MASTER

Argonne National Laboratory
FORTAN SUBROUTINES FOR
HOUSEHOLDER'S METHOD IN
THE COMPLEX CASE AND EIGENSYSTEMS
OF HERMITIAN MATRICES

by

Dennis J. Mueller

RELEASED FOR ANNOUNCEMENT

IN NUCLEAR SCIENCE ABSTRACTS

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

ANL-7231
Mathematics and Computers
(TID-4500)
AEC Research and
Development Report

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

CFSTI PRICES

H.C. \$ 2.00; MN. 65

FORTRAN SUBROUTINES, FOR
HOUSEHOLDER'S METHOD IN
THE COMPLEX CASE AND EIGENSYSTEMS
OF HERMITIAN MATRICES

by

Dennis J. Mueller[†]

Applied Mathematics Division

RELEASED FOR ANNOUNCEMENT
IN NUCLEAR SCIENCE ABSTRACTS

July 1966

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission," includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

[†]Department of Mathematics
Yale University
New Haven, Connecticut

Operated by The University of Chicago
under
Contract W-31-109-eng-38
with the
U. S. Atomic Energy Commission

THIS PAGE
WAS INTENTIONALLY
LEFT BLANK

TABLE OF CONTENTS

<u>No.</u>	<u>Title</u>	<u>Page</u>
ABSTRACT.		5
I. INTRODUCTION.		5
II. HOUSEHOLDER'S METHOD FOR COMPLEX MATRICES		6
III. THE DIAGONAL UNITARY SIMILARITY		9
IV. COMPUTATIONAL NOTES AND FORTRAN SUBROUTINES		10
A. Outlines and Notes.		10
B. Details of the Similarity $P_r A^{(r-1)} P_r$		10
C. Subroutine <i>CODIAG</i> : Householder Reduction of Hermitian Matrices.		14
D. Subroutine <i>HESSEN</i> : Householder Reduction of Nonhermitian Matrices.		16
E. Subroutine <i>CHANGE</i> : Back Transformation of Eigenvectors . . .		18
V. Test Results.		19
APPENDIX. Subroutine <i>EIGEN</i> : Eigensystems of Real Symmetric Matrices		25
ACKNOWLEDGMENTS		31
REFERENCES.		31

THIS PAGE
WAS THIS PAGE ALLY
WAS INTENTIONALLY
LEFT BLANK

FORTRAN SUBROUTINES FOR
HOUSEHOLDER'S METHOD IN
THE COMPLEX CASE AND EIGENSYSTEMS
OF HERMITIAN MATRICES

by

Dennis J. Mueller

ABSTRACT

This report presents FORTRAN subroutines that use Householder's method to reduce a complex matrix to upper Hessenberg form in which the lower subdiagonal is real. A separate subroutine is given for hermitian matrices. Here the reduced matrix is real, symmetric, and tridiagonal. Its eigensystem can be calculated by a standard routine given in the appendix. A subroutine is included, which transforms eigenvectors of the reduced matrix to those of the original matrix.

I. INTRODUCTION

This report presents FORTRAN subroutines that reduce a complex matrix to upper Hessenberg form (null below the first subdiagonal) as the first step in determining the eigensystem of the matrix. These subroutines are direct counterparts to the ALGOL procedures given in Reference 1.

The reduction to Hessenberg form is accomplished by Householder's method extended to the complex case. This is followed by a diagonal unitary similarity, which causes the lower subdiagonal elements of the Hessenberg matrix to become real.

If the complex matrix is hermitian, the resulting matrix will be real, symmetric, and tridiagonal. The eigensystem of such a matrix can be calculated by standard routines (cf. References 2 and 3 for ALGOL procedures and the appendix of this report for a FORTRAN subroutine).

In the nonhermitian case, the matrix is in proper form for the application of Francis' *QR* algorithm to find the eigenvalues. This algorithm was first presented as an ALGOL-like procedure in Reference 4. A CDC-3600 FORTRAN version, which includes a Householder reduction and which uses complex declarations, has been written by Mr. Burton S. Garbow of Argonne National Laboratory.[†]

We include a subroutine which transforms the eigenvectors of the Hessenberg (or tridiagonal) matrix to those of the starting matrix.

[†] Available as Library routine ANL-F251 from Control Data Corporation, CO-OP Distribution Center, 3145 Porter Drive, Palo Alto, California 94304.

Current practice in many laboratories is to construct from the given n -by- n complex matrix $A = A_1 + iA_2$ the real $2n$ -by- $2n$ matrix

$$\tilde{A} = \begin{bmatrix} A_1 & A_2 \\ -A_2 & A_1 \end{bmatrix}$$

and use real arithmetic codes on it. The main reason for performing complex arithmetic on A rather than real arithmetic on $\tilde{A}$ is that the former requires less computation and less storage.

It is well known that if A is hermitian, $\tilde{A}$ gives a matrix with the same eigenvalues but with the multiplicity of each doubled. Since the work is generally proportional to the cube of the order, introducing $\tilde{A}$ should make the computation eight times as long but, insofar as real arithmetic can replace complex arithmetic, this is reduced by a factor of four, leaving a penalty factor of two for failure to use the appropriate complex arithmetic. Where, as in some recent programming systems, a type declaration of complex mode is in fact implemented with closed subroutines by the compiler, the advantage in speed of a complex arithmetic code may be lost. But like the ALGOL procedures in Reference 1, the FORTRAN subroutines here are programmed to do complex arithmetic explicitly in terms of the real and imaginary parts. Complex declarations are not used.

If A is nonhermitian, the eigenvalues of $\tilde{A}$ occur in pairs $\lambda_k \pm i\mu_k$, and one of each such pair is an eigenvalue of A . One can decide which value belongs to A by noting the structure of one of the corresponding eigenvectors of $\tilde{A}$. In several engineering applications, however, only the eigenvalues of a complex matrix A are required. Here an additional advantage of using complex arithmetic on A is that the eigenvalues of A are determined without using eigenvectors.

A low level of FORTRAN is used in the subroutines presented here so that the routines will be usable with a reasonably large number of compilers.

II. HOUSEHOLDER'S METHOD FOR COMPLEX MATRICES

Householder's method for the general complex matrix is derived here, since the equations are needed for their FORTRAN implementation.[†] A few of the remarks concern simplifications that occur if the starting matrix is hermitian.

Starting with the complex matrix $A^{(1)}$ of order n , we perform $n - 2$ similarity transformations of the form

$$A^{(r)} = P_r A^{(r-1)} P_r, \quad r = 2, 3, \dots, n - 1. \quad (2.1)$$

Each unitary hermitian matrix $P_r = P_r^* = P_r^{-1}$ has the structure

[†]This form of the derivation was suggested by J. H. Wilkinson, who was kind enough to read an earlier, somewhat longer, version.

$$P_r = \begin{bmatrix} I_{r-1} & 0 \\ 0 & H_r \end{bmatrix} = \begin{bmatrix} I_{r-1} & 0 \\ 0 & I - 2ww^* \end{bmatrix},$$

in which I_{r-1} is the identity matrix of order $r - 1$, w is a column vector with $n - r + 1$ components, and w^* denotes the conjugate transpose of w . The demand that H_r be unitary requires $w^*w = 1$.

We obtain the explicit form for H_r by examining in detail the transformation of $A^{(r-1)}$ into $A^{(r)}$. To do this we write $A^{(r-1)}$ in block form as

$$A^{(r-1)} = \begin{bmatrix} L & M \\ 0 & d \cdot \bar{N} \end{bmatrix},$$

where L is a square matrix of order $r - 1$, which is already in upper Hessenberg form, and d is a column vector with $n - r + 1$ components. If $A^{(1)}$ is hermitian, all of the rows of matrix M are zero, except the last, which is then d^* .

The result of the transformation is

$$A^{(r)} = P_r A^{(r-1)} P_r = \begin{bmatrix} L & MH_r \\ 0 & H_r d \cdot \bar{H}_r N H_r \end{bmatrix}.$$

The effect of H_r on d is to reflect d in the hyperplane orthogonal to w ($H_r w = -w$, $H_r v = v$ if $w^*v = 0$). To complete the inductive argument, we need to obtain

$$H_r d = \kappa e_1, \quad (2.2)$$

where κ is some complex scalar and $e_1^* = (1, 0, \dots, 0)$.

Such scalars do exist but are not necessarily real. This is shown in a concise, very readable discussion in Wilkinson's book,⁵ §§45-46, on pp. 48-50. (We enthusiastically commend this reference to the reader.)

With such a scalar determined, H_r reflects d in the hyperplane orthogonal to $d - \kappa e_1$. That is, take w parallel to $d - \kappa e_1$. The details of the derivation are quoted from Reference 1, beginning with the determination of κ :

"Since H_r is unitary, the Euclidean norm of d is equal to that of $H_r d$:

$$|\kappa|^2 = d^* d = d_1^* d_1 + d_2^* d_2 + \dots + d_{n-r+1}^* d_{n-r+1} = V_r, \quad (2.3)$$

which defines V_r . We skip the transformation with P_r in case $V_r \leq \gamma \times \text{tolerance}^2$, where γ is the square of the relative machine precision and *tolerance* is the maximum absolute value of the real and imaginary parts of elements of $A^{(1)}$. For the case $V_r > \gamma \times \text{tolerance}^2$ we premultiply both sides of (2.2) by d^* to obtain

$$d^* H_r d = d^* \kappa e_1 = \kappa d_1^*. \quad (2.4)$$

Since H_r is hermitian, the left member of (2.4) is real. Therefore κ must have the phase of d_1 (or π plus the phase of d_1). Then (2.3) and (2.4) yield

$$\kappa = \begin{cases} \pm V_r d_1 / (V_r d_1^* d_1)^{1/2} & \text{if } d_1 \neq 0, \\ \pm V_r^{1/2} & \text{if } d_1 = 0. \end{cases} \quad (2.5)$$

Here the form $(V_r d_1^* d_1)$ is used in order to reduce to one the number of square roots required at each stage.

