The University of Connecticut
SCHOOL OF ENGINEERING
Storrs, Connecticut 06268

Department of Electrical Engineering

(NASA-CR-131654) AN ALGORITHM FOR CALCULATION OF THE JORDAN CANONICAL FORM OF A MATRIX (Connecticut Univ.) 24 p HC $3.25 CSCL 12A

N73-22557

Unclas
G3/19 17667

AN ALGORITHM FOR CALCULATION OF THE JORDAN CANONICAL FORM OF A MATRIX

B. Sridhar
D. Jordan

Technical Report TR-73-1

March 1973

This work has been sponsored in part by the National Aeronautics and Space Administration Research Grant NGL-07-002-002

AN ALGORITHM FOR CALCULATION OF THE

JORDAN CANONICAL FORM OF A MATRIX

B. Sridhar
and
D. Jordan

Electrical Engineering Department
University of Connecticut
Storrs, Connecticut 06268

ABSTRACT

Jordan canonical forms are used extensively in the literature on control systems. However, very few methods are available to compute them numerically. Most numerical methods compute a set of basis vectors in terms of which the given matrix is diagonalized when such a change of basis is possible. Here, a simple and efficient method is suggested for computing the Jordan canonical form and the corresponding transformation matrix. The method is based on the definition of a generalized eigenvector, and a natural extension of Gauss elimination techniques.

This work has been sponsored in part by the National Aeronautics and Space Administration Research Grant NGL-07-002-002.

II

INTRODUCTION

It is well-known that any matrix may be brought into the Jordan canonical form by a similarity transformation [1]. There are several methods available to compute the eigenvectors of a matrix when the eigenvalues are distinct [2-3]. Some of these could be used to compute the eigenvectors for matrices with multiple roots. In Varah's method [4] multiple eigenvalues are handled by perturbing the multiple eigenvalue to produce distinct eigenvalues. Eberlin and Boothroyd [5] also compute eigenvectors for multiple eigenvalues. However, none of these methods generate the basis vectors necessary to transform the given matrix into it's Jordan canonical form. Chen [6] has suggested a procedure for computing the Jordan canonical form. Here, a simple and efficient algorithm, based on the notion of a generalized eigenvector, and using Gauss elimination techniques is given to compute the Jordan form of an nxn matrix.

BACKGROUND

Given the nxn matrix A, we want to find the matrix T such that $T^{-1}AT$ is a Jordan matrix J. Let $(\lambda_1, \lambda_2, ---, \lambda_m)$ be the eigenvalues of A with multiplicity $(n_1, n_2, ---, n_m)$ respectively. The number of eigenvectors associated with the eigenvalue λ_i is given by α_i = n-Rank $(A-\lambda_i I)$. The Jordan matrix, J, has the form

$$J = \text{diag}\,[J_{11}, J_{12}, ---, J_{1\alpha_1} \vdots J_{21}, J_{22}, ---, J_{2\alpha_2} \vdots ---$$

$$\vdots J_{m1}, J_{m2}, ---, J_{m\alpha_m}] \qquad (1)$$

with
$$J_{ik} = \begin{bmatrix} \lambda_i & 1 & & & \\ & \lambda_i & \cdot & & \bigcirc \\ & & \cdot & \cdot & \\ & & & \cdot & 1 \\ & \bigcirc & & & \lambda_i \end{bmatrix} \qquad \begin{aligned} i &= 1,2,---,m \\ k &= 1,\ 2,\ ---,\ \alpha_i \end{aligned} \tag{2}$$

Let β_{ik} be the dimension of the block J_{ik} and define

$$\sigma_{ik} = \sum_{\ell=1}^{i-1} \sum_{j=1}^{\alpha_\ell} \beta_{\ell j} + \sum_{j=1}^{k} \beta_{ij}. \qquad \text{with } \sigma_{10} = 0 \tag{3}$$

Let the generalized eigenvectors and the eigenvector corresponding to J_{ik} be $\underline{t}_{\sigma_{i(k-1)}+1}$, $\underline{t}_{\sigma_{i(k-1)}+2}$, ..., $\underline{t}_{\sigma_{i(k-1)}+\beta_{ik}-1}$ and $\underline{t}_{\sigma_{ik}}$ respectively. The transformation matrix T is made up of the n columns $(\underline{t}_1, \underline{t}_2, --- \underline{t}_{\sigma_{11}}, \underline{t}_{\sigma_{11}+1}, \ldots, \underline{t}_{\sigma_{12}}, \ldots, \underline{t}_{\sigma_{1(\alpha_1-1)}+1}, \ldots, \underline{t}_{\sigma_{1\alpha_1}}, \ldots, \underline{t}_{\sigma_{m\alpha_m}}$. The similarity transformation satisfies the relation

$$\begin{aligned} &AT = TJ \\ \text{i.e.} \quad &A[\underline{t}_1, \underline{t}_2, \ldots, \underline{t}_n] = [\underline{t}_1, \underline{t}_2, \ldots, \underline{t}_n]J \end{aligned} \tag{4}$$

Then, the eigenvectors of λ_r satisfy the relation

$$(A-\lambda_r I)\underline{t}_\ell = \underline{0} \qquad \ell = \sigma_{r1}, \sigma_{r2}, \ldots, \sigma_{r\alpha_r} \tag{5}$$

Given an eigenvector of λ_r the corresponding generalized eigenvectors satisfy the recursive relationship

$$\begin{aligned} (A-\lambda_r I)\ \underline{t}_{\ell-1} = \underline{t}_\ell \qquad &\ell = \sigma_{rk}, \sigma_{rk}-1, \ldots, \sigma_{rk}-\beta_{rk}+1 \\ &k = 1,\ 2,\ \ldots,\ \alpha_k. \end{aligned} \tag{6}$$

The solution of equations (5) and (6) yields the transformation matrix T.

