A PROGRAM TO COMPUTE THE CONDITION NUMBERS OF MATRIX EIGENVALUES WITHOUT COMPUTING EIGENVECTORS

by

S. P. Chan, R. Feldman and B. N. Parlett

Memorandum No. ERL-M517

April 1975

A PROGRAM TO COMPUTE THE CONDITION NUMBERS OF MATRIX EIGENVALUES WITHOUT COMPUTING EIGENVECTORS

by

S. P. Chan, R. Feldman and B. N. Parlett

Memorandum No. ERL-M517

April 1975

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A PROGRAM TO COMPUTE THE CONDITION NUMBERS OF MATRIX EIGENVALUES WITHOUT COMPUTING EIGENVECTORS[†]

by

S.P. Chan
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California
Berkeley, California 94720

R. Feldman
Department of Mathematics
University of California
Berkeley, California 94720

and

B.N. Parlett
Department of Mathematics
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California
Berkeley, California 94720

April 1975

Abstract

The condition number of an eigenvalue measures the sensitivity of that eigenvalue to small changes in the matrix elements. Such extra information is nice, sometimes useful, but how much does it cost?

A program is presented here for the most difficult case of a real square matrix whose eigenvalues are wanted <u>without</u> their corresponding eigenvectors. The program requires no extra storage space (this is our reason for presenting it) and the running time is about 50% longer than for the <u>fastest</u> reliable program which only computes eigenvalues.

[†]Research sponsored by Office of Naval Research Contract N00014-69-A-0200-1017.

There are many industrial applications in which the matrix elements are known to only two or three decimal figures. Each condition number will indicate how accurately such a matrix determines the associated eigenvalue. When no digits in an eigenvalue are reliable the suspect eigenvalue should be tagged and this information passed on to a higher level in the whole computation.

A number of programming devices keep the code, storage, and running time down to a minimum.

An interesting case study is included.

Key words: eigenvalue, condition number

TABLE OF CONTENTS

Level 1 is a description designed for a busy colleague.

Level 2 is a description designed for publication.

Level 3 is a description designed for a programmer.

Page

Abstract . . . . Level 1 1

1 Theoretical Background . . . . Level 2 3

1.1 The Sensitivity of Eigenvalues 3

1.2 Invariance Properties 7

1.3 The Use and Cost of Condition Numbers 7

1.4 Operation Counts 10

2 Applicability . . . . Level 2 13

3 Organizational Details . . . . Level 2 14

3.1 Standardization 14

3.2 The Computation of the Eigenvectors of a Standardized Real, Block Upper Triangular Matrix 16

3.3 Closed Form Solution for Equations of Type 1 19

3.4 The Condition Number of Conjugate Pairs of Eigenvalues 21

4 Flow Chart for CONDIT . . . . Level 3 24

5 Formal Parameters and Usage . . . . Level 3 25

6 Programs and Comments . . . . Level 3 27

CONDIT (with facing comments) . . . . Level 3 33

7 Results 40

8 Bibliography 44

1. THEORETICAL BACKGROUND

1.1 The Sensitivity of Eigenvalues

Several good programs are available for the computation of the eigenvalues of real and complex matrices [Wilkinson & Reinsch, EISPACK, IMSL]. Due to the limitations of finite precision arithmetic these programs cannot produce, in general, the exact eigenvalues of the given matrix A. However the computed numbers are always (very close to) the eigenvalues of a matrix $A+E$ which is very close to A. This matrix E is not unique and error analyses [Wilkinson, 1965] have shown the existence of E's with satisfactorily small upper bounds on $\|E\|/\|A\|$. Here $\|\cdot\|$ denotes an appropriate matrix norm.

It follows from these remarks that a good program will not always deliver accurate approximations to A's eigenvalues. It can happen that some, or all, of the eigenvalues are very sensitive to changes in the matrix elements. So some, or all, of the eigenvalues of $A+E$ may differ sharply from those of A. Actually this is true only for non-normal matrices. Real symmetric matrices -- indeed all normal matrices -- determine their eigenvalues very well; the change induced in an eigenvalue of such an A cannot exceed the spectral norm of E (which is defined below).

Two questions arise. How can this sensitivity be measured and how cheaply can it be computed?

Simple Eigenvalues

To any simple eigenvalue λ of A there correspond both a column vector x and a row eigenvector y^* (the conjugate transpose of y) which are unique to within a scalar multiple. Thus

$$Ax = x\lambda \ , \quad y^*A = \lambda y^* \tag{1}$$

and $x = y$ if A is normal (i.e. $A^*A = AA^*$). The most popular measure of λ's sensitivity was called by Wilkinson [Wilkinson, 1965] the spectral condition number $\mathrm{cond}(\lambda)$. Let θ denote the acute angle between x and y, then

$$\mathrm{cond}(\lambda) \equiv \mathrm{secant}\ \theta = \|y\|\cdot\|x\|/|y^*x| \quad \text{where} \quad \|v\| = \sqrt{v^*v}\ . \tag{2}$$

This definition gives a number in $[1,\infty)$ which is monotonic increasing with λ's sensitivity to changes in A.

In order to justify this definition two popular matrix norms will be used;

$$\begin{aligned} \|M\| &\equiv \max_{v\neq 0} \|Mv\|/\|v\| = \sqrt{\lambda_{\max}(M^*M)}\ , \\ \|M\|_E &\equiv \sqrt{\sum_i \sum_j |m_{ij}|^2} = \sqrt{\mathrm{trace}(M^*M)}\ . \end{aligned} \tag{3}$$

Let $|\delta\lambda|$ be the change in λ corresponding to a change δA in A. It can be shown that

$$\mathrm{cond}(\lambda) = \sup|\delta\lambda|/\|\delta A\|_E \text{ over all non null infinitesimal } \delta A\ . \tag{4}$$

Another useful characterization of $\mathrm{cond}(\lambda)$ is the following. The spectral projector P_λ of λ is the matrix which projects every vector into a multiple of λ's eigenvector. It is easy to verify that for simple λ

$$P_\lambda = xy^*/y^*x \qquad (y^*x \text{ is a scalar})\ , \tag{5}$$

and, by using the fact that P_λ is of rank one, one can show that

$$\mathrm{cond}(\lambda) = \|P_\lambda\|_E = \|P_\lambda\| \ . \tag{6}$$

It is this characterization which can be generalized.

Multiple Eigenvalues

When λ has geometric multiplicity m almost all perturbations of A break λ into m simple eigenvalues in such a way that $\sup|\delta\lambda|/\|\delta A\|$ is unbounded. Thus it is customary to set

$$\mathrm{cond}(\lambda) = \infty$$

in this case.

There is more to be said however. A reasonable definition (see [Kahan 1972]) puts

$$\mathrm{cond}(\lambda) \equiv \sup|\delta\lambda|/\|\delta A\| \tag{7}$$

over all non null infinitesimal δA which preserve λ's multiplicity. This number can be estimated because

$$\mathrm{cond}(\lambda) \leq \|P_\lambda\|_E/m$$

where the spectral projector P_λ satisfies

$$AP_\lambda = P_\lambda A = \lambda P_\lambda + N_\lambda$$

and N_λ is nilpotent (i.e. $N_\lambda^m = 0$). Moreover P_λ can be found from the expression

$$P_\lambda = X(Y^*X)^{-1}Y^* \tag{8}$$

where the columns of X and rows of Y^* are bases for λ's invariant subspaces.

We have followed the usual practice (cond = ∞) in our program CONDIT but wish to point out that it is feasible to bring into adjacent positions on the diagonal of the Schur form any associated ill conditioned eigenvalues. The spectral projector for this group of eigenvalues can then be found from (8) and if its norm is small then the group can be designated as a cluster. That is another project.

If more specific information is required then the individual elements of P_λ will be involved because

$$\frac{\partial \lambda}{\partial a_{ij}} = e_j^* P_\lambda e_i \qquad (\lambda \text{ simple}) \ . \tag{9}$$

A warning should be offered at this point. The measures presented above are based on the Euclidean vector norm and the convention that A acts on vectors in Euclidean n-space. It can happen that this model is quite inappropriate for certain applications and then the conventional condition numbers will be irrelevant. However it is only the order of magnitude (base 10) of cond(λ) which is wanted, in most cases, and this will be constant over a large range of norms.

1.2 Invariance Properties

When the role of the matrix is to be stressed the condition number is written $\mathrm{cond}(\lambda,A)$.

Theorem. *If Q is unitary, i.e. $Q^*Q = QQ^* = I$, and λ is a simple eigenvalue of A then*

$$\mathrm{cond}(\lambda,QAQ^*) = \mathrm{cond}(\lambda,A) .$$

Proof. Let $Ax = x\lambda$, $y^*A = \lambda y^*$. Then

$$(QAQ^*)(Qx) = (Qx)\lambda , \quad (y^*Q^*)(QAQ^*) = \lambda(y^*Q^*) .$$

Because λ is simple $y^*x \neq 0$ and

$$\begin{aligned}\mathrm{cond}(\lambda,QAQ^*) &= \|y^*Q^*\|\cdot\|Qx\|/|(y^*Q^*)(Qx)| ,\\ &= \|y^*\|\cdot\|x\|/|y^*x| ,\\ &= \mathrm{cond}(\lambda,A) ,\end{aligned}$$

because the Euclidean norm is unitarily invariant. □

Corollary. *If a given matrix B is reduced to Hessenberg form H by unitary similarities (such as Householder transformations) and the QR algorithm is applied to H to produce, in the limit, a quasi-triangular matrix T then*

$$\mathrm{cond}(\lambda,B) = \mathrm{cond}(\lambda,T) .$$

1.3 The Use and Cost of Condition Numbers

A computed eigenvalue λ of a given matrix A is an exact eigenvalue of many matrices including some close to A. Let $A+E$ designate one of the closest matrices. Provided that $(\|E\|_E/\|A\|_E)^2$ is negligible the error in λ is bounded by $\text{cond}(\lambda)\|E\|_E$. Error analyses [Wilkinson 1966] give an upper bound β on $(\|E\|_E/\|A\|_E)$ when the Householder/QR method is used. It follows that

$$\log_{10}(|\lambda|/\beta\cdot\text{cond}(\lambda)\cdot\|A\|_E)$$

gives the number of decimal digits in λ which are assuredly correct.

