

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.

Authors' 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 $A \in \mathbb{R}^{m \times m}$ and $B \in \mathbb{R}^{n \times n}$ be given matrices and define the linear transformation $\phi: \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$ by

$$(1.1) \quad \phi(X) = AX + XB$$

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

$$(1.2) \quad \phi(X) = AX + XB = C$$

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 $A_1 = U^{-1}AU$ and $B_1 = V^{-1}BV$.

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 3 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

$$u = 3^{-t}$$

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

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

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

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

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

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

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, $\|x_2(U) + x_2(V)\| \approx 10^{-3}$ 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 $U^T U = I$ implies $x_2(U) = 1$.) 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

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

A matrix is quasi-upper triangular if it is upper triangular with the possible exception of 2×2 "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 2×2 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

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

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

F into their respective columns

$$Y = [y_1 \mid y_2 \mid \dots \mid y_n] \quad F = [f_1 \mid f_2 \mid \dots \mid f_n]$$

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

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

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

Thus, y_k is the solution of an $m \times m$ 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

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

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

system

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

where

$$(2.6) \quad p_{ij} = \begin{cases} \begin{bmatrix} r_{ij} & 0 \\ 0 & r_{ij} \end{bmatrix} & (i \neq j) \\ \begin{bmatrix} r_{ii} + s_{k-1,k-1} & s_{k-1,k} \\ s_{k,k-1} & r_{ii} + s_{kk} \end{bmatrix} & (i = j) \end{cases}$$

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

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

Since $R = (r_{ij})$ is quasi-upper triangular, the above $2m \times 2m$ matrix is upper triangular with the possible exception of 4×4 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 2×2 bumps.

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

1. Compute the real Schur decompositions:

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

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

2. Update the righthand side:

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

3. Back substitute for Y :

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

4. Obtain solution:

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

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

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 (rxn) 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

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

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

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

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

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

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

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

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

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

$$(3.4) \quad p_{ij} = \begin{cases} \begin{bmatrix} h_{ij} & 0 \\ 0 & h_{ij} \end{bmatrix} & (i \neq j) \\ \begin{bmatrix} h_{ii} + s_{k-1,k-1} & s_{k-1,k} \\ s_{k,k-1} & h_{ii} + s_{kk} \end{bmatrix} & (i = j) \end{cases}$$

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 $m \times m$ array for U as in the Bartels-Stewart algorithm. Summarizing the Hessenberg-Schur algorithm and the associated operation counts we have:

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

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

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

2. Update the righthand side:

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

3. Back substitute for Y :

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

4. Obtain solution:

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

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

To obtain the operation count associated with the determination of Y , we assumed that S has $\frac{n}{2}$ 2×2 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

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

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 ($m \times m$) for the original A and subsequently H and U
- B ($n \times n$) for the original B and subsequently S
- V ($n \times n$) for the orthogonal matrix V
- C ($m \times n$) 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 $\psi(X) = C$. This system of equations can be written in the form

$$(4.1) \quad Px = c$$

where

$$(4.2) \quad P = (I_n \otimes A) + (B^T \otimes I_m)$$

and

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

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

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 $R^{m \times n}$:

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

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 $u \| A \|_F \| A \|_F \| B \|_F$, 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

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

where

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

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

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

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 $k \times k$ matrices and let x , Δx , d , and Δd , be k -vectors. Assume that M is nonsingular and d nonzero. If

$$(4.7) \quad Mx = d$$

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

and

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

then

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

Proof

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

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

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

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

then

$$(4.12) \quad \frac{\|x - \hat{x}\|_F}{\|x\|_F} \leq 4u [\|A\|_F + \|B\|_F] \| \phi^{-1} \|$$

Proof

Defining $x = \text{vec}(X)$, $\hat{x} = \text{vec}(\hat{X})$, $c = \text{vec}(C)$, $g = \text{vec}(G)$, $P = (I_n \otimes A) + (B^T \otimes I_m)$, 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

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

The theorem follows since

$$\|g\|_2 \leq u\|c\|_F \leq u(\|A\|_F + \|B\|_F) \|x\|_2$$

For the 2×2 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

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

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

$$\hat{AX} + \hat{XB} = C + \begin{bmatrix} u & 0 \\ 0 & 0 \end{bmatrix}$$

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 $u \cdot \phi^{-1}$ 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 associated with the Hessenberg-Schur algorithm. This amounts to applying some standard results from Wilkinson [12]. His inverse error analysis pertaining to orthogonal matrices can be applied to the computation $H = U^T A U$, $S = V^T B^T V$, $F = U^T C V$, and $X = U Y V^T$ while his well-known analysis of Gaussian elimination and back-substitution can be used in connection with the determination of Y . (Y is essentially obtained by using Gaussian elimination and back substitution to solve the system $((I_n \otimes H) + (S \otimes I_m))\text{vec}(Y) = \text{vec}(F)$.) Denoting computed quantities with the "hat" notation, we can account for the rounding errors in the Hessenberg-Schur algorithm in the following way:

- (5.1) $\hat{U} = U_1 + E_U$ $U_1^T U_1 = I$, $\|E_U\|_F \leq \epsilon$
(5.2) $\hat{V} = V_1 + E_V$ $V_1^T V_1 = I$, $\|E_V\|_F \leq \epsilon$
(5.3) $\hat{H} = U_1^T (A + E_1) U_1$ $\|E_1\|_F \leq \epsilon \|A\|_F$
(5.4) $\hat{S} = V_1^T (B + E_2)^T V_1$ $\|E_2\|_F \leq \epsilon \|B\|_F$
(5.5) $\hat{F} = U_1^T (C + E_3) V_1$ $\|E_3\|_F \leq \epsilon \|C\|_F$
(5.6) $(\hat{T} + E_4)\hat{y} = \hat{f}$ $\|E_4\|_2 \leq \epsilon \|\hat{T}\|_2$
(5.7) $\hat{X} = U_1 (\hat{Y} + E_5) V_1^T$ $\|E_5\|_F \leq \epsilon \|\hat{Y}\|_F$

where

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

$$(5.9) \quad \hat{y} = \text{vec}(\hat{Y})$$

$$(5.10) \quad \hat{f} = \text{vec}(\hat{F})$$

and c 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

$$(5.11) \quad \frac{\|x - \hat{x}\|_F}{\|x\|_F} \leq \frac{\|W - \hat{Y}\|_F}{\|W\|_F} + c \frac{\|\hat{Y}\|_F}{\|W\|_F}$$

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

$$(5.12) \quad T W = f$$

where $w = \text{vec}(W)$, $f = \text{vec}(U_1^T C V_1)$ 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,