"From $(I - 2ww^*) d = \kappa e_1$ we obtain

$$2ww^* d = d - \kappa e_1 = u,$$

where

$$u_1 = \begin{cases} d_1 (1 + V_r / (V_r d_1^* d_1)^{1/2}) & \text{if } d_1 \neq 0, \\ V_r^{1/2} & \text{if } d_1 = 0, \end{cases} \quad (2.6)$$

$$u_j = d_j, \quad j = 2, 3, \dots, n - r + 1.$$

For (2.6) we have selected the minus signs in (2.5) since in the case $d_1 \neq 0$ this causes both terms to be positive in the divisor that appears in equation (2.7) below.

"Since $w^* d$ is a scalar, the vector u is proportional to w . For $d_1 \neq 0$ we have

$$\begin{aligned} u^* u &= d_1^* d_1 (1 + V_r / (V_r d_1^* d_1)^{1/2})^2 + d_2^* d_2 + d_3^* d_3 + \dots \\ &= 2V_r + 2(V_r d_1^* d_1)^{1/2}. \end{aligned}$$

Then from $ww^*/w^*w = ww^* = uu^*/u^*u$ we obtain H_r as

$$I - 2ww^* = I - 2uu^*/(2V_r + 2(V_r d_1^* d_1)^{1/2}) = I - uu^*/\delta, \quad (2.7)$$

which defines δ for this case. For $d_1 = 0$ we have

$$u^* u = V_r + d_2^* d_2 + d_3^* d_3 + \dots = 2V_r$$

so that H_r is

$$I - 2ww^* = I - 2uu^*/2V_r = I - uu^*/\delta, \quad (2.8)$$

which defines δ for this case."

The results of the derivation are also given in Reference 5, §73, on pp. 342-343.

III. THE DIAGONAL UNITARY SIMILARITY

By the end of the Householder reduction, we have an upper Hessenberg matrix $A^{(n-1)}$. Write its lower subdiagonal elements as

$$a_{r+1,r}^{(n-1)} = b_r \exp(i\phi_r)$$

where $b_r \geq 0$ and $0 \leq \phi_r < 2\pi$. Define the complex diagonal matrix $T = \text{diag}(\tau_1, \dots, \tau_n)$ by

$$\tau_1 = 1,$$

$$\tau_{r+1} = \tau_r \exp(-i\phi_r), \quad r = 1, \dots, n-1.$$

Then T is unitary, and the similarity $TA^{(n-1)}T^*$ replaces each subdiagonal element $a_{r+1,r}^{(n-1)}$ by b_r .

If $A^{(1)}$ is hermitian, the resulting matrix will be real, symmetric and tridiagonal:

$$TA^{(n-1)}T^* = \begin{bmatrix} c_1 & b_1 & & \\ & b_1 & c_2 & \text{etc.} \end{bmatrix} \quad (\text{tridiagonal}).$$

The eigensystem of such a matrix can be calculated by standard methods. For example, one can use Givens' method of bisections to calculate the eigenvalues and Wielandt inverse iteration to find the eigenvectors. These two methods are used by subroutine *EIGEN*, described in the appendix. Once *EIGEN* has completed its task, each eigenvector y of $TA^{(n-1)}T^*$ can be transformed to an eigenvector

$$P_2 P_3 \cdots P_{n-2} P_{n-1} T^* y \quad (3.2)$$

of $A^{(1)}$ to determine completely the eigensystem of $A^{(1)}$.

For the case in which $A^{(1)}$ is nonhermitian, the eigenvalues of $TA^{(n-1)}T^*$ can be calculated by the complex version of Francis' *QR* algorithm, which has been programmed by B. S. Garbow as ANL-F251, cited in Section I.

Apparently no routine generally available in either ALGOL or FORTRAN is designed to calculate eigenvectors of a complex Hessenberg matrix by inverse iteration. If the eigenvectors as well as eigenvalues of a non-hermitian matrix are required, an alternative is to use the algorithm by Eberlein.⁶ Although no codes are included, the amount of programming necessary should not be prohibitive. This method is a generalization of the Jacobi method and does not use an initial reduction of the matrix to Hessenberg form.

IV. COMPUTATIONAL NOTES AND FORTRAN SUBROUTINES

A. Outlines and Notes

The eigensystem of a hermitian matrix can be calculated by using subroutines *CODIAG*, *EIGEN*, and *CHANGE*. Subroutine *CODIAG* reduces the hermitian matrix $A^{(1)}$ to the real symmetric tridiagonal matrix $TA^{(n-1)T*}$. Subroutine *EIGEN*, given in the appendix, can calculate the eigensystem of $TA^{(n-1)T*}$. Subroutine *CHANGE* transforms the eigenvectors of $TA^{(n-1)T*}$ to those of $A^{(1)}$.

In the nonhermitian case, subroutine *HESSEN* reduces the complex matrix $A^{(1)}$ to the upper Hessenberg matrix $TA^{(n-1)T*}$ whose lower subdiagonal is real. Subroutine *CHANGE* can be used to transform eigenvectors of $TA^{(n-1)T*}$ to those of $A^{(1)}$.

The FORTRAN used in these subroutines lies (hopefully) in the intersection of a number of versions of that language currently in use. We note several minor changes which would make the subroutines usable with at least one additional FORTRAN compiler. One compiler requires that the terminal *F* in the function names *SQRTF* and *ABSF* be dropped. Also, in a dialect of FORTRAN that does not allow dynamic dimensioning of arrays, the variable *m*, which appears only in the calling sequences and *DIMENSION* statements of the subroutines in this section, may be deleted from the calling sequences and replaced by an actual integer in the *DIMENSION* statements.

Several versions of FORTRAN that do permit dynamic dimensioning of arrays in subroutines, unfortunately require that each such array have its name and the variables corresponding to the dimensions appear in the calling sequence. Thus the calling sequences for two of our FORTRAN subroutines are longer than those for the corresponding ALGOL procedures. But if dynamic dimensioning is not used and *m* is replaced by an actual integer, the two calling sequences may be shortened; the array names *f* and *g* may be deleted from the calling sequence of subroutine *HESSEN*, and the array name *alpha* may be deleted from the calling sequence of subroutine *CHANGE*.

FORTRAN variable names that end in the digit 1 (e.g., *h1*) or whose first subscript is 1 (e.g., $A(1,i,j)$) represent the real parts of the respective complex quantities; the digit 2 plays the corresponding role for the imaginary parts.

B. Details of the Similarity $P_r A^{(r-1)} P_r$

We first give the details of the transformation $A^{(r-1)} \rightarrow A^{(r)}$ for the case in which $A^{(1)}$ is nonhermitian and $V_r > \gamma \times \text{tolerance}^2$. From Section II, we have

$$A^{(r-1)} = \begin{bmatrix} L & M \\ 0 & d \cdot N \end{bmatrix}$$

and

$$A^{(r)} = P_r A^{(r-1)} P_r = \begin{bmatrix} L & & & MH_r \\ 0 & H_r d & & H_r NH_r \\ & & & \end{bmatrix},$$

where $P_r = I_{r-1} \oplus H_r$, $\text{order}(H_r) = n - r + 1$.

The only nonzero element of $H_r d$ is the first component. From (2.2) and (2.5) we have

$$a_{r,r-1}^{(r)} = \kappa = \begin{cases} -V_r^{1/2}, & \text{if } a_{r,r-1}^{(r-1)} = 0, \\ -V_r a_{r,r-1}^{(r-1)} / \left(V_r a_{r,r-1}^{(r-1)*} a_{r,r-1}^{(r-1)} \right)^{1/2}, & \text{otherwise.} \end{cases}$$

For the remaining portions of $A^{(r)}$ we write $v = u/\delta$ so that[†]

$$MH_r = M(I - uu^*/\delta) = M - (Mu) v^*$$

and

$$\begin{aligned} H_r NH_r &= (I - uu^*/\delta) N(I - uu^*/\delta) \\ &= N - v(u^*N) - (Nu) v^* + (u^*Nv) uv^*. \end{aligned}$$

Writing ρ for the scalar u^*Nv , we have

$$H_r NH_r = N - v(u^*N) - (Nu - \rho u) v^*.$$

The vector v is temporarily stored in an unused portion of the vector τ (eventually τ houses the diagonal of the unitary matrix T). We also define the column vector

$$g = \begin{bmatrix} Mu \\ Nu \end{bmatrix}$$

and the row vector $f = (0; u^*N)$, where the 0 means that the first $r-1$ components of f are zero. The elements in columns r through n of $A^{(r)}$ are then obtained from

$$a_{i,j}^{(r)} = \begin{cases} a_{i,j}^{(r-1)} - g_i \tau_j^*, & j \geq r, \quad i < r, \\ a_{i,j}^{(r-1)} - \tau_i f_j - \left(g_i - \rho a_{i,r-1}^{(r-1)} \right) \tau_j^*, & j \geq r, \quad i \geq r. \end{cases}$$

We now list each FORTRAN variable name used by subroutine *HESSEN* in the formation of $a_{i,j}^{(r)}$. The FORTRAN name is given on the left-hand side and the corresponding mathematical expression on the right-hand side of each equation below. When the mathematical expression represents a complex quantity, an ordered pair of real variable names is given for the FORTRAN equivalent:

[†]We let $v_j = u_{j-r+1}/\delta$ since this simplifies indexing later on.

$$toler = tolerance,$$

$$(x1, x2) = \tau_i f_j,$$

$$(q1, q2) = g_i - \rho a_{i,r-1}^{(r-1)},$$

$$(t1, t2) = \left(g_i - \rho a_{i,r-1}^{(r-1)} \right) \tau_j^*,$$

$$delta = \delta,$$

$$(A(1,i,j), A(2,i,j)) = a_{i,j}^{(r-1)}, \text{ later replaced by } a_{i,j}^{(r)}.$$

When $A^{(1)}$ is hermitian, we have

$$A^{(r-1)} = \begin{bmatrix} L & \cdot & 0 \\ \cdot & \cdot & \cdot \\ 0 & d & N \end{bmatrix}$$

and

$$A^{(r)} = P_r A^{(r-1)} P_r = \begin{bmatrix} L & \cdot & 0 \\ \cdot & \cdot & d^* H_r \\ 0 & H_r d & H_r N H_r \end{bmatrix}.$$

Thus, only the lower triangle (including the diagonal) of each $A^{(r)}$ must be calculated. Again, the first element of $H_r d$ is calculated separately.