When no figures in λ can be relied on then a warning tag should be attached to λ for most applications. Conversely when an adequate number of figures are certified as correct in each eigenvalue of A then the subsequent calculations are placed on a sounder footing.

These estimates of the number of correct figures have proved useful in comparison of rival eigenvalue programs and in debugging big programs of which the eigenvalue calculations were merely a part.

A natural question at this stage is how much extra does it cost to compute $\text{cond}(\lambda)$ as well as λ? The answer must depend on whether the user also computes x and/or y along with λ. We focus on real matrices and real arithmetic.

[A] If a complete Jordan factorization $A = X\Lambda Y^*$ $(Y^*X = I)$ is computed then each $\text{cond}(\lambda_i)$ can be found from the definition $\|x_i\|\|y_i^*\|/|y_i^*x|$ at negligible extra cost in storage and time. No special program is needed and this case will not be considered further. Few dependable Jordan factorization routines are currently available.

[B] If a program is used which yields X and Λ but not Y^* then it is necessary to compute the triangular factorization L_xU_x and store it in an extra array. Then $\text{cond}(\lambda_i) = \|e_i^*X^{-1}\|\cdot\|Xe_i\|$. To invert X costs n^3 basic operations whereas X and Λ may be found in approximately $7n^3$ operations using the double QR transformation.

No special program is needed. The time penalty is slight but the extra storage requirement is substantial. This case will not be discussed further.

[C] The eigenvalues λ of A may be found (EISPACK path, ELMHES, HQR) in under $4\frac{1}{6}n^3$ operations and with no supplementary $n \times n$ storage arrays provided that A can be overwritten. This is the most interesting case. No extra arrays are needed for the computation of $\mathrm{cond}(\lambda_i)$, $i = 1,\ldots,n$ but the multiplication count rises to approximately $7n^3$. See the section on Operation Counts for more details. The $O(n^2)$ terms bring down the ratio of running times and the increase is approximately 50% (± 15%).

Our method is easily described. The given matrix A is reduced to Hessenberg form H by orthogonal similarity transformations. Then H is transformed to quasi-triangular Schur form T by the double QR algorithm working on the whole of H and not just the remaining principal submatrices. None of the orthogonal transforming matrices is retained. Finally the column and row eigenvectors of T are found, for each λ, by back substitution and then discarded immediately after $\mathrm{cond}(\lambda)$ has been calculated.

By Theorem 1 $\mathrm{cond}(\lambda,T) = \mathrm{cond}(\lambda,A)$.

For simplicity all condition numbers exceeding 10^{30} are recorded as 10^{30}.

The program uses only real arithmetic even if A has complex eigenvalues.

1.4 Operation Counts

In [Parlett & Wang 1975] it is pointed out that straightforward counts of multiplications and additions are unreliable indicators of running times. Nevertheless they are good to within a factor of 2 and they do give insight into the way the algorithm spends its time. An op is defined as a scalar multiplication or division followed by an addition.

ORTHES: The $(n-j)^{th}$ step transforms the last j rows and columns while reducing column $(n-j)$ to upper Hessenberg form.

Row Operations: $A \to A' = A - w\gamma(w^TA)$, $\gamma = 2/w^Tw$, $w^T = (0,\dots,0,x,\dots,x)$

Computation	γ	w^TA	$v^T = \gamma(w^TA)$	$A - wv^T$	Total
Cost	j	j^2	j	j^2	$2j(j+1)$

Column Operations: $A' \to A'' = A' - \gamma A'ww^T$

Computation	γ	$A'w$	$u = \gamma A'w$	$A' - uw^T$	Total
Cost	0	nj	n	nj	$n(2j+1)$

Grand Total: $\sum_{j=1}^{n-1} n(2j+1) + 2j(j+1) = \frac{5}{3}n^2(n-1) + 0(n)$

The program ELMHES is approximately twice as fast as ORTHES but will not preserve condition numbers.

CONDIT: It suffices to assume that all eigenvalues are real. To find the column and row eigenvectors for the j^{th} eigenvalue requires backsolving triangular systems of $(j-1)$ and $(n-j-1)$ equations respectively.

Computation	x	y^*	Cond
Cost	$\sum_{i=1}^{j-1} i$	$\sum_{i=1}^{n-j-1} i$	j

<u>Grand Total</u>: $\frac{1}{3}n^3 + \frac{1}{2}n^2 + O(n)$

HQR: A typical double QR transformation acts on a $j \times j$ submatrix of a Hessenberg matrix. To restore column k to Hessenberg form requires the following operations.

Computation	Key quantities	Row operations	Column operations
Cost	9	$\sum_{\ell=k}^{j} 5$	$\sum_{\ell=1}^{\min(k+3,j)} 5$

Total: $\sum_{k=1}^{j} [9 + 5(j-k+1) + 5(k+3)] = 5j^2 + 29j + O(j)$

Assume four initial full transformations with $j = n$ and then two iterations per eigenvalue.

<u>Grant Total</u>: $\frac{10}{3}n^3 + 54n^2 + O(n)$

QR2NØZ: The same transformations as in HQR must act on the whole matrix. This changes the range of the row operation and not the column because the $j \times j$ submatrix being transformed is the leading principal submatrix.

Computation	Key quantities	Row operations	Column operations
Cost	9	$\sum_{\ell=k}^{n} 5$	$\sum_{\ell=1}^{\min(k+3,j)} 5$

Total: $\sum_{k=1}^{j} [9 + 5(n-k+1) + 5(k+3)] = 5nj + 29j + O(j)$

With the same assumptions as above

<u>Grand Total</u>: $5n^3 + 54n^2 + O(n)$

Summary of Op Counts

ELMHES + HQR :	$4\frac{1}{6}n^3 + 53\frac{1}{6}n^2 + 0(n)$
ORTHES + HQR :	$5n^3 + 52\frac{1}{3}n^2 + 0(n)$
ORTHES + QR2NOZ + CONDIT:	$7n^3 + 52\frac{5}{6}n^2 + 0(n)$

The actual timings were more favorable to our program than these operation counts suggest. The assumption of two iterations per eigenvalue is unrealistic. In practice there are more iterations with larger values of j and fewer with small values. With $20 \le n \le 60$ our program ran, on the average, 50% longer than did ELMHES + HQR; the worst case ran 65% longer.

2. APPLICABILITY

The program accepts real square matrices which can be stored in the high speed memory of the computer.

The condition numbers of all eigenvalues of all normal matrices (and this includes symmetric matrices) are unity and consequently the program is intended for use with nonnormal matrices.

Before our programs QR2NOZ and CONDIT are used A should be reduced to Hessenberg form H by orthogonal congruences. We recommend the procedure ORTHES in [Wilkinson & Reinsch, II/13] and its Fortran counterpart ORTHES [Eispack Guide, p. 297].

Our program QR2NOZ is an adaptation of HQR2 (Eispack Guide, p. 248) designed to avoid the formation of the product of all the similarity transformations used in the double QR algorithm and the calculation of the eigenvectors of the final matrix of the QR sequence. A listing is included for completeness.

3. ORGANIZATIONAL DETAILS

3.1 Standardization

(i) In the course of the QR algorithm applied to H it is possible for two real eigenvalues to be found, at the same time, as the roots of a 2×2 diagonal block

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} .$$

It is convenient in such cases to do a supplementary plane rotation which will reduce this diagonal block to upper triangular form and change the corresponding rows and columns of H accordingly.

If this transformation is done at the time the eigenvalues λ_1 and λ_2 are recorded then some of the quantities which determine the correct angle of rotation will be available.

This device is employed in HQR2 and has been carried over to QR2NOZ. The details are given below.

The parameters $c = \cos\theta$, $s = \sin\theta$ are determined so that

$$\begin{pmatrix} c & -s \\ s & c \end{pmatrix}\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\begin{pmatrix} c & s \\ -s & c \end{pmatrix}$$

is upper triangular. Thus

$$\gamma c^2 - dcs - \beta s^2 = 0 , \qquad d = \delta - \alpha .$$

Let $t = (d/2)^2 + \beta\gamma$ then

$$\begin{aligned} \cot\theta &= (d/2 + \text{sign}(d)\sqrt{t})/2\gamma , \\ s &= \text{sign}(\cot\theta)(1 + \cot^2\theta)^{-1/2} , \\ c &= s \cdot \cot\theta . \end{aligned}$$

(ii) It is also convenient to perform a supplementary plane rotation after a pair of complex conjugate eigenvalues, $\lambda \pm i\mu$, has been recorded in the course of the QR algorithm. In this case the transformation of the diagonal block is

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\begin{pmatrix} c & -s \\ s & c \end{pmatrix} = \begin{pmatrix} \lambda & \theta \\ \xi & \lambda \end{pmatrix}$$

where $\xi\theta = -\mu^2$. This device is not used in HQR2.

Note that it is not in general possible to transform

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \rightarrow \begin{pmatrix} \lambda & -\mu \\ \mu & \lambda \end{pmatrix}$$

using <u>orthogonal</u> similarity transformations.