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

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

$$\|\Delta T\|_2 \leq c(2+c) (\|A\|_F + \|B\|_F)$$

and with (5.5) and (5.10) that

$$\|\Delta F\|_2 \leq c\|C\|_F \leq c(\|A\|_2 + \|B\|_F)\|W\|_F$$

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

$$(5.14) \quad \|\phi^{-1}\| c(2+c) (\|A\|_2 + \|B\|_2) < 1/2$$

then the Lemma is applicable and we find

$$\frac{\|W - \hat{Y}\|_F}{\|W\|_F} = \frac{\|W - \hat{Y}\|_2}{\|W\|_2} \leq (6c + 2c^2) \|\phi^{-1}\| (\|A\|_F + \|B\|_F)$$

Furthermore, (5.14) can also be used to show

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

and therefore from (5.11) we have

$$(5.15) \quad \frac{\|X - \hat{X}\|_F}{\|X\|_F} \leq (9c + 2c^2) \|\phi^{-1}\| (\|A\|_F + \|B\|_F)$$

Inequality (5.15) indicates that errors no worse in magnitude than $O(\| \delta^{-1} \| \epsilon)$ 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

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

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

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

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

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 $U^T C V$ and $U Y V^T$ where U and V are orthogonal.
- NSOLVE
HESOLV - These routines combine to solve upper Hessenberg systems using Gaussian elimination with partial pivoting.
- BACKSB
- N2SOLV
H2SOLV - These routines combine to solve the $2m$ -by- $2m$ block Hessenberg systems encountered whenever S has a 2-by-2 bump.
- BACKSB

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 25×25 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	$\frac{W_{HS}}{W_{BS}}(m,n)$	HS Execution time 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

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

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

where

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

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 ($\epsilon = 10^{-16}$), we obtained the following results for an $m = 10$, $n = 4$ problem:

t	$\ \phi^{-1}\ $	$\ X - \hat{X}\ _F$	$\frac{\ A\hat{X} + \hat{X}B - C\ _F}{\ X\ _F (\ A\ _F + \ B\ _F)}$
		$\ X\ _F$	
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

$$(7.1) \quad AXM + X = C$$

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

$$U^T A U = H \quad U^T U = I, \quad H \text{ upper Hessenberg}$$

and

$$V^T M^T V = S \quad V^T V = I, \quad S \text{ quasi-upper triangular}$$

and $F = U^T C V$, then (7.1) transforms to

$$(7.2) \quad H Y S^T + Y = F$$

where $Y = U^T X V$. As in the Hessenberg-Schur algorithm, once y_{k+1}, \dots, 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

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

The presence of 2×2 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

$$(7.3) \quad AXM + LXB = C$$

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

$$Q^T A U = P \quad (\text{quasi- upper triangular})$$

$$Q^T L U = R \quad (\text{upper triangular})$$

$$Z^T B^T V = S \quad (\text{quasi-upper triangular})$$

$$Z^T M^T V = T \quad (\text{upper triangular})$$

If $Y = U^T X V$ and $F = Q^T C Z$, 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

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

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,B), the QZ algorithm first computes orthogonal Q and U such that $Q^T A U = H$ is upper Hessenberg and $Q^T L U = R$ 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 2×2 bumps on the diagonal of S pose no serious difficulties.

3. 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 $A^T S + SA + Q = 0$ ", 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 - EISPACK 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.

1.
2.
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.
25.
26.
27.
28.
29.
30.
31.
32.
33.
34.
35.
36.
37.
38.
39.
40.
41.
42.
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.
59.

SUBROUTINE AXXRC (NDIM,N,MDIM,M,A,V
C
1. C
2. C
3. C
4. C
5. C
6. C
7. C
8. C
9. C
10. C
11. C
12. C
13. C
14. C
15. C
16. C
17. C
18. C
19. C
20. C
21. C
22. C
23. C
24. C
25. C
26. C
27. C
28. C
29. C
30. C
31. C
32. C
33. C
34. C
35. C
36. C
37. C
38. C
39. C
40. C
41. C
42. C
43. C
44. C
45. C
46. C
47. C
48. C
49. C
50. C
51. C
52. C
53. C
54. C
55. C
56. C
57. C
58. C
59. C

INTEGER I,IFRR,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)

THIS ROUTINE SOLVES THE MATRIX EQUATION

A = M * M MATRIX
X = M * N MATRIX
P = N * N MATRIX
C = N * N MATRIX.

THE PARAMETERS FOR THIS SUBROUTINE ARE

NDIM - DECLARED ROW DIMENSION OF THE MATRIX B
N - ROW DIMENSION OF B; COLUMN DIMENSION OF C
MDIM - DECLARED ROW DIMENSION OF A, C, AND P
M - ROW DIMENSION OF A, C, AND P
A - M * M MATRIX (DOUBLE PRECISION)
R - N * N MATRIX (DOUBLE PRECISION)
V - N * N MATRIX (DOUBLE PRECISION)
WHICH TRANSFORMS B(T) TO UPPER TRIANGULAR FORM
C - M * N MATRIX WHICH HOLDS THE TRANSFORMED B
ORT - DOUBLE PRECISION VECTOR USED AS A WORKING
VECTOR OF LENGTH AT LEAST MAX(N,M)
D - DOUBLE PRECISION VECTOR FOR THE SCALES
AT LEAST 2M² + 7M
IPR - INTEGER VECTOR FOR INTERNAL COUNTERS
EPS - ERROR TOLERANCE. EPS SHOULD BE
A SMALL NUMBER WHICH SATISFIES FL(1 + EPS) < 1
IERR - INTEGER VARIABLE USED TO SET AN ERROR
IERR=0 => NO ERROR RETURN.
IERR=J => J-TH EIGENVALUE COMPUTED
IERR=-J => A SINGULAR MATRIX
SOLVING FOR THE EIGENVALUES.

METHOD:

FIRST OF ALL, B(TRANSPOSE) IS FORMED
THROUGHOUT THE PROGRAM. (IN THE
USED TO REPRESENT B(TRANSPOSE)).
B(T) IS TRANSFORMED TO REAL UPPER
HESENBERG FORM USING ORTHOGONAL TRANSFORMATIONS.
THE RIGHT HAND SIDE C IS MULTIPLIED BY THE
MATRICES AND THE SOLUTION OF THIS
COMPUTED. THIS SOLUTION IS THEN
MULTIPLIED BY THE TRANSPOSED
FORMATION MATRICES IN ORDER TO GET
ORIGINAL PROBLEM.

NOTE:

IN ORDER TO GET OPTIMAL EFFICIENCY
BE $\leq M$. IF THIS IS NOT TRUE FOR
EQUATION AND SOLVE THE RESULTING EQUATION.

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

,N,NCIM

N).

)

ATION AX + XB = C WHERE

ARE:

HE MATRIX A

CIMENSION 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

ON INPUT AND X ON OUTPUT

C BY SUBROUTINE RG

INTERNAL USE OF LENGTH

M

USE OF LENGTH AT LEAST 4N

EQUAL THE SMALLEST NUMBER

> 1.

