A HESSENBERG-SCHUR METHOD FOR THE PROBLEM $AX + XB = C$

by

Gene Golub
Stephen Nash
Charles Van Loan

TR78-354

Department of Computer Science
Cornell University
Ithaca, NY 14853

A HESSENBERG-SCHUR METHOD FOR THE PROBLEM $AX + XB = C$

G.H.Golub, S.Nash, and C.Van Loan

Abstract:

One of the most effective methods for solving the matrix equation $AX + XB = C$ is the Bartels-Stewart algorithm. Key to this technique is the orthogonal reduction of A and B to triangular form using the QR algorithm for eigenvalues. A new method is proposed which differs from the Bartels-Stewart algorithm in that A is only reduced to Hessenberg form. The resulting algorithm is between 30 and 70 percent faster depending upon the dimensions of the matrices A and B. The stability of the new method is demonstrated through a roundoff error analysis and supported by numerical tests. Finally, it is shown how the techniques described can be applied and generalized to other matrix equation problems.

uthors' Addresses:

Professor Gene Golub
Department of Computer Science
Serra House, Stanford University
Stanford, California 94305

Mr. Stephen Nash
Department of Computer Science
Serra House, Stanford University
Stanford, California 94305

Professor Charles Van Loan
Department of Computer Science
Upson Hall, Cornell University
Ithaca, New York 14853

1. Introduction

Let $A \in R^{m\times m}$ and $B \in R^{n\times n}$ be given matrices and define the linear transformation $\phi: R^{m\times n} \to R^{m\times n}$ by

$$\phi(X) = AX + XB \tag{1.1}$$

This linear transformation is nonsingular if and only if A and $-B$ have no eigenvalues in common which we shall hereafter assume. Linear equations of the form

$$\phi(X) = AX + XB = C \tag{1.2}$$

arise in many problem areas and numerous algorithms have been proposed [4,10]. Among them, the Bartels-Stewart algorithm [1] has enjoyed considerable success [2]. In this paper we discuss a modification of their technique which is just as accurate and considerably faster.

This new method is called the "Hessenberg-Schur Algorithm" and like the Bartels-Stewart Algorithm is an example of a "transformation method". Such methods are based upon the equivalence of the problems

$$AX + XB = C$$

and

$$(U^{-1}AU)(U^{-1}XV) + (U^{-1}XV)(V^{-1}BV) = U^{-1}CV$$

and generally involve the following four steps:

Step 1.

Transform A and B into "simple" form via the similarity transformations $A_1 = U^{-1}A\,U$ and $B_1 = V^{-1}B\,V$.

Step 2.

Solve $UF = CV$ for F.

Step 3.

Solve the transformed system $A_1Y + YB_1 = F$ for Y.

Step 4.

Solve $XV = UY$ for X.

A brief look at the effect of roundoff error in Steps 2 and 4 serves as a nice introduction to both the Bartels-Stewart and Hessenberg-Schur algorithms.

In these steps linear systems are solved which involve the transformation matrices U and V. Suppose Gaussian elimination with pivoting is used for this purpose and that the computations are performed on a computer whose floating point numbers have t base β digits in the mantissa. Using the standard Wilkinson inverse error analysis [3,12] it follows that relative errors of order $u[\kappa_2(U) + \kappa_2(V)]$ can be expected to contaminate the computed solution $\hat{X}$ where

$$u = \beta^{-t}$$

is the machine precision and $\kappa_2(\cdot)$ is defined by

$$\kappa_2(W) = \| W \|_2 \, \| W^{-1} \|_2 = \max_{x \neq 0} \sqrt{\frac{(Wx)^T (Wx)}{x^T x}} \cdot \max_{y \neq 0} \sqrt{\frac{y^T}{(Wy)}}$$

When $\kappa_2(W)$ is large with respect to u, then we say that W is "ill-conditioned".

Unfortunately, several of the possible reductions in Step 1 can lead to ill-conditioned U and V matrices. For example, if A and B are diagonalizable, then there exist U and V so

$$U^{-1}AU = \mathrm{diag}(\alpha_1, \alpha_2, \ldots, \alpha_m) = A_1$$

$$V^{-1}BV = \mathrm{diag}(\beta_1, \beta_2, \ldots, \beta_m) = B_1$$

The matrix $Y = (y_{ij})$ in Step 3 is then prescribed by the simple formulae $y_{ij} = f_{ij}/(\alpha_i + \beta_j)$. If we apply this algorithm to the problem

$$A = \begin{bmatrix} 1.234567891 & 3.515985621 \\ 0 & 1.234078268 \end{bmatrix} \qquad B = \begin{bmatrix} .3458968425 & 0 \\ .6521859685 & .3450509462 \end{bmatrix}$$

$$C = \begin{bmatrix} 5.748636323 & 5.095604458 \\ 2.232161079 & 1.579129214 \end{bmatrix}$$

and use HP-67 arithmetic ($u = 10^{-10}$), we find

$$\hat{X} = \begin{bmatrix} 1.003948200 & .999995000 \\ .999997700 & 1.000000000 \end{bmatrix}$$

Now in this example, $u[\ \kappa_2(U) + \kappa_2(V)] \approx 10^{-3}$ and so we should not be surprised to learn that to full working precision,

$$X = \begin{bmatrix} 1.000000000 & 1.000000000 \\ 1.000000000 & 1.000000000 \end{bmatrix}$$

Conclusion: we should avoid ill-conditioned transformation matrices. Methods which involve the computation of Jordan or companion forms in Step 1 do not do this. (c.f. [6,9].)

This leads us to consider transformation methods which rely on orthogonal U and V . (Recall that $U^TU = I$ implies $\kappa_2(U) = 1$.) In the next two sections we describe two such techniques: one old and one new. The first of these is the Bartels-Stewart algorithm. This method involves the orthogonal reduction of A and B to triangular form using the QR algorithm. The main point of this paper is to show how this algorithm can be streamlined by only reducing A to Hessenberg form. The resulting algorithm is described in Section 3 and its roundoff properties are shown to be very desirable in Sections 4 and 5. Our claims regarding speed and accuracy are substantiated in Section 6 where we report on several numerical tests. Finally, we conclude by showing how the techniques in this paper can be extended to other matrix equation problems.

2. The Bartels-Stewart Algorithm

The crux of the Bartels-Stewart algorithm [1] is the computation of the real Schur decompositions

$$(2.1)\qquad \begin{aligned} R &= U^T A\, U && R \text{ quasi-upper triangular, } U \text{ orthogonal} \\ S &= V^T B^T V && S \text{ quasi-upper triangular, } V \text{ orthogonal} \end{aligned}$$

A matrix is quasi-upper triangular if it is upper triangular with the possible exception of 2x2 "bumps" on the diagonal, e.g.

$$\begin{matrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x \end{matrix}$$

The 2x2 bumps, if they exist, correspond to complex conjugate eigenvalue pairs. (It is desirable to avoid complex arithmetic when solving a real $AX + XB = C$ problem.) The above quasi-triangular forms may be found through application of the well-known QR algorithm [12].

From our remarks in Section 1, the reductions (2.1) lead to a system of the form

$$(2.2)\qquad RY + YS^T = F \qquad (F = U^T C\, V\ ,\ Y = U^T X\, V\ )$$

which can readily be solved for Y . To see how, partition Y and

F into their respective columns

$$Y = [\, y_1 \mid y_2 \mid \ldots \mid y_n \,] \qquad F = [\, f_1 \mid f_2 \mid \ldots \mid f_n \,]$$

and assume that we have reached a stage where $y_{k+1}, \ldots, y_n$ are known.

If $s_{k,k-1} = 0$, then by comparing the k-th columns in (2.2) we find

$$(R + s_{kk} I) y_k = f_k - \sum_{j=k+1}^{n} s_{kj}\, y_j \tag{2.3}$$

Thus, y_k is the solution of an mxm quasi-triangular system of equations. This system requires $m^2/2$ operations to evaluate once the right hand side is computed. (In all our operation counts, we tabulate only multiplicative operations and ignore low order terms as is traditional.)

If $s_{k,k-1}$ is nonzero, then by equating columns k-1 and k in (2.2) we obtain

$$\begin{aligned} R[y_{k-1} \mid y_k] + [y_{k-1} \mid y_k] \begin{bmatrix} s_{k-1,k-1} & s_{k,k-1} \\ s_{k-1,k} & s_{kk} \end{bmatrix} \\ = [f_{k-1} \mid f_k] - \sum_{j=k+1}^{n} [s_{k-1,j}\, y_j \mid s_{kj}\, y_j] \equiv [\, g \mid w \,] \end{aligned} \tag{2.4}$$

By expressing this linear system in standard matrix-vector form we can solve for the components of $y_{k-1}^T = (y_{1,k-1}, \ldots, y_{m,k-1})$ and $y_k^T = (y_{1,k}, \ldots, y_{m,k})$. In particular, we obtain the 2m-by-2m system

$$
(2.5) \qquad \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1m} \\ P_{21} & P_{22} & \cdots & P_{2m} \\ \vdots & \vdots & & \vdots \\ P_{m1} & P_{m2} & \cdots & P_{mm} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix}
$$

where

$$
(2.6) \qquad P_{ij} = \begin{cases} \begin{bmatrix} r_{ij} & 0 \\ 0 & r_{ij} \end{bmatrix} & (i \neq j) \\[2ex] \begin{bmatrix} r_{ii} + s_{k-1,k-1} & s_{k-1,k} \\ s_{k,k-1} & r_{ii} + s_{kk} \end{bmatrix} & (i = j) \end{cases}
$$

and for $i = 1,\ldots, m$

$$
z_i = \begin{bmatrix} y_{i,k-1} \\ y_{i,k} \end{bmatrix} \qquad \text{and} \qquad d_i = \begin{bmatrix} g_i \\ w_i \end{bmatrix}
$$

Since $R = (r_{ij})$ is quasi-upper triangular, the above 2m x 2m matrix is upper triangular with the possible exception of 4x4 bumps on the diagonal. Using Gaussian elimination with partial pivoting to solve a linear system with this structure and dimension requires m^2 operations once the right hand side is evaluated. Aside from a few scalar variables, no additional storage is required beyond what is needed in those "nice" situations when S is actually upper triangular and therefore free of 2x2 bumps.

In summary, the Bartels-Stewart algorithm involves the following steps and work counts:

1. Compute the real Schur decompositions:

$$R = U^T A U \quad \text{(save U)} \qquad 10\, m^3$$

$$S^T = V^T B^T V \quad \text{(save V)} \qquad 10\, n^3$$

2. Update the righthand side:

$$F = U^T C V \qquad m^2 n + m n^2$$

3. Back substitute for Y :

$$RY + YS^T = F \qquad (m^2 n + m n^2)/2$$

4. Obtain solution:

$$X = U Y V^T \qquad m^2 n + m n^2$$

$$w_{BS}(m,n) = 10m^3 + 10n^3 + \frac{5}{2} [m^2 n + mn^2]$$

The work counts associated with the Schur forms are estimates based upon experience with the QR algorithm which normally requires between one and two iterations per eigenvalue [11,12].

