The Calculation of the Eigenvectors of a General Complex Matrix by Inverse Iteration

By J. M. Varah*

Introduction. In this paper, we are concerned with finding approximations to the eigenvectors of a general complex $n \times n$ matrix A, assuming good approximations to the eigenvalues have been found. In Section 1, we describe the method used, which amounts to one step of the well-known inverse iteration technique for a particular initial vector. We show that the eigenvector approximation thus obtained is as accurate as can be expected using single-precision arithmetic. An Algol 60 program using this method for the case of a general real matrix is given in the microfiche section, and the details of the program and an example are given in Section 2. In a second paper [7], we will describe a technique which gives rigorous a posteriori error bounds for such an approximate eigensystem, and which, applied repeatedly, can yield a better approximate eigensystem if the matrix A is known to more than single precision.

1. The Basic Iteration. We are concerned with finding the column eigenvectors of a general complex $n \times n$ matrix A, that is, the columns of a matrix X so that $AX = X\Lambda$, where Λ is the diagonal matrix of eigenvalues $\{\lambda_i\}_1^n$ of A. We assume the following:

(i) A has been scaled so that $\|A\|_2 = 1$, for convenience.

(ii) A has been reduced by a similarity transformation to upper Hessenberg form, i.e. $a_{ij} = 0$ for $j \leqq i - 2$. This reduction is usually done by Householder transformations or by elementary row and column operations. Both are discussed in Wilkinson [6, Chapter 6].

(iii) approximations $\{\lambda_i'\}_1^n$ to the eigenvalues of A have been found, and are such that each is an exact eigenvalue of a slightly different matrix $A + E_\lambda$. The most popular eigenvalue method lately has been the QR method, which is known to give such approximate eigenvalues. This and other methods, including a Laguerre iteration developed by Parlett [3] are discussed in Chapters 7 and 8 of Wilkinson [6].

To find the eigenvectors, the inverse iteration technique has been very successful; namely we perform the following iteration for each eigenvalue approximation λ:

$$(1) \qquad (A - \lambda I)y^{(k)} = y^{(k-1)}, \qquad k = 1, 2, \cdots,$$

with $y^{(0)}$ some arbitrary vector. The iteration is performed by first forming the LU decomposition of $(A - \lambda I)$ using Gaussian elimination. That is, we form a unit lower triangular matrix L and an upper triangular matrix U such that $LU = P(A - \lambda I) + F$, where P is a permutation matrix and F is a matrix of rounding errors. Then the two resulting triangular systems are solved.

It is our purpose here to show how the inverse iteration technique can provide

Received August 25, 1967. Revised February 23, 1968.

* Supported in part by the Office of Naval Research under contract Nonr-225(37) (NR-044-211).

as good an eigenvector approximation as can be expected. In fact, since A is generally not represented exactly in the machine and because of rounding errors, all we can expect of an eigenvector approximation is that it can be an exact eigenvector of some matrix $A + E$, with $\|E\|_2 \leqq c\eta_1$, where c is close to 1 and

$$\eta_1 = \max \{\eta\colon fl(1 + \eta) = 1\} .$$

Here $fl(q)$ denotes the floating-point machine-calculated value of the arithmetic expression q.

THEOREM 1. *If at any step in the iteration* (1), *we find*

$$\frac{\|y^{(k)}\|_2}{\|y^{(k-1)}\|_2} > \frac{1}{r_3}\eta_1^{-1} ,$$

then the computed vector $y^{(k)}$ is the exact eigenvector of a matrix $A + E$, where $\|E\|_2 \leqq r\eta_1$, $r = r_1 + r_2 + r_3$, and r_1 and r_2 are small machine constants.

Proof. In the machine computation, since we use Gaussian elimination with partial pivoting, we know from Wilkinson [5, p. 98] that $P^{-1}LU = A - \lambda I + F$, where $\|F\|_2 \leqq r_1\eta_1$. Moreover, in solving the triangular systems obtained, we know from Wilkinson [5, p. 102] that

$$(P^{-1}LU + G_k)y^{(k)} = y^{(k-1)} ,$$

where $\|G_k\|_2 \leqq r_2\eta_1$. Here r_1 and r_2 depend on the machine arithmetic. Thus we have exactly

$$(A - \lambda I + F + G_k)y^{(k)} = y^{(k-1)} .$$

Now let $u = y^{(k)}/\|y^{(k)}\|_2$ and $v = y^{(k-1)}/\|y^{(k)}\|_2$. Then the above gives

$$(A - \lambda I + F + G_k - vu^*)u = 0 .$$

Taking $E = F + G - vu^*$ gives

$$\|E\|_2 \leqq (r_1 + r_2)\eta_1 + \|y^{(k-1)}\|_2/\|y^{(k)}\|_2 < r\eta_1 .$$

This completes the proof of the theorem.

So one strategy would be to use the basic iteration (1) and iterate until $\|y^{(k)}\|_2/\|y^{(k-1)}\|_2$ is close to η_1^{-1}. Wilkinson [6, Chapters 5 and 9] uses this technique, with $y^{(0)} = P^{-1}Le$ and $e = (1, 1, \cdots, 1)^T$. He iterates until the norm ratio is close to η_1^{-1} and then does one more step. This strategy has two faults: first of all, we cannot guarantee that we will obtain a norm ratio close to η_1^{-1}. In fact, one can give examples where the iterates increase in norm by only $\eta_1^{-1/n}$ at each step. In practice however, such a large norm ratio is usually obtained on the first iteration. But even so, for matrices with poorly conditioned eigenvalues, if we continue to iterate, the norm ratios are much less than η_1^{-1} on each subsequent iteration, so even the "one more step" is not a good idea. We take a different viewpoint: we can guarantee a large norm on the very first iteration by choosing the initial vector properly.

THEOREM 2. *Let λ be an approximate eigenvalue of A with $(A - \lambda I + F_1)$ singular and $\|F_1\|_2 = r \cdot \eta_1$, and let $P^{-1}LU$ be the LU decomposition of $(A - \lambda I)$. Let G be a matrix of linearly independent columns $\{g_i\}_1^n$ and consider solving $UZ = G$*

for the columns z_i of Z. Suppose $\|z_k\|_2 \geqq \|z_i\|_2$, $i \neq k$, and let $y^{(1)}$ be the computed solution to $Uz_k = g_k$. Then $y^{(1)}$ satisfies

$$\|y^{(0)}\|_2/\|y^{(1)}\|_2 \leqq K\cdot\eta_1\,,$$

with

$$K = n\sqrt{2}\cdot\mathrm{cond}_2(G)\cdot\left[\left(\frac{n+1}{\sqrt{2}}\right)(r_1 + r) + r_2\right],$$

where r_1 and r_2 are given in Theorem 1. *Thus $y^{(1)}$ is an exact eigenvector of $(A + E)$ with $\|E\|_2 \leqq (K + r_1 + r_2)\eta_1$.*