The purpose of the transformation is to yield a simple solution to certain systems of linear equations which must be solved. The supplementary plane rotation is done at the stage when the eigenvalues are being recorded in QR2NOZ. In this case $t \equiv p^2 + \beta\gamma < 0$, $p = (\alpha-\delta)/2$. We want to choose $c = \cos\theta$ and $s = \sin\theta$ so that

$$\alpha c^2 + (\beta+\gamma)cs + \delta s^2 = \delta c^2 - (\beta+\gamma)cs + \alpha s^2 .$$

Hence

$$\tan 2\theta = \frac{2sc}{c^2-s^2} = -\frac{2p}{\sigma} = \frac{2|p|}{|\sigma|}\text{sign}(-p\sigma) , \qquad \sigma = \beta+\gamma .$$

Let $\tau = \sqrt{\sigma^2 + 4p^2}$. Then

$$\cos\theta = q = \sqrt{\tfrac{1}{2}(1+\cos 2\theta)} = \sqrt{(1+|\sigma|/\tau)/2} ,$$

$$\sin\theta = \sin 2\theta/2\cos\theta = |p|\text{sign}(-p\sigma)/\tau q .$$

3.2 The Computation of the Eigenvectors of a Standardized Real, Block Upper Triangular Matrix

For each real eigenvalue λ the eigenvectors u, w^* satisfy

$$Tu = u\lambda , \qquad w^*T = \lambda w^* .$$

For each complex conjugate pair of eigenvalues $\lambda \pm i\mu$ the eigenvectors $u_1 \pm iu_2$, $w_1^* \mp iw_2^*$ satisfy

$$T(u_1,u_2) = (u_1,u_2)\hat{E} , \qquad (w_1,w_2)^*T = \hat{E}(w_1,w_2)^*$$

where

$$\hat{E} = \begin{bmatrix} \lambda & \mu \\ -\mu & \lambda \end{bmatrix} .$$

In effecting the back substitution process in real arithmetic there are four different cases which can occur, depending on whether the matrices D and E shown below are 1×1 or 2×2.

$$T = \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ & D & \cdot & \cdot & \cdot \\ & & \cdot & \cdot & \cdot \\ \bigcirc & & & E & \cdot \\ & & & & \cdot \end{bmatrix} \qquad D = \begin{cases} \alpha \text{ or} \\ \begin{bmatrix} \alpha & \beta \\ \gamma & \alpha \end{bmatrix} \end{cases} \qquad E = \begin{cases} \lambda \text{ or} \\ \begin{bmatrix} \lambda & \phi \\ \theta & \lambda \end{bmatrix} \end{cases} ,$$

where $\theta\phi = -\mu^2$.

The positions of D and E should be exchanged when considering the row eigenvectors.

Type 1: pair-pair (E is 2×2, D is 2×2).

Imagine that the elements of u_1, u_2 in the same row as D are about to be computed. All elements below these have already been found, the elements below E being 0.

Let j1, j2 be the rows of T in which D lies. Then the unknowns are

$$V = \begin{pmatrix} u_1(j1) & u_2(j1) \\ u_1(j2) & u_2(j2) \end{pmatrix} .$$

The equation to be solved in the column case is

$$- DV + V\hat{E} = R$$

where

$$R = \begin{pmatrix} r_1(j1) & r_2(j1) \\ r_1(j2) & r_2(j2) \end{pmatrix} , \quad r_\nu(m) = \sum_{k=j2+1}^{i+1} t_{m,k} u_\nu(k) \ , \begin{cases} m = j1, j2 , \\ \nu = 1, 2 . \end{cases} \qquad (1)$$

In the row case let

$$V = \begin{pmatrix} w_1(j1) & w_2(j1) \\ w_1(j2) & w_2(j2) \end{pmatrix}$$

then the equations to be solved are

$$- V^T D + \hat{E} V^T = R^T \qquad (2)$$

where R is as above except that k runs from i to j1 - 1. Transposing yields

$$- D^T V + V\hat{E}^T = R .$$

Comparing this with the column case we see that it is only necessary to transpose D and $\hat{E}$ (i.e., to exchange β, γ and μ, $-\mu$) in order to use the same code for both cases. The way that this exchange is accomplished is described in Level Three.

The way in which these four linear equations in four unknowns are solved is described in the next section.

Type 2: pair-single (E is 2×2, D is 1×1).

The relevant equations are

$$-\alpha(u_1(j),u_2(j)) + (u_1(j),u_2(j))\hat{E} = (r_1(j),r_2(j))$$

and

$$-\alpha(w_1(j),w_2(j)) + (w_1(j),w_2(j))\hat{E}^T = (r_1(j),r_2(j)) \ .$$

Let $d = \lambda - \alpha$, $\text{den} = d^2 + \mu^2$, $\text{val} = \begin{cases} \mu & \text{(for column)} \\ -\mu & \text{(for row)} \ . \end{cases}$ The solution for both cases is

$$\begin{aligned} v_1 &= (r_1 \cdot d + r_2 \cdot \text{val})/\text{den} \ , \\ v_2 &= (-r_1 \cdot \text{val} + r_2 \cdot d)/\text{den} \ . \end{aligned} \qquad (3)$$

Type 3: single-pair (E is 1×1, D is 2×2).

The relevant equations are

$$- D\begin{pmatrix} u_1(j1) \\ u_1(j2) \end{pmatrix} + \begin{pmatrix} u_1(j1) \\ u_1(j2) \end{pmatrix}\lambda = \begin{pmatrix} r_1(j1) \\ r_1(j2) \end{pmatrix}$$

and the same equation for w_1 with D^T in place of D. Set $d = \lambda - \alpha$, $\text{den} = d^2 - \beta\gamma$. The solution is

$$\begin{aligned} v_1 &= (r_1(j1) \cdot d + r_1(j2) \cdot \tilde{\beta})/\text{den} \ , \\ v_2 &= (r_1(j1) \cdot \tilde{\gamma} + r_1(j2) \cdot d)/\text{den} \ , \end{aligned} \qquad (4)$$

where $\tilde{\beta} = \begin{Bmatrix} \beta & \text{(for column)} \\ \gamma & \text{(for row)} \end{Bmatrix}$ and $\tilde{\gamma} = \begin{Bmatrix} \gamma & \text{(for column)} \\ \beta & \text{(for row)} \end{Bmatrix}$. In practice $\tilde{\beta} = T(JJ,J)$, $\tilde{\gamma} = T(J,JJ)$ and the setting of J and JJ is described at Level Three.

Type 4: single-single (E is 1×1, D is 1×1).

$$v_1 = r_1(j)/\text{den}\ , \qquad \text{den} = \lambda - \alpha\ .$$

Type 5: formula breakdown.

If in any of the previous cases D = E then the formulae for solution breakdown. There are two cases to consider.

(i) Linear Independence. Any element v_j for which the formula yields 0/0 can be set to any value, the most convenient is 0. This represents the existence of a whole space of eigenvectors associated with E.

(ii) Defective Case. Any element v_j for which the formula yields a value exceeding 1/TOL will cause the condition number to exceed 1/TOL. If this case is detected computation is interrupted, the condition number is set to 1/TOL and the program proceeds to the next eigenvalue.

We propose that $\text{TOL} = 10^{-30}$ will be suitable for most applications and most computers.

These tests make the code simple and machine independent. However, it is possible to devise matrices for which the given value 1/TOL for the condition is very misleading. We know of no failsafe procedure which does not involve deciding the rank of $T - \xi$ for all ξ in a neighborhood of λ. This is a costly, difficult, and often unrewarding task.

3.3 Closed Form Solution for Equations of Type 1

The equations to be solved are of the form

$$-DV + V\hat{E} = R$$

where

$$D = \begin{pmatrix} \alpha & \beta \\ \gamma & \alpha \end{pmatrix}, \quad \hat{E} = \begin{pmatrix} \lambda & \mu \\ -\mu & \lambda \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix}.$$

The standardization of the block triangular matrix T forces the diagonal elements of D to be equal. This yields simple formulas for the elements of V.

Rewrite the equation as

$$(v_{11}\ v_{12}\ v_{21}\ v_{22}) \begin{pmatrix} B & -\gamma I_2 \\ -\beta I_2 & B \end{pmatrix} = (r_{11}\ r_{12}\ r_{21}\ r_{22}) = r^T$$

where

$$B = \begin{pmatrix} \lambda-\alpha & \mu \\ -\mu & \lambda-\alpha \end{pmatrix}.$$

Observe that

$$\begin{pmatrix} B & -\gamma I_2 \\ -\beta I_2 & B \end{pmatrix} \begin{pmatrix} B^T & \gamma I_2 \\ \beta I_2 & B^T \end{pmatrix} = \tau I_4 + 2\mu J_4$$

where

$$\tau = (\lambda-\alpha)^2 + \mu^2 - \beta\gamma \quad (\text{and} \quad \beta\gamma < 0),$$

$$J_4 = \begin{bmatrix} 0 & 0 & 0 & \gamma \\ 0 & 0 & -\gamma & 0 \\ 0 & \beta & 0 & 0 \\ -\beta & 0 & 0 & 0 \end{bmatrix}, \quad I_k \text{ is the } k \times k \text{ identity matrix.}$$

Further

$$(\tau I_4 + 2\mu J_4)(\tau I_4 - 2\mu J_4) = [\tau^2 - 4\mu^2(-\beta\gamma)] I_4 .$$

Hence

$$(v_{11}\ v_{12}\ v_{21}\ v_{22})(\tau^2 + 4\mu^2\beta\gamma) = r^T \begin{pmatrix} B^T & \gamma I_2 \\ \beta I_2 & B^T \end{pmatrix} (\tau I_4 - 2\mu J_4)$$

$$= r^T \begin{bmatrix} e & -f & \gamma g & -\gamma h \\ f & e & \gamma h & \gamma g \\ \beta g & -\beta h & e & -f \\ \beta h & \beta g & f & e \end{bmatrix}$$

where

$$d = \lambda - \alpha \ , \quad e = d\tau \ , \quad f = \mu(\tau + 2\beta\gamma) \ , \quad g = \tau - 2\mu^2 \ , \quad h = 2d\mu \ .$$

Note that

$$\begin{aligned} \tau^2 + 4\mu^2\beta\gamma &= (d^2 + \mu^2 - \beta\gamma)^2 + 4\mu^2\beta\gamma \\ &= g^2 + h^2 \\ &(= 0 \ \text{ if and only if } \ \alpha = \lambda, \ \ \mu^2 = -\beta\gamma) \ . \end{aligned}$$

These same formulae will be valid for the row eigenvectors provided that we exchange (β,γ) and $(\mu,-\mu)$.

