ALGORITHM GRICSR SOLVING CONTINUOUS-TIME ALGEBRAIC RICCATI EQUATIONS USING GAUSSIAN SYMPLECTIC TRANSFORMATIONS

VASILE SIMA

Computer Process Control Laboratory
Research Institute for Informatics
8-10 Averescu Avenue
71316 Bucharest 1
ROMANIA

ABSTRACT

An efficient algorithm — GRICSR — for solving continuous-time algebraic matrix Riccati equations (CAREs) is presented and numerical results obtained by using its implementation are discussed. The algorithm employs condition-controlled Gaussian symplectic transformations, a symmetric updating scheme for computing the stabilizing solution of CARE, and accurate approximations of the eigenvalues of the Hamiltonian matrix involved in the optimal control problem; these eigenvalues are used as shifts in an QR-like process. The Hamiltonian structure is preserved throughout the algorithm. The main computational steps are sketched, and details of the Fortran implementation are mentioned. Numerical results show that the algorithm can be used safely, even when large values of condition numbers of the transformation matrices are allowed.

Keywords: Computational methods; control system design; eigenvalues; Hamiltonian matrices; invariant subspaces; linear algebra; optimal control; symplectic matrices.

1. INTRODUCTION

Many control/estimation analysis and design problems include, as a basic procedural step, the solution of an algebraic matrix Riccati equation (ARE). The AREs are a fundamental tool for modern control engineering and other domains. For instance, the application of the new H_∞ theory is largely dependent on the numerical solution of AREs. Therefore, it is of primary importance to have efficient techniques and algorithms for solving AREs.

This paper discusses a recently developed technique for solving the continuous-time algebraic matrix Riccati equation (CARE) of the form

$$A^T X + XA - XBR^{-1}B^T X + Q = 0, \tag{1}$$

where $A \in \mathrm{R}^{n\times n}$ and $B \in \mathrm{R}^{n\times m}$ are the system state and control (input) matrices, respectively, $Q \in \mathrm{R}^{n\times n}$ and $R \in \mathrm{R}^{m\times m}$ are the performance index state and control weighting matrices, respectively, and the superscript "T" denotes the transposition. Usually, $Q = Q^T \geq 0$ and $R = R^T > 0$. In some applications, Q is chosen as $C^T\tilde{Q}C$, where $C \in \mathrm{R}^{l\times n}$ is the system output matrix and $\tilde{Q}$ is the output weighting matrix. Frequently, a set of "error integrators" are incorporated in the control system, to improve the stationary properties of the controlled process, by using an integral action in the controller; the error can be the difference between the desired and actual output. The above formulation corresponds to an optimal control problem,

but it can be easily translated to an optimal filtering problem. If certain well-known conditions hold, then CARE (1) has a unique solution $X = X^T \geq 0$. This is called the *stabilizing solution*, because the corresponding closed-loop system is stable, that is $\lambda(A + BG) \subset \mathbb{C}^-$, where $\lambda(M)$ denotes the spectrum of the matrix M, and $G \in \mathbb{R}^{m\times n}$ is the *optimal control matrix*, defined as $G \triangleq -R^{-1}B^TX$.

It is well-known (Laub, 1979) that the stabilizing solution of an ARE can be obtained taking a basis for the stable invariant subspace of a matrix of order $2n$, defined in terms of the system and performance index matrices. For instance, denoting $H \triangleq \begin{bmatrix} A & F \\ Q & -A^T \end{bmatrix}$, $F \triangleq BR^{-1}B^T$, if $[S_{11}^T \quad S_{21}^T]^T \in \mathbb{R}^{2n\times n}$ generates the stable H-invariant subspace and S_{11} is nonsingular, then $X = -S_{21}S_{11}^{-1}$ is the stabilizing solution of (1). (If matrix H is not dichotomic, that is it has an eigenvalue $\lambda \in \lambda(H)$ with $\mathrm{Re}(\lambda) = 0$, or if S_{11} is singular, then there is no such solution of (1).) The QR algorithm has been used for determining and ordering the real Schur form (RSF) of matrix H, to obtain such a basis. However, this approach has the disadvantage of not exploiting the particular structure of H, involving special properties of its spectrum. (Matrix H is *Hamiltonian*, hence the eigenvalues appear in pairs $(\lambda, -\lambda)$ and, in case of dichotomy, there are exactly n stable eigenvalues.) Moreover, the symmetric matrix X is essentially obtained as a product of two matrices, S_{21} and S_{11}^{-1}, and this could lead to unsymmetry and inaccuracy.

This paper uses a structure preserving algorithm for solving CARE. The key idea is to compute a basis for the stable invariant subspace by a structure preserving QR-like algorithm, exclusively using similarity transformations with symplectic matrices. Such and algorithm, called the *SR algorithm*, has been proposed by Bunse-Gerstner and Mehrmann (1986) and uses *orthogonal* symplectic matrices as much as possible in practice. Another, more efficient algorithm (Bunse-Gerstner, Mehrmann and Watkins, 1989), which constitutes the basis for the implementation discussed below, exclusively uses non-orthogonal transformations. A variant of the SR algorithm based on Gaussian elimination is used, combined with the symmetric updating method in (Byers and Mehrmann, 1985), and with Van Loan's algorithm (Van Loan, 1984) for getting accurate approximations of the eigenvalues of H. In conjunction with Newton's algorithm in a defect correction procedure (Mehrmann and Tan, 1988), this approach is very efficient and accurate. Algorithmic details and computational experience with GRICSR are presented in (Sima, 1992). A short presentation is included in (Sima, 1991).

Several definitions are needed for the following presentation. A matrix S is *symplectic* if $S^TJS = J$, where $J \triangleq \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}$, and I is the identity matrix of order n (also written I_n). A matrix $M \triangleq \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$, $M_{ij} \in \mathbb{R}^{n\times n}$, is called *J-Hessenberg (matrix)* if M_{11}, M_{21} and M_{22} are upper triangular matrices, and M_{12} is in Hessenberg form (HF). M is called *J-triangular* if M_{11}, M_{12}, M_{21} and M_{22} are upper triangular, and M_{21} has zero diagonal elements, and M is *J-tridiagonal* if M_{11}, M_{21} and M_{22} are diagonal, and M_{12} is tridiagonal.

Let $\mathcal{S}_{2n}$ denote the class of symplectic matrices of order $2n$. By definition, if $S, S' \in \mathcal{S}_{2n}$ and S is nonsingular, then $S^{-1} \in \mathcal{S}_{2n}$ and $SS' \in \mathcal{S}_{2n}$. Given a matrix $M \in \mathbb{R}^{2n\times 2n}$, a decomposition $M = SR$, where $S \in \mathcal{S}_{2n}$ and R is J-triangular, is called an *SR decomposition* of M. There are mathematical results

(Bunse-Gerstner and Mehrmann, 1986) which characterize the matrices that have no SR decompositions, or transformations to J-Hessenberg form. Briefly speaking, the set of matrices with no SR decompositions has the Lebesgue measure zero and the SR decomposition of a matrix M is essentially *unique*.

In the following section, the notation "$i = n : m$" means that the variable i successively takes the values $n, n+1, \ldots, m$; similarly, $A(i : j, k : l)$ denotes the submatrix formed by the rows $i : j$ and columns $k : l$ of the matrix A.

