October, 1978

LIDS-R-859

Research Supported By:
Contract ERDA-E(49-18)-2087

A SCHUR METHOD FOR SOLVING ALGEBRAIC RICCATI EQUATIONS

Alan J. Laub

Laboratory for Information and Decision Systems
Formerly
Electronic Systems Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MASSACHUSETTS 02139

October, 1978 LIDS-R-859

A SCHUR METHOD FOR SOLVING ALGEBRAIC RICCATI EQUATIONS*

by

Alan J. Laub**

ABSTRACT

In this paper a new algorithm for solving algebraic Riccati equations (both continuous-time and discrete-time versions) is presented. The method studied is a variant of the classical eigenvector approach and uses instead an appropriate set of Schur vectors thereby gaining substantial numerical advantages. Complete proofs of the Schur approach are given as well as considerable discussion of numerical issues. The method is apparently quite numerically stable and performs reliably on systems with dense matrices up to order 100 or so, storage being the main limiting factor.

*This research was supported by the U.S. Energy Research and Development Agency under contract ERDA-E(49-18)-2087.

**Laboratory for Information and Decision Systems, Rm. 35-331, Massachusetts Institute of Technology, Cambridge, MA 02139; ph.: (617)-253-2125.

1. Introduction

In this paper a new algorithm for solving algebraic Riccati equations (both continuous-time and discrete-time versions) is presented. These equations play fundamental roles in the analysis, synthesis, and design of linear-quadratic-Gaussian control and estimation systems as well as in many other branches of applied mathematics. It is not the purpose of this paper to survey the extensive literature available for these equations but, rather, we refer the reader to, for example, [1], [2], [3], [4], and [5] for references. Nor is it our intention to investigate any but the unique (under suitable hypotheses) symmetric, nonnegative definite solution of an algebraic Riccati equation even though the algorithm to be presented does also have the potential to produce other solutions. For further reference to the "geometry" of the Riccati equation we refer to [3], [6], and [7].

The method studied here is a variant of the classical eigenvector approach to Riccati equations, the essentials of which date back to at least von Escherich in 1898 [8]. The approach has also found its way into the control literature in papers by, for example, MacFarlane [9], Potter [10], and Vaughn [11]. Its use in that literature is often associated with the name of Potter. However, the use of eigenvectors is often highly unsatisfactory from a numerical point of view and the present method uses the so-called and much more numerically attractive Schur vectors to get a basis for a certain subspace of interest in the problem.

Other authors such as Fath [12] and Willems [3], to name two, have also noted that any basis of the subspace would suffice but the specific use of Schur vectors was inhibited by a not-entirely-straightforward problem of ordering triangular canonical forms - a problem which is discussed at length in the sequel. The paper by Fath is very much in the spirit of the work presented here and is one of the very few in the literature which seriously addresses numerical issues.

One of the best summaries of the eigenvector approach to solving algebraic Riccati equations is the work of Martensson [13]. This work extends [10] to the case of "multiple closed-loop eigenvalues". It will be shown in the sequel how the present approach recovers all the theoretical results of [10] and [13] while providing significant numerical advantages.

Most numerical comparisons of Riccati algorithms tend to definitely favor the standard eigenvector approach - its numerical difficulties notwithstanding - over other approaches such as Newton's method [14] or methods based on integrating a Riccati differential equation. Typical of such comparisons are [7], [15], and [16]. It will be demonstrated in this paper that if you previously liked the eigenvector approach, you will like the Schur vector approach at least twice as much. This statement, while somewhat simplistic, is based on the fact that a Schur vector approach provides a substantially more efficient, useful, and reliable technique for numerically solving algebraic Riccati equations. The method is intended primarily for the solution of dense, moderate-sized equations (say, order ≤ 100) rather than large, sparse equations. While the algorithm in its present state offers much scope for improvement, it still represents an order-of-magnitude improvement over current methods for solving algebraic Riccati equations.

Briefly, the rest of the paper is organized as follows. This section is concluded with some notation and linear algebra review. In Sections 2 and 3 the continuous-time and discrete-time Riccati equations, respectively, are treated. In Section 4 numerical issues such as algorithm implementation, balancing, scaling, operation counts, timing, storage, stability, and conditioning are considered. In Section 5 we emphasize the advantages of the Schur vector approach and make some further general remarks. Six examples are given in Section 6 and some concluding remarks are made in Section 7.

1.1 Notation

Throughout the paper $A \in \mathbb{F}^{m\mathrm{x}n}$ will denote an mxn matrix with coefficients in a field $\mathbb{F}$. The field will usually be the real numbers $\mathbb{R}$ or the complex numbers $\mathbb{C}$. The notations A^T and A^H will denote transpose and conjugate transpose, respectively, while A^{-T} will denote $(A^T)^{-1} = (A^{-1})^T$. The notation A^+ will denote the Moore-Penrose pseudoinverse of the matrix A. For $A \in \mathbb{R}^{n\mathrm{x}n}$ its spectrum (set of n eigenvalues) will be denoted by $\sigma(A)$. When a matrix $A \in \mathbb{R}^{2n\mathrm{x}2n}$ is partitioned into four nxn blocks as

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

we shall frequently refer to the individual blocks A_{ij} without further discussion.

1.2 Linear Algebra Review

Definition 1: $A \in \mathbb{R}^{n\times n}$ is orthogonal if $A^T = A^{-1}$.

Definition 2: $A \in \mathbb{C}^{n\times n}$ is unitary if $A^H = A^{-1}$.

Let $J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \in \mathbb{R}^{2n\times 2n}$ where I denotes the n^{th} order identity matrix.
Note that $J^T = J^{-1} = -J$.

Definition 3: $A \in \mathbb{R}^{2n\times 2n}$ is Hamiltonian if $J^{-1}A^TJ = -A$.

Definition 4: $A \in \mathbb{R}^{2n\times 2n}$ is symplectic if $J^{-1}A^TJ = A^{-1}$.

Hamiltonian and symplectic matrices are obviously closely related. For a discussion of this relationship and a review of "symplectic algebra" see [17], [18]. We will use the following two theorems from symplectic algebra. Their proofs (see [18]) are trivial (and hence will be omitted).

Theorem 1: 1. Let $A \in \mathbb{R}^{2n\times 2n}$ be Hamiltonian. Then $\lambda \in \sigma(A)$ implies $-\lambda \in \sigma(A)$ with the same multiplicity. 2. Let $A \in \mathbb{R}^{2n\times 2n}$ be symplectic. Then $\lambda \in \sigma(A)$ implies $\frac{1}{\lambda} \in \sigma(A)$ with the same multiplicity.

There is a relationship between the right and left eigenvectors of these symplectically associated eigenvalues. See [18] for details.

Theorem 2: Let $A \in \mathbb{R}^{2n\times 2n}$ be Hamiltonian (or symplectic). Let $U \in \mathbb{R}^{2n\times 2n}$ be symplectic. Then $U^{-1}AU$ is Hamiltonian (or symplectic).

Finally, we need two theorems from classical similarity theory which form the theoretical cornerstone of modern numerical linear algebra. See [19], for example, for a textbook treatment.

Theorem 3 (Schur canonical form): Let $A \in \mathbb{R}^{n\times n}$ have eigenvalues $\lambda_1,\ldots,\lambda_n$. Then there exists a unitary similarity transformation U such that $U^H AU$ is upper triangular with diagonal elements $\lambda_1,\ldots,\lambda_n$ in that order.

In fact, it is possible to work only over $\mathbb{R}$ by reducing to quasi-upper-triangular form with 2x2 blocks on the (block) diagonal corresponding to complex conjugate eigenvalues and 1x1 blocks corresponding to the real eigenvalues. We refer to this canonical form as the real Schur form (RSF) or the Murnaghan-Wintner [20] canonical form.

Theorem 4 (RSF): Let $A \in \mathbb{R}^{n\times n}$. Then there exists an orthogonal similarity transformation U such that $U^T AU$ is quasi-upper-triangular. Moreover, U can be chosen so that the 2x2 and 1x1 diagonal blocks appear in any desired order.

If in Theorem 4 we partition $U^T AU$ into $\begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix}$ where $S_{11} \in \mathbb{R}^{k\times k}$, $0<k\leq n$, we shall refer to the first k vectors of U as the Schur vectors corresponding to $\sigma(S_{11}) \subseteq \sigma(A)$. The Schur vectors corresponding to the eigenvalues of S_{11} span the eigenspace corresponding to those eigenvalues even when some of the eigenvalues are multiple (see [21]). We shall use this property heavily in the sequel.

2. The Continuous-Time Algebraic Riccati Equation

In this section we shall present a method for using a certain set of Schur vectors to solve (for X) the continuous-time algebraic Riccati equation

$$F^TX + XF - XGX + H = 0 . \tag{1}$$

All matrices are in $\mathbb{R}^{nxn}$ and $G = G^T \geq 0$, $H = H^T \geq 0$.

It is assumed that (F,B) is a stabilizable pair [1] where B is a full-rank factorization (FRF) of G(i.e., $BB^T = G$ and rank(B) = rank(G)) and (C,F) is a detectable pair [1] where C is a FRF of H (i.e., $C^TC = H$ and rank(C) = rank(H)). Under these assumptions, (1) is known to have a unique nonnegative definite solution [1]. There are, of course, many other solutions to (1) but for the algorithm presented here the emphasis will be on computing the nonnegative definite one.

Now consider the Hamiltonian matrix

$$Z = \begin{pmatrix} F & -G \\ -H & -F^T \end{pmatrix} \in \mathbb{R}^{2nx2n} \tag{2}$$

Our assumptions guarantee that Z has no pure imaginary eigenvalues. Thus by Theorem 4 we can find an orthogonal transformation $U \in \mathbb{R}^{2nx2n}$ which puts Z in RSF:

$$U^TZU = S = \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix} \tag{3}$$

where $S_{ij} \in \mathbb{R}^{nxn}$. It is possible to arrange, moreover, that the real parts of the spectrum of S_{11} are negative while the real parts of the spectrum of S_{22} are positive. U is conformably partitioned into four nxn blocks:

$$U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \tag{4}$$

We then have the following theorem.

Theorem 5: With respect to the notation and assumptions above:

1. U_{11} is invertible and $X = U_{21}U_{11}^{-1}$ solves (1).

2. $\sigma(S_{11}) = \sigma(F - GX)$ = the "closed-loop" spectrum.

3. $X = X^T$.

4. $X \geq 0$.

Proof:

1. We first prove that U_{11} is invertible. To avoid complicating the proof unnecessarily by having to consider 2x2 blocks of S_{11}, we will for simplicity assume that $S \in \mathbb{C}^{2nx2n}$ is upper triangular and U is unitary. Suppose $U_{11} \in \mathbb{C}^{nxn}$ is singular. Without any loss of generality, we may assume that U_{11} is of the form $(0, \hat{U}_{11})$ where $\hat{U}_{11} \in \mathbb{C}^{nx(n-1)}$. Thus, we have

$$\begin{pmatrix} F & -G \\ -H & -F^T \end{pmatrix} \begin{pmatrix} 0 \\ u \end{pmatrix} = \begin{pmatrix} 0 \\ u \end{pmatrix} \cdot (-\lambda) \tag{5}$$

where $u \in \mathbb{C}^{nx1}$ and $(-\lambda)$ with $\mathrm{Re}\lambda > 0$ is the upper left element of S. But then for any X we have

$$\begin{aligned} (F-GX)^T u &= F^T u - X^T G u \\ &= \lambda u \qquad \text{by (5)}. \end{aligned}$$

However, we also have $F^T u = \lambda u$ by (5). Thus we have an eigenvalue λ of F with positive real part which is uncontrollable. This contradicts the assumption of stabilizability so U_{11} must be invertible.

We now show that $X = U_{21}U_{11}^{-1}$ solves (1). Simply substitute into (1):

$$
\begin{aligned}
F^T X + XF - XGX + H &\equiv -(I,X)JZ\begin{pmatrix} I \\ X \end{pmatrix} \\
&= (U_{21}U_{11}^{-1}, -I)\; Z\begin{pmatrix} I \\ U_{21}U_{11}^{-1} \end{pmatrix} \\
&= (U_{21}U_{11}^{-1}, -I)\; Z\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} U_{11}^{-1} \\
&= (U_{21}U_{11}^{-1}, -I)\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} S_{11}U_{11}^{-1} \qquad \text{from (3)} \\
&= 0 \; .
\end{aligned}
$$

2. From $\begin{pmatrix} F & -G \\ -H & -F^T \end{pmatrix}\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} = \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} S_{11}$

we have

$$
\begin{aligned}
U_{11}S_{11} &= FU_{11} - GU_{21} \\
&= (F - GX)U_{11} \; .
\end{aligned}
$$

Thus $U_{11}^{-1}(F - GX)U_{11} = S_{11}$ so $\sigma(S_{11}) = \sigma(F - GX)$.

3. Let $Y = U_{11}^T U_{21}$. (6)

Then

$$X = U_{11}^{-T} Y U_{11}^{-1} \tag{7}$$

so to prove that X is symmetric it clearly suffices to show that Y is symmetric, i.e., $U_{11}^T U_{21} - U_{21}^T U_{11} = 0$.

Now consider the skew-symmetric, orthogonal matrix $M = U^T J U$. Using the fact that Z is Hamiltonian, it is easy to show that

$$S^T M = -MS$$

where S was given in (3). Thus $S_{11}^T M_{11} + M_{11} S_{11} = 0$. But since S_{11} is stable, it follows from classical Lyapunov theory (see, e.g., [22]) that $M_{11} = 0$. But $M_{11} = U_{11}^T U_{21} - U_{21}^T U_{11}$ so $U_{11}^T U_{21} = U_{21}^T U_{11}$.

Remark: It can be shown that the matrix M is of the general form

$$M = \begin{pmatrix} 0 & M_{12} \\ -M_{12}^T & 0 \end{pmatrix} \text{ where } M_{12} \text{ is orthogonal.}$$

4. From (6) and (7) it clearly suffices to prove that $U_{11}^T U_{21} \geq 0$. Define

$$V(t) = \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} e^{tS_{11}} .$$

Note that $V(0) = \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}$ while $\lim_{t \to +\infty} V(t) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ since S_{11} is stable. Then

$$\dot{V}(t) = \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} S_{11} e^{tS_{11}}$$

$$= Z \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} e^{tS_{11}} \qquad \text{by (3)}$$

$$= ZV(t) .$$

Now let $W(t) = V^T(0)LV(0) - V^T(t)LV(t)$ where $L = \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix}$. Then

$$W(t) = -\int_0^t \frac{d}{ds}[V^T(s)LV(s)]ds$$

$$= -\int_0^t V^T(s)[Z^TL + LZ]V(s)ds$$

$$= -\int_0^t V^T(s)\begin{bmatrix} -H & 0 \\ 0 & -G \end{bmatrix} V(s)ds$$

$$\geq 0 \text{ for all } t \geq 0 .$$

Thus $\lim_{t\to +\infty} W(t) = V^T(0)LV(0) = U_{11}^T U_{21} \geq 0$.

This completes the proof of the theorem. □

Further discussion of this theorem and computational considerations are deferred until Section 4.

3. The Discrete-Time Algebraic Riccati Equation

In this section we shall present an analogous method using certain Schur vectors to solve the discrete-time algebraic Riccati equation

$$F^T X F - X - F^T X G_1 (G_2 + G_1^T X G_1)^{-1} G_1^T X F + H = 0 \quad . \qquad (8)^*$$

Here $F, H, X \in \mathbb{R}^{n \times n}$, $G_1 \in \mathbb{R}^{n \times m}$, $G_2 \in \mathbb{R}^{m \times m}$, and $H = H^T \geq 0$, $G_2 = G_2^T > 0$. Also, $m \leq n$. The details of the method for this equation are sufficiently different from the continuous-time case that we shall explicitly present most of them.

It is assumed that (F, G_1) is a stabilizable pair and that (C, F) is a detectable pair where C is a FRF of H (i.e., $C^T C = H$ and $\text{rank}(C) = \text{rank}(H)$). We also assume that F is invertible - a common assumption on the open-loop dynamics of a discrete-time system [23]. The details for the case when F is singular can be found in Appendix 1.

Under the above assumptions (8) is known to have a unique nonnegative definite solution [23] and the method proposed below will be directed towards finding that solution.

Setting $G = G_1 G_2^{-1} G_1^T$ we consider this time the symplectic matrix

$$Z = \begin{pmatrix} F + GF^{-T}H & -GF^{-T} \\ -F^{-T}H & F^{-T} \end{pmatrix} \qquad (9)$$

Our assumptions guarantee that Z has no eigenvalues on the unit circle. By Theorem 4 we can find an orthogonal transformation $U \in \mathbb{R}^{2n \times 2n}$ which puts Z in RSF:

---

*Note that an alternate equivalent form of (8) when X is invertible is: $F^T (X^{-1} + G_1 G_2^{-1} G_1^T)^{-1} F - X + H = 0$

$$U^T Z U = S = \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix} \tag{10}$$

where $S_{ij} \in \mathbb{R}^{nxn}$.