Proof. Since $(A - \lambda I + F_1)$ is singular and $A - \lambda I = P^{-1}LU - F$, with $\|F\|_2 \leqq r_1\cdot\eta_1$, then

$$(P^{-1}LU - F + F_1)v = 0$$

for some vector v. Since L and U are both nonsingular,

$$v = U^{-1}L^{-1}P(F - F_1)v\,,$$

giving

$$\|U^{-1}\|_2^{-1} \leqq \|P\|_2\|L^{-1}\|_2(r + r_1)\eta_1\,.$$

Now since A is in upper Hessenberg form, L has only one nonzero element in each column below the diagonal, and because partial pivoting is used, this element is less than or equal to one in magnitude. Let $L_{k_j,j}$ be the nonzero off-diagonal element in the jth column. Then solving for the ith row of L^{-1} (whose elements are denoted by L_{ij}^{-1}),

$$|L_{ij}^{-1}| = \left|\sum_{k=j+1}^{i} L_{ik}^{-1}L_{kj}\right| \leqq |L_{i,k_j}^{-1}\cdot L_{k_j,j}|\,, \qquad j = i-1, i-2, \cdots, 1\,.$$

We claim $|L_{ij}^{-1}| \leqq 1$ for all i, j. To see this, fix i. We proceed by induction on $j = i - 1$, $i - 2$, $\cdots$, 1, using the above equation. Now clearly $|L_{i,i-1}^{-1}| = |L_{i,i-1}| \leqq 1$ and $L_{ii}^{-1} = 1$. For $j < i - 1$,

$$|L_{ij}^{-1}| \leqq |L_{i,k_j}^{-1}|\cdot 1 \leqq 1$$

using the induction hypothesis, since $k_j > j$. Thus

$$\|L^{-1}\|_2 \leqq (n(n+1)/2)^{1/2} < (n+1)/\sqrt{2}\,,$$

and

$$\|U^{-1}\|_2^{-1} \leqq ((n+1)/\sqrt{2})(r + r_1)\eta_1\,.$$

Now we are solving $UZ = G$ and hence

$$\|Z\|_2 \geqq \|U^{-1}\|_2/\|G^{-1}\|_2\,.$$

Let the columns of Z be z_i, $i = 1, \cdots, n$, and suppose $\|z_k\|_2 \geqq \|z_i\|_2$, $i \neq k$. Then

$$\|z_k\|_2 \geqq \frac{1}{\sqrt{n}}\|Z\|_2 \geqq \frac{1}{\sqrt{n}}\frac{\|U^{-1}\|_2}{\|G^{-1}\|_2}\,.$$

Now let our iterate $y^{(1)}$ be the computed approximation to z_k, that is, the machine approximation we would obtain solving for the right-hand side g_k. Then $y^{(1)}$ satisfies $(U + G_1)y^{(1)} = g_k$, with $\|G_1\|_2 \leqq r_2\eta_1$, as in Theorem 1. Hence

$$\frac{1}{\|y^{(1)}\|_2} \leqq \frac{\|I + U^{-1}G_1\|_2}{\|z_k\|_2} \leqq \sqrt{n}\|G^{-1}\|_2(\|U^{-1}\|_2^{-1} + r_2\eta_1) .$$

When we solve for $y^{(1)}$ in this way, our initial vector $y^{(0)} = P^{-1}Lg_k$, so that

$$\|y^{(0)}\|_2 \leqq \|P^{-1}\|_2\|L\|_2 \cdot \|G\|_2 \leqq (2n)^{1/2}\|G\|_2 ;$$

and hence

$$\frac{\|y^{(0)}\|_2}{\|y^{(1)}\|_2} \leqq n\sqrt{2}(\|G\|_2 \cdot \|G^{-1}\|_2) \cdot \left[\left(\frac{n+1}{\sqrt{2}}\right)(r + r_1) + r_2\right]\eta_1 = K \cdot \eta_1 .$$

Finally, applying Theorem 1, we see that $y^{(1)}$ is an exact eigenvector of $(A + E)$, with $\|E\|_2 \leqq (K + r_1 + r_2)\eta_1$. This completes the proof of Theorem 2.

In effect, this just says that since $(A - \lambda I)$ is very nearly singular, the near-singularity will be reflected in an inverse iterate with a large norm, for some initial vector. So in theory we could take any nonsingular G and solve for each column. But this takes too much work: solving for every column takes n^3 operations. However, we do not have to find the column which gives the maximum solution norm, only one where the norm is large. So we want to use a G in practice that gives a large norm for the very first column in most cases.

As Wilkinson noticed in [4], taking $G = I$ can be very poor. Here we are just solving for the columns of U^{-1}, and many of the matrices which arise have no small diagonal elements in U although it is nearly singular. So it may be necessary to solve for several columns of U^{-1} to find one with a large norm. Wilkinson has shown that the initial vector $e = (1, 1, \cdots, 1)^T$ gives good results in practice; actually any such vector with lots of nonzero components would probably work as well. However, it can happen that $z = U^{-1}e$ is not large in norm either; we still must have other choices available.

The strategy we propose is to take for G the orthogonal matrix

$$g_{ij} = \cos\frac{2\pi(i-1)(j-1)}{n} + \sin\frac{2\pi(i-1)(j-1)}{n} . \tag{2}$$

The first column is exactly the vector e and in practice this almost always gives a solution with a large norm. If it does not however, we go on and try the second column, etc. Also note that $G^TG = nI$ so that $\mathrm{cond}_2\,(G) = 1$ and thus this gives the best possible bound in Theorem 2.