The alternative to using this closed form solution is to code up a special version of Gaussian Elimination with pivoting. It is the pivoting which would lengthen the code considerably.

3.4 <u>The Condition Number of Conjugate Pairs of Eigenvalues</u>

Let $\lambda \pm i\mu$ be a complex pair of eigenvalues of the real Schur matrix T obtained by the QR algorithm. In the course of the algorithm the following real equations are solved for real n-vectors u_1, u_2, w_1, w_2

$$T(u_1,u_2) = (u_1,u_2)\begin{pmatrix} \rho & \mu \\ -\mu & \rho \end{pmatrix} \ , \quad (w_1,w_2)^*T = \begin{pmatrix} \rho & \mu \\ -\mu & \rho \end{pmatrix}(w_1,w_2)^* \ . \qquad (5)$$

Thus $\text{span}(u_1,u_2)$ and $\text{span}(w_1^*,w_2^*)$ are real invariant subspaces under T. However $\{u_1,u_2\}$ and $\{w_1^*,w_2^*\}$ are very special bases of these spaces.

Lemma. With the notation given above $u_1 \pm iu_2$ and $w_1^* \mp iw_2^*$ are the column and row eigenvectors belonging to $\lambda \pm i\mu$.

Proof. From (5)

$$\begin{aligned} Tu_1 &= u_1\lambda - u_2\mu , & w_1^*T &= \lambda w_1^* + \mu w_2^* , \\ Tu_2 &= u_1\mu + u_2\lambda , & w_2^*T &= -\mu w_1^* + \lambda w_2^* . \end{aligned}$$

Hence

$$\begin{aligned} T(u_1 + iu_2) &= u_1(\lambda + i\mu) + iu_2(i\mu + \lambda) , \\ (w_1^* - iw_2^*)T &= (\lambda + i\mu)w_1^* - (\lambda + i\mu)iw_2^* . \end{aligned}$$

The eigenvectors for $\lambda - i\mu$ are obtained in the same way. $\square$

Consequently

$$\mathrm{cond}(\lambda \pm i\mu) = \|u_1 + iu_2\| \cdot \|w_1^* - iw_2^*\| / [w_1^*u_1 + w_2^*u_2 + i(w_1^*u_2 - w_2^*u_2)] .$$

Use was made in the lemma of the quasi-triangular nature of T. A consequence of this form is that u_1 and w_1 can be packed into the same real n-vector with two overlapping elements as indicated.

$$\left.\begin{aligned} u_1^* &= (x,\dots,x,p_i,q_i,0,\dots,0) \\ w_i^* &= (0,\dots,0,\bar{p}_i,\bar{q}_i,x,\dots,x) \end{aligned}\right\} \quad i = 1,\ 2$$

The equations to be satisfied by p_i, q_i are of the form

$$\begin{pmatrix} \lambda & \beta \\ \gamma & \lambda \end{pmatrix}\begin{pmatrix} p_1 & p_2 \\ q_1 & q_2 \end{pmatrix} - \begin{pmatrix} p_1 & p_2 \\ q_1 & q_2 \end{pmatrix}\begin{pmatrix} \lambda & \mu \\ -\mu & \lambda \end{pmatrix} = 0 ,$$

$$\begin{pmatrix} \bar{p}_1 & \bar{q}_1 \\ \bar{p}_2 & \bar{q}_2 \end{pmatrix}\begin{pmatrix} \lambda & \beta \\ \gamma & \lambda \end{pmatrix} - \begin{pmatrix} \lambda & \mu \\ -\mu & \lambda \end{pmatrix}\begin{pmatrix} \bar{p}_1 & \bar{q}_1 \\ \bar{p}_2 & \bar{q}_2 \end{pmatrix} = 0$$

where $\mu^2 = -\beta\gamma$. These equations reduce to

$$\beta q_2 = p_1\mu \quad , \qquad \bar{p}_1\beta = \mu\bar{q}_2 \ ,$$
$$\beta q_1 = -\mu p_2 \ , \qquad \beta\bar{p}_2 = -\mu\bar{q}_1 \ .$$

The simplest solution (which we adopt) takes

$$p_1 = \bar{p}_1 = 1 \ , \quad q_1 = p_2 = \bar{q}_1 = \bar{p}_2 = 0 \ , \quad q_2 = \mu/\beta \ , \quad \bar{q}_2 = 1/q_2 \ .$$

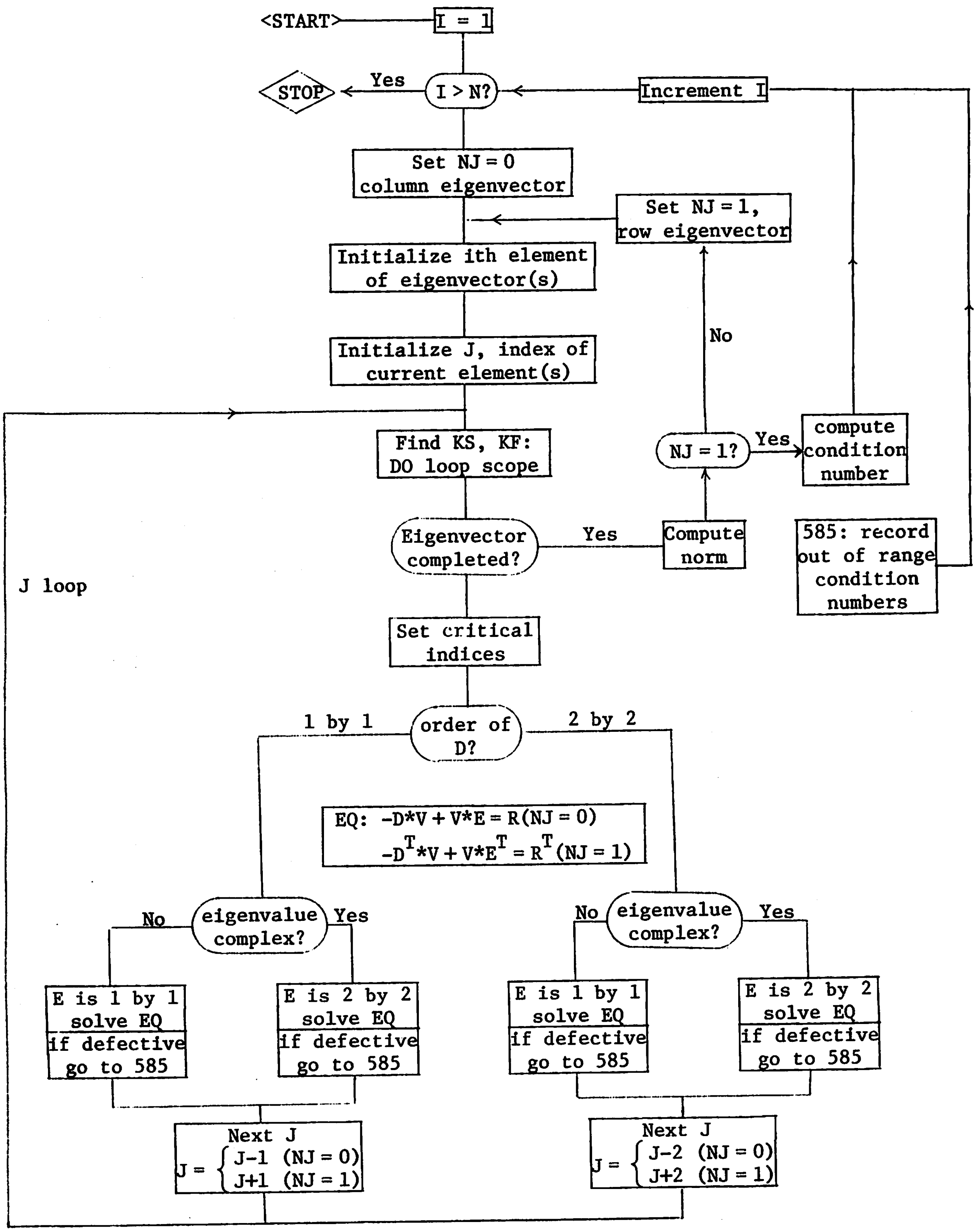
With this choice

$$(w_1 - iw_2)^*(u_1 + iu_2) = (\bar{p}_1 p_1 + \bar{p}_2 p_2) + (\bar{q}_1 q_1 + \bar{q}_2 q_2) = 2$$

and

$$\operatorname{cond}(\lambda \pm i\mu) = [(\|u_1\|^2 + \|u_2\|^2)(\|w_1\|^2 + \|w_2\|^2)]^{1/2}/2 \ . \qquad (6)$$