T ERROR CODES.

IF B HAS NOT BEEN DETERMINED

ATIONS.

X 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
TRANSFORMATIONS.

ED BY THESE TRANSFORMATION

TRANSFORMED SYSTEM IS

ULLTIPLIED BY THE TRANS-

TAIN THE SCLUTION TO THE

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

FORM B (TRANPOSE) FOR INTERNAL USE

$$\begin{aligned} & \text{C}_1 = \text{C}_2 = \text{C}_3 = \text{C}_4 \\ & \text{B}(1,1) = \text{B}(1,2) = \text{B}(2,1) = \text{B}(2,2) \end{aligned}$$

CONTINUE

A AND B(T) WILL BE TRANSFORMED INTO THE
MODIFICATIONS OF EISPACK RUTINES (NOOI

WARNING: DO NOT CHANGE THE ORDER OF THE

```

CALL FG(NCIM,N,B,1,V,DET,EFS,IERR)
CALL RG(MDTM,M,A,0,A,DET,EPS,IERR)

```

NOW, TRANSFORM THE RIGHT HAND SIDE.

CALL TRANSF(A,TRT,I,C,V,O,N,N,MDIN,NCIM)

NOW SOLVE THE SYSTEM OF EQUATIONS (ELDO).
 THE SYSTEM IS BLOCK UPPER HESSENBERG WITH
 MATRICES ON THE DIAGONAL ($A + B(I,I)$) *
 THE IDENTITY OFF THE DIAGONAL.
 TESTS ARE MADE TO SEE WHETHER A SINGLE
 MUST BE HANDLED AT THE CURRENT STAGE AND
 SUBROUTINES ARE CALLED.

SET RELATIVE ERROR TOLERANCE

FEFS = EPS*N*N*M*N

IND = N -

IF (IND.EQ.0) GO TO 40

```

10 IF (DAFS(B(IND + 1,IND)).LE.REFS) GO TO 11
      CALL N2SOLV(A,B,C,D,NDIM,N,MDIM,M,IND,I)
      IF (IEPF.NE.0) RETURN

```

GO TO 30

20 CALL NSOLVF(A,B,C,E,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,

THE SOLUTION OF THE TRANSFORMED SYSTEM

THE SOLUTION OF THE TRANSFORMED SYSTEM
HAS NOW BEEN OBTAINED. IT IS TRANSFORM
THE SOLUTION OF THE ORIGINAL SYSTEM OF

```
CALL TRANSF(A,ORT,0,C,V,I,N,N,MDIM,NCIM)
RETURN
ENC
```

SUBROUTINE RGNDIN(N,A,MATZ,Z,ORT,EES,I)

```
INTEGER N,NDIM,IEFF,MATZ
REAL*8 A(NDIM,N),Z(NDIM,N),ORT(N),EES
```

APPROPRIATE FORMS USING
FIEC BY S. NASH).

E NEXT TWO STATEMENTS

,D)

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

OR A DOUBLE BLOCK
D APPROPRIATE

3 20
PR,REPS,IERR)

PR,REPS,IERR)

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

OF EQUATIONS
ED BACK INTO
EQUATIONS.

,C)

ERR)

N

NSION OF TWO-DIMENSIONAL
N THE CALLING PROGRAM

NS.

W BY

US AND COLUMNS
UAROUNDING

NEAR ALGEEFA. 339-358(1971).
TIN AND WILKINSON.
THE ALGCL PROCEDURE ORTHES.

1

C FOR DETAILS
LTINE AXBC.

CRMATIC MATRIX
RAGE; IF MATZ=0, ORT HOLDS

BE THE SAME AS A).

MATRIX IF A IS TRANSFORMED TO

> COMPUTE THE SCHUR FORM OF A

-

TRANSFERN A TO UPPER-

-

INPUT) AND TRANSFORMED

-

MATRIX A

-

COLUMNAR TRANSFORMATIONS

THE EIGENVECTORS OF

EXCEPT THAT HOR2 HAS

WATZ IS ZERO CR NOT.

UPPER-HESSENBERG

120. THIS SUBROUTINE EECUCES A TO EITHER
121. OR SCHUR FORM DEFENDING CN WHEATHER
122. THE ROUTINES USEC ARE FROM EISPACK 1
123. BEEN MODIFIED SO AS NOT TC COMPUTER
124. AND ONLY REEUCES A IN SCHUR FCRN.
125. ARE USED THRCUGHCLT.
126. PARAMETERS:
127. NDIM - DECLARED FOR DIMENSION OF THE
128. ACTUAL DIMENSION OF THE MATRIX
129. A - MATRIX TO BE TRANSFORMED (CN
130. MATX - INTGER VARIABLE: NATZ = 0 =
131. HESSNBERG FORM: OTHERWISE CO
132. Z - STORES THE TRANSFORM NATZ
133. ORT - VECTJR USED FOR INTERNAL STOP
134. EPS - INFRMATION ABUT THE TRANSF
135. IERR - ERROR CODE: SEE SUBRUT
136. IERR - ERROR TOLERENCE, AS IN SUBR
137. 141. CALL CRTAN(NDIM,N,A,ORT,Z)
142. IF (MATZ.EQ.0) RETLN
143. CALL CRTHE(NDIM,A,A,ORT)
144. CALL CRTAN(NDIM,N,A,ORT,Z)
145. 146. RETURN
147. C
148. 149. 150. SUBROUTINE CRTHE(NDIM,N,A,ORT)
151. INTGER I,J,M,N,II,JI,LA,MP,NDIM,KP
152. 153. 154. 155. 156. 157. 158. 159. 160. 161. 162. 163. 164. 165. 166. 167. 168. 169. 170. 171. 172. 173. 174. 175. 176. 177. 178. 179.

```

180. C A CONTAINS THE HESSENBERG MATRIX
181. C THE ORTHOGONAL TRANSFORMATION
182. C IS STORED IN THE REMAINING PART OF
183. C HESSENBERG MATRIX;
184. C
185. C ORT CONTAINS FURTHER INFORMATION
186. C ONLY ELEMENTS 1 THROUGH N ARE
187. C
188. C QUESTIONS AND COMMENTS SHOULD BE
189. C APPLIED MATHEMATICS DIVISION, ARGONNE
190. C -----
191. C
192. C
193. C LA = N - 1
194. C KP1 = 2
195. C IF (LA .LT. KP1) GO TO 200
196. C
197. C DO 180 M = KP1, LA
198. C H = 0.000
199. C ORT(M) = 0.000
200. C SCALE = 0.000
201. C
202. C SCALE COLUMN (ALGOL TOL THEN NOT)
203. C
204. C 90
205. C SCALE = SCALF + DABS(A(I,M-1))
206. C
207. C IF (SCALE .EQ. 0.000) GO TO 180
208. C MP = M + N
209. C
210. C FOR I=N STEP -1 UNTIL M DO --
211. C
212. C
213. C 100
214. C II = M, N
215. C I = MP - II
216. C ORT(I) = A(I,M-1) / SCALE
217. C H = H + ORT(I) * ORT(I)
218. C CONTINUE
219. C
220. C G = -DSIGN(DSCRT(H),CRT(M))
221. C H = H - ORT(M) * G
222. C ORT(M) = ORT(N) - G
223. C
224. C 130 J = N, N
225. C F = 0.000
226. C
227. C FOR I=N STEP -1 UNTIL M DO
228. C
229. C 110 II = N, N
230. C I = MP - II
231. C F = F + CRT(I) * A(I,J)
232. C CONTINUE
233. C
234. C F = F / H
235. C
236. C 120 I = N, N
237. C A(I,J) = A(I,J) - F * ORT(I)
238. C
239. C CONTINUE