We define the column vector $g = Nu$, and again write $v = u/\delta$. In this case the scalar $\rho = u^* N v = g^* v$ is a real number since N is hermitian and, as before, δ is real. To economize on the amount of memory space required by the procedure, the vectors v and g are temporarily stored in unused portions of other arrays. Vector v is stored in the latter portion of the complex vector τ . The real parts of the components of g are stored in the real vector c , while the imaginary parts are stored in the real vector b . The vectors c and b later contain the diagonal and subdiagonal, respectively, of the real symmetric tridiagonal matrix $TA^{(n-1)}T^*$. But they are not needed for this purpose until the calculation of $A^{(n-1)}$ has been completed.

For this case we have

$$\begin{aligned} H_r N H_r &= N - v(u^* N) - (Nu - \rho u) v^* \\ &= N - v g^* - (g - \rho u) v^*. \end{aligned}$$

Then, writing (c_i, b_i) for g_{i-r+1} , we have

$$a_{i,j}^{(r)} = a_{i,j}^{(r-1)} - \tau_i (c_j - b_j) - \left\{ (c_i, b_i) - \rho a_{i,r-1}^{(r-1)} \right\} \tau_j^*, \quad i \geq j \geq r.$$

$a_{i,j}^{(n)}$ The subroutine *CODIAG* uses FORTRAN variable names in the formation of that correspond closely to their counterparts in subroutine *HESSEN*.

If the entire starting matrix, including the upper portion, is supplied to subroutine *CODIAG* when it is called, the output matrix *A* will contain the following quantities:

$$A = \begin{bmatrix} a_{1,1}^{(n-1)} & a_{1,2}^{(1)} & a_{1,3}^{(1)} & \cdot & \cdot & \cdot & a_{1,n}^{(1)} \\ u_1^{(2)} & a_{2,2}^{(n-1)} & a_{2,3}^{(1)} & \cdot & \cdot & \cdot & a_{2,n}^{(1)} \\ u_2^{(2)} & u_1^{(3)} & a_{3,3}^{(n-1)} & \cdot & \cdot & \cdot & a_{3,n}^{(1)} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & u_1^{(n-1)} & a_{n-1,n-1}^{(n-1)} & a_{n-1,n}^{(1)} \\ u_{n-1}^{(2)} & u_{n-2}^{(3)} & \cdot & \cdot & \cdot & u_2^{(n-1)} & a_{n,n-1}^{(n-1)} & a_{n,n}^{(n-1)} \end{bmatrix}$$

In the nonhermitian case, the resulting matrix has the same structure except for the portion above the diagonal; an element in position (i,j) with $i < j$ becomes the corresponding element of $TA^{(n-1)T*}$. In both the hermitian and nonhermitian cases, the subdiagonal of $TA^{(n-1)T*}$ is stored separate from *A* in the real vector *b*. In both cases, the diagonal of $A^{(n-1)}$ is also the diagonal of $TA^{(n-1)T*}$. In the hermitian case, this diagonal is also stored as the real vector *c*.

We note one difference between the ALGOL procedures in Reference 1 and the FORTRAN subroutines here. In the determination of the complex vector *tau*, a division occurs in the calculation of the complex quantity

$$\exp(i\phi_r) = a_{r+1,r}^{(n-1)} / |a_{r+1,r}^{(n-1)}|.$$

In the ALGOL procedures, $\exp(i\phi_r)$ is considered to be 1.0 if $|a_{r+1,r}^{(n-1)}| = 0.0$. In the FORTRAN subroutines given here, $\exp(i\phi_r)$ is considered to be 1.0 if

$$|a_{r+1,r}^{(n-1)}| \leq \gamma^{1/2} \times \text{tolerance}.$$

Again, γ is the square of the relative machine precision, and *tolerance* is the maximum absolute value of the real and imaginary parts of elements of $A^{(1)}$. This test affects only the calculation of *tau* and need not be "undone" during the back transformation of eigenvectors.

C. Subroutine CODIAG: Householder Reduction of Hermitian Matrices

This subroutine uses a 2-by-n-by-n real array rather than an n-by-n complex array to house matrix *A*. The complex vector *tau* is actually a 2-by-n real array. The vectors *b* and *c* are real n-component arrays.

The subroutine assumes that memory space for the full 2-by-n-by-n array has been reserved for *A* by the calling program. However, only the lower triangle of the original hermitian matrix needs to be supplied at the start. At the end of the reduction, the portion of *A* above the diagonal contains the information (if any) with which it was supplied at the start. In particular, this area can be used to store the upper triangle, except for the diagonal, of the original matrix. Then if the calling program stores the diagonal of the original matrix as a real vector *r* separate from *A*, the entire starting matrix can later be recovered for the testing of the computed eigenvectors.

```

SUBROUTINE CODIAG(A,N,M,GAMMA,TOLER,B,C,TAU)
C   AS INPUT, A IS THE HERMITIAN MATRIX OF ORDER N THAT IS TO BE
C   REDUCED TO TRIDIAGONAL REAL SYMMETRIC FORM BY THE METHOD OF
C   HOUSEHOLDER FOLLOWED BY A DIAGONAL UNITARY SIMILARITY. M IS THE
C   NUMBER USED IN DIMENSIONING THE ARRAYS IN THE CALLING PROGRAM.
C   GAMMA IS THE SQUARE OF THE RELATIVE MACHINE PRECISION. AS OUTPUT,
C   TOLER IS THE MAXIMUM ABSOLUTE VALUE OF THE REAL AND IMAGINARY PARTS
C   OF ELEMENTS OF INPUT MATRIX A. THE REAL VECTOR B CONTAINS THE
C   SUBDIAGONAL OF THE TRIDIAGONAL REAL MATRIX (WITH AN ADDITIONAL
C   ELEMENT B(N) = 0.0). REAL VECTOR C CONTAINS THE DIAGONAL OF THIS
C   REAL MATRIX. INFORMATION ON THE HOUSEHOLDER REDUCTION IS WRITTEN
C   OVER THE LOWER PORTION OF MATRIX A. COMPLEX VECTOR TAU CONTAINS
C   THE DIAGONAL OF THE DIAGONAL UNITARY MATRIX T.
    DIMENSION A(2,M,M),B(M),C(M),TAU(2,M)
    NM1 = N - 1
    TOLER = 0.0
    DO 21 I=1,N
    DO 21 J=1,I
    DO 21 L=1,2
    IF(ABSF(A(L,I,J)) - TOLER) 21,21,22
22  TOLER = ABSF(A(L,I,J))
21  CONTINUE

C   PERFORM N - 1 SIMILARITY TRANSFORMATIONS
    DO 23 NR=2,NM1
    NRM1 = NR - 1
    VR = 0.0
    TAU(1,NR) = 0.0
    TAU(2,NR) = 0.0
    TAU(2,1) = 0.0
    DO 24 L=NR,N
24  VR = VR + A(1,L,NRM1)**2 + A(2,L,NRM1)**2
    IF(GAMMA*TOLER**2 - VR) 25,23,23
25  IF(ABSF(A(1,NR,NRM1)) + ABSF(A(2,NR,NRM1))) 27,26,27
26  A(1,NR,NRM1) = SQRTF(VR)
    DELTA = VR
    TAU(1,1) = -A(1,NR,NRM1)
    GOTO 28
27  ROOT = SQRTF((A(1,NR,NRM1)**2 + A(2,NR,NRM1)**2)*VR)
    DELTA = VR + ROOT
    RATIO = VR/ROOT
    TAU(1,1) = -RATIO*A(1,NR,NRM1)
    TAU(2,1) = RATIO*A(2,NR,NRM1)
    A(1,NR,NRM1) = (RATIO + 1.0)*A(1,NR,NRM1)