4. FLOW CHART FOR CONDIT



5. FORMAL PARAMETERS AND USAGE

```
C
C                        SUBROUTINE ORTHES
C        FROM EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)
C
C     PURPOSE
C     THE FORTRAN SUBROUTINE ORTHES REDUCES A REAL MATRIX TO UPPER
C        HESSENBERG FORM USING ORTHOGONAL SIMILARITY TRANSFORMATIONS.
C
C     CALLING SEQUENCE
C     THE SUBROUTINE STATEMENT IS
C          SUBROUTINE ORTHES(NM,N,LOW,IGH,A,ORT).
C
C     NM       IS AN INTEGER INPUT VARIABLE SET EQUAL TO THE ROW DIMENSION
C              OF THE TWO DIMENSIONAL ARRAY A AS SPECIFIED IN THE
C              CALLING PROGRAM.
C     N        IS AN INTEGER INPUT VARIABLE SET EQUAL TO THE ORDER OF THE
C              MATRIX A. N MUST BE NOT GREATER THAN NM.
C     LOW,IGH  ARE INTEGER INPUT VARIABLES INDICATING THE BOUNDARY INDICES
C              FOR THE BALANCED MATRIX.  IF THE MATRIX IS NOT BALANCED,SET
C              LOW TO 1 AND IGH TO N.
C     A        IS A REAL TWO-DIMENSIONAL VARIABLE WITH ROW DIMENSION NM
C              AND COMUMN DIMENSION AT LEAST N.  ON INPUT, A CONTAINS THE
C              MATRIX OF ORDER N TO BE REDUCED TO HESSENBERG FORM.  ON
C              OUTPUT, A CONTAINS THE UPPER HESSENBERG MATRIX AS WELL AS
C              SOME INFORMATION ABOUT THE ORTHOGONAL TRANSFORMATIONS USED
C              IN THE REDUCTION.
C     ORT      IS A REAL OUTPUT ONE-DIMENSIONAL VARIABLE OF DIMENSION AT
C              LEAST IGH CONTAINING THE REMAINING INFORMATION ABOUT THE
C              ORTHOGONAL TRANSFORMATIONS.
C
C
C
C
C
C
C                        SUBROUTINE QR2NOZ
C
C     PURPOSE
C     THE FORTRAN SUBROUTINE QR2NOZ COMPUTES THE EIGENVALUES OF A REAL
C          UPPER HESSENBERG MATRIX USING THE QR METHOD AND REDUCES THE
C          MATRIX TO A STANDARIZED QUASI-TRIANGULAR FORM.  COMPUTATIONS
C          ARE DONE IN REAL ARITHMETIC.
C
C     THE SUBROUTINE STATEMENT IS
C        SUBROUTINE QR2NOZ(NM,N,LOW,IGH,H,WR,WI,IERR).
C
C     NM       IS AN INTEGER INPUT VARIABLE SET EQUAL TO THE ROW DIMENSION
C              OF THE ARRAY H AS SPECIFIED IN THE CALLING PROGRAM.
C     N        IS AN INTEGER INPUT VARIABLE SET EQUAL TO THE ORDER OF THE
C              MATRIX H.  N  .LE.  NM
C     LOW,IGH  ARE INTEGER INPUT VARIABLES INDICATING THE BOUNDARY INDICES
C              FOR THE BALANCED MATRIX.  IF THE MATRIX IS NOT BALANCED SET
C              LOW TO 1 AND IGH TO N.
C     H        IS A REAL TWO-DIMENSIONAL ARRAY WITH ROW DIMENSION NM AND
C              COLUMN DIMENSION AT LEAST N.  ON INPUT IT CONTAINS THE
C              UPPER HESSENBERG MATRIX OF ORDER N.  ON OUTPUT IT CONTAINS
C              THE STANDARIZED QUASI-TRIANGULAR MATRIX.
C     WR,WI    ARE REAL OUTPUT ONE-DIMENSIONAL VARIABLES OF DIMENSION AT
C              LEAST N CONTAINING THE REAL AND IMAGINARY PARTS,
C              RESPECTIVELY, OF THE EIGENVALUES OF THE HESSENBERG MATRIX.
C              THE EIGENVALUES ARE UNORDERED EXCEPT THAT COMPLEX CONJUATE
C              PAIRS OF EIGENVALUES APPEAR CONSECUTIVELY WITH THE
C              EIGENVALUE HAVING THE POSITIVE IMAGINARY PART FIRST.
C     IERR     IS AN INTEGER OUTPUT VARIABLE SET EQUAL TO AN ERROR
C              COMPLETION CODE.  IF MORE THAN 30 ITERATIONS ARE REQUIRED
C              TO DETERMINE AN EIGENVALUE, THIS SUBROUTINE TERMINATES WITH
C              IERR SET EQUAL TO THE INDEX OF THE EIGENVALUE FOR WHICH
C              FAILURE OCCURS.  THE EIGENVALUES IN THE WR AND WI ARRAYS
C              SHOULD BE CORRECT FOR INDICES  IERR+1,IERR+2,...,N. IF ALL
C              THE EIGENVALUES ARE DETERMINED WITHIN 30 ITERATIONS, IERR
C              IS SET TO ZERO.
C
```

```
C
C
C
C                    SUBROUTINE CONDIT
C
C  CONDIT COMPUTES THE CONDITION NUMBERS OF THE EIGENVALUES OF A
C     STANDARIZED QUASI-TRIANGULAR MATRIX.
C
C   THE SUBROUTINE STATEMENT IS
C                    SUBROUTINE CONDIT(NM,N,A,V1,V2,WI,COND).
C ON INPUT
C   NM      MUST BE SET TO THE ROW DIMENSION OF THE TWO DIMENSIONAL
C           ARRAY AS DECLARED IN THE CALLING PROGRAM.
C   N       IS THE ORDER OF THE MATRIX.  N.LE.NM
C   A       CONTAINS THE STANDARIZED QUASI-TRIANGULAR MATRIX PRODUCED BY
C           QR2NOZ.
C   WI      CONTAINS THE IMAGINARY PARTS OF THE EIGENVALUES.  THE
C           EIGENVALUES ARE UNORDERED EXCEPT THAT COMPLEX CONJUGATE PAIRS
C           APPEAR CONSECUTIVELY.
C   V1,V2   ARE FOR TEMPORARY STORAGE.
C
C ON OUTPUT
C   A       IS UNALTERED.
C   COND    CONTAINS THE CONDITION NUMBERS CORRESPONDING TO THE
C           EIGENVALUES IN (V2,WI).  COND = 1./TOL IF THE USUAL
C           FORMULA WOULD CAUSE OVERFLOW OR YIELD A VALUE EXCEEDING
C           1/TOL.  TOL NEED NOT DEPEND ON THE COMPUTER.
C   V2      CONTAINS THE REAL PARTS OF THE EIGENVALUES.
C
C
C
C
C                             TYPICAL USAGE
C
C
C     DIMENSION A(50,50),WR(50),WI(50),COND(50),ORT(50)
C
C*******************ENTER MATRIX A AND DIMENSIONS N,NM*****************
C
C     LOW = 1
C     IGH = N
C     CALL ORTHES(NM,N,LOW,IGH,A,ORT)
      CALL QR2NOZ(NM,N,LOW,IGH,A,WR,WI,IERR)
C     CALL CONDIT(NM,N,A,ORT,WR,WI,COND)
C
C**********************************************************************
C
C  NOTE THE USE OF ORT AND WR IN CONDIT
C
C
```

6. PROGRAMS AND COMMENTS

QR2NOZ is a modification of the EISPACK program HQR2.

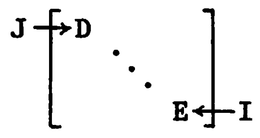
```
      SUBROUTINE  QR2NOZ(NM,N,LOW,IGH,H,WR,WI,IERR)
      DIMENSION H(NM,N),WR(N),WI(N)
      REAL NORM,MACHEP
      INTEGER EN,ENM2
      LOGICAL NOTLAS
      DATA MACHEP     /01642400000000000000/
      IERR = 0
C
C  STORE ROOTS ISOLATED BY BALANC
C
      DO 50 I = 1,N
          IF (I.GE.LOW .AND. I.LE.IGH) GOTO 50
          WR(I) = H(I,I)
          WI(I) = 0.0
   50 CONTINUE
C
      EN = IGH
      T = 0.0
C
C   SEARCH FOR NEXT EIGENVALUES
C
   60 IF(EN.LT.LOW) RETURN
      ITS = 0
      NA = EN - 1
      ENM2 = NA - 1
C
C   LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C      FOR L=EN STEP -1 UNTIL LOW DO
C
   70 IF (EN.EQ.LOW) GOTO 90
      DO 80 LL=LOW,NA
         L=EN+LOW-LL
         IF(ABS(H(L,L-1)).LE.MACHEP*(ABS(H(L-1,L-1))
     X            + ABS(H(L,L)))GO TO 100
   80 CONTINUE
   90 L = LOW
C
C     FORM SHIFT
C
  100 X = H(EN,EN)
      IF (L.EQ.EN) GOTO 270
      Y = H(NA,NA)
      W = H(EN,NA) * H(NA,EN)
      IF (L.EQ.NA) GOTO 300
      IF (ITS.EQ.30) GOTO 1000
      IF (ITS.NE.10 .AND. ITS.NE.20) GOTO 130
C
C     FORM EXCEPTIONAL SHIFT
C
      T = T + X
C
      DO 120 I = LOW ,EN
  120 H(I,I) = H(I,I) - X
```

```
      S = ABS(H(EN,NA)) + ABS(H(NA,ENM2))
      X = 0.75 * S
      Y = X
      W = -0.4375*S*S
  130 ITS = ITS + 1
C
C    LOOK FOR TWO CONSECUTIVE SMALL SUB-DIAGONAL
C     ELEMENTS.  FOR M=EN-2 STEP -1 UNTIL L DO
C
      DO 140 MM = L,ENM2
         M = ENM2 + L - MM
         ZZ = H(M,M)
         R = X - ZZ
         S = Y - ZZ
         P = (R*S -W)/H(M+1,M)    +   H(M,M+1)
         Q = H(M+1,M+1) - ZZ - R - S
         R = H(M+2,M+1)
         S = ABS(P) + ABS(Q) + ABS(R)
         P = P/S
         Q = Q/S
         R = R/S
         IF (M.EQ.L) GOTO 150
         IF (ABS(H(M,M-1))*(ABS(Q) + ABS(R)).LE.MACHEP*ABS(P)
     X      *(ABS(H(M-1,M-1)) + ABS(ZZ) + ABS(H(M+1,M+1)))) GOTO 150
  140 CONTINUE
C
  150 MP2 = M + 2
C
      DO 160 I = MP2,EN
         H(I,I-2) = 0.0
         IF (I.EQ.MP2) GOTO 160
         H(I,I-3) = 0.0
  160 CONTINUE
C
C    DOUBLE QR STEP INVOLVING ROWS L TO EN
C      AND COLUMNS M TO EN.
C
      DO 260 K = M,NA
         NOTLAS = K.NE.NA
         IF (K.EQ.M) GOTO 170
         P = H(K,K-1)
         Q = H(K+1,K-1)
         R = 0.0
         IF (NOTLAS) R = H(K+2,K-1)
         X = ABS(P) + ABS(Q) + ABS(R)
         IF (X.EQ.0.0) GOTO 260
         P = P/X
         Q = Q/X
         R = R/X
  170    S = SIGN(SQRT(P*P + Q*Q + R*R),P)
         IF (K.EQ.M) GOTO 180
         H(K,K-1) = -S*X
         GOTO 190
  180    IF (L.NE.M) H(K,K-1) = -H(K,K-1)
  190      P = P + S
           X = P/S
           Y = Q/S
           ZZ = R/S
           Q = Q/P
           R = R/P
C
C   ROW MODIFICATION
C
           DO 210 J = K,N
                P = H(K,J) + Q*H(K+1,J)
                IF (.NOT.NOTLAS) GOTO 200
                P = P + R*H(K+2,J)
                H(K+2,J) = H(K+2,J) - P*ZZ
  200           H(K+1,J) = H(K+1,J) - P*Y
                H(K,J) = H(K,J) - P*X
  210      CONTINUE
C
            J = MIN0(EN,K+3)
C
C   COLUMN MODIFICATION
C
           DO 230 I = 1,J
                P = X*H(I,K) + Y*H(I,K+1)
                IF (.NOT.NOTLAS) GOTO 220
                P = P + ZZ*H(I,K+2)
                H(I,K+2) = H(I,K+2) - P*R
  220           H(I,K+1) = H(I,K+1) - P*Q
                H(I,K) = H(I,K) - P
  230      CONTINUE
  260 CONTINUE
      GO TO 70
C
C  ONE ROOT FOUND
C
  270 H(EN,EN)=X+T
      WR(EN)=H(EN,EN)
      WI(EN)=0.0
C
  290 EN = NA
      GOTO 60
C
C  TWO ROOTS FOUND
C
  300 P = (Y-X)/2.0
      Q = P*P + W
      ZZ = SQRT(ABS(Q))
      H(EN,EN) = X + T
      X = H(EN,EN)
      H(NA,NA) = Y + T
      IF (Q.LT.0.0) GOTO 310
      ZZ = P + SIGN(ZZ,P)
C
C    REAL PAIR
```