COMPUTATION OF THE EIGENVECTORS:

Let $\bar{A} = (A - \lambda_r I)$. We can choose non-singular matrices P_r and Q_r such that $P_r \bar{A} Q_r = U_r$, where, U_r has the form

$$U_r = \left[\begin{array}{cc} U_{11} & A_{12} \\ \hline \multicolumn{2}{c}{0} \end{array}\right] \} \ \alpha_r \text{ rows}$$

Here U_{11} is an $(n-\alpha_r)x(n-\alpha_r)$ upper triangular matrix with $|U_{11}| \neq 0$ and A_{12} is an $(n-\alpha_r)x\alpha_r$ matrix. Given $(A-\lambda_r I)$, P_r, Q_r and U_r can be obtained by Gauss elimination with full pivoting [7]. The α_r eigenvectors corresponding to the eigenvalue λ_r are obtained by solving the equation

$$U_r \underline{t}_\ell = \underline{0} \tag{7}$$

using a back substitution scheme employing α_r independent selections of the last α_r components of $\underline{t}_\ell$. Full pivoting guarantees that this will result in α_r linearly independent solutions which become the α_r independent eigenvectors corresponding to λ_r. Substitution of these eigenvectors in equation (6) yields the set of generalized eigenvectors.

Algorithm:

1. Find the eigenvalues of A. Label them $\lambda_1, \lambda_2, \ldots, \lambda_m$.
2. Solve the equation $U_r \underline{t}_\ell = \underline{0}$ for all eigenvectors corresponding to λ_r using independent selection of undetermined constants. The solution involves undefined variables v_r, w_r, ... Generate an independent set of eigenvectors for λ_r by setting each undefined variable in turn equal to 1 while holding all other variables equal to 0. Denote the eigenvectors by $\underline{t}_{\sigma_{r1}}$, $\underline{t}_{\sigma_{r2}}$, ..., $\underline{t}_{\sigma_{r\alpha_r}}$.

3. For each eigenvector $\underline{t}_{\sigma_{ri}}$, $i = 1, 2, \ldots, \alpha_r$ form $P_r Q_r \underline{t}_{\sigma_{ri}}$ and solve

$$U_r \underline{t}_{\sigma_{ri}-1} = P_r Q_r \underline{t}_{\sigma_{ri}}$$

for generalized eigenvector corresponding to eigenvector $\underline{t}_{\sigma_{ri}}$ with the undetermined constants taking values given to them while evaluating $\underline{t}_{\sigma_{ri}}$.

4. Repeat step 3 by forming $P_r Q_r \underline{t}_{\sigma_{ri}-1}$ and solve $U_r \underline{t}_{\sigma_{ri}-2} = P_r Q_r \underline{t}_{\sigma_{ri}-1}$.

5. Continue to generate generalized eigenvectors as in step 4 until the equation $U_r \underline{t}_{\sigma_{ri}-j-1} = P_r Q_r \underline{t}_{\sigma_{ri}-j}$ becomes inconsistent i.e. when a non-zero quantity appears on the right hand side corresponding to zero rows of U_r. This gives the basis vectors corresponding to the eigenvalue λ_r.

6. Repeat step 2 thru 5 for $r = 1, 2, \ldots, m$. to obtain all the basis vectors and hence the matrix T.

7. Obtain the Jordan canonical form from $J = T^{-1}AT$. Note that J need not be calculated directly since the block structure of (1) is determined by the number of generalized eigenvectors that are generated for each eigenvector.

Computational Discussion: The computation of the eigenvectors and the generalized eigenvectors depend on the accuracy with which the eigenvalues of A are computed. Francis' [8] algorithm is suggested for computing the eigenvalues. When the eigenvalues are approximate the calculation of the eigenvector can be refined as suggested by Wilkinson [9].

The algorithm suggested in this paper results in a large reduction in the amount of computation necessary to obtain the Jordan canonical form. The number of computations necessary for an n^{th} order system with m distinct eigenvalues is shown in Table 1.

TABLE 1

STEP	NUMBER OF COMPUTATIONS
$P_i(A-\lambda_i I)Q$	$\sum_{i=1}^{n-1} i^2 + \sum_{i=1}^{n} i = \frac{1}{3}(n^3-n)$
Total elimination for m eigenvalues	$m(n^3-n)/3$
Back substitution	$\leq \sum_{i=1}^{n-1} i = \frac{n^2-n}{2}$
Total for n back substitution	$\leq \frac{n^3-n^2}{2}$
To construct a right hand side $(P_i Q_i \underline{x})$	$\sum_{i=1}^{n} i = \frac{n^2-n}{2}$
Total R.H.S.	$n(n^2-n)/2 = \frac{n^3-n^2}{2}$
Total	$\frac{mn^3}{3} - \frac{mn}{3} + n^3-n^2 = 0(\frac{m+1}{3} n^3)$

A similar analysis of Chen's algorithm [6] shows that the number of computations are of the order $0(\frac{5}{3}n^4)$. Thus the algorithm suggested here results in at least a fivefold saving in the number of computations. The method does not require the evaluation of the rank of matrices of powers of $(A-\lambda_r I)$ as in Chen's method.

Examples:

The algorithm is applied to find the eigenvectors and the Jordan canonical form of two different matrices.

A. Fourth order matrix:

$$\begin{bmatrix} 6 & -3 & 4 & 1 \\ 4 & 2 & 4 & 0 \\ 4 & -2 & 3 & 1 \\ 4 & 2 & 3 & 1 \end{bmatrix}$$

This matrix is taken from Eberlin and Boothroyd [5]. The eigenvalues of the matrix are 5.23606797749979 (double root) and 0.763932022500210 (double root).

The eigenvector and the generalized eigenvector associated with the double root 5.23606797749979 are

$$\begin{bmatrix} 0.4270509831 \\ 1.0000000000 \\ 0.3819660113 \\ 1.1458980340 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0.5868810394 \\ 1.0000000000 \\ 0.4721359550 \\ 1.0901699410 \end{bmatrix} \quad \text{respectively.}$$

For the double root 0.763932022500210 the corresponding vectors are given by

$$\begin{bmatrix} -0.3726779962 \\ 0.1273220038 \\ 0.3333333333 \\ 1.0000000000 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0.2197175016 \\ 0.4182146692 \\ -0.3171224407 \\ 1.0000000000 \end{bmatrix}$$

Notice that the two eigenvectors and the two generalized eigenvectors are all independent unlike in [5]. The Jordan canonical form can be readily written as

$$\left[\begin{array}{cc|cc} 5.2360 & 1 & & \\ 0 & 5.2360 & & \bigcirc \\ \hline & & 0.7639 & 1 \\ & \bigcirc & 0 & 0.7639 \end{array}\right]$$

The execution time was 1.57 secs with a WATFIV (Univ. of Waterloo - Fast Fortran) compiler.

