = G^T$,

$$G = R^{-1}\,\mathrm{op}(\mathcal{L}(X))^{\,T}, \tag{7}$$

$$G = \mathrm{op}(\hat{R}(X))^{\,-1}\,\mathrm{op}(\hat{\mathcal{L}}(X))^{\,T}, \tag{8}$$

for continuous-time and discrete-time systems, respectively. The basic methods for solving AREs are the Schur vector method [9] and the deflating subspaces method [10], [11], which use Hamiltonian or symplectic matrices or matrix pencils. The *Hamiltonian matrix* associated to (5) is

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

while the *symplectic matrix* associated to (6) is

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

assuming A is well-conditioned. When A is ill-conditioned, it is preferable to work with the *symplectic pencil*

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

The Schur vector method computes the ordered real Schur form T of the Hamiltonian or symplectic matrix H, i.e., an orthogonal matrix $U \in \mathbb{R}^{2n\times 2n}$ 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 (if H has no eigenvalues on the boundary of the stability region). 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 this solution is $X = U_{21}U_{11}^{-1}$.

The deflating subspace approach proceeds analogously. Define the *extended Hamiltonian pencil* associated to (3),

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

and *extended symplectic pencil* corresponding to (4),

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

Then the ordered generalized real Schur form of $L - \lambda M$ is computed, i.e., orthogonal matrices $U, V \in \mathbb{R}^{(2n+m)\times(2n+m)}$ are found 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 by $G = -U_3U_{11}^{-1}$. 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 approach using the extended pencils yields better numerical accuracy if R is ill-conditioned as rounding errors introduced by forming R^{-1} are avoided.

The methods above are implemented in SLICOT and MATLAB. For the standard continuous-time case (5) with op$(M) = M$, SLICOT also includes an implementation of the matrix sign function method [12]. It is planned to extend SLICOT by integrating CARE and DARE solvers based on Newton's method (with line search). This will be particularly useful for refining an approximate solution to maximal accuracy. Moreover, solvers based on structure-preserving methods for the underlying eigenproblems are under consideration. These methods can deal with situations where eigenvalues of the corresponding matrix (pencil) are close to, or on the boundary of the stability region. SLICOT Library also includes condition estimators for AREs, which enable to assess the solution accuracy and problem sensitivity to small perturbations in the data.

III. SPECIALIZED SOLVERS FOR RICCATI EQUATIONS

The results in [13] and other papers, show that the high-level MATLAB interfaces to the SLICOT codes offer improved efficiency (at comparable accuracy) over the standard software tools, such as those included in MATLAB 6.5.1 and previous versions. The results in this paper (Section IV) also show that SLICOT interfaces have performances comparable with MATLAB 7.0.1 codes. (Note that MATLAB 7.0.1 Lyapunov and Riccati solvers have significantly better efficiency than the solvers available in MATLAB 6.5.1 and previous versions.) However, the SLICOT and MATLAB solvers do have some limitations, mainly coming from their generality. These solvers cannot compete in terms of efficiency with specialized solvers designed for specific classes of large-scale problems, such as the iterative algorithms solving stable Lyapunov equations with low rank solutions.

Several efficient numerical methods for solving large Lyapunov equations appeared in the last years. These methods can be used for solving Riccati equations using Newton's algorithm. One approach was developed in [14] for continuous-time stable Lyapunov equations, extended also for CAREs. The proposed techniques have been incorporated in LYAPACK (LYApunov PACKage), implemented in MATLAB, and freely available on the SLICOT Web site.

The considered Lyapunov equations have the form

$$FX + XF^T = -PP^T, \qquad (12)$$

where $F \in \mathbb{R}^{n\times n}$ and $P \in \mathbb{R}^{n\times t}$. When solving CAREs iteratively, the matrix F has the form $F = A^T - KB^T$. It is assumed that F is stable, A is structured or sparse, $t \ll n$, the order n is large enough, for instance, $n > 500$, and the equations are sufficiently well-conditioned. Sufficient conditions to ensure that $t \ll n$ are $m \ll n$, $p \ll n$, and Q in (5) is given in a factored form, $Q = C^T\hat{Q}C$. A variant of Newton method is used to solve Riccati equations by this approach, and a factor Z of the solution matrix X, $X = ZZ^T$, can be obtained.

