CNA-32

AN ALGORITHM FOR THE GENERALIZED MATRIX EIGENVALUE PROBLEM $Ax = \lambda Bx$

by

C. B. Moler*
Stanford University

G. W. Stewart**

October, 1971 CNA-32

*Supported in part by an NSF grant at Stanford University and by an ONR contract at the University of Michigan.

**Supported in part by NSF grant GP-23655 at The University of Texas in Austin.

This report is issued jointly as STAN-CS-232-71 at Stanford University.

CENTER FOR NUMERICAL ANALYSIS

THE UNIVERSITY OF TEXAS AT AUSTIN

AN ALGORITHM FOR THE GENERALIZED MATRIX EIGENVALUE PROBLEM $Ax = \lambda Bx$

by

C. B. Moler
Computer Science Department
Stanford University

Department of Mathematics
University of Michigan

and

G. W. Stewart
Departments of Computer Science and Mathematics
University of Texas at Austin

Abstract

A new method, called the QZ algorithm, is presented for the solution of the matrix eigenvalue problem $Ax = \lambda Bx$ with general square matrices A and B. Particular attention is paid to the degeneracies which result when B is singular. No inversions of B or its submatrices are used. The algorithm is a generalization of the QR algorithm, and reduces to it when $B = I$. A Fortran program and some illustrative examples are included.

AN ALGORITHM FOR THE GENERALIZED MATRIX EIGENVALUE PROBLEM $Ax = \lambda Bx$

C. B. Moler[*]

G. W. Stewart[**]

1. Introduction

We shall be concerned with the matrix eigenvalue problem of determining the nontrivial solutions of the equation

$$Ax = \lambda Bx \quad ,$$

where A and B are real matrices of order n. When B is nonsingular this problem is formally equivalent to the usual eigenvalue problem $B^{-1}Ax = \lambda x$.

When B is singular, however, such a reduction is not possible, and in fact the characteristic polynomial $\det(A-\lambda B)$ is of degree less than n, so that there is not a complete set of eigenvalues for the problem. In some cases the missing eigenvalues may be regarded as "infinite". In other cases the entire problem may be poorly posed. The term infinite eigenvalue is justified by the fact that if B is perturbed slightly so that it is no longer singular, there may appear a number of large eigenvalues that grow unboundedly as the perturbation is reduced to zero. However, if $\det(A-\lambda B)$ vanishes identically, say when A and B have a common null space, then any λ may be regarded as an eigenvalue. Such problems have unusually pathological features, and we refer to them as "ill-disposed" problems.

---

[*] Computer Science Department, Stanford University, and Department of Mathematics, University of Michigan.

[**] Departments of Computer Science and Mathematics, University of Texas at Austin.

In numerical work the sharp distinction between singular and non-singular matrices is blurred, and the pathological features associated with singular B carry over to the case of nearly singular B. The object of this paper is to describe an algorithm for computing the eigenvalues and corresponding eigenvectors that is unaffected by nearly singular B. The algorithm, the heart of which we call the QZ-algorithm, is essentially an iterative method for computing the decomposition contained in the following theorem [10].

Theorem. There are unitary matrices Q and Z so that QAZ and QBZ are both upper triangular.

We say that the eigenvalue problems $QAZy = \lambda QBZy$ and $Ax = \lambda Bx$ are unitarily equivalent. The two problems obviously have the same eigenvalues, and their eigenvectors are related by the equation $x = Zy$.

The algorithm proceeds in four stages. In the first, which is a generalization of the Householder reduction of a single matrix to Hessenberg form [4,5], A is reduced to upper Hessenberg form and at the same time B is reduced to upper triangular form. In the second step, which is a generalization of the Francis implicit double shift QR algorithm [3,8], A is reduced to quasi-triangular form while the triangular form of B is maintained. In the third stage the quasi-triangular matrix is effectively reduced to triangular form and the eigenvalues extracted. In the fourth stage the eigenvectors are obtained from the triangular matrices and then transformed back into the original coordinate system.

The transformations used in reducing A and B are applied in such a way that Wilkinson's general analysis of the roundoff errors in unitary

transformations [11] shows that the computed matrices are exactly unitarily equivalent to slightly perturbed matrices $A+E$ and $B+F$. This means that the computed eigenvalues, which are the ratios of the diagonal elements of the final matrices, are the exact eigenvalues of the perturbed problem $(A+E)x = \lambda(B+F)x$. If an eigenvalue is well conditioned in the sense that it is insensitive to small perturbations in A and B (see [10] for a detailed analysis), then it will be computed accurately. This accuracy is independent of the singularity or nonsingularity of B.