B. System matrix of Boeing Helicopter

The following 8x8 matrix arises in the design of a helicopter stabilization system using Pole-placement theory [10].

$$\begin{bmatrix} .021 & .025 & -29.64 & .6968 & .1879 & 0 & -.0941 & 0 \\ -.0903 & -.802 & -80.98 & -1.878 & .5524 & 0 & -8.517 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -.0058 & .0145 & 1.4672 & -1.460 & .45 & 0 & .068 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -784 & -35 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -784 & -35 \end{bmatrix}$$

The eigenvalues of the system computed by using Francis' method are 0.50432908, -2.3585084, $-0.19350035 \pm j\ 0.35283477$ and $-17.5 \pm j\ 21.857493$ (double root). The eigenvectors corresponding to the distinct roots are

$$\begin{bmatrix} 1.0000000000 \\ 0.9167473189 \\ -0.0157197678 \\ -0.0079269851 \\ 0.0000000000 \\ 0.0000000000 \\ 0.0000000000 \\ 0.0000000000 \end{bmatrix}, \begin{bmatrix} 0.2528902161 \\ 1.0000000000 \\ 0.0200347219 \\ -0.0472520599 \\ 0.0000000000 \\ 0.0000000000 \\ 0.0000000000 \\ 0.0000000000 \end{bmatrix}, \begin{bmatrix} -0.0949009676 \mp j\ 0.6460398691 \\ 1.0000000000 + j\ 0.0000000000 \\ -0.0074706563 \mp j\ 0.0035914411 \\ 0.0026856551 \mp j\ 0.0019501968 \\ 0.0000000000 + j\ 0.0000000000 \\ 0.0000000000 + j\ 0.0000000000 \\ 0.0000000000 + j\ 0.0000000000 \\ 0.0000000000 + j\ 0.0000000000 \end{bmatrix}$$

respectively. Each of the double roots has two eigenvectors associated with it. These are

$$\begin{bmatrix} -0.0000183498 \pm j\ 0.0002379966 \\ -0.0001564383 \pm j\ 0.0007421192 \\ 0.0000193897 \mp j\ 0.0000084539 \\ -0.0001545381 \pm j\ 0.0005715717 \\ -0.0223214285 \mp j\ 0.0278794553 \\ 1.0000000000 + j\ 0.0000000000 \\ 0.0000000000 + j\ 0.0000000000 \\ 0.0000000000 + j\ 0.0000000000 \end{bmatrix} \text{ and } \begin{bmatrix} 0.0000224177 \mp j\ 0.0001119734 \\ 0.0026667158 \mp j\ 0.0107258554 \\ 0.0000031152 \mp j\ 0.0000011743 \\ -0.0000288496 \pm j\ 0.0000886422 \\ 0.0000000000 + j\ 0.0000000000 \\ 0.0000000000 + j\ 0.0000000000 \\ -0.0223214285 \mp j\ 0.0278794553 \\ 1.0000000000 + j\ 0.0000000000 \end{bmatrix} .$$

Since the multiple eigenvalues have as many eigenvectors as their multiplicity, the Jordan canonical form for this matrix is diagonal and is given by

$$\text{diag } [.50432908,\ -2.3585084,\ -0.19350035 + j\ 0.35283477,\ -0.19350035 - j\ 0.35283477,\ -17.5 + j\ 21.857493,\ -17.5 + j\ 21.857493,\ -17.5 - j\ 21.857493,\ -17.5 - j\ 21.857493]$$

The execution time using a WATFIV compiler was 8.69 secs.

Flowchart and Computer Program:

These are given in Appendix A and Appendix B, respectively.

Conclusion:

A method has been outlined to find the basis vectors to transform a given nxn matrix to its Jordan canonical form. The method is simple and efficient. It does not require the evaluation of the rank of matrices of powers of $(A-\lambda_i I)$ as in Chen's method [6]. There is at least a fivefold reduction in the number of computations. Two examples are given to illustrate the method.

References

1. Turnbull, H. W. and Aitken, A. C. An Introduction to the Theory of Canonical Matrices, Blackie and Son, London, 1948.

2. Wilkinson, J. H., The Algebraic Eigenvalue Problem, London, Oxford University Press, 1965.

3. Wilkinson, J. H., and Reinch, C., Linear Algebra, Springer-Verlag, New York, 1971.

4. Varah, J., Ph. D. Thesis, Stanford University (1967).

5. Eberlin, P. J., and Boothroyd, J., "Solution to the eigenproblem by a norm reducing Jacobi Type method," Numer. Math. 11, 1-12 (1968).