The LYAPACK approach uses the *Low Rank Cholesky Factor Alternate Directions Iterations* (LRCF-ADI) technique. The LRCF-ADI performance depends on certain real or complex conjugate *ADI shift parameters*, p, computed by an heuristic algorithm, see [14]. The ADI iteration for Lyapunov equation (12) is given by

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

for $i = 1, 2, \ldots$, where $X_0 = 0$. The technique adopted for Riccati equations is LRCF-NM (Low Rank Cholesky Factor Newton Method), including its implicit variant, LRCF-NM-I, which directly determines the optimal regulator gain matrix, K^T, without computing the Riccati equation solution X. The LRCF-NM and LRCF-NM-I techniques are essentially a combination of the classical Newton method with the LRCF-ADI iteration for Lyapunov equations. Newton method performs the following calculations at the kth iteration:

1) /$*$ Solve in $X^{(k)}$ the stable Lyapunov equation $*$/

$$(A - BK^{(k-1)^T})^T X^{(k)} + X^{(k)}(A - BK^{(k-1)^T}) = -C^T\hat{Q}C - K^{(k-1)}RK^{(k-1)^T};$$

2) $K^{(k)} = X^{(k)}BR^{-1}$; /$*$ Update $K^{(k)}$. $*$/

for $k = 1, 2, \ldots$, which generates a sequence of matrices $X^{(k)}$. Under usual conditions, this sequence converges to the stabilizing solution X, if the initial matrix $K^{(0)}$ is stabilizing, that is, $A - BK^{(0)^T}$ is stable. The convergence is global and ultimately quadratic. A suitable matrix $K^{(0)}$ can be computed using a partial pole placement algorithm, for instance, described in [15].

A. Newton Algorithm Using Low Rank Cholesky Factors

The LRCF-NM technique determines a matrix Z, so that ZZ^H approximates the solution X of (5). Assuming $\hat{Q} \geq 0$, $R > 0$, the matrices $\hat{Q}$ and R can be factored (using, for example, Cholesky factorization) in the form

$$\hat{Q} = \widetilde{Q}\widetilde{Q}^T, \qquad R = \widetilde{R}\widetilde{R}^T,$$

where the matrices $\widetilde{Q} \in \mathbb{R}^{p\times h}$ $(h \leq p)$ and $\widetilde{R} \in \mathbb{R}^{m\times m}$ have maximal rank. Consequently, the Lyapunov equations have the structure

$$F^{(k)}X^{(k)} + X^{(k)}F^{(k)^T} = -P^{(k)}P^{(k)^T}, \qquad (13)$$

where $F^{(k)} = A^T - K^{(k-1)}B^T$, $P^{(k)} = \left[\, C^T\widetilde{Q} \;\; K^{(k-1)}\widetilde{R} \,\right]$, and $P^{(k)}$ has only $t = h + m \ll n$ columns. Therefore, these Lyapunov equations can be efficiently solved using the LRCF-ADI iteration, [14]. The Lyapunov equations solutions form a sequence of approximations to the solutions of Riccati equation (5). The following algorithm is obtained:

Algorithm LRCF_NM :