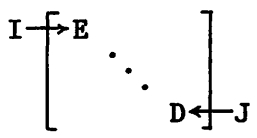
```
C
      WR(NA) = X + ZZ
      WR(EN) = WR(NA)
      IF ( ZZ.NE.0.0) WR(EN) = X - W/ZZ
      WI(NA) = 0.0
      WI(EN) = 0.0
      X = H(EN,NA)
      R = SQRT(X*X + ZZ*ZZ)
      P = X/R
      Q = ZZ/R
      GOTO 320
C
C   COMPLEX PAIR
C
  310 WR(NA) = X + P
      WR(EN) = X + P
      WI(NA) = ZZ
      WI(EN) = -ZZ
C
C  MAKE DIAGONAL ELEMENTS EQUAL
C
      IF (P.EQ.0.0) GOTO 380
      BPC = H(EN,NA) + H(NA,EN)
      TX = SQRT(BPC*BPC + 4.0* P*P)
      Q = SQRT(.5 * (1.0 + ABS(BPC)/TX))
      P = SIGN(P/(Q*TX),-BPC*P)
C
C     ROW MODIFICATION
C
  320 DO 330 J = NA,N
          ZZ = H(NA,J)
          H(NA,J) = Q*ZZ + P*H(EN,J)
          H(EN,J) = Q*H(EN,J) - P*ZZ
  330 CONTINUE
C
C      COLUMN MODIFICATION
C
      DO 340 I = 1,N
          ZZ = H(I,NA)
          H(I,NA) = Q*ZZ + P*H(I,EN)
          H(I,EN) = Q*H(I,EN) - P*ZZ
  340 CONTINUE
  380 EN = ENM2
      GOTO 60
 1000 IERR = EN
      RETURN
      END
      SUBROUTINE CONDIT(NM,N,A,V1,V2,WI,COND)
C
      DIMENSION A(NM,NM),V1(NM),V2(NM),WI(NM),COND(NM)
      DIMENSION R1(2),R2(2)
      DATA TOL/1.E-30/
C ------------------------------------------------------------------
      I = 1
500   IF (I.GT.N) GOTO 590
      VALR = A(I,I)
      VALI = WI(I)
      VALI2 = VALI*VALI
C NJ GIVES EIGENVECTOR TYPE, 0 FOR COLUMN, 1 FOR ROW
      NJ = 0
C     INITIALIZE NONZERO ELEMENTS OF EIGENVECTOR (V1,V2)
      V1(I) = 1.0
      V2(I) = 0.0
505   J = I - 1 + 2*NJ
      IF (VALI.EQ.0.0) GOTO 510
      V2(I+1) = VALI/A(I,I+1)
      V1(I+1) = 0.0
      IF (NJ.EQ.1) V2(I+1) = 1.0/V2(I+1)
      J = I - 1 + 3*NJ
C
C FIND THE INDICES OF ELEMENTS COMPUTED SO FAR
C
510   KS = J + 1 + NJ*(I-J-1)
      KF = I + 1 + NJ*(J-I-2)
      IF (VALI.EQ.0.0.AND.NJ.EQ.0) KF = KF - 1
C
C TEST FOR COMPLETION OF EIGENVECTOR
C
      IF ((J+NJ.LT.1).OR.(J+NJ.GT.N+1)) GOTO 560
C
```

The same section of program (the J loop) computes the column and the row eigenvector for the Ith eigenvalue. J, which always points to the block D, decreases for the column eigenvector (NJ = 1) and increases for the row eigenvector, as shown in the following diagram:

NJ = 0



NJ = 1



The J loop computes first the column eigenvector and then the row eigenvector.

lines 505-1	We always give values to V2 even when only V1 is needed.
lines 505 and after	Initial J = $\left\{\begin{matrix} I-1 & (NJ=0) \\ I+1 & (NJ=1) \end{matrix}\right\}$ unless VALI = u $\neq$ 0 (complex eigenvalue), in which case J = $\left\{\begin{matrix} I-1 & (NJ=0) \\ I+2 & (NJ=1) \end{matrix}\right\}$.
line 505+2	If VALI $\neq$ 0, initialize V as in Section 3.4.
lines 510	The lower limit KS = $\left\{\begin{matrix} J+1 & (NJ=0) \\ I & (NJ=1) \end{matrix}\right\}$; the upper limit KF = $\left\{\begin{matrix} I+1 & (NJ=0) \\ J-1 & (NJ=1) \end{matrix}\right\}$, unless E is 1 by 1 and NJ = 0, in which case KF := KF - 1 = I. See equations 3.2-1, 3.2-2, and comments to line 560.
line 510+3	V is completely computed if for NJ = 0, J < 1, i.e. NJ + J < 1 or if for NJ = 1, J > N, i.e. J + NJ > N + 1.

```
C
C***********************************************************************
C*SOLVE -D*V + V*E = R FOR V = (V1,V2).  D IS A DIAGONAL BLOCK IN ROWS *
C* J1,J2, AND E IS THE REAL CANONICAL FORM OF THE ITH EIGENVALUE.      *
C* EITHER D OR E OR BOTH CAN BE  1 BY 1                                *
C***********************************************************************
C FIND J1 AND J2 (J1.LE.J2) FOR ALL CASES
C
      JJ = J
      IF (WI(J).NE.0.0) JJ = J - 1 + 2*NJ
      J0 = NJ*(J-JJ)
      J1 = JJ + J0
      J2 = J - J0
      D1 = VALR - A(J,J)
C
C CALCULATE RIGHT HAND SIDE R
C
      DO 530  L = J1,J2
      LJ = L - J1 + 1
      R1(LJ) = R2(LJ) = 0.0








            IF ( VALI.NE.0.0) GOTO 520
            DO 515 K = KS,KF
            LK = NJ*(K-L)
            AA = A(L+LK,K-LK)
 515        R1(LJ) = R1(LJ) + AA*V1(K)
      GOTO 530
 520        DO 525 K = KS,KF
            LK = NJ * (K-L)
            AA = A(L+LK,K-LK)
            R1(LJ) = R1(LJ) + AA*V1(K)
 525        R2(LJ) = R2(LJ) + AA*V2(K)
 530  CONTINUE
C
      IF (JJ.NE.J) GOTO 545
C
```

lines 510+4 The pair {J,JJ} is the same as the pair {J1,J2}. However $J1 \leq J2$ whereas $J \geq JJ$ when NJ = 0 and $J \leq JJ$ when NJ = 1. By this device D is transposed when NJ = 1 as required by Section 3.2.

We need

$J1 = J2 = JJ = J$ when D is 1×1,

$J1 = JJ = J-1$, $J2 = J$ when D is 2×2 and NJ = 0,

$J1 = J$, $J2 = JJ = J+1$ when D is 2×2 and NJ = 1.