```

IX. INFORMATION ABOUT
NS USED IN THE REDUCTION
TRIANGLE UNDER THE

CN ABOUT THE TRANSFORMATIONS.
E USED.

DIRECTED TO R. S. GARROW,
CNAE NATIONAL LABORATORY

NEEDED)

C

--

ORTHESES.

T THE TRANS-

DGCNAL TRANS-

CRTFES

CALLING PROGRAM
OF TWO-DIMENSIONAL

ES.

A REAL GENERAL

AL SIMILARITY

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

300. C ON OUTPUT:
 301. C Z CONTAINS THE TRANSFORMATION
 302. C REDUCTION BY ORTHES;
 303. C ORT HAS BEEN ALTERED.
 304. C
 305. C
 306. C
 307. C
 308. C
 309. C
 310. C
 311. C
 312. C
 313. C
 314. C
 315. C
 316. C
 317. 60 DO 60 J = 1, N
 318. C Z(I,J) = 0.000
 319. C Z(I,I) = 1.000
 320. 80 CONTINUE
 321. C
 322. C KL = N - 2
 323. C IF (KL .LT. 1) GO TO 200
 324. C
 325. C FOR MP=N-1 STEP -1 UNTIL 2 DO --
 326. C
 327. C DO 140 MM = 1, KL
 328. C MP = N - MM
 329. C IF (A(MP,MP-1) .EQ. 0.000) GO
 330. C MP1 = MP + 1
 331. C
 332. 100 DO 100 I = MP1, N
 333. C DRT(I) = A(I,MP-1)
 334. C
 335. C DO 130 J = MP, N
 336. C G = 0.000
 337. C
 338. C DO 110 I = NF, N
 339. C G = G + DRT(I) * Z(I,J)
 340. C
 341. C DIVISOR BELOW IS NEGATIVE
 342. C DOUBLE DIVISION AVOIDS POSS
 343. C
 344. C G = (G / DRT(MP)) / A(MP,N)
 345. C
 346. C DO 120 I = NF, N
 347. C Z(I,J) = Z(I,J) + G * DRT(I)
 348. C
 349. 130 CONTINUE
 350. C
 351. 140 CONTINUE
 352. C
 353. 200 RETURN
 354. END
 355. C
 356. C
 357. C
 358. C
 359. C

MATRIX PRODUCED IN THE

DIRECTED TO B. S. GARBOW,
GENE NATIONAL LABORATORY

TC 140

OF H FORMED IN ORTHES.
SIELE UNDERFLOW.

(P-1)

I)

ECIFYING

ARITHMETIC.

RATIUNS.

HAS NOT BEEN

IX:

PRODUCED BY FORTRAN

E CALLING PROGRAM

OF TWO - DIMENSIONAL

SENDING MATRIX H.

NASH SO THAT IT ONLY

ALGEBRA. 372-396(1971).

AND WILKINSON.

ALGOL PROCEDURE HOR2.

REAL AND

PLEX NUMBERS

DRM, EPS

DIM, NNA.

SLEEFNUTTINE HRS2(NCM,N,H,Z,EPS,IERR)

DREAL(Z3) = Z3
DIMAG(Z3) = (0.000,-1.000) * Z3

THIS SUBROUTINE IS A TRANSLATION OF THE
NLM. NTH. 16. 1E1-204 (1970) BY PETERS

NOM MUST BE SET TO THE ROW DIMENSION:
ARRAY PARAMETERS AS DECLARED IN THE

N IS THE ORDER OF THE MATRIX:
H CONTAINS THE UPPER HESSENBERG MATRIX

Z CONTAINS THE TRANSFORMATION MATRIX
AFTER THE REDUCTION BY ORTHES.

ON OUTPUT:
H HAS BEEN DESTROYED.

Z CONTAINS THE TRANSFORMATION MATRIX
AFTER THE REDUCTION BY ORTHES.

IERR IS SET TO
ZER0 FOR NORMAL RETURN.
IF THE J-TH EIGENVALUE IS
DETERMINED AFTER 30 ITERATIONS.

THE RELATIVE PRECISION OF FLOATING POINT
EPS IS A MACHINE DEPENDENT PARAMETER SP

CCMPUTE MATRIX NGFM

K = 1
NORM = 0.000
IERR = 0

```

420.      DO 50 I = 1, N
421.      C
422.      DO 40 J = K, N
423.      C
424.      NORM = NORM + DABS(H(I,J))
425.      C
426.      K = I
427.      50 CONTINUE
428.      C
429.      EN = N
430.      T = 0.000
431.      C
432.      C
433.      60 IF (EN .LT. 1) GO TO 340
434.      ITS = 0
435.      NA = FN - 1
436.      ENM2 = NA - 1
437.      C
438.      C
439.      C
440.      C
441.      70 DO 80 LL = 1, EN
442.          L = EN + 1 - LL
443.          IF (L .EQ. 1) GO TO 100
444.          S = DABS(H(L-1,L-1)) + DABS(H(L,L))
445.          IF (S .EQ. 0.000) S = NORM
446.          IF (DABS(H(L,L-1)) .LE. EPS * S) GO TO 80
447.          80 CONTINUE
448.          C
449.          C
450.          C
451.      100 X = H(FN,EN)
452.          IF (L .EQ. FN) GO TO 270
453.          Y = H(NA,NA)
454.          W = H(FN,NA) * H(NA,EN)
455.          IF (L .EQ. NA) GO TO 280
456.          IF (ITS .EQ. 30) GO TO 1000
457.          IF (ITS .NE. 10 .AND. ITS .NE. 20) GO TO 120
458.          C
459.          C
460.          C
461.          T = T + X
462.          C
463.          DO 120 I = 1, FN
464.          120 H(I,I) = H(I,I) - X
465.          C
466.          S = DAES(H(EN,NA)) + DAEF(H(NA,ENM2))
467.          X = 0.7500 * S
468.          Y = X
469.          W = -0.437500 * S * S
470.          ITS = ITS + 1
471.          C
472.          C
473.          C
474.          C
475.          C
476.          DO 140 MM = L, ENM2
477.              N = ENM2 + L - MM
478.              ZZ = H(M,M)
479.              R = X - ZZ