In terms of storage, the algorithm needs five arrays for the following purposes:

A	(mxm)	for the original A and subsequently R
U	(mxm)	for the orthogonal matrix U
B	(nxn)	for the original B and subsequently S
V	(nxn)	for the orthogonal matrix V
C	(mxn)	for the original C and subsequently Y and X

3. The Hessenberg-Schur Algorithm

In this section we describe a new algorithm, called the Hessenberg-Schur algorithm, which differs from the Bartels-Stewart method in that the decompositions (2.1) are replaced by

$$(3.1)\qquad \begin{aligned} H &= U^T A U && H \text{ upper Hessenberg , } U \text{ orthogonal} \\ S &= V^T B^T V && S \text{ quasi-upper triangular, } V \text{ orthogonal} \end{aligned}$$

A matrix $H = (h_{ij})$ is upper Hessenberg if $h_{ij} = 0$ for all $i > j+1$. The orthogonal reduction of A to upper Hessenberg form can be accomplished with Householder matrices in $\frac{5}{3} m^3$ operations. See [12, p. 347] for a description of this algorithm. The reductions (3.1) lead to a system of the form

$$(3.2)\qquad HY + YS^T = F$$

which may be solved in a manner similar to what is done in the Bartels-Stewart algorithm. In particular, assume that $y_{k+1},\ldots,y_n$ have been computed.

If $s_{k-1,k} = 0$, then y_k can be determined by solving the mxm Hessenberg system

$$(3.3)\qquad (H + s_{kk}I)y_k = f_k - \sum_{j=k+1}^{n} s_{kj}\, y_j$$

When Gaussian elimination with partial pivoting is used for this purpose, m^2 operations are needed once the righthand side is known.

If $s_{k,k-1}$ is nonzero, then by equating columns k-1 and k in (3.2) we find

$$H[y_{k-1}|y_k] + [y_{k-1}|y_k]\begin{bmatrix} s_{k-1,k-1} & s_{k,k-1} \\ s_{k-1,k} & s_{kk} \end{bmatrix}$$

$$= [f_{k-1}|f_k] - \sum_{j=k+1}^{n} [s_{k-1,j}\, y_j | s_{kj} y_j] \equiv [\, g \mid w \,]$$

This leads to the $2m \times 2m$ system (2.5) where the P_{ij} are defined by

$$(3.4) \qquad P_{ij} = \begin{cases} \begin{bmatrix} h_{ij} & 0 \\ 0 & h_{ij} \end{bmatrix} & (i \neq j) \\[2ex] \begin{bmatrix} h_{ii} + s_{k-1,k-1} & s_{k-1,k} \\ s_{k,k-1} & h_{ii} + s_{kk} \end{bmatrix} & (i = j) \end{cases}$$

Since H is upper Hessenberg, it follows that (2.5) is an upper triangular system with two nonzero subdiagonals. Using Gaussian elimination with partial pivoting, this system can be solved in $6m^2$ operations once the righthand side is known. Unfortunately, a $2m^2$ workspace is required to carry out the computations.

Part of this increase in storage is compensated for by the fact that the orthogonal matrix U can be stored in factored form below the diagonal of H [12,p.350]. This implies that we do not need an mxm array for U as in the Bartels-Stewart algorithm. Summarizing the Hessenberg-Schur algorithm and the associated work counts we have:

1. Reduce A to upper Hessenberg and B^T to quasi-upper triangular:

   $H = U^T A U$ (store U in factored form) $\frac{5}{3} m^3$

   $S = V^T B V$ (save V) $10 n^3$

2. Update the righthand side:

   $F = U^T C V$ $m^2 n + mn^2$

3. Back substitute for Y :

   $HY + YS^T = F$ $3m^2 n + \frac{1}{2} mn^2$

4. Obtain solution:

   $X = U Y V^T$ $m^2 n + mn^2$

$$w_{HS}(m,n) = \frac{5}{3} m^3 + 10n^3 + 5m^2 n + \frac{5}{2} mn^2$$

To obtain the operation count associated with the determination of Y, we assumed that S has $\frac{n}{2}$ 2x2 bumps along its diagonal. (This is the "worst" case.)

Unlike the work count for the Bartels-Stewart algorithm, $w_{HS}(m,n)$ is not symmetric in m and n . Indeed, scrutiny of $w_{HS}(m,n)$ reveals that it clearly pays to have $m \geq n$. This can always be assured, for if $m < n$, we merely apply the Hessenberg-Schur algorithm to the transposed problem

$$B^T X^T + X^T A^T = C^T$$

Comparing $w_{BS}(m,n)$ and $w_{HS}(m,n)$ we find

$$\frac{w_{HS}(m,n)}{w_{BS}(m,n)} = \frac{1 + 3(n/m) + \frac{3}{2}(n/m)^2 + 6(n/m)^3}{6 + \frac{3}{2}(n/m) + \frac{3}{2}(n/m)^2 + 6(n/m)^3} \tag{3.5}$$

which indicates that substantial savings accrue when the Hessenberg-Schur method is favored. For example, if $m = 4n$, then $w_{HS}(m,n) = .30\, w_{BS}(m,n)$.

The storage requirements of the new method are a little greater than those for the Bartels-Stewart algorithm:

A (mxm)	for the original A and subsequently H and U
B (nxn)	for the original B and subsequently S
V (nxn)	for the orthogonal matrix V
C (mxn)	for the original C and subsequently Y and X
W ($2m^2$)	for handling the possible system (2.5),(3.4)

4. A Perturbation Analysis

In the next section we shall assess the effect of rounding errors on the Hessenberg-Schur algorithm. The assessment will largely be in the form of a bound on the relative error in the computed solution $\hat{X}$. To insure a correct interpretation of our results, it is first necessary to investigate the amount of error which we can expect any algorithm to generate given finite precision arithmetic.

To do this we need to make some observations about the sensitivity of the underlying problem $\phi(X) = C$. This system of equations can be written in the form

$$Px = c \tag{4.1}$$

where

$$P = (I_n \otimes A) + (B^T \otimes I_m) \tag{4.2}$$

and

$$x = \text{vec}(X) = (x_{11}, x_{21}, \ldots, x_{m1}, x_{12}, x_{22}, \ldots, x_{m2}, \ldots, x_{1n}, \ldots, x_{mn})^T$$

$$c = \text{vec}(C) = (c_{11}, c_{21}, \ldots, c_{m1}, c_{12}, c_{22}, \ldots, c_{m2}, \ldots, c_{1n}, \ldots, c_{mn})^T$$

Here, the Kronecker product $W \otimes Z$ of two matrices W and Z is the block matrix whose (i,j) block is $w_{ij}Z$.

Based on our knowledge of linear system sensitivity, we know that if P is ill-conditioned, then small changes in A, B, and/or C can induce relatively large changes in the solution.

To relate this to the transformation ϕ, we need to define a norm on the space of linear transformations from $R^{m\times n}$ to $R^{m\times n}$:

$$f: R^{m\times n} \to R^{m\times n} \qquad \| f \|_2 = \max_{X \in R^{m\times n}} \frac{\| f(X) \|_F}{\| X \|_F}$$

Here, the Frobenius norm $\|\cdot\|_F$ is defined by $\| W \|_F^2 = \sum_{i,j} |w_{ij}|^2$.

Notice that for the linear transformation ϕ defined by (1.1) we have

$$\| \phi \| = \| P \|_2 \leq \| A \|_2 + \| B \|_2$$

where P is defined by (4.2). If ϕ is nonsingular, then

$$\| \phi^{-1} \| = \left[\min_{X \in R^{m\times n}} \frac{\| \phi(X) \|_F}{\| X \|_F} \right]^{-1} = \| P^{-1} \|_2$$

Now consider solving $AX + XB = C$ on a computer having machine precision u. In general, rounding errors of order $u \| A \|_F$, $u \| B \|_F$, and $u \| C \|_F$ will normally be present in A, B, and C respectively before any algorithm even begins execution. Hence, the "best" thing we can say about a computed solution $\hat{X}$ is that it satisfies

(4.3) $$(A + E)\hat{X} + \hat{X}(B + F) = (C + G)$$

where

(4.4) $$\| E \|_F \leq u\| A \|_F$$

(4.5) $$\| F \|_F \leq u\| B \|_F$$

(4.6) $$\| G \|_F \leq u\| C \|_F$$

How accurate would such an $\hat{X}$ be? To answer this question, we first establish a simple result concerning perturbed linear systems.

Lemma

Let M and ΔM be kxk matrices and let x, Δx, d, and Δd, be k-vectors. Assume that M is nonsingular and d nonzero. If

(4.7) $$Mx = d$$

(4.8) $$(M + \Delta M)(x + \Delta x) = (d + \Delta d)$$

and

(4.9) $$\| M^{-1} \|_2 \, \|\Delta M \|_2 \leq 1/2$$

then

(4.10) $$\frac{\| \Delta x \|_2}{\| x \|_2} \leq 2 \| M^{-1} \|_2 \left[\| \Delta M \|_2 + \frac{\| \Delta d \|_2}{\| x \|_2}\right]$$

Proof

Applying M^{-1} to (4.8), rearranging, and taking norms we find

$$\|\Delta x\|_2 \le \|M^{-1}\|_2 \; \|\Delta M\|_2 \; [\; \|x\|_2 + \|\Delta x\|_2 \;] \; + \|M^{-1}\|_2 \; \|\Delta d\|_2$$

The Lemma now follows by using (4.9). ▌

We now apply this result to the perturbed linear system given in (4.3).

Theorem

Assume that $AX + XB = C$, $(A + E)\hat{X} + \hat{X}(B + F) = (C + G)$, and (4.4), (4.5), and (4.6) hold. If $\phi(Z) = AZ + ZB$ is nonsingular, C nonzero, and

(4.11) $$u \; [\; \|A\|_F + \|B\|_F \;] \; \|\phi^{-1}\| \le 1/2 \quad ,$$

then

(4.12) $$\frac{\|X - \hat{X}\|_F}{\|X\|_F} \le 4\,u\,[\; \|A\|_F + \|B\|_F \;] \; \|\phi^{-1}\|$$

Proof

Defining $x = \mathrm{vec}(X)$, $\hat{x} = \mathrm{vec}(\hat{X})$, $c = \mathrm{vec}(C)$, $g = \mathrm{vec}(G)$, $P = (I_n \otimes A) + (B^T \otimes I_m)$, and $\Delta P = (I_n \otimes E) + (F^T \otimes I_m)$, we find

$$Px = c$$

$$(P + \Delta P)\hat{x} = c + g$$

$$\| \Phi \| = \| P \|_2 \leq \| A \|_2 + \| B \|_2 \leq \| A \|_F + \| B \|_F$$

and

$$\| \Delta P \|_2 \leq u [\| A \|_F + \| B \|_F]$$

Since $\| \Phi^{-1} \| = \| P^{-1} \|_2$ we have

$$\| P^{-1} \|_2 \; \| \Delta P \|_2 \leq u [\| A \|_F + \| B \|_F] \| \Phi^{-1} \| \leq 1/2$$

The above Lemma can now be applied and with a little manipulation we find

$$\frac{\| x - \hat{x} \|_F}{\| x \|_F} = \frac{\| x - \hat{x} \|_2}{\| x \|_2} \leq 2 \| \Phi^{-1} \| [u(\| A \|_F + \| B \|_F) + \frac{\| g \|_2}{\| x \|_2}]$$

The theorem follows since

$$\| g \|_2 \leq u \| C \|_F \leq u [\| A \|_F + \| B \|_F] \| x \|_2$$

∎

For the 2x2 example given in Section 1 , the upper bound in (4.12) has a value of about 10^{-9}. This indicates that an $AX + XB = C$ problem can be very well-conditioned even if the eigenvector matrices for A and B are poorly conditioned.