It is possible to arrange, moreover, that the spectrum of S_{11} lies inside the unit circle while the spectrum of S_{22} lies outside the unit circle. Again U is partitioned conformably. We then have the following theorem.

Theorem 6: With respect to the notation and assumptions above:

1. U_{11} is invertible and $X = U_{21}U_{11}^{-1}$ solves (8).

2. $\sigma(S_{11}) = \sigma(F - G_1(G_2+G_1^T X G_1)^{-1} G_1^T XF)$

   $= \sigma(F - GF^{-T}(X-H))$

   $= \sigma(F - G(X^{-1} + G)^{-1}F)$ when X is invertible

   = the "closed-loop" spectrum.

3. $X = X^T$.

4. $X \geq 0$.

Proof:

1. We proceed as in the proof of Theorem 5. Again we assume that U_{11} is singular and of the form $U_{11} = (0, \hat{U}_{11})$ where $\hat{U}_{11} \in \mathbb{C}^{nx(n-1)}$. Then since $U^T Z^{-1} U = S^{-1}$ we have

$$\begin{pmatrix} F^{-1} & F^{-1}G \\ HF^{-1} & F^T + HF^{-1}G \end{pmatrix} \begin{pmatrix} 0 \\ u \end{pmatrix} = \begin{pmatrix} 0 \\ u \end{pmatrix} \lambda \tag{11}$$

where $u \in \mathbb{C}^{nx1}$ and $|\lambda| > 1$. But then for any X we have

$$(F - GF^{-T}(X-H))^T u = (F^T + HF^{-1}G)u - X^T F^{-1} Gu$$

$$= \lambda u$$

by (11). However, we also have $F^T u = \lambda u$ by (11). Thus we have $\lambda \in \sigma(F)$ with $|\lambda| > 1$ which is uncontrollable. This contradicts the assumption of stabilizability so U_{11} must be invertible. To show that $X = U_{21}U_{11}^{-1}$ solves (8) we have:

$$F^T XF - X - F^T XG_1(G_2+G_1^T XG_1)^{-1}G_1^T XF + H$$

$$\equiv F^T XF - X - F^T XGF^{-T}(X - H) + H$$

$$\equiv -F^T(I,X)JZ\begin{pmatrix} I \\ X \end{pmatrix}$$

$$= -F^T(-U_{21}U_{11}^{-1}, I)\ Z\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} U_{11}^{-1}$$

$$= -F^T(-U_{21}U_{11}^{-1}, I)\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} S_{11}U_{11}^{-1} \qquad \text{from (10)}$$

$$= 0 .$$

<u>2.</u> From $\begin{pmatrix} F + GF^{-T}H & -GF^{-T} \\ -F^{-T}H & F^{-T} \end{pmatrix} \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} = \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} S_{11}$

we have $U_{11}S_{11} = (F + GF^{-T}H)U_{11} - GF^{-T}U_{21}$

$$= (F - GF^{-T}(X - H))U_{11} .$$

Thus $\sigma(S_{11}) = \sigma(F - GF^{-T}(X-H))$. The other equalities follow by well-known matrix identities.

3. Let $Y = U_{11}^T U_{21}$. Since $X = U_{11}^{-T} Y U_{11}^{-1}$ it suffices, as in Theorem 5, to prove that Y is symmetric. The proof is essentially the same: since Z is symplectic we have

$$S^T M = -MS^{-1}$$

where $M = U^T JU$ and S was given in (10). Then $S_{11}^T M_{11} S_{11} + M_{11} = 0$ whence $M_{11} = 0$ by classical Lyapunov theory. But $M_{11} = U_{11}^T U_{21} - U_{21}^T U_{11}$ so symmetry follows.

4. As in Theorem 5 it suffices to prove that $U_{11}^T U_{21} \geq 0$. Define $V(k) = \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} S_{11}^k$. Note that $V(0) = \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}$ while $\lim_{k \to +\infty} V(k) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ since S_{11} is stable. Then

$$\begin{aligned} V(k+1) &= \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} S_{11}^{k+1} \\ &= ZV(k) \end{aligned}$$

by (10). Now let $W(k) = V^T(0)LV(0) - V^T(k)LV(k)$ where $L = \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix}$. Then

$$\begin{aligned} W(k) &= \sum_{j=0}^{k-1} [V^T(j)LV(j) - V^T(j+1)LV(j+1)] \\ &= \sum_{j=0}^{k-1} V^T(j)[L - Z^T LZ]V(j) \\ &= \sum_{j=0}^{k-1} V^T(j) \begin{bmatrix} H + HF^{-1}GF^{-T}H & -HF^{-1}GF^{-T} \\ -F^{-1}GF^{-T}H & F^{-1}GF^{-T} \end{bmatrix} V(j) \end{aligned}$$

Now, according to a theorem of Albert [24], a matrix

$$A = \begin{pmatrix} A_{11} & A_{12}^T \\ A_{12} & A_{22} \end{pmatrix}$$

with $A_{11} = A_{11}^T \in \mathbb{R}^{nxn}$, $A_{22} = A_{22}^T \in \mathbb{R}^{mxm}$ is nonnegative definite if and only if:

(i) $A_{22} \geq 0$

(ii) $A_{22}A_{22}^+A_{12} = A_{12}$

and

(iii) $A_{11} - A_{12}^T A_{22}^+ A_{12} \geq 0.$

For the matrix $A = \begin{pmatrix} H + HEH & -HE \\ -EH & E \end{pmatrix}$ where $E = F^{-1}GF^{-T}$ we clearly have (i) satisfied. We also have (ii) satisfied since $EE^+(-EH) = -EH$ by an elementary defining property of the Moore-Penrose pseudoinverse [25]. Finally, to verify (iii) we note that

$$H + HEH - (-HE)E^+(-EH) = H \geq 0 .$$

Thus $W(k) \geq 0$ for all $k \geq 0$ so

$$\lim_{k \to +\infty} W(k) = V^T(0)LV(0) = U_{11}^T U_{21} \geq 0 .$$

This completes the proof of the theorem. □

We now turn to some general numerical considerations regarding the Schur vector approach.

4. Numerical Considerations

There are two steps to the Schur vector approach. The first is reduction of a $2n \times 2n$ matrix to an ordered real Schur form; the second is the solution of an n^{th} order linear matrix equation. We shall discuss these in the context of the continuous-time case noting differences for the discrete-time case where appropriate.

4.1 Algorithm Implementation

It is well-known (see [21], for example) that the double Francis QR algorithm applied to a real general matrix does not guarantee any special order for the eigenvalues on the diagonal of the Schur form. However, it is also known how the real Schur form can be arbitrarily reordered via orthogonal similarities; see [21] for details. Thus any further orthogonal similarities required to ensure that $\sigma(S_{11})$ in (3) lies in the left-half complex plane can be combined with the U initially used to get a RSF to get a final orthogonal matrix which effects the desired ordered RSF.

Stewart has recently published FORTRAN subroutines for calculating and ordering the RSF of a real upper Hessenberg matrix [26]. The 1×1 or 2×2 blocks are ordered so that the eigenvalues appear in descending order of magnitude along the diagonal. Stewart's software (HQR3) may thus be used directly if one is willing to first apply to the Z of (2) an appropriate bilinear transformation which maps the left-half-plane to the exterior of the unit circle. Since the transformed Z is an analytic function of Z, the U that reduces it to an ordered RSF - with half the eigenvalues outside the unit circle - is the desired U from which the solution of (1) may be constructed. Alternatively, Stewart's software can be modified to directly reorder a RSF by algebraic sign.

In the discrete-time case, HQR3 can be used directly by working with

$$Z^{-1} = \begin{pmatrix} F^{-1} & F^{-1}G \\ HF^{-1} & F^T + HF^{-1}G \end{pmatrix} .$$

The U which puts $\sigma(S_{11})$ outside the unit circle is thus the same U which puts the upper left nxn block of the RSF of Z inside the unit circle.

In summary then, to use HQR3 we would recommend using the following sequence of subroutines (or their equivalents):

BALANC	to balance a real general matrix
ORTHES	to reduce the balanced matrix to upper Hessenberg form using orthogonal transformations
ORTRAN	to accumulate the transformations from the Hessenberg reduction
HQR3	to determine an ordered RSF from the Hessenberg matrix
BALBAK	to backtransform the orthogonal matrix to a non-singular matrix corresponding to the original matrix.

The subroutines BALANC, ORTHES, ORTRAN, BALBAK are all available in EISPACK [27].

The second step to be implemented is the solution of an $n^{\underline{th}}$ order linear matrix equation

$$XU_{11} = U_{21}$$

to find $X = U_{21}U_{11}^{-1}$. For this step we would recommend a good linear equation solver such as DECOMP and SOLVE available in [28] or the appropriate routines available in the forthcoming LINPACK [29]. A routine such as DECOMP computes the LU-factorization of U_{11} and SOLVE performs the forward and backward substitutions. A good estimate of the condition number of U_{11} with respect to inversion is available with good linear equation software and this estimate should be inspected. A badly conditioned U_{11} usually results from a "badly conditioned Riccati equation". This matter will be discussed further in Section 4.4. While we have no analytical proof at this time, we have observed empirically that a condition number estimate on the order of 10^t for U_{11} usually results in a loss of about t digits of accuracy in X.

One final note on implementation. Since X is symmetric it is usually more convenient, with standard linear equation software, to solve the equation

$$U_{11}^T X = U_{21}^T$$

to find $X = U_{11}^{-T} U_{21}^T = U_{21} U_{11}^{-1}$.

4.2 Balancing and Scaling

Note that the use of balancing in the above implementation results in a nonsingular (but not necessarily orthogonal) matrix which reduces Z to RSF. More specifically, suppose P is a permutation matrix and D is a diagonal matrix such that PD balances Z, i.e.,

$$D^{-1} PZPD = Z_b$$

where Z_b is the balanced matrix; see [30] for details. We then find an orthogonal matrix U which reduces Z_b to ordered RSF:

$$U^T Z_b U = S \ .$$

Then PDU (produced by BALBAK) is clearly a nonsingular matrix which reduces Z to ordered RSF. The first n columns of PDU span the eigenspace corresponding to eigenvalues of Z with negative real parts and that is the only property we require of the transformation. For simplicity in the sequel, we shall speak of the transformation reducing Z to RSF as simply an orthogonal matrix U with the understanding that the more computationally attractive transformation is of the form PDU.

An alternative approach to direct balancing of Z is to attempt some sort of scaling in the problem which generates the Riccati equation. To illustrate, consider the linear optimal control problem of finding a feedback controller $u(t) = Kx(t)$ which minimizes the performance index

$$J(u) = \int_0^{+\infty} [x^T(t)Hx(t) + u^T(t)Ru(t)]dt$$

with plant constraint dynamics given by

$$\dot{x}(t) = Fx(t) + Bu(t) \quad ; \qquad x(0) = x_0 \quad .$$

We assume $H = H^T \geq 0$, $R = R^T > 0$ and (F,B) controllable, (F,C) observable where $C^TC = H$ and $\text{rank}(C) = \text{rank}(H)$. Then the optimal solution is well-known to be

$$u(t) = -R^{-1}B^TXx$$

where X solves the Riccati equation

$$F^TX + XF - XBR^{-1}B^TX + H = 0 \quad .$$

Now suppose we change coordinates via a nonsingular transformation $x(t) = Tw(t)$. Then in terms of the new state w our problem is to minimize

$$\int_{0}^{+\infty} [w^T(t)(T^THT)w(t) + u^T(t)Ru(t)]dt$$

subject to

$$\dot{w}(t) = (T^{-1}FT)w(t) + (T^{-1}B)u(t) \quad .$$

The Hamiltonian matrix Z for this transformed system is now given by

$$Z_w = \begin{pmatrix} T^{-1}FT & -T^{-1}BR^{-1}B^TT^{-T} \\ -T^THT & -T^TF^TT^{-T} \end{pmatrix}$$

and the associated solution X_w of the transformed Riccati equation is related to the original X by $X = T^{-T}X_wT^{-1}$. One interpretation of T then is as a scaling transformation, a diagonal matrix, for example, in an attempt to "balance" the elements of Z_w. Applying such a procedure, even in an ad hoc way, is frequently very useful from a computational point of view.

Another way to look at the above procedure is that Z_w is symplectically similar to Z via the transformation $\begin{pmatrix} T & 0 \\ 0 & T^{-T} \end{pmatrix}$, i.e.,

$$Z_w = \begin{pmatrix} T & 0 \\ 0 & T^{-T} \end{pmatrix}^{-1} Z \begin{pmatrix} T & 0 \\ 0 & T^{-T} \end{pmatrix} \quad .$$

It is well-known that Z_w is again Hamiltonian (or symplectic in the discrete-time case) since the similarity transformation is symplectic. One can then pose the problem of transforming Z by other, more elaborate symplectic similarities so as to achieve various desirable numerical properties or canonical forms. This topic for further research is presently being investigated.

4.3 Operation Counts, Timing, and Storage

We shall give approximate operation counts for the solution of n^{th} order algebraic Riccati equations of the form (1) or (8). Each operation is assumed to be roughly equivalent to forming a + (b x c) where a,b,c are floating point numbers. It is almost impossible to give an accurate operation count for the algorithm described above since so many factors are variable such as the ordering of the RSF. We shall indicate only a ballpark $O(n^3)$ figure.

Let us assume then that we already have at hand the 2n x 2n matrix Z of the form (2) or (9). Note, however, that unlike forming Z in (2), Z in (9) requires approximately $4\,n^3$ additional operations to construct, given only F, G, and H. This will turn out to be fairly negligible compared to the counts for the overall process. Furthermore, we shall give only order of n^3 counts for these rough estimates. The three main steps are:

		Operations
(i)	reduction of Z to upper Hessenberg from	$\frac{5}{3}(2n)^3$
(ii)	reduction of upper Hessenberg form to RSF	$\geq 4k(2n)^3$
(iii)	solution of $XU_{11} = U_{21}$	$\frac{4}{3}\,n^3$

The number k represents the average number of QR steps required per eigenvalue and is usually over-estimated by 1.5. We write $\geq 4k(2n)^3$ since, in general, the reduction may need more operations if ordering is required. Using k = 1.5 we see that the total number of operations required is at least $63\,n^3$. Should the ordering of the RSF require, say, 25% more operations than the unordered RSF, we have a ballpark estimate of about 75 n^3 for the entire process.

Timing estimates for steps (i) and (ii) may be obtained from [27] for a variety of computing environments. The additional time for balancing and for step (iii) would then add no more than about 5% to those times while the additional time for ordering the RSF is variable, but typically adds no more than about 15%. For example, adding 20% to the published figures [27] for an IBM 370/165 (a typical medium speed machine) under OS/360 at the University of Toronto using FORTRAN H Extended with Opt. = 2 and double precision arithmetic, we can construct the following table:

Riccati Equation Order n =	10	20	30	40
CPU Time (Sec.)	0.2	1.3	4.0	9.0

In fact, these times are in fairly close agreement with actual observed times for randomly chosen test examples of these orders. Note the approximately cubic behavior of time versus order.

Extrapolating these figures for a 64<u>th</u> order equation (see Example 5 in Section 6) one might expect a CPU-time in the neighborhood of 38 sec. In fact, for that particular example the time was approximately 34 sec.

It must be re-emphasized here that timing estimates derived as above are very approximate and depend on numerous factors in the actual computing environment as well as the particular input data. However, such estimates can provide very useful and quite reliable information if interpreted as providing essentially order of magnitude figures.

With respect to storage considerations the algorithm requires $8n^2 + cn$ (c = a small constant) storage locations. This fairly large figure limits applicability of the algorithm to Riccati equations on the order of about 100 or less in many common computing environments. Of course, CPU time becomes a significant factor for $n>100$, also.

4.4 Stability and Conditioning

This section will be largely speculative in nature as very few hard results are presently available. A number of areas of continuing research will be described.

With respect to stability, the implementation discussed in Section 4.1 consists of two effectively stable steps. The crucial step is the QR step and the present algorithm is probably essentially as stable as QR. The overall two step process is apparently quite stable numerically but we have no proof of that statement.