2. Description of the Program. The Algol 60 program, M-9, in the microfiche section herein, applies to a *real* matrix A reduced to upper Hessenberg form. The reason for this is twofold: first, for an Algol compiler that allows complex declarations, the code for a general complex matrix is simpler than that given here because only the (a) iteration below is necessary. Secondly, if the matrix is real, time is saved by using a specialized program. For best results, the input matrix A should be "balanced" so that its ith row and column sums are roughly equal. This can be done by diagonal similarity transformations (see Osborne [2]). Also, the matrix should be scaled so that $\|A\|_2$ is close to 1.0.

```
EIGENVALUE[1] =  2.0198988648@+01   EIGENVECTOR IS:
    1.0000000000@+00   9.5049257081@-01   3.5885440618@-01  -1.2947598931@-01  -2.8295959779@-01  -2.1767080856@-01
   -1.0883427067@-01  -3.8788232741@-02  -9.9274025056@-03  -1.7546995870@-03  -1.9338839202@-04  -1.0072898815@-05

EIGENVALUE[2] =  3.2228891501@+01   EIGENVECTOR IS:
    1.0000000000@+00   9.6897193936@-01   5.9759795218@-01   2.7840244571@-01   1.0288340823@-01   3.0584831848@-02
    7.2899354677@-03   1.3698147455@-03   1.9634925155@-04   2.0246135087@-05   1.3409273260@-06   4.2924337063@-08

EIGENVALUE[3] =  6.9615330859@+00   EIGENVECTOR IS:
    4.8032108884@-01   4.1132463613@-01  -4.0672169692@-01  -9.3915106538@-01  -2.7842787684@-01   8.4081357526@-01
    1.0000000000@+00   1.3167381711@-01  -6.0547326737@-01  -5.9415708167@-01  -2.4788610192@-01  -4.1580932136@-02

EIGENVALUE[4] =  1.2311077399@+01   EIGENVECTOR IS:
    1.0000000000@+00   9.1877234075@-01  -4.9361637825@-02  -7.9164937359@-01  -6.9125977538@-01  -1.2067975736@-01
    2.8216867121@-01   3.1806397598@-01   1.7762888070@-01   5.9858132974@-02   1.1710862360@-02   1.0353443655@-03

EIGENVALUE[5] =  3.5118559487@+00   EIGENVECTOR IS:
    4.7498360229@-02   3.3973215412@-02  -1.2447724100@-01  -1.8577101736@-01   1.8613096422@-01   5.5631618640@-01
    2.6900127471@-02  -9.3122494800@-01  -7.0435792319@-01   5.5687197343@-01   1.0000000000@+00   3.9811200183@-01

EIGENVALUE[6] =  6.4350525615@-01   EIGENVECTOR IS:
    7.1312649043@-06  -3.9506309847@-06  -1.1971237326@-04   1.2771168673@-04   1.6035343992@-03  -2.4760405037@-03
   -1.6071423302@-02   3.1989825659@-02   1.0715205688@-01  -2.5820837688@-01  -3.5649474385@-01   1.0000000000@+00

EIGENVALUE[7] =  1.5539887133@+00   EIGENVECTOR IS:
   -9.1010872164@-04  -3.2444889580@-04   6.3265935123@-03   4.3432428434@-03  -3.5092426109@-02  -3.4869462223@-02
    1.4564436171@-01   1.8655354663@-01  -4.0210925855@-01  -6.2354260941@-01   5.5398871333@-01   1.0000000000@+00

EIGENVALUE[8] =  3.0810080871@-02   EIGENVECTOR IS:
   -6.4047556693@-10   1.9357384498@-08  -3.7946706035@-07   5.6540521781@-06  -6.7012054873@-05   6.3988766210@-04
   -4.9038588375@-03   2.9647587789@-02  -1.3680097240@-01   4.5425950922@-01  -9.6918991910@-01   1.0000000000@+00

EIGENVALUE[9] =  4.9966341158@-02   EIGENVECTOR IS:
    4.1785424075@-10  -8.4131677178@-09   6.8441983137@-08   3.8244724534@-07  -1.9599506013@-05   3.1142189090@-04
   -3.1754323545@-03   2.2980195187@-02  -1.1917616946@-01   4.2629880587@-01  -9.5003365882@-01   1.0000000000@+00

EIGENVALUE[10] =  8.0922601311@-02   EIGENVECTOR IS:
   -9.8100456316@-10   1.0871466573@-08   1.0148002296@-08  -1.4586958825@-06   1.5438485371@-05  -3.1135873949@-05
   -9.8184089832@-04   1.3459812276@-02  -9.2204212922@-02   3.8189033172@-01  -9.1907739866@-01   1.0000000000@+00

EIGENVALUE[11] =  1.4371453177@-01   EIGENVECTOR IS:
    6.2148609788@-09  -3.7160031784@-08  -2.5415059753@-07   4.0999735649@-06  -8.5126562425@-06  -1.7750831715@-04
    1.4722685561@-03  -1.3612559882@-03  -4.3111291413@-02   2.9475513567@-01  -8.5628546822@-01   1.0000000000@+00

EIGENVALUE[12] =  2.8474589251@-01   EIGENVECTOR IS:
   -1.1905377896@-07   2.9901088099@-07   3.8481130750@-06  -2.0167059978@-05  -7.0970102545@-05   7.4486811840@-04
   -1.2635568886@-04  -1.5378034943@-02   4.0846879300@-02   1.1342127288@-01  -7.1525410747@-01   1.0000000000@+00
```

An Algol 60 program for the reduction to upper Hessenberg form using Householder transformations is given in [1] and a similar program using elementary similarity operations is forthcoming in *Numerische Mathematik*. An excellent program for calculating the eigenvalues by the QR method is forthcoming by Wilkinson in *Numerische Mathematik*, and the Algol 60 code for the Laguerre iteration of Parlett is given in [3].

The eigenvectors of the Hessenberg matrix are found as follows:

(a) if the eigenvalue approximation λ is real, we form the LU decomposition of $(A - \lambda I)$ and solve $Uy^{(k)} = g_k$, $k = 1, 2, \cdots$, where the g_k are the columns of G, defined by (2). We continue until $\|y^{(k)}\|_2 > K\eta_1^{-1}$, where K is a fixed tolerance. The process requires about $n^2(k + 1)/2$ multiplications;

(b) if $\lambda = \xi + i\nu$, $\nu \neq 0$, we proceed as in Wilkinson [6, p. 630]. Since $\bar{\lambda}$ is also an eigenvalue approximation, we form the LU decomposition of

$$B = (A - \lambda I)(A - \bar{\lambda} I)$$

and solve

$$Uy^{(k)} = g_k, \qquad z^{(k)} = -\frac{1}{\nu}(A - \xi I)y^{(k)}, \qquad k = 1, 2, \cdots.$$

We continue until $\|y^{(k)} + iz^{(k)}\|_2 > K\eta_1^{-1}$. Since the matrix A^2 is formed only once, the work per eigenvalue is about $n^2(k + 11/6)/2$ multiplications.

If any zero pivots are encountered in the LU decomposition, they are replaced by $\eta_1 \max_{i,j} |a_{ij}|$, so that we can backsolve. If any eigenvalue approximations are identical, they are perturbed by an amount close to η_1 so that a different LU is formed. Then if there are distinct eigenvectors corresponding to these eigenvalues, the two inverse iterations should give two distinct eigenvectors.

This program has been tested on scores of matrices using the Burroughs B5500 at Stanford University. We give one example:

$$\begin{aligned} a_{ij} &= n + 1 - \max(i, j), \quad \text{if } j \geqq i - 1 \\ &= 0, \quad \text{otherwise} \end{aligned} \quad \text{and} \quad n = 12.$$

This is the Frank matrix discussed by Wilkinson ([6, p. 92] and [5, p. 151]). It is in upper Hessenberg form and its eigenvalues are all real and positive. However, the Wilkinson conditions of the eigenvalues deteriorate as we go from biggest to smallest, and are about 10^{-7} for the last three eigenvalues. The eigenvalue approximations were found using an early version of Wilkinson's QR program on the B5500 where $\eta_1 = \frac{1}{2} 8^{-12}$. The smallest three eigenvalue approximations are only correct to about two significant figures; nevertheless each is an exact eigenvalue of a slightly different matrix, as are the eigenvector approximations and each of the latter was found on the first inverse iteration.

Acknowledgment. The author would like to thank Professor G. E. Forsythe for his supervision and guidance, and Dr. J. H. Wilkinson for his many helpful suggestions.