We conclude this section with the remark that the bound in (4.12) can roughly be attained. Setting

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} \delta-1 & 0 \\ 1 & 3 \end{bmatrix} \qquad C = \begin{bmatrix} 2+\delta & 5 \\ 1+\delta & 4 \end{bmatrix}$$

it is easy to verify that $\kappa(\Phi) = \| \Phi \| \, \| \Phi^{-1} \| = O(1/\delta)$ and

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} .$$

(Think of δ as a small positive constant.) Now if

$$A\hat{X} + \hat{X}B = C + \begin{bmatrix} u & 0 \\ 0 & 0 \end{bmatrix}$$

it is easy to show

$$\hat{X} = X + \begin{bmatrix} u/\delta & 0 \\ 0 & 0 \end{bmatrix}$$

Thus, if $\| \Phi^{-1} \|$ is large, then small changes in A, B, or C can induce relatively large changes in X.

In general, given the random nature of roundoff error, we conclude from the analysis in this section that errors of order $u \| \Phi^{-1} \|$ can be expected to contaminate the computed solution no matter what algorithm we employ to solve $AX + XB = C$.

5. Roundoff Error Analysis of the Hessenberg-Schur Algorithm

We now take a detailed look at the roundoff errors associat with the Hessenberg-Schur algorithm. This amounts to applying som standard results from Wilkinson [12]. His inverse error analysis pertaining to orthogonal matrices can be applied to the computati $H = U^TAU$, $S = V^TB^TV$, $F = U^TC\,V$, and $X = UYV^T$ while his well-known analysis of Gaussian elimination and back-substitution can used in connection with the determination of Y . (Y is essential obtained by using Gaussian elimination and back substitution to solve the system $[(I_n \otimes H) + (S \otimes I_m)]\mathrm{vec}(Y) = \mathrm{vec}(F)$.) Denoti computed quantities with the "hat" notation, we can account for th rounding errors in the Hessenberg-Schur algorithm in the followin way:

$$\hat{U} = U_1 + E_u \qquad U_1^TU_1 = I\ ,\ \|E_u\|_F \le \varepsilon \tag{5.1}$$

$$\hat{V} = V_1 + E_v \qquad V_1^TV_1 = I\ ,\ \|E_v\|_F \le \varepsilon \tag{5.2}$$

$$\hat{H} = U_1^T(A + E_1)U_1 \qquad \|E_1\|_F \le \varepsilon\|A\|_F \tag{5.3}$$

$$\hat{S} = V_1^T(B + E_2)^TV_1 \qquad \|E_2\|_F \le \varepsilon\|B\|_F \tag{5.4}$$

$$\hat{F} = U_1^T(C + E_3)V_1 \qquad \|E_3\|_F \le \varepsilon\|C\|_F \tag{5.5}$$

$$(\hat{T} + E_4)\hat{y} = \hat{f} \qquad \|E_4\|_2 \le \varepsilon\|\hat{T}\|_2 \tag{5.6}$$

$$\hat{X} = U_1(\hat{Y} + E_5)V_1^T \qquad \|E_5\|_F \le \varepsilon\|\hat{Y}\|_F \tag{5.7}$$

where

$$\hat{T} = (I_n \otimes H) + (S \otimes I_m) \tag{5.8}$$

$$\hat{y} = \mathrm{vec}(\hat{Y}) \tag{5.9}$$

$$\hat{f} = \mathrm{vec}(\hat{F}) \tag{5.10}$$

and ε is a small multiple of the machine precision u. (We have used the 2-norm in (5.6) for convenience.) We now proceed to bound the relative error in $\hat{X}$.

Defining $W = U_1^T X V_1$, it follows from (5.07) and the orthogonal invariance of the Frobenius norm that

$$\frac{\| X - \hat{X} \|_F}{\| X \|_F} \leq \frac{\| W - \hat{Y} \|_F}{\| W \|_F} + \varepsilon \frac{\| \hat{Y} \|_F}{\| W \|_F} \tag{5.11}$$

Since W solves $(U_1^T A U_1)Z + Z(V_1^T B V_1) = U_1^T C V_1$, we have

$$Tw = f \tag{5.12}$$

where $w = \mathrm{vec}(W)$, $f = \mathrm{vec}(U_1^T C V_1)$ and $T = (I_n \otimes U_1^T A U_1) + (V_1^T B^T V_1 \otimes I_m)$

Notice from (5.6), (5.8), (5.9) and (5.10) that $\hat{y}$ satisfies a perturbed version of this system. In particular,

$$(T + \Delta T)\hat{y} = f + \Delta f \tag{5.13}$$

where $\Delta T = (\hat{T} - T) + E_4$ and $\Delta f = \mathrm{vec}(U_1^T E_3 V_1)$. Manipulation with (5.3), (5.4) and (5.6) shows

$$\| \Delta T \|_2 \leq \varepsilon(2+\varepsilon) [\| A \|_F + \| B \|_F]$$

and with (5.5) and (5.10) that

$$\| \Delta f \|_2 \leq \varepsilon \| C \|_F \leq \varepsilon [\| A \|_F + \| B \|_F] \| W \|_F$$

Now $\| \phi^{-1} \| = \| T^{-1} \|_2$ and if we make the assumption

$$(5.14) \qquad \| \phi^{-1} \| \varepsilon(2+\varepsilon) [\| A \|_2 + \| B \|_2] < 1/2$$

then the Lemma is applicable and we find

$$\frac{\| W - \hat{Y} \|_F}{\| W \|_F} = \frac{\| w - \hat{y} \|_2}{\| w \|_2} < (6\varepsilon + 2\varepsilon^2) \| \phi^{-1} \| [\| A \|_F + \| B \|_F]$$

Furthermore, (5.14) can also be used to show

$$\| \hat{Y} \|_F \leq 3 \| W \|_F$$

and therefore from (5.11) we have

$$(5.15) \qquad \frac{\| X - \hat{X} \|_F}{\| X \|_F} \leq (9\varepsilon + 2\varepsilon^2) \| \phi^{-1} \| [\| A \|_F + \| B \|_F]$$

Inequality (5.15) indicates that errors no worse in magnitude than $O(\|\Phi^{-1}\|\,\epsilon)$ will contaminate the computed $\hat{X}$. Since ϵ is a small multiple of the machine precision u, we see that (5.15) is essentially the same result as (4.12) which was established under the "ideal" assumptions (4.3)-(4.6). Likewise, assumption (5.14) corresponds to assumption (4.11). Conclusion: the roundoff properties of the Hessenberg-Schur algorithm are as good as can be expected from any algorithm designed to solve $AX + XB = C$.

We finish this section with two remarks. First, the entire analysis is applicable to the Bartels-Stewart algorithm. We simply replace (5.3) with

$$(5.3') \qquad R = U_1^T(A + E_1)U_1 \qquad \| E_1 \|_F \le \epsilon \| A \|_F$$

where R is now quasi-triangular instead of Hessenberg.

Our second remark concerns another standard by which the quality of $\hat{X}$ can be judged. In some applications, one may be more interested in the norm of the residual $\| A\hat{X} + \hat{X}B - C \|_F$ than the relative error. An analysis similar to that above reveals that if (5.1)-(5.10) and (5.14) hold, then

$$(5.16) \qquad \| A\hat{X} + \hat{X}B - C \|_F \le (10\epsilon + 3\epsilon^2)(\|A\|_F + \|B\|_F)\|X\|_F$$

Notice that the bound does not involve $\| \Phi^{-1} \|$.

6. The FORTRAN Codes and Their Performance

A collection of FORTRAN subroutines have been written which implement the Hessenberg-Schur algorithm. Here is a summary of what the chief routines in the package do:

- AXXBC - This is the main calling subroutine and the only one which the user "sees". It assumes $m > n$.
- ORTHES - This subroutine reduces a matrix to upper Hessenberg form using Householder matrices. All the information pertaining to the reduction is stored below the main diagonal of the reduced matrix.
- ORTRAN - This subroutine is used to explicitly form the orthogonal matrix obtained by ORTHES.
- HQR2 - This subroutine reduces an upper Hessenberg matrix to upper quasi-triangular form using the QR algorithm.
- TRANSF - This subroutine computes products of the form U^TCV and and UYV^T where U and V are orthogonal.
- NSOLVE, HESOLV, BACKSB - These routines combine to solve upper Hessenberg systems using Gaussian elimination with partial pivoting.
- N2SOLV, H2SOLV, BACKSB - These routines combine to solve the 2m-by-2m block Hessenberg systems encountered whenever S has a 2-by-2 bump.

The above codes are designed to handle double precision A, B, and C and require about 23,000 bytes of storage. This amount of memory is put into perspective with the remark that when a 25x25 problem is solved, the program itself accounts for one-half of the total storage.

To assess the effectiveness of our subroutines we ran two sets of tests. In the first set we compared the execution times for our method and the Bartels-Stewart algorithm. For a given value of n/m, many examples were run ranging in dimension from 10 to 50. The timing ratios were then averaged. The following table summarizes what we found:

n/m	$\frac{w_{HS}(m,n)}{w_{BS}(m,n)}$	$\frac{\text{HS Execution time}}{\text{BS Execution time}}$ (average)
1.00	.76	.84
.75	.63	.70
.50	.46	.54
.25	.30	.35

TABLE 1. Timings

Although the predicted savings (second column) are a little greater than those actually obtained (third column), the results certainly confirm the superior efficiency of the Hessenberg-Schur algorithm.

We also compared the accuracy of the two methods on the same class of examples and found them indistinguishable. This is to be expected because the favorable error analysis in the previous section applies to both algorithms.

The second class of test problems was designed to examine the behavior of the algorithm on ill-conditioned $AX + XB = C$ examples. This was accomplished by letting A and B have the form

$$A = \mathrm{diag}(1,2,3,\ldots,m) + N_m$$

$$B = 2^{-t} I_n - \mathrm{diag}(n,n-1,\ldots,1) + N_n^T$$

where

$$N_k = \begin{bmatrix} 0 & & & & \\ 1 & 0 & & \bigcirc & \\ 1 & 1 & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & 1 & 1 \cdots & 1 & 0 \end{bmatrix} \quad k \times k$$

Notice that there is coalescence among the eigenvalues of A and $-B$ as t gets large. This enables us to control the sensitivity of the transformation $\phi(X) = AX + XB$. (In particular, it is easy to show that $\| \phi^{-1} \| \geq 2^t$.) To facilitate the checking of errors, C is chosen so that the solution X is the matrix whose entries are each one. Using an IBM 370/168 with extended precision ($u = 16^{-14}$), we obtained the following results for an $m = 10$, $n = 4$ problem:

t	$\| \phi^{-1} \|$	$\dfrac{\| X - \hat{X} \|_F}{\| X \|_F}$	$\dfrac{\| A\hat{X} + \hat{X}B - C \|_F}{\| X \|_F (\| A \|_F + \| B \|_F)}$
1	2.3×10^{1}	2.1×10^{-14}	8.2×10^{-16}
10	8.3×10^{3}	5.0×10^{-12}	6.7×10^{-16}
15	2.7×10^{5}	1.4×10^{-10}	8.5×10^{-16}
20	8.3×10^{7}	9.3×10^{-9}	9.3×10^{-16}
25	2.8×10^{8}	1.6×10^{-7}	6.1×10^{-16}
30	9.0×10^{9}	8.6×10^{-6}	8.1×10^{-16}

TABLE 2. Errors and Residuals

The quantity $\|\Phi^{-1}\|$ is the reciprocal of the smallest singular value of the matrix $P = (I_4 \otimes A) + (B^T \otimes I_{10})$ and can be found by using the subroutine SVD which is in EISPACK.