Concerning the conditioning of (1) (or (8)) almost no analytical results are known. The study of (1) is obviously more complex than the study of even the Lyapunov equation

$$F^T X + XF + H = 0 \tag{12}$$

where $H = H^T \geq 0$. And yet very little numerical analysis is known for (12). In case F is normal, a condition number with respect to inversion of the Lyapunov operator $\mathcal{F}X = F^T X + XF$ is easily shown to be given by

$$\frac{\max\limits_{i,j} |\lambda_i(F) + \lambda_j(F)|}{\min\limits_{i,j} |\lambda_i(F) + \lambda_j(F)|} .$$

But in the general case, a condition number in terms of F rather than $F^T \otimes I + I \otimes F^T$ ($\otimes$ denotes Kronecker product) has not been determined. Some empirical observations on the accuracy of solutions of certain instances of (12) suggest that one factor influencing conditioning of (12) is the proximity of the spectrum of F to the imaginary axis. To be more specific, suppose F has an eigenvalue at $a \pm jb$ with $\left|\frac{b}{a}\right| >> 1$ (typically $a < 0$ is very small). If $\left|\frac{b}{a}\right| = 0(10^t)$ we lose approximately t digits of accuracy and we might expect a condition number for the solution of (12) to also be $0(10^t)$ in this situation.

There are some close connections between (12) and (1) (and the respective discrete-time versions) and we shall indicate some preliminary observations here. A perturbation analysis or the notion of a condition number for (1) is intimately related to the condition of an associated Lyapunov equation, namely one whose "F-matrix" approximates the closed-loop matrix F-GX where X solves (1). To illustrate, suppose $X = Y + E$ where $Y = Y^T$ may be interpreted as an approximation of X. Then

$$
\begin{aligned}
0 &= F^T(Y+E) + (Y+E)F - (Y+E)G(Y+E) + H \\
&\approx (F-GY)^T E + E(F-GY) + (F^T Y + YF - YGY + H) \\
&= \hat{F}^T E + E\hat{F} + \hat{H}
\end{aligned}
$$

where we have neglected the second-order term EGE. Thus conditioning of (1) should be closely related to nearness of the closed-loop spectrum (σ(F-GX)) to the imaginary axis. Observations similar to these have been made elsewhere; see, for example, Bucy [31] where the problem is posed as one of structural stability. A condition number might, in some sense, be thought of as a quantitative measure of the degree of structural stability.

Another factor involved in the conditioning of (1) relates to the assumptions of stabilizability of (F,B) and detectability of (C,F). For example, near-unstabilizability of (F,B) in either a parametric sense or in a control energy sense (i.e., near-singular controllability Gramian) definitely causes (1) to become badly conditioned. Our experience has been that the ill-conditioning manifests itself in the algorithm by a badly conditioned U_{11}.

Work related to the conditioning of (1) and (8) is under continuing investigation and will be the subject of another paper. Such analysis is, of course, independent of the particular algorithm used to solve (1) or (8), but is useful to understand how ill-conditioning can be expected to manifest itself in a given algorithm.

5. Advantages of the Schur Vector Approach and Further General Remarks

5.1 Advantages of the Schur Vector Approach

The advantages of this algorithm over others using eigenvectors (such as Potter's approach [10] and its extensions) are obvious. Firstly, the reduction to RSF is an intermediate step in computing eigenvectors anyway (using the double Francis QR algorithm) so the Schur approach must, by definition, be faster usually by a factor of at least two. Secondly, and more importantly, this algorithm will not suffer as severely from the numerical hazards inherent in computing eigenvectors associated with multiple or near-multiple eigenvalues. The computation of eigenvectors is fraught with difficulties (see, e.g. [21] for a cogent discussion) and the eigenvectors themselves are simply not needed. All that is needed is a basis for the eigenspace spanned by the eigenvalues of Z with negative real parts (with an analogous statement for the discrete-time case). As good a basis as is possible (in the presence of rounding error) for this subspace can be found from the Schur vectors comprising the matrix $\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}$, independently of individual eigenvalue multiplicities. The reader is strongly urged to consult [32] and [21] (especially pp. 609-610) for further numerical details.

The fact that any basis for the stable eigenspace can be used to construct the Riccati equation solution has been noted by many people; see [12] or [3] among others. The main stumbling block with using the Schur vectors was the ordering problem with the RSF but once that is handled satisfactorily the algorithm is easy.

The Schur vector approach derives its desirable numerical properties from the underlying QR-type process. To summarize: if you like the eigenvector approach for solving the algebraic Riccati equation you'll like the Schur vector approach at least twice as much.

Like the eigenvector approach, the Schur vector approach has the advantage of producing the closed-loop eigenvalues (or whatever is appropriate to the particular application from which the Riccati equation arises) essentially for free. And finally, an important advantage of the Schur vector approach, in addition to its general reliability for engineering applications, is its speed in comparison with other methods. We have already mentioned the advantage, by definition, over previous eigenvector approaches but there is also generally an even more significant speed advantage over iterative methods. This advantage is particularly apparent in poorly conditioned problems and in cases in which the iterative method has a bad starting value. Of course, it is impossible to make the comparison between a direct versus iterative method any more precise for general problems but we have found it not at all uncommon for an iterative method, such as straightforward Newton [14], to take ten to thirty times as long - if, indeed, there was convergence at all.

5.2 Miscellaneous General Remarks

Remark 1: There are, in general, as many as $\binom{2n}{n}$ solutions of an $n^{\underline{th}}$ order Riccati equation corresponding to as many as $\binom{2n}{n}$ choices of n of the 2n eigenvalues of Z. Any of these solutions may also be generated by the Schur approach, as for the eigenvector approach, by an appropriate reordering of the RSF. For most control and filtering applications we are interested in the unique nonnegative definite solution and have thus concentrated the exposition on that particular case.

Remark 2: One of the most complete sources for an eigenvector-oriented proof of Theorem 5 for the general case of multiple eigenvalues is Martensson [13]. But even a casual glance at that proof exposes the awkwardness of fussing with eigenvectors and principal vectors. The proof using Schur vectors is extremely clean and easy by comparison and neatly avoids any difficulties with multiple eigenvalues. This observation is but one instance of the more general observation that Schur vectors can probably always replace principal vectors (or generalized eigenvectors) corresponding to multiple eigenvalues throughout linear control/systems theory. Principal vectors are not generally reliably computable in the presence of roundoff error anyway (see [21]) and a basis for an eigenspace - but not the particular one corresponding to the principal vectors - is all that is normally needed. Use of Schur vectors will not only frequently provide cleaner proofs but is also numerically much more attractive.

Remark 3: As an alternative to the direct proofs provided in Sections 2 and 3 one could simply appeal to the proofs given for the eigenvector approach and note that the Schur vectors are related to the eigenvectors by a nonsingular transformation. Specifically, with Z, U, and S as before, let $V \in \mathbb{R}_{2n}^{2n\times 2n}$ put Z in real Jordan form

$$V^{-1}ZV = \begin{pmatrix} -\Lambda & 0 \\ 0 & \Lambda \end{pmatrix}$$

($\mathbb{R}_{2n}^{2n\times 2n}$ denotes the set of $2n \times 2n$ matrices or rank 2n, i.e., invertible) where $-\Lambda$ is the real Jordan form of the eigenvalues of Z with negative real parts (analogous remarks apply as usual, for the discrete-time case). Furthermore, let $T \in \mathbb{R}_n^{n\times n}$ transform S_{11} to the real Jordan form $-\Lambda$. Then

$$Z\begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} = \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix}(-\Lambda)$$

and

$$Z\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} = \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} S_{11} \quad .$$

We thus have

$$Z\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} T = \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} TT^{-1}S_{11}T$$

$$= \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} T(-\Lambda)$$

Since eigenvectors are unique up to nonzero scalar multiple we must have

$$\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} T = \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} D$$

where D is diagonal and invertible. Thus $\begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix} = \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} DT^{-1}$ and since $V_{21}V_{11}^{-1}$ solves (1), $U_{21}U_{11}^{-1}$ must also solve (1) since

$$U_{21}U_{11}^{-1} = V_{21}DT^{-1}(V_{11}DT^{-1})^{-1} = V_{21}V_{11}^{-1} \quad .$$

However, we have chosen to provide self-contained proofs because of their simplicity and also because the proof in Section 3 is not as widely seen as its continuous-time counterpart.

Remark 4: The same Schur vector approach employed in this paper can also be used instead of the eigenvector approach for the nonsymmetric matrix quadratic equation

$$XEX + FX + XG + H = 0 \tag{13}$$

where $E \in \mathbb{R}^{m\times n}$, $F \in \mathbb{R}^{n\times n}$, $G \in \mathbb{R}^{m\times m}$, $H \in \mathbb{R}^{n\times m}$, and $X \in \mathbb{R}^{n\times m}$. In this case we work with the $(m+n) \times (m+n)$ matrix

$$Z = \begin{pmatrix} -G & -E \\ H & F \end{pmatrix}$$

and various solutions of (13) are determined by generating appropriate combinations of m eigenvalues of Z along the diagonal of the RSF of Z. The corresponding m Schur vectors give the solution $X = U_{21}U_{11}^{-1}$ as before where $U_{11} \in \mathbb{R}^{m\times m}$, $U_{21} \in \mathbb{R}^{n\times m}$. The analogous remarks apply for the corresponding nonsymmetric "discrete-time equation". Proofs are essentially the same in both cases. Further details on the eigenvector approach can be found in [33], [34].

Remark 5: Special cases of the matrix quadratic equations such as (1), (8), or (13) include the Lyapunov equation (12) (or its discrete-time counterpart $F^TXF - X + H = 0$) and the Sylvester equation

$$FX + XG + H = 0 \tag{14}$$

(or its discrete-time counterpart $FXG - X + H = 0$).

Thus setting an appropriate block of the Z matrix equal to 0 provides a method of solving such "linear equations" and, in fact, this method has even been proposed in the literature [35]. However, the approach probably has little to recommend it from a numerical point of view as compared to applying the Bartels-Stewart algorithm [39] and we mention it only in passing.

6. Examples

In this section we give a few examples both to illustrate various points discussed previously and to provide some numerical results for comparison with other approaches. All computations were done at M.I.T. on an IBM 370/168 using FORTRAN H Extended (Opt. = 2) and double precision arithmetic.

Example 1: The Schur vector approach is obviously not well-suited to hand computation - which partly explains its desirable numerical properties. However, to pacify a certain segment of the population a "hand example" is provided in complete detail. Consider the equation

$$A^TX + XA - XBR^{-1}B^TX + Q = 0 \tag{15}$$

which arises in a linear-quadratic optimal control context with

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad R = 1, \quad Q = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}.$$

Then

$$Z = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & -2 & -1 & 0 \end{pmatrix}$$

and the matrix

$$U = \begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{5}}{10} & -\frac{3\sqrt{5}}{10} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{\sqrt{5}}{10} & -\frac{3\sqrt{5}}{10} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{3\sqrt{5}}{10} & \frac{\sqrt{5}}{10} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{3\sqrt{5}}{10} & \frac{\sqrt{5}}{10} & \frac{1}{2} \end{pmatrix}$$

is an orthogonal matrix which reduces Z to RSF

$$S = U^T Z U = \begin{pmatrix} -1 & 0 & 1 & -\frac{1}{2} \\ 0 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} .$$

Then the unique positive definite solution of (15) is given by the solution of the linear matrix equation

$$XU_{11} = U_{21}$$

or

$$\begin{pmatrix} x_{11} & x_{12} \\ x_{12} & x_{22} \end{pmatrix} \begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{5}}{10} \\ -\frac{1}{2} & -\frac{\sqrt{5}}{10} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & -\frac{3\sqrt{5}}{10} \\ -\frac{1}{2} & -\frac{3\sqrt{5}}{10} \end{pmatrix} .$$

Thus $X = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ and it can quickly be checked that the spectrum of the "closed-loop matrix" $(A - BR^{-1}B^T X) = \begin{pmatrix} 0 & 1 \\ -1 & -2 \end{pmatrix}$ is $\{-1, -1\}$ as was evident from S_{11}.

Example 2: For checking purposes consider the solution of (15) with the following uncontrollable but stabilizable, and unobservable but detectable data:

$$A = \begin{pmatrix} 4 & 3 \\ -\frac{9}{2} & -\frac{7}{2} \end{pmatrix}, \qquad B = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \qquad R = 1, \qquad Q = \begin{pmatrix} 9 & 6 \\ 6 & 4 \end{pmatrix} .$$

The solution of (15) is $X = \begin{pmatrix} 9c & 6c \\ 6c & 4c \end{pmatrix}$ where $c = 1 + \sqrt{2}$ and the closed-loop spectrum is $\{-\frac{1}{2}, -\sqrt{2}\}$. These values were all obtained correctly to at least 14 significant figures as were the values for the corresponding discrete-time problem

$$A^T XA - X - A^T XB(R + B^T XB)^{-1} B^T XA + Q = 0 \tag{16}$$

the solution of which is

$$X = \begin{pmatrix} 9d & 6d \\ 6d & 4d \end{pmatrix}$$

where $d = \frac{1 + \sqrt{5}}{2}$ and the closed-loop spectrum is $\{-\frac{1}{2}, \frac{3 - \sqrt{5}}{2}\}$.

Example 3: For further comparison purposes consider the discrete-time system of Example 6.15 in [36] where

$$A = \begin{pmatrix} 0.9512 & 0 \\ 0 & 0.9048 \end{pmatrix}, \quad B = \begin{pmatrix} 4.877 & 4.877 \\ -1.1895 & 3.569 \end{pmatrix},$$

$$R = \begin{pmatrix} \frac{1}{3} & 0 \\ 0 & 3 \end{pmatrix}, \quad Q = \begin{pmatrix} 0.005 & 0 \\ 0 & 0.02 \end{pmatrix}.$$

The solution of (16) is given by

$$X = \begin{pmatrix} 0.010459082320970 & 0.003224644477419 \\ 0.003224644477419 & 0.050397741135643 \end{pmatrix}$$

and the feedback gain $\bar{F} = (R + B^T XB)^{-1} B^T XA$ is given by

$$\bar{F} = \begin{pmatrix} 0.071251660724426 & -0.070287376494153 \\ 0.013569839235296 & 0.045479287667006 \end{pmatrix}$$

Note the typographical error in the (1,2)-element of $\overline{F}$ in [36]. The closed-loop eigenvalues are given by

$$0.508333461684191 \text{ and } 0.688069670988913 .$$

These are definitely different from [36] but have the same sum. Our numbers do appear to be the correct ones.

<u>Example 4</u>: We now consider somewhat higher order Riccati equations arising from position and velocity control for a string of high-speed vehicles. The matrices are taken from a paper by Athans, Levine, and Levis [37]. For a string of N vehicles it is necessary to solve the Riccati equation

$$A_N^T X_N + X_N A_N - X_N B_N R_N^{-1} B_N^T X_N + Q_N = 0$$

where all matrices are of order n = 2N-1 and are given by

$$A_N = \begin{pmatrix} A_{11} & A_{12} & & & & \bigcirc & \\ & A_{22} & A_{23} & & & & \\ & & \ddots & \ddots & & & \\ & & & A_{N-2,N-2} & A_{N-2,N-1} & & \\ & \bigcirc & & & A_{N-1,N-1} & & \begin{matrix} 0 \\ -1 \end{matrix} \\ & & & & 0 \quad 0 & & -1 \end{pmatrix}$$

where $A_{k,k} = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}$, $A_{k,k+1} = \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix}$

and $B_N R_N^{-1} B_N^T = \text{diag}\{1,0,1,0,\ldots,0,1\}$

$$Q_N = \text{diag}\{0,10,0,10,\ldots,10,0\} \quad .$$

For the case of 5 vehicles we repeated the calculations presented in [37]. The correct values for X rounded to six significant figures are:

$$\begin{pmatrix}
1.36302 & 2.61722 & -0.705427 & 0.936860 & -0.293666 & 0.477354 & -0.197375 & 0.211212 & -0.166552 \\
 & 7.59255 & -1.68036 & 1.47522 & -0.459506 & 0.665147 & -0.266142 & 0.280654 & -0.211212 \\
 & & 1.77478 & 2.15771 & -0.609136 & 0.670717 & -0.262843 & 0.266142 & -0.197375 \\
 & & & 8.25770 & -1.94650 & 1.75587 & -0.670717 & 0.665147 & -0.477354 \\
 & & & & 1.80560 & 1.94650 & -0.609136 & 0.459506 & -0.293666 \\
 & & & & & 8.25770 & -2.15771 & 1.47522 & -0.936860 \\
 & & & & & & 1.77478 & 1.68036 & -0.705427 \\
 & \text{[SYMMETRIC]} & & & & & & 7.59255 & -2.61722 \\
 & & & & & & & & 1.36302
\end{pmatrix}$$

While 4 or 5 decimal places are published in [37], it can be seen that, surprisingly, only the first and sometimes the second were correct. Substitution of our full 16 decimal place solution into the Riccati equation gives a residual of norm on the order of 10^{-14} (consistent with a condition estimate of U_{11} of 26.3) while the residual for the solution in [37] has a large norm on the order of 10^{-1}. The closed-loop eigenvalues for the above problem (again rounded to six significant figures) are:

-1.00000
$-1.10779 \pm 0.852759\ j$
$-1.45215 \pm 1.26836\ j$
$-1.67581 \pm 1.51932\ j$
$-1.80486 \pm 1.66057\ j$

We also computed the Riccati solution and closed-loop eigenvalues for the cases of 10 and 20 vehicles. This involved the solutions of 19<u>th</u> and 39<u>th</u> order Riccati equations, respectively, and rather than reproduce all the numbers here we give only the first five and last five elements of the first row (or column) of X and the fastest and slowest closed-loop modes. Again all values are rounded to just six significant figures; the complete numerical solutions are available from the author.