$\boxed{Z = \texttt{LRCF\_NM}(\ A, B, C, \hat{Q}, R, K^{(0)}\ )}$

/$*$ Compute the Cholesky factor $Z = Z^{(k_{\max})}$ so that ZZ^H approximates the stabilizing solution X of CARE (5). It is assumed that the matrix $A - BK^{(0)^T}$ is stable. $*$/

```
$\hat{Q} = \widetilde{Q}\widetilde{Q}^T$;      $R = \widetilde{R}\widetilde{R}^T$;
FOR $k = 1\colon k_{\max}$,
    $F^{(k)} = A^T - K^{(k-1)}B^T$;  /$*$ 1. Compute $F^{(k)}$. $*$/
    /$*$ 2. Determine (sub)optimal ADI parameters. $*$/
    $\bigl(p_1^{(k)}, p_2^{(k)}, \cdots\bigr) = \texttt{LP\_PARA}(\ F^{(k)}\ )$;
    $P^{(k)} = \left[\, C^T\widetilde{Q} \;\; K^{(k-1)}\widetilde{R} \,\right]$;
    /$*$ 3. Determine Cholesky factor $Z^{(k)}$ for (13). $*$/
    $Z^{(k)} = \texttt{LRCF\_ADI}(\ F^{(k)}, P^{(k)}, p_1^{(k)}, p_2^{(k)}, \cdots\ )$;
    $K^{(k)} = Z^{(k)}\bigl(Z^{(k)^H}BR^{-1}\,\bigr)$;  /$*$ 4. Update $K^{(k)}$. $*$/
END FOR
```

The algorithm LRCF_NM is implemented in the LYAPACK function `lp_lrnm`; the algorithms LP_PARA and LRCF_ADI are implemented in the LYAPACK functions `lp_para` and `lp_lradi`, respectively [14]. The advantage of LRCF_NM is that it delivers an approximation of the low rank Cholesky factor, Z, of the solution X, without storing the solution, sometimes impossible when the system order, n, is very high. Moreover, LRCF_NM often requires a significantly lower computational effort than the standard algorithms, based on the real Schur form reduction.

B. Implicit Newton Algorithm Using Low Rank Factors

The implicit version of the Newton algorithm using low rank Cholesky factors, called LRCF_NM_I, is based on the fact that the solution of the optimal regulator problem is determined by the gain matrix K, which generally has much fewer columns than the low rank Cholesky factor of the solution X of the associated CARE, and than X itself. Mathematically, LRCF_NM_I is equivalent to LRCF_NM, but the approximation of the matrix K is obtained without forming the iterates $Z^{(k)}$ of LRCF_NM, nor the iterates $Z_i^{(k)}$ of the inner algorithm, LRCF_ADI. The calculation of $Z^{(k)}$ at step 3 of LRCF_NM and of $K^{(k)}$ at step 4 are replaced by directly finding the matrix $K^{(k)}$ during the inner LRCF_ADI iterations, via

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

$$K_i^{(k)} := Z_i^{(k)}Z_i^{(k)^H}BR^{-1} = \sum_{j=1}^{i} V_j^{(k)}\bigl(V_j^{(k)^H}BR^{-1}\bigr).$$

Specifically, the steps 3 and 4 above are replaced by

```
        /* 3. Initialize. $\Re p$ denotes the real part of $p$. */
$V_1^{(k)} = \sqrt{-2\Re p_1^{(k)}}\left(F^{(k)} + p_1^{(k)} I_n\right)^{-1} P^{(k)}$;
                                  /* 4. Update $V_i^{(k)}$ and $K_i^{(k)}$. */
FOR $i = 2{:}\, i_{\max}^{(k)}$,
    $V_i^{(k)} = \sqrt{-2\Re p_i^{(k)}/\Re p_{i-1}^{(k)}}\Big(V_{i-1}^{(k)} - \big(p_i^{(k)} +$
            $+ \bar{p}_{i-1}^{(k)}\big)\left(F^{(k)} + p_i^{(k)} I_n\right)^{-1} V_{i-1}^{(k)}\Big)$;
    $K_i^{(k)} = K_{i-1}^{(k)} + V_i^{(k)}\left(V_i^{(k)^H} B R^{-1}\right)$;
END FOR
$K^{(k)} = K_{i_{\max}}^{(k)}$;                       /* 5. Update $K^{(k)}$. */
```

The algorithm LRCF_NM_I is available as an option, called `lp_lrnm_i` below, in the LYAPACK function `lp_lrnm`. This function uses several stopping criteria, including:

- tolerance for the normalized residual norm, NRN;
- stagnation of the normalized residual norm;
- smallness of the relative changes of matrix K;
- stagnation of the relative changes of matrix K.