The results of Table 2 affirm the key results (5.15) and (5.16). In particular, we see that small residuals are obtained regardless of the norm of Φ^{-1}. In contrast, the accuracy of $\hat{X}$ deteriorates as $\|\Phi^{-1}\|$ becomes large.

We conclude this section with the remark that the Hessenberg-Schur algorithm offers no advantage over the Bartels-Stewart method for the important case when $B = A^T$, i.e. the Lyapunov problem. This is because the latter algorithm requires only one Schur decomposition to solve $AX + XA^T = C$.

7. Extensions to Other Matrix Equation Problems

In this final section we indicate how the Hessenberg-Schur "idea" can be applied to two other matrix equation problems. Consider first the problem

$$AXM + X = C \tag{7.1}$$

where $A \in R^{m\times m}$, $M \in R^{n\times n}$, $C \in R^{m\times n}$, and $X \in R^{m\times n}$. If

$$U^TAU = H \qquad U^TU = I \text{ , } H \text{ upper Hessenberg}$$

and

$$V^TM^TV = S \qquad V^TV = I \text{ , } S \text{ quasi-upper triangular}$$

and $F = U^TC\,V$, then (7.1) transforms to

$$H\,Y\,S^T + Y = F \tag{7.2}$$

where $Y = U^TX\,V$. As in the Hessenberg-Schur algorithm, once $y_{k+1},\ldots,y_n$ are known, y_k can be found by solving a Hessenberg system. (Recall y_k is the k-th column of Y.) To see how, assume $s_{k,k-1} = 0$ and equate k-th columns in (7.2):

$$H\left(\sum_{j=k}^{n} s_{kj}y_j\right) + y_k = f_k$$

Hence, y_k can be found by solving

$$(s_{kk}H + I)y_k = \left[f_k - H\sum_{j=k+1}^{n} s_{kj}\,y_j\right]$$

The presence of 2x2 bumps on the diagonal of T can be handled in a fashion similar to what is done in the Hessenberg-Schur method.

This algorithm which we have sketched should be 30 - 70 percent faster than the Bartels-Stewart type technique in which both A and M are reduced to triangular form via the QR algorithm. (See [5].)

The second matrix equation problem we wish to consider involves finding $X \in R^{m\times n}$ such that

$$AXM + LXB = C \tag{7.3}$$

where $A, L \in R^{m\times m}$, $M, B \in R^{n\times n}$, and $C \in R^{m\times n}$. For a discussion of these and more general problems, see [7] and [13] .
If M and L are nonsingular, then (7.3) can be put into "standard" $AX + XB = C$ form:

$$(L^{-1}A)X + X(BM^{-1}) = L^{-1}CM^{-1}$$

If M and/or L is poorly conditioned, it may make more numerical sense to apply the QZ algorithm of Moler and Stewart [8] to effect a stable transformation of (7.3). In particular, their techniques allow us to compute orthogonal U, V, Q, and Z such that

$$\begin{aligned}
Q^TA\,U &= P \qquad \text{(quasi- upper triangular)}\\
Q^TL\,U &= R \qquad \text{(upper triangular)}\\
Z^TB^TV &= S \qquad \text{(quasi-upper triangular)}\\
Z^TM^TV &= T \qquad \text{(upper traingular)}
\end{aligned}$$

If $Y = U^TXV$ and $F = Q^TCZ$, then (7.3) transforms to

$$PYT^T + RYS^T = F$$

Comparing k-th columns and assuming $s_{k,k-1} = t_{k,k-1} = 0$ we find

$$P \sum_{j=k}^{n} t_{kj} y_j + R \sum_{j=k}^{n} s_{kj} y_j = f_k$$

and so

$$(7.4) \qquad (t_{kk}P + s_{kk}R) y_k = f_k - P \sum_{j=k+1}^{n} t_{kj} y_j - R \sum_{j=k+1}^{n} s_{kj} y_j$$

This quasi-triangular system can then be solved for y_k once the righthand side is known and under the assumption that the matrix $(t_{kk}P + s_{kk}R)$ is nonsingular. (Note that T, P, S, and R can all be singular without $t_{kk}P + s_{kk}R$ being singular.)

Now, as in the Hessenberg-Schur algorithm, significant economies can be made if A is only reduced to Hessenberg form. This is easily accomplished for when applied to the matrix pair (A,L), the QZ algorithm first computes orthogonal Q and U such that $Q^TAU = H$ is upper Hessenberg and $Q^TLU = R$ is upper triangular. The systems in (7.4) are now Hessenberg form and can consequently be solved very quickly. Again, we leave it to the reader to verify that the presence of 2x2 bumps on the diagonal of S pose no serious difficulties.

8. Conclusions

We have presented a new algorithm for solving the matrix equation $AX + XB = C$. The technique relies upon orthogonal matrix transformations and is not only extremely stable, but considerably faster than its nearest competitor, the Bartels-Stewart algorithm. We have included perturbation and roundoff analyses for the purpose of justifying the favorable performance of our method. Although these analyses may appear boring, they are critical to the development of reliable software for this important computational problem.

REFERENCES

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

[2] Belanger,P.R. and McGillivray,T.P. (1976), "Computational Experience with the Solution of the Matrix Lyapunov Equations", IEEE Trans.Auto.Contr.,AC-21,799-800.

[3] Forsythe,G.E. and Moler,C.B. (1967), Computer Solution of Linear Algebraic Systems, Prentice-Hall, Englewood Cliffs

[4] Hagander,P. (1972), "Numerical Solution of $A^TS + SA + Q = 0$", Inf.Sci.,4,35-50.

[5] Kitagawa,G., (1977), "An Algorithm for Solving the Matrix Equation $X = FXF^T + S$ ", Int.J.Cont.,25, 745-753.

[6] Kreisselmeier,G., (1973), "A Solution of the Bilinear Matrix Equation $AY + YB = -Q$" , SIAM J.Appl.Math., 23, 334-338.

[7] Lancaster,P. (1970), " Explicit Solutions of Linear Matrix Equations", SIAM Review, 12, 544-566.

[8] Moler,C.B. and Stewart,G.W.(1973), "An Algorithm for Generalized Matrix Eigenvalue Problems", SIAM J.Numer.Anal., 10, 241-256.

[9] Molinari,B.P. (1969), "Algebraic Solution of Matrix Linear Equations in Control Theory", Proc.IEE. , 116, 1748-1754.

[10] Rothschild,D. and Jameson,A. (1970), "Comparison of Four Numerical Algorithms for Solving the Lyapunov Matrix Equation", Int'l.J.Cont., 11, 181-198.

[11] Smith,B.T. et al, (1970), Matrix Eigensystem Routines - EISPAC Guide, Lecture Notes in Computer Science, Springer Verlag New York.

[12] Wilkinson,J.H. (1965), The Algebraic Eigenvalue Problem , Oxfo University Press, Oxford.

[13] Wimmer,H. and Ziebur,A.D. (1972), "Solving the Matrix Equation $\sum_{p=1}^{r} f_p(A)Xg_p(B) = C^*$ ", SIAM Review, 14, 318-323.

```
      SUBROUTINE AXXBC(NDIM,N,MDIM,M,A,V
C
      INTEGER I,IERR,INC,IPR(1),J,M,NDIM
      REAL*8 A(MDIM,M),B(NDIM,N),V(NDIM,
     *     C(MDIM,N),ORT(1),EPS,REPS,D(1
C
C     THIS ROUTINE SOLVES THE MATRIX EQU
C
C         A - M * M MATRIX
C         X - M * N MATRIX
C         B - N * N MATRIX
C         C - M * N MATRIX.
C
C     THE PARAMETERS FOR THIS SUBROUTINE
C
C     NDIM - DECLARED ROW DIMENSION OF T
C     N    - ROW DIMENSION OF B; COLUMN
C     MDIM - DECLARED ROW DIMENSION OF A
C     M    - ROW DIMENSION OF A, C, AND
C     A    - M * M MATRIX (DOUBLE PRECIS
C     B    - N * N MATRIX (DOUBLE PRECIS
C     V    - N * N MATRIX (DOUBLE PRECIS
C            WHICH TRANSFORMS B(T) TO UP
C     C    - M * N MATRIX WHICH HOLDS C
C     ORT  - DOUBLE PRECISION VECTOR USE
C            OF LENGTH AT LEAST MAX(N,M)
C     D    - DOUBLE PRECISION VECTOR FOR
C            AT LEAST 2M**2 + 7M
C     IPR  - INTEGER VECTOR FOR INTERNAL
C     EPS  - ERROR TOLERANCE. EPS SHOULD
C            WHICH SATISFIES FL(1 + EPS)
C     IERR - INTEGER VARIABLE USED TO SE
C            IERR=0 => PROPER RETURN.
C            IERR=J => J-TH EIGENVALUE C
C                      AFTER 30 OR ITERA
C            IERR=-J => A SINGULAR MATRI
C                       SOLVING FOR THE
C
C     METHOD:
C
C     FIRST OF ALL, B(TRANSPOSE) IS FORM
C     B THROUGHOUT THE PROGRAM. (IN THE
C     USED TO REPRESENT B(TRANSPOSE)).
C     B(T) IS TRANSFORMED TO REAL UPPER
C     HESSENBERG FORM USING ORTHOGONAL T
C     THE RIGHT HAND SIDE C IS MULTIPLIE
C     MATRICES AND THE SOLUTION OF THIS
C     COMPUTED.  THIS SOLUTION IS THEN M
C     FORMATION MATRICES IN ORDER TO OBT
C     ORIGINAL PROBLEM.
C
C     NOTE:
C
C     IN ORDER TO GET OPTIMAL EFFICIENCY
C     BE <= M.  IF THIS IS NOT TRUE FOR
C     EQUATION AND SOLVE THE RESULTING P
C
C     ----------------------------------
C
```

```
,B,C,ORT,D,IPR,EPS,IERR)

,N,NCIM
N),
)

ATION AX + XB = C WHERE



ARE:

HE MATRIX B
DIMENSION OF B AND C
, C, AND V
V: COLUMN DIMENSION OF A AND V
ICN)
ICN)
ICN) USED TO STORE THE MATRIX
FER SCHUR FORM
CN INPUT AND X ON OUTPUT
D BY SUBROUTINE RG

INTERNAL USE OF LENGTH

USE OF LENGTH AT LEAST 4N
EQUAL THE SMALLEST NUMBER
> 1.
T ERROR CODES.

IF B HAS NOT BEEN DETERMINED
ATIONS.
IX WAS ENCOUNTERED WHEN
J-TH COLUMN OF X.



ED AND IS USED INSTEAD OF
FOLLOWING DISCUSSION, B(T) IS

SCHUR FORM AND A TO UPPER
RANSFORMATIONS.
ED BY THESE TRANSFORMATION
TRANSFORMED SYSTEM IS
ULTIPLIED BY THE TRANS-
AIN THE SOLUTION TO THE



Y FROM THIS ROUTINE, N SHOULD
YOUR PROBLEM, TRANSPOSE THE
PROBLEM.

--------------------------------
```