The use of unitary transformations in the reduction also simplifies the problem of convergence: a quantity may be set to zero if a perturbation of the same size can be tolerated in the original matrix.

Our computer program does not actually produce the eigenvalues λ_i but instead returns α_i and β_i, the diagonal elements of the triangular matrices QAZ and QBZ. The divisions in $\lambda_i = \alpha_i/\beta_i$ become the responsibility of the program's user. We emphasize this point because the α_i and β_i contain more information than the eigenvalues themselves.

Since our algorithm is an extension of the QR algorithm, the well known properties of the QR algorithm apply to describe the behavior of our algorithm.

In their survey article [9], Peters and Wilkinson describe another approach for the case when B is nearly singular. In their method one computes an approximate null space for B and removes it from the problem. The technique is reapplied to the deflated problem, and so on until a well conditioned problem is obtained. The method has the crucial drawback that

one must determine the rank of B. If a wrong decision is reached, the well-conditioned eigenvalues may be seriously affected.

The special case where A is symmetric and B is positive definite has been extensively treated. For the case of well-conditioned B the "Cholesky-Wilkinson" method [6] enjoys a well deserved popularity. A modification of this algorithm for band matrices is given by Crawford [1]. A variant of the Peters-Wilkinson method for nearly semidefinite B has been given by Fix and Heiberger [2]. Although our method does not preserve symmetry and is consequently more time consuming than these algorithms, its stability may make it preferable when B is nearly semidefinite.

2. Reduction to Hessenberg-Triangular Form

In this section we shall give an algorithm whereby A is reduced to upper Hessenberg form and simultaneously B is reduced to triangular form. While a treatment of the reductions in this and the following sections can be given in terms of standard plane rotations and elementary Hermitian matrices, we find it convenient from a computational point of view to work exclusively with a modified form of the elementary Hermitians. Accordingly, we introduce the following notation.

By $\mathcal{U}_r(k)$ we mean the class of symmetric, orthogonal matrices of the form

$$I + vu^T$$

where $v^T u = -2$, v is a scalar multiple of u, only components $k, k+1, \ldots, k+r-1$ of u are nonzero, and $u_k = 1$. Given any vector x, it is easy to choose a member Q of $\mathcal{U}_r(k)$ so that

$$Qx = x + (u^T x)v$$

has its $k+1, \ldots, k+r-1$ components equal to zero, its k-th component changed and all other components unchanged. Since $u_k = 1$, the computation of Qy for any y requires only $2r-1$ multiplications and $2r-1$ additions. (In particular, use of a matrix in $\mathcal{U}_2$ requires only 3 multiplications instead of the 4 required by a standard plane rotation.)

For the most part, we shall use only matrices in $\mathcal{U}_2$ and $\mathcal{U}_3$. When a matrix Q in $\mathcal{U}_3(k)$ premultiplies a matrix A only rows k, $k+1$, and $k+2$ in QA are changed. If the elements k, $k+1$, and $k+2$ in a column of A are zero, they remain zero in QA. Likewise, if $Z \in \mathcal{U}_3(k)$, only columns k, $k+1$, and $k+2$ are changed in AZ. If some row has elements k, $k+1$, and $k+2$ zero, then they remain zero in AZ. Similar considerations hold for the class $\mathcal{U}_2$.

All our transformations will be denoted by Q's and Z's with various subscripts. The Q's will always be premultpliers, that is row operations. The Z's will always be postmultipliers, or column operations. The letter Q is being used in its traditional role to denote orthogonal matrices. The letter Z was chosen to denote orthogonal matrices which introduce zeros in strategic locations.

The first step in the reduction is to reduce B to upper triangular form by premultiplication by Householder reflections. The details of this reduction are well known (e.g. see [4,11]) and we confine ourselves to a brief description to illustrate our notation. At the k-th stage of the reduction (illustrated below for $k = 3$ and $n = 5$), the elements below the first k-1 diagonal elements of B are zero.

$$\begin{matrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & x^1 & x & x \\ 0 & 0 & x^1 & x & x \end{matrix}$$

Each x represents an arbitrary nonzero element. Each x^1 represents an element to be annihilated in the next step. A matrix $Q_k \in \mathcal{U}_{n-k+1}(k)$ is chosen to annihilate $b_{k+1,k}, b_{k+2,k}, \ldots, b_{n,k}$, and B is overwritten by $Q_k B$, giving a matrix of the form illustrated below.

$$\begin{matrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & x^1 & x \end{matrix}$$

This process is repeated until $k = n-1$. Of course A is overwritten by $Q_{n-1}Q_{n-2}\cdots Q_1 A$.