6. Chen, C. T., Introduction To Linear System Theory, Holt, Rinehart & Winston, Inc., New York, 1970.

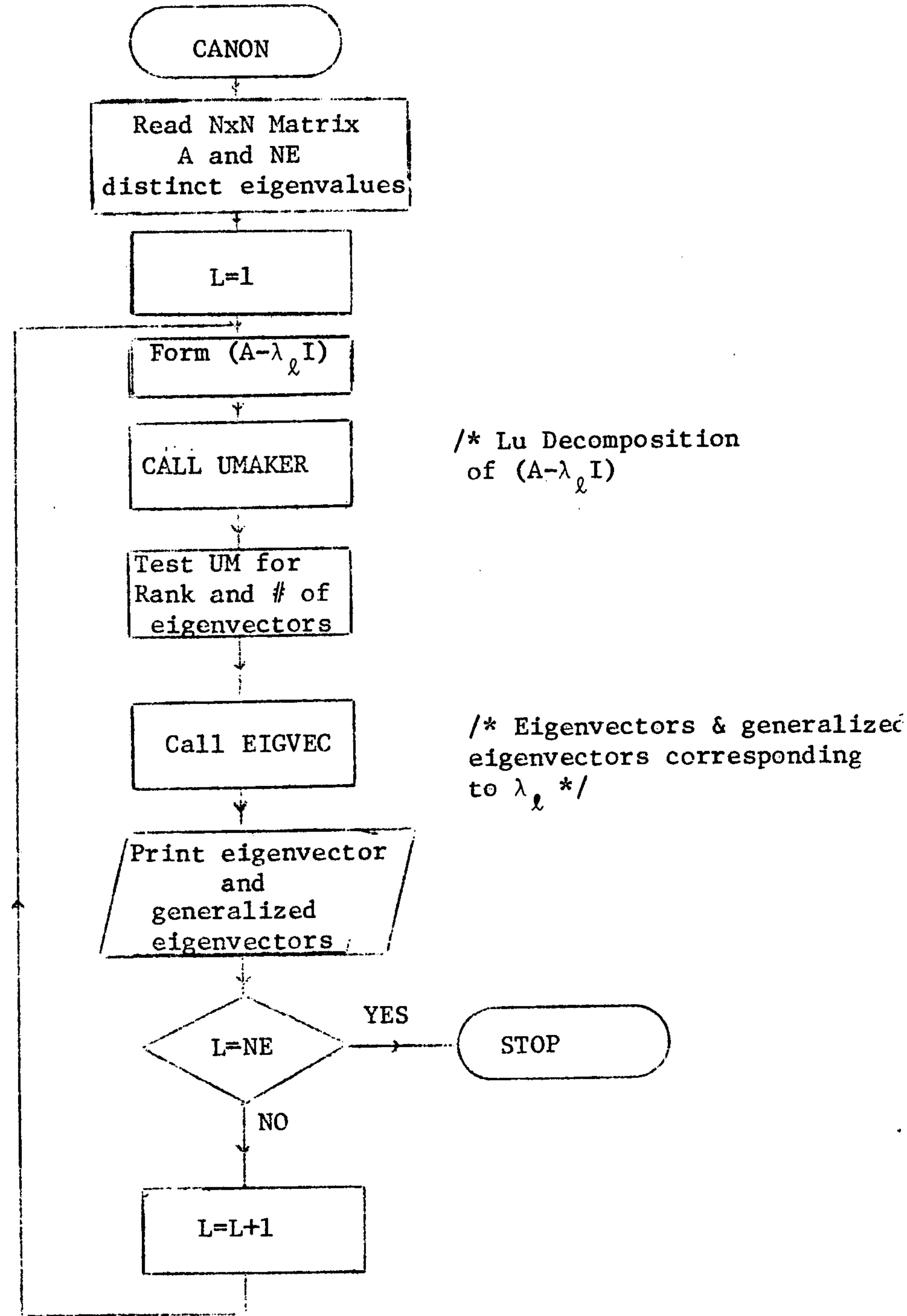
7. Forsythe, G., and Moler, C. B., Computer Solution of Linear Algebraic Systems, Prentice-Hall, Inc. 1967.

8. Francis, J.A.F., The QR Transformation, Part I and II, Computer Journal #4, 265-271, 332-345, 1962.

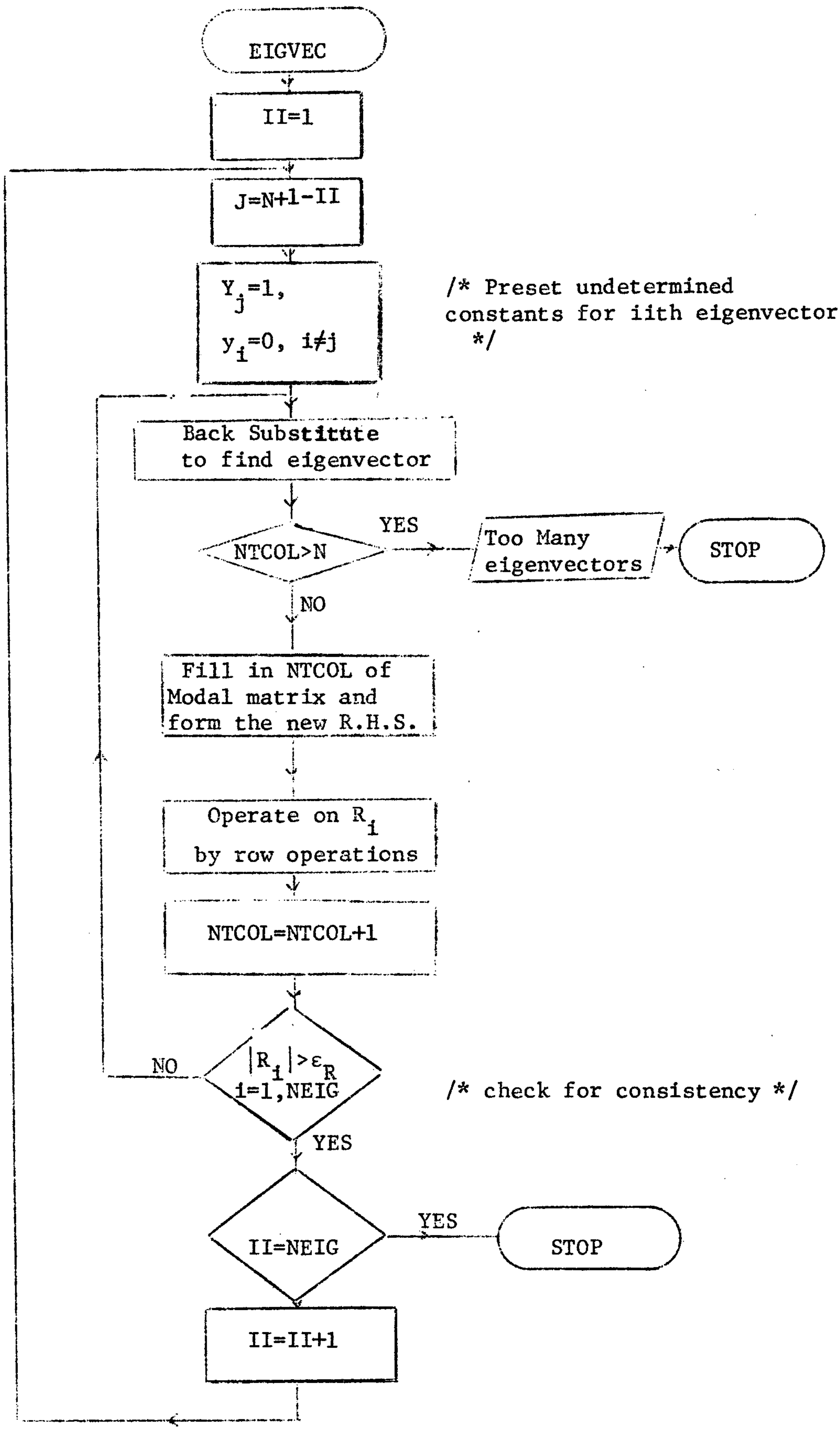
9. Wilkinson, J. H., "The Calculation of Eigenvectors of Codiagonal Matrices", Computer Journal #1, 148-152, 1958.