Computer Science Department
Stanford University
Stanford, California 94305

1. D. J. Mueller, "Householder's method for complex matrices and eigensystems of hermitian matrices," *Numer. Math.*, v. 8, 1966, pp. 72–92. MR **33** #872.
2. E. E. Osborne, "On pre-conditioning of matrices," *J. Assoc. Comput. Mach.*, v. 7, 1960, pp. 338–345. MR **26** #892.
3. B. N. Parlett, "Laguerre's method applied to the matrix eigenvalue problem," *Math. Comp.*, v. 18, 1964, pp. 464–485. MR **29** #2948.
4. J. H. Wilkinson, "The calculation of the eigenvectors of codiagonal matrices," *Comput. J.*, v. 1, 1958, pp. 90–96.
5. J. H. Wilkinson, *Rounding Errors in Algebraic Processes*, Notes on Applied Science No. 32, HMSO, London; Prentice-Hall, Englewood Cliffs, N. J., 1963. MR **28** #4661.
6. J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965. MR **32** #1894.
7. J. M. Varah, "Rigorous machine bounds for the eigensystem of a general complex matrix," *Math. Comp.*, v. 22, 1968, pp. 793–801.

```
          N SO THAT ITS COMPONENT OF LARGEST MAGNITUDE IS 1.0, AND SETS
          VNORM EQUAL TO THE MAGNITUDE OF THE LARGEST COMPONENT BEFORE
          NORMALIZATION;
     REAL S,S1,V1; INTEGER I,J;
     S:=0;
     FOR I:=1 STEP 1 UNTIL N DO
     BEGIN
          S1:=ABS(V[I]);
          IF S1>S THEN
          BEGIN
               S:=S1; J:=I
          END
     END;
     VNORM:=S;
     IF VNORM ≠ 0 THEN
     BEGIN
          V1:=V[J]; V[J]:=1;
          FOR I:=1 STEP 1 UNTIL J-1,J+1 STEP 1 UNTIL N DO
               V[I]:=V[I]/V1
     END
END NORMREAL;

PROCEDURE NORMCOMPLEX(N,U,V,VNORM);
     VALUE N;
     INTEGER N; ARRAY U,V; REAL VNORM;
BEGIN COMMENT THIS PROCEDURE NORMALIZES THE COMPLEX VECTOR U+VI OF
          DIMENSION N SO THAT ITS COMPONENT OF LARGEST MAGNITUDE IS 1+0I,
          AND SETS VNORM EQUAL TO THE MAGNITUDE OF THE LARGEST COMPONENT
          BEFORE NORMALIZATION;
     INTEGER I,J; REAL S,S1,U1,U2,V1,V2,R,DEN;
     S:=0;
     FOR I:=1 STEP 1 UNTIL N DO
     BEGIN
          S1:=ABSC(U[I],V[I]);
          IF S1>S THEN
          BEGIN
               S:=S1; J:=I
          END
     END;
     VNORM:=S;
     IF VNORM≠0 THEN
     BEGIN
          U1:=U[J]; V1:=V[J];
          U[J]:=1; V[J]:=0;
          IF ABS(U1)≥ABS(V1) THEN
          BEGIN
               R:=V1/U1; DEN:=U1+R×V1;
               FOR I:=1 STEP 1 UNTIL J-1,J+1 STEP 1 UNTIL N DO
               BEGIN
                    U2:=U[I]; V2:=V[I];
                    U[I]:=(U2+R×V2)/DEN; V[I]:=(V2-R×U2)/DEN
               END
          END ELSE
          BEGIN
               R:=U1/V1; DEN:=V1+R×U1;
          FOR I:=1 STEP 1 UNTIL J-1,J+1 STEP 1 UNTIL N DO
          BEGIN
             U2:=U[I]; V2:=V[I];
             U[I]:=(V2+R×U2)/DEN; V[I]:=(R×V2-U2)/DEN
          END
       END
    END OF CONDITIONAL
END NORMCOMPLEX;

PROCEDURE GAUSSM(N,M,A,EPS,PIVOT);
    VALUE N,M,EPS;
    INTEGER M,N; ARRAY A; REAL EPS; INTEGER ARRAY PIVOT;
BEGIN COMMENT THIS PROCEDURE REDUCES THE N×N MATRIX A,HAVING M
       SUB-DIAGONALS,TO LU FORM BY GAUSSIAN ELIMINATION WITH
       INTERCHANGES. ANY ZERO PIVOTAL ELEMENTS ARE REPLACED BY EPS;
    INTEGER I,J,K,IMAX; REAL T,SUM,QUOT;
    FOR K:=1 STEP 1 UNTIL N DO
    BEGIN
       SUM:=0;
       FOR I:=K STEP 1 UNTIL MIN(K+M,N) DO
       BEGIN
          T:=ABS(A[I,K]);
          IF T>SUM THEN
          BEGIN
             SUM:=T; IMAX:=I
          END
       END;
       IF SUM=0 THEN
       BEGIN COMMENT K-TH COLUMN IS ZERO - REPLACE DIAGONAL ELEMENT
          BY EPS;
          A[K,K]:=EPS; IMAX:=K
       END;
       PIVOT[K]:=IMAX;
       IF IMAX ≠ K THEN
       FOR J:=1 STEP 1 UNTIL N DO
       BEGIN COMMENT INTERCHANGE ROWS IMAX AND K;
          T:=A[K,J]; A[K,J]:=A[IMAX,J]; A[IMAX,J]:=T
       END;
       FOR I:=K+1 STEP 1 UNTIL MIN(K+M,N) DO
       BEGIN
          A[I,K]:=QUOT:=A[I,K]/A[K,K];
          FOR J:=K+1 STEP 1 UNTIL N DO
             A[I,J]:=A[I,J]-QUOT×A[K,J]
       END
    END K
END GAUSSM;

PROCEDURE SOLVE(N,A,X,Y,PIVOT);
    VALUE N; INTEGER N;
    ARRAY A,X,Y; INTEGER ARRAY PIVOT;
BEGIN COMMENT THIS PROCEDURE SOLVES U×Y=X GIVEN A IN LU FORM.
       THE SOLUTION Y IS TESTED FOR OVERFLOW AND SCALED DOWN IF
       OVERFLOW HAS OCCURRED;
    INTEGER I,K;
    FOR K:=N STEP -1 UNTIL 1 DO
```

```
   BEGIN
SCALE:
      OVERFLOW:=FALSE; COMMENT THIS IS A MACHINE DEPENDENT DEVICE,
         WHICH OPERATES LIKE A BOOLEAN VARIABLE AND IS SET TO TRUE
         BY THE MACHINE IF OVERFLOW OCCURS;
      Y[K]:=-INNERPRODUCT(I,K+1,N,A[K,I],Y[I],-X[K])/A[K,K];
      IF OVERFLOW THEN
      BEGIN COMMENT SCALE DOWN ITERATE;
         FOR I:=K+1 STEP 1 UNTIL N DO Y[I]:=Y[I]×MACHEPS;
         GO TO SCALE
      END
   END K
END SOLVE;