2. OUTLINE OF SOLUTION TECHNIQUE

Algorithm GRICSR reduces a Hamiltonian matrix H to a special ordered RSF matrix, using elementary symplectic transformations with the Gaussian elimination matrices, defined below. Let $k \in \{1, \ldots, n\}$, $v \in \mathrm{R}$, $w = (0, \ldots, 0, w_{k+1}, \ldots, w_n)^T \in \mathrm{R}^n$, let e_k be the k-th coordinate vector in R^n, and let $\gamma(L, V) \triangleq \begin{bmatrix} L & V \\ 0 & L^{-T} \end{bmatrix}$. Define

$$G_1(k, v) = \gamma(I_n, V_1), \quad G_2(k, v) = \gamma(I_n, V_2), \quad G_3(k, w) = \gamma(L, 0), \tag{2}$$

where $V_1 = v(e_{k-1}e_k^T + e_k e_{k-1}^T)$, $V_2 = ve_k e_k^T$, $L = I_n + we_k^T$. Note that G_1, G_2 and G_3 are symplectic, easily invertible, and can be used to annihilate certain elements of any vector $x \in \mathrm{R}^{2n}$. As in the case of Gaussian elimination, pivoting is needed for numerical stability. The permutation matrices involved must also be symplectic. One uses $P_{k,l} = \mathrm{diag}(P, P)$, where P interchanges the k-th and l-th elements of x. Such a permutation is denoted $P(k, x)$. Since the matrices G_1, G_2 and G_3 may be ill-conditioned (if $|v|$ or $\| w \|$ is large), it is necessary that their condition would be monitored and remedies should be provided for unacceptable situations. This question will be further discussed.

Algorithm 1 Reduction of a Hamiltonian matrix $H \in \mathbf{R}^{2n \times 2n}$ to J-tridiagonal form using Gaussian symplectic transformations.

1) For $j = 1 : n-1$
   1) $H \leftarrow THT^{-1}, S \leftarrow ST^{-1}, T = G_3^T(j+1, He_j)P(n+j+1, He_j)$.
   2) $H \leftarrow THT^{-1}, S \leftarrow ST^{-1}, T = G_2^T(j+1, He_j)$.
   3) $H \leftarrow THT^{-1}, S \leftarrow ST^{-1}, T = G_3(j+1, He_j)P(j+1, He_j)$.
   4) If $h_{j+1,j} \neq 0$ and $h_{n+j,j} = 0$ then Stop, else
      $H \leftarrow THT^{-1}, S \leftarrow ST^{-1}, T = G_1(j+1, e_{j+1}^T He_j)$.
   5) $H \leftarrow THT^{-1}, S \leftarrow ST^{-1}, T = G_3^T(j+1, He_{n+j})P(n+j+1, He_{n+j})$.
   6) $H \leftarrow THT^{-1}, S \leftarrow ST^{-1}, T = G_2^T(j+1, He_{n+j})$.
   7) $H \leftarrow THT^{-1}, S \leftarrow ST^{-1}, T = G_3(j+1, He_{n+j})P(j+1, He_{n+j})$.

The various transformations T are not actually computed, but applied to H and accumulated in S. If $S = I_{2n}$ on input to Algorithm 1, then S contains on output the transformation matrix that reduces H to J-tridiagonal form. If S is chosen so that $Se_1 = \lambda e_1$, then this property is preserved by the algorithm.

A gain in efficiency for Algorithm 1 is possible if, instead of accumulation, a direct updating of the matrices corresponding to the solution of Riccati equation takes place, exploiting the symmetry (Byers and Mehrmann, 1985). In this case, not the entire matrix $S = (S_{ij})$, $i, j = 1 : 2$, must be stored, but only S_{11}^{-1}, $X = -S_{21}S_{11}^{-1}$

and $Y = S_{11}^{-1}S_{12}$, where X and Y are symmetric matrices, since S is symplectic. All the information required about S can be reconstructed from the matrices S_{11}^{-1}, X and Y, which take half of the memory space for S. Indeed, S_{11}^{-1}, X and Y can be updated after having applied the transformations with matrices G_1, G_2 and G_3 from (2). Let $\tilde{S}$ be the updated matrix S after such a transformation. In case of updating by G_1, the results are

$$\tilde{S}_{11}^{-1} = S_{11}^{-1}, \quad \tilde{X} = X, \quad \tilde{Y} = S_{11}^{-1}(-S_{11}V_1 + S_{12}) = Y - V_1. \tag{3}$$

In case of updating by G_3, the results are

$$\tilde{S}_{11}^{-1} = LS_{11}^{-1}, \quad \tilde{X} = X, \quad \tilde{Y} = LYL^T. \tag{4}$$

The results corresponding to updating by G_3^T are similar. Finally, in case of updating by G_2^T, denoting $\alpha \triangleq 1 - ve_k^T Y e_k$, the results are

$$\tilde{S}_{11}^{-1} = S_{11}^{-1} + \frac{v}{\alpha} Y e_k e_k^T S_{11}^{-1}, \quad \tilde{X} = X + \frac{v}{\alpha} S_{11}^{-T} e_k e_k^T S_{11}^{-1}, \quad \tilde{Y} = Y + \frac{v}{\alpha} Y e_k e_k^T Y. \tag{5}$$

Note that $\alpha \neq 0$ was assumed; if $\alpha = 0$, then $\tilde{S}_{11}$ is singular. Obviously, no use of (5) will be recommended if the *updating factor* $|v/\alpha|$ is large (for instance, if α is negligible compared with v, to machine precision), because the loss of accuracy could be then significant.

Using the formulae (3) – (5), the cost of updating the matrices X, Y and S_{11}^{-1} in Algorithm 1, adapted to this case, is almost half of the initial value. Some other advantage of using the updating relations is that the symmetry of X is preserved at each stage of the algorithm. This is valid for the next algorithms too.

Algorithm 1, for reducing a Hamiltonian matrix to J-tridiagonal form, is used to compute the RSF, in the same way as the reduction to HF is used in the QR algorithm. The *SR algorithm* below is the most general algorithm performing this computation by exclusively using symplectic transformations which preserve the Hamiltonian structure. All the computations are possible by means of the G_1, G_2, G_3 and P transformations.

Algorithm 2 Reduction of a Hamiltonian matrix to the RSF using symplectic transformations.

1) $H_1 = S_0^{-1} H S_0$, $S = S_0$, $S_0 \in \mathcal{S}_{2n}$, H_1 J-tridiagonal.
2) For $k = 1, 2, \ldots$
   1) $H_{k+1} = S_k^{-1} H_k S_k$, $S \leftarrow SS_k$, $S_k \in \mathcal{S}_{2n}$, H_{k+1} J-tridiagonal, where S_k is obtained from the SR decomposition, $p_k(H_k) = S_k R_k$, of a polynomial p_k in H_k.