After this reduction, A and B have the form

A					B				
x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	0	x	x	x	x
x	x	x	x	x	0	0	x	x	x
x	x	x	x	x	0	0	0	x	x
x^1	x	x	x	x	0	0	0	0	x

The problem now is reduce A to upper Hessenberg form while preserving the triangularity of B . This is done as follows (for $k = 5$). First $Q \in \mathcal{R}_2(4)$ is determined to annihilate a_{51} . The matrices QA and QB , which overwrite A and B then have the form

x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	0	x	x	x	x
x	x	x	x	x	0	0	x	x	x
x	x	x	x	x	0	0	0	x	x
0	x	x	x	x	0	0	0	x^1	x

The transformation has introduced a nonzero element on the $(5,4)$-position of B . However, a $Z \in \mathcal{R}_2(4)$ can be used to restore the zero without disturbing the zero introduced in A .

This step is typical of all the others. The elements of A are annihilated by Q's in the order illustrated below.

x	x	x	x	x
x	x	x	x	x
x^3	x	x	x	x
x^2	x^5	x	x	x
x^1	x^4	x^6	x	x

As each element of A is annihilated, it introduces a nonzero element on the subdiagonal of B, which is immediately annihilated by a suitably chosen Z. The entire algorithm, including the Householder triangularization of B may be summed up as follows.

1) For $k = 1,2,\ldots,n-1$
   1) Choose $Q_k \in \mathcal{H}_{n-k+1}(k)$ to annihilate $b_{k+1,k}, b_{k+2,k}, \ldots, b_{n,k}$.
   2) $B \leftarrow Q_k B$, $A \leftarrow Q_k A$
2) For $k = 1,2,\ldots,n-2$
   1) For $\ell = n-1,n-2,\ldots,k+1$
      1) Choose $Q_{k\ell} \in \mathcal{H}_2(\ell)$ to annihilate $a_{\ell+1,k}$
      2) $A \leftarrow Q_{k\ell} A$, $B \leftarrow Q_{k\ell} B$
      3) Choose $Z_{k\ell} \in \mathcal{H}_2(\ell)$ to annihilate $b_{\ell+1,\ell}$
      4) $B \leftarrow B Z_{k\ell}$, $A \leftarrow A Z_{k\ell}$

The complete reduction requires about $\frac{17}{3} n^3$ multiplications, $\frac{17}{3} n^3$ additions and n^2 square roots. If eigenvectors are also to be computed, the product of the Z's must be accumulated. This requires an additional $\frac{3}{2} n^3$ multiplications and $\frac{3}{2} n^3$ additions. The product of the Q's is not required for the computation of eigenvectors.

3. The Explicit QZ Step

In this and the next section we assume that A is upper Hessenberg and B is upper triangular. In this section we shall propose an iterative technique for reducing A to upper triangular form while maintaining the triangularity of B. The idea of our approach is to pretend that B is nonsingular and examine the standard QR algorithm for $C = AB^{-1}$. The manipulations are then interpreted as unitary equivalences on A and B.

Specifically suppose that one step of the QR algorithm with shift k is applied to C. Then Q is determined as an orthogonal transformation such that the matrix

$$R = Q(C-kI) \tag{3.1}$$

is upper triangular. The next iterate C' is defined as

$$C' = RQ^T + kI \equiv QCQ^T .$$

If we set

$$A' = QAZ$$

and

$$B' = QBZ ,$$

where Z is any unitary matrix, then

$$A'B'^{-1} = QAZZ^TB^{-1}Q^T = QAB^{-1}Q^T = C' .$$

The matrix Q is determined by the requirement that R be upper triangular. We choose Z so that A' is upper Hessenberg and B' is upper triangular. This insures that the nice distribution of zeros, introduced by the algorithm of Section 2, is preserved by the QZ step. Thus a tentative form of our algorithm might read

1) Determine Q so that QC is upper triangular,

2) Determine Z so that QAZ is upper Hessenberg and QBZ is upper triangular,

3) $A \leftarrow QAZ$, $B \leftarrow QBZ$.

The problem is then to give algorithms for computing Q and Z which do not explicitly require $C = AB^{-1}$.

The determination Q is relatively easy. For from (3.1) and the definition of C it follows that

$$Q(A-kB) = RB \equiv S \quad . \tag{3.2}$$

Since R and B are upper triangular, so is S . Thus Q is the unitary matrix that reduces A-kB to upper triangular form. Since A-kB is upper Hessenberg, Q can be expressed in the form

$$Q = Q_{n-1}Q_{n-2}\cdots Q_1 \quad , \tag{3.3}$$

where $Q_k \epsilon \mathscr{U}_2(k)$.