COMMENT NOW BEGIN MAIN PROCEDURE HESSVECTORS;

   INTEGER I,J,K,M,P,ITNS;
   REAL ANORM,RABS2,NORM,VNORM,RR,RI,EPS,NORMTOLERANCE,T,C,C1,C2,
      S,S1,S2,RR2;
   ARRAY B[1:N,1:N],X,UT,VT[1:N];
   INTEGER ARRAY PIVOT[1:N];
   BOOLEAN ASQCALC;

   ASQCALC:=FALSE;
   T:=6.28318530718/N; COMMENT 2×PI/N;
   C:=COS(T); S:=SIN(T);
   COMMENT NOW FIND MAXIMUM ROW SUM OF A;
   ANORM:=0;
   FOR I:=1 STEP 1 UNTIL N DO
   BEGIN
      T:=0;
      FOR J:=MAX(1,I-1) STEP 1 UNTIL N DO T:=T+ABS(A[I,J]);
      ANORM:=MAX(ANORM,T)
   END;
   EPS:=ANORM×MACHEPS;
   IF EPS=0 THEN EPS:=MACHNEGL; COMMENT SMALLEST NORMALIZED POS. NO.;
   COMMENT EPS IS USED IN PROCEDURE GAUSSM IN PLACE OF A ZERO PIVOT;
   NORMTOLERANCE:=1/(100×N×EPS);

   COMMENT NOW FIND EIGENVECTORS;
   M:=1; COMMENT SO THE INCREMENT VARIABLE IN THE FOR LOOP HAS A
      VALUE ON ENTRY;
   FOR K:=1 STEP M UNTIL N DO
   BEGIN
      COMMENT FIRST PERTURB EIGENVALUE IF IDENTICAL TO SOME PREVIOUS;
      J:=0;
      FOR I:=1 STEP 1 UNTIL K-1 DO
         IF RTR[I]=RTR[K] ∧ RTI[I]=RTI[K] THEN J:=J+1;
      RR:=RTR[K]+3×J×MACHEPS×MAX(1,ABS(RTR[K]));
      RI:=RTI[K];
      IF RI≠0 ∧ (¬ ASQCALC) THEN
      BEGIN COMMENT COMPUTE A↑2 WHEN FIRST COMPLEX EIGENVALUE
            ENCOUNTERED;
         FOR I:=1 STEP 1 UNTIL N DO
            FOR J:=1 STEP 1 UNTIL N DO
```

```
        ASQ[I,J]:=INNERPRODUCT(P,MAX(I-1,1),MIN(J+1,N),
                                    A[I,P],A[P,J],0);
   ASQCALC:=TRUE
END;

COMMENT NOW GENERATE B, THE MATRIX TO BE REDUCED;
IF RI=0 THEN
BEGIN COMMENT REAL EIGENVALUE - B=A-RR×IDENTITY;
   M:=1;
   FOR I:=1 STEP 1 UNTIL N DO
   BEGIN
      FOR J:=1 STEP 1 UNTIL I-2 DO B[I,J]:=0;
      FOR J:=MAX(I-1,1) STEP 1 UNTIL N DO
         B[I,J]:=A[I,J];
      B[I,I]:=B[I,I]-RR
   END
END ELSE
BEGIN COMMENT COMPLEX EIGENVALUE -
      B=(A-(RR+RI×I)×IDENTITY)×(A-(RR-RI×I)×IDENTITY);
   M:=2; RABSQ:=RR↑2+RI↑2; RR2:=2×RR;
   FOR I:=1 STEP 1 UNTIL N DO
   BEGIN
      FOR J:=1 STEP 1 UNTIL I-3 DO B[I,J]:=0;
      IF I≥3 THEN B[I,I-2]:=ASQ[I,I-2];
      FOR J:=MAX(I-1,1) STEP 1 UNTIL N DO
         B[I,J]:=ASQ[I,J]-RR2×A[I,J];
      B[I,I]:=B[I,I]+RABSQ
   END
END GENERATION OF B;

COMMENT NOW REDUCE B TO LU FORM - M GIVES THE NUMBER OF
   SUBDIAGONALS OF B;
GAUSSM(N,M,B,EPS,PIVOT);
C1:=1; S1:=0;
FOR I:=1 STEP 1 UNTIL N DO X[I]:=1;
FOR ITNS:=1 STEP 1 UNTIL N DO
BEGIN
   SOLVE(N,B,X,UT,PIVOT);
   IF RI=0 THEN
      NORMREAL(N,UT,VNORM)
   ELSE
   BEGIN
      COMMENT CALCULATE VT FROM (A-RR×IDENTITY)×UT+RI×VT=0;
      FOR I:=1 STEP 1 UNTIL N DO
         VT[I]:=-INNERPRODUCT(J,MAX(I-1,1),N,A[I,J],UT[J],
                                    -RR×UT[I])/RI;
      NORMCOMPLEX(N,UT,VT,VNORM)
   END;
   IF VNORM ≥ NORMTOLERANCE THEN GO TO BIGNORM;
   COMMENT IF NOT, GENERATE NEW INITIAL VECTOR USING ORTHOGONAL
      COSINE VECTORS;
   T:=C1×C-S1×S; S1:=S1×C+C1×S; C1:=T;
   X[1]:=C2:=1; S2:=0;
   FOR J:=2 STEP 1 UNTIL N DO
   BEGIN T:=C2×C1-S2×S1;
```

```
            S2:=S2×C1+C2×S1;
            C2:=T; X[J]:=C2+S2
         END
      END ITNS;

      COMMENT ITERATE DID NOT CONVERGE - SET VECTOR = 0;
      FOR I:=1 STEP 1 UNTIL N DO U[I,K]:=0;
      IF KI≠0 THEN FOR I:=1 STEP 1 UNTIL N DO U[I,K+1]:=0;
      GO TO FIN;
BIGNORM:
      COMMENT ITERATE HAS CONVERGED - STORE IN U;
      FOR I:=1 STEP 1 UNTIL N DO U[I,K]:=UT[I];
      IF KI≠0 THEN FOR I:=1 STEP 1 UNTIL N DO U[I,K+1]:=VT[I];
FIN:
   END K
END HESSVECTORS;
```

```
PROCEDURE EIGENSYSTEMBOUNDS(N,A,LAMBDA,X,DELTA,BETA,T,Y,LAMBDAIMP,
   EIGENVALUERADIUS,XIMP,EIGENVECTORRADIUS,POORINVERSE);
   VALUE N,BETA,DELTA,T;
   INTEGER N,T,BETA; REAL DELTA; BOOLEAN POORINVERSE;
   LONG COMPLEX ARRAY A,LAMBDA,X,Y,LAMBDAIMP,XIMP;
   LONG REAL ARRAY EIGENVALUERADIUS,EIGENVECTORRADIUS;
BEGIN COMMENT EIGENSYSTEMBOUNDS FINDS IMPROVED ESTIMATES AND BOUNDS
   FOR THE EIGENSYSTEM OF THE N×N COMPLEX MATRIX A, GIVEN A FULL SET
   OF APPROXIMATE EIGENVALUES IN LAMBDA, AND A FULL SET OF CORRESPON-
   DING APPROXIMATE EIGENVECTORS IN X. DELTA IS THE MAXIMUM RELATIVE
   ERROR IN THE ELEMENTS OF A REPRESENTED IN THE MACHINE, BETA IS THE
   NUMBER BASE OF THE MACHINE, AND T IS THE NUMBER OF BASE BETA DIGITS
   IN EACH FLOATING POINT NUMBER. Y IS THE OUTPUT APPROXIMATE INVERSE
   FOR X. THE IMPROVED APPROXIMATIONS TO THE EIGENVALUES OF A ARE
   STORED IN LAMBDAIMP, WITH THE BOUND FOR THE DISTANCE BETWEEN THESE
   NUMBERS AND THE TRUE EIGENVALUES IN EIGENVALUERADIUS. SIMILARLY THE
   IMPROVED APPROXIMATIONS TO THE EIGENVECTORS ARE STORED BY COLUMNS
   IN XIMP, WITH THE BOUNDS IN THE CORRESPONDING ELEMENTS OF
   EIGENVECTORRADIUS. POORINVERSE IS SET TO TRUE IF THE MATRIX X COULD
   NOT BE INVERTED, OR IF THE CALCULATED INVERSE Y IS SO POOR THAT THE