M

```
C        FORM B(TRANSPOSE) FOR INTERNAL USE
C
         DO 5 I = 1,N
            DO 5 J = I,N
               D(1) = B(I,J)
               B(I,J) = B(J,I)
               B(J,I) = D(1)
5        CONTINUE
C
C        A AND B(T) WILL BE TRANSFORMED INTO THE
C        MODIFICATIONS OF EISPACK ROUTINES (MODI
C
C        WARNING:  DO NOT CHANGE THE ORDER OF TH
C
         CALL RG(NDIM,N,B,1,V,ORT,EPS,IERR)
         CALL RG(MDIM,M,A,0,A,ORT,EPS,IERR)
C
C        NOW, TRANSFORM THE RIGHT HAND SIDE,
C
         CALL TRANSF(A,ORT,1,C,V,0,M,N,MDIM,NDIM
C
C        NOW SOLVE THE SYSTEM OF EQUATIONS (BLOC
C        THE SYSTEM IS BLOCK UPPER HESSENBERG WI
C        MATRICES ON THE DIAGONAL (A + B(I,I) *
C        THE IDENTITY OFF THE DIAGONAL.
C        TESTS ARE MADE TO SEE WHETHER A SINGLE
C        MUST BE HANDLED AT THE CURRENT STAGE AN
C        SUBROUTINES ARE CALLED.
C
C        SET RELATIVE ERROR TOLERANCE
C
         REPS = EPS*N*N*M*M
         IND = N - 1
         IF (IND.EQ.0) GO TO 40
10       IF (DABS(B(IND + 1,IND)).LE.REPS)  GO T
         CALL N2SOLV(A,B,C,D,NDIM,N,MDIM,M,IND,I
         IF (IERR.NE.0) RETURN
         GO TO 30
20       CALL NSOLVE(A,B,C,D,NDIM,N,MDIM,M,IND,I
         IF (IERR.NE.0) RETURN
30       IF (IND.GT.0) GO TO 10
40       IF (IND.EQ.0) CALL NSOLVE(A,B,C,D,NDIM,
C
C        THE SOLUTION OF THE TRANSFORMED SYSTEM
C        HAS NOW BEEN OBTAINED.  IT IS TRANSFORM
C        THE SOLUTION OF THE ORIGINAL SYSTEM OF
C
         CALL TRANSF(A,ORT,0,C,V,1,M,N,MDIM,NDIM
         RETURN
         END
C
C
C
C
C
         SUBROUTINE RG(NDIM,N,A,MATZ,Z,ORT,EPS,I
C
         INTEGER N,NDIM,IERR,MATZ
         REAL*8 A(NDIM,N),Z(NDIM,N),ORT(N),EPS
C
```

```
 APPRCPRIATE FORMS USING
FIEC BY S. NASH).

E NEXT TWO STATEMENTS




,0)

K BACK SUBSTITUTION)
TH UPPER HESSENBERG
I) ANC MULTIPLES OF

OR A CCUELE BLOCK
D APPRCPRIATE





0 20
PR,REPS,IERR)


PR,REPS,IERR)


N,MDIM,M,IND,IPR,REPS,IERR)

OF EQUATICNS
ED BACK INTO
EQUATIONS.

,C)







ERR)
```

```
 UPPER-HESSENBERG
MATZ IS ZERO OR NOT.
EXCEPT THAT HQR2 HAS
THE EIGENVECTORS OF
 ORTHOGONAL TRANSFORMATIONS


E MATRIX A
IX A
 (INPUT) AND TRANSFORMED

> TRANSFORM A TO UPPER-
OMPUTE THE SCHUR FORM OF A
TRIX IF A IS TRANSFORMED TO
 BE THE SAME AS A).
RAGE: IF MATZ=0, ORT HOLDS
ORMATION MATRIX
UTINE AXXBC.
C FOR DETAILS




1



 THE ALGOL PROCEDURE ORTHES,
TIN AND WILKINSON.
NEAR ALGEBRA, 339-358(1971).

UBROUTINE
S AND COLUMNS
M BY
NS.


NSION OF TWO-DIMENSIONAL
N THE CALLING PROGRAM
```

```
C     THIS SUBROUTINE REDUCES A TO EITHER
C     OR SCHUR FORM DEPENDING ON WHETHER
C     THE ROUTINES USED ARE FROM EISPACK
C     BEEN MODIFIED SO AS NOT TO COMPUTE
C     A AND ONLY REDUCES A TO SCHUR FORM.
C     ARE USED THROUGHOUT.
C     PARAMETERS:
C     NDIM - DECLARED ROW DIMENSION OF TH
C     N    - ACTUAL DIMENSION OF THE MATR
C     A    - MATRIX TO BE TRANSFORMED (ON
C            MATRIX (ON OUTPUT)
C     MATZ - INTEGER VARIABLE; MATZ = 0 =
C            HESSENBERG FORM; OTHERWISE C
C     Z    - STORES THE TRANSFORMATION MA
C          - SCHUR FORM (IF MATZ=0, Z CAN
C     ORT  - VECTOR USED FOR INTERNAL STO
C            INFORMATION ABOUT THE TRANSF
C     EPS  - ERROR TOLERANCE, AS IN SUBRO
C     IERR - ERROR CODE; SEE ROUTINE AXXE
C
      IERR=0
      CALL ORTHES(NDIM,N,A,ORT)
      IF (MATZ.EQ.0) RETURN
      CALL ORTRAN(NDIM,N,A,ORT,Z)
      CALL HQR2(NDIM,N,A,Z,EPS,IERR)
   10 RETURN
      END
C
C
C
C
C
      SUBROUTINE ORTHES(NDIM,N,A,ORT)
C
      INTEGER I,J,M,N,II,JJ,LA,MP,NDIM,KP
      REAL*8 A(NDIM,N),ORT(N)
      REAL*8 F,G,H,SCALE
      REAL*8 DSQRT,DABS,DSIGN
C
C     THIS SUBROUTINE IS A TRANSLATION OF
C     NUM. MATH. 12, 349-368(1968) BY MAR
C     HANDBOOK FOR AUTO. COMP., VOL.II-LI
C
C     GIVEN A REAL GENERAL MATRIX, THIS S
C     REDUCES A SUBMATRIX SITUATED IN ROW
C     1 THROUGH N TO UPPER HESSENBERG FOR
C     ORTHOGONAL SIMILARITY TRANSFORMATIO
C
C     ON INPUT:
C
C        NDIM MUST BE SET TO THE ROW DIME
C          ARRAY PARAMETERS AS DECLARED I
C          DIMENSION STATEMENT;
C
C        N IS THE ORDER OF THE MATRIX;
C
C        A CONTAINS THE INPUT MATRIX.
C
C     ON OUTPUT:
C
```

```
C             A CONTAINS THE HESSENBERG MATR
C               THE ORTHOGONAL TRANSFORMATIO
C               IS STORED IN THE REMAINING T
C               HESSENBERG MATRIX;
C
C             ORT CONTAINS FURTHER INFORMATI
C               ONLY ELEMENTS 1 THROUGH N AR
C
C          QUESTIONS AND COMMENTS SHOULD BE
C          APPLIED MATHEMATICS DIVISION, ARG
C
C          ----------------------------------
C
           LA = N - 1
           KP1 = 2
           IF (LA .LT. KP1) GO TO 200
C
           DO 180 M = KP1, LA
              H = 0.0D0
              ORT(M) = 0.0D0
              SCALE = 0.0D0
C
C          SCALE COLUMN (ALGOL TOL THEN NOT
C
              DO 90 I = M, N
   90         SCALE = SCALE + DABS(A(I,M-1))
C
              IF (SCALE .EQ. 0.0D0) GO TO 18
              MP = M + N
C
C             FOR I=N STEP -1 UNTIL M DO --
C
              DO 100 II = M, N
                 I = MP - II
                 ORT(I) = A(I,M-1) / SCALE
                 H = H + ORT(I) * ORT(I)
  100         CONTINUE
C
              G = -DSIGN(DSQRT(H),ORT(M))
              H = H - ORT(M) * G
              ORT(M) = ORT(M) - G
C
C             FORM (I-(U*UT)/H) * A
C
              DO 130 J = M, N
                 F = 0.0D0
C
C                FOR I=N STEP -1 UNTIL M DO
C
                 DO 110 II = M, N
                    I = MP - II
                    F = F + ORT(I) * A(I,J)
  110            CONTINUE
C
                 F = F / H
C
                 DO 120 I = M, N
  120            A(I,J) = A(I,J) - F * ORT(I
C
  130         CONTINUE
```

```
IX.  INFORMATION ABOUT
NS USED IN THE REDUCTION
RIANGLE UNDER THE

ON ABOUT THE TRANSFORMATIONS.
E USED.

DIRECTED TO B. S. GARBOW,
ONNE NATIONAL LABORATORY

------------------------------









NEEDED)




C

























--











)
```

ALGOL PROCEDURE ORTRANS.
NC WILKINSON.
ALGEBRA, 372-395(1971).

AL SIMILARITY
A REAL GENERAL
ES.