```

```

      A(2,NR,NRM1) = (RATIO + 1.0)*A(2,NR,NRM1)
28      CONTINUE
C      THE MATRIX TO BE USED IN THE SIMILARITY TRANSFORMATION HAS BEEN
C      DETERMINED. THE TRANSFORMATION FOLLOWS.
      DO 29 J=NR,N
        TAU(1,J) = A(1,J,NRM1)/DELTA
        TAU(2,J) = A(2,J,NRM1)/DELTA
        C(J) = 0.0
        B(J) = 0.0
        DO 30 L=NR,J
          C(J) = C(J) + A(1,J,L)*A(1,L,NRM1) - A(2,J,L)*A(2,L,NRM1)
30        B(J) = B(J) + A(1,J,L)*A(2,L,NRM1) + A(2,J,L)*A(1,L,NRM1)
          JP1 = J + 1
          DO 29 L=JP1,N
            C(J) = C(J) + A(1,L,J)*A(1,L,NRM1) + A(2,L,J)*A(2,L,NRM1)
29          B(J) = B(J) + A(1,L,J)*A(2,L,NRM1) - A(2,L,J)*A(1,L,NRM1)
          RHO = 0.0
          DO 31 L=NR,N
            RHO = RHO + C(L)*TAU(1,L) + B(L)*TAU(2,L)
31          DO 32 I=NR,N
            DO 32 J=NR,I
              X1 = TAU(1,I)*C(J) + TAU(2,I)*B(J)
              X2 = TAU(2,I)*C(J) - TAU(1,I)*B(J)
              Q1 = C(I) - RHO*A(1,I,NRM1)
              Q2 = B(I) - RHO*A(2,I,NRM1)
              T1 = Q1*TAU(1,J) + Q2*TAU(2,J)
              T2 = Q2*TAU(1,J) - Q1*TAU(2,J)
              A(1,I,J) = A(1,I,J) - X1 - T1
              A(2,I,J) = A(2,I,J) - X2 - T2
32            CONTINUE
              TAU(1,NR) = TAU(1,1)
              TAU(2,NR) = TAU(2,1)
23          CONTINUE
C      THE MATRIX HAS BEEN REDUCED TO TRIDIAGONAL HERMITIAN FORM. THE
C      SUBDIAGONAL HAS BEEN TEMPORARILY STORED IN VECTOR TAU. STORE THE
C      DIAGONAL OF THE REDUCED MATRIX IN C.
      DO 33 I=1,N
33      C(I) = A(1,I,I)
C      PERFORM THE DIAGONAL UNITARY SIMILARITY
      IF(NM1) 35,35,34
34      TAU(1,N) = A(1,N,NM1)
      TAU(2,N) = -A(2,N,NM1)
35      TAU(1,1) = 1.0
      TAU(2,1) = 0.0
      B(N) = 0.0
      TESTBB = SQRTF(GAMMA)*TOLER
C      CALCULATE SUBDIAGONAL B OF THE REAL SYMMETRIC TRIDIAGONAL MATRIX.
C      CALCULATE TAU, THE DIAGONAL OF THE DIAGONAL UNITARY MATRIX T.
      DO 36 I=2,N
        IM1 = I - 1
        BB = SQRTF(TAU(1,I)**2 + TAU(2,I)**2)
        B(IM1) = BB
        IF(TESTBB - BB) 38,37,37
37      TAU(1,I) = 1.0
        TAU(2,I) = 0.0
        BB = 1.0
38      TT1 = TAU(1,I)*TAU(1,IM1) - TAU(2,I)*TAU(2,IM1)
        TT2 = TAU(1,I)*TAU(2,IM1) + TAU(2,I)*TAU(1,IM1)
        TAU(1,I) = TT1/BB
        TAU(2,I) = TT2/BB
36      CONTINUE
      END

```

D. Subroutine HESSEN: Householder Reduction of Nonhermitian Matrices

This subroutine also uses a 2-by-n-by-n real array to house the complex matrix A. The complex vector *tau* is actually a 2-by-n real array.

```

SUBROUTINE HESSEN(A,N,M,GAMMA,TOLER,B,TAU,F,G)
C   AS INPUT, A IS THE COMPLEX MATRIX OF ORDER N THAT IS TO BE REDUCED
C   TO AN UPPER HESSENBERG MATRIX WHOSE LOWER SUBDIAGONAL IS REAL.
C   THIS IS DONE BY THE METHOD OF HOUSEHOLDER FOLLOWED BY A DIAGONAL
C   UNITARY SIMILARITY. M IS THE NUMBER USED IN DIMENSIONING THE
C   ARRAYS IN THE CALLING PROGRAM. GAMMA IS THE SQUARE OF THE
C   RELATIVE MACHINE PRECISION. AS OUTPUT, TOLER IS THE MAXIMUM
C   ABSOLUTE VALUE OF THE REAL AND IMAGINARY PARTS OF ELEMENTS OF
C   INPUT MATRIX A. THE UPPER TRIANGLE OF THE OUTPUT MATRIX A HOUSES
C   THE HESSENBERG MATRIX EXCEPT FOR ITS LOWER SUBDIAGONAL, WHICH IS
C   STORED IN REAL VECTOR B (WITH AN ADDITIONAL ELEMENT B(N) = 0.0).
C   INFORMATION ON THE HOUSEHOLDER REDUCTION IS WRITTEN OVER THE LOWER
C   PORTION OF MATRIX A. THE COMPLEX VECTOR TAU GIVES THE DIAGONAL
C   OF THE DIAGONAL UNITARY MATRIX T. ARRAYS F AND G ARE USED ONLY BY
C   THIS SUBROUTINE BUT MUST BE DIMENSIONED IN THE CALLING PROGRAM.
C   DIMENSION A(2,M,M),TAU(2,M),B(M),F(2,M),G(2,M)
      NM1 = N - 1
      TOLER = 0.0
      DO 1 I=1,N
        DO 1 J=1,N
          DO 1 L=1,2
            IF(ABSF(A(L,I,J)) - TOLER) 1,1,2
          2   TOLER = ABSF(A(L,I,J))
        1   CONTINUE
      C
      C   PERFORM N - 1 SIMILARITY TRANSFORMATIONS
      DO 3 NR=2,NM1
        NRM1 = NR - 1
        VR = 0.0
        TAU(1,NR) = 0.0
        TAU(2,NR) = 0.0
        TAU(2,1) = 0.0
        DO 4 L=NR,N
          4   VR = VR + A(1,L,NRM1)**2 + A(2,L,NRM1)**2
          IF(GAMMA*TOLER**2 - VR) 5,3,3
          5   IF(ABSF(A(1,NR,NRM1)) + ABSF(A(2,NR,NRM1))) 7,6,7
          6   A(1,NR,NRM1) = SQRTF(VR)
          TAU(1,1) = -A(1,NR,NRM1)
          DELTA = VR
          GOTO 8
          7   ROOT = SQRTF((A(1,NR,NRM1)**2 + A(2,NR,NRM1)**2)*VR)
          DELTA = VR + ROOT
          RATIO = VR/ROOT
          TAU(1,1) = -RATIO*A(1,NR,NRM1)
          TAU(2,1) = RATIO*A(2,NR,NRM1)
          A(1,NR,NRM1) = (RATIO + 1.0)*A(1,NR,NRM1)
          A(2,NR,NRM1) = (RATIO + 1.0)*A(2,NR,NRM1)
          8   CONTINUE
      C   THE MATRIX TO BE USED IN THE SIMILARITY TRANSFORMATION HAS BEEN
      C   DETERMINED. THE TRANSFORMATION FOLLOWS.
      DO 9 J=1,N
        G(1,J) = 0.0
        G(2,J) = 0.0
        F(1,J) = 0.0
        F(2,J) = 0.0
        DO 9 L=NR,N
          G(1,J) = G(1,J) + A(1,J,L)*A(1,L,NRM1) - A(2,J,L)*A(2,L,NRM1)
          G(2,J) = G(2,J) + A(2,J,L)*A(1,L,NRM1) + A(1,J,L)*A(2,L,NRM1)
          IF(J - NR) 9,10,10

```

```

10  F(1,J) = F(1,J) + A(1,L,J)*A(1,L,NRM1) + A(2,L,J)*A(2,L,NRM1)
    F(2,J) = F(2,J) + A(2,L,J)*A(1,L,NRM1) - A(1,L,J)*A(2,L,NRM1)
9   CONTINUE
    RHO1 = 0.0
    RHO2 = 0.0
    DO 11 L=NR,N
      TAU(1,L) = A(1,L,NRM1)/DELTA
      TAU(2,L) = A(2,L,NRM1)/DELTA
      RHO1 = RHO1 + G(1,L)*TAU(1,L) + G(2,L)*TAU(2,L)
11  RHO2 = RHO2 + G(2,L)*TAU(1,L) - G(1,L)*TAU(2,L)
      DO 12 I=1,N
        DO 12 J=NR,N
          IF(NR - I) 14,14,13
13    A(1,I,J) = A(1,I,J) - G(1,I)*TAU(1,J) - G(2,I)*TAU(2,J)
        A(2,I,J) = A(2,I,J) - G(2,I)*TAU(1,J) + G(1,I)*TAU(2,J)
        GOTO 12
14    X1 = TAU(1,I)*F(1,J) - TAU(2,I)*F(2,J)
        X2 = TAU(2,I)*F(1,J) + TAU(1,I)*F(2,J)
        Q1 = G(1,I) - RHO1*A(1,I,NRM1) + RHO2*A(2,I,NRM1)
        Q2 = G(2,I) - RHO2*A(1,I,NRM1) - RHO1*A(2,I,NRM1)
        T1 = Q1*TAU(1,J) + Q2*TAU(2,J)
        T2 = Q2*TAU(1,J) - Q1*TAU(2,J)
        A(1,I,J) = A(1,I,J) - X1 - T1
        A(2,I,J) = A(2,I,J) - X2 - T2
12  CONTINUE
      TAU(1,NR) = TAU(1,1)
      TAU(2,NR) = TAU(2,1)
3   CONTINUE
C
C   THE MATRIX IS NOW IN UPPER HESSENBERG FORM.  PERFORM THE DIAGONAL
C   UNITARY SIMILARITY.
    TAU(1,1) = 1.0
    TAU(2,1) = 0.0
    B(N) = 0.0
    IF(NM1) 16,16,15
15  TAU(1,N) = A(1,N,NM1)
    TAU(2,N) = -A(2,N,NM1)
16  TESTBB = SQRTF(GAMMA)*TOLER
    DO 17 I=2,N
      IM1 = I - 1
C   PERFORM THE SIMILARITY WHICH CAUSES THE ELEMENT IN POSITION
C   (I,I-1) TO BECOME REAL.
      BB = SQRTF(TAU(1,I)**2 + TAU(2,I)**2)
      B(IM1) = BB
      IF(TESTBB - BB) 19,18,18
18  TAU(1,I) = 1.0
      TAU(2,I) = 0.0
      BB = 1.0
19  TT1 = TAU(1,I)*TAU(1,IM1) - TAU(2,I)*TAU(2,IM1)
      TT2 = TAU(1,I)*TAU(2,IM1) + TAU(2,I)*TAU(1,IM1)
      TAU(1,I) = TT1/BB
      TAU(2,I) = TT2/BB
C   MULTIPLY ROW I BY TAU(I)
      IP1 = I + 1
      DO 20 L=IP1,N
        TT1 = A(1,I,L)*TAU(1,I) - A(2,I,L)*TAU(2,I)
        TT2 = A(2,I,L)*TAU(1,I) + A(1,I,L)*TAU(2,I)
        A(1,I,L) = TT1
20  A(2,I,L) = TT2
C   MULTIPLY COLUMN I BY THE COMPLEX CONJUGATE OF TAU(I)
      DO 17 L=1,IM1
        TT1 = A(1,L,I)*TAU(1,I) + A(2,L,I)*TAU(2,I)
        TT2 = A(2,L,I)*TAU(1,I) - A(1,L,I)*TAU(2,I)
        A(1,L,I) = TT1
17  A(2,L,I) = TT2
    END