   RESIDUAL MATRIX E=I-X×Y HAS L-INFINITY NORM LARGER THAN 1.0.
      THE LANGUAGE USED IS STANDARD ALGOL 60 EXCEPT FOR THE ADDITION
   OF COMPLEX AND LONG (MEANING DOUBLE PRECISION) DECLARATIONS. WE
   ASSUME THAT THE FUNCTION ABS IS DEFINED FOR COMPLEX ARGUMENTS AND
   GIVES THE MODULUS;

COMMENT FIRST DECLARE OTHER PROCEDURES;

REAL PROCEDURE MAX(A,B);
   VALUE A,B; REAL A,B;
   MAX:=IF A>B THEN A ELSE B;

REAL PROCEDURE MIN(A,B);
   VALUE A,B; REAL A,B;
   MIN:=IF A<B THEN A ELSE B;

COMPLEX PROCEDURE ROUND(A);
   VALUE A; LONG COMPLEX A;
BEGIN COMMENT THE BODY OF ROUND IS LEFT UNDEFINED. THE USER MUST
   INSERT A PROGRAM WHICH ROUNDS A LONG COMPLEX VARIABLE TO SINGLE
   PRECISION;
END ROUND;

PROCEDURE INVERSE(N,X,Y,E,SINGULAR);
   VALUE N; INTEGER N; COMPLEX ARRAY X,Y,E; LABEL SINGULAR;
BEGIN COMMENT THE BODY OF INVERSE IS LEFT UNDEFINED. THE USER MUST
   INSERT A PROGRAM WHICH FINDS A SINGLE PRECISION APPROXIMATE INVERSE
   FOR X AND STORES IT IN Y LEAVING X UNCHANGED. THE PROGRAM MUST
   ALSO COMPUTE THE RESIDUAL MATRIX E = I-XY USING DOUBLE PRECISION
   ACCUMULATION OF INNER PRODUCTS. FOR THE USUAL METHOD OF GAUSSIAN
   ELIMINATION WITH ITERATIVE IMPROVEMENT, E IS THE LAST RESIDUAL
   MATRIX COMPUTED BY THE ITERATIVE IMPROVEMENT. WE ASSUME THE PROGRAM
   TRANSFERS TO LABEL SINGULAR IF X IS TOO SINGULAR TO INVERT;
END INVERSE;
```

```
COMMENT NOW BEGIN MAIN PROCEDURE EIGENSYSTEMBOUNDS;
REAL S1,S2,S3,S4,S5,AMAX,YSUM,EMAXROWSUM,ETA1,ETA2,EPS1,EPS2,
   FACTOR,RADIUS,RADIISUM,USUM;
LONG REAL MAXCOMP; COMPLEX SC1; LONG COMPLEX SCD1;
INTEGER I,J,K,M,JMAX;
REAL ARRAY Q,PQ[1:N,1:N],YROWSUM,XCOLSUM,FMAXCOL,PQROWSUM,ROOTDISTANCE,
   ABSBOUND,DISTERROR,UBND,VBND,YEROWSUM[1:N];
COMPLEX ARRAY F,P[1:N,1:N],ROOTDIFFERENCE,U[1:N];
LONG COMPLEX ARRAY V[1:N];

COMMENT FIRST CALCULATE BASIC ROUNDING ERRORS;
ETA1:=1.06×2×BETA↑(1-T); COMMENT THIS IS A BOUND FOR THE ROUNDING ERROR
   MADE IN EACH OF THE FOLLOWING REAL SINGLE PRECISION OPERATIONS:
   ADD,SUBTRACT,MULTIPLY,DIVIDE,SQUARE ROOT. IT IS MULTIPLIED BY 1.06
   TO PERMIT EASIER CALCULATION OF ACCUMULATED ERROR. THIS ROUNDING
   ERROR BOUND IS PESSIMISTIC, AND COULD PROBABLY BE DECREASED FOR
   THE ACTUAL MACHINE USED;
ETA2:=1.06×2×BETA↑(1-2×T); COMMENT A BOUND FOR THE CORRESPONDING REAL
   DOUBLE PRECISION OPERATIONS;
EPS1:=N×ETA1; COMMENT USED IN THE BOUND FOR A REAL SINGLE PRECISION
   INNER PRODUCT OF N TERMS;
EPS2:=(N+0.5)×ETA2; COMMENT USED IN THE BOUND FOR A COMPLEX DOUBLE
   PRECISION INNER PRODUCT OF N TERMS;
POORINVERSE:=FALSE;
COMMENT NOW FORM APPROXIMATE X-INVERSE;
INVERSE(N,X,Y,F,SINGULAR);

COMMENT NOW FIND ROW SUMS OF Y AND Y×E,COLUMN SUMS OF X, MAXIMUM
   ELEMENT OF A, AND MAXIMUM ROW SUM OF E, WHERE E=I-XY IS STORED IN F;
AMAX:=YSUM:=EMAXROWSUM:=0;
FOR I:=1 STEP 1 UNTIL N DO
BEGIN S1:=S2:=S3:=S4:=0;
   FOR J:=1 STEP 1 UNTIL N DO
   BEGIN
      S1:=S1+ABS(Y[I,J]);
      S2:=S2+ABS(X[J,I]);
      S3:=S3+ABS(F[I,J]);
      SCD1:=0;
      FOR K:=1 STEP 1 UNTIL N DO
         SCD1:=SCD1+Y[I,K]×F[K,J]; COMMENT COMPLEX DOUBLE PRECISION
            OPERATION;
      S4:=S4+ABS(ROUND(SCD1));
      AMAX:=MAX(AMAX,ABS(A[I,J]))
   END J;
   YROWSUM[I]:=S1×(1+EPS1+3×ETA1);
   YSUM:=YSUM+YROWSUM[I];
   YEROWSUM[I]:=S4;
   XCOLSUM[I]:=S2×(1+EPS1+3×ETA1);
   IF S3 > EMAXROWSUM THEN EMAXROWSUM := S3
END I;
AMAX:=AMAX×(1+5×ETA1);
YSUM:=YSUM×(1+EPS1);
EMAXROWSUM:=(EMAXROWSUM×(1+EPS1+4×ETA1) + (EPS2+ETA2)×YSUM + ETA2)
         ×(1+5×ETA1);
FOR I:=1 STEP 1 UNTIL N DO
```

```
   YEROWSUM[I]:=(YEROWSUM[I]×(1+EPS1+4×ETA1) + YROWSUM[I]
                 ×((EPS2+ETA2)×YSUM+(ETA1+EPS2)×EMAXROWSUM+ETA2))
                ×(1+6×ETA1);
IF EMAXROWSUM ≥ 1 THEN
BEGIN COMMENT THE CALCULATED INVERSE Y IS SUCH A POOR INVERSE FOR X
      THAT OUR BOUND FOR THE L-INFINITY NORM OF THE RESIDUAL MATRIX
      IS LARGER THAN 1.0;
   POORINVERSE:=TRUE;
   GO TO FIN
END;

COMMENT NOW FORM RESIDUAL F=A×X-X×(DIAG(LAMBDA));
FOR I:=1 STEP 1 UNTIL N DO
BEGIN S1:=0; SC1:=LAMBDA[I];
   FOR J:=1 STEP 1 UNTIL N DO
   BEGIN SCD1:=0;
      FOR K:=1 STEP 1 UNTIL N DO
         SCD1:=SCD1+A[J,K]×X[K,I]; COMMENT COMPLEX DOUBLE PRECISION
            OPERATION;
      SCD1:=SCD1-X[J,I]×SC1; COMMENT COMPLEX DOUBLE PRECISION
         OPERATION;
      F[J,I]:=ROUND(SCD1);
      S1:=MAX(S1,ABS(F[J,I]))
   END J;
   FMAXCOL[I]:=S1×(1+4×ETA1)
END I;

COMMENT NOW FORM P=Y×F;
FOR I:=1 STEP 1 UNTIL N DO
BEGIN
   FOR J:=1 STEP 1 UNTIL N DO
   BEGIN SCD1:=0;
      FOR K:=1 STEP 1 UNTIL N DO
         SCD1:=SCD1+Y[I,K]×F[K,J]; COMMENT COMPLEX DOUBLE PRECISION
            OPERATION;
      P[I,J]:=ROUND(SCD1)
   END J;
   LAMBDAIMP[I]:=LAMBDA[I]+P[I,I]; COMMENT COMPLEX DOUBLE PRECISION
      OPERATION;
END I;