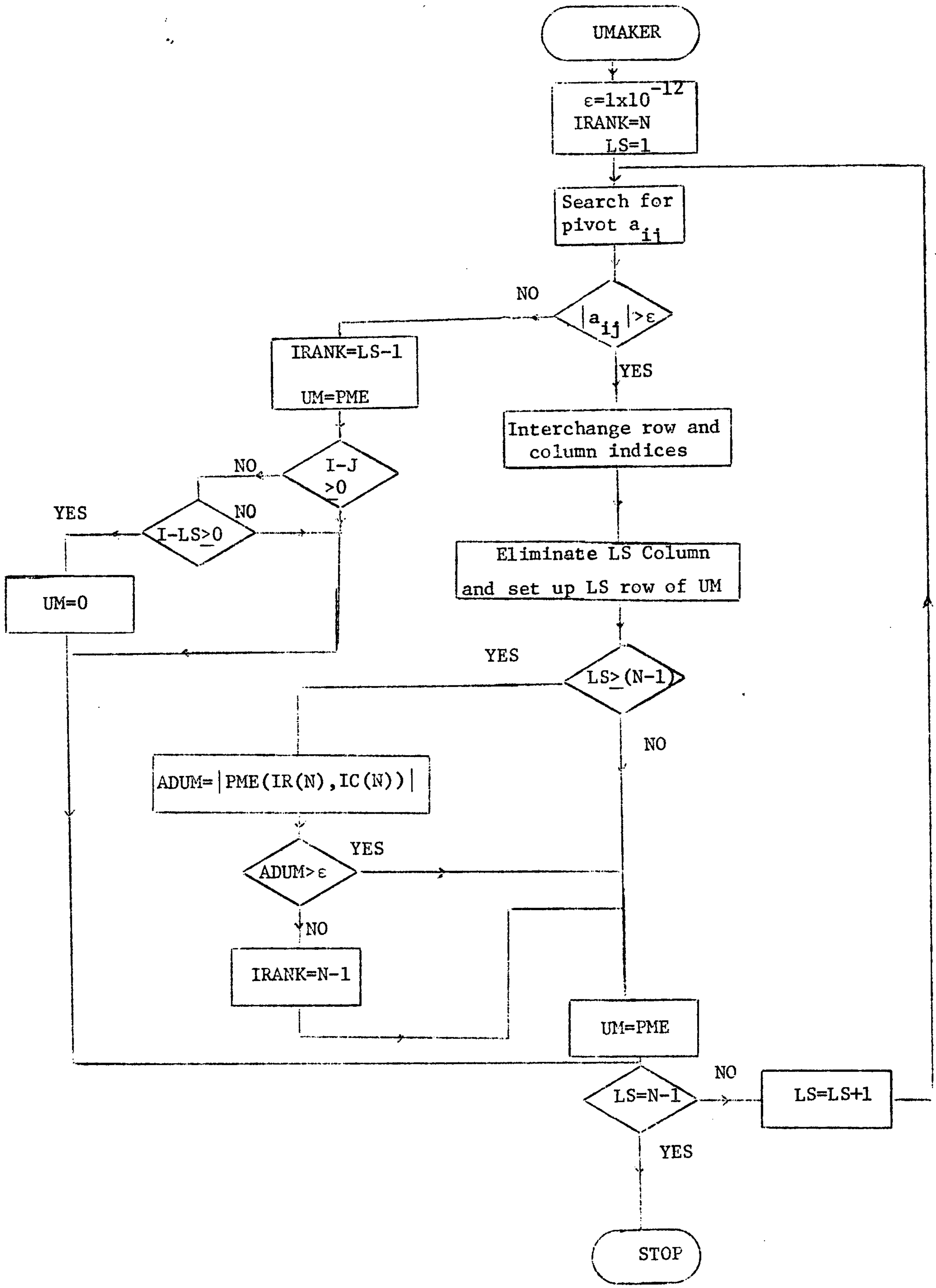
10. Sridhar,B,,and Lindorff,D.P.,"Application of Pole-placement Theory to Helicopter Stabilization Systems,"Sixth Hawaii International Conference on System Sciences, January 1973.

APPENDIX A



Main Program


EIGVEC
II=1
J=N+1-II
$Y_j=1$,
$y_i=0,\ i\neq j$
/* Preset undetermined constants for iith eigenvector */
Back Substitute to find eigenvector
NTCOL>N
YES
Too Many eigenvectors
STOP
NO
Fill in NTCOL of Modal matrix and form the new R.H.S.
Operate on R_i by row operations
NTCOL=NTCOL+1
$|R_i|>\varepsilon_R$ $i=1,NEIG$
/* check for consistency */
NO
YES
II=NEIG
YES
STOP
II=II+1


UMAKER
ε=1x10^-12
IRANK=N
LS=1
Search for pivot a_ij
|a_ij|>ε
NO
YES
IRANK=LS-1
UM=PME
Interchange row and column indices
I-J >0
NO
I-LS≥0
YES
NO
UM=0
Eliminate LS Column and set up LS row of UM
LS≥(N-1)
YES
NO
ADUM=|PME(IR(N),IC(N))|
ADUM>ε
YES
NO
IRANK=N-1
UM=PME
LS=N-1
NO
LS=LS+1
YES
STOP