```

E. Subroutine *CHANGE*: Back Transformation of Eigenvectors

This procedure transforms the eigenvectors of $TA^{(n-1)}T^*$ into eigenvectors of $A^{(1)}$. Here $TA^{(n-1)}T^*$ is the matrix calculated by either of the subroutines *CODIAG* or *HESSEN*. If y is an eigenvector of this matrix, then $P_2P_3 \cdots P_{n-1}T^*y$ is an eigenvector of $A^{(1)}$.

The procedure uses τu , the diagonal of T , to construct T^* . It uses the vectors $u^{(2)}, \dots, u^{(n-1)}$ which are stored in the lower portion of matrix A to reconstruct the Householder matrices $P_2, \dots, P_{n-1}$. Each P_r also requires the corresponding number δ since $P_r = I_{r-1} \oplus H_r$, where $H_r = I - u^{(r)}u^{(r)*}/\delta$. But each δ is obtained from $\delta = |u_1^{(r)}| \cdot |b_{r-1}|$, which requires that vector b also be supplied to the subroutine.

We use Z_r to denote the complex matrix whose columns are some or all of the eigenvectors of $A^{(r)}$. The calculations proceed as follows: First, Z_{n-1} is obtained by premultiplying each eigenvector of the reduced matrix by T^* . Then we perform

$$Z_{r-1} = P_r Z_r = (I_{r-1} \oplus H_r) Z_r, \quad r = n-1, \dots, 3, 2.$$

In passing from Z_r to Z_{r-1} , only the last $n-r+1$ rows of Z_r are altered. We denote this portion of Z_r as the matrix $\underline{Z}_r$. Thus we have

$$H_r \underline{Z}_r = (I - u^{(r)}u^{(r)*}/\delta) \underline{Z}_r = \underline{Z}_r - u^{(r)}\alpha,$$

in which the row vector $\alpha = u^{(r)*} \underline{Z}_r / \delta$.

If $A^{(1)}$ is hermitian, the input matrix $Z1$ is just the real matrix of eigenvectors calculated by subroutine *EIGEN*. Its columns are some or all of the eigenvectors of the real symmetric tridiagonal matrix. The matrix $Z2$ should be set equal to the zero matrix before subroutine *CHANGE* is called.

If $A^{(1)}$ is nonhermitian, the eigenvectors of the reduced matrix will in general already be complex so that the input matrix $Z2$ is not necessarily zero.

```

C   MATRIX. ARRAY ALPHA IS USED ONLY IN THIS SUBROUTINE BUT MUST BE
C   DIMENSIONED IN THE CALLING PROGRAM.
      DIMENSION B(M),A(2,M,M),TAU(2,M),Z1(M,M),Z2(M,M),ALPHA(2,M)
C
C   TRANSFORM THE EIGENVECTORS OF THE REAL SYMMETRIC TRIDIAGONAL
C   MATRIX TO THOSE OF THE HERMITIAN TRIDIAGONAL MATRIX OBTAINED BY
C   THE METHOD OF HOUSEHOLDER.
      DO 41 J=2,N
      DO 41 K=1,M1
      P1 = Z1(J,K)*TAU(1,J) + Z2(J,K)*TAU(2,J)
      P2 = Z2(J,K)*TAU(1,J) - Z1(J,K)*TAU(2,J)
      Z1(J,K) = P1
41   Z2(J,K) = P2
C
C   RECOVER THE HOUSEHOLDER MATRICES IN REVERSE ORDER.
      DO 42 L=3,N
      NR = N - L + 2
      NRM1 = NR - 1
      IF(GAMMA*TOLER**2 - B(NRM1)) 43,42,42
43   DELTA = B(NRM1)*SQRTF(A(1,NR,NRM1)**2 + A(2,NR,NRM1)**2)
      DO 44 J=1,M1
      ALPHA(1,J) = 0.0
      ALPHA(2,J) = 0.0
      DO 45 K=NR,N
      ALPHA(1,J) = ALPHA(1,J) + A(1,K,NRM1)*Z1(K,J) + A(2,K,NRM1)*Z2(K,J)
45   ALPHA(2,J) = ALPHA(2,J) - A(2,K,NRM1)*Z1(K,J) + A(1,K,NRM1)*Z2(K,J)
      ALPHA(1,J) = ALPHA(1,J)/DELTA
44   ALPHA(2,J) = ALPHA(2,J)/DELTA
C   TRANSFORM THE EIGENVECTORS FROM STAGE NR TO STAGE NR-1.
      DO 42 J=1,M1
      DO 42 K=NR,N
      Z1(K,J) = Z1(K,J) - A(1,K,NRM1)*ALPHA(1,J) + A(2,K,NRM1)*ALPHA(2,J)
      Z2(K,J) = Z2(K,J) - A(1,K,NRM1)*ALPHA(2,J) - A(2,K,NRM1)*ALPHA(1,J)
42   CONTINUE
      END

```

V. TEST RESULTS

The FORTRAN subroutines given in Section IV of this report were compiled with the appropriate calling programs and tested on the CDC-3600 computer at Argonne National Laboratory.

For the first numerical example, we give a 5-by-5 hermitian matrix whose complete eigensystem was calculated by subroutines *CODIAG*, *EIGEN*, and *CHANGE*.

First, the starting matrix $A^{(1)}$ is given. Then the entire matrix A is listed with the quantities it contains at the end of subroutine *CODIAG*. The structure of this matrix is noted in Section IV-B. Following this, the diagonal of the unitary matrix T and the subdiagonal b of $TA^{(n-1)}T^*$ are given. The Euclidean norm of the reduced matrix is calculated by subroutine *EIGEN* during the eigenvalue calculation and is printed next. Then the eigenvalues and eigenvectors of $TA^{(n-1)}T^*$ are listed as calculated by *EIGEN*. And finally, the eigenvalues and eigenvectors of $A^{(1)}$ are given. Here the eigenvectors of $A^{(1)}$ have been obtained from those of $TA^{(n-1)}T^*$ by subroutine *CHANGE*. Following each eigenvector, we give the Euclidean norm of the residual; i.e., we give $\|A^{(1)}v_j - \lambda_j v_j\|_2$, where v_j is eigenvector number j of $A^{(1)}$ and corresponds to eigenvalue λ_j .

STARTING MATRIX...