COMMENT NOW CALCULATE Q, THE MATRIX BOUNDING THE ERROR IN P;
S1:=1-EMAXROWSUM-(1+EMAXROWSUM)×ETA1;
S2:=AMAX×(ETA2+EPS2+DELTA);
S3:=3×ETA2;
FOR I:=1 STEP 1 UNTIL N DO
BEGIN S4:=EPS2×FMAXCOL[I]+(ETA1×FMAXCOL[I]+S2×XCOLSUM[I]
                          +S3×ABS(LAMBDA[I])×(1+4×ETA1))/S1;
   S5:=FMAXCOL[I]/S1;
   FOR J:=1 STEP 1 UNTIL N DO
      Q[J,I]:=(ETA1×ABS(P[J,I])×(1+4×ETA1)+YROWSUM[J]×S4
               +YEROWSUM[J]×S5) × (1+10×ETA1);
   Q[I,I]:=Q[I,I]+ETA2×(ABS(LAMBDA[I])+ABS(P[I,I]))
END I;
```

```
COMMENT NOW FIND MATRIX PQ = ABS(P)+Q AND ITS OFF-DIAGONAL ROW SUMS;
FOR I:=1 STEP 1 UNTIL N DO
BEGIN S1:=0;
   FOR J:=1 STEP 1 UNTIL I-1,I+1 STEP 1 UNTIL N DO
   BEGIN PQ[I,J]:=(ABS(P[I,J])+Q[I,J])×(1+5×ETA1);
      S1:=S1+PQ[I,J]
   END;
   PQROWSUM[I]:=S1×(1+EPS1)
END;

COMMENT NOW BOUND EIGENVALUES AND EIGENVECTORS - I IS THE INDEX OF THE
   ROOT CONSIDERED;
FOR I:=1 STEP 1 UNTIL N DO
BEGIN COMMENT FIRST FIND DISTANCES BETWEEN ROOT I AND REST;
   FOR J:=1 STEP 1 UNTIL I-1,I+1 STEP 1 UNTIL N DO
   BEGIN ROOTDIFFERENCE[J]:=ROUND(LAMBDAIMP[I]-LAMBDAIMP[J]);
         COMMENT COMPLEX DOUBLE PRECISION OPERATION;
      DISTERROR[J]:=(ETA1+ETA2)×(ABS(ROUND(LAMBDAIMP[I]))
         +ABS(ROUND(LAMBDAIMP[J])))×(1+8×ETA1);
      ROOTDISTANCE[J]:=ABS(ROOTDIFFERENCE[J])
   END;
   COMMENT NOW BOUND THE I-TH EIGENVALUE USING THE GERSCHGORIN THEOREM.
      THE SMALLEST ISOLATED DISC IS FOUND USING A DIAGONAL SIMILARITY
      TRANSFORMATION WITH MULTIPLIER (BETA↑M). FIRST FIND THE
      APPROXIMATE EXPONENT M;
   IF I=1 THEN K:=2 ELSE K:=1;
   S1:=IF ROOTDISTANCE[K]=0 THEN 0 ELSE ROOTDISTANCE[K]/PQ[K,I];
   FOR J:=K+1 STEP 1 UNTIL I-1,I+K STEP 1 UNTIL N DO
      S1:=MIN(S1,IF ROOTDISTANCE[J]=0 THEN 0
                 ELSE ROOTDISTANCE[J]/PQ[J,I]);
   IF S1<1 THEN M:=0 ELSE M:=ENTIER(LN(S1)/LN(BETA));
   FACTOR:=BETA↑M;

   COMMENT NOW MAKE SURE I-TH DISC IS ISOLATED;
NEWDISC:
   RADIUS:=(PQROWSUM[I]/FACTOR+Q[I,I])×(1+ETA1);
   FOR J:=1 STEP 1 UNTIL I-1,I+1 STEP 1 UNTIL N DO
   BEGIN
      RADIISUM:=(4×ETA1×ROOTDISTANCE[J]+DISTERROR[J]+Q[J,J]+PQROWSUM[J]
         +RADIUS+FACTOR×PQ[J,I])×(1+7×ETA1);
      IF ROOTDISTANCE[J]≤RADIISUM THEN
      BEGIN COMMENT DISC IS NOT ISOLATED - REDUCE M IF POSSIBLE;
         IF M>0 THEN
         BEGIN M:=M-1;
            FACTOR:=FACTOR/BETA;
            GO TO NEWDISC
         END ELSE
         BEGIN EIGENVALUERADIUS[I]:=-1;
            COMMENT DISCS I AND J CANNOT BE SEPARATED-HERE ONE COULD
               OUTPUT THIS FACT AND GIVE THE OVERLAPPING DISCS:
               DISCI=ABS(LAMBDA-LAMBDA[I])<RADIUS,
               DISCJ=ABS(LAMBDA-LAMBDA[J])<(RADIISUM-RADIUS);
            FOR K:=1 STEP 1 UNTIL N DO XIMP[K,I]:=X[K,I];
            GO TO ENDVECTOR
         END
   YEROWSUM[I]:=(YEROWSUM[I]×(1+EPS1+4×ETA1) + YROWSUM[I]
                 ×((EPS2+ETA2)×YSUM+(ETA1+EPS2)×EMAXROWSUM+ETA2))
                ×(1+6×ETA1);
IF EMAXROWSUM ≥ 1 THEN
BEGIN COMMENT THE CALCULATED INVERSE Y IS SUCH A POOR INVERSE FOR X
      THAT OUR BOUND FOR THE L-INFINITY NORM OF THE RESIDUAL MATRIX
      IS LARGER THAN 1.0;
   POORINVERSE:=TRUE;
   GO TO FIN
END;

COMMENT NOW FORM RESIDUAL F=A×X-X×(DIAG(LAMBDA));
FOR I:=1 STEP 1 UNTIL N DO
BEGIN S1:=0; SC1:=LAMBDA[I];
   FOR J:=1 STEP 1 UNTIL N DO
   BEGIN SCD1:=0;
      FOR K:=1 STEP 1 UNTIL N DO
         SCD1:=SCD1+A[J,K]×X[K,I]; COMMENT COMPLEX DOUBLE PRECISION
            OPERATION;
      SCD1:=SCD1-X[J,I]×SC1; COMMENT COMPLEX DOUBLE PRECISION
         OPERATION;
      F[J,I]:=ROUND(SCD1);
      S1:=MAX(S1,ABS(F[J,I]))
   END J;
   FMAXCOL[I]:=S1×(1+4×ETA1)
END I;

COMMENT NOW FORM P=Y×F;
FOR I:=1 STEP 1 UNTIL N DO
BEGIN
   FOR J:=1 STEP 1 UNTIL N DO
   BEGIN SCD1:=0;
      FOR K:=1 STEP 1 UNTIL N DO
         SCD1:=SCD1+Y[I,K]×F[K,J]; COMMENT COMPLEX DOUBLE PRECISION
            OPERATION;
      P[I,J]:=ROUND(SCD1)
   END J;
   LAMBDAIMPL[I]:=LAMBDA[I]+P[I,I]; COMMENT COMPLEX DOUBLE PRECISION
      OPERATION;
END I;

COMMENT NOW CALCULATE Q, THE MATRIX BOUNDING THE ERROR IN P;
S1:=1-EMAXROWSUM-(1+EMAXROWSUM)×ETA1;
S2:=AMAX×(ETA2+EPS2+DELTA);
S3:=3×ETA2;
FOR I:=1 STEP 1 UNTIL N DO
BEGIN S4:=EPS2×FMAXCOL[I]+(ETA1×FMAXCOL[I]+S2×XCOLSUM[I]
                          +S3×ABS(LAMBDA[I])×(1+4×ETA1))/S1;
   S5:=FMAXCOL[I]/S1;
   FOR J:=1 STEP 1 UNTIL N DO
      Q[J,I]:=(ETA1×ABS(P[J,I])×(1+4×ETA1)+YROWSUM[J]×S4
              +YEROWSUM[J]×S5) × (1+10×ETA1);
   Q[I,I]:=Q[I,I]+ETA2×(ABS(LAMBDA[I])+ABS(P[I,I]))
END I;
```

```
    END;
    XTMP[JMAX,I]:=1.0; COMMENT COMPLEX DOUBLE PRECISION OPERATION;
    EIGENVECTORRADIUS[JMAX,I]:=VBND[JMAX]/S1×(1+2×ETA1);
ENDVECTOR:
END I;
GO TO FIN;
SINGULAR:
POORINVERSE:=TRUE; COMMENT THE MATRIX X COULD NOT BE INVERTED;
FIN:
END EIGENSYSTEMBOUNDS;
```