To calculate Z we apply Q in its factored form (3.3) to B and determine Z in a factored form so that B stays upper triangular. Specifically Q_1B has the form (k = 5)

$$\begin{matrix} x & x & x & x & x \\ x^1 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x \end{matrix}$$

If Q_1B is postmultiplied by a suitable $Z_1 \epsilon \mathscr{U}_2(1)$ the nonzero element below the diagonal can be removed. Similarly $Q_2Q_1BZ_1$ has the form

$$\begin{matrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & x^{1} & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x \end{matrix}$$

and the offending nonzero element can be removed by a $Z_2 \in \mathcal{U}_2(2)$. Proceeding in this way, we construct Z in the form

$$Z = Z_1 Z_2 \cdots Z_{n-1} \ ,$$

where $Z_k \in \mathcal{U}_2(k)$.

Although QBZ is upper triangular, it is not at all clear that QAZ is upper Hessenberg. To see that it is, rewrite equation (3.2) in the form

$$QAZ = SZ + kQBZ \ . \tag{3.4}$$

From the particular form of Z and the fact that S is upper triangular, it follows that SZ is upper Hessenberg. Thus (3.4) expresses QAZ as the sum of an upper Hessenberg and an upper triangular matrix. In fact (3.4) represents a computationally convenient form for computing QAZ .

We summarize as follows.

1) Determine $Q = Q_{n-1} Q_{n-2} \cdots Q_1$ $(Q_k \in \mathcal{U}_2(k))$ so that $S = Q(A-kB)$ is upper triangular.

2) Determine $Z = Z_1 Z_2 \cdots Z_{n-1}$ $(Z_k \in \mathcal{U}_2(k))$ so that $B' = QBZ$ is upper triangular

3) $A' = SZ + kB'$

If this algorithm is applied iteratively with shifts $k_1, k_2, \ldots$, there result sequences of matrices $A_1, A_2, \ldots$, $B_1, B_2, \ldots$, and $C_1, C_2, \ldots$ satisfying

$$A_{\nu+1} = QA_\nu Z \qquad B_{\nu+1} = QB_\nu Z$$

$$C_{\nu+1} = QC_\nu Q^T \qquad C_\nu = A_\nu B_\nu^{-1} \quad ,$$

provided B_1 is nonsingular. The matrices A_ν are upper Hessenberg and the B_ν are upper triangular. The C_ν are the upper Hessenberg matrices that would result from applying the QR algorithm with shifts $k_1, k_2, \ldots$ to C_1. As C_ν tends to upper triangular form, so must A_ν, since B_ν^{-1} is upper triangular.

Most of the properties of the QR algorithm carry over to the QZ algorithm. The eigenvalues will tend to appear in descending order as one proceeds along the diagonal. The convergence of $a_{n,n-1}^{(\nu)}$ to zero may be accelerated by employing one of the conventional shifting strategies. Once $a_{n,n-1}^{(\nu)}$ becomes negligible one can deflate the problem by working with the leading principal submatrices of order $n-1$. If some other subdiagonal element of A_ν, say $a_{\ell,\ell-1}^{(\nu)}$, becomes negligible, one can effect a further savings by working with rows and columns ℓ through n. Because we have used unitary transformations, an element of A_ν or B_ν can be regarded as negligible if a perturbation of the same size as the element can be tolerated in A_1 or B_1.

The algorithm given above is potentially unstable. If k is large compared with A and B, the formula (3.4) will involve subtractive cancellation and A' will be computed inaccurately. Since the shift k approximates the eigenvalue currently being found and the problem may have very large eigenvalues, there is a real possibility of encountering a large shift. Fortunately the large eigenvalues tend to be found last so that by the time a large shift emerges the small eigenvalues will have been computed stably. (The large eigenvalues are of course ill-conditioned

and cannot be computed accurately.) To be safe one might perform the first few iterations with a zero shift in order to give the larger eigenvalues a chance to percolate to the top.

4. Implicit Shifts

The potential instability in the explicit algorithm results from the fact that we have used formula (3.4) rather than unitary equivalences to compute A'. One way out of this difficulty is to generalize the implicit shift method for the QR algorithm to the QZ algorithm so that both A' and B' are computed by unitary equivalences. The implicit shift technique has the additional advantage that it can be adapted to perform two shifts at a time. For real matrices this means that a double shift in which the shifts are conjugate pairs can be performed in real arithmetic.