ORDER

5

HERMITIAN MATRIX

ROW	COLUMN	REAL PART	IMAGINARY PART
1	1	-8.45000000-001	0.00000000+000
1	2	5.20000000+000	1.03000000-001
1	3	3.01000000-001	-4.54000000-002
1	4	-9.60000000+000	9.36000000-001
1	5	7.34000000-002	7.26000000+000
2	1	5.20000000+000	-1.03000000-001
2	2	-6.20000000+000	0.00000000+000
2	3	-3.39000000+000	-4.07000000-001
2	4	1.22000000-001	9.10000000-001
2	5	4.19000000+000	-3.66000000+000
3	1	3.01000000-001	4.54000000-002
3	2	-3.39000000+000	4.07000000-001
3	3	1.90000000-002	0.00000000+000
3	4	9.35000000-001	-2.71000000-001
3	5	-5.72000000-002	2.82000000+000
4	1	-9.60000000+000	-9.36000000-001
4	2	1.22000000-001	-9.10000000-001
4	3	9.35000000-001	2.71000000-001
4	4	7.21000000+000	0.00000000+000
4	5	3.37000000-001	6.03000000-002
5	1	7.34000000-002	-7.26000000+000
5	2	4.19000000+000	3.66000000+000
5	3	-5.72000000-002	-2.82000000+000
5	4	3.37000000-001	-6.03000000-002
5	5	-1.23000000+000	0.00000000+000

QUANTITIES WRITTEN OVER THE STARTING MATRIX

ROW	COLUMN	REAL PART	IMAGINARY PART
1	1	-8.45000000-001	0.00000000+000
1	2	5.20000000+000	1.03000000-001
1	3	3.01000000-001	-4.54000000-002
1	4	-9.60000000+000	9.36000000-001
1	5	7.34000000-002	7.26000000+000
2	1	1.83462751+001	-3.63397371-001
2	2	9.38304796-001	-1.16415322-010
2	3	-3.39000000+000	-4.07000000-001
2	4	1.22000000-001	9.10000000-001
2	5	4.19000000+000	-3.66000000+000
3	1	3.01000000-001	4.54000000-002
3	2	6.20844966+000	-5.88073847+000
3	3	-3.61993001+000	0.00000000+000
3	4	9.35000000-001	-2.71000000-001
3	5	-5.72000000-002	2.82000000+000
4	1	-9.60000000+000	-9.36000000-001
4	2	6.81129531+000	2.97258158+000
4	3	3.48508707+000	1.18715342+000
4	4	-1.57258011+000	0.00000000+000
4	5	3.37000000-001	6.03000000-002
5	1	7.34000000-002	-7.26000000+000
5	2	-2.65181948+000	-7.08762719-001
5	3	-2.72061157+000	2.96786834-001
5	4	2.54201170-001	-3.26101076+000
5	5	4.05320532+000	1.16415322-010

DIAGONAL (TAU) OF MATRIX T, SUBDIAGONAL (B) OF REAL SYMM. TRIDIAGONAL MATRIX

I	REAL PART OF TAU(I)	IMAG. PART OF TAU(I)	B(I)
1	1.0000000000+000	0.0000000000+000	1.3148853742+001
2	-9.9980388535-001	-1.9803807730-002	7.9455424739+000
3	7.1224700270-001	7.0192891886-001	2.8580249122+000
4	-9.0053766854-001	-4.3477799796-001	3.2709034582+000
5	3.6347694664-001	-9.3160319303-001	0.0000000000+000

EUCLIDEAN NORM OF REAL SYMMETRIC MATRIX = 2.33105566+001

EIGENSYSTEM OF TRIDIAGONAL REAL SYMMETRIC MATRIX

EIGENVALUE NO. 1 IS 1.5180165225+001

EIGENVECTOR NO.	COMPONENT NO.	COMPONENT
1	1	6.0020595562-001
1	2	7.3150099593-001
1	3	3.1790337412-001
1	4	5.7536753536-002
1	5	1.6913619515-002

EIGENVALUE NO. 2 IS 5.6787293557+000

EIGENVECTOR NO.	COMPONENT NO.	COMPONENT
2	1	-7.6036024999-002
2	2	-3.7724843397-002
2	3	1.0332268695-001
2	4	4.4104122764-001
2	5	8.8746967099-001

EIGENVALUE NO. 3 IS -8.3398680005-001

EIGENVECTOR NO.	COMPONENT NO.	COMPONENT
3	1	-3.6469200754-001
3	2	-3.0545825733-004
3	3	6.0358663298-001
3	4	5.8921288768-001
3	5	-3.9434882506-001

EIGENVALUE NO. 4 IS -5.1498456275+000

EIGENVECTOR NO.	COMPONENT NO.	COMPONENT
4	1	4.0197656815-001
4	2	-1.3160440491-001
4	3	-5.6437980204-001
4	4	6.6798643225-001
4	5	-2.3741247795-001

EIGENVALUE NO. 5 IS -1.5921062149+001

EIGENVECTOR NO.	COMPONENT NO.	COMPONENT
5	1	-5.8256837542-001
5	2	6.6795457663-001
5	3	-4.5323336442-001
5	4	9.3778765455-002
5	5	-1.5356822904-002

EIGENSYSTEM OF THE HERMITIAN STARTING MATRIX

EIGENVALUE NO. 1 IS 1.5180165225+001

VECTOR NO.	ELEMENT NO.	REAL PART	IMAGINARY PART
1	1	6.0020595562-001	0.0000000000+000
1	2	1.1153667099-001	-1.0113204497-001
1	3	-8.0164384174-003	4.4444702492-002
1	4	-7.3128735544-001	-9.0661151810-002
1	5	4.6033597956-002	-2.6443404390-001

NORM OF RESIDUAL = 1.3367381725-009

EIGENVALUE NO. 2 IS 5.6787293557+000

VECTOR NO.	ELEMENT NO.	REAL PART	IMAGINARY PART
2	1	-7.6036024999-002	0.0000000000+000
2	2	3.2040141319-001	2.3552437565-001
2	3	-5.7099241395-001	-7.4653168438-002
2	4	-2.8835772888-001	1.3085291647-001
2	5	3.0068390130-002	6.3506497287-001

NORM OF RESIDUAL = 9.4724473550-010

EIGENVALUE NO. 3 IS -8.3398680005-001

VECTOR NO.	ELEMENT NO.	REAL PART	IMAGINARY PART
3	1	-3.6469200754-001	0.0000000000+000
3	2	-3.6716877160-001	-5.5136126637-002
3	3	3.1017423861-001	5.2476001510-001
3	4	-4.3312631669-001	-1.6805017327-001
3	5	-1.4500284666-001	3.4741994195-001

NORM OF RESIDUAL = 6.6193221839-010

EIGENVALUE NO. 4 IS -5.1498456275+000

VECTOR NO.	ELEMENT NO.	REAL PART	IMAGINARY PART
4	1	4.0197656815-001	0.0000000000+000
4	2	3.7506135334-001	4.3149942367-001
4	3	2.8165156458-001	4.5739951652-001
4	4	2.7584267210-001	1.0775984595-002
4	5	-3.5437344505-001	1.4568563954-001

NORM OF RESIDUAL = 1.1434384996-009

EIGENVALUE NO. 5 IS -1.5921062149+001

VECTOR NO.	ELEMENT NO.	REAL PART	IMAGINARY PART
5	1	-5.8256837542-001	0.0000000000+000
5	2	5.6174061319-001	1.9731927448-001
5	3	6.6436395422-002	4.2639475306-002
5	4	-2.5431770783-001	2.1514203799-003
5	5	-1.1024471280-001	-4.7229094207-001

NORM OF RESIDUAL = 1.6758039001-009

As the second numerical example, we give a 6-by-6 nonhermitian matrix and its reduction to upper Hessenberg form in which the lower subdiagonal is real. First, the starting matrix $A^{(1)}$ is given. Following it is a list of the vectors $u^{(2)}$, $u^{(3)}$, $u^{(4)}$, and $u^{(5)}$ that are used in the Householder reduction and which are written over the lower portion of the matrix by the time subroutine *HESSEN* has completed the reduction. Next, the upper triangle, i.e., the diagonal and everything above it, of the reduced matrix $TA^{(n-1)}T^*$ is given. Finally, the diagonal τ of the unitary matrix T and the lower subdiagonal b of the reduced matrix are given.

STARTING MATRIX...

ORDER

6 NONHERMITIAN MATRIX

ROW	COLUMN	REAL PART	IMAGINARY PART
1	1	2.13000000-001	-3.14000000+000
1	2	6.66000000+000	8.76000000-001
1	3	-1.01000000-001	-3.05000000-001
1	4	7.34000000-002	-7.26000000+000
1	5	4.19000000+000	3.66000000+000
1	6	-4.78000000+000	6.78000000+000
2	1	-5.27000000-002	-2.82000000+000
2	2	4.19000000+000	-3.66000000+000
2	3	3.37000000-001	-6.03000000-002
2	4	5.31000000-001	8.12000000-002
2	5	-1.23000000+000	0.00000000+000
2	6	-6.27000000-001	5.55000000-002
3	1	7.34000000-002	7.26000000+000
3	2	8.07000000-001	5.26000000+000
3	3	8.09000000+000	-3.14000000-001
3	4	3.33000000+000	8.02000000-002
3	5	3.01000000-001	4.54000000-002
3	6	5.31000000-001	-8.12000000-002
4	1	4.03000000-001	-5.32000000-002
4	2	5.55000000-002	7.71000000-001
4	3	-1.01000000-001	3.05000000-001
4	4	-3.39000000+000	4.07000000-001
4	5	5.20000000+000	1.03000000-001
4	6	7.81000000-002	5.25000000-001
5	1	8.08000000-001	-4.41000000+000
5	2	8.07000000-001	-5.26000000+000
5	3	-5.44000000-001	-6.24000000-002
5	4	8.27000000-002	1.35000000-002
5	5	-1.09000000-002	-3.91000000+000
5	6	1.80000000-001	-3.72000000+000
6	1	3.01000000-001	-4.54000000-002
6	2	-9.60000000+000	9.36000000-001
6	3	9.35000000-001	-2.71000000-001
6	4	6.66000000+000	-8.76000000-001
6	5	5.20000000+000	-1.03000000-001
6	6	8.08000000-001	4.41000000+000

QUANTITIES WRITTEN OVER THE LOWER TRIANGLE

ROW	COLUMN	REAL PART	IMAGINARY PART
2	1	-2.20890184-001	-1.18199301+001
3	1	7.34000000-002	7.26000000+000
4	1	4.03000000-001	-5.32000000-002
5	1	8.08000000-001	-4.41000000+000
6	1	3.01000000-001	-4.54000000-002
3	2	6.88861269+000	3.00143389+000
4	2	-2.37625302+000	-2.37842587-001
5	2	1.34794188+000	2.01225948+000
6	2	1.43897628+000	-1.17995231+000
4	3	-8.55704008+000	-8.53485506-001
5	3	-3.19861569-002	4.50530106-001
6	3	-6.20721966-001	-4.40114534+000
5	4	-9.70724167-001	3.90268557+000
6	4	1.92143351+000	-1.81558248+000