RIGOROUS MACHINE BOUNDS FÒR THE EIGENSYSTEM OF A GENERAL COMPLEX MATRIX

J. M. Varah

See article in this issue for explanation of symbols in table.

```
PROCEDURE EIGENSYSTEMBOUNDS(N,A,LAMBDA,X,DELTA,BETA,T,Y,LAMBDAIMP,
   EIGENVALUERADIUS,XIMP,EIGENVECTORRADIUS,POORINVERSE);
   VALUE N,BETA,DELTA,T;
   INTEGER N,T,BETA; REAL DELTA; BOOLEAN POORINVERSE;
   LONG COMPLEX ARRAY A,LAMBDA,X,Y,LAMBDAIMP,XIMP;
   LONG REAL ARRAY EIGENVALUERADIUS,EIGENVECTORRADIUS;
BEGIN COMMENT EIGENSYSTEMBOUNDS FINDS IMPROVED ESTIMATES AND BOUNDS
   FOR THE EIGENSYSTEM OF THE N×N COMPLEX MATRIX A, GIVEN A FULL SET
   OF APPROXIMATE EIGENVALUES IN LAMBDA, AND A FULL SET OF CORRESPON-
   DING APPROXIMATE EIGENVECTORS IN X. DELTA IS THE MAXIMUM RELATIVE
   ERROR IN THE ELEMENTS OF A REPRESENTED IN THE MACHINE, BETA IS THE
   NUMBER BASE OF THE MACHINE, AND T IS THE NUMBER OF BASE BETA DIGITS
   IN EACH FLOATING POINT NUMBER. Y IS THE OUTPUT APPROXIMATE INVERSE
   FOR X. THE IMPROVED APPROXIMATIONS TO THE EIGENVALUES OF A ARE
   STORED IN LAMBDAIMP, WITH THE BOUND FOR THE DISTANCE BETWEEN THESE
   NUMBERS AND THE TRUE EIGENVALUES IN EIGENVALUERADIUS. SIMILARLY THE
   IMPROVED APPROXIMATIONS TO THE EIGENVECTORS ARE STORED BY COLUMNS
   IN XIMP, WITH THE BOUNDS IN THE CORRESPONDING ELEMENTS OF
   EIGENVECTORRADIUS. POORINVERSE IS SET TO TRUE IF THE MATRIX X COULD
   NOT BE INVERTED, OR IF THE CALCULATED INVERSE Y IS SO POOR THAT THE
   RESIDUAL MATRIX E=I-X×Y HAS L-INFINITY NORM LARGER THAN 1.0.
     THE LANGUAGE USED IS STANDARD ALGOL 60 EXCEPT FOR THE ADDITION
   OF COMPLEX AND LONG (MEANING DOUBLE PRECISION) DECLARATIONS. WE
   ASSUME THAT THE FUNCTION ABS IS DEFINED FOR COMPLEX ARGUMENTS AND
   GIVES THE MODULUS;

COMMENT FIRST DECLARE OTHER PROCEDURES;

REAL PROCEDURE MAX(A,B);
   VALUE A,B; REAL A,B;
   MAX:=IF A>B THEN A ELSE B;

REAL PROCEDURE MIN(A,B);
   VALUE A,B; REAL A,B;
   MIN:=IF A<B THEN A ELSE B;

COMPLEX PROCEDURE ROUND(A);
   VALUE A; LONG COMPLEX A;
BEGIN COMMENT THE BODY OF ROUND IS LEFT UNDEFINED. THE USER MUST
   INSERT A PROGRAM WHICH ROUNDS A LONG COMPLEX VARIABLE TO SINGLE
   PRECISION;
END ROUND;

PROCEDURE INVERSE(N,X,Y,E,SINGULAR);
   VALUE N; INTEGER N; COMPLEX ARRAY X,Y,E; LABEL SINGULAR;
BEGIN COMMENT THE BODY OF INVERSE IS LEFT UNDEFINED. THE USER MUST
   INSERT A PROGRAM WHICH FINDS A SINGLE PRECISION APPROXIMATE INVERSE
   FOR X AND STORES IT IN Y LEAVING X UNCHANGED. THE PROGRAM MUST
   ALSO COMPUTE THE RESIDUAL MATRIX E = I-XY USING DOUBLE PRECISION
   ACCUMULATION OF INNER PRODUCTS. FOR THE USUAL METHOD OF GAUSSIAN
   ELIMINATION WITH ITERATIVE IMPROVEMENT, E IS THE LAST RESIDUAL
   MATRIX COMPUTED BY THE ITERATIVE IMPROVEMENT. WE ASSUME THE PROGRAM
   TRANSFERS TO LABEL SINGULAR IF X IS TOO SINGULAR TO INVERT;
END INVERSE;
```