APPENDIX B

```
C          MAIN PROGRAM
C
           IMPLICIT REAL *8(A-H,O-Y)
           COMMON AM,PMR,PMI,UMR,UMI,TMR,TMI,PEIGR,PEIGI,PMER,PMEI,YR,YI,
          1EPSA,EPR,IR,IC,IEIG,NTCOL,LL,NE,N,NEIG,IRANK,IOPT
           DIMENSION AM(12,12),PMR(12,12),PMI(12,12),UMR(12,12),
          1UMI(12,12),TMR(12,12),TMI(12,12),PEIGR(12),PEIGI(12),PMER(12,12),
          2PMEI(12,12),YR(12),YI(12),IR(12),IC(12),IEIG(15,16)
C
C          THIS ROUTINE IS DESIGNED TO FIND ALL THE EIGENVECTORS AND
C          GENERALIZED EIGENVECTORS OF AN N*N REAL MATRIX GIVEN THE SET
C          OF DISTINCT EIGENVALUES.  THE PRINTOUT INDICATES THE
C          APPROPRIATE JORDAN CANONICAL FORM
C
C          VARIABLES
C
C          ALL VARIABLES FROM A TO H ARE DOUBLE PRECISION
C          ALL VARIABLES FROM O TO Y ARE COMPLEX EITH REAL PART
C          ENDING IN R AND IMAGINARY PART ENDING IN I
C
C          AM         ORIGINAL MATRIX
C          PEIG,R+I   EIGENVALUES
C          PM,R+I     (A-LAMBDA*I) MATRIX
C          UM,R+I     DECOMPOSED MATRIX, U ABOVE DIAGONAL, M BELOW
C          TM,R+I     MATRIX OF GENERALIZED EIGENVECTORS--MODAL MATRIX
C
C          N          DIMENSION OF AM
C          NE         NUMBER OF EIGENVALUES
C          IC         COLUMN INTERCHANGE INDEX VECTOR
C          IR         ROW INTERCHANGE INDEX VECTOR
C          IOPT       OPTION(=1) FOR INTRMEDIATE PRINTOUTS
C          IEND       OPTION(=1) FOR ADDITIONAL PROBLEM TO FOLLOW
C
C
C
      4000 FORMAT(1H1)
      4001 FORMAT(8I10)
      4002 FORMAT(4D20.10)
      4003 FORMAT(/////)
      4004 FORMAT(//)
      4005 FORMAT(2D30.15)
      4010 FORMAT(//,5X,'MATRIX DIMENSION = ',I3,
          1' NUMBER OF DISTINCT EIGENVALUES = 'I3,/)
      4011 FORMAT(5X,'A MATRIX',/)
      4012 FORMAT(5X,'DISTINCT EIGENVALUES',//,5X,9HREAL PART,9X,
          114HIMAGINARY PART,/)
      4013 FORMAT(5X,'ROW INTERCHANGE INDEX',/)
      4014 FORMAT(5X,'COLUMN INTERCHANGE INDEX',/)
      4015 FORMAT(5X,'DECOMPOSED MATRIX',/)
      4016 FORMAT(5X,'NUMBER OF EIGENVECTORS CORRESPONDING TO EIG ',I2,
          1' IS ',I3,/)
      4017 FORMAT(5X,'BLOCK NUMBER ',I2,' HAS ',I2,
          1' GENERALIZED EIGENVECTORS',/,5X,'THE FIRST IS THE EIGENVECTOR',/)
      4018 FORMAT(5X,9HREAL PART,9X,14HIMAGINARY PART,/)
C
C          INPUT A MATRIX AND EIGENVALUES
C
        10 READ 4001,N,NE,IOPT,IEND
           PRINT 4000
```

```
      PRINT 4003
      PRINT 4010,N,NE
      READ 4002,((AM(I,J),J=1,N),I=1,N)
      PRINT 4011
      PRINT 4002,((AM(I,J),J=1,N),I=1,N)
      DO 50 I=1,N
      DO 50 J=1,N
      TMR(I,J)=0.0D0
      TMI(I,J)=0.0D0
      PMR(I,J)=AM(I,J)
   50 PMI(I,J)=0.0D0
      PRINT 4004
      PRINT 4012
      DO 60 I=1,NE
      READ 4005,A,B
      PRINT 4005,A,B
      PEIGR(I)=A
   60 PEIGI(I)=B
C
C         START COMPUTING
C
      PRINT 4000
      PRINT 4003
      NTCOL=1
      DO 500 L=1,NE
      LL=L
C
C         FORM (A-LAMBDA*I)
C
      DO 100 I=1,N
      DO 100 J=1,N
      PMER(I,J)=PMR(I,J)
      PMEI(I,J)=PMI(I,J)
      IF (I-J)100,95,100
   95 PMER(I,J)=PMER(I,J)-PEIGR(L)
      PMEI(I,J)=PMEI(I,J)-PEIGI(L)
  100 CONTINUE
C
      CALL UMAKER
      IF(IOPT)107,107,105
C
C         IF IOPT=1 PRINTOUT DECOMPOSED MATRIX
C
  105 PRINT 4003
      PRINT 4013
      PRINT 4001,(IR(I),I=1,N)
      PRINT 4014
      PRINT 4001,(IC(I),I=1,N)
      PRINT 4015
      PRINT 4002,((UMR(I,J),J=1,N),I=1,N)
      PRINT 4004
      PRINT 4002,((UMI(I,J),J=1,N),I=1,N)
C
C         TEST UM FOR RANK AND NUMBER OF EIGENVECTORS
C
  107 CONTINUE
      CALL CABS(EPSA,UMR(N,N),UMI(N,N))
      EPSA=EPSA*100.0D0
      IF(EPSA-1.0D-12)110,110,112
  110 EPR=1.0D-12
```

```
      GO TO 115
  112 EPR=EPSA
  115 CONTINUE
      NEIG=1
      NM=N-1
      DO 125 I=1,NM
      IA=N-I
      CALL CABS(ATEST,UMR(IA,IA),UMI(IA,IA))
      IF(ATEST-EPR)120,120,118
  118 GO TO 130
  120 NEIG=NEIG+1
  125 CONTINUE
  130 CONTINUE
      IEIG(L,1)=NEIG
      PRINT 4016,L,NEIG
C
      CALL EIGVEC
C
C         PRINTOUT RESULTS
C