First row (column) of Riccati Solution N=10 n=19	N=20 n=39	Fastest and Slowest Closed-Loop Modes N=10 n=19	N=20 n=39
1.40826	1.42021	-1.83667	-1.84459
2.66762	2.68008	± 1.69509 j	± 1.70368 j
-0.658219	-0.646127	⋮	⋮
1.04031	1.06539	-0.862954	-0.662288
-0.242133	-0.229761	± 0.494661 j	
⋮	⋮		
-0.0515334	-0.0123718		
0.103453	0.0250824		
-0.0472086	-0.0120915		
0.0504036	0.0124632		
-0.0452352	-0.0119545		

The closed-loop eigenvalues for the case of, say, 10 vehicles interlace and include, as a subset, those of 5 vehicles. Similarly, those for 20 vehicles interlace and include, as a subset, those of 10 (and hence 5) vehicles. It appears evident that both the elements of the Riccati solution and the closed-loop eigenvalues are converging to values in some finite region.

Example 5: This example involves circulant matrices. We wish to solve (15) with

$$A = \begin{pmatrix} -2 & 1 & 0 & \cdots & 0 & 1 \\ 1 & \ddots & \ddots & & & 0 \\ 0 & \ddots & \ddots & \ddots & \bigcirc & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \bigcirc & & \ddots & \ddots & 1 \\ 1 & 0 & \cdots & 0 & 1 & -2 \end{pmatrix}$$

and $BR^{-1}B^T = I$, $Q = I$. The matrices A, $BR^{-1}B^T$, Q are all circulant so the Riccati solution $X \in \mathbb{R}^{nxn}$ is known to be circulant of the form

$$X = \begin{pmatrix} x_0 & x_{n-1} & x_{n-2} & \cdots & x_1 \\ x_1 & x_0 & x_{n-1} & \cdots & x_2 \\ x_2 & x_1 & x_0 & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ x_{n-1} & x_{n-2} & \cdots & \cdots & x_0 \end{pmatrix} .$$

In fact, there is a simple transformation which "diagonalizes" the Riccati equation and allows the solution of (15) to be recovered via the solution of n scalar quadratic equations and an inverse discrete Fourier transform. The details of this procedure and related analysis of circulant systems can be found in the work of Wall [38]. For this example, we have $n = 64$ and the x_i are given by

$$x_i = \frac{1}{64} \sum_{k=0}^{63} \left\{ -2+2\cos\left(\frac{2\pi k}{64}\right) + \sqrt{5-4\cos\left(\frac{2\pi k}{64}\right) + 4\cos^2\left(\frac{2\pi k}{64}\right)} \right\} \omega_{64}^{ik}$$

where ω_{64} is a 64-th root of unity. The solution was computed by the Schur vector approach and checked by means of the circulant analysis of Wall. Our computed Riccati solution had at least 13 significant figures. For reference purposes we list

x_{11} = 0.37884325313566

x_{12} = 0.18581947375535

.
.
.

x_{44} = 0.37884325313567

x_{45} = 0.18581947375536

.
.
.

The closed-loop eigenvalues are all real and are arranged as follows:

-4.1231056256177

-4.1137632861146 }
-4.1137632861146 } 31 eigenvalues of multiplicity 2
. }
. }
. }

-0.99999999999991

This $64^{\underline{th}}$ order example required approximately 50 sec. of CPU time on the 370/168 at M.I.T. and approximately 34 sec. on the 370/165 at the University of Toronto - both using FORTRAN H Extended (Opt. =2), double precision.

Example 6: This example is one which would be expected to cause problems on physical grounds and which appears to give rise to an "ill-conditioned Riccati equation". Consider the solution of (15) with

$$A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & 0 \\ \vdots & & 0 & & 1 \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

$$Q = \mathrm{diag}\{q,0,\ldots,0\}, \quad R = r \; .$$

Here we have a system of n integrators connected in series. It is desired to apply a feedback controller to the $n^{\underline{th}}$ system (which is to be integrated n times) so as to achieve overall asymptotic stability. Only deviations of x_1 (the $n^{\underline{th}}$ integral of the constant system) from 0 are penalized. The controllability Gramian

$$W_t = \int_0^t e^{sA} BB^T e^{sA^T} ds \quad ,$$

while positive definite for all $t>0$, becomes more nearly singular as n increases. The system is "hard to control" in the sense of requiring a large amount of control energy (as measured by $||W_t^{-1}||$).

The closed-loop eigenvalues are easily seen to be the roots of

$$\lambda^{2n} + (-1)^n \frac{q}{r} = 0$$

with negative real parts. These eigenvalues lie in a classic Butterworth pattern. It can also be easily verified that

$$x_{1n} = \sqrt{\frac{q}{r}}$$

$$= \text{product of the closed-loop eigenvalues} \; .$$

We attempted the solution of (15) with the above matrices and $q = r = 1$. While the closed-loop eigenvalues were determined quite accurately as expected (approximately 14 decimal places using IBM double precision), the Riccati solution was increasingly less accurate as n increased due to the increasingly ill-conditioned nature of U_{11}. For example, for $n = 21$ there was already a loss of 10 digits of accuracy (consistent with a condition estimate of $O(10^{10})$ for U_{11}) in x_{1n} (=1). Other computed elements of X were as large as $O(10^{9})$ in magnitude.

Repeating the calculations with $q = 10^4$, $r = 1$ there was a loss of approximately 12 digits of accuracy in x_{1n}(=100) for $n = 21$. In this case other elements of X were as large as $O(10^{11})$ in magnitude. Again, the closed-loop eigenvalues were determined very accurately.

Our attempts to get Newton's method to converge on the above problem were unsuccessful.

Obviously, there is more that can be said analytically about this problem. Our interest here has been only to highlight some of the numerical difficulties.

7. Concluding Remarks

We have discussed in considerable detail a new algorithm for solving algebraic Riccati equations. A number of numerical issues have been addressed and various examples given. The method is apparently quite numerically stable and performs reliably on systems with dense matrices of up to order 100 or so, storage being the main limiting factor.

For some reason, numerical analysts have never really studied algebraic Riccati equations. The algorithm presented here can undoubtedly be refined considerably from a numerical point of view but it nonetheless represents an immense improvement over algorithms heretofore proposed.

Some topics of continuing research in this area will include:

(i) conditioning of Riccati equations,

(ii) use of software to sort blocks of the RSF diagonal into just the two appropriate groups rather than within the two groups as well,

(iii) making numerically viable the use of symplectic transformations such as in [17] to reduce the Hamiltonian or symplectic matrix Z to a convenient canonical form.

Each of these topics is of research interest in its own right in addition to the application to Riccati equations.

8. References

[1] Wonham, W.M., On a Matrix Riccati Equation of Stochastic Control, SIAM J. Contr., 6(1968), 681-697.

[2] Reid, W.T., Riccati Differential Equations, Academic Press, New York, 1972.

[3] Willems, J.C., Least Squares Stationary Optimal Control and the Algebraic Riccati Equation, IEEE Trans. Aut. Contr., AC-16(1971), 621-634.

[4] Silverman, L.M., Discrete Riccati Equations: Alternative Algorithms, Asymptotic Properties, and System Theory Interpretations, in Advances in Control Systems, Vol. 12, (Leondes, Ed.), Academic Press, New York, 1976, pp. 313-386.

[5] Lainiotis, D.G., Partitioned Riccati Solutions and Integration-Free Doubling Algorithms, IEEE Trans. Auto. Contr., AC-21 (1976), 677-689.

[6] Rodriguez-Canabal, J., The Geometry of the Riccati Equation, Stochastics, 1(1973), 129-149.

[7] Pachter, M., and T.E. Bullock, Ordering and Stability Properties of the Riccati Equation, Na. Res. Inst. for Math. Sci. Report, WISK 264, Pretoria, June 1977.

[8] Von Escherich, G., Die Zweite Variation der Einfachen Integrale, Wiener Sitzungsberichte, 8(1898), 1191-1250.

[9] MacFarlane, A.G.J., An Eigenvector Solution of the Optimal Linear Regulator Problem, J. Electron. Contr., 14(1963), 643-654.

[10] Potter, J.E., Matrix Quadratic Solutions, SIAM J. Appl. Math., 14(1966), 496-501.

[11] Vaughn, D.R., A Nonrecursive Algebraic Solution for the Discrete Riccati Equation, IEEE Trans. Auto. Contr., AC-15(1970), 597-599.

[12] Fath, A.F., Computational Aspects of the Linear Optimal Regulator Problem, IEEE Trans. Auto. Contr., AC-14(1969), 547-550.

[13] Mårtensson, K., New Approaches to the Numerical Solution of Optimal Control Problems, Lund Institute of Technol., Divis. of Auto. Contr., Report No. 7206, Lund, Sweden, Mar. 1972.

[14] Kleinman, D.L., On An Iterative Technique for Riccati Equation Computations, IEEE Trans. Auto. Contr., AC-13(1968), 114-115.

[15] Farrar, F.A., and R.C. DiPietro, Comparative Evaluation of Numerical Methods for Solving the Algebraic Matrix Riccati Equation , United Technologies Research Center Report No. R76-140268-1, East Hartford, CT, Dec. 1976.

[16] Hewer, G.A., and G. Nazaroff, A Survey of Numerical Methods for the Solution of Algebraic Riccati Equations, Naval Weapons Center Report, China Lake, CA.

[17] Laub, A.J., and K.R. Meyer, Canonical Forms for Hamiltonian and Symplectic Matrices, Celestial Mechanics, 9(1974), 213-238.

[18] Laub, A.J., Canonical Forms for σ-Symplectic Matrices, M.S. Thesis, School of Mathematics, Univ. of Minnesota, Dec. 1972.

[19] Stewart, G.W., Introduction to Matrix Computations, Academic Press, New York, 1973.

[20] Murnaghan, F.D., and A. Wintner, A Canonical Form for Real Matrices Under Orthogonal Transformations, Proc. Na. Acad. Sci., 17(1931), 417-420.

[21] Golub, G.H., and J.H. Wilkinson, Ill-Conditioned Eigensystems and the Computation of the Jordan Canonical Form, SIAM Rev., 18(1976), 578-619.

[22] Gantmacher, F.R., The Theory of Matrices, Chelsea, New York, 1959.

[23] Dorato, P., and A. Levis, Optimal Linear Regulators: The Discrete-Time Case, IEEE Trans. Auto. Contr., AC-16(1971), 613-620.

[24] Albert, A., Conditions for Positive and Nonnegative Definiteness in Terms of Pseudoinverses, SIAM J. Appl. Math., 17(1969), 434-440.

[25] Penrose, R., A Generalized Inverse for Matrices, Proc. Cambr. Phil. Soc., 51(1955), 406-413.

[26] Stewart, G.W., HQR3 and EXCHNG: Fortran Subroutines for Calculating and Ordering the Eigenvalues of a Real Upper Hessenberg Matrix, ACM Trans. Math. Software, 2(1976), 275-280.

[27] Smith, B.T., et.al., Matrix Eigensystems Routines -- EISPACK Guide, Second Edition, Lect. Notes in Comp. Sci., Vol. 6, Springer-Verlag, New York, 1976.

[28] Forsythe, G.E., M.A. Malcolm, and C.B. Moler, Computer Methods for Mathematical Computations, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[29] Dongarra, J.J., J.R. Bunch, C.B. Moler, and G.W. Stewart, Preliminary LINPACK User's Guide, LINPACK Working Note #9, Argonne National Laboratory, Appl. Math. Div., TM-313, Aug. 1977.

[30] Parlett, B.N., and C. Reinsch, Balancing a Matrix for Calculation of Eigenvalues and Eigenvectors, Numer. Math., 13(1969), 296-304.

[31] Bucy, R.S., Structural Stability for the Riccati Equation, SIAM J. Contr., 13(1975), 749-753.

[32] Wilkinson, J.H., The Algebraic Eigenvalue Problem, Oxford University Press, London, 1965.

[33] Coppel, W.A., Matrix Quadratic Equations, Bull. Austral. Math. Soc., 10(1974), 377-401.

[34] Meyer, H.-B., The Matrix Equation $AZ+B-ZCZ-ZD=0$, SIAM J. Appl. Math., 30(1976), 136-142.

[35] Bar-Ness, Y., and G. Langholz, The Solution of the Matrix Equation $XC-BX=D$ as an Eigenvalue Problem, Int. J. Sys. Sci., 8(1977), 385-392.

[36] Kwakernaak, H., and R. Sivan, Linear Optimal Control Systems, Wiley, New York, 1972.

[37] Athans, M., W.S. Levine, and A. Levis, A System for the Optimal and Suboptimal Position and Velocity Control for a String of High-Speed Vehicles, Proc. 5th International Analogue Computation Meetings, Lausanne, Switzerland, Sept. 1967.

[38] Wall, J.E., Control and Estimation for Large-Scale Systems Having Spatial Symmetry, Ph.D. Thesis, M.I.T., Aug. 1978; Electronic Systems Lab. Rept. ESL-TH-842.

[39] Bartels, R.H., and G.W. Stewart, Solution of the Matrix Equation $AX+XB=C$, Comm. ACM, 15(1972), 820-826.

APPENDIX 1

We outline here how to set up the "symplectic approach" when the matrix F in

$$F^T XF - X - F^T XG_1(G_2+G_1^T XG_1)^{-1}G_1^T XF + H = 0$$

is singular. All other assumptions and notation of Section 3 will be the same.

Letting x_k denote the state at time t_k and λ_k the corresponding adjoint vector, recall the Hamiltonian difference equations arising from the discrete maximum principle:

$$\begin{pmatrix} I & G \\ 0 & F^T \end{pmatrix} \begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} F & 0 \\ -H & I \end{pmatrix} \begin{pmatrix} x_k \\ \lambda_k \end{pmatrix} .$$

Note that if F were invertible we could work with the symplectic matrix

$$\begin{pmatrix} I & G \\ 0 & F^T \end{pmatrix}^{-1} \begin{pmatrix} F & 0 \\ -H & I \end{pmatrix} = \begin{pmatrix} F + GF^{-T}H & -GF^{-T} \\ -F^{-T}H & F^{-T} \end{pmatrix}$$

which is just (9). Here, instead, we shall be concerned with a "symplectic generalized eigenvalue problem"

$$Lz = \lambda Mz$$

with

$$L = \begin{pmatrix} F & 0 \\ -H & I \end{pmatrix} \qquad M = \begin{pmatrix} I & G \\ 0 & F^T \end{pmatrix}$$

and symplectic in the sense that if $\lambda \neq 0$ is a generalized eigenvalue then $\frac{1}{\lambda}$ is a generalized eigenvalue. In fact, L and M are characterized by the property that

$$LJL^T = MJM^T \quad \text{where} \quad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}.$$

In our specific situation $LJL^T = MJM^T = \begin{pmatrix} 0 & F \\ -F^T & 0 \end{pmatrix}$.

There is even more "reciprocal symmetry" in the problem. With F singular there must be least one generalized eigenvalue at 0 and to each such generalized eigenvalue there corresponds its reciprocal at ∞. The generalized eigenvalues can then be arranged in two groups of n as before:

$$\underbrace{0,\ldots,0,\lambda_1,\ldots,\lambda_k}_{n},\ \underbrace{\frac{1}{\lambda_1},\ldots,\frac{1}{\lambda_k},\ \infty,\ldots,\infty}_{n}$$

with $0<|\lambda_i|<1$. We then find a basis for the generalized eigenspace corresponding to $0,\ldots,0,\lambda_1,\ldots,\lambda_k$ and proceed essentially as before. The details are omitted here as they are the subject of a forthcoming paper with T. Pappas.

APPENDIX 2

In this appendix we provide FORTRAN source listings for one possible implementation of the Schur vector approach described in the paper. Subroutines for solving both the continuous-time algebraic Riccati equation (1) [RICCND] and the discrete-time algebraic Riccati equation (8) [RICDSD] are given. The subroutine names are derived from the following nomenclature convention for a family of subroutines to solve Riccati and various other matrix equations:

subroutine name: XXXYYZ

where

XXX = { RIC Riccati equation; LYP Lyapunov equation; SYL Sylvester equation

YY = { CN continuous-time version; DS discrete-time version

Z = { S single (short) precision version; D double (long) precision version

Subroutine RICCND calls or further requires the following additional subroutines:

BALANC, BALBAK, DDCOMP, DSOLVE, EXCHNG, HQR3, MLINEQ, ORTHES, ORTRAN, QRSTEP, SPLIT

Subroutine RICDSD requires each of the 11 subroutines above as well as the two additional subroutines MULWOA, MULWOB.