The normalized residual norm corresponding to the low rank Cholesky factor $Z^{(k)}$ is computed by the formula

$$\mathrm{NRN}(Z^{(k)}) = \frac{\| \mathcal{R}\left(Z^{(k)} Z^{(k)^H}\right) \|_F}{\| C^T \hat{Q} C \|_F},$$

where $\mathcal{R}(X)$ denotes the expression in the right hand side of the CARE (5) as a function of X.

IV. NUMERICAL RESULTS

This section presents few results of an extensive performance investigation of some ARE solvers. The numerical results have been obtained on an Intel Pentium 4 computer at 3 GHz, with 1 GB RAM, with the relative machine precision $\epsilon \approx 2.22 \times 10^{-16}$, using Windows XP (Service Pack 2) operating system, and Compaq Visual Fortran V6.5 compiler. Besides the version MATLAB 7.0.1.24704 (R14) Service Pack 1, some tests used the version MATLAB 6.5.1.199709. The SLICOT-based MATLAB executable MEX-functions have been built using MATLAB 6.5.1.199709 and MATLAB-provided optimized LAPACK and BLAS subroutines (except for LAPACK real Schur form routines `dgees` and `dgges`, since the MATLAB ones did not work). Note that the MATLAB 7.0.1 version for Windows machines does not offer the possibility to directly incorporate optimized LAPACK and BLAS routines.

A. Numerical Results for Standard Algorithms

A first set of tests refer to continuos-time AREs (5) with matrices randomly generated from a uniform distribution in the (0,1) interval, with order n a multiple of 30 (from 30 to 600) and $m = n/5$. (See [16] for the results obtained on random AREs and on SLICOT CARE/DARE benchmark collections using MATLAB 6.1. The system and weighting matrices have been obtained using the MATLAB commands

```
A = rand(n,n);    B = rand(n,m);
Q = rand(n,n) + n*eye(n);  Q = Q + Q';
R = rand(m,m) + m*eye(m);  R = R + R';
```

Fig. 1 presents, in the top part, the execution times using SLICOT `slcares` and MATLAB `care`, and in the bottom part, the ratios between the MATLAB and SLICOT execution times. The SLICOT calculations are usually somewhat faster, and delivered solutions with smaller relative residuals than MATLAB calculations, as illustrated in Fig. 2.

B. Numerical Results for Standard and Specialized Algorithms

The test matrices have been generated using the LYAPACK functions `fdm_2d_matrix` and `fdm_2d_vector`. The matrix $A \in \mathbb{R}^{n\times n}$ is band with 5 nonzero diagonals, obtained by finite difference discretization on an equidistant grid of a partial differential equation for an instationary convection-diffusion heat equation on a unit square with homogeneous first kind boundary conditions. The matrices $B \in \mathbb{R}^n$ and $C^T \in \mathbb{R}^n$ are delivered by the LYAPACK function `fdm_2d_vector`.

Several Riccati equations of small, medium and large size have been considered, solved using the functions `slcaresc` from SLICOT, `care` from MATLAB, and `lp_lrnm(_i)` from LYAPACK. Fig. 3, 4, and 5 illustrate the execution times for the mentioned solvers in the ranges $n \leq 196$, $225 \leq n \leq 400$, and $441 \leq n \leq 1024$, respectively, as well as their ratios taking `slcaresc` as a reference. For these calculations, `slcaresc` and `care` are comparable in