The polynomials $p_k(x)$ are usually chosen as either $x - \lambda_k$, where $\lambda_k = H_k(2n, 2n)$, or $(x - \lambda_k)(x - \bar{\lambda}_k)$, where $\lambda_k, \bar{\lambda}_k \in \lambda(H_k(2n-1:2n, 2n-1:2n))$. However, since if $\lambda \in \lambda(H) \setminus \mathrm{R}$, then $-\lambda$, $\bar{\lambda}$, $-\bar{\lambda} \in \lambda(H)$, one can use a polynomial of degree four, $p_k(x) = (x - \lambda_1)(x - \lambda_2)(x - \lambda_3)(x - \lambda_4)$, where λ_1 to λ_4 are the eigenvalues of a submatrix of the form (for $j = n - 1$)

$$N_j = \begin{bmatrix} a_j & 0 & c_j & b_j \\ 0 & a_{j+1} & b_j & c_{j+1} \\ q_j & 0 & -a_j & 0 \\ 0 & q_{j+1} & 0 & -a_{j+1} \end{bmatrix}. \tag{6}$$

In general, for large values of k, H_k and $p_k(H_k)$ have SR decompositions (since $\{H_k\}$ tends to a J-tridiagonal matrix). But for small values of k, it is possible that $p_k(H_k)$ has no SR decomposition, or is close to such a matrix. This fact will be detected by obtaining a large condition number for a transformation G_i, $i \in \{1,2,3\}$. If a condition number exceeds a certain bound, c_b, the current iteration is abandoned and a random symplectic matrix is used; one says that an *exceptional* transformation is employed. To avoid an infinite loop, no more than 20 exceptional transformations are allowed per iteration.

Algorithm 2 is implemented without explicitly computing the decomposition $p_k(H_k) = S_k R_k$. The following computations must be performed at each iteration k. The first column x of $p_k(H_k)$ is computed. Then, a symplectic matrix $\tilde{S}$ that reduces x to a multiple of e_1 (in R^{2n}) is determined, the matrix $\tilde{S}H_k\tilde{S}^{-1}$ is calculated and, finally, the symplectic reduction of $\tilde{S}H_k\tilde{S}^{-1}$ to J-tridiagonal form, $H_{k+1} = \hat{S}\tilde{S}H_k\tilde{S}^{-1}\hat{S}^{-1}$, is performed, updating the matrices S_{11}^{-1}, X and Y accordingly; if S_{11}^{-1} is (nearly) singular, then an exceptional transformation is used. If p_k is the four degree polynomial above, then $x \in \mathrm{Im}(e_1, e_2, e_3)$, hence $\tilde{S}$ is simply computed, and $\tilde{S}H_k\tilde{S}^{-1}$ is a J-tridiagonal Hamiltonian matrix, except for six extra nonzero elements in the upper-left corner of each submatrix $H_{ij} \in \mathrm{R}^{n\times n}$, $i,j = 1:2$. In order to reduce this matrix to J-tridiagonal form, a specialization of Algorithm 1 is used.

In general, the sequence $\{H_k\}$ converges to a matrix with 4×4 principal submatrices like (6), each having the spectrum of the form $\{\lambda, -\lambda, \bar{\lambda}, -\bar{\lambda}\}$, where $\lambda \in \mathrm{C} \setminus \{\mathrm{R} \cup \mathrm{Rj}\}$, that is not all b_i elements will necessarily tend to zero. If some eigenvalues are real, then 2×2 submatrices will also appear. At the beginning of each iteration, one checks if some elements b_j are negligible, that is smaller (in magnitude) than $\varepsilon \triangleq \varepsilon_M \parallel H \parallel_1$, where ε_M is the relative machine precision. If $|b_j| < \varepsilon$, then the eigenvalue problem is decomposed in two smaller subproblems which are solved consecutively. The iterative process continues until a complete splitting into 2×2 or 4×4 subproblems occurs, that is until at least every second element b_* is negligible.

The next stage is the separation of the stable H-invariant subspace. Let $N_j = \begin{bmatrix} a_j & c_j \\ q_j & -a_j \end{bmatrix}$ be a resulted 2×2 submatrix. Its eigenvalues are $\lambda = (a_j^2 + c_j q_j)^{\frac{1}{2}}$ and $-\lambda$. If $q_j = 0$ and $a_j = -\lambda$, the spectrum covers the desired form and no further processing is needed. If $q_j = 0$ and $a_j = \lambda$, then the desired form is obtained using a G_2-type transformation, so chosen that $G_2^T[c_j \quad -2a_j]^T$ has the second element zero. Hence, it follows that $v = 2a_j/c_j$. Finally, if $q_j \neq 0$, the desired form is obtained by applying a G_2-type transformation, so chosen that $G_2^T[a_j - \lambda \quad q_j]^T$ has the second element zero. Therefore, $v = -q_j/(a_j - \lambda)$ and this value gives $G_2^T N_j G_2^{-T} = \begin{bmatrix} -\lambda & c \\ 0 & \lambda \end{bmatrix}$.

Similar ideas are used to order the eigenvalues for an 4×4 submatrix (6) with $b_j \neq 0$. Clearly, the computing formulae are more complicated (Sima, 1992).

The eigenvalues of N_j in (6) are computed considering that $\lambda(N_j) = \{\lambda, \mu, -\lambda, -\mu\}$, where $\mu = \bar{\lambda}$, hence $\lambda(N_j^2) = \{\lambda^2, \mu^2\}$, $\mu^2 = \bar{\lambda}^2$. Since $\lambda(N_j^2) = \lambda(V)$, where $V \triangleq \begin{bmatrix} a_j^2 + q_j c_j & q_{j+1} b_j \\ q_j b_j & a_{j+1}^2 + q_{j+1} c_{j+1} \end{bmatrix}$, λ^2 and μ^2 are the roots of the appropriate

quadratic equation.

The convergence rate of Algorithm 2 is improved by replacing the eigenvalues λ_i, $i = 1:4$, defining the polynomials p_k, with the corresponding eigenvalues of matrix H, economically computed in advance (with both relative and absolute great accuracy) using the following algorithm (Van Loan, 1984).

Algorithm 3 Reduction of the square of a Hamiltonian matrix H.

1) $W = H^2$.
2) $S^T W S = \begin{bmatrix} \bar{Q} & R \\ 0 & -\bar{Q}^T \end{bmatrix}$, $S \in \mathcal{S}_{2n}$, $S^T S = I_{2n}$, $\bar{Q}$ in HF.

One says that the matrix on the right hand side at step 2) of Algorithm 3 is in *Hamiltonian-Hessenberg form*. The use of an orthogonal symplectic matrix S certifies good numerical properties. The matrix W is not actually constructed, but all the necessary operations are implicitly performed on the $n \times n$ submatrices of the Hamiltonian matrix H. Only the submatrix $H(1:n, 1:n)$ and the lower triangles of $H(1:n, n+1:2n)$ and $H(n+1:2n, 1:n)$ are stored. The eigenvalues of H are obtained knowing that $\lambda_i = (\lambda_i(\bar{Q}))^{\frac{1}{2}}$, $\lambda_{n+i} = -\lambda_i$, $i = 1:n$, where $\lambda_i(\bar{Q})$, $i = 1:n$, are computed by the QR algorithm, applied to the matrix $\bar{Q}$ in HF.

All the features above are combined in the following algorithm.

Algorithm 4 Solving algebraic matrix Riccati equation (1) using symplectic transformations.