This is achieved without IF statements by utilizing J0.

lines 515 In order to avoid repetition of a condition, two inner DO loops are used, and VALI need only be tested in the outer loop. If E is 1 by 1, (VALI = 0.0), R is computed from the first inner DO loop, i.e., only R1(LJ) is computed (since V is real). If E is 2 by 2, the second inner DO loop computes R1(LJ) and R2(LJ). If D is 1 by 1, J1 = J2; hence LJ = 1. If D is 2 by 2, $J1 \neq J2$, and LJ = 1, 2. KS and KF, the indices of the previously computed elements, are correctly set for the two cases. It is only necessary to reverse the indices of A: for NJ = 0, LK = 0, AA = A(L+LK,K-LK) = A(L,K). For NJ = 1, LK = K - L, AA = A(L+LK,K-LK) = A(K,L). See equations 3.2-1, 3.2-2.

```
C
C******************************D IS 1 BY 1 ********************************
C
      IF (VALI.NE.0.0) GOTO 535
C   E IS 1 BY 1 (D IS 1 BY 1)
C
      IF (ABS(D1).LT.TOL*ABS(R1(1))) GOTO 585
      V1(J) = V2(J) = 0.0
      IF (D1.NE.0.0) V1(J) = R1(1)/D1
      GOTO 540
C
C    E IS 2 BY 2 ( D IS 1 BY 1 )
C
535   DEN = D1*D1 + VALI2
      VAL = VALI*(-1.0)**NJ
      V1(J) = R1(1)*D1 + R2(1)*VAL
      V2(J) = R2(1)*D1 - R1(1)*VAL
      VMAX = AMAX1(ABS(V1(J)),ABS(V2(J)))
      IF (DEN.LT.TOL*VMAX) GOTO 585
      V1(J) = V1(J)/DEN
      V2(J) = V2(J)/DEN
C  NEXT J
540   J = J - 1 + 2*NJ
      GOTO 510
C
C
C*********************************D IS 2 BY 2*****************************
C
545   IF (VALI.NE.0.0) GOTO 550
C
C   E IS 1 BY 1 (D IS 2 BY 2)
C
      DEN = D1*D1 + WI(J)**2
      V2(J1) = V2(J2) = 0.0
      V1(J1) = R1(1)*D1 + R1(2)*A(JJ,J)
      V1(J2) = R1(1)*A(J,JJ) + R1(2)*D1
      VMAX = AMAX1(ABS(V1(J1)),ABS(V1(J2)))
      IF (DEN.LT.TOL*VMAX) GOTO 585
      V1(J1) = V1(J1)/DEN


      V1(J2) = V1(J2)/DEN
      GOTO 555
C
C E IS 2 BY 2 (D IS 2 BY 2).  CLOSED FORM SOLUTION
C
550   B = A(JJ,J)
      C = A(J,JJ)
      VAL = VALI*(-1.0)**NJ
      BXC = B*C
      H = D1*D1 + VALI2 - BXC
      E = D1 *H
      F = VAL*(H + 2.0*BXC)
      G = H - 2.0*VALI2
      H = 2.0*D1*VAL
      V1(J1) = R1(1)*E + R2(1)*F + R1(2)*B*G + R2(2)*B*H
      V2(J1) = -R1(1)*F + R2(1)*E - R1(2)*B*H + R2(2)*B*G
      V1(J2) = R1(1)*C*G + R2(1)*C*H + R1(2)*E + R2(2)*F
      V2(J2) = -R1(1)*C*H + R2(1)*C*G - R1(2)*F + R2(2)*E
      VMAX = AMAX1(ABS(V1(J1)),ABS(V2(J1)),ABS(V1(J2)),ABS(V2(J2)))
      DEN = G*G + H*H
      IF (DEN.LT.TOL*VMAX) GOTO 585
      IF (DEN.EQ.0.0) GOTO 555
      V1(J1) = V1(J1)/DEN
      V2(J1) = V2(J1)/DEN
      V1(J2) = V1(J2)/DEN
      V2(J2) = V2(J2)/DEN
C
C
```

line 535-5	If VALI $\neq$ 0, E is 2 by 2.
line 535-3	Since E is 1 by 1, V2(J) is set to zero.
lines 535-4	Since there is a strict LT, the defective case (GOTO 585) holds for ABS(D1) = 0, ABS(R1) $\neq$ 0. If both are zero, the less than condition does not hold, and the special zero solution is chosen. (See Section 3.2, type 5).
line 535+1	The sign of VAL depends on NJ (see equation 3.2-3).
line 535+5	DEN > 0 (DEN is set at line 535), since VALI $\neq$ 0. Hence special solution does not occur.
line 545+1	DEN is again greater than zero.
line 545+2	Since E is 1 by 1, V2(J1) and V2(J2) are set to zero.
lines 545+3	Because of the special definition of J and JJ, we have for

lines 550

$$\begin{aligned} NJ = 0:\ & A(JJ,J) = A(J-1,J) = D(1,2) = \beta \\ & A(J,JJ) = A(J,J-1) = D(2,1) = \gamma \\ NJ = 1:\ & A(JJ,J) = A(J+1,J) = D(2,1) = \gamma \\ & A(J,JJ) = A(J,J+1) = D(1,2) = \beta \end{aligned}$$

See equations 3.2-2 and 3.2-4.

line 550+16	If DEN = 0.0 = VMAX, we go to 555, skipping the lines where V is set. But from line 550+13, we see that VMAX = 0.0 implies V1(J1) = V1(J2) = V2(J1) = V2(J2) = 0 (the special solution).

```
C
C NEXT J
555    J = J - 2 + 4*NJ
       GOTO 510
C
C COMPUTE EIGENVECTOR NORM
C
560    VMAX = 0.0
       DO 565 K = KS,KF
565    VMAX = VMAX + V1(K)**2 + V2(K)**2
       IF (NJ.EQ.1) GOTO 570
C
C PREPARE TO COMPUTE ROW EIGENVECTOR
C
       NJ = 1
       CNORM2 = VMAX
       GOTO 505
C
C COMPUTE CONDITION NUMBER
C
570    COND(I) = SQRT(CNORM2*VMAX)
575    IF (VALI.EQ.0.0) GOTO 580
       COND(I) = COND(I)/2.0
       COND(I+1) = COND(I)
       I = I+1
C NEXT I
```

```
580    I = I+1
       GOTO 520
C
C DEFECTIVE CASE
C
585    COND(I) = 1.0/TOL
       GOTO 575
C
C PLACE REAL PART OF EIGENVALUE IN V2
C
590    DO 595 I = 1,N
595    V2(I) = A(I,I)
       RETURN
       END
```

line 560 Verification of the correct index limits KS and KF for computation of ‖V‖:

NJ = 0, I ≠ 1: For the last computation of V1(J1), etc., before NJ is set to 1, J = 2 (D is 2 by 2) or 1 (D is 1 by 1). After J is incremented, J = 0. Then KS = J+1 = 1, KF = I+1 (E is 2 by 2) or I (E is 1 by 1), and the vector is complete.

NJ = 0, I = 1: Initialization sets J = 0, hence KS = 1, KF = 2 or 1, i.e., only the initialized elements are summed.

NJ = 1, WI(N) = 0, I < N: For the last computation J = N. After J is incremented, J = N+1. Hence KS = I, KF = J-1 = N.

NJ = 1, WI(N) = 0, I = N: Initialization gives J = I+1 = N+1. Hence KS = I = N, KF = J-1 = N.

NJ = 1, WI(N) ≠ 0, I < N-1: For the last computation, J = N-1. After J is incremented, J = N+1, KS = I, KF = J-1 = N.

NJ = 1, WI(N) ≠ 0, I = N-1: Initialization gives J = I-1+3 = N+1. Hence KS = I = N-1, KF = J-1 = N. Incrementation of I gives I = N+1, and the program ends.

line 565 If E is 1 by 1, V2(K) for K = KS,KS+1,...,KF was set to zero when the equation was solved.

line 575+1 See 3.4-1 for explanation of halving of cond when E is 2 by 2.

7. RESULTS

The matrix $\underset{\sim}{L}^o$ descripted in Figure 1 came (in punched card form) from a large industrial company. It was causing their eigenvalue program to fail.

An inspection of the form of $\underset{\sim}{L}^o$ suggests that perhaps the strange diagonal element in $\underset{\sim}{I}^o$ and the discordant sign of the (1,1) element of $\underset{\sim}{T}_1$ were key punch errors. So let us consider the matrices resulting from the removal of these anomalies.

$$\underset{\sim}{L} = \begin{pmatrix} \underset{\sim}{0} & \underset{\sim}{X} \\ \underset{\sim}{I} & \underset{\sim}{0} \end{pmatrix}, \quad \underset{\sim}{M} = \begin{pmatrix} \underset{\sim}{0} & \underset{\sim}{Y} \\ \underset{\sim}{I} & \underset{\sim}{0} \end{pmatrix}$$

$$\underset{\sim}{Y} = 10^8 \begin{pmatrix} \underset{\sim}{D} & \underset{\sim}{T}_1^- & \underset{\sim}{0} \\ \underset{\sim}{T}_2 & \underset{\sim}{T}_1 & -\underset{\sim}{F}_2 \\ \underset{\sim}{0} & -\underset{\sim}{F}_3 & \underset{\sim}{F}_4 \end{pmatrix},$$

where $\underset{\sim}{T}_1^-$ is obtained from $\underset{\sim}{T}_1$ by reversing the sign of its (1,1) element.

Notice that $\underset{\sim}{L}$'s eigenvalues are the square roots of $\underset{\sim}{X}$'s:

$$\begin{pmatrix} \underset{\sim}{0} & \underset{\sim}{X} \\ \underset{\sim}{I} & \underset{\sim}{0} \end{pmatrix} \begin{pmatrix} \underset{\sim}{u} \\ \underset{\sim}{v} \end{pmatrix} = \lambda \begin{pmatrix} \underset{\sim}{u} \\ \underset{\sim}{v} \end{pmatrix} \longleftrightarrow \underset{\sim\sim}{Xv} = \lambda^2 \underset{\sim}{v}, \quad \underset{\sim}{u} = \lambda \underset{\sim}{v} \quad .$$

The eigenvalues of L^o, L and M are given in Table 1 and we offer the following comments. Every eigenvalue of L^o is moderately ill-conditioned and the zero pair appear to belong to a quadratic elementary divisor (only one eigenvector). Perhaps some of this is due to the unbalanced nature of L^o. The thirteenth row of L^o is null and this must be permuted out of the way before the rest of the matrix is balanced.