All the additional subroutines required have also been listed here with the exception of BALANC, BALBAK, ORTHES, and ORTRAN which are available in EISPACK [27].

These subroutines are being used in the environment described in Section 6 as part of a package called LQGPACK. This package is a preliminary version of a set of subroutines being developed at M.I.T.'s Laboratory for Information and Decision Systems to solve linear-quadratic-Gaussian control and estimation problems. The package has also been run in a single precision version on a CDC 6600. However, at this time we make no claims of portability of the code to other machines. The code listed here is solely for illustrative purposes.

Finally, we add two additional technical notes:

NOTE 1: A fairly reliable estimate of the condition number of U_{11} with respect to inversion is returned by RICCND or RICDSD in WORK (1).

NOTE 2: The subroutine HQR3 contains a small error which can occasionally cause RICCND or RICDSD to give erroneous or misleading information. The trouble arises when ORTHES produces an upper Hessenberg form with a zero on the first subdiagonal. HQR3 then correctly orders the resulting RSF both above and below that zero element but not necessarily globally. In practice this almost never happens and it has only ever been observed for certain low-order examples with all coefficient matrices diagonal.

This error in HQR3 can and will be corrected. In the interim, the error can either be ignored (a safe strategy for virtually all "real problems") or temporarily patched by the following scheme.

Let $a_{i+1,i}$ be a zero element of the upper Hessenberg matrix A (the output of ORTHES). Then before HQR3 is called, $a_{i+1,i}$ should be replaced by $\epsilon \cdot \left(|a_{i,i}| + |a_{i+1,i+1}|\right)$ where ϵ is the machine precision (EPS) defined by

$$\epsilon = \min_{\delta} \{\delta: f\ell(1+|\delta|) \neq 1\}$$

($f\ell(\cdot)$ denotes floating point operation).