UPPER TRIANGLE OF REDUCED MATRIX

ROW	COLUMN	REAL PART	IMAGINARY PART
1	1	2.13000000-001	-3.14000000+000
1	2	2.48455490+000	-3.97439526+000
1	3	-1.00839179+001	1.80709263+000
1	4	3.95659094+000	2.33984048-001
1	5	1.93541019+000	-9.51328813-001
1	6	5.09142796-001	7.08296895+000
2	2	5.36021600+000	-3.98607212+000
2	3	1.20401686+000	5.00866500+000
2	4	3.78717592-001	4.58872333+000
2	5	2.98825609-001	-2.19060689+000
2	6	-3.74331633-001	4.81356683+000
3	3	-3.61239629+000	-2.15932606+000
3	4	2.51740544+000	-2.32443449+000
3	5	4.79397881-001	-4.62044853+000
3	6	2.42494620-001	-2.19813477+000
4	4	3.69270426+000	9.35638632-001
4	5	-5.18478606+000	7.17656033-001
4	6	-3.68525768-002	5.98586574+000
5	5	-8.88407086-001	5.61365882+000
5	6	4.65396071+000	-9.46076388-001
6	6	5.13498312+000	-3.47089928+000

DIAGONAL TAU OF MATRIX T, AND SUBDIAGONAL B OF THE REDUCED MATRIX

I	REAL PART OF TAU(I)	IMAG. PART OF TAU(I)	B(I)
1	1.0000000000+000	0.0000000000+000	9.0015015553+000
2	1.8684680836-002	-9.9982542614-001	4.7573150595+000
3	3.8224153907-001	9.2406244695-001	5.4602457511+000
4	4.7206591527-001	8.8156325452-001	2.8796388500+000
5	-7.4155037788-001	6.7089718821-001	3.7569264936+000
6	8.4574331443-001	5.3358996076-001	0.0000000000+000

APPENDIX

Subroutine EIGEN: Eigensystems
of Real Symmetric Matrices

Subroutine *EIGEN* calculates the eigensystem of a real symmetric (not necessarily tridiagonal) matrix. The subroutine was written by Mr. B. S. Garbow of Argonne National Laboratory. It was prepared from another version of the subroutine, which used some special features of CDC-3600 FORTRAN to conserve memory space and shorten the length of the routine. I am grateful to Mr. Garbow for preparing this version of the subroutine, which uses no features of FORTRAN not found in IBM FORTRAN II.

There is a single machine-dependent quantity, *E1*, defined at the first executable statement. Its value is related to fractional accuracy obtainable; for the IBM-7090, for example, 5×10^{-8} should replace the 1×10^{-10} used on the 3600.

The subroutine uses, in sequence, the Householder reduction to take the symmetric matrix to tridiagonal form, the Givens bisection technique to obtain the eigenvalues of the tridiagonal matrix, Wielandt inverse iteration to obtain the eigenvectors of the tridiagonal matrix, and a back transformation to obtain the eigenvectors of the original real symmetric matrix.

We first describe the calling sequence in which one wishes to compute the eigensystem of a real symmetric matrix. Matrix *A* is initially this *nsub*-by-*nsub* matrix. The entire matrix must be supplied to the subroutine. The routine later replaces some or all of the columns of *A* with some or all of the eigenvectors of the original matrix. If eigenvectors are desired, the 100-by-100 matrix *B* must be available for intermediate storage and must be dimensioned in the calling program. If no eigenvectors are desired, *B* may be treated as a dummy variable and need not be dimensioned in the calling program. The absolute value of *msub* indicates how many eigenvalues are desired, beginning with the largest eigenvalue. The real vector *valu* contains as output $|msub|$ of the eigenvalues of the symmetric matrix. *valu*(1) is the largest (most positive) eigenvalue, *valu*(2) is the second largest, etc. If multiple roots are encountered, they are repeated in *valu* as often as their multiplicities. If *msub* > 0, an eigenvector will be calculated for each eigenvalue. If *msub* < 0, no eigenvectors will be calculated. The output quantity *srnorm* is the Euclidean norm of the matrix and is calculated even if *msub* = 0. The final variable in the calling sequence is, indeed, a dummy variable.

When the eigensystem of a hermitian matrix is to be found, the Householder reduction and back transformation necessarily require complex arithmetic and are performed by subroutines *CODIAG* and *CHANGE*. Thus, in using subroutine *EIGEN*, we wish to skip the Householder reduction that is provided in the routine, supply the tridiagonal matrix to the routine, calculate the eigenvalues and eigenvectors of this matrix, and then skip the back transformation provided in the routine. Mr. Garbow has made allowance for the use of his subroutine in this manner.

The calling sequence in this case is described as follows: *Nsub* is a negative integer whose absolute value is the order of the tridiagonal matrix.

(This causes the Householder reduction and back transformation to be skipped; the subroutine effectively begins after statement 200 and ends with statement 970.) *B* is a dummy variable and need not be dimensioned as an array in the calling program. The diagonal and subdiagonal of the tridiagonal matrix are supplied to the subroutine through the first 200 words of the *COMMON* area, which uses 700 words total. The diagonal is stored in the initial segment of *diag*, and the subdiagonal is stored in the initial segment of *superd*. As before, the absolute value of *msub* is the number of eigenvalues to be calculated, beginning with the largest eigenvalue. The eigenvalues calculated by the subroutine are stored in *valu*. If *msub* > 0, an eigenvector will be calculated for each eigenvalue and will be stored in a column of *A*, which is a real matrix of order 100, used only as an output array. If *msub* < 0, no eigenvectors will be calculated. Again, *srnorm* is the Euclidean norm of the (tridiagonal) matrix and is calculated even if *msub* = 0. The final variable in the calling sequence is a dummy variable.

The function *SIGNF* that is used in the subroutine is defined as

$$\text{SIGNF}(x1, x2) = \begin{cases} -|x1|, & \text{if } x2 < 0 \\ |x1|, & \text{if } x2 > 0. \end{cases}$$

The function (prudently or not) considers +0.0 to be positive and -0.0 to be negative. The function *XABSF* takes the absolute value of an integer-valued argument and is an integer-valued function. The function *ABSF* takes the absolute value of a real-valued argument and is real-valued.

In the calculation of the eigenvectors that correspond to a multiple eigenvalue, subroutine *EIGEN* includes a technique that is mentioned in Reference 3 but not incorporated into ALGOL procedure "tridi-inverse iteration." When the calculation of the k^{th} eigenvector ($k > 1$) corresponding to some multiple eigenvalue starts, the starting vector used in the Wielandt inverse iteration is orthogonalized with respect to the $k - 1$ eigenvectors already calculated. This removes the necessity for perturbing the eigenvalue and, in our experience at Argonne, has been more successful in obtaining independent eigenvectors for a multiple eigenvalue than the technique employed in the ALGOL procedure. Subroutine *EIGEN* performs a final orthogonalization of the eigenvectors for a multiple eigenvalue.

The techniques used in subroutine *EIGEN* for the eigensystem calculations are described in References 7-9.

```

SUBROUTINE EIGEN (A,B,NSUB,VALU,MSUB,SRNORM,DUMMY)
C
C   EIGENVALUES AND EIGENVECTORS OF A REAL SYMMETRIC MATRIX
C
  DIMENSION A(100,100), B(100,100), VALL(100)
  DIMENSION  DIAG(100), SUPERD(100), WVEC(100), PVEC(100),
1    QVEC(100), VALL(100), Q(100), U(100),
2    INDEX(100), FACTOR(100), V(100), T(100,3)
  COMMON DIAG, SUPERD, WVEC, PVEC, QVEC, VALL, Q, U, INDEX, FACTOR,
1    V, T, ANORM2, ANORM, VTEMP, VNORM2, VNORM
  EQUIVALENCE (WVEC,VALL, FACTOR,U), (PVEC,QVEC,Q,V), (I1,T1),
1    (I2,T2,ITER), (TEMP, T0), (SUM,MATCH), (I,P),
2    (DIV,SCALAR,TAU), (ANORM2,ANORM,SUPERD(100)),
3    (VTEMP,VNORM2,VNORM,INDEX(100))
C
```

```

C      HOUSEHOLDER SIMILARITY TRANSFORMATION TO CO-DIAGONAL FORM
C      -----
C      E1=1.0E-10
C      N=NSUB
C      M=MSUB
C      IF (M) 50, 50, 5
C
C      GENERATE IDENTITY MATRIX
C
C      5 IF (N-1) 220, 50, 10
C      10 DO 40 I=2,N
C          DO 40 J=2,N
C              B(J,I)=0.0
C              IF (I-J) 40, 25, 40
C      25 B(J,I)=1.0
C      40 CONTINUE
C
C      50 DO 200 I=1,N
C
C      REDUCE COLUMN OF MATRIX
C
C      I1=I+1
C      I2=I1+1
C      IF (I2-N) 65, 65, 160
C      65 SUM=0.0
C          DO 70 J=I2,N
C      70 SUM=SUM+A(J,I)**2
C          IF (SUM) 75,160, 75
C      75 J=I1
C          TEMP=A(J,I)
C          SUM=SQRTF(SUM+ TEMP **2)
C          A(J,I)=-SIGNF(SUM, TEMP)
C          WVEC(J)=SQRTF(1.0+ABSF(TEMP)/SUM)
C          DIV=SIGNF(WVEC(J)*SUM, TEMP)
C          DO 85 J=I2,N
C      85 WVEC(J)=A(J,I)/DIV
C          SCALAR=0.0
C          DO 95 J=I1,N
C              PVEC(J)=0.0
C          DO 90 K=I1,N
C      90 PVEC(J)=PVEC(J)+A(K,J)*WVEC(K)
C          SCALAR=SCALAR+PVEC(J)*WVEC(J)
C      95 CONTINUE
C          SCALAR=SCALAR/2.0
C          DO 120 J=I1,N
C              QVEC(J)=PVEC(J)-SCALAR*WVEC(J)
C          DO 120 K=I1,J
C              A(K,J)=A(K,J)-(WVEC(K)*QVEC(J)+WVEC(J)*QVEC(K))
C              A(J,K)=A(K,J)
C      120 CONTINUE
C          IF (M) 160, 160, 130
C
C      SAVE ROTATION FOR LATER APPLICATION TO CO-DIAGONAL VECTORS
C
C      130 DO 150 K=2,N
C          TEMP=0.0
C          DO 140 J=I1,N
C      140 TEMP=TEMP+WVEC(J)*B(J,K)
C          DO 150 J=I1,N
C              B(J,K)=B(J,K)-WVEC(J)*TEMP
C      150 CONTINUE
C
C      MOVE CO-DIAGONAL FORM ELEMENTS FOR ITERATIVE PROCEDURE
C