OF TWO-DIMENSIONAL
CALLING PROGRAM

OGONAL TRANS-
ORTHES

T THE TRANS-
ORTHES.

```
      C
      C        FORM (I-(U*UT)/H)*A*(I-(U*UT)/H)
      C
               DO 160 I = 1, N
                  F = 0.0D0
      C
      C           FOR J=N STEP -1 UNTIL M DO --
      C
                  DO 140 JJ = M, N
                     J = MP - JJ
                     F = F + ORT(J) * A(I,J)
      140         CONTINUE
      C
                  F = F / H
      C
                  DO 150 J = M, N
      150         A(I,J) = A(I,J) - F * ORT(J)
      C
      160      CONTINUE
      C
               ORT(M) = SCALE * ORT(M)
               A(M,M-1) = SCALE * G
      180   CONTINUE
      200   RETURN
            END
      C
      C
      C
      C
      C
            SUBROUTINE ORTRAN(NDIM,N,A,ORT,Z)
      C
            INTEGER I,J,N,KL,MM,MP,NDIM,MP1
            REAL*8 A(NDIM,N),ORT(N),Z(NDIM,N)
            REAL*8 G
      C
      C     THIS SUBROUTINE IS A TRANSLATION OF THE
      C     NUM. MATH. 16, 181-204(1970) BY PETERS A
      C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR
      C
      C     THIS SUBROUTINE ACCUMULATES THE ORTHOGON
      C     TRANSFORMATIONS USED IN THE REDUCTION OF
      C     MATRIX TO UPPER HESSENBERG FORM BY  ORTH
      C
      C     ON INPUT:
      C
      C        NDIM MUST BE SET TO THE ROW DIMENSION
      C          ARRAY PARAMETERS AS DECLARED IN THE
      C          DIMENSION STATEMENT;
      C
      C        N IS THE ORDER OF THE MATRIX;
      C
      C        A CONTAINS INFORMATION ABOUT THE ORTH
      C          FORMATIONS USED IN THE REDUCTION BY
      C          IN ITS STRICT LOWER TRIANGLE;
      C
      C        ORT CONTAINS FURTHER INFORMATION ABOU
      C          FORMATIONS USED IN THE REDUCTION BY
      C          ONLY ELEMENTS 1 THROUGH N ARE USED.
      C
```

```
C      ON OUTPUT:
C
C         Z CONTAINS THE TRANSFORMATION
C           REDUCTION BY  ORTHES;
C
C         ORT HAS BEEN ALTERED.
C
C      QUESTIONS AND COMMENTS SHOULD BE
C      APPLIED MATHEMATICS DIVISION, AR
C
C      ------------------------------------
C
C      INITIALIZE Z TO IDENTITY MATRIX
C
       DO 80 I = 1, N
C
          DO 60 J = 1, N
   60     Z(I,J) = 0.0D0
C
          Z(I,I) = 1.0D0
   80  CONTINUE
C
       KL = N - 2
       IF (KL .LT. 1) GO TO 200
C
C      FOR MP=N-1 STEP -1 UNTIL 2 DO --
C
       DO 140 MM = 1, KL
          MP = N - MM
          IF (A(MP,MP-1) .EQ. 0.0D0) GO
          MP1 = MP + 1
C
          DO 100 I = MP1, N
  100     ORT(I) = A(I,MP-1)
C
          DO 130 J = MP, N
             G = 0.0D0
C
             DO 110 I = MP, N
  110        G = G + ORT(I) * Z(I,J)
C
C            DIVISOR BELOW IS NEGATIVE
C            DOUBLE DIVISION AVOIDS POS
C
             G = (G / ORT(MP)) / A(MP,M
C
             DO 120 I = MP, N
  120        Z(I,J) = Z(I,J) + G * ORT(
C
  130     CONTINUE
C
  140  CONTINUE
C
  200  RETURN
       END
C
C
C
C
C
```

MATRIX PRODUCED IN THE

DIRECTED TO B. S. GARBOW,
GONNE NATIONAL LABORATORY

TO 140

OF H FORMED IN ORTHES.
SIBLE UNDERFLOW

P-1)