The source listings now follow.

```
      SUBROUTINE RICCND (NZ,NF,NG,NH,N,NN,Z,W,F,G,H,ER,EI,WORK,
     +                   SCALE,ITYPE,IPVL,IPVS)
C
C     *****PARAMETERS:
      INTEGER NZ,NF,NG,NH,N,NN,ITYPE(NN),IPVL(NN),IPVS(N)
      DOUBLE PRECISION Z(NZ,NN),W(NZ,NN),F(NF,N),G(NG,N),H(NH,N),
     +                 ER(NN),EI(NN),WORK(N),SCALE(NN)
C
C     *****LOCAL VARIABLES:
      INTEGER I,J,LOW,IGH,NLOW,NUP
      DOUBLE PRECISION EPS,EPSP1,ZNORM,T,ALPHA,COND
C
C     *****FUNCTIONS:
      DOUBLE PRECISION DABS,DSQRT
C
C     *****SUBROUTINES CALLED:
C     BALANC,BALBAK,HQR3,MLINEQ,ORTHES,ORTRAN
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     THIS SUBROUTINE SOLVES THE CONTINUOUS-TIME
C     ALGEBRAIC MATRIX RICCATI EQUATION
C                           T
C                          F *X + X*F - X*G*X + H = 0
C
C     BY LAUB'S VARIANT OF THE HAMILTONIAN-EIGENVECTOR APPROACH.
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C
C        NZ,NF,NG,NH         ROW DIMENSIONS OF THE ARRAYS CONTAINING
C                            Z (AND W),F,G, AND H, RESPECTIVELY, AS
C                            DECLARED IN THE CALLING PROGRAM DIMENSION
C                            STATEMENT;
C
C        N                   ORDER OF THE MATRICES F,G,H;
C
C        NN                  = 2*N = ORDER OF THE INTERNALLY GENERATED
C                            MATRICES Z AND W;
C
C        F                   AN N X N (REAL) MATRIX;
C
C        G,H                 N X N SYMMETRIC, NONNEGATIVE DEFINITE
C                            (REAL) MATRICES.
C
C     ON OUTPUT:
C
C        H                   AN N X N ARRAY CONTAINING THE UNIQUE POSITIVE
C                            (OR NONNEGATIVE) DEFINITE SOLUTION OF THE
C                            RICCATI EQUATION;
C
C        ER,EI               REAL SCRATCH VECTORS OF LENGTH 2*N; ON OUTPUT
C                            (ER(I),EI(I)), I=1,N CONTAIN THE REAL AND
C                            IMAGINARY PARTS, RESPECTIVELY, OF THE N
```

```
C                          CLOSED LOOP EIGENVALUES (I.E., THE
C                          SPECTRUM OF F - G*X);
C
C        Z,W               2*N X 2*N REAL SCRATCH ARRAYS USED FOR
C                          COMPUTATIONS INVOLVING THE HAMILTONIAN
C                          MATRIX ASSOCIATED WITH THE RICCATI EQUATION;
C
C        WORK,SCALE        REAL SCRATCH VECTORS OF LENGTHS N, 2*N,
C                          RESPECTIVELY;  ON OUTPUT, WORK(1) CONTAINS A
C                          CONDITION NUMBER ESTIMATE FOR THE FINAL NTH
C                          ORDER LINEAR MATRIX EQUATION SOLVED;
C
C        ITYPE,IPVL,IPVS   INTEGER SCRATCH VECTORS OF LENGTHS 2*N, 2*N,
C                          N, RESPECTIVELY.
C
C     ***NOTE:  ALL SCRATCH ARRAYS MUST BE DECLARED AND INCLUDED
C               IN THE CALL.***
C
C     *****ALGORITHM NOTES:
C     IT IS ASSUMED THAT G AND H ARE NONNEGATIVE DEFINITE AND THAT (F,B)
C                                                          T
C     IS STABILIZABLE AND (C,F) IS DETECTABLE WHERE B*B  = G
C                                             T
C     (B OF FULL RANK = RANK(G)) AND C*C = H (C OF FULL
C     RANK = RANK(H)) IN WHICH CASE THE SOLUTION (RETURNED IN THE
C     ARRAY H) IS UNIQUE AND NONNEGATIVE DEFINITE.
C
C     *****HISTORY:
C     WRITTEN BY ALAN J. LAUB (ELEC. SYS. LAB., M.I.T., RM. 35-331,
C     CAMBRIDGE, MA 02139, PH.: (617) - 253-2125), SEPTEMBER 1977.
C     MOST RECENT VERSION:  SEP. 15, 1978.
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     EPS IS AN INTERNALLY GENERATED MACHINE DEPENDENT PARAMETER
C     SPECIFYING THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C     FOR EXAMPLE, EPS = 16.0D0**(-13) FOR DOUBLE PRECISION ARITHMETIC
C     ON IBM S360/S370.
C
      EPS=1.0D0
    5 EPS=0.5D0*EPS
      EPSP1=EPS+1.0D0
      IF (EPSP1.GT.1.0D0) GO TO 5
      EPS=2.0D0*EPS
C
C     SET UP HAMILTONIAN MATRIX
C
      DO 20 J=1,N
         DO 10 I=1,N
               Z(I,J)=F(I,J)
               Z(N+I,J)=-H(I,J)
               Z(I,N+J)=-G(I,J)
               Z(N+I,N+J)=-F(J,I)
   10    CONTINUE
   20 CONTINUE
```

```
C
C     BALANCE Z
C
      CALL BALANC (NZ,NN,Z,LOW,IGH,SCALE)
C
C     COMPUTE 1-NORM OF Z
C
      ZNORM=0.0D0
      DO 40 J=1,NN
         T=0.0D0
         DO 30 I=1,NN
               T=T+DABS(Z(I,J))
30       CONTINUE
         IF (T.GT.ZNORM) ZNORM=T
40    CONTINUE
      ALPHA=DSQRT(ZNORM)+1.0D0
C
C                                  -1
C     COMPUTE W = (ALPHA*I + Z)   *(ALPHA*I - Z), AN ANALYTIC FUNCTION
C     OF Z MAPPING THE LEFT HALF PLANE TO THE EXTERIOR OF THE UNIT
C     DISK.  THIS PERMITS DIRECT APPLICATION OF HQR3.  THIS STEP MAY
C     BE REMOVED IF HQR3 IS MODIFIED APPROPRIATELY.
C
      DO 60 J=1,NN
         DO 50 I=1,NN
               W(I,J)=-Z(I,J)
50       CONTINUE
         W(J,J)=ALPHA+W(J,J)
         Z(J,J)=ALPHA+Z(J,J)
60    CONTINUE
      CALL MLINEQ (NZ,NZ,NN,NN,Z,W,COND,IPVL,ER)
C
C     REDUCE W TO REAL SCHUR FORM WITH EIGENVALUES OUTSIDE THE UNIT
C     DISK IN THE UPPER LEFT N X N UPPER QUASI-TRIANGULAR BLOCK
C
      NLOW=1
      NUP=NN
      CALL ORTHES (NZ,NN,NLOW,NUP,W,ER)
      CALL ORTRAN (NZ,NN,NLOW,NUP,W,ER,Z)
      DO 15 I=2,NN
         IF(W(I,I-1).EQ.0.0D0) W(I,I-1)=1.0D-14
   15 CONTINUE
      CALL HQR3 (W,Z,NN,NLOW,NUP,EPS,ER,EI,ITYPE,NZ,NZ)
C
C     COMPUTE SOLUTION OF THE RICCATI EQUATION FROM THE ORTHOGONAL
C     MATRIX NOW IN THE ARRAY Z.  STORE THE RESULT IN THE ARRAY H.
C
      CALL BALBAK (NZ,NN,LOW,IGH,SCALE,NN,Z)
      DO 80 J=1,N
         DO 70 I=1,N
               F(I,J)=Z(J,I)
               H(I,J)=Z(N+J,I)
70       CONTINUE
80    CONTINUE
      CALL MLINEQ (NF,NH,N,N,F,H,COND,IPVS,WORK)
```

```
      WORK(1)=COND
C
C     TRANSFORM BACK TO GET THE CLOSED LOOP SPECTRUM
C
      DO 110 I=1,N
         IF (ITYPE(I).GE.0) GO TO 90
         WRITE (6,44400) I
44400    FORMAT (1X,I4,1X,41HTH EIGENVALUE NOT SUCCESSFULLY CALCULATED)
         RETURN
90       IF (ITYPE(I).GT.0) GO TO 100
         ER(I)=ALPHA*(1.0D0-ER(I))/(1.0D0+ER(I))
         EI(I)=0.0D0
         GO TO 110
100      IF (ITYPE(I).EQ.2) GO TO 110
         T=ALPHA/((1.0D0+ER(I))**2+EI(I)**2)
         ER(I)=(1.0D0-ER(I)**2-EI(I)**2)*T
         EI(I)=-2.0D0*EI(I)*T
         ER(I+1)=ER(I)
         EI(I+1)=-EI(I)
110   CONTINUE
      RETURN
C
C     LAST LINE OF RICCND
C
      END
```

```
      SUBROUTINE RICDSD (NZ,NF,NG,NH,N,NN,Z,W,F,G,H,ER,EI,WORK,
     +                   SCALE,ITYPE,IPVT)
C
C     *****PARAMETERS:
      INTEGER NZ,NF,NG,NH,N,NN,ITYPE(NN),IPVT(N)
      DOUBLE PRECISION Z(NZ,NN),W(NZ,NN),F(NF,N),G(NG,N),H(NH,N),
     +                 ER(NN),EI(NN),WORK(N),SCALE(NN)
C
C     *****LOCAL VARIABLES:
      INTEGER I,J,K,LOW,IGH,NLOW,NUP
      DOUBLE PRECISION EPS,EPSP1,COND,CONDP1
C
C     *****SUBROUTINES CALLED:
C     BALANC,BALBAK,DDCOMP,DSOLVE,HQR3,MLINEQ,MULWOA,MULWOB,
C     ORTHES,ORTRAN
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     THIS DOUBLE PRECISION SUBROUTINE SOLVES THE DISCRETE-TIME
C     ALGEBRAIC MATRIX RICCATI EQUATION
C
C                  T          T                   T       -1     T
C          X = F *X*F - F *X*G1*((G2 + G1 *X*G1)  )*G1 *X*F + H
C
C     BY LAUB'S VARIANT OF THE HAMILTONIAN-EIGENVECTOR APPROACH.
C     THE MATRIX F IS ASSUMED TO BE NONSINGULAR AND THE MATRICES G1 AND
C     G2 ARE ASSUMED TO BE COMBINED INTO THE SQUARE ARRAY G AS FOLLOWS:
C                                 -1   T
C                       G = G1*G2  *G1
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C
C        NZ,NF,NG,NH        ROW DIMENSIONS OF THE ARRAYS CONTAINING
C                           Z (AND W),F,G, AND H, RESPECTIVELY, AS
C                           DECLARED IN THE CALLING PROGRAM DIMENSION
C                           STATEMENT;
C
C        N                  ORDER OF THE MATRICES F,G,H;
C
C        NN                 = 2*N = ORDER OF THE INTERNALLY GENERATED
C                           MATRICES Z AND W;
C
C        F                  A NONSINGULAR N X N (REAL) MATRIX;
C
C        G,H                N X N SYMMETRIC, NONNEGATIVE DEFINITE
C                           (REAL) MATRICES.
C
C     ON OUTPUT:
C
C        H                  AN N X N ARRAY CONTAINING THE UNIQUE POSITIVE
C                           (OR NONNEGATIVE) DEFINITE SOLUTION OF THE
C                           RICCATI EQUATION;
C
```

```
C       ER,EI                 REAL SCRATCH VECTORS OF LENGTH 2*N; ON OUTPUT
C                             (ER(I),EI(I)), I=1,N  CONTAIN THE REAL AND
C                             IMAGINARY PARTS, RESPECTIVELY, OF THE N
C                             CLOSED LOOP EIGENVALUES (I.E., THE
C                             SPECTRUM OF                T       -1     T
C                                        F - G1*((G2 + G1 *X*G1)   )*G1 *X*F
C                                                      -T
C                                      = F - G*F   *(X - H));
C
C       Z,W                   2*N X 2*N REAL SCRATCH ARRAYS USED FOR
C                             COMPUTATIONS INVOLVING THE SYMPLECTIC
C                             MATRIX ASSOCIATED WITH THE RICCATI EQUATION;
C
C       WORK,SCALE            REAL SCRATCH VECTORS OF LENGTHS N, 2*N,
C                             RESPECTIVELY;  ON OUTPUT, WORK(1) CONTAINS A
C                             CONDITION NUMBER ESTIMATE FOR THE FINAL NTH
C                             ORDER LINEAR MATRIX EQUATION SOLVED;
C
C       ITYPE,IPVT            INTEGER SCRATCH VECTORS OF LENGTHS 2*N, N,
C                             RESPECTIVELY.
C
C     ***NOTE:  ALL SCRATCH ARRAYS MUST BE DECLARED AND INCLUDED
C               IN THE CALL.***
C
C     *****ALGORITHM NOTES:
C     IT IS ASSUMED THAT:
C        (1)  F IS NONSINGULAR
C        (2)  G AND H ARE NONNEGATIVE DEFINITE
C        (3)  (F,G1) IS STABILIZABLE AND (C,F) IS DETECTABLE WHERE
C              T
C             C *C = H (C OF FULL RANK = RANK(H)).
C     UNDER THESE ASSUMPTIONS THE SOLUTION (RETURNED IN THE ARRAY H) IS
C     UNIQUE AND NONNEGATIVE DEFINITE.
C
C     *****HISTORY:
C     WRITTEN BY ALAN J. LAUB (ELEC. SYS. LAB., M.I.T., RM. 35-331,
C     CAMBRIDGE, MA 02139, PH.: (617) - 253-2125), SEPTEMBER 1977.
C     MOST RECENT VERSION:  SEP. 15, 1978.
C
C     :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     EPS IS AN INTERNALLY GENERATED MACHINE DEPENDENT PARAMETER
C     SPECIFYING THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C     FOR EXAMPLE, EPS = 16.0D0**(-13) FOR DOUBLE PRECISION ARITHMETIC
C     ON IBM S360/S370.
C
      EPS=1.0D0
    5 EPS=0.5D0*EPS
      EPSP1=EPS+1.0D0
      IF (EPSP1.GT.1.0D0) GO TO 5
      EPS=2.0D0*EPS
C
C     SET UP SYMPLECTIC MATRIX Z
C
      DO 20 J=1,N
```

```
          DO 10 I=1,N
                Z(N+I,N+J)=F(J,I)
10        CONTINUE
20     CONTINUE
       CALL DDCOMP (NF,N,F,COND,IPVT,WORK)
       CONDP1=COND+1.0D0
       IF (CONDP1.GT.COND) GO TO 30
       WRITE (6,44400)
44400 FORMAT (42H1F MATRIX IS SINGULAR TO WORKING PRECISION)
       RETURN
30     DO 60 J=1,N
          DO 40 I=1,N
                WORK(I)=0.0D0
40        CONTINUE
          WORK(J)=1.0D0
          CALL DSOLVE (NF,N,F,WORK,IPVT)
          DO 50 I=1,N
                Z(I,J)=WORK(I)
50        CONTINUE
60     CONTINUE
       DO 80 J=1,N
          DO 70 I=1,N
                F(I,J)=Z(I,J)
70        CONTINUE
80     CONTINUE
       CALL MULWOA (NH,NF,N,H,F,WORK)
       DO 120 J=1,N
          DO 90 I=1,N
                Z(I,N+J)=0.0D0
                Z(N+I,J)=H(I,J)
90        CONTINUE
          DO 110 K=1,N
                DO 100 I=1,N
                      Z(I,N+J)=Z(I,N+J)+F(I,K)*G(K,J)
100             CONTINUE
110       CONTINUE
120    CONTINUE
       CALL MULWOB (NH,NG,N,H,G,WORK)
       DO 140 J=1,N
          DO 130 I=1,N
                Z(N+I,N+J)=Z(N+I,N+J)+G(I,J)
130       CONTINUE
140    CONTINUE
C
C      BALANCE Z
C
       CALL BALANC (NZ,NN,Z,LOW,IGH,SCALE)
C
C      REDUCE Z TO REAL SCHUR FORM WITH EIGENVALUES OUTSIDE THE UNIT
C      DISK IN THE UPPER LEFT N X N UPPER QUASI-TRIANGULAR BLOCK
C
       NLOW=1
       NUP=NN
       CALL ORTHES (NZ,NN,NLOW,NUP,Z,ER)
       CALL ORTRAN (NZ,NN,NLOW,NUP,Z,ER,W)
```

```
      CALL HQR3 (Z,W,NN,NLOW,NUP,EPS,ER,EI,ITYPE,NZ,NZ)
C
C     COMPUTE SOLUTION OF THE RICCATI EQUATION FROM THE ORTHOGONAL
C     MATRIX NOW IN THE ARRAY W.  STORE THE RESULT IN THE ARRAY H.
C
      CALL BALBAK (NZ,NN,LOW,IGH,SCALE,NN,W)
      DO 160 J=1,N
         DO 150 I=1,N
             F(I,J)=W(J,I)
             H(I,J)=W(N+J,I)
150      CONTINUE
160   CONTINUE
      CALL MLINEQ (NF,NH,N,N,F,H,COND,IPVT,WORK)
      WORK(1)=COND
C
C     TRANSFORM TO GET THE CLOSED LOOP SPECTRUM
C
      DO 190 I=1,N
         IF (ITYPE(I).GE.0) GO TO 170
         WRITE (6,44410) I
44410    FORMAT (1X,I4,1X,41HTH EIGENVALUE NOT SUCCESSFULLY CALCULATED)
         RETURN
170      IF (ITYPE(I).GT.0) GO TO 180
         ER(I)=1.0D0/ER(I)
         EI(I)=0.0D0
         GO TO 190
180      IF (ITYPE(I).EQ.2) GO TO 190
         T=ER(I)**2+EI(I)**2
         ER(I)=ER(I)/T
         EI(I)=EI(I)/T
         ER(I+1)=ER(I)
         EI(I+1)=-EI(I)
190   CONTINUE
      RETURN
C
C     LAST LINE OF RICDSD
C
      END
```

```
      SUBROUTINE DDCOMP (NA,N,A,COND,IPVT,WORK)
C
C     *****PARAMETERS:
      INTEGER NA,N,IPVT(N)
      DOUBLE PRECISION A(NA,N),COND,WORK(N)
C
C     *****LOCAL VARIABLES:
      INTEGER NM1,I,J,K,KP1,KB,KM1,M
      DOUBLE PRECISION EK,T,ANORM,YNORM,ZNORM
C
C     *****FUNCTIONS:
      DOUBLE PRECISION DABS
C
C     :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     THIS SUBROUTINE COMPUTES AN LU-DECOMPOSITION OF THE REAL N X N
C     MATRIX A BY GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING.
C     A CONDITION NUMBER OF A IS ESTIMATED.
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C
C        NA               ROW DIMENSION OF THE ARRAY CONTAINING A AS
C                         DECLARED IN THE CALLING PROGRAM DIMENSION
C                         STATEMENT;
C
C        N                ORDER OF THE MATRIX;
C
C        A                N X N MATRIX TO BE TRIANGULARIZED.
C
C     ON OUTPUT:
C
C        A                N X N ARRAY CONTAINING AN UPPER TRIANGULAR
C                         MATRIX U AND A PERMUTED VERSION OF A LOWER
C                         TRIANGULAR MATRIX I-L SO THAT
C                         (PERMUTATION MATRIX)*A = L*U.
C
C        COND             AN ESTIMATE OF THE CONDITION OF A FOR THE
C                         LINEAR SYSTEM
C                               A*X = B.
C                         CHANGES IN A AND B MAY CAUSE CHANGES COND
C                         TIMES AS LARGE IN X.   IF COND + 1.0D0 = COND,
C                         A IS SINGULAR TO WORKING PRECISION.  COND IS
C                         SET TO 1.0D+32 IF "EXACT" SINGULARITY IS
C                         DETECTED.
C
C        IPVT             PIVOT VECTOR OF LENGTH N.
C                         IPVT(K) = THE INDEX OF THE K-TH PIVOT ROW.
C                         IPVT(N) = (-1)**(NUMBER OF INTERCHANGES).
C
C        WORK             REAL SCRATCH VECTOR OF LENGTH N.
C                         ITS INPUT CONTENTS ARE IGNORED.  ITS OUTPUT
C                         CONTENTS ARE USUALLY UNIMPORTANT.
C
```

```
C     *****APPLICATIONS AND USAGE RESTRICTIONS:
C     DDCOMP CAN BE USED IN CONJUNCTION WITH DSOLVE TO COMPUTE SOLUTIONS
C     TO SYSTEMS OF LINEAR EQUATIONS.  IF NEAR-SINGULARITY IS
C     DETECTED SOLUTIONS ARE MORE RELIABLY COMPUTED VIA SINGULAR
C     VALUE DECOMPOSITION OF A.
C     DDCOMP CAN ALSO BE USED TO COMPUTE THE DETERMINANT OF A.
C     ON OUTPUT SIMPLY COMPUTE:
C             DET(A)  = IPVT(N)*A(1,1)*A(2,2)* . . . *A(N,N).
C
C     *****ALGORITHM NOTES:
C     DDCOMP IS A DOUBLE PRECISION ADAPTATION OF THE SUBROUTINE DECOMP
C     (SEE REFERENCE (1) FOR DETAILS).  THIS ALGORITHM IMPLEMENTS
C     GAUSSIAN ELIMINATION IN A MODERATELY UNCONVENTIONAL MANNER
C     TO PROVIDE POTENTIAL EFFICIENCY ADVANTAGES UNDER CERTAIN
C     OPERATING SYSTEMS (SEE REFERENCE (2) FOR DETAILS).
C