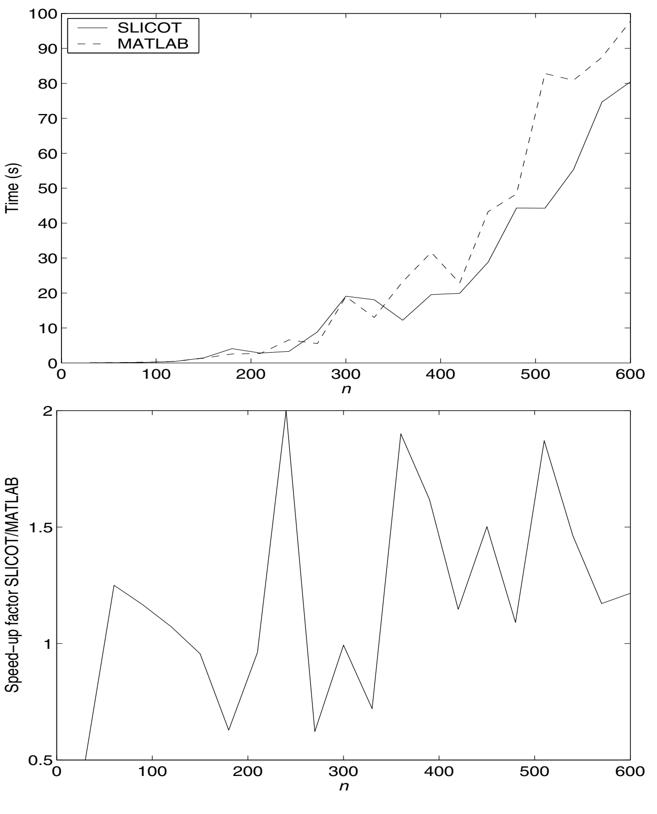

Fig. 1. Top: execution times using SLICOT `slcares` and MATLAB `care`. Bottom: ratios between the MATLAB and SLICOT execution times.

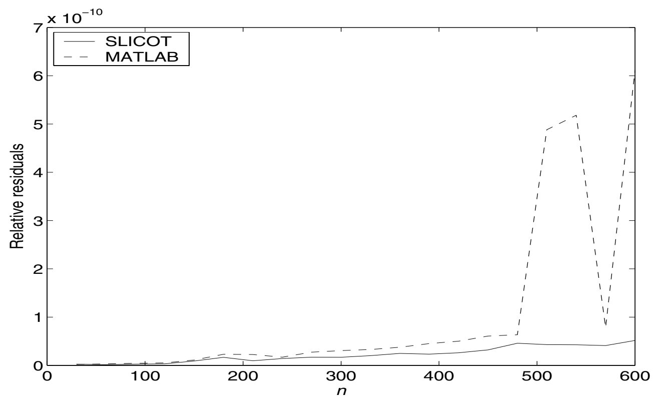

Fig. 2. Relative residuals of the solutions computed by SLICOT and MATLAB.

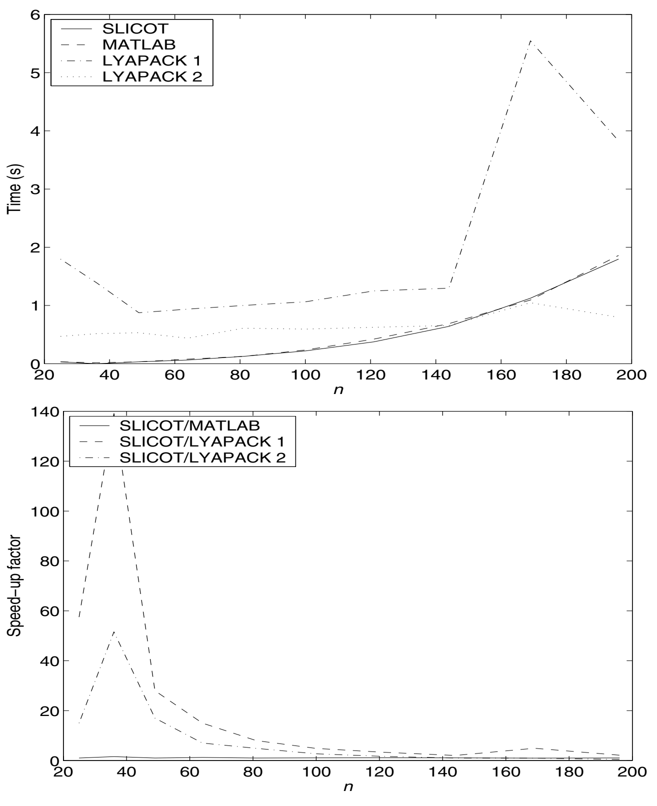

Fig. 3. Riccati solvers comparison, $n \leq 196$. Top: Execution times. Bottom: Ratios between execution times `care/slcaresc`, `lp_lrnm/slcaresc` and `lp_lrnm_i/slcaresc`, or the performance gain of `slcaresc` (SLICOT) compared to `care` (MATLAB), `lp_lrnm` (LYAPACK 1) and `lp_lrnm_i` (LYAPACK 2).