```

ELEMENT

)

GO TO 100

0 TO 130

)

ND

(R) * L E * EPS * DABS(P)
BS((-(N+1,M+1)))) GO TO 150


```

540. C
541. C COLUMN MODIFICATION
542. C
543. C
544. C
545. C
546. C
547. C
548. C
549. C
550. C
551. C
552. C
553. C
554. C
555. C
556. C
557. C
558. C
559. C
560. C
561. C
562. C
563. C
564. C
565. C
566. C
567. C
568. C
569. C
570. C
571. C
572. C
573. C
574. C
575. C
576. C
577. C
578. C
579. C
580. C
581. C
582. C
583. C
584. C
585. C
586. C
587. C
588. C
589. C
590. C
591. C
592. C
593. C
594. C
595. C
596. C
597. C
598. C
599. C

      DO 230 I = 1, J
      P = X * H(I,K) + Y * H(I,K+1)
      IF (.NOT. NCLAS) GO TO 220
      P = P + ZZ * H(I,K+2)
      H(I,K+2) = H(I,K+2) - P * R
      H(I,K+1) = H(I,K+1) - P * Q
      H(I,K) = H(I,K) - P
      CONTINUE

      DO 250 I = 1, N
      P = X * Z(I,K) + Y * Z(I,K+1)
      IF (.NOT. NCLAS) GO TO 240
      P = P + ZZ * Z(I,K+2)
      Z(I,K+2) = Z(I,K+2) - P * R
      Z(I,K+1) = Z(I,K+1) - P * Q
      Z(I,K) = Z(I,K) - P
      CONTINUE

      DO 260 I = 1, N
      P = X * Z(I,K) + Y * Z(I,K+1)
      IF (.NOT. NCLAS) GO TO 250
      P = P + ZZ * Z(I,K+2)
      Z(I,K+2) = Z(I,K+2) - P * R
      Z(I,K+1) = Z(I,K+1) - P * Q
      Z(I,K) = Z(I,K) - P
      CONTINUE

      GO TO 70

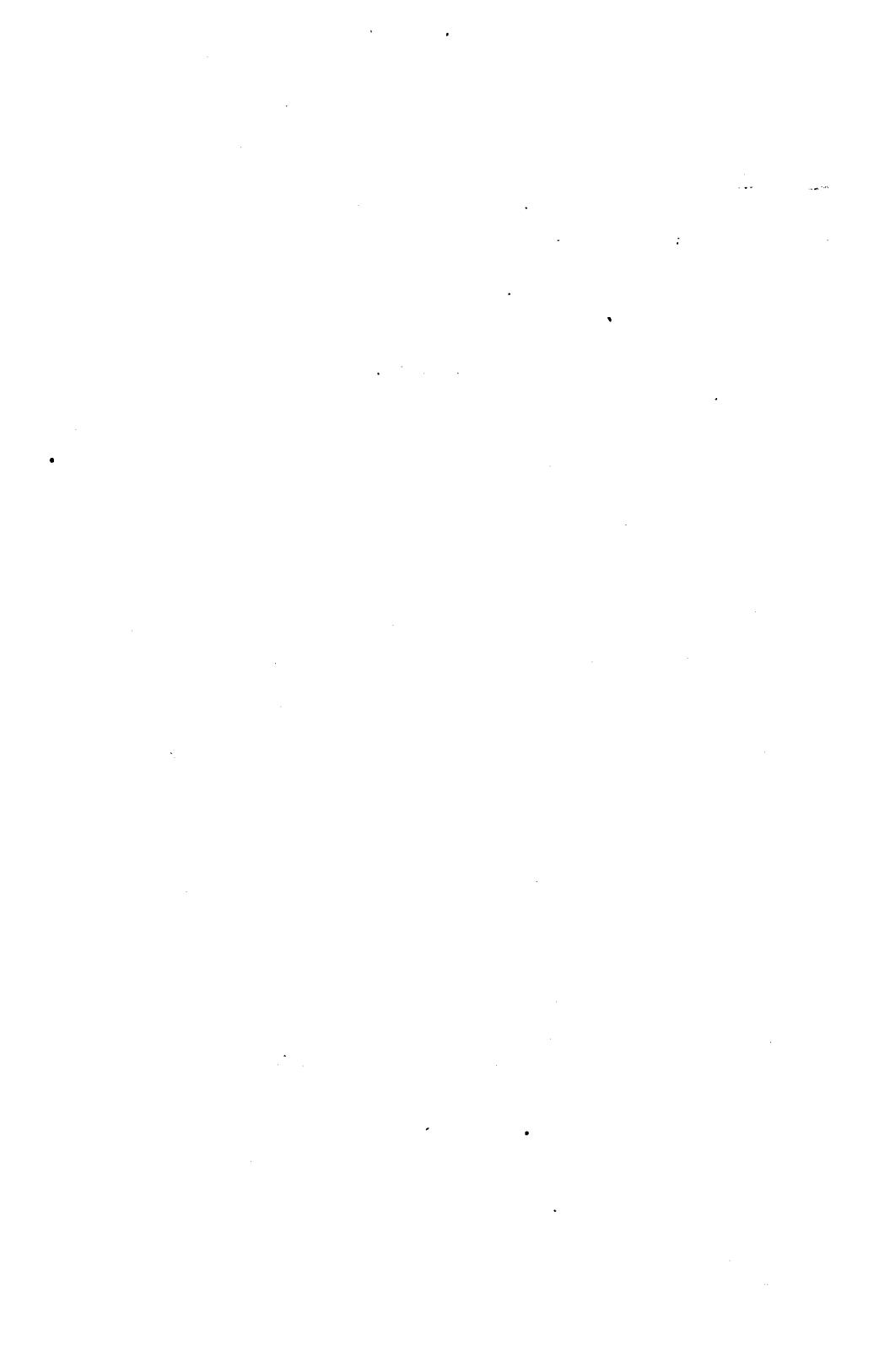
      C ONE ROOT FOUND
      C
      EN = X + T
      FN = NA
      GO TO 60

      C TWO ROOTS FOUND
      C
      P = (Y - X) / 2.000
      Q = P * P + W
      ZZ = DSQRT(DABS(Q))
      H(EN,FN) = X + T
      X = H(EN,EN)
      H(NA,NA) = Y + T
      IF (Q .LT. 0.0001) GO TO 320

      C REAL PAIR
      C
      ZZ = P + DSIGN(ZZ,F)
      X = H(FN,NA)
      S = DABS(X) + DABS(ZZ)
      P = X / S
      Q = ZZ / S
      R = DSQRT(P*P+Q*Q)
      P = P / R
      Q = Q / R

      DO 290 J = NA, N
      ZZ = H(NA,J)
      H(NA,J) = Q * ZZ + P * H(EN,J)
      H(FN,J) = Q * H(EN,J) - P * ZZ

```



JITLINE CRT(AN)

ACE OF ORT.