I)

```
DIM,NN,

ORM,EPS



= REAL AND
PLEX NUMBERS


 ALGOL PROCEDURE HQR2,
AND WILKINSON.
 ALGEBRA, 372-395(1971).
 NASH SO THAT IT ONLY
SSENBERG MATRIX H.


N OF TWO - DIMENSIONAL
E CALLING PROGRAM


IX:

 PRODUCED BY ORTRAN



.


HAS NOT BEEN
RATIONS.

----------------------------

ECIFYING
F ARITHMETIC.
```

```
      SUBROUTINE HQR2(NDIM,N,H,Z,EPS,IERR)
C
      INTEGER   I,J,K,L,M,N,EN,II,JJ,LL,MM,NA,N
     X          ITS,MP2,ENM2,IERR
      REAL*8 H(NDIM,N),Z(NDIM,N)
      REAL*8 P,Q,R,S,T,W,X,Y,RA,SA,VI,VR,ZZ,N
      REAL*8 DSQRT,DABS,DSIGN
      INTEGER MIN0
      LOGICAL NOTLAS
      COMPLEX*16 Z3
      COMPLEX*16 DCMPLX
      REAL*8 DREAL,DIMAG
C
C     STATEMENT FUNCTIONS ENABLE EXTRACTION OF
C     IMAGINARY PARTS OF DOUBLE PRECISION COMP
C
      DREAL(Z3) = Z3
      DIMAG(Z3) = (0.0D0,-1.0D0) * Z3
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE
C     NUM. MATH. 16, 181-204(1970) BY PETERS
C     HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR
C     IT HAS BEEN FURTHER MODIFIED BY STEPHEN
C     COMPUTES THE SCHUR FORM OF THE UPPER-HES
C
C     ON INPUT:
C
C        NDIM MUST BE SET TO THE ROW DIMENSION
C          ARRAY PARAMETERS AS DECLARED IN THE
C          DIMENSION STATEMENT;
C
C        N IS THE ORDER OF THE MATRIX;
C
C        H CONTAINS THE UPPER HESSENBERG MATR
C
C        Z CONTAINS THE TRANSFORMATION MATRIX
C          AFTER THE REDUCTION BY ORTHES.
C
C     ON OUTPUT:
C
C        H HAS BEEN DESTROYED;
C
C        Z CONTAINS THE TRANSFORMATION MATRIX
C
C        IERR IS SET TO
C          ZERO       FOR NORMAL RETURN,
C          J          IF THE J-TH EIGENVALUE
C                     DETERMINED AFTER 30 ITE
C
C     ----------------------------------------
C
C     EPS IS A MACHINE DEPENDENT PARAMETER SP
C     THE RELATIVE PRECISION OF FLOATING POIN
C
      IERR = 0
      NORM = 0.0D0
      K = 1
C
C     COMPUTE MATRIX NORM
C
```

```
      DO 50 I = 1, N
C
         DO 40 J = K, N
   40    NORM = NORM + DABS(H(I,J))
C
         K = I
   50 CONTINUE
C
      EN = N
      T = 0.0D0
C
C     SEARCH FOR NEXT EIGENVALUES
C
   60 IF (EN .LT. 1) GO TO 340
      ITS = 0
      NA = EN - 1
      ENM2 = NA - 1
C
C     LOOK FOR SINGLE SMALL SUB-DIAGONAL EL
C     FOR L=EN STEP -1 UNTIL 1 DO --
C
   70 DO 80 LL = 1, EN
         L = EN + 1 - LL
         IF (L .EQ. 1) GO TO 100
         S = DABS(H(L-1,L-1)) + DABS(H(L,L)
         IF (S .EQ. 0.0D0) S = NORM
         IF (DABS(H(L,L-1)) .LE. EPS * S) G
   80 CONTINUE
C
C     FORM SHIFT
C
  100 X = H(EN,EN)
      IF (L .EQ. EN) GO TO 270
      Y = H(NA,NA)
      W = H(EN,NA) * H(NA,EN)
      IF (L .EQ. NA) GO TO 280
      IF (ITS .EQ. 30) GO TO 1000
      IF (ITS .NE. 10 .AND. ITS .NE. 20) GO
C
C     FORM EXCEPTIONAL SHIFT
C
      T = T + X
C
      DO 120 I = 1, EN
  120 H(I,I) = H(I,I) - X
C
      S = DABS(H(EN,NA)) + DABS(H(NA,ENM2)
      X = 0.75D0 * S
      Y = X
      W = -0.4375D0 * S * S
  130 ITS = ITS + 1
C
C     LOOK FOR TWO CONSECUTIVE SMALL
C     SUB-DIAGONAL ELEMENTS.
C     FOR M=EN-2 STEP -1 UNTIL L DO --
C
      DO 140 MM = L, ENM2
         M = ENM2 + L - MM
         ZZ = H(M,M)
         R = X - ZZ
```

```
.EMENT

))
GO TO 100

O TO 130

)
```

```
)

(R)) .LE. EPS * DABS(P)
BS(H(N+1,M+1)))) GO TO 150

ND
```

```
               S = Y - ZZ
               P = (P * S - W) / H(M+1,M) + H(M,M+1
               Q = H(M+1,M+1) - ZZ - R - S
               R = H(M+2,M+1)
               S = DABS(P) + DABS(Q) + DABS(R)
               P = P / S
               Q = Q / S
               R = R / S
               IF (M .EQ. L) GO TO 150
               IF (DABS(H(M,M-1)) * (DABS(Q) + DABS
     X           * (DABS(H(M-1,M-1)) + DABS(ZZ) + DA
  140       CONTINUE
C
  150       MP2 = M + 2
C
            DO 160 I = MP2, EN
               H(I,I-2) = 0.0D0
               IF (I .EQ. MP2) GO TO 160
               H(I,I-3) = 0.0D0
  160       CONTINUE
C
C           DOUBLE QR STEP INVOLVING ROWS L TO EN A
C           COLUMNS M TO EN
C
            DO 260 K = M, NA
               NOTLAS = K .NE. NA
               IF (K .EQ. M) GO TO 170
               P = H(K,K-1)
               Q = H(K+1,K-1)
               R = 0.0D0
               IF (NOTLAS) R = H(K+2,K-1)
               X = DABS(P) + DABS(Q) + DABS(R)
               IF (X .EQ. 0.0D0) GO TO 260
               P = P / X
               Q = Q / X
               R = R / X
  170          S = DSIGN(DSQRT(P*P+Q*Q+R*R),P)
               IF (K .EQ. M) GO TO 180
               H(K,K-1) = -S * X
               GO TO 190
  180          IF (L .NE. M) H(K,K-1) = -H(K,K-1)
  190          P = P + S
               X = P / S
               Y = Q / S
               ZZ = R / S
               Q = Q / P
               R = R / P
C
C              ROW MODIFICATION
C
               DO 210 J = K, N
                  P = H(K,J) + Q * H(K+1,J)
                  IF (.NOT. NOTLAS) GO TO 200
                  P = P + R * H(K+2,J)
                  H(K+2,J) = H(K+2,J) - P * ZZ
  200             H(K+1,J) = H(K+1,J) - P * Y
                  H(K,J) = H(K,J) - P * X
  210          CONTINUE
C
               J = MIN0(EN,K+3)
```

```
C
C          COLUMN MODIFICATION
C
           DO 230 I = 1, J
              P = X * H(I,K) + Y * H(I,K+1)
              IF (.NOT. NOTLAS) GO TO 220
              P = P + ZZ * H(I,K+2)
              H(I,K+2) = H(I,K+2) - P * R
  220         H(I,K+1) = H(I,K+1) - P * Q
              H(I,K) = H(I,K) - P
  230      CONTINUE
C
C          ACCUMULATE TRANSFORMATIONS
C
           DO 250 I = 1, N
              P = X * Z(I,K) + Y * Z(I,K+1)
              IF (.NOT. NOTLAS) GO TO 240
              P = P + ZZ * Z(I,K+2)
              Z(I,K+2) = Z(I,K+2) - P * R
  240         Z(I,K+1) = Z(I,K+1) - P * Q
              Z(I,K) = Z(I,K) - P
  250      CONTINUE
C
  260   CONTINUE
C
        GO TO 70
C
C       ONE ROOT FOUND
C
  270   H(EN,EN) = X + T
        EN = NA
        GO TO 60
C
C       TWO ROOTS FOUND
C
  280   P = (Y - X) / 2.0D0
        Q = P * P + W
        ZZ = DSQRT(DABS(Q))
        H(EN,EN) = X + T
        X = H(EN,EN)
        H(NA,NA) = Y + T
        IF (Q .LT. 0.0D0) GO TO 320
C
C       REAL PAIR
C
        ZZ = P + DSIGN(ZZ,P)
        X = H(EN,NA)
        S = DABS(X) + DABS(ZZ)
        P = X / S
        Q = ZZ / S
        R = DSQRT(P*P+Q*Q)
        P = P / R
        Q = Q / R
C
C       ROW MODIFICATION
C
        DO 290 J = NA, N
           ZZ = H(NA,J)
           H(NA,J) = Q * ZZ + P * H(EN,J)
           H(EN,J) = Q * H(EN,J) - P * ZZ
```

```
)

,MDIM,NDIM,D)

,N2
T(M),C(1),G

 THE INTEGER VARIABLES
E) OR V(TRANSPOSE)

ED, IF ITX = 1, THEN

(THE TRANSFORMATION
HE LOWER PART OF A
EE SUBROUTINE ORTHES)
AS FOR THE SUBROUTINE
ACE OF ORT.

-----------------------

TINE ORTRAN)
```

```
  250   CONTINUE
C
C       COLUMN MODIFICATION
C
        DO 300 I = 1, EN
           ZZ = H(I,NA)
           H(I,NA) = C * ZZ + P * H(I,EN)
           H(I,EN) = Q * H(I,EN) - P * ZZ
  300   CONTINUE
C
C       ACCUMULATE TRANSFORMATIONS
C
        DO 310 I = 1, N
           ZZ = Z(I,NA)
           Z(I,NA) = C * ZZ + P * Z(I,EN)
           Z(I,EN) = Q * Z(I,EN) - P * ZZ
  310   CONTINUE
C
        GO TO 330
C
C       COMPLEX PAIR
C
  320   CONTINUE
  330   EN = ENM2
        GO TO 60
C
C       ALL ROOTS FOUND.  BACKSUBSTITUTE TO FIND
C       VECTORS OF UPPER TRIANGULAR FORM
C
 1000   IERR = EN
  340   RETURN
        END
C
C
C
C
C
        SUBROUTINE TRANSF(A,ORT,IT1,C,V,IT2,M,N
C
        INTEGER I,IT1,IT2,J,K,K1,K2,KK,M,MDIM,N
        REAL*8 V(NDIM,N),C(MDIM,N),A(MDIM,M),OR
C
C       SUBROUTINE TO FORM C(NEW) = U * C * V.
C       IT1 AND IT2 INDICATE WHETHER U(TRANSPOS
C       ARE REQUIRED INSTEAD OF U OR V.
C       IF ITX = 0 THEN THE ACTUAL MATRIX IS US
C       ITS TRANSPOSE IS USED.
C       IN ORDER TO SAVE STORAGE, THE MATRIX U
C       MATRIX FOR THE MATRIX A) IS STORED IN T
C       AND IN THE VECTOR ORT IN CODED FORM. (S
C       THE PARAMETERS FOR THIS SUBROUTINE ARE
C       AXXBC EXCEPT THAT HERE ORT TAKES THE FL
C
C       ----------------------------------------
C
C       FORM U * C AND STORE IN C (AS IN SUBROU
C
        M2 = M - 2
C
C       CHECK FOR SMALL MATRIX
```

```
C
      IF (M2.LE.0) GO TO 45
      DO 40 KK = 1,M2
         K = M2 - KK + 1
         IF (IT1.EQ.1) K = KK
         K1 = K + 1
         IF (A(K1,K).EQ.0.D0) GO TO 40
         D(K1) = ORT(K1)
         K2 = K + 2
         DO 10 I = K2,M
            D(I) = A(I,K)
10       CONTINUE
         DO 30 J = 1,N
            G = 0.D0
            DO 20 I = K1,M
               G = G + D(I) * C(I,J)
20          CONTINUE
            G = G / ORT(K1) / A(K1,K)
            DO 30 I = K1,M
            C(I,J) = C(I,J) + G * D(I)
30       CONTINUE
40    CONTINUE
C
C     FORM C(NEW) = (U * C) * V
C
45    DO 60 I = 1,M
         DO 50 J = 1,N
            D(J) = 0.D0
            DO 50 K = 1,N
               IF (IT2.EQ.0) D(J) = D(J) + (
               IF (IT2.EQ.1) D(J) = D(J) + (
50          CONTINUE
            DO 60 J = 1,N
               C(I,J) = D(J)
60    CONTINUE
      RETURN
      END
C
C
C
C
C
      SUBROUTINE NSOLVE(A,B,C,D,MDIM,M,NDIM,
C
      INTEGER I,I1,IERR,IND,IPR(1),IROW1,J,
      REAL*8 A(MDIM,M),B(NDIM,N),C(MDIM,N),(
C
C     THIS SUBROUTINE SETS UP AND SOLVES (W
C     THE SUBROUTINE HESOLV) A SINGLE SYSTEM
C
C      ( A + B(IND+1,IND+1) * I )(X(IND+1))
C
C     (X(I),C(I) - COLUMN I OF THE MATRICES
C     AND STORES THE RESULT IN THE APPROPRI
C     MATRIX C.  THE UPPER-HESSENBERG MATRI
C     VECTOR D IN ORDER TO SAVE STORAGE (EN
C     DIAGONAL ARE IGNORED).
C     THE RIGHT HAND SIDE IS ALSO STORED IN
C     TO REDUCE THE NUMBER OF VECTORS USED
C     THE PARAMETERS ARE AS IN THE SUBROUTI
```

```
C(I,K) * V(K,J)
C(I,K) * V(J,K)








,M,INC,IPR,REPS,IERR)

V,M1,MDIM,N,NDIM,MFIN
C(1),REPS

ITH THE HELP OF
M OF EQUATIONS

 = (C(INC+1))

 X AND C)
ATE ROW OF THE
X IS STORED IN THE
TRIES BELOW THE SUB-

 THE VECTOR D IN ORDER
IN THE SUBROUTINES.
NE AXXBC.
```

```
---------------------.
ECESSARY
C,N,M,MDIM,NDIM)

WITHIN THE VECTOR D

)

ORE THE RESULT IN C.

)
,J1,K,N,N1,MFIN

ENBERG SYSTEM OF EQUATIONS
HE VECTOR C.  THE METHOD USED
L PIVOTING.  THE PARAMETERS

IGHT HAND SIDE (ON INPUT),

RDS INTERCHANGES
```

```
C
C     ------------------------------------------
C
C     PERFORM BLOCK BACK-SUBSTITUTION IF N
C
      IF (IND.LT.N - 1) CALL BACKSB(C,B,IN
C
C     SET UP THE SYSTEM OF EQUATIONS
C
      MFIN = (M * (M + 1)) / 2 + M
      DO 20 I = 1,M
C
C        FIND INDEX OF BEGINNING OF ROW I
C