speed for all sizes, with `slcaresc` possibly slightly slower for large orders. Both functions are faster than `lp_lrnm` and `lp_lrnm_i` for orders less than 150, and could be faster than or comparable to `lp_lrnm` also for larger orders, till about 300. But for Riccati equations of order larger than 300, `lp_lrnm` and, especially `lp_lrnm_i`, are very fast for this problem. Actually, the execution times needed for `lp_lrnm` and `lp_lrnm_i` vary approximately linearly with the equation order, while for `slcaresc` and `care` the execution times increase as n^3. The normalized residuals for `lp_lrnm` and `lp_lrnm_i` are approximately one or two orders of magnitude smaller and increase slower with the Riccati equation order than for the other two solvers.

The results above show the superiority of the solvers `lp_lrnm` and `lp_lrnm_i` over `slcaresc` and `care`, for solving Riccati equations of order larger than, say 300, concerning both the execution times and the residuals of computed solutions. However, it should be mentioned that, contrary to `(sl)care(sc)`, `lp_lrnm` and `lp_lrnm_i` are not general solvers. In order to use them advantageously, the following assumptions (and additional ones) must be fulfilled:

- the matrix A is structured or sparse;
- the solution X has a small rank in comparison with n.

The functions `lp_lrnm` and `lp_lrnm_i` use the (sparse) structure of matrix A and operations of the form Ab or $A^{-1}b$, where b is a vector.

For comparison purposes, some results obtained using MATLAB 6.5.1.199709 (R13) Service Pack 1, are illustrated in Fig. 6. For these calculations, the SLICOT function `slcares` is faster than the MATLAB function `care`, and over 10-fold faster for CAREs of order larger than 400. Again, for Riccati equations of order larger than 300, `lp_lrnm` and, especially, `lp_lrnm_i` are much faster.

V. CONCLUSIONS

Various state-of-the-art, uni-processor algebraic matrix Riccati equation solvers for automatic control computations have been investigated and compared for various problem sizes. The results confirm the natural expectation that general-purpose solvers, such as those currently implemented in the SLICOT Library and MATLAB, cannot compete in efficiency, for large-scale problems, with specialized solvers designed for certain problem classes. One specialized approach, applicable to a class of continuous-time algebraic Riccati equation, uses an iterative algorithm based on low rank Cholesky factorization. The structure of the matrix $A - BK^T$ is exploited. It is assumed that the right hand side terms of the Lyapunov equations solved at each iteration of the Newton algorithm, and the solution X, have a small rank. However, the SLICOT solvers are very efficient for small-size problems. Moreover, they are general solvers and offer extended functionality and broad computational abilities.

REFERENCES

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

[2] V. Mehrmann, *The Autonomous Linear Quadratic Control Problem. Theory and Numerical Solution*, ser. Lect. Notes in Control and Information Sciences. Berlin: Springer-Verlag, 1991, vol. 163.

[3] V. Sima, *Algorithms for Linear-Quadratic Optimization*, ser. Pure and Applied Mathematics: A Series of Monographs and Textbooks. New York: Marcel Dekker, Inc., 1996, vol. 200.

[4] *The* MATLAB *Control Toolbox Version 5. For Use with* MATLAB, The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760–2098, 2000.

[5] 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*, Boston: Birkhäuser, 1999, vol. 1, chapter 10, pp. 499–539.