AS FOR THE SUBROUTINE

SEE SUPERROUTINE ORTHES)

HE LOWER PART OF A

(THE TRANSFORMATION

ED, IF ITX = 1, THEN

E) OR V(TRANSPOSE)

THE INTEGER VARIABLES

T(M),C(1),G

N2

,MDIM,NDIM,D)


```

660. C
661. IF (N2.LE.0) GO TO 45
662. DO 40 KK = 1,N2
663. K = N2 - KK + 1
664. IF (IT1.EQ.1) K = KK
665. K1 = K + 1
666. IF (A(K1,K).EQ.0.0D0) GO TO 40
667. D(K1) = ORT(K1)
668. K2 = K + 2
669. DO 10 I = K2,N
670. D(I) = A(I,K)
671. CONTINUE
672. DC 30 J = 1,N
673. G = 0.0D0
674. DO 20 I = K1,M
675. G = G + D(I) * C(I,J)
676. CONTINUE
677. G = G / ORT(K1) / A(K1,K)
678. DC 30 I = K1,M
679. C(I,J) = C(I,J) + G * D(I)
680. CONTINUE
681. 30
682. 40
683. C
684. C
685. 45
686. DO 60 I = 1,M
687. DO 50 J = 1,N
688. D(J) = 0.0D0
689. DO 50 K = 1,N
690. IF (IT2.EQ.0) D(J) = D(J) +
691. IF (IT2.EQ.1) D(J) = D(J) +
692. CONTINUE
693. DO 60 J = 1,N
694. C(I,J) = D(J)
695. 60
696. CONTINUE
697. RETURN
698. END
699. C
700. C
701. C
702. C
703. C
704. C
705. C
706. C
707. C
708. C
709. C
710. C
711. C
712. C
713. C
714. C
715. C
716. C
717. C
718. C
719. C
C
SUBROUTINE NSOLVE(A,B,C,D,NDIM,N,MDIM)
C
INTEGER I,II,IFRR,IND,IPR(1),IROW1,J,N
REAL*8 A(MDIM,M),B(NDIM,N),C(MDIM,N),D(MDIM,M)
C
THIS SUBROUTINE SETS UP AND SOLVES (WITH THE SUBROUTINE HESOLV) A SINGLE SYSTEM
C

$$(A + E(IND+1,INC+1) * I)(X(IND+1))$$

C

$$(X(I),C(I) - \text{COLUMN } I \text{ OF THE MATRICES}$$


$$\text{AND STORES THE RESULT IN THE AFFROFFIE MATRIX } C.$$


$$\text{THE UPPER-HESENBERG MATRIX }$$


$$\text{VECTOR } D \text{ IN ORDER TO SAVE STORAGE (ENTIRE }$$


$$\text{DIAGONAL ARE IGNORED).}$$


$$\text{THE RIGHT HAND SIDE IS ALSO STORED IN }$$


$$\text{TO REDUCE THE NUMBER OF VECTORS USED. THE }$$


$$\text{PARAMETERS ARE AS IN THE SUBROUTINE }$$


```

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

,M,INC,IPR,REPS,IERR)

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

WITH THE HELP OF
A SET OF EQUATIONS

= (C(IND+1))

X AND C)

ONE ROW OF THE
X IS STORED IN THE
TRIES BELOW THE SUB-

THE VECTOR D IN ORDER
IN THE SUBROUTINES.
THE AXBC.

ECESSARY

C,N,M,NDIN,NDIM)

WITHIN THE VECTOR D

CREATE THE RESULT IN C.

SENR3RG SYSTEM OF EQUATIONS
THE VECTOR D. THE METHOD USED
AL PIVOTTING. THE PARAMETERS
SIGHT HAND SIDE (ON INPUT),
CRDS INTERCHANGES

J1,K,N,N1,MFIN

```

720• -----  

721• C PERFORM BLOCK BACK-SUBSTITUTION IF N  

722• C IF (IND.LT.N - 1) CALL BACKSB(C,B,IND)  

723• C PERFORM BACK-BACK-SUBSTITUTION IF N  

724• C IF (IND.LT.N - 1) CALL BACKSB(C,B,IND)  

725• C SET UP THE SYSTEM OF EQUATIONS  

726• C MFIN = (M + (N + 1)) / 2 + M  

727• C DO 20 I = 1,M  

728• C FIND INDEX OF BEGINNING OF ROW I  

729• C M1 = IRW1(I,N)  

730• C DO 10 J = 1,N  

731• C IF (I.EQ.1) II = 1  

732• C IF (I.EQ.1) II = 1  

733• C D(M1+J-1)+1 = A(I,J)  

734• C CONTINUE  

735• C IF (I.EQ.1) II = 1  

736• C DO 10 J = 1,N  

737• C D(M1+J-1)+1 = A(I,J)  

738• C CONTINUE  

739• C 10 IF (I.EQ.1) II = 1  

740• C J = 2  

741• C IF (I.EQ.1) II = 1  

742• C D(M1+J) = D(M1+J) + B(IND+1,IND+1)  

743• C CONTINUE  

744• C STORE THE RIGHT HAND SIDE  

745• C DMFIN = C(I,IND+1)  

746• C 20 DMFIN = C(I,IND+1)  

747• C SOLVE THE SYSTEM OF EQUATIONS AND ST  

748• C CALL FSOLV(D,IPR,M,REFS,IERR)  

749• C IF (IERR.NE.0) GC TO 40  

750• C DO 30 I = 1,M  

751• C CALL FSOLV(D,IPR,M,REFS,IERR)  

752• C IF (IERR.NE.0) GC TO 40  

753• C DO 30 I = 1,M  

754• C C(I,IND+1) = C(I,IND+1)  

755• C 20 CONTINUE  

756• C IND = IND - 1  

757• C RETURN  

758• C IERR = -IND - 1  

759• C END  

760• C SUPEROLUTE HESOLV(D,IPR,M,REFS,IERR)  

761• C THIS SURJUETINE SOLVES AN UPPER-HE  

762• C WHICH IS STORED IN PACKED FORM IN T  

763• C IS GAUSSIAN ELIMINATION WITH PARTI  

764• C IN THE SUBROUTINE ARE:  

765• C C MATRIX OF COEFFICIENTS, I  

766• C AND SOLUTION (CN OUTPUT)  

767• C M = SIZE OF SYSTEM  

768• C IPR = INTEGER WHICH RECD  

769• C RPPS = - ERROR  

770• C C THIS SURJUETINE SOLVES AN UPPER-HE  

771• C WHICH IS STORED IN PACKED FORM IN T  

772• C IS GAUSSIAN ELIMINATION WITH PARTI  

773• C IN THE SUBROUTINE ARE:  

774• C C MATRIX OF COEFFICIENTS, I  

775• C AND SOLUTION (CN OUTPUT)  

776• C M = SIZE OF SYSTEM  

777• C IPR = - ERROR  

778• C RPPS = - ERROR