C     *****REFERENCES:
C     (1)   FORSYTHE,G.E., MALCOLM,M.A., AND MOLER,C.B.,  COMPUTER
C           METHODS FOR MATHEMATICAL COMPUTATIONS, PRENTICE-HALL, 1977.
C     (2)   MOLER,C.B.,  MATRIX COMPUTATIONS WITH FORTRAN AND PAGING,
C           COMM. ACM, 15(1972), 268-270.
C
C     *****HISTORY:
C     ADAPTATION AND DOCUMENTATION WRITTEN BY ALAN J. LAUB
C     (ELEC. SYS. LAB., M.I.T., RM. 35-331, CAMBRIDGE, MA 02139,
C     PH.: (617)-253-2125), AUGUST 1977.
C     MOST RECENT VERSION:  SEP. 21, 1977.
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      IPVT(N)=1
      IF (N.EQ.1) GO TO 80
      NM1=N-1
C
C     COMPUTE 1-NORM OF A
C
      ANORM=0.0D0
      DO 10 J=1,N
         T=0.0D0
         DO 5 I=1,N
              T=T+DABS(A(I,J))
5        CONTINUE
         IF (T.GT.ANORM) ANORM=T
10    CONTINUE
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
      DO 35 K=1,NM1
         KP1=K+1
C
C     FIND PIVOT
C
         M=K
         DO 15 I=KP1,N
              IF (DABS(A(I,K)).GT.DABS(A(M,K))) M=I
```

```
 15       CONTINUE
          IPVT(K)=M
          IF (M.NE.K) IPVT(N)=-IPVT(N)
          T=A(M,K)
          A(M,K)=A(K,K)
          A(K,K)=T
C
C     SKIP STEP IF PIVOT IS ZERO
C
          IF (T.EQ.0.0D0) GO TO 35
C
C     COMPUTE MULTIPLIERS
C
          DO 20 I=KP1,N
             A(I,K)=-A(I,K)/T
 20       CONTINUE
C
C     INTERCHANGE AND ELIMINATE BY COLUMNS
C
          DO 30 J=KP1,N
             T=A(M,J)
             A(M,J)=A(K,J)
             A(K,J)=T
             IF (T.EQ.0.0D0) GO TO 30
             DO 25 I=KP1,N
                A(I,J)=A(I,J)+A(I,K)*T
 25          CONTINUE
 30       CONTINUE
 35    CONTINUE
C
C     COND = (1-NORM OF A)*(AN ESTIMATE OF 1-NORM OF A-INVERSE).
C     ESTIMATE OBTAINED BY ONE STEP OF INVERSE ITERATION FOR THE
C     SMALL SINGULAR VECTOR.  THIS INVOLVES SOLVING TWO SYSTEMS
C                      T
C     OF EQUATIONS: A *Y = E  AND  A*Z = Y   WHERE E
C     IS A VECTOR OF +1 OR -1 CHOSEN TO CAUSE GROWTH IN Y.
C     ESTIMATE = (1-NORM OF Z)/(1-NORM OF Y).
C
C                T
C     SOLVE    A *Y = E
C
       DO 50 K=1,N
          T=0.0D0
          IF (K.EQ.1) GO TO 45
          KM1=K-1
          DO 40 I=1,KM1
             T=T+A(I,K)*WORK(I)
 40       CONTINUE
 45       EK=1.0D0
          IF (T.LT.0.0D0) EK=-1.0D0
          IF (A(K,K).EQ.0.0D0) GO TO 90
          WORK(K)=-(EK+T)/A(K,K)
 50    CONTINUE
       DO 60 KB=1,NM1
          K=N-KB
```

```
         T=0.0D0
         KP1=K+1
         DO 55 I=KP1,N
            T=T+A(I,K)*WORK(K)
55       CONTINUE
         WORK(K)=T
         M=IPVT(K)
         IF (M.EQ.K) GO TO 60
         T=WORK(M)
         WORK(M)=WORK(K)
         WORK(K)=T
60    CONTINUE
C
      YNORM=0.0D0
      DO 65 I=1,N
         YNORM=YNORM+DABS(WORK(I))
65    CONTINUE
C
C     SOLVE   A*Z = Y
C
      CALL DSOLVE (NA,N,A,WORK,IPVT)
C
      ZNORM=0.0D0
      DO 70 I=1,N
         ZNORM=ZNORM+DABS(WORK(I))
70    CONTINUE
C
C     ESTIMATE CONDITION
C
      COND=ANORM*ZNORM/YNORM
      IF (COND.LT.1.0D0) COND=1.0D0
      RETURN
C
C     1-BY-1 CASE
C
80    COND=1.0D0
      IF (A(1,1).NE.0.0D0) RETURN
C
C     "EXACT" SINGULARITY
C
90    COND=1.0D+32
      RETURN
C
C     LAST LINE OF DDCOMP
C
      END
```

```
      SUBROUTINE DSOLVE (NA,N,A,B,IPVT)
C
C     *****PARAMETERS:
      INTEGER NA,N,IPVT(N)
      DOUBLE PRECISION A(NA,N),B(N)
C
C     *****LOCAL VARIABLES:
      INTEGER KB,KM1,NM1,KP1,I,K,M
      DOUBLE PRECISION T
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     THIS SUBROUTINE SOLVES THE LINEAR SYSTEM    A*X = B
C     BY FORWARD ELIMINATION AND BACK SUBSTITUTION USING THE
C     TRIANGULAR FACTORS OF A PROVIDED BY DDCOMP.
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C
C        NA                 ROW DIMENSION OF THE ARRAY CONTAINING A
C                           AS DECLARED IN THE CALLING PROGRAM DIMENSION
C                           STATEMENT;
C
C        N                  ORDER OF THE MATRIX A;
C
C        A                  TRIANGULARIZED MATRIX OBTAINED FROM DDCOMP;
C
C        B                  RIGHT HAND SIDE VECTOR OF LENGTH N;
C
C        IPVT               PIVOT VECTOR OF LENGTH N OBTAINED FROM DDCOMP.
C
C     ON OUTPUT:
C
C        B                  SOLUTION VECTOR, X, OF LENGTH N.
C
C     *****APPLICATIONS AND USAGE RESTRICTIONS:
C     DSOLVE SHOULD NOT BE USED IN CASE DDCOMP HAS DETECTED NEAR-
C     SINGULARITY.  SINGULAR VALUE ANALYSIS IS THEN MORE RELIABLE.
C
C     *****ALGORITHM NOTES:
C     DSOLVE IS A DOUBLE PRECISION ADAPTATION OF THE SUBROUTINE SOLVE
C     (SEE REFERENCE (1) IN THE DDCOMP DOCUMENTATION FOR DETAILS).
C
C     *****HISTORY:
C     ADAPTATION AND DOCUMENTATION WRITTEN BY ALAN J. LAUB
C     (ELEC. SYS. LAB., M.I.T., RM. 35-331, CAMBRIDGE, MA 02139,
C     PH.: (617)-253-2125), AUGUST 1977.
C     MOST RECENT VERSION:  SEP. 21, 1977.
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     FORWARD ELIMINATION
C
      IF (N.EQ.1) GO TO 50
```

```
      NM1=N-1
      DO 20 K=1,NM1
         KP1=K+1
         M=IPVT(K)
         T=B(M)
         B(M)=B(K)
         B(K)=T
         DO 10 I=KP1,N
              B(I)=B(I)+A(I,K)*T
10       CONTINUE
20    CONTINUE
C
C     BACK SUBSTITUTION
C
      DO 40 KB=1,NM1
         KM1=N-KB
         K=KM1+1
         B(K)=B(K)/A(K,K)
         T=-B(K)
         DO 30 I=1,KM1
              B(I)=B(I)+A(I,K)*T
30       CONTINUE
40    CONTINUE
50    B(1)=B(1)/A(1,1)
      RETURN
C
C     LAST LINE OF DSOLVE
C
      END
```

```
      SUBROUTINE EXCHNG (A,V,N,L,B1,B2,EPS,FAIL,NA,NV)
C
C     *****PARAMETERS
      INTEGER B1,B2,L,NA,NV
      DOUBLE PRECISION A(NA,N),EPS,V(NV,N)
      LOGICAL FAIL
C
C     *****LOCAL VARIABLES:
      INTEGER I,IT,J,L1,M
      DOUBLE PRECISION P,Q,R,S,W,X,Y,Z
C
C     *****FUNCTIONS:
      DOUBLE PRECISION DABS,DSQRT,DMAX1
C
C     *****SUBROUTINES CALLED:
C     QRSTEP
C
C     :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     GIVEN THE UPPER HESSENBERG MATRIX A WITH CONSECUTIVE B1 X B1 AND
C     B2 X B2 DIAGONAL BLOCKS (B1, B2.LE.2) STARTING AT A(L,L), THIS
C     SUBROUTINE PRODUCES A UNITARY SIMILARITY TRANSFORMATION THAT
C     EXCHANGES THE BLOCKS ALONG WITH THEIR EIGENVALUES. THE
C     TRANSFORMATION IS ACCUMULATED IN V.
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C        NA,NV               ROW DIMENSIONS OF THE ARRAYS CONTAINING A
C                            AND V, RESPECTIVELY, AS DECLARED IN THE
C                            CALLING PROGRAM DIMENSION STATEMENT;
C
C        A                   N X N MATRIX WHOSE BLOCKS ARE TO BE
C                            INTERCHANGED;
C
C        N                   ORDER OF THE MATRIX A;
C
C        L                   POSITION OF THE BLOCKS;
C
C        B1                  AN INTEGER CONTAINING THE SIZE OF THE FIRST
C                            BLOCK;
C
C        B2                  AN INTEGER CONTAINING THE SIZE OF THE SECOND
C                            BLOCK;
C
C        EPS                 A CONVERGENCE CRITERION (CF. HQR3).
C
C     ON OUTPUT:
C
C        FAIL                A LOGICAL VARIABLE WHICH IS .FALSE. ON A
C                            NORMAL RETURN. IF THIRTY ITERATIONS WERE
C                            PERFORMED WITHOUT CONVERGENCE, FAIL IS SET TO
C                            .TRUE. AND THE ELEMENT A(L+B2,L+B2-1) CANNOT
C                            BE ASSUMED ZERO.
C
```

```
C       *****HISTORY:
C       DOCUMENTED BY J.A.K. CARRIG (ELEC. SYS. LAB., M.I.T., RM. 35-307,
C       CAMBRIDGE, MA 02139, PH.: (617) - 253-2165, SEPTEMBER 1978.
C       MOST RECENT VERSION: SEPT. 21, 1978.
C
C       :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
        FAIL=.FALSE.
        IF (B1.EQ.2) GO TO 70
        IF (B2.EQ.2) GO TO 40
        L1=L+1
        Q=A(L+1,L+1)-A(L,L)
        P=A(L,L+1)
        R=DMAX1(P,Q)
        IF (R.EQ.0.0D0) RETURN
        P=P/R
        Q=Q/R
        R=DSQRT(P**2+Q**2)
        P=P/R
        Q=Q/R
        DO 10 J=L,N
           S=P*A(L,J)+Q*A(L+1,J)
           A(L+1,J)=P*A(L+1,J)-Q*A(L,J)
           A(L,J)=S
10      CONTINUE
        DO 20 I=1,L1
           S=P*A(I,L)+Q*A(I,L+1)
           A(I,L+1)=P*A(I,L+1)-Q*A(I,L)
           A(I,L)=S
20      CONTINUE
        DO 30 I=1,N
           S=P*V(I,L)+Q*V(I,L+1)
           V(I,L+1)=P*V(I,L+1)-Q*V(I,L)
           V(I,L)=S
30      CONTINUE
        A(L+1,L)=0.0D0
        RETURN
40      CONTINUE
        X=A(L,L)
        P=1.0D0
        Q=1.0D0
        R=1.0D0
        CALL QRSTEP (A,V,P,Q,R,L,L+2,N,NA,NV)
        IT=0
50      IT=IT+1
        IF (IT.LE.30) GO TO 60
        FAIL=.TRUE.
        RETURN
60      CONTINUE
        P=A(L,L)-X
        Q=A(L+1,L)
        R=0.0D0
        CALL QRSTEP (A,V,P,Q,R,L,L+2,N,NA,NV)
        IF (DABS(A(L+2,L+1)).GT.EPS*(DABS(A(L+1,L+1))+DABS(A(L+2,L+2))))
       +    GO TO 50
```

```
      A(L+2,L+1)=0.0D0
      RETURN
70    CONTINUE
      M=L+2
      IF (B2.EQ.2) M=M+1
      X=A(L+1,L+1)
      Y=A(L,L)
      W=A(L+1,L)*A(L,L+1)
      P=1.0D0
      Q=1.0D0
      R=1.0D0
      CALL QRSTEP (A,V,P,Q,R,L,M,N,NA,NV)
      IT=0
80    IT=IT+1
      IF (IT.LE.30) GO TO 90
      FAIL=.TRUE.
      RETURN
90    CONTINUE
      Z=A(L,L)
      R=X-Z
      S=Y-Z
      P=(R*S-W)/A(L+1,L)+A(L,L+1)
      Q=A(L+1,L+1)-Z-R-S
      R=A(L+2,L+1)
      S=DABS(P)+DABS(Q)+DABS(R)
      P=P/S
      Q=Q/S
      R=R/S
      CALL QRSTEP (A,V,P,Q,R,L,M,N,NA,NV)
      IF (DABS(A(M-1,M-2)).GT.EPS*(DABS(A(M-1,M-1))+DABS(A(M-2,M-2))))
     +    GO TO 80
      A(M-1,M-2)=0.0D0
      RETURN
C
C     LAST LINE OF EXCHNG
C
      END
```

```
      SUBROUTINE HQR3 (A,V,N,NLOW,NUP,EPS,ER,EI,ITYPE,NA,NV)
C
C     *****PARAMETERS:
      INTEGER N,NA,NLOW,NUP,NV,ITYPE(N)
      DOUBLE PRECISION A(NA,N),EI(N),ER(N),EPS,V(NV,N)
C
C     *****LOCAL VARIABLES:
      LOGICAL FAIL
      INTEGER I,IT,L,MU,NL,NU
      DOUBLE PRECISION E1,E2,P,Q,R,S,T,W,X,Y,Z
C
C     *****FUNCTIONS:
      DOUBLE PRECISION DABS
C
C     *****SUBROUTINES CALLED:
C     EXCHNG,QRSTEP,SPLIT
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     THIS SUBROUTINE REDUCES THE UPPER HESSENBERG MATRIX A TO QUASI-
C     TRIANGULAR FORM BY UNITARY SIMILARITY TRANSFORMATIONS. THE
C     EIGENVALUES OF A, WHICH ARE CONTAINED IN THE 1 X 1 AND 2 X 2
C     DIAGONAL BLOCKS OF THE REDUCED MATRIX, ARE ORDERED IN DESCENDING
C     ORDER OF MAGNITUDE ALONG THE DIAGONAL. THE TRANSFORMATIONS ARE
C     ACCUMULATED IN THE ARRAY V.
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C        NA,NV               ROW DIMENSIONS OF THE ARRAYS CONTAINING A AND
C                            V, RESPECTIVELY, AS DECLARED IN THE CALLING
C                            PROGRAM DIMENSION STATEMENT;
C
C        A                   N X N ARRAY CONTAINING THE UPPER HESSENBERG
C                            MATRIX TO BE REDUCED;
C
C        N                   ORDER OF THE MATRICES A AND V;
C
C        NLOW,NUP            A(NLOW,NLOW-1) AND A(NUP,1+NUP) ARE ASSUMED
C                            TO BE ZERO, AND ONLY ROWS NLOW THROUGH NUP
C                            AND COLUMNS NLOW THROUGH NUP ARE TRANSFORMED,
C                            RESULTING IN THE CALCULATION OF EIGENVALUES
C                            NLOW THROUGH NUP;
C
C        EPS                 A CONVERGENCE CRITERION USED TO DETERMINE WHEN
C                            A SUBDIAGONAL ELEMENT OF A IS NEGLIGIBLE.
C                            SPECIFICALLY, A(I+1,I) IS REGARDED AS
C                            NEGLIGIBLE IF DABS(A(I+1),I)).LE.EPS*
C                            (DABS(A(I+1,I+1))). THIS MEANS THAT THE FINAL
C                            MATRIX RETURNED BY THE PROGRAM WILL BE EXACTLY
C                            SIMILAR TO A + E WHERE E IS OF ORDER
C                            EPS*NORM(A), FOR ANY REASONABLY BALANCED NORM
C                            SUCH AS THE ROW-SUM NORM;
C
C        ITYPE               AN INTEGER VECTOR OF LENGTH N WHOSE
```

```
C                       I-TH ENTRY IS
C                       0  IF THE I-TH EIGENVALUE IS REAL,
C                       1  IF THE I-TH EIGENVALUE IS COMPLEX WITH
C                          POSITIVE IMAGINARY PART,
C                       2  IF THE I-TH EIGENVALUE IS COMPLEX WITH
C                          NEGATIVE IMAGINARY PART,
C                      -1  IF THE I-TH EIGENVALUE WAS NOT CALCULATED
C                          SUCCESSFULLY.
C
C     ON OUTPUT:
C
C     A                 N X N ARRAY CONTAINING THE REDUCED, QUASI-
C                       TRIANGULAR MATRIX;
C
C     V                 N X N ARRAY CONTAINING THE REDUCING
C                       TRANSFORMATIONS TO BE MULTIPLIED;
C
C     ER,EI             REAL SCRATCH VECTORS OF LENGTH N WHICH ON
C                       RETURN CONTAIN THE REAL AND IMAGINARY PARTS,
C                       RESPECTIVELY, OF THE EIGENVALUES.
C
C    *****HISTORY:
C    DOCUMENTED BY J.A.K. CARRIG, (ELEC. SYS. LAB., M.I.T., RM. 35-307,
C    CAMBRIDGE, MA 02139, PH.: (617)- 253-2165), SEPT 1978.
C    MOST RECENT VERSION: SEPT 21, 1978.
C
C    ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      DO 10 I=NLOW,NUP
         ITYPE(I)=-1
10    CONTINUE
      T=0.0D0
      NU=NUP
20    IF (NU.LT.NLOW) GO TO 240
      IT=0
30    CONTINUE
      L=NU
40    CONTINUE
      IF (L.EQ.NLOW) GO TO 50
      IF (DABS(A(L,L-1)).LT.EPS*(DABS(A(L-1,L-1))+DABS(A(L,L))))
     +     GO TO 50
      L=L-1
      GO TO 40
50    CONTINUE
      X=A(NU,NU)
      IF (L.EQ.NU) GO TO 160
      Y=A(NU-1,NU-1)
      W=A(NU,NU-1)*A(NU-1,NU)
      IF (L.EQ.NU-1) GO TO 100
      IF (IT.EQ.30) GO TO 240
      IF (IT.NE.10 .AND. IT.NE.20) GO TO 70
      T=T+X
      DO 60 I=NLOW,NU
         A(I,I)=A(I,I)-X
60    CONTINUE
```

```
      S=DABS(A(NU,NU-1))+DABS(A(NU-1,NU-2))
      X=0.75D0*S
      Y=X
      W=-0.4375D0*S**2
70    CONTINUE
      IT=IT+1
      NL=NU-2
80    CONTINUE
      Z=A(NL,NL)
      R=X-Z
      S=Y-Z
      P=(R*S-W)/A(NL+1,NL)+A(NL,NL+1)
      Q=A(NL+1,NL+1)-Z-R-S
      R=A(NL+2,NL+1)
      S=DABS(P)+DABS(Q)+DABS(R)
      P=P/S
      Q=Q/S
      R=R/S
      IF (NL.EQ.L) GO TO 90
      IF (DABS(A(NL,NL-1))*(DABS(Q)+DABS(R)).LE.EPS*DABS(P)*
     +    (DABS(A(NL-1,NL-1))+DABS(Z)+DABS(A(NL+1,NL+1)))) GO TO 90
      NL=NL-1
      GO TO 80
90    CONTINUE
      CALL QRSTEP (A,V,P,Q,R,NL,NU,N,NA,NV)
      GO TO 30
100   IF (NU.NE.NLOW+1) A(NU-1,NU-2)=0.0D0
      A(NU,NU)=A(NU,NU)+T
      A(NU-1,NU-1)=A(NU-1,NU-1)+T
      ITYPE(NU)=0
      ITYPE(NU-1)=0
      MU=NU
110   CONTINUE
      NL=MU-1
      CALL SPLIT (A,V,N,NL,E1,E2,NA,NV)
      IF (A(MU,MU-1).EQ.0.0D0) GO TO 170
      IF (MU.EQ.NUP) GO TO 230
      IF (MU.EQ.NUP-1) GO TO 130
      IF (A(MU+2,MU+1).EQ.0.0D0) GO TO 130
      IF (A(MU-1,MU-1)*A(MU,MU)-A(MU-1,MU)*A(MU,MU-1).GE.A(MU+1,MU+1)*
     +   A(MU+2,MU+2)-A(MU+1,MU+2)*A(MU+2,MU+1)) GO TO 230
      CALL EXCHNG (A,V,N,NL,2,2,EPS,FAIL,NA,NV)
      IF (.NOT.FAIL) GO TO 120
      ITYPE(NL)=-1
      ITYPE(NL+1)=-1
      ITYPE(NL+2)=-1
      ITYPE(NL+3)=-1
      GO TO 240
120   CONTINUE
      MU=MU+2
      GO TO 150
130   CONTINUE
      IF (A(MU-1,MU-1)*A(MU,MU)-A(MU-1,MU)*A(MU,MU-1).GE.
     +   A(MU+1,MU+1)**2) GO TO 230
      CALL EXCHNG (A,V,N,NL,2,1,EPS,FAIL,NA,NV)
```

```
      IF (.NOT.FAIL) GO TO 140
      ITYPE(NL)=-1
      ITYPE(NL+1)=-1
      ITYPE(NL+2)=-1
      GO TO 240
140   CONTINUE
      MU=MU+1
150   CONTINUE
      GO TO 110
160   NL=0
      A(NU,NU)=A(NU,NU)+T
      IF (NU.NE.NLOW) A(NU,NU-1)=0.0D0
      ITYPE(NU)=0
      MU=NU
170   CONTINUE
180   CONTINUE
      IF (MU.EQ.NUP) GO TO 220
      IF (MU.EQ.NUP-1) GO TO 200
      IF (A(MU+2,MU+1).EQ.0.0D0) GO TO 200
      IF (A(MU,MU)**2.GE.A(MU+1,MU+1)*A(MU+2,MU+2)-A(MU+1,MU+2)*
     +    A(MU+2,MU+1)) GO TO 230
      CALL EXCHNG (A,V,N,MU,1,2,EPS,FAIL,NA,NV)
      IF (.NOT.FAIL) GO TO 190
      ITYPE(MU)=-1
      ITYPE(MU+1)=-1
      ITYPE(MU+2)=-1
      GO TO 240
190   CONTINUE
      MU=MU+2
      GO TO 210
200   CONTINUE
      IF (DABS(A(MU,MU)).GE.DABS(A(MU+1,MU+1))) GO TO 220
      CALL EXCHNG (A,V,N,MU,1,1,EPS,FAIL,NA,NV)
      MU=MU+1