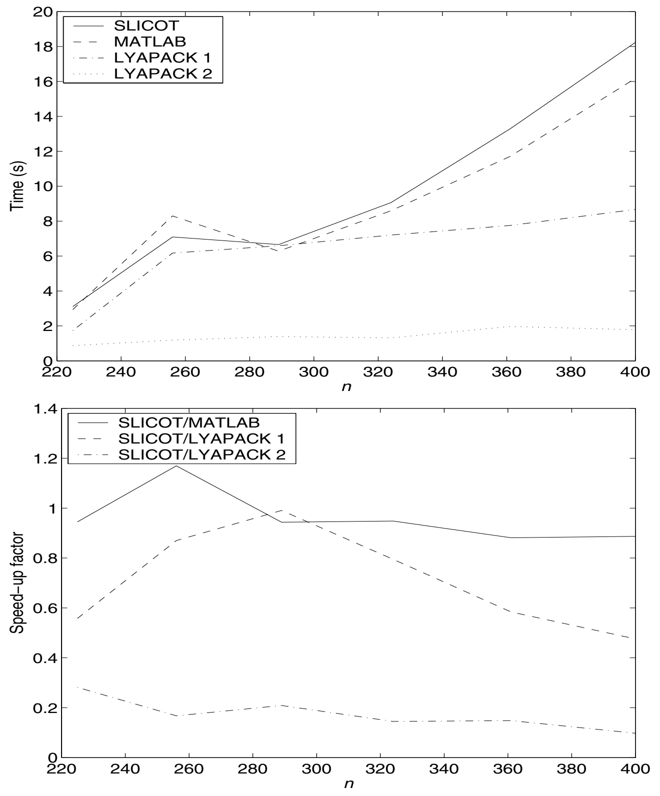

Fig. 4. Riccati solvers comparison, $225 \leq n \leq 400$. Top: Execution times. Bottom: Ratios between execution times `care/slcaresc`, `lp_lrnm/slcaresc` and `lp_lrnm_i/slcaresc`.

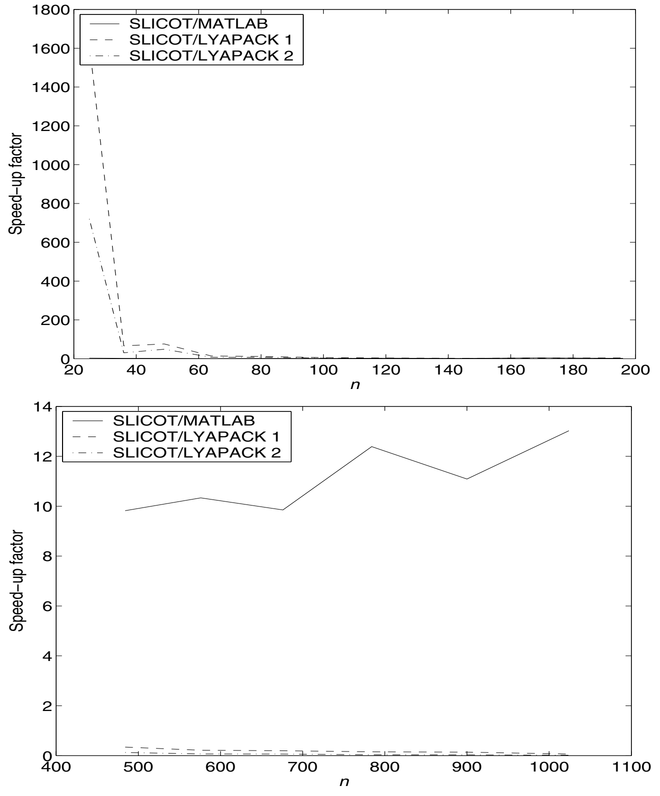

Fig. 6. Riccati solvers comparison under MATLAB 6.5.1: Ratios between execution times `care/slcares`, `lp_lrnm/slcares` and `lp_lrnm_i/slcares`. Top: $n \leq 196$. Bottom: $441 \leq n \leq 1024$.