      ICT=1
      DO 135 KKK=1,L
      IF(KKK-L)132,135,135
  132 KB=IEIG(KKK,1)+1
      DO 133 KC=2,KB
  133 ICT=ICT+IEIG(KKK,KC)
  135 CONTINUE
      DO 150 J=1,NEIG
      JJ=IEIG(L,J+1)
      PRINT 4004
      PRINT 4017,J,JJ
      PRINT 4018
      DO 150 K=1,JJ
      PRINT 4004
      DO 140 KK=1,N
      PRINT 4002,TMR(KK,ICT),TMI(KK,ICT)
  140 CONTINUE
      ICT=ICT+1
  150 CONTINUE
C
  500 CONTINUE
C
C         TERMINATION
C
      IF(IEND)510,510,10
  510 CONTINUE
      PRINT 4000
      STOP
      END

      SUBROUTINE UMAKER
      IMPLICIT REAL *8(A-H,O-Y)
      COMMON AM,PMR,PMI,UMR,UMI,TMR,TMI,PEIGR,PEIGI,PMER,PMEI,YR,YI,
     1EPSA,EPR,IR,IC,IEIG,NTCOL,LL,NE,N,NEIG,IRANK,IOPT
      DIMENSION AM(12,12),PMR(12,12),PMI(12,12),UMR(12,12),
     1UMI(12,12),TMR(12,12),TMI(12,12),PEIGR(12),PEIGI(12),PMER(12,12),
     2PMEI(12,12),YR(12),YI(12),IR(12),IC(12),IEIG(15,16)
C
C        THIS SUBROUTINE CALCULATES THE LU DECOMPOSITION OF
C        PME,R+I WITH FULL PIVOTING
```

```
C
C          INPUT VARIABLES
C          PME,R+I     MATRIX TO BE DECOMPOSED
C          N           DIMENSION
C
C          OUTPUT VARIABLES
C          UM,R+I      DECOMPOSED MATRIX, U UPPER TRIANGLE INCLUDING
C                      DIAGONAL, MULTIPLIERS BELOW DIAGONAL
C          IR          ROW INTERCHANGE INDEX VECTOR
C          IC          COLUMN INTERCHANGE INDEX VECTOR
C
C
C          PRESET ROW AND COLUMN INTERCHANGES
C
      DO 100 I=1,N
      IR(I)=I
  100 IC(I)=I
      EPS=1.0D-20
      IRANK=N
      NN=N-1
C
C          BEGIN ELIMINATION PROCEDURE
C
      DO 200 LS=1,NN
      LSS=LS
      AMAG=0.0D0
C
C          SEARCH FOR PIVOT
C
      DO 120 I=LS,N
      DO 120 J=LS,N
      CALL CABS(ADUM,PMER(IR(I),IC(J)),PMEI(IR(I),IC(J)))
      IF(ADUM-AMAG)120,120,105
  105 IS=I
      JS=J
      AMAG=ADUM
  120 CONTINUE
C
C          TEST FOR COMPLETION
C
      IF(AMAG-EPS)125,125,130
  125 IRANK=LS-1
      GO TO 300
  130 CONTINUE
C
C          INTERCHANGING ROW AND COLUMN INDICES
C
      IT=IR(LS)
      IR(LS)=IR(IS)
      IR(IS)=IT
      IT=IC(LS)
      IC(LS)=IC(JS)
      IC(JS)=IT
C
C          ELIMINATE IC(LS) COLUMN AND SET UP UM,R+I LS ROW
C
      LSP=LS+1
      DO 150 I=LSP,N
      CALL CDIV(QR,CI,PMER(IR(I),IC(LS)),PMEI(IR(I),IC(LS)),
     1PMER(IR(LS),IC(LS)),PMEI(IR(LS),IC(LS)))
```

```
      UMR(I,LS)=CR
      UMI(I,LS)=QI
      DO 150 J=LSP,N
      CALL CMUL(CTR,CTI,CR,QI,PMER(IR(LS),IC(J)),PMEI(IR(LS),IC(J)))
      PMER(IR(I),IC(J))=PMER(IR(I),IC(J))-QTR
      PMEI(IR(I),IC(J))=PMEI(IR(I),IC(J))-QTI
  150 CONTINUE
C
C        PATCH UP RANK TEST
C
      IF(LS-NN)170,160,160
  160 CALL CABS(ADUM,PMER(IR(N),IC(N)),PMEI(IR(N),IC(N)))
      IF(ADUM-EPS)165,165,170
  165 IRANK=N-1
  170 CONTINUE
  200 CONTINUE
      GO TO 350
C
C        WINDUP PROCEDURE
C
  300 DO 310 I=1,N
      DO 310 J=I,N
      UMR(I,J)=PMER(IR(I),IC(J))
      UMI(I,J)=PMEI(IR(I),IC(J))
      IF(I-J)302,310,310
  302 IF(I-LS)310,304,304
  304 UMR(J,I)=0.0D0
      UMI(J,I)=0.0D0
  310 CONTINUE
      GO TO 400
  350 DO 360 I=1,N
      DO 360 J=I,N
      UMR(I,J)=PMER(IR(I),IC(J))
      UMI(I,J)=PMEI(IR(I),IC(J))
  360 CONTINUE
  400 CONTINUE
      RETURN
      END

      SUBROUTINE EIGVEC
      IMPLICIT REAL *8(A-H,O-Y)
      COMMON AM,PMR,PMI,UMR,UMI,TMR,TMI,PEIGR,PEIGI,PMER,PMEI,YR,YI,
     1EPSA,EPR,IR,IC,IEIG,NTCOL,LL,NE,N,NEIG,IRANK,IOPT
      DIMENSION AM(12,12),PMR(12,12),PMI(12,12),UMR(12,12),
     1UMI(12,12),TMR(12,12),TMI(12,12),PEIGR(12),PEIGI(12),PMER(12,12),
     2PMEI(12,12),YR(12),YI(12),IR(12),IC(12),IEIG(15,16)
      DIMENSION RR(12),RI(12),SA(12),SB(12),IRA(12)
C
C        THIS SUBROUTINE TAKES THE DECOMPOSED MATRIX OF UMAKER WITH
C        KNOWN RANK (N-NEIG) AND CALCULATES ALL THE EIGENVECTORS AND
C        GENERALIZED EIGENVECTORS OF THE CURRENT EIGENVALUE (PEIG(L))
C
C        INPUT VARIABLES
C        UM,R+I     DECOMPOSED MATRIX
C        N          DIMENSION
C        IR,IC      ROW AND COLUMN INTERCHANGE INDICES