1) Apply Algorithm 3 to the matrix $H = \begin{bmatrix} A & F \\ Q & -A^T \end{bmatrix}$.
2) Compute the stable eigenvalues of H, $\lambda_i = -(\lambda_i(\bar{Q}))^{\frac{1}{2}}$.
3) Initialize the matrices: $X = 0$, $Y = 0$, $S_{11}^{-1} = I_n$.
4) Reduce H to J-tridiagonal form, using Algorithm 1, but updating S_{11}^{-1}, X and Y according to relations (3) – (5), instead of accumulating the transformations.
5) Apply Algorithm 2, but update S_{11}^{-1}, X and Y, instead of accumulating the transformations. As polynomial p_k, choose either $(\lambda - \lambda_i)(\lambda + \lambda_i)$, or $(\lambda - \lambda_i)(\lambda - \bar{\lambda}_i)(\lambda + \lambda_i)(\lambda + \bar{\lambda}_i)$, depending on $\lambda_i \in \mathrm{R}$ or $\lambda_i \in \mathrm{C}$, respectively, where i is initialized by n and is decreased by 1 or 3, respectively, until convergence is achieved.

The use of the eigenvalues which are computed based on Algorithm 3 as (double or quadruple) implicit shifts, enables a very fast convergence of the SR algorithm. As experimentally proved, the average number of SR iterations per stable eigenvalue is usually 1. Bunse-Gerstner, Mehrmann and Watkins (1989) estimate that the algorithm requires about $28n^3$ *flops*, but the overhead is quite involved, compared to other algorithms. The memory space needed for the data is of $6n^2 + m^2 + nm + 5n$ locations.

Algorithm 4 computes an approximate symmetric solution $\hat{X}$ of CARE. Since Gaussian elementary transformations are used, $\hat{X}$ may be quite inaccurate, if the transformation matrices used in the process have large condition numbers. Therefore, if the norm of the residual of $\hat{X}$ is too large, relative to the norm of $\hat{X}$, then

the Newton method — as described in (Sima, 1990) — is used in an iteration of the defect correction procedure (Mehrmann and Tan, 1988), to improve the accuracy of solution. This enables a large bound for the condition numbers be employed (for instance, $c_b = 10^{-2}/\epsilon_M$), and unnecessary exceptional transformations be avoided.

3. USAGE

Besides GRICSR subroutine, the called routines SQRED and GJTRID, which can be used independently, are also presented. Only the double precision versions are currently available.

GRICSR subroutine computes an approximate solution of the continuous-time algebraic matrix Riccati equation (1) using the Gaussian symplectic transformations method.

SQRED subroutine performs the implicit reduction of the square of a Hamiltonian matrix to Hamiltonian-Hessenberg form, using *orthogonal* symplectic transformations.

GJTRID subroutine performs the reduction of a Hamiltonian matrix to J-tridiagonal form, using Gaussian symplectic transformations.

3.1. Description of User's Interface

Calling sequences

```
      CALL GRICSR (N,M,A,B,Q,R,NIT,CB,X,IER,F,Y,SINV,REIG,IEIG,V,W,
     1             NA,NB,NQ,NR,NX)
      CALL SQRED  (N,A,G,H,X,W,NA)
      CALL GJTRID (A,Q,F,X,Y,SINV,D1,D2,BS,N,V,W,ESC,CM,CP,CB,BAD,
     1             FAIL,NA,NQ,NX)
```

Input parameters

N – is an integer variable that specifies the order of the matrices A, Q and X, and the number of rows of matrix B. **N** represents the dimension of the system state vector. **N** must be no greater than **NA**, **NB**, **NQ** and **NX**.

M – is an integer variable that specifies the order of matrix R and the number of columns of matrix B. **M** represents the dimension of the system input vector. **M** must be no greater than **NR**.

A – is a double precision real two-dimensional array with row dimension **NA** and column dimension at least **N**. **A** contains the **N** by **N** system state matrix.

B – is a double precision real two-dimensional array with row dimension **NB** and column dimension at least **M**. **B** contains the **N** by **M** system input matrix. **B** is not modified on output.

Q – is a double precision real two-dimensional array with row dimension NQ and column dimension at least N. Q contains the N by N symmetric positive semi-definite state weighting matrix, in full storage. The elements of the strict upper triangle are not referenced.

R – is a double precision real two-dimensional array with row dimension NR and column dimension at least M. R contains the M by M symmetric positive definite input weighting matrix, in full storage. The elements of the strict lower triangle are not referenced.

NIT – is an integer variable that specifies the maximal number of iterations of the SR algorithm.

CB – is a double precision real variable that contains the maximal allowable value for the condition numbers of transformation matrices.

X – is a double precision real two-dimensional array with row dimension N and column dimension at least N.

GRICSR first uses X as a working space and then initializes the strict lower triangle to 0.

On the first call of GJTRID, the routine uses the initialization provided by GRICSR and then updates the lower triangle of array X, corresponding to updating the matrix X.

In SQRED, X is used as a working array.

F – is a double precision real two-dimensional array with row dimension N and column dimension at least N.

GRICSR initializes its lower triangle to the lower triangle of matrix $F \triangleq BR^{-1}B^T$ and then uses its strict upper triangle as a working area.

GJTRID uses the initialization provided by GRICSR in the lower triangle.

Y – is a double precision real two-dimensional (working) array with row dimension N and column dimension at least N.

GRICSR initializes the lower triangle of Y to 0 and uses its strict upper triangle as a working space.

On the first call of GJTRID, the routine uses the initialization provided by GRICSR and then updates the lower triangle of array Y, corresponding to updating the matrix Y.

SINV – is a double precision real two-dimensional (working) array with row dimension N and column dimension at least N+1 in GRICSR and N in GJTRID.

GRICSR first uses SINV as a working space and then initializes the first N by N locations to I_n.

On the first call of GJTRID, the routine uses the initialization provided by GRICSR and then updates the content of array SINV, corresponding to updating the matrix S_{11}^{-1}.

`V,W` – are double precision real one-dimensional working arrays of dimensions at least `N`. In GRICSR, `W` may be dimensioned max(`N`,6).

`NA,NB, NQ,NR,NX` – are integer variables set equal to the row dimensions of the two-dimensional arrays `A`, `B`, `Q`, `R` and `X`, respectively.

`G` – is a double precision real one-dimensional array of dimension `N*(N+1)/2`. `G` contains the lower triangle, packed column-wise, of the symmetric matrix $F \triangleq BR^{-1}B^T$.

`H` – is a double precision real one-dimensional array of dimension `N*(N+1)/2`. `H` contains the lower triangle, packed column-wise, of the symmetric matrix Q.

`D1,D2` – are integer variables set equal to the lower and upper indices, respectively, of the submatrices in A, Q and F which must be dealt with in the reduction to J-tridiagonal form.

`BS` – is an integer variable that specifies the maximal number of subdiagonal elements in a column which must be zeroed.

`ESC` – is an integer variable that specifies the number of exceptional transformations performed up to the current call. On the first call of GJTRID, `ESC` should be initialized to 0.

`CM` – is a double precision real variable that contains the largest condition number encountered up to the current call. On the first call of GJTRID, `CM` should be initialized to 1.0D0.