210   CONTINUE
      GO TO 180
220   CONTINUE
      MU=NL
      NL=0
      IF (MU.NE.0) GO TO 170
230   CONTINUE
      NU=L-1
      GO TO 20
240   IF (NU.LT.NLOW) GO TO 260
      DO 250 I=1,NU
         A(I,I)=A(I,I)+T
250   CONTINUE
260   CONTINUE
      NU=NUP
270   CONTINUE
      IF (ITYPE(NU).NE.-1) GO TO 280
      NU=NU-1
      GO TO 310
280   CONTINUE
      IF (NU.EQ.NLOW) GO TO 290
```

```
      IF (A(NU,NU-1).EQ.0.0D0) GO TO 290
      CALL SPLIT (A,V,N,NU-1,E1,E2,NA,NV)
      IF (A(NU,NU-1).EQ.0.0D0) GO TO 290
      ER(NU)=E1
      EI(NU-1)=E2
      ER(NU-1)=ER(NU)
      EI(NU)=-EI(NU-1)
      ITYPE(NU-1)=1
      ITYPE(NU)=2
      NU=NU-2
      GO TO 300
290   CONTINUE
      ER(NU)=A(NU,NU)
      EI(NU)=0.0D0
      NU=NU-1
300   CONTINUE
310   CONTINUE
      IF (NU.GE.NLOW) GO TO 270
      RETURN
I
C
C     LAST LINE OF HQR3
C
      END
```

```
      SUBROUTINE MLINEQ (NA,NB,N,M,A,B,COND,IPVT,WORK)
C
C     *****PARAMETERS:
      INTEGER NA,NB,N,M,IPVT(N)
      DOUBLE PRECISION A(NA,N),B(NB,M),COND,WORK(N)
C
C     *****LOCAL VARIABLES:
      INTEGER I,J,KIN,KOUT
      DOUBLE PRECISION CONDP1
C
C     *****SUBROUTINES CALLED:
C     DDCOMP,DSOLVE
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     THIS SUBROUTINE SOLVES THE MATRIX LINEAR EQUATION
C                               A*X = B
C     WHERE A IS AN N X N (INVERTIBLE) MATRIX AND B IS AN N X M
C     MATRIX.  SUBROUTINE DDCOMP IS CALLED ONCE FOR THE LU-DECOMP-
C     OSITION OF A AND SUBROUTINE DSOLVE IS CALLED M TIMES FOR
C     FORWARD ELIMINATION AND BACK SUBSTITUTION TO PRODUCE THE
C     M COLUMNS OF THE SOLUTION MATRIX X = (A-INVERSE)*B.  AN
C     ESTIMATE OF THE CONDITION OF A IS RETURNED.  SHOULD A BE
C     SINGULAR TO WORKING ACCURACY, A MESSAGE TO THAT EFFECT IS
C     PRODUCED.
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C
C        NA,NB                ROW DIMENSIONS OF THE ARRAYS CONTAINING A AND
C                             B, RESPECTIVELY, AS DECLARED IN THE CALLING
C                             PROGRAM DIMENSION STATEMENT;
C
C        N                    ORDER OF THE MATRIX A AND NUMBER OF ROWS OF
C                             THE MATRIX B;
C
C        M                    NUMBER OF COLUMNS OF THE MATRIX B;
C
C        A                    N X N COEFFICIENT MATRIX;
C
C        B                    N X M RIGHT HAND SIDE MATRIX.
C
C     ON OUTPUT:
C
C        B                    SOLUTION MATRIX  X = (A-INVERSE)*B;
C
C        COND                 AN ESTIMATE OF THE CONDITION OF A;
C
C        IPVT                 PIVOT VECTOR OF LENGTH N (SEE DDCOMP
C                             DOCUMENTATION);
C
C        WORK                 A REAL SCRATCH VECTOR OF LENGTH N.
C
C     *****APPLICATIONS AND USAGE RESTRICTIONS:
```

```
C     (1)THE VALUE OF COND SHOULD ALWAYS BE CHECKED BY THE CALLING
C        PROGRAM.  SHOULD A BE NEAR-SINGULAR (OR SINGULAR TO WORKING
C        ACCURACY) THE DATA SHOULD BE INVESTIGATED FOR POSSIBLE
C        ERRORS.  IF THERE ARE NONE AND THE PROBLEM IS APPARENTLY
C        WELL-POSED AND/OR MEANINGFUL, SINGULAR VALUE ANALYSIS IS
C        THEN A MORE RELIABLE SOLUTION TECHNIQUE (CF. EISPACK
C        SUBROUTINES  SVD  AND  MINFIT).
C     (2)MLINEQ CAN BE USED TO COMPUTE THE INVERSE OF A:  SIMPLY SOLVE
C        A*X = I WHERE  I  IS THE N X N IDENTITY MATRIX.
C     (3)IF THE SOLUTION TO X*A = B  (X = B*(A-INVERSE)) IS DESIRED,
C        SIMPLY TRANSPOSE THE SOLUTION OF
C         T     T
C        A *X = B .
C
C     *****ALGORITHM NOTES:
C     THE CONTENTS OF A ARE MODIFIED BY THIS SUBROUTINE.  SHOULD THE
C     ORIGINAL COEFFICIENTS OF A BE NEEDED SUBSEQUENTLY, THE
C     CONTENTS OF A SHOULD BE SAVED PRIOR TO THE CALL TO MLINEQ.
C
C     *****HISTORY:
C     WRITTEN BY ALAN J. LAUB (ELEC. SYS. LAB., M.I.T., RM. 35-331,
C     CAMBRIDGE, MA 02139,  PH.: (617)-253-2125), AUGUST 1977.
C     MOST RECENT VERSION:  SEP. 21, 1977.
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      COMMON/INOU/KIN,KOUT
      CALL DDCOMP (NA,N,A,COND,IPVT,WORK)
      CONDP1=COND+1.0D0
      IF (CONDP1.GT.COND) GO TO 100
      WRITE (KOUT,44400)
44400 FORMAT (40H1MATRIX IS SINGULAR TO WORKING PRECISION)
      RETURN
100   DO 400 J=1,M
         DO 200 I=1,N
            WORK(I)=B(I,J)
200      CONTINUE
C
C     COMPUTE  (J-TH COLUMN OF X) = (A-INVERSE)*(J-TH COLUMN OF B)
C
         CALL DSOLVE (NA,N,A,WORK,IPVT)
         DO 300 I=1,N
            B(I,J)=WORK(I)
300      CONTINUE
400   CONTINUE
      RETURN
C
C     LAST LINE OF MLINEQ
C
      END
```

```
      SUBROUTINE MULWOA (NA,NB,N,A,B,WORK)
C
C     *****PARAMETERS:
      INTEGER NA,NB,N
      DOUBLE PRECISION A(NA,N),B(NB,N),WORK(N)
C
C     *****LOCAL VARIABLES:
      INTEGER I,J,K
C
C     *****SUBROUTINES CALLED:
C     NONE
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     THIS SUBROUTINE OVERWRITES THE ARRAY A WITH THE MATRIX PRODUCT
C        A*B.  BOTH A AND B ARE N X N ARRAYS AND MUST BE DISTINCT.
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C
C        NA,NB                ROW DIMENSIONS OF THE ARRAYS CONTAINING A AND
C                             B, RESPECTIVELY, AS DECLARED IN THE CALLING
C                             PROGRAM DIMENSION STATEMENT;
C
C        N                    ORDER OF THE MATRICES A AND B;
C
C        A                    AN N X N MATRIX;
C
C        B                    AN N X N MATRIX.
C
C     ON OUTPUT:
C
C        A                    AN N X N ARRAY CONTAINING A*B;
C
C        WORK                 A REAL SCRATCH VECTOR OF LENGTH N.
C
C     *****HISTORY:
C     WRITTEN BY ALAN J. LAUB (ELEC. SYS. LAB., M.I.T., RM. 35-331,
C     CAMBRIDGE, MA 02139,  PH.: (617)-253-2125), SEPTEMBER 1977.
C     MOST RECENT VERSION: SEP. 21, 1977.
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      DO 40 I=1,N
         DO 20 J=1,N
            WORK(J)=0.0D0
            DO 10 K=1,N
               WORK(J)=WORK(J)+A(I,K)*B(K,J)
10          CONTINUE
20       CONTINUE
         DO 30 J=1,N
            A(I,J)=WORK(J)
30       CONTINUE
40    CONTINUE
```

```
      RETURN                                                        MUL00560
C                                                                   MUL00570
C     LAST LINE OF MULWOA                                           MUL00580
C                                                                   MUL00590
      END                                                           MUL00600
```

```
      SUBROUTINE MULWOB (NA,NB,N,A,B,WORK)
C
C     *****PARAMETERS:
      INTEGER NA,NB,N
      DOUBLE PRECISION A(NA,N),B(NB,N),WORK(N)
C
C     *****LOCAL VARIABLES:
      INTEGER I,J,K
C
C     *****SUBROUTINES CALLED:
C     NONE
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     THIS SUBROUTINE OVERWRITES THE ARRAY B WITH THE MATRIX PRODUCT
C        A*B.  BOTH A AND B ARE N X N ARRAYS AND MUST BE DISTINCT.
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C
C        NA,NB              ROW DIMENSIONS OF THE ARRAYS CONTAINING A AND
C                           B, RESPECTIVELY, AS DECLARED IN THE CALLING
C                           PROGRAM DIMENSION STATEMENT;
C
C        N                  ORDER OF THE MATRICES A AND B;
C
C        A                  AN N X N MATRIX;
C
C        B                  AN N X N MATRIX.
C
C     ON OUTPUT:
C
C        B                  AN N X N ARRAY CONTAINING A*B;
C
C        WORK               A REAL SCRATCH VECTOR OF LENGTH N.
C
C     *****HISTORY:
C     WRITTEN BY ALAN J. LAUB (ELEC. SYS. LAB., M.I.T., RM. 35-331,
C     CAMBRIDGE, MA 02139,  PH.: (617)-253-2125), SEPTEMBER 1977.
C     MOST RECENT VERSION: SEP. 21, 1977.
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      DO 50 J=1,N
         DO 10 I=1,N
            WORK(I)=0.0D0
   10    CONTINUE
         DO 30 K=1,N
            DO 20 I=1,N
               WORK(I)=WORK(I)+A(I,K)*B(K,J)
   20       CONTINUE
   30    CONTINUE
         DO 40 I=1,N
            B(I,J)=WORK(I)
```

```
40         CONTINUE
50      CONTINUE
        RETURN
C
C       LAST LINE OF MULWOB
C
        END
```

```
      SUBROUTINE QRSTEP (A,V,P,Q,R,NL,NU,N,NA,NV)
C
C     *****PARAMETERS:
      INTEGER N,NA,NL,NU,NV
      DOUBLE PRECISION A(NA,N),P,Q,R,V(NV,N)
C
C     *****LOCAL VARIABLES:
      LOGICAL LAST
      INTEGER I,J,K,NL2,NL3,NUM1
      DOUBLE PRECISION S,X,Y,Z
C
C     *****FUNCTIONS:
      DOUBLE PRECISION DABS,DSQRT
C
C     *****SUBROUTINES CALLED:
C     NONE
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     THIS SUBROUTINE PERFORMS ONE IMPLICIT QR STEP ON THE UPPER
C     HESSENBERG MATRIX A. THE SHIFT IS DETERMINED BY THE NUMBERS P,Q,
C     AND R, AND THE STEP IS APPLIED TO ROWS AND COLUMNS NL THROUTH NU.
C     THE TRANSFORMATIONS ARE ACCUMULATED IN THE ARRAY V.
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C        NA,NV              ROW DIMENSIONS OF THE ARRAYS CONTAINING A
C                           AND V, RESPECTIVELY, AS DECLARED IN THE
C                           CALLING PROGRAM DIMENSION STATEMENT;
C
C        A                  N X N UPPER HESSENBERG MATRIX ON WHICH THE QR
C                           STEP IS TO BE PERFORMED;
C
C        P,Q,R              PARAMETERS WHICH DETERMINE THE SHIFT;
C
C        NL                 THE LOWER LIMIT OF THE STEP;
C
C        NU                 THE UPPER LIMIT OF THE STEP;
C
C        N                  ORDER OF THE MATRIX A.
C
C     ON OUTPUT:
C
C        V                  N X N REAL SCRATCH ARRAY CONTAINING THE
C                           ACCUMULATED TRANSFORMATIONS.
C
C     *****HISTORY:
C     DOCUMENTED BY J.A.K. CARRIG (ELEC. SYS. LAB., M.I.T., RM. 35-307,
C     CAMBRIDGE, MA 02139, PH.: (617) - 253-2165), SEPTEMBER 1978.
C     MOST RECENT VERSION: SEPT 21, 1978.
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      NL2=NL+2
```

```
      DO 10 I=NL2,NU
         A(I,I-2)=0.0D0
10    CONTINUE
      IF (NL2.EQ.NU) GO TO 30
      NL3=NL+3
      DO 20 I=NL3,NU
         A(I,I-3)=0.0D0
20    CONTINUE
30    CONTINUE
      NUM1=NU-1
      DO 130 K=NL,NUM1
         LAST=K.EQ.NUM1
         IF (K.EQ.NL) GO TO 40
         P=A(K,K-1)
         Q=A(K+1,K-1)
         R=0.0D0
         IF (.NOT.LAST) R=A(K+2,K-1)
         X=DABS(P)+DABS(Q)+DABS(R)
         IF (X.EQ.0.0D0) GO TO 130
         P=P/X
         Q=Q/X
         R=R/X
40       CONTINUE
         S=DSQRT(P**2+Q**2+R**2)
         IF (P.LT.0.0D0) S=-S
         IF (K.EQ.NL) GO TO 50
         A(K,K-1)=-S*X
         GO TO 60
50       CONTINUE
         IF (NL.NE.1) A(K,K-1)=-A(K,K-1)
60       CONTINUE
         P=P+S
         X=P/S
         Y=Q/S
         Z=R/S
         Q=Q/P
         R=R/P
         DO 80 J=K,N
               P=A(K,J)+Q*A(K+1,J)
               IF (LAST) GO TO 70
               P=P+R*A(K+2,J)
               A(K+2,J)=A(K+2,J)-P*Z
70             CONTINUE
               A(K+1,J)=A(K+1,J)-P*Y
               A(K,J)=A(K,J)-P*X
80       CONTINUE
         J=MIN0(K+3,NU)
         DO 100 I=1,J
               P=X*A(I,K)+Y*A(I,K+1)
               IF (LAST) GO TO 90
               P=P+Z*A(I,K+2)
               A(I,K+2)=A(I,K+2)-P*R
90             CONTINUE
               A(I,K+1)=A(I,K+1)-P*Q
               A(I,K)=A(I,K)-P
```

```
100      CONTINUE
         DO 120 I=1,N
            P=X*V(I,K)+Y*V(I,K+1)
            IF (LAST) GO TO 110
            P=P+Z*V(I,K+2)
            V(I,K+2)=V(I,K+2)-P*R
110         CONTINUE
            V(I,K+1)=V(I,K+1)-P*Q
            V(I,K)=V(I,K)-P
120      CONTINUE
130   CONTINUE
      RETURN
C
C     LAST LINE OF QRSTEP
C
      END
```

```
      SUBROUTINE SPLIT (A,V,N,L,E1,E2,NA,NV)
C
C     *****PARAMETERS:
      INTEGER L,N,NA,NV
      DOUBLE PRECISION A(NA,N),V(NV,N),E1,E2
C
C     *****LOCAL VARIABLES:
      INTEGER I,J,L1
      DOUBLE PRECISION P,Q,R,T,U,W,X,Y,Z
C
C     *****FUNCTIONS:
      DOUBLE PRECISION DABS,DSQRT
C
C     ****SUBROUTINES CALLED:
C     NONE
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
C     *****PURPOSE:
C     GIVEN THE UPPER-HESSENBERG MATRIX A WITH A 2 X 2 BLOCK STARTING AT
C     A(L,L), THIS PROGRAM DETERMINES IF THE CORRESPONDING EIGENVALUES
C     ARE REAL OR COMPLEX. IF THEY ARE REAL, A ROTATION IS DETERMINED
C     THAT REDUCES THE BLOCK TO UPPER-TRIANGULAR FORM WITH THE
C     EIGENVALUE OF LARGEST ABSOLUTE VALUE APPEARING FIRST. THE
C     ROTATION IS ACCUMULATED IN THE ARRAY V.
C
C     *****PARAMETER DESCRIPTION:
C     ON INPUT:
C        NA,NV              ROW DIMENSIONS OF THE ARRAYS CONTAINING
C                           A AND V, RESPECTIVELY, AS DECLARED IN THE
C                           CALLING PROGRAM DIMENSION STATEMENT;
C
C        A                  THE UPPER HESSENBERG MATRIX WHOSE 2 X 2 BLOCK
C                           IS TO BE SPLIT;
C
C        N                  ORDER OF THE MATRIX A;
C
C        L                  POSITION OF THE 2 X 2 BLOCK.
C
C     ON OUTPUT:
C
C        V                  AN N X N ARRAY CONTAINING THE ACCUMULATED
C                           SPLITTING TRANSFORMATION;
C
C        E1,E2              REAL SCALARS. IF THE EIGENVALUES ARE COMPLEX,
C                           E1 AND E2 CONTAIN THEIR COMMON REAL PART AND
C                           POSITIVE IMAGINARY PART (RESPECTIVELY).
C                           IF THE EIGENVALUES ARE REAL, E1 CONTAINS THE
C                           ONE LARGEST IN ABSOLUTE VALUE AND E2 CONTAINS
C                           THE OTHER ONE.
C
C     *****HISTORY:
C     DOCUMENTED BY J.A.K. CARRIG (ELEC. SYS. LAB., M.I.T., R. 35-307,
C     CAMBRIDGE, MA 02139, PH.: (617) - 253-2165), SEPT 1978.
C     MOST RECENT VERSION: SEPT 21, 1978.
```

```
C
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C
      X=A(L+1,L+1)
      Y=A(L,L)
      W=A(L,L+1)*A(L+1,L)
      P=(Y-X)/2.0D0
      Q=P**2+W
      IF (Q.GE.0.0D0) GO TO 10
      E1=P+X
      E2=DSQRT(-Q)
      RETURN
 10   CONTINUE
      Z=DSQRT(Q)
      IF (P.LT.0.0D0) GO TO 20
      Z=P+Z
      GO TO 30
 20   CONTINUE
      Z=P-Z
 30   CONTINUE
      IF (Z.EQ.0.0D0) GO TO 40
      R=-W/Z
      GO TO 50
 40   CONTINUE
      R=0.0D0
 50   CONTINUE
      IF (DABS(X+Z).GE.DABS(X+R)) Z=R
      Y=Y-X-Z
      X=-Z
      T=A(L,L+1)
      U=A(L+1,L)
      IF (DABS(Y)+DABS(U).LE.DABS(T)+DABS(X)) GO TO 60
      Q=U
      P=Y
      GO TO 70
 60   CONTINUE
      Q=X
      P=T
 70   CONTINUE
      R=DSQRT(P**2+Q**2)
      IF (R.GT.0.0D0) GO TO 80
      E1=A(L,L)
      E2=A(L+1,L+1)
      A(L+1,L)=0.0D0
      RETURN
 80   CONTINUE
      P=P/R
      Q=Q/R
      DO 90 J=L,N
          Z=A(L,J)
          A(L,J)=P*Z+Q*A(L+1,J)
          A(L+1,J)=P*A(L+1,J)-Q*Z
 90   CONTINUE
      L1=L+1
      DO 100 I=1,L1
```

```
          Z=A(I,L)                                                       SPL01110
          A(I,L)=P*Z+Q*A(I,L+1)                                          SPL01120
          A(I,L+1)=P*A(I,L+1)-Q*Z                                        SPL01130
  100   CONTINUE                                                         SPL01140
        DO 110 I=1,N                                                     SPL01150
          Z=V(I,L)                                                       SPL01160
          V(I,L)=P*Z+Q*V(I,L+1)                                          SPL01170
          V(I,L+1)=P*V(I,L+1)-Q*Z                                        SPL01180
  110   CONTINUE                                                         SPL01190
        A(L+1,L)=0.0D0                                                   SPL01200
        E1=A(L,L)                                                        SPL01210
        E2=A(L+1,L+1)                                                    SPL01220
        RETURN                                                           SPL01230
C                                                                        SPL01240
C       LAST LINE OF SPLIT                                               SPL01250
C                                                                        SPL01260
        END                                                              SPL01270
```