C        NEIG       NUMBER OF EIGENVECTORS
C        NTCOL      CURRENT COLUMN OF TM
C
C        OUTPUT VARIABLES
```

```
C          TM,R+I     COLUMNS OF MODAL MATRIX - ALSO EIGENVECTORS
C                     AND GENERALIZED EIGENVECTORS
C          IEIG       NUMBERS OF EIGENVECTORS AND GENERALIZED EIFENVECTORS
C                     CORRESPONDING TO EACH EIGENVALUE
C
C
C          BEGIN SEARCH FOR EIGENVECTORS
C
      NOK=N-NEIG
      DO 200 II=1,NEIG
      NUM=1
      NI=N+1-II
C
C          PRESET UNDETERMINED CONSTANTS FOR II-TH EIGENVECTOR
C
      DO 50 J=1,N
      RR(J)=0.0D0
      RI(J)=0.0D0
      YR(J)=0.0D0
   50 YI(J)=0.0D0
      YR(NI)=1.0D0
C
C          BACK SUBSTITUTE TO FIND EIGENVECTOR
C
   60 CONTINUE
      DO 75 J=1,NOK
      JJ=NOK+1-J
      JK=J+NEIG-1
      DO 70 K=1,JK
      KK=N+1-K
      CALL CMUL(QTR,QTI,UMR(JJ,KK),UMI(JJ,KK),YR(KK),YI(KK))
      SA(K)=-QTR
      SB(K)=-QTI
   70 CONTINUE
      CALL SUM(JK,SA,SMR)
      CALL SUM(JK,SB,SMI)
      SMR=SMR+RR(JJ)
      SMI=SMI+RI(JJ)
      CALL CDIV(QTR,QTI,SMR,SMI,UMR(JJ,JJ),UMI(JJ,JJ))
      YR(JJ)=QTR
      YI(JJ)=QTI
   75 CONTINUE
C
C          FIND ALL GENERALIZED EIGENVECTORS
C
      NGE=1
   76 IF(NTCOL-N)79,79,77
   77 PRINT 4050
 4050 FORMAT(5X,'TOO MANY EIGENVECTORS FOUND',//)
      STOP
   79 DO 80 I=1,N
      TMR(IC(I),NTCOL)=YR(I)
   80 TMI(IC(I),NTCOL)=YI(I)
C
C          OPERATE ON RIGHT HAND SIDE BY ROW OPS
C
      DO 90 I=1,N
      IRA(I)=I
      RR(I)=TMR(I,NTCOL)
   90 RI(I)=TMI(I,NTCOL)
```

```
      NTCOL=NTCOL+1
      NM=N-1
      DO 120 I=1,NM
      IF(IRA(I)-IR(I))94,100,94
   94 DO 98 IS=I,N
      IF(IRA(IS)-IR(I))98,96,98
   96 IST=IS
   98 CONTINUE
      IT=IRA(I)
      IRA(I)=IRA(IST)
      IRA(IST)=IT
      RRT=RR(IST)
      RR(IST)=RR(I)
      RR(I)=RRT
      RIT=RI(IST)
      RI(IST)=RI(I)
      RI(I)=RIT
  100 CONTINUE
      IP=I+1
      DO 110 J=IP,N
      CALL CMUL(QTR,QTI,UMR(J,I),UMI(J,I),RR(I),RI(I))
      RR(J)=RR(J)-QTR
      RI(J)=RI(J)-QTI
  110 CONTINUE
  120 CONTINUE
C
C        CHECK FOR INCONSISTENCY
C
      IF(IOPT)127,127,125
  125 PRINT 4300
      PRINT 4002,(RR(J),J=1,N)
 4300 FORMAT(5X,'RIGHT HAND SIDE',/)
 4002 FORMAT(4D20.10)
  127 DO 130 J=1,NEIG
      JJ=N-J+1
      CALL CABS(ADUM,RR(JJ),RI(JJ))
      IF(ADUM-EPP)130,130,135
  130 CONTINUE
      NUM=NUM+1
      GO TO 140
  135 IIP=II+1
      IEIG(LL,IIP)=NUM
      GO TO 200
  140 CONTINUE
C
C        IF CONSISTENT THEN BACK SUBSTITITE FOR GENERALIZED EIGENVECTOR
C
      GO TO 60
  200 CONTINUE
      RETURN
      END

      SUBROUTINE CMUL(A,B,C1,C2,D1,D2)
      IMPLICIT REAL *8(A-H,O-Y)
      A=C1*D1-C2*D2
      B=C1*D2+C2*D1
      RETURN
      END

      SUBROUTINE CDIV(A,B,C1,C2,D1,D2)
```

```
      IMPLICIT REAL *8(A-H,O-Y)
      E=D1*D1+D2*D2
      IF(E-1.0D-40)50,100,100
   50 PRINT 4000
 4000 FORMAT(5X,'DIVICE CHECK IN CDIV -- DIVIDEND RETURNED',/)
      A=C1
      B=C2
      GO TO 110
  100 A=(C1*D1+C2*D2)/E
      B=(C2*D1-C1*D2)/E
  110 CONTINUE
      RETURN
      END

      SUBROUTINE CABS(A,C1,C2)
      IMPLICIT REAL *8(A-H,O-Y)
      A=DSQRT(C1*C1+C2*C2)
      RETURN
      END

      SUBROUTINE SUM(LENGTH,S,SM)
      IMPLICIT REAL *8(A-H,O-Y)
      DIMENSION WORKA(127),WORKAA(128),S(12)
      EQUIVALENCE(JZ,ZJ)
      EQUIVALENCE(WORKA(1),WORKAA(2))
      DBLZRO=0.0D0
      SUMN=DBLZRO
C
C          ZERO OUT THE ACCUMULATING ARRAY
C
 1000 DO 1010 L=1,128
 1010 WORKAA(L)=DBLZRO
C
C          DO JOHN'S ALGORITHM
C
      DO 1050 I=1,LENGTH
      WORK=S(I)
      LASTJZ=-1
      ZJ=WORK
      IF(ZJ)1012,1050,1013
 1012 ZJ=-ZJ
 1013 JZ=JZ/16777216
      SUMN=WORKA(JZ)+WORK
      WORKA(JZ)=DBLZRO
      GO TO 1015
 1014 SUMN=SUMN+WORKA(JZ)
      WORKA(JZ)=DBLZRO
 1015 ZJ=SUMN
      IF(ZJ)1016,1050,1017
 1016 ZJ=-ZJ
 1017 JZ=JZ/16777216
      IF(JZ-LASTJZ)1020,1030,1020
 1020 LASTJZ=JZ
      GO TO 1014
 1030 WORKA(JZ)=SUMN
 1050 CONTINUE
      SUMN=DBLZRO
      DO 1060 L=1,128
 1060 SUMN=SUMN+WORKAA(L)
  999 SM=SUMN
      RETURN
      END
```