`CP` – is a double precision real variable that contains the decimal logarithm of the product of condition numbers of transformation matrices used up to the current call. On the first call of GJTRID, `CP` should be initialized to 0.0D0.

`BAD` – is a double precision real variable that contains the modulus of the largest (in modulus) updating factor of a G_2-type transformation, encountered up to the current call. On the first call of GJTRID, `BAD` should be initialized to 0.0D0.

Output parameters

`A` – on normal return from GRICSR and GJTRID, contains the diagonal matrix A of J-tridiagonal form corresponding to the Hamiltonian matrix.

On return from SQRED, contains the matrix A obtained after application of the orthogonal symplectic transformations.

`Q` – contains the diagonal matrix Q of the J-tridiagonal form corresponding to the Hamiltonian matrix. In GRICSR, `Q` should practically be zero.

`R` – the upper triangle contains the Cholesky factor of the matrix R (if $R > 0$). The strict lower triangle is not modified.

NIT – contains the number of iterations performed by the SR algorithm.

X – on normal return from GRICSR, X contains the symmetric solution of the Riccati equation.

On normal return from GJTRID, the lower triangular part contains the updated lower triangle of the matrix X.

IER – is an integer variable set equal to a diagnostic or error completion code.

IER = 0 is the normal completion code.

IER = $-$I, I $\in \{1, 2, \ldots, \text{N}\}$: if the QR algorithm has not converged in 30 iterations when determining the eigenvalues I and N+I of the Hamiltonian matrix.

IER = 1 : if the dimensions are unsuitable (N > min(NA,NB,NQ,NX) or N $\leq$ 0).

IER = 2 : if matrix R is not positive definite.

IER = 3 : if the given matrix Q is numerically zero (diagnostic). A possible solution of Riccati equation (1) is $X = 0$.

IER = 4 : if the Hamiltonian matrix has eigenvalues with zero real parts.

IER = 5 : if the J-tridiagonal form could not be determined.

IER = 6 : if the SR algorithm failed.

IER = 7 : if the SR algorithm has not converged in NIT iterations.

F – contains the symmetric tridiagonal matrix F of the J-tridiagonal form corresponding to the Hamiltonian matrix.

Y – the lower triangular part contains the updated lower triangle of the matrix Y.

SINV – contains the updated matrix S_{11}^{-1}.

REIG, IEIG – are double precision real one-dimensional arrays of dimensions at least N which contain the real and imaginary parts, respectively, of the stable eigenvalues of the Hamiltonian matrix (the poles of the optimal system).

W – if IER $\in \{0, 5, 6, 7\}$ on output from GRICSR, then the first six locations of W contain the following performance information:

W(1) – the number of exceptional transformations used in the initial reduction to J-tridiagonal form;

W(2) – the number of exceptional transformations used in the SR algorithm;

W(3) – the decimal logarithm of the product of condition numbers;

W(4) – the largest condition number encountered;

W(5) – the largest (in modulus) updating factor encountered;

W(6) – the number of SR iterations per eigenvalue (considering N eigenvalues).

G – contains the lower triangle (packed column-wise) of the matrix F obtained after application of orthogonal symplectic transformations.

H – contains the lower triangle (packed column-wise) of the matrix Q obtained after application of orthogonal symplectic transformations.

BS – contains the current D2 value, if an exceptional transformation was performed.

ESC – contains the number of exceptional transformations performed at the reduction.

CM – contains the largest condition number encountered.

CP – contains the decimal logarithm of the product of condition numbers.

BAD – contains the modulus of the largest (in modulus) updating factor.

FAIL – is a logical variable that contains an error information item. FAIL is .FALSE. on normal output, and .TRUE. if one of the transformations involved is unacceptably ill-conditioned.

Called Subprograms

GRICSR: SQRED, HQR1, SEIG, GJTRID, GSRIT, DPOFA, DTRSL, D1MACH, DAXPY, DCOPY, DCSQRT, DDOT, IDAMAX

SQRED : DROTG, H12, SYSIMH

GJTRID: DCOPY, EXCTRA, G1GEN, G2UPD, G3GEN, G3TUP, G3UPD, IDAMAX, PERM, SIMHG1

Applicability and restrictions

GRICSR subroutine may be used only if the Hamiltonian matrix is dichotomic, that is if it has no eigenvalues on the imaginary axis in C. Moreover, it is assumed that matrix R is positive definite. The last condition is required merely to compute the matrix $F \triangleq BR^{-1}B^T$.

The usage of symplectic transformations guarantees that the Hamiltonian structure is preserved during the entire computational process, hence that the eigenvalues appear in pairs $(\lambda, -\lambda)$ and, in case of dichotomy, there are exactly n stable eigenvalues. Due to the rounding errors, the above statements are generally not true if a non-symplectic method is used.

The usage of orthogonal symplectic transformations in SQRED guarantees good numerical properties. The large eigenvalues (in magnitude), computed using the results returned by SQRED, have great accuracy.

For $n \leq 2$, GRICSR determines the solution in zero iterations, by merely performing the separation of the stable part of spectrum of the Hamiltonian matrix. For $n > 2$, the usage of the eigenvalues computed based on Algorithm 3 as (double or quadruple) implicit transformations, has enabled that the convergence of the SR algorithm be achieved in a small number of iterations. In the experiments performed, the average number of iterations per eigenvalue (considering n eigenvalues) was 1. For a single example the value 1.5 was obtained, frequent values being in the range (0.8, 1.2).

GRICSR computes an approximate symmetric solution $\hat{X}$ of the Riccati equation (1). Since Gaussian elementary transformations are used, $\hat{X}$ may be quite inaccurate. Therefore, it is generally advisable to apply the defect correction procedure in (Mehrmann and Tan, 1988). In the examples discussed below and in all other supplementary tests, an iteration of this procedure, based on Newton's algorithm, was performed. Newton's process has always converged (with usual tolerance ε = 5.0D-6) in two — or rarely — three iterations, even if $\hat{X}$ had only the first significant digit correct. Generally, the norm of residual of $\hat{X}$ was $0(\varepsilon_M^{\frac{1}{2}})$. When the defect correction procedure is used, the maximal allowable value of the condition numbers (CB) may be very large. In the tests performed, the number of exceptional transformations was very small (often 0).

GRICSR subroutine should not be used for problems known to be very ill-conditioned, such as Example 6 in (Laub, 1979) for $n \geq 10$, $q = 10^4$. For instance, for this example, with $n = 10$, either the SR algorithm does not converge (if CB $\geq$ 4.0D15), or the maximum allowable number of 20 exceptional transformations per call of GJTRID is exceeded on the first iteration (if CB $\leq$ 1.0D15).