```

```

160 J=I
    DIAG(I)=A(J,I)
    SUPERD(I)=A(J+1,I)
200 CONTINUE
C
C      GIVENS EIGENVALUE ITERATION FROM STURM CHAIN OF CO-DIAGONAL MINORS
C      -----
220 N=XABSF(N)
    M=XABSF(M)
C
C      CALCULATE NORM OF MATRIX AND INITIALIZE EIGENVALUE BOUNDS
C
    ANORM2=DIAG(1)**2
    IF (N-1) 235, 235, 225
225 DO 230 L=2,N
    Q(L-1)=SUPERD(L-1)**2
    ANORM2=DIAG(L)**2+Q(L-1)+Q(L-1)+ANORM2
230 CONTINUE
235 ANORM=SQRTF(ANORM2)
    IF (M) 1000, 1000, 238
238 DO 240 L=1,M
    VALU(L)=ANORM
    VALL(L)=-ANORM
240 CONTINUE
    EPS1=ANORM*E1
    IF (EPS1) 250, 1000, 250
C
250 DO 570 L=1,M
C
C      CHOOSE NEW TRIAL VALUE WHILE TESTING BOUNDS FOR CONVERGENCE
C
260 TAU=(VALU(L)+VALL(L))/2.0
    IF (2.0*(TAU-VALL(L))-EPS1) 570, 570, 270
C
C      DETERMINE SIGNS OF PRINCIPAL MINORS
C
270 MATCH=0
    T2=0.0
    T1=1.0
    DO 450 L1=1,N
    P=DIAG(L1)-TAU
    IF (T2) 330, 300, 330
300 T1=SIGNF(1.0,T1)
330 IF (T1) 400, 370, 400
370 T0=-SIGNF(1.0,T2)
    T2=0.0
    IF (Q(L1-1)) 410, 300, 410
400 T0=P-Q(L1-1)*T2/T1
    T2=1.0
C
C      COUNT AGREEMENTS IN SIGN (ZERO CONSIDERED POSITIVE)
C
410 IF (T0) 440, 420, 430
420 T2=T1
    IF (T2) 440, 430, 430
430 MATCH=MATCH+1
440 T1=T0
450 CONTINUE
C
C      ESTABLISH TIGHTER BOUNDS ON EIGENVALUES
C
    DO 530 L1=L,M
    IF (L1-MATCH) 500, 500, 470
470 IF (VALU(L1)-TAU) 260, 260, 480
480 VALU(L1)=TAU
    GO TO 530

```

```

500 VALL(L1)=TAU
530 CONTINUE
    GO TO 260
C
570 CONTINUE
C
C      EIGENVECTORS OF CO-DIAGONAL SYMMETRIC MATRIX--INVERSE ITERATION
C      -----
C      M=MSUB
C      IF (M) 1000, 1000, 710
C
710 DO 970 I=1,M
C
C      CHECK FOR REPEATED VALUE
C
      IF (I-1) 725, 725, 720
720 IF (VALU(I-1)-VALU(I)-EPS1) 730, 725, 725
725 I1=-1
730 I1=I1+1
C
C      TRIANGULARIZE CO-DIAGONAL FORM AFTER EIGENVALUE SUBTRACTION
C
      DO 760 L=1,N
      V(L)=EPS1
      T(L,2)=DIAG(L)-VALU(I)
      IF (L-N) 740, 735, 740
735 T(L,3)=0.0
      GO TO 760
740 T(L,3)=SUPERD(L)
      IF (T(L,3)) 750, 745, 750
745 T(L,3)=EPS1
750 T(L+1,1)=T(L,3)
760 CONTINUE
      DO 820 J=1,N
      T(J,1)=T(J,2)
      T(J,2)=T(J,3)
      T(J,3)=0.0
      VTEMP=ABSF(T(J,1))
      IF (J-N) 785, 770, 785
770 IF (VTEMP) 820, 780, 820
780 T(J,1)=EPS1
      GO TO 820
785 INDEX(J)=0
      IF (ABSF(T(J+1,1))-VTEMP) 810, 810, 790
790 INDEX(J)=1
      DO 800 K=1,3
      VTEMP=T(J,K)
      T(J,K)=T(J+1,K)
      T(J+1,K)=VTEMP
800 CONTINUE
810 VTEMP =T(J+1,1)/T(J,1)
      FACTOR(J)=VTEMP
      T(J+1,2)=T(J+1,2)- VTEMP *T(J,2)
      T(J+1,3)=T(J+1,3)- VTEMP *T(J,3)
820 CONTINUE
      ITER=1
      IF (I1) 925, 860, 925
C
C      BACK SUBSTITUTE TO OBTAIN EIGENVECTOR
C
860 DO 870 L1=1,N
      L=N+1-L1
      V( L )=(V( L )-T(L,2)*V( L+1 )-T(L,3)*V( L+2 ))/T(L,1)
870 CONTINUE
      GO TO (875,920), ITER

```

```

C
C      PERFORM SECOND ITERATION
C
875 ITER=2
   IF (N-1) 945, 945, 880
880 DO 910 L=2,N
   IF (INDEX(L-1)) 890, 900, 890
890 VTEMP=V(L-1)
   V(L-1)=V(L)
   V(L)=VTEMP
900 V(L)=V(L)-FACTOR(L-1)*V(L-1)
910 CONTINUE
   GO TO 860

C
C      ORTHOGONALIZE VECTOR TO OTHERS ASSOCIATED WITH REPEATED VALUE
C
920 IF (I1) 945, 945, 925
925 DO 940 L1=1,I1
   K=I-L1
   VTEMP=0.0
   DO 930 J=1,N
930 VTEMP=VTEMP+A(J,K)*V(J)
   DO 940 J=1,N
940 V(J)=V(J)-A(J,K)*VTEMP
   GO TO (880,945), ITER

C
C      NORMALIZE VECTOR
C
945 VNORM2=0.0
   DO 950 L=1,N
950 VNORM2=VNORM2+V( L )**2
   VNORM=SQRTF(VNORM2)
   DO 960 J=1,N
960 A(J,I)=V( J )/VNORM

C
970 CONTINUE

C
C      ROTATION OF CO-DIAGONAL VECTORS INTO MATRIX EIGENVECTORS
C
C      -----
N=NSUB
   IF (N-1) 1000, 1000, 975
975 DO 990 I=1,M
   DO 980 K=2,N
   U(K)=0.0
   DO 980 J=2,N
980 U(K)=U(K)+B(J,K)*A(J,I)
   DO 990 J=2,N
990 A(J,I)=U(J)

C
1000 SRNORM=ANORM
   RETURN
   END

```

ACKNOWLEDGMENTS

I would like to thank Dr. J. Wallace Givens, Jr., Director of the Applied Mathematics Division, Argonne National Laboratory, for his hospitality during my visit to Argonne to write this report.

REFERENCES

1. Mueller, D. J., *Householder's Method for Complex Matrices and Eigenvalues of Hermitian Matrices*, Numerische Mathematik 8, 72-92 (1966).
2. Wilkinson, J. H., *Calculation of the Eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection*, Numerische Mathematik 4, 362-367 (1962).
3. *Calculation of the Eigenvectors of a Symmetric Tridiagonal Matrix by Inverse Iteration*, Numerische Mathematik 4, 368-376 (1962).
4. Francis, J. G. F., *The QR Transformation, Part 2*, The Computer Journal 4, 332-345 (1961/62).
5. Wilkinson, J. H., *The Algebraic Eigenvalue Problem*, The Oxford University Press (1965).
6. Eberlein, P. J., *A Jacobi-like Method for the Automatic Computation of Eigenvalues and Eigenvectors of an Arbitrary Matrix*, J. Soc. Indus. Appl. Math. 10, 74-88 (1962).
7. Wilkinson, J. H., *Householder's Method for the Solution of the Algebraic Eigenproblem*, The Computer Journal 3, 23-27 (1960).
8. Givens, W., *Numerical Computation of the Characteristic Values of a Real Symmetric Matrix*, Oak Ridge National Laboratory Report No. 1574 (1954).
9. Wilkinson, J. H., *The Calculation of the Eigenvectors of Codiagonal Matrices*, The Computer Journal 1, 90-96 (1958).