Figure 1

24 × 24 Matrix for Case Study

Is the 0 diagonal element in $\underset{\sim}{I}^o$ a keypunch error or did it really belong in the user's problem?

$$\underset{\sim}{L}^o = \begin{pmatrix} \underset{\sim}{0} & \underset{\sim}{X} \\ \underset{\sim}{I}^o & \underset{\sim}{0} \end{pmatrix} ; \qquad \underset{\sim}{I}^o = \mathrm{diag}(0,1,1,\ldots,1);$$

$$\underset{\sim}{X} = \begin{pmatrix} \underset{\sim}{D} & \underset{\sim}{T}_1 & \underset{\sim}{0} \\ \underset{\sim}{T}_2 & \underset{\sim}{F}_1 & -\underset{\sim}{F}_2 \\ \underset{\sim}{0} & -\underset{\sim}{F}_3 & \underset{\sim}{F}_4 \end{pmatrix} 10^8 ; \qquad \underset{\sim}{F}\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & 0 & 0 & b \\ 0 & a & -b & 0 \\ 0 & -c & d & 0 \\ c & 0 & 0 & d \end{pmatrix} ;$$

$$\underset{\sim}{D} = \mathrm{diag}(.5221, .3563, .5552 \times 10^{-3}, .1328) ;$$

$$\underset{\sim}{T}_1 = \begin{pmatrix} .5221 & 0 & 0 & -.8951 \\ 0 & -.3563 & .6109 & 0 \\ 0 & 0 & -.5552\times10^{-3} & 0 \\ 0 & 0 & 0 & -.1328 \end{pmatrix} ;$$

$$\underset{\sim}{T}_2 = \begin{pmatrix} -.0976 & 0 & 0 & 0 \\ 0 & -.0666 & 0 & 0 \\ 0 & .02659 & -.4218\times10^{-4} & 0 \\ -.03896 & 0 & 0 & -.1009\times10^{-1} \end{pmatrix}$$

$$\underset{\sim}{F}_1 = \begin{pmatrix} 2.859 & 0 & 0 & 1.079 \\ 0 & 2.828 & -1.026 & 0 \\ 0 & -.2389 & .4294 & 0 \\ .2513 & 0 & 0 & .4607 \end{pmatrix} ;$$

$$\underset{\sim}{F}_2 = \underset{\sim}{F}\begin{pmatrix} 2.761 & .5891 \\ .2123 & .3590 \end{pmatrix} ; \quad \underset{\sim}{F}_3 = \underset{\sim}{F}\begin{pmatrix} 1.627 & .5373 \\ .1096 & .2868 \end{pmatrix} ; \quad \underset{\sim}{F}_4 = \underset{\sim}{F}\begin{pmatrix} 1.849 & .3731 \\ .1178 & .4970 \end{pmatrix} ;$$

$$\|\underset{\sim}{L}^o\|_\infty \div 5 \times 10^8 .$$

Table 1

Eigenvalues and Condition Numbers of L^o, L, M

The imaginary pair of eigenvalues had real parts less than 10^{-6} (a relative error of 10^{-11}). V denotes a digit that changed when the matrix was balanced.

$\lambda_i(L^o)$	Cond(L^o)	Cond$(\hat{L}^o)$ (balanced)	$\lambda_i(L)$	Cond(L)	$\lambda_i(M)$	Cond(M)
$\pm .21534594 \times 10^5$	10^4	1	$\pm .21534594 \times 10^5$	10^4	$\pm .21534594 \times 10^5$	10^4
$\pm .18667890 \times 10^5$	10^4	10^3	$\pm .18654343 \times 10^5$	10^4	$\pm .18692513 \times 10^5$	10^4
$\pm .11076317 \times 10^5 i$	6×10^3	4×10^3	$\pm .1098663V \times 10^5 i$	6×10^3	$\pm .11142610 \times 10^5 i$	6×10^3
$\pm .82614552 \times 10^4$	6×10^3	5×10^3	$\pm .8646568V \times 10^4$	10^4	$\pm .86339960 \times 10^4 *$	10^4
$\pm .83281998 \times 10^4$	6×10^3	1	$\pm .83281998 \times 10^4$	6×10^3	$\pm .83281998 \times 10^4$	6×10^3
$\pm .41438248 \times 10^4$	7×10^3	10^4	$\pm .7026706V \times 10^4$	10^4	$\pm .71883679 \times 10^4 *$	10^4
$\pm .64209960 \times 10^4$	7×10^3	3	$\pm .64209960 \times 10^4$	6×10^3	$\pm .64209960 \times 10^4$	7×10^3
$\pm .33632953 \times 10^4$	5×10^3	9×10^3	$\pm .38448590 \times 10^4$	3×10^3	$\pm .38458962 \times 10^4 *$	3×10^3
$\pm .35102700 \times 10^4$	2×10^3	4	$\pm .35102700 \times 10^4$	2×10^3	$\pm .35102700 \times 10^4$	2×10^3
$\pm .30530592 \times 10^4$	3×10^3	3	$\pm .30530592 \times 10^4$	3×10^3	$\pm .30530592 \times 10^4$	3×10^3
$\pm .0$[†]	10^{15}	10^{30}	$\pm .29241979 \times 10^4$	3×10^3	$\pm .29134631 \times 10^4$	3×10^3
$\pm .23559292 \times 10^3$	10^2	1	$\pm .23559291 \times 10^3$	10^2	$\pm .23559292 \times 10^3$	10^2

[†]The unbalanced matrix L^o had a negative eigenvalue -2×10^{-8} instead of -0.

The result was that none of the computed eigenvalues changed but half of them became almost perfectly conditioned.

In fact we can say that the ill-condition of all six pairs is due to the zero element in position (13,1). When this is replaced by 1 we obtain the matrix L which has six pairs of eigenvalues almost identical to the well conditioned pairs of the balanced L^o. Four of the other six pairs are changed completely, the remaining two ($\pm .186 \times 10^5$ and $\pm .11 \times 10^5 i$) are substantially altered. Interestingly the balanced versions of L and M are almost normal and we have not bothered to record the condition numbers. The six pairs of eigenvalues which were unchanged by the move from L^o to L were also invariant in the change from L to M. The other six pairs had relative errors less than 2.5%.

We can tell in advance what the balanced form of L and M will be:

$$\hat{L} = \begin{pmatrix} 0 & 10^{-4}X \\ 10^4 I & 0 \end{pmatrix}, \quad \hat{M} = \begin{pmatrix} 0 & 10^{-4}Y \\ 10^4 I & 0 \end{pmatrix}.$$

The change from L^o to L is tiny relative to $\|L^o\|$ ($\doteq 10^{-8}\|L^o\|$) but the change from $\hat{L}^o$ to $\hat{L}$ is approximately $\|\hat{L}\|$.

We conclude that the suspicious element in L^o was probably a key punch error. Concerning the (1,1) element of T_1 we cannot say, both L and M are reasonable matrices and indeed the change of sign does not affect the leading two decimals in any eigenvalue.

8. BIBLIOGRAPHY

1. EISPACK GUIDE, Lecture Notes in Computer Science No. 6, Springer-Verlag (1974).

2. IMSL - International Mathematical and Statistical Library, 7500 Bellaire Blvd., Houston, Texas 77036.

3. W. Kahan, "Conserving Confluence Curbs Ill-Condition," Technical Report No. 6, Computer Science Department, University of California, Berkeley (1972).

4. B.N. Parlett and Y. Wang, "The Influence of the Compiler on the Cost of Mathematical Software - in Particular on the Cost of Triangular Factorization," ACM Transactions on Mathematical Software, Vol. 1, No. 1, March 1975, pp. 35-46.

5. J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford (1975).

6. J.H. Wilkinson, C. Reinsch, Handbook for Automatic Computation, II. Linear Algebra, Springer-Verlag (1971).

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Computer Science Division University of California Berkeley, California 94720	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

A PROGRAM TO COMPUTE THE CONDITION NUMBERS OF MATRIX EIGENVALUES WITHOUT COMPUTING EIGENVECTORS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Scientific Final

5. AUTHOR(S) *(First name, middle initial, last name)*

S.P. Chan, R. Feldman and B.N. Parlett

6. REPORT DATE April 1975	7a. TOTAL NO. OF PAGES 46	7b. NO. OF REFS 6

8a. CONTRACT OR GRANT NO. ONR-N00014-69-A-0200-1017	9a. ORIGINATOR'S REPORT NUMBER(S) Electronics Research Laboratory Memorandum M-517
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Mathematics Branch Office of Naval Research Washington, D.C. 20360

13. ABSTRACT

The condition number of an eigenvalue measures the sensitivity of that eigenvalue to small changes in the matrix elements. Such extra information is nice, sometimes useful, but how much does it cost?

A program is presented here for the most difficult case of a real square matrix whose eigenvalues are wanted without their corresponding eigenvectors. The program requires no extra storage space (this is our reason for presenting it) and the running time is about 50% longer than for the fastest reliable program which only computes eigenvalues.

There are many industrial applications in which the matrix elements are known to only two or three decimal figures. Each condition number will indicate how accurately such a matrix determines the associated eigenvalue. When no digits in an eigenvalue are reliable the suspect eigenvalue should be tagged and this information passed on to a higher level in the whole computation.

A number of programming devices keep the code, storage, and running time down to a minimum.

An interesting case study is included.

Unclassified
Security Classification

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
eigenvalue						
condition number						

DD FORM 1 NOV 65 1473 (BACK)
(PAGE 2)

Unclassified
Security Classification