3.2. Examples

Example 1 The following program sequence computes the approximate solution X of the algebraic matrix Riccati equation (1), and the corresponding optimal stabilizing matrix G, using Gaussian symplectic transformations. The system output is weighted with the matrix $\tilde{Q}$ in the performance index, and a proportional-plus-integral optimal control law is designed. It is assumed that the matrices A, B, C, $\tilde{Q}$ and $\hat{Q}$ (where $\hat{Q}$ is the integrator output weighting matrix), of dimensions $n \times n$, $n \times m$, $l \times n$, $l \times l$, and $l \times l$, respectively, as well as R, of dimension $m \times m$, are given. The matrices $\tilde{Q}$ and $\hat{Q}$ are stored in the arrays Q and QI, respectively. The extended matrices are constructed in the arrays A, B, C and Q. The approximate solutions are improved using an iteration of the defect correction procedure with Newton's algorithm. The matrix G is obtained in the array H.

```
      INTEGER I,IER,INFO,J,L,M,N,NA,NB,NC,NI,NIT,NITN,NH,NN,NQ,NQI,NR,NX
C
C     The arrays A, AOLD, Q, QI, R, X and F must have at least N+L,
C       N+L, N+L, L, M, N+L and N+L rows and columns, respectively.
C     The array B must be dimensioned as (N+L,M).
C     The array C must be dimensioned as (max(L+L,M),N+L).
C     The array H must be dimensioned as (M,N+L).
C     The array AOLD will contain a copy of the matrix A, possibly
C       extended for integrators.
C     The working arrays SINV, Y, DQ and DR must have at least
C       (N+L,N+L+1), (N+L,N+L), N+L and M locations, respectively.
C     The working arrays V, W, REIG and IEIG must have at least
C       N+L locations.
C
      DOUBLE PRECISION A(NA,NN),B(NB,M),C(NC,NN),Q(NQ,NN),QI(NQI,L),
     1                 R(NR,M),AOLD(NN,NN),H(NH,NN),X(NX,NN),
     2                 F(NN,NN),Y(NN*NN),SINV(NN,NN+1),V(NN),W(NN),
     3                 DQ(NN),DR(M),REIG(NN),IEIG(NN)
      DOUBLE PRECISION ZERO,ONE
      PARAMETER (ZERO = 0.0D0, ONE = 1.0D0)
      DOUBLE PRECISION DASUM,EPS,QNORM,XNORM,CB
      EQUIVALENCE (QI(1,1),H(1,1))
C
      NITN = NIT
```

```
C     Construct the matrices for proportional-integral control.
      DO 20 J = 1,N
         DO 10 I = 1,L
   10    A(N+I,J) = -C(I,J)
         CALL DCOPY (L,ZERO,0,C(L+1,J),1)
   20 CONTINUE
      DO 30 J = 1,L
         CALL DCOPY (N+L,ZERO,0,A(1,N+J),1)
         CALL DCOPY (L+L,ZERO,0,C(1,N+J),1)
         C(L+J,N+J) = ONE
         CALL DCOPY (L,ZERO,0,Q(L+1,J),1)
         CALL DCOPY (L,ZERO,0,Q(1,L+J),1)
         CALL DCOPY (L,QI(1,J),1,Q(L+1,L+J),1)
   30 CONTINUE
      DO 40 J = 1,M
   40 CALL DCOPY (L,ZERO,0,B(N+1,J),1)
C     Update problem dimensions.
      N = N + L
      L = L + L
C     Compute the state weighting matrix.
      CALL UTAU (L,N,Q,C,Y,NQ,NC)
C     Save A in AOLD and the diagonals of Q and R in DQ and DR.
      DO 50 J = 1,N
   50 CALL DCOPY (N,A(1,J),1,AOLD(1,J),1)
      CALL DCOPY (N,Q,NQ+1,DQ,1)
      CALL DCOPY (M,R,NR+1,DR,1)
C     Solve the Riccati equation.
      CALL GRICSR (N,M,A,B,Q,R,NIT,CB,X,IER,F,Y,SINV,REIG,IEIG,V,W,
     1             NA,NB,NQ,NR,NX)
      IF (IER.EQ.0 .OR. IER.EQ.7) THEN
C        Compute the gain matrix in array C.
         CALL DCOPY (M,DR,1,R,NR+1)
         DO 60 J = 1,M-1
   60    CALL DCOPY (M-J,R(J+1,J),1,R(J,J+1),NR)
         CALL OPTR (N,M,AOLD,B,R,X,0,C,INFO,CB,NN,NB,NR,NX,NC)
         IF (INFO.EQ.0) THEN
C           Construct the packed upper triangle of Q and save X
C             in array A.
            CALL DCOPY (N,DQ,1,Q,NQ+1)
            NI = 1
            DO 70 J = 1,N
               CALL DCOPY (J,Q(1,J),1,Y(NI),1)
               CALL DCOPY (N,X(1,J),1,A(1,J),1)
               NI = NI + J
   70       CONTINUE
C           Initialize the stabilizing matrix with 0.
            DO 80 J = 1,N
   80       CALL DCOPY (M,ZERO,0,H(1,J),1)
            CALL DCOPY (M,DR,1,R,NR+1)
            DO 90 J = 1,M
   90       CALL DCOPY (M-J,R(J+1,J),1,R(J,J+1),NR)
C           Compute the residual of the Riccati equation solution.
            CALL REZRIC (N,M,AOLD,B,Y,C,X,0,SINV,NN,NB,NC,NX)
            QNORM = ZERO
            XNORM = ZERO
            NI = 1
            DO 100 J = 1,N
               QNORM = MAX(QNORM,DASUM(J,Y(NI),1))
               XNORM = MAX(XNORM,DASUM(N,X(1,J),1))
               NI = NI + J
  100       CONTINUE
            QNORM = QNORM/XNORM + ONE
            IF (QNORM.EQ.ONE) GO TO ...
C           Refine the solution of the Riccati equation and compute
C             the optimal stabilizing matrix.
```

```
            CALL NTN (N,M,AOLD,B,Y,R,H,EPS,NITN,X,IER,SINV,F,Q,
     1                NN,NB,NR,NH,NX)
            IF (IER.EQ.0 .OR. IER.EQ.3) THEN
               DO 110 J = 1,N
                  CALL DAXPY (N,ONE,A(1,J),1,X(1,J),1)
                  CALL DAXPY (M,ONE,C(1,J),1,H(1,J),1)
  110          CONTINUE
            ENDIF
         ENDIF
      ENDIF
```

The routine NTN is obtained by modifying NTNC subroutine from the BIMAS library (Varga and Sima, 1985). Versions of UTAU and OPTR subroutines could be found in BIMAS.

Example 2 The program sequence listed in Example 1 was used for the interconnected power system model (Hung and MacFarlane, 1982), $n = 7$, $m = 2$, $l = 3$, without error integrators, and the quadratic performance index defined by the matrices $Q = C^TC$ and $R = 100I_2$. The results obtained are:

$$X = \begin{bmatrix} .5821 & .0945 & .4653 & .0431 & -.1347 & -.0033 & -.0564 \\ .0945 & .0456 & .1717 & -.2203 & -.0033 & -.0048 & -.0181 \\ .4653 & .1717 & .6805 & -.6875 & -.0564 & -.0181 & -.0819 \\ .0431 & -.2203 & -.6875 & 4.3064 & -.0431 & .2203 & .6875 \\ -.1347 & -.0033 & -.0564 & -.0431 & .5821 & .0945 & .4653 \\ -.0033 & -.0048 & -.0181 & .2203 & .0945 & .0456 & .1717 \\ -.0564 & -.0181 & -.0819 & .6875 & .4653 & .1717 & .6805 \end{bmatrix},$$

$$G = \begin{bmatrix} -.0118 & -.0057 & -.0215 & .0275 & .0004 & .0006 & .0023 \\ .0004 & .0006 & .0023 & -.0275 & -.0118 & -.0057 & -.0215 \end{bmatrix}.$$

```
Residual of solution (in 1-norm)  =  3.8980D-011
Norm of residual/norm of solution =  6.2789D-012
```

The solution X of the Riccati equation (1) was obtained with Algorithm 4 after NIT = 8 iterations. The eigenvalues of the optimal (closed-loop) system are: $\{-1.6815, -1.3866 \pm 2.2079\mathrm{j}, -.5623 \pm 3.1319\mathrm{j}, -13.1443, -13.1622\}$. Note that the 1-norm of the residual of X is $0(10^{-10})$. Using then NTN with EPS = 5.0D-6, two iterations were performed; the computed correction had elements with moduli less than 10^{-10}, and the final value of the convergence criterion was 3.9522D-12.