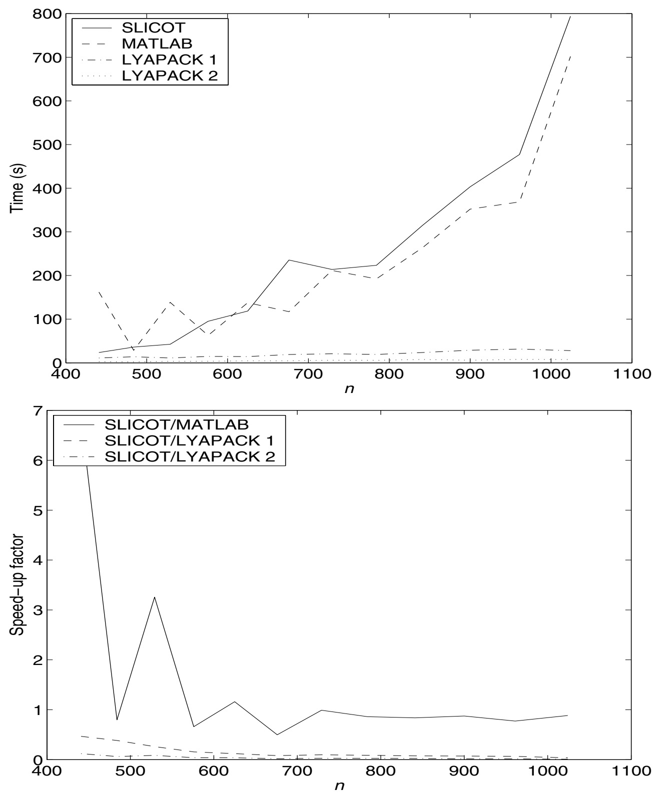

Fig. 5. Riccati solvers comparison, $441 \leq n \leq 1024$. Top: Execution times. Bottom: Ratios between execution times `care/slcaresc`, `lp_lrnm/slcaresc` and `lp_lrnm_i/slcaresc`.

[6] S. Van Huffel, V. Sima, A. Varga, S. Hammarling, and F. Delebecque, "High-performance numerical software for control," *IEEE Control Syst. Mag.*, vol. 24, no. 1, pp. 60–76, 2004.

[7] C. Gomez, Ed., *Engineering and Scientific Computing with Scilab*. Boston: Birkhäuser, 1999.

[8] 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: Third Edition*, ser. Software · Environments · Tools. Philadelphia: SIAM, 1999.

[9] A. J. Laub, "A Schur method for solving algebraic Riccati equations," *IEEE Trans. Automat. Contr.*, vol. AC–24, no. 6, pp. 913–921, 1979.

[10] T. Pappas, A. J. Laub, and N. R. Sandell, "On the numerical solution of the discrete-time algebraic Riccati equation," *IEEE Trans. Automat. Contr.*, vol. AC–25, no. 4, pp. 631–641, 1980.

[11] P. Van Dooren, "A generalized eigenvalue approach for solving Riccati equations," *SIAM J. Sci. Stat. Comput.*, vol. 2, no. 2, pp. 121–135, 1981.

[12] R. Byers, "Solving the algebraic Riccati equation with the matrix sign function," *Lin. Alg. Appl.*, vol. 85, no. 1, pp. 267–279, 1987.

[13] V. Sima and P. Benner, "Solving linear matrix equations with SLICOT," in *Proceedings of the European Control Conference ECC'03*, 1–4 September, 2003, Cambridge, UK, Special Session *Matrix Equations in Systems and Control*.

[14] T. Penzl, "LYAPACK Users Guide," Technische Universität Chemnitz, Sonderforschungsbereich 393, "Numerische Simulation auf massiv parallelen Rechnern", Chemnitz, Tech. Rep. SFB393/00–33, Aug. 2000.

[15] A. Varga, "A Schur method for pole assignment," *IEEE Trans. Automat. Contr.*, vol. AC–26, no. 2, pp. 517–519, 1981.

[16] P. Benner and V. Sima, "Solving algebraic Riccati equations with SLICOT," in *CD-ROM Proceedings of The 11th Mediterranean Conference on Control and Automation MED'03*, June 18–20 2003, Rhodes, Greece, invited session IV01, "Computational Toolboxes in Control Design", Paper IV01-01, 6 pp.