```

```

780.      C      IERR = ERROR CODE (IERR=0 => NC
781.      C      (IERR=-1 => S)
782.      C
783.      C
784.      C
785.      C
786.      C
787.      C      INITIIZE INTERCHANGE VECTORS AND
788.      C
789.      C      IERR = 0
790.      C      MFIN = (M * (M + 1)) / 2 + N
791.      C      DO 10 I = 1,M
792.      10     IPR(N+I) = IFCW1(I,M)
793.      C      IPR(I) = I + MFIN
794.      C      CONTINUE
795.      C      M1 = M - 1
796.      C      REDUCE THE MATRIX TO UPPER TRIANG
797.      C      IF (N.EQ.1) GO TO 35
798.      C      DO 30 I = 1,M1
799.      C      IF (DABS(D(IPR(M+I)+1)).GT.DAB
800.      C      INTERCHANGE ROWS IF NECESSARY
801.      C
802.      C      K = IPR(M+I)
803.      C      IPR(M+I) = IPR(M+I+1)
804.      C      IPR(M+I+1) = K
805.      C      K = IPR(I)
806.      C      IPR(I) = IPR(I+1)
807.      C      IPR(I+1) = K
808.      C
809.      C      CHECK FOR COMPUTATIONALLY SING
810.      C
811.      C      IF (DABS(D(IPR(M+I)+1)).LT.RE
812.      20     IPR(M+I+1) = IPR(M+I+1) + 1
813.      C
814.      C      ELIMINATE SUBDIAGONAL ELEMENT
815.      C
816.      C      MULT = D(IPR(M+I+1)) / D(IPR(I))
817.      C      D(IPR(I+1)) = D(IPR(I+1)) - M
818.      C      I1 = I + 1
819.      C      DO 30 J = I1,N
820.      C      D(IPR(I+I+1)+J-I) = D(IPR(J))
821.      C      MULT
822.      C
823.      30     CONTINUE
824.      35     IF (DABS(D(IPR(M+M)+1)).LT.REPS)
825.      C
826.      C      PERFORM BACK - SUBSTITUTION
827.      C
828.      C      D(IPR(M)) = D(IPR(M)) / D(IPR(M))
829.      C      IF (M1.EQ.0) RETURN
830.      C      DO 50 I1 = 1,M1
831.      C      I = N - I1
832.      C      I2 = I + 1
833.      C      MULT = 0.00
834.      C      DO 40 J1 = I2,N
835.      C      J = J1 - I2 + 2
836.      C      MULT = MULT + D(IPR(J1))
837.      40     CONTINUE
838.      C      C(IPR(I)) = (D(IPR(I)) - MULT
839.      50     CONTINUE

```

FINAL RETURN)
INGULAR MATRIX)

PARAMETERS.

ULAR FORM

S(C(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)
-*C(IPR(M+I)+J+1-I)

) GO TO 60

+N)+1)

* C(IPR(M+I)+J)
LT) / D(IPR(N+I)+1)

AND THEN USING GAUSSIAN

TO INSURE THAT THERE ARE

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

THE SET OF EQUATIONS

SYSTEM OF EQUATIONS.

C(1),BFS

J1,J2,K,LROWS,M,M1.

CIN,M,IND,IPR,REPS,IERR)

2

CR THE VECTOR D) ONE BEFORE
ATRIX.

I,I) * C(J,I)

EC.
RCW.

NCM

NCIN)

840• 60 RETURN IERR = -1
841• 61 RETURN
842• 62 END
843• 63 E44•
844• 64 E45•
845• 65 E46•
846• 66 E47•
847• 67 E48•
848• 68 E49•
849• 69 E50•
850• 70 E51•
851• 71 E52•
852• 72 E53•
853• 73 E54•
854• 74 E55•
855• 75 E56•
856• 76 E57•
857• 77 E58•
858• 78 E59•
859• 79 E60•
860• 80 E61•
861• 81 E62•
862• 82 E63•
863• 83 E64•
864• 84 E65•
865• 85 E66•
866• 86 E67•
867• 87 E68•
868• 88 E69•
869• 89 E70•
870• 90 E71•
871• 91 E72•
872• 92 E73•
873• 93 E74•
874• 94 E75•
875• 95 E76•
876• 96 E77•
877• 97 E78•
878• 98 E79•
879• 99 E80•
880• 100 E81•
881• 101 E82•
882• 102 E83•
883• 103 E84•
884• 104 E85•
885• 105 E86•
886• 106 E87•
887• 107 E88•
888• 108 E89•
889• 109 E90•
890• 110 E91•
891• 111 E92•
892• 112 E93•
893• 113 E94•
894• 114 E95•
895• 115 E96•
896• 116 E97•
897• 117 E98•
898• 118 E99•
899•

SUBROUTINE ACKSBC(B,C,D,IND,N,MDIN,M)
INTEGER INC,IND1,IND2,J,N,MDIN,N,
IERR,IERR=0
IF(IERR=0)GO TO 10
CALLBACK(B,C,D,IND,INC,IND1,IND2,J,N,MDIN,N)
PARMETERS ARE AS IN SUBROUTINE ACKSBC
IND1 = IND + 1
IND2 = IND + 2
DO 10 J = IND2,N
C(J,IND1) = C(J,IND1) - B(IND
CONTINUE
10 RETURN
ENC

THIS FUNCTION COUNTS THE INDEX (I
THE BEGINNING OF RCV I IN THE MM M
IF ((I-1)*N - (I-2)*(I-3)) /
IROW1 = (I-1)*N - (I-2)*(I-3) /
RETURN
ENC

INTEGER FUNCION IRWMI(I,M)
THIS FUNCTION COUNTS THE INDEX (I
THE BEGINNING OF RCV I IN THE MM M
IF ((I-1)*N - (I-2)*(I-3)) /
IROW1 = (I-1)*N - (I-2)*(I-3) /
RETURN
ENC