Example 3 Using the program sequence in Example 1 for the chemical reactor model (Hung and MacFarlane, 1982), $n = 4$, $m = 2$, $l = 2$, and the quadratic performance index defined by the matrices $Q = \mathrm{diag}(\tilde{Q}, \hat{Q}) = I_4$, $R = 100I_2$, the results were:

$$X = \begin{bmatrix} 63.3241 & 10.2177 & 45.1674 & -23.8206 & -4.5368 & .9664 \\ 10.2177 & 3.4175 & 7.5223 & -2.8706 & -.4927 & -2.2582 \\ 45.1674 & 7.5223 & 32.2611 & -16.8068 & -3.1083 & .6648 \\ -23.8206 & -2.8706 & -16.8068 & 10.2998 & 3.7302 & 2.0161 \\ -4.5368 & -.4927 & -3.1083 & 3.7302 & 6.4133 & 8.5448 \\ .9664 & -2.2582 & .6648 & 2.0161 & 8.5448 & 22.8617 \end{bmatrix},$$

$$G = \begin{bmatrix} -.8228 & -.2469 & -.6028 & .2369 & .0209 & .0978 \\ 1.4210 & .2367 & 1.0149 & -.5287 & -.0978 & .0209 \end{bmatrix}.$$

```
Residual of solution (in 1-norm)  =  3.6756D-010
Norm of residual/norm of solution =  2.4830D-012
```

Algorithm 4 performed NIT = 8 iterations. The eigenvalues of the optimal (closed-loop) system are: $\{-.0373, -.4346 \pm .3702j, -1.9971, -5.0867, -8.6686\}$. Using then NTN with EPS = 5.0D-6, two iterations were performed; the computed correction had elements with moduli less than 10^{-10}, and the final value of the convergence criterion was 3.4050D-11.

Example 4 This example considers the system defined by $A = \tilde{A}^T$, $B = \tilde{C}^T$, where $\tilde{A}$ and $\tilde{C}$ are the matrices of the chemical reactor model (Hung and MacFarlane, 1982), and the quadratic performance index defined by $Q = I_4$, $R = 10I_2$. The results are listed in (Sima, 1992). Algorithm 4 performed NIT = 4 iterations. The eigenvalues of the optimal (closed-loop) system are: $\{-.1980, -2.0443, -5.0671, -8.6716\}$. Using then NTN with EPS = 5.0D-6, two iterations were performed; the computed correction had elements with moduli less than 10^{-11}, and the final value of the convergence criterion was 0.0D0.

Table 1 summarizes the performance statistics obtained using the program sequence in Example 1 for the optimization problems described above. The item "log(prod. condition nr.)" means the logarithm of the product of condition numbers, and "Iterations/eigenvalues" – number of iterations per eigenvalue.

Table 1: Performance Statistics.

	Ex. 2	Ex. 3	Ex. 4
log(prod. condition nr.)	9.0829D+01	1.3345D+02	3.3742D+01
Largest condition number	1.4876D+03	4.3493D+04	1.9210D+02
Largest updating factor	7.0708D+02	2.1205D+04	2.8921D+02
Iterations/eigenvalues	1.1429D+00	1.3333D+00	1.0000D+00

No exceptional transformations have been performed neither in the initial reduction, nor in the iterative process. Detailed results and conclusions on the numerical experiments are reported in (Sima, 1992).

3.3. Implementation Details

After initializing the performance parameters, GRICSR subroutine constructs the lower triangle of the matrix $F \triangleq BR^{-1}B^T$ and computes the tolerance ε, used in the negligibility tests ($\varepsilon = \varepsilon_u \parallel H \parallel$, where ε_u is the rounding unit of the computer machine). Then GRICSR prepares the call of SQRED routine, taking into account that SQRED changes the contents of the input arrays. To reduce the memory requirements, matrix A is saved in the array W and in the strict upper triangular parts of the arrays F and Y. Moreover, the columns of the lower triangle of matrix F are concatenated in the array SINV, and then, similarly, those of matrix Q. For this reason, GRICSR treats SINV as a one-dimensional array, of dimension N × (N+1).

The results delivered by SQRED in the arrays A and SINV are used to compute the eigenvalues of the Hamiltonian matrix. To simplify processing, and since

memory space is available (for later calculations), the matrices F and Q, obtained in SINV, are restructured to the two-dimensional storage; F is entirely stored in the array X, and the lower triangle of Q is stored into the corresponding portion of Y. The matrix $V \triangleq A^2 + FQ$, in HF, is computed and $\lambda(V)$ is then determined. Since $\lambda(H^2) = \lambda(V)$, it follows that $\lambda_i(H) = -(\lambda_i(V))^{\frac{1}{2}}$ and $\lambda_{n+i}(H) = -\lambda_i(H)$, $i = 1 : n$. The components of indices $n + i$ are not stored. The eigenvalues with negative real parts, $\lambda_i(H)$, $i = 1 : n$ (representing the optimal system poles), are ordered in the increasing order of the moduli of real parts, and in case of equality, in the increasing order of the moduli of imaginary parts.

Next, GRICSR reconstructs the matrix A from the information stored in W and in the strict upper triangles of F and Y, and initializes the array SINV to I, and the lower triangles of the arrays X and Y to 0. The GJTRID and GSRIT routines, which implement the initial reduction to J-tridiagonal form and the SR algorithm, respectively, are then called; at the same time, the stable eigenvalues of H are moved by GSRIT into the first n positions.

Finally, the upper triangle of the solution matrix is constructed, and the information regarding the performances is stored in W.