         M1 = IROW1(I,M)
         I1 = I - 1
         IF (I.EQ.1) I1 = 1
         DO 10 J = I1,M
            D(M1+J-I1+1) = A(I,J)
10       CONTINUE
         J = 2
         IF (I.EQ.1) J = 1
         D(M1+J) = D(M1+J) + B(IND+1,IND+1
C
C        STORE THE RIGHT HAND SIDE
C
         D(MFIN+I) = C(I,IND+1)
20    CONTINUE
C
C     SOLVE THE SYSTEM OF EQUATIONS AND ST
C
      CALL HESOLV(D,IPR,M,REPS,IERR)
      IF (IERR.NE.0) GO TO 40
      DO 30 I = 1,M
         C(I,IND+1) = D(IPR(I))
30    CONTINUE
      IND = IND - 1
      RETURN
40    IERR = -IND - 1
      RETURN
      END
C
C
C
C
C
      SUBROUTINE HESOLV(D,IPR,M,REPS,IERR
C
      INTEGER I,I1,I2,IERR,IPR(1),IROW1,J
      REAL*8 D(1),DABS,MULT,REPS
C
C     THIS SUBROUTINE SOLVES AN UPPER-HES
C     WHICH IS STORED IN PACKED FORM IN T
C     IS GAUSSIAN ELIMINATION WITH PARTIA
C     IN THE SUBROUTINE ARE:
C     D        -  MATRIX OF COEFFICIENTS, I
C                 AND SOLUTION (ON OUTPUT)
C     IPR      -  INTEGER VECTOR WHICH REC(
C     M        -  SIZE OF SYSTEM
C     REPS     -  ERROR TOLERANCE
```

```
C      IERR   -  ERROR CODE (IERR=0 => 
C                           (IERR=-1 =>S
C
C      ------------------------------------
C
C      INITIALIZE INTERCHANGE VECTORS AN
C
       IERR = 0
       MFIN = (M * (M + 1)) / 2 + M
       DO 10 I = 1,M
          IPR(M+I) = IFCW1(I,M)
          IPR(I) = I + MFIN
10     CONTINUE
       M1 = M - 1
C
C      REDUCE THE MATRIX TO UPPER TRIANG
C
       IF (M.EQ.1) GO TO 35
       DO 30 I = 1,M1
          IF (DABS(D(IPR(M+I)+1)).GT.DAB
C
C         INTERCHANGE ROWS IF NECESSARY
C
          K = IPR(M+I)
          IPR(M+I) = IPR(M+I+1)
          IPR(M+I+1) = K
          K = IPR(I)
          IPR(I) = IPR(I+1)
          IPR(I+1) = K
C
C         CHECK FOR COMPUTATIONALLY SIN(
C
20        IF (DABS(D(IPR(M+I)+1)).LT.RE
          IPR(M+I+1) = IPR(M+I+1) + 1
C
C         ELIMINATE SUBDIAGONAL ELEMENT
C
          MULT = D(IPR(M+I+1)) / D(IPR(
          D(IPR(I+1)) = D(IPR(I+1)) - M
          I1 = I + 1
          DO 30 J = I1,M
             D(IPR(M+I+1)+J-I) = D(IPR(
     *                                 MULT
30     CONTINUE
35     IF (DABS(D(IPR(M+M)+1)).LT.REPS!
C
C      PERFORM BACK - SUBSTITUTION
C
       D(IPR(M)) = D(IPR(M)) / D(IPR(M
       IF (M1.EQ.0) RETURN
       DO 50 I1 = 1,M1
          I = M - I1
          I2 = I + 1
          MULT = 0.D0
          DO 40 J1 = I2,M
             J = J1 - I2 + 2
             MULT = MULT + D(IPR(J1))
40        CONTINUE
          D(IPR(I)) = (D(IPR(I)) - MUL
50     CONTINUE
```

```
 FNAL RETURN)
 INGULAR MATRIX)

 ---------------------------

 : PARAMETERS.






ULAR FORM


S(D(IPR(M+I+1)+1))) GO TO 20







GULAR MATRIX

PS) GO TO 60

 CF A

M+I)+1)
ULT * D(IPR(I))

M+I+1)+J-I) -
*D(IPR(M+I)+J+1-I)

)GO TO 60


M)+1)





*D(IPR(M+I)+J)

T / D(IPR(M+I)+1)
```

```
NDIM)

NDIM

ROW'.
EC.

1,I) * C(J,I)

CR THE VECTOR D) ONE BEFORE
ATRIX.

 2

IM,M,IND,IPR,REPS,IERR)

,J,J1,J2,K,LROW2,M,M1,

),C(1),REFS

STEM OF EQUATIONS.
 THE SET OF EQUATIONS

)  )( X(IND)  )    ( C(IND)  )
   )(         ) =  (         )
I  )(X(IND+1))     (C(IND+1))

TO INSURE THAT THERE ARE
 AND THEN USING GAUSSIAN
```

```
      RETURN
60    IERR = -1
      RETURN
      END
C
C
C
C
      SUBROUTINE BACKSB(C,B,IND,N,M,NDIM,
C
      INTEGER I,IND,IND1,IND2,J,M,MDIM,N,
      REAL*8 B(NDIM,N),C(MDIM,N)
C
C     BLOCK BACK - SUBSTITUTION FOR ONE '
C     PARAMETERS ARE AS IN SUBROUTINE AXX
C
      IND1 = IND + 1
      IND2 = IND + 2
      DO 10 I = IND2,N
         DO 10 J = 1,M
            C(J,IND1) = C(J,IND1) - B(IND
10    CONTINUE
      RETURN
      END
C
C
C
C
C
      INTEGER FUNCTION IROW1(I,M)
C
      INTEGER I,M
C
C     THIS FUNCTION COMPUTES THE INDEX (F
C     THE BEGINNING OF ROW I IN THE M*M M
C
      IROW1 = (I-1) * M - (I-2) * (I-3) /
      IF (I.EQ.1) IROW1 = 0
      RETURN
      END
C
C
C
C
C
      SUBROUTINE N2SOLV(A,B,C,D,NDIM,N,MD
C
      INTEGER I,I1,IERR,IND,IER(1),IROW2,
     *   MDIM,N,NDIM,MFIN
      REAL*8 A(MDIM,M),B(NDIM,N),C(MDIM,N
C
C     THIS SUBROUTINE SETS UP A DOUBLE SY
C     IT SOLVES (USING SUBROUTINE H2SOLV)
C
C     ( A+B(IND,IND)*I         B(IND,IND+1
C     (
C     ( B(IND+1,IND)*I   A+B(IND+1,IND+1)*
C
C     BY INTERCHANGING ROWS AND COLUMNS (
C     ZEROES BELOW THE SUB-SUB-DIAGONAL)
```

```
C     ELIMINATION.  IN ORDER TO TAKE ADVAN
C     ZEROES IN THIS MATRIX, THE MATRIX IS
C     D IN PACKED FORM. (ENTRIES BELOW THE
C     IGNORED) THE RIGHT HAND SIDE AND SOL
C     THE PARAMETERS ARE AS IN THE SUBROU
C
C     ------------------------------------
C
C     PERFORM BLOCK BACK-SUBSTITUTION IF N
C
      IF (IND.LT.N-1) CALL BACKS2(C,B,IND
C
C     SET UP THE SYSTEM OF EQUATIONS FOR N
C
      M2 = 2 * M
      MFIN = (M2 * (M2 + 1)) / 2 + 4 * M
      DO 20 I = 1,M
C
C        FIND BEGINNING AND LENGTH OF ROW
C
         M1 = IROW2(2*I-1,M)
         K = LROW2(2*I-1,M)
         I1 = I - 1
         IF (I.EQ.1) I1 = 1
         DO 10 J = I1,M
            J1 = 2 * (J - I1 + 1)
            J2 = 1
            IF (M1.EQ.0) J2 = 0
            D(M1+J1-1) = A(I,J)
            D(M1+J1) = 0.D0
            D(M1+K+J1-J2) = A(I,J)
            D(M1+K+J1-1-J2) = 0.D0
   10    CONTINUE
         J1 = 3
         IF (I.EQ.1) J1 = 1
         D(J1+M1) = D(J1+M1) + B(IND,IND)
         D(J1+M1+1) = D(J1+M1+1) + B(IND,
         IF (I.NE.1) J1 = 2
         D(J1+M1+K) = D(J1+M1+K) + B(IND+
         D(J1+M1+K+1) = D(J1+M1+K+1) + B(
C
C        STORE RIGHT HAND SIDE
C
         D(2*I+MFIN) = C(I,IND+1)
         D(2*I-1+MFIN) = C(I,IND)
   20 CONTINUE
C
C     SOLVE THE SYSTEM OF EQUATIONS AND ST
C
      CALL H2SOLV(D,IPR,M,REPS,IERR)
      IF (IERR.NE.0) GO TO 40
      DO 30 I = 1,M
         C(I,IND) = D(IPR(2*I-1))
         C(I,IND+1) = D(IPR(2*I))
   30 CONTINUE
      IND = IND - 2
      RETURN
   40 IERR = -IND - 1
      RETURN
      END
```

```
TAGE OF THE NUMBER OF
E STORED IN THE VECTOR
E SUB-SUB-DIAGONAL ARE
LUTION ARE ALSO STORED IN D.
INE AXXBC.

------------------------

NECESSARY

,N,M,MDIM,NDIM)

-2SOLV



 2I-1 IN THE VECTOR D.



IND+1)

1,IND)
IND+1,IND+1)



TORE THE RESULT BACK IN C
```

```
,K,K1,L,N,N2,N21,MFIN

* 2M EQUATIONS
SUB - SUB - DIAGONAL
CTOR IN PACKED FORM.
N.

T HAND SIDE (ON INPUT).

RMUTATIONS

  RETURN)
LAR MATRIX)

--------------------------

METERS.

MAX) GO TO 20

MATRIX
```

```
C
C
C
C
C
      SUBROUTINE H2SOLV(D,IPR,M,REPS,IERR)
C
      INTEGER I,I1,I2,IERR,IPR(1),IROW2,J,J1
      REAL*8 D(1),MAX,DABS
C
C     THIS SUBROUTINE SOLVES A SYSTEM OF 2M
C     WHERE THE MATRIX HAS ZEROES BELOW THE
C     AND WHERE THE MATRIX IS STORED AS A VE
C     THE METHOD USED IS GAUSSIAN ELIMINATIC
C     PARAMETERS:
C     D        -   MATRIX OF COEFFICIENTS, RIGH
C                  AND SOLUTION (ON OUTPUT).
C     IPR      -   VECTOR USED TO RECORD ROW PE
C     M        -   SIZE OF SYSTEM
C     REPS     -   ERROR TOLERANCE
C     IERR     -   ERROR CODE (IERR=0 => NORMAL
C                            (IERR=-1 => SINGU
C
C     -----------------------------------------
C
C     INITIALIZE PIVOT VECTORS AND SIZE PARA
C
      IERR = 0
      M2 = 2 * M
      MFIN = (M2 * (M2 + 1)) / 2 + 4 * M
      DO 10 I = 1,M2
         IPR(M2+I) = IROW2(I,M)
         IPR(I) = I + MFIN
   10 CONTINUE
      M21 = M2 - 1
C
C     TRANSFORM MATRIX TO TRIANGULAR FORM
C
      DO 40 I = 1,M21
         I1 = 2
         IF (I.EQ.M21) I1 = 1
         L = 0
C
C        INTERCHANGE ROWS IF NECESSARY
C
         MAX = DABS(D(IPR(M2+I) + 1))
         DO 20 J = 1,I1
            IF (DABS(D(IPR(M2+J+I) + 1)).LE.
            MAX = DABS(D(IPR(M2+J+I) + 1))
            L = J
   20    CONTINUE
C
C        CHECK FOR COMPUTATIONALLY SINGULAR
C
         IF (MAX.LE.REPS) GO TO 80
         IF (L.EQ.0) GO TO 30
         K = IPR(M2+I)
         IPR(M2+I) = IPR(M2+L+I)
         IPR(M2+L+I) = K
         K = IPR(I)
```

```
                  IPR(I) = IPR(I+L)
                  IPR(I+L) = K
   30             CONTINUE
C
C                 ADJUST POINTERS TO BEGINNINGS
C
                  IPR(M2+I+1) = IPR(M2+I+1) + 1
                  IF (I.NE.M21) IPR(M2+I+2) = I
                  IP1 = I + 1
C
C                 ELIMINATE SUBDIAGONAL ELEMENT
C
                  DO 40 J = 1,I1
                     MAX = D(IPR(M2+I+J)) / D(I
                     D(IPR(I+J)) = D(IPR(I+J))
                     DO 40 K1 = IP1,M2
                        K = K1 - I
                        D(IPR(M2+I+J)+K) = D(IP
     *
   40         CONTINUE
              IF (DABS(D(IPR(M2+M2)+1)).LE.REP
C
C             PERFORM BACK SUBSTITUTION
C
              D(IPR(M2)) = D(IPR(M2)) / D(IPR(
              DO 60 I1 = 1,M21
                  I = M2 - I1
                  I2 = I + 1
                  MAX = 0.D0
                  DO 50 J1 = I2,M2
                     J = J1 - I2 + 2
                     MAX = MAX + D(IPR(J1)) * D
   50             CONTINUE
                  D(IPR(I)) = (D(IPR(I)) - MAX)
   60         CONTINUE
   70         RETURN
   80         IERR = -1
              GO TO 70
              END
C
C
C
C
C
              SUBROUTINE BACKS2(C,B,IND,N,M,MD
C
              INTEGER I,IND,IND1,IND2,J,M,MDIM
              REAL*8 B(NDIM,N),C(MDIM,N)
C
C             BLOCK BACK - SUBSTITUTION FOR TW
C             PARAMETERS ARE AS IN SUBROUTINE
C
              IND1 = IND + 1
              IND2 = IND + 2
              DO 10 I = IND2,N
                 DO 10 J = 1,M
                    C(J,IND1) = C(J,IND1) - B(
                    C(J,IND) = C(J,IND) - B(IN
   10         CONTINUE
              RETURN
```

```
CF ROWS

PR(M2+I+2) + 1

S IN THE MATRIX

PR(M2+I)+1)
- MAX * D(IPR(I))

R(M2+I+J)+K) -
     MAX * D(IPR(M2+I)+1+K)

S) GC T] 80

M2+M2)+1)

(IPR(M2+I)+J)

 / C(IPR(M2+I)+1)

IM,NDIM)

,N,NCIM

] 'RCWS'.
AXXBC.

INC1,I) * C(J,I)
D,I) * C(J,I)
```

```
R THE VECTOR D) ONE
'HE 2N*2N MATRIX.
```

```
ROW I IN THE
IN PACKED FORM IN THE
```

```
      END
C
C
C
C
C
      INTEGER FUNCTION IROW2(I,M)
C
      INTEGER I,K,M
C
C     THIS FUNCTION FINDS THE INDEX (FOR
C     BEFORE THE BEGINNING OF ROW I  IN T
C
      IROW2 = 2 * (I - 1) * M
      K = (I - 4) * (I - 3) / 2
      IF (I.LE.2) K = 0
      IROW2 = IROW2 - K
      RETURN
      END
C
C
C
C
C
      INTEGER FUNCTION LROW2(I,M)
C
      INTEGER I,K,M
C
C     THIS FUNCTION FINDS THE LENGTH OF
C     DOUBLE MATRIX (WHEN IT IS STORED I
C     VECTOR D)
C
      K = I - 3
      IF (I.LE.2) K = 0
      LROW2 = 2 * M - K
      IF (I.LE.2) LROW2 = 2 * M
      RETURN
      END
```