IRWMI(I,M)

```

900. C ELIMINATION. IN ORDER TO TAKE ADVANTAGE OF THE
901. C ZEROES IN THIS MATRIX, THE MATRIX IS STORED IN PACKED FORM. (ENTRIES BELOW THE
902. C DIAGONAL ARE IGNORED) THE RIGHT HAND SIDE AND SOURCE PARAMETERS ARE AS IN THE SUBROUTINE
903. C
904. C -----
905. C
906. C
907. C
908. C PERFORM BLOCK BACK-SUBSTITUTION IF N > M
909. C
910. C IF (IND.LT.N-1) CALL BACKS2(C,B,IND)
911. C
912. C SET UP THE SYSTEM OF EQUATIONS FOR ROW I
913. C
914. C M2 = 2 * M
915. C MFIN = (M2 + (M2 + 1)) / 2 + 4 * N
916. C DO 20 I = 1,M
917. C
918. C CCC FIND BEGINNING AND LENGTH OF ROW I
919. C
920. C M1 = IRROW2(2*I-1,M)
921. C K = LRROW2(2*I-1,M)
922. C I1 = I - 1
923. C IF (I.EQ.1) I1 = 1
924. C DO 10 J = I1,M
925. C J1 = 2 * (J - I1 + 1)
926. C J2 = 1
927. C IF (M1.EQ.0) J2 = 0
928. C D(M1+J1-1) = A(I,J)
929. C D(M1+J1) = 0.D0
930. C D(M1+K+J1-J2) = A(I,J)
931. C D(M1+K+J1-1-J2) = 0.D0
932. C CONTINUE
933. C 10 J1 = 3
934. C IF (I.EQ.1) J1 = 1
935. C D(J1+M1) = D(J1+M1) + B(IND,IND)
936. C D(J1+M1+1) = D(J1+M1+1) + B(IND,IND+1)
937. C IF (I.NE.1) J1 = 2
938. C D(J1+M1+K) = D(J1+M1+K) + B(IND+1,IND)
939. C D(J1+M1+K+1) = D(J1+M1+K+1) + B(IND+1,IND+1)
940. C
941. C CCC STORE RIGHT HAND SIDE
942. C
943. C D(2*I+MFIN) = C(I,IND+1)
944. C D(2*I-1+MFIN) = C(I,IND)
945. C CONTINUE
946. C 20
947. C CCC SOLVE THE SYSTEM OF EQUATIONS AND STORE RESULTS
948. C
949. C CALL H2SCLV(D,IPR,N,REFS,IERR)
950. C IF (IFRR.NE.0) GO TO 40
951. C DO 30 I = 1,M
952. C C(I,IND) = D(IFR(2*I-1))
953. C C(I,IND+1) = C(IPR(2*I))
954. C
955. C 30 CONTINUE
956. C IND = IND - 2
957. C RETURN
958. C 40 IERR = -IND - 1
959. C RETURN
END

```

STAGE OF THE NUMBER OF
S STORED IN THE VECTOR
E SUB-SUB-DIAGONAL ARE
LUTION ARE ALSO STORED IN D.
LINE AXXBC.

NECESSARY

,N,M,MDIM,NDIM)

-2SOLV

2I-1 IN THE VECTOR D.

IND+1)

1,IND)

IND+1,IND+1)

TCRE THE RESULT BACK IN C

MATRIX

MAX) GC TO 20

METERS.

LAR MATRIX)
RETURN)

RMUTATIONS

IT HARD SIDE (ON INPUT).

N.
* 2M EQUATIONS
SUS - SL3 - DIAGONAL
CTOR IN DACKED FORM.

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


```

1020. IPR(I) = IPR(I+L)
1021. IPR(I+L) = K
1022. CONTINUE
1023. C
1024. C
1025. C
1026. C
1027. C
1028. C
1029. C
1030. C
1031. C
1032. C
1033. C
1034. C
1035. C
1036. C
1037. C
1038. C
1039. 40 CONTINUE
1040. IF (DARS(D(IPR(M2+N2)+1)).LE.REP
1041. C
1042. C
1043. C
1044. C
1045. C
1046. C
1047. C
1048. C
1049. C
1050. C
1051. C
1052. 50 C
1053. C
1054. 60 C
1055. 70 C
1056. 80 C
1057. C
1058. C
1059. C
1060. C
1061. C
1062. C
1063. C
1064. C
1065. C
1066. C
1067. C
1068. C
1069. C
1070. C
1071. C
1072. C
1073. C
1074. C
1075. C
1076. C
1077. C
1078. 10 C
1079. C

      IPR(M2+I+1) = IPR(M2+I+1) + 1
      IF (I.NE.M21) IPR(M2+I+2) = I
      IPI = I + 1

      ADJUST POINTERS TO BEGINNINGS

      ELIMINATE SUBDIAGONAL ELEMENT

      DO 40 J = 1,I1
      MAX = D(IPR(M2+I+J)) / D(IPR(I+J))
      D(IPR(I+J)) = D(IPR(I+J))
      DO 40 K1 = IF1,M2
      K = K1 - I
      D(IPR(M2+I+J)+K) = D(IPR(M2+I+J))

      * CONTINUE
      IF (DARS(D(IPR(M2+N2)+1)).LE.REP
      PERFORM BACK SUBSTITUTION

      D(IPR(M2)) = D(IPR(M2)) / C(IPR(M2))
      DO 60 I1 = 1,N21
      I = M2 - I1
      I2 = I + 1
      MAX = 0.D0
      DO 50 J1 = I2,N2
      J = J1 - I2 + 2
      MAX = MAX + D(IPR(J1)) * D(IPR(J1))
      CONTINUE
      D(IPR(I)) = (C(IPR(I)) - MAX)
      CONTINUE
      RETURN
      IEPR = -1
      GO TO 70
      END

      SUBROUTINE BACKS2(C,B,IND,N,M,MDIM)
      INTEGER I,IND,IND1,IND2,J,M,MDIM
      REAL*8 B(MDIM,N),C(MDIM,N)

      BLOCK BACK - SUBSTITUTION FOR TWO
      PARAMETERS ARE AS IN SUBROUTINE

      INC1 = IND + 1
      IND2 = IND + 2
      DO 10 I = IND2,N
      DO 10 J = 1,M
      C(J,IND1) = C(J,IND1) - E(C(J,IND))
      C(J,IND) = C(J,IND) - B(INC1)
      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 T7 80

M2+N2)+1)

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

IM,NDIM)

,N,NCIM

J 'FCWS'•
AXXBC•

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

THE VECTOR(D) ONE
IS THE 2N*2N MATRIX.

ROM I IN THE
IN PACKAGE FCRM IN THE

• 11117
• 11116
• 11115
• 11114
• 11113
• 11112
• 11111
• 11110
• 11109
• 11108
• 11107
• 11106
• 11105
• 11104
• 11103
• 11102
• 11101
• 11000
• 10999
• 10998
• 10997
• 10996
• 10995
• 10994
• 10993
• 10992
• 10991
• 10990
• 10989
• 10988
• 10987
• 10986
• 10985
• 10984
• 10983
• 10982
• 10981
• 10980

三