SQRED subroutine implements the implicit variant of Van Loan's algorithm (Van Loan, 1984). Orthogonal transformations — Householder transformations U, computed by H12 routine, and standard plane rotations V, computed by DROTG routine (Lawson and co-workers, 1979) — are exclusively used. Only matrix A and the lower triangles (packed column-wise) of the matrices F and Q are stored in SQRED. The routine has a loop $k = 1 : n - 1$. The elements $k + 1 : n$ of the vector $Q(Ae_k) - A^T(Qe_k)$ are first computed in the working array W and the Householder transformation U, that modifies the $(k+1)$-th element and annihilates the elements $k + 2 : n$, is determined. The transformation U is applied to A, in the form $A \leftarrow UAU$, using two calls to H12 routine (the first call determining U at the same time), and then U is similarly applied to Q and F, by calling SYSIMH subroutine. (To simplify the code, SYSIMH reconverts the packed storage to the two-dimensional storage in the working array X.) Next, SQRED computes $a_1 \triangleq (A(Ae_k) + F(Qe_k))_{k+1}$ and $a_2 \triangleq (Q(Ae_k) - A^T(Qe_k))_{k+1}$, determines the rotation V that modifies a_1 and zeroes a_2 and applies V to A ($A \leftarrow VAV^T$) and, similarly, to F and Q. To accomplish these operations, it is necessary to save in advance the $(k + 1)$-th row of F in the array W and the element $a_{k+1,k+1}$ in a local variable. Finally, a Householder transformation U, that modifies the $(k + 1)$-th element and annihilates the elements $k + 2 : n$ of the vector $A(Ae_k) + F(Qe_k)$ (computed in W), is determined and applied as above.

GJTRID subroutine implements Algorithm 1. But facilities for working only with the submatrices (D1:D2,D1:D2) in A, F and Q, and for annihilating at most BS sub- and superdiagonal elements in each row and column of these matrices are included. BS has the value D2 at the initial reduction to the J-tridiagonal form, or when some exceptional transformations are used, but during the SR algorithm, BS can have the values 1 or 2. (Initially, D1 = 1, D2 = N.) The calculation and application of the symplectic matrices G_1, G_2, P and G_3 is made by calling the subroutines G1GEN, SIMHG1, G2UPD, PERM, G3GEN, G3TUP and G3UPD, and the routines called by them. Mind that no special routine is included for the calculation of a G_2-type transformation, since this is simply obtained by calling

G1GEN, but interchanging beforehand the values of its first two arguments. Both G1GEN and G3GEN estimate the condition number of the transformation matrix, test its size and accumulate the decimal logarithm of the product of condition numbers of all transformations performed up to the current call. (The logarithm is used since the evaluation of the product could produce overflows.) If any of the transformations involved in the iterative process is too ill conditioned, then GJTRID resorts to an exceptional transformation (actually representing the product of a random G_2-type transformation and a random G_3-type transformation), by calling the routine EXCTRA. At most 20 exceptional transformations are allowed on one single call of GJTRID. It must also be emphasized here that the routines can work with (`D1:D2,D1:D2`) submatrices and use exclusively the lower triangles of the arrays `Q`, `F`, `X` and `Y`. However, the indices vary from 1 to `N` when the matrices X, Y and S_{11}^{-1} are updated.

GSRIT subroutine implements the iterative SR algorithm. At the beginning of each iteration, GSRIT tests if there is a negligible subdiagonal element in F, and, if so, the problem is split into two subproblems. Once a submatrix N_j or order two or four has been detected, GSRIT applies the transformations needed to move the stable eigenvalues in the upper-left corner, by calling the subroutines DEF2X2 or DEF4X4, respectively. (This is equivalent to finding one vector or two vectors, respectively, of a basis for the stable H-invariant subspace.) DEF4X4 also deals with the case when $\lambda(N_j) = \{\lambda, \mu, -\lambda, -\mu\}$, $\lambda, \mu \in \mathrm{R}$. Clearly, if the convergence has not been obtained for any submatrix N_j at the current iteration in GSRIT , then the iterative process continues by using a double or quadruple transformation, calling the subroutine GSR2 or GSR4, respectively. The "exact" eigenvalues, computed by GRICSR after the call of SQRED, are used. GSR2 and GSR4 apply an SR step; the nonzero elements of the first column of matrix $p_k(H_k)$ are computed, then the symplectic transformation reducing this column to e_1 is determined, and finally, the matrix resulted from applying this transformation is reduced to J-tridiagonal form.

4. CONCLUSIONS

An efficient algorithm for solving CAREs using Gaussian symplectic transformations has been discussed, and numerical results obtained based on an implementation of this algorithm have been described. The underlying method exploits the special spectral properties of the Hamiltonian matrix associated with the optimal problem, and delivers a symmetric solution. The method works well in practice, except for very ill-conditioned CAREs, for which other methods also encounter difficulties. The numerical experience has shown that large condition numbers of the transformation matrices are tolerable, particularly when a defect correction procedure is used for increasing the accuracy of the computed solution.

Acknowledgment

The author is grateful to Professor dr. Volker Mehrmann from Bielefeld University, Germany, for kindly offering a series of papers. Part of our codes are modifications of some preliminary versions developed by dr. Mehrmann.

REFERENCES

[1] BUNSE-GERSTNER, A., MEHRMANN, V., **A symplectic QR like algorithm for the solution of the real algebraic Riccati equation.** IEEE TRANS. AUTOMAT. CONTROL AC-31 (1986), pp. 1104–1113.

[2] BUNSE-GERSTNER, A., MEHRMANN, V., WATKINS, D., **An SR algorithm for Hamiltonian matrices based on Gaussian elimination.** METHODS OF OPER. RES., Athenäum Verlag (1989).

[3] BYERS, R., MEHRMANN, V., **Symmetric updating of the solution of the algebraic Riccati equation.** METHODS OF OPER. RES. 54, Proc. of X Symp. on Oper. Res., Anton Hain, Königstein (1985), pp. 117–125.

[4] HUNG, Y.S., MACFARLANE, A.G.J., **Multivariable Feedback: A Quasi-Classical Approach.** Lect. Notes in Control and Information Sciences (40), SPRINGER VERLAG, Berlin (1982).

[5] LAUB, A.J., **A Schur method for solving algebraic Riccati equations.** IEEE TRANS. AUTOMAT. CONTROL AC-24 (1979), pp. 913–921.

[6] LAWSON, C., HANSON, R., KINKAID, D., KROGH, F., **Basic linear algebra subprograms for Fortran usage.** ACM TRANS. MATH. SOFTW. 5 (1979), pp. 303–323.

[7] MEHRMANN, V., TAN, E., **Defect correction methods for the solution of the algebraic Riccati equations.** IEEE TRANS. AUTOMAT. CONTROL AC-33 (1988), pp. 695–698.

[8] SIMA V., **Computational experience in solving the algebraic Riccati equations.** STUDIES AND RESEARCHES IN COMPUTERS AND INFORMATICS 2 (1990), pp. 77–96.

[9] SIMA V., **On Hamiltonian-symplectic algorithms for solving algebraic Riccati equations.** Proc. of the First European Control Conference ECC'91, Grenoble, France, July 2–5 1991, EDITION HERMES, vol. 3 (1991), pp. 2202–2207.

[10] SIMA, V., **Computational experience in solving algebraic matrix Riccati equations using Gaussian symplectic transformations.** Submitted for The 12-th IFAC World Congress, 19th – 23rd July 1993, Sydney, Australia (1992).

[11] VAN LOAN, C.F., **A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix.** LIN. ALG. APPL. 61 (1984), pp. 233–251.

[12] VARGA, A., SIMA, V., **BIMAS — A basic mathematical package for computer-aided systems analysis and design.** In J. Gertler, L. Keviczky (Eds.), *A Bridge between Control Science and Technology*, PERGAMON PRESS, Oxford, vol. 1 (1985).