Since we are primarily interested in real matrices, we will concentrate on double shifts. The method is based on the following observation. Suppose that A is upper Hessenberg and B is upper triangular and nonsingular. Then if Q and Z are unitary matrices such that QAZ is upper Hessenberg and QBZ is upper triangular, then Q is determined by its first row. In fact, AB^{-1} and $QAB^{-1}Q^H$ are both upper Hessenberg, so that, by the theorem on page 352 of [11], Q is determined by its first row.

Thus we must do two things. First, find the first row of Q. Second, determine Q and Z so that Q has the correct first row, QAZ is upper Hessenberg, and QBZ is upper triangular. The first part is relatively easy. The first row of Q is the first row that would be obtained from a double shifted QR applied to AB^{-1}. Since A is upper Hessenberg and B upper triangular, it is easy to calculate the first two columns of AB^{-1}. But these, along with the shifts, completely determine the first row of Q. Only nonsingularity of the upper 2-by-2

submatrix of B is actually required here. If either b_{11} or b_{22} is too small, so that this submatrix is nearly singular, a type of deflation can be carried out. We will return to this point later.

The second part is a little more difficult, and is really the crux of the algorithm since it retains the Hessenberg and triangular forms. Only the first three elements of the first row of Q are nonzero. Thus, if Q_1 is a matrix in $\mathcal{N}_3(1)$ with the same first row of Q, then Q_1A and Q_1B have the following form (when $n = 6$)

$$\begin{matrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \end{matrix} \qquad \begin{matrix} x & x & x & x & x & x \\ x^2 & x & x & x & x & x \\ x^1 & x^1 & x & x & x & x \\ 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & 0 & x \end{matrix} .$$

As in the standard implicit shift QR algorithm, it is convenient to think of Q_1 as the reflection which annihilates two of the three nonzero elements in a fictitious "zeroth" column of A.

We must reduce Q_1A to upper Hessenberg and Q_1B to upper triangular by unitary equivalences. However, we may not premultiply by anything which affects the first row. This is done as follows. The matrix Q_1B has three nonzero elements outside the triangle. These can be annihilated by two Z's, a Z_1' in $\mathcal{N}_3(1)$ which annihilates the (3,1) and (3,2) elements and then a Z_1'' which annihilates the resulting (2,1) element. Let $Z_1 = Z_1'Z_1''$. Then Q_1BZ_1 is upper triangular. Applying Z_1 to Q_1A gives Q_1AZ_1 with the following form

$$\begin{matrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ x^1 & x & x & x & x & x \\ x^1 & x & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \end{matrix}$$

This is multiplied by Q_2 in $\mathcal{U}_3(2)$ that annihilates the $(3,1)$ and $(4,1)$ elements. Then $Q_2Q_1AZ_1$ and $Q_2Q_1BZ_1$ have the forms

$$\begin{matrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & 0 & 0 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \end{matrix} \qquad \begin{matrix} x & x & x & x & x & x \\ 0 & x & x & x & x & x \\ 0 & x^2 & x & x & x & x \\ 0 & x^1 & x^1 & x & x & x \\ 0 & 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & 0 & x \end{matrix}$$

The first columns are now in the desired form. The nonzero elements outside the desired structure have been "chased" into the lower 5-by-5 submatrices.

Now, postmultiply by Z_2, a product of a matrix in $\mathcal{U}_3(2)$ and a matrix in $\mathcal{U}_2(2)$ that reduces the current B to triangular form. Then premultiply by Q_3 in $\mathcal{U}_3(3)$ to annihilate two elements outside the Hessenberg structure of the resulting A.

The process continues in a similar way, chasing the unwanted nonzero elements towards the lower, right-hand corners. It ends with a slightly simpler step which uses Q_{n-2} in $\mathcal{U}_2(n-1)$ to annihilate the $(n,n-2)$ element of the current A, thereby producing a Hessenberg matrix, and Z_{n-2} in $\mathcal{U}_2(n-1)$ which annihilates the $(n,n-1)$ element of the current B, producing a triangular B but not destroying the Hessenberg A.

The fictitious zeroth column of A is determined in part by the shifts. In analogy with the implicit double shift algorithm, we take the shifts k_1 and k_2 to be the two zeros of the two-by-two problem

$$\det(\bar{A}-k\bar{B}) = 0$$

where

$$\bar{A} = \begin{pmatrix} a_{n-1,n-1} & a_{n-1,n} \\ a_{n,n-1} & a_{n,n} \end{pmatrix}, \quad \bar{B} = \begin{pmatrix} b_{n-1,n-1} & b_{n-1,n} \\ 0 & b_{n,n} \end{pmatrix}$$

It is not desirable to compute k_1 and k_2 explicitly, or even to find the coefficients in the quadratic polynomial $\det(\bar{A}-k\bar{B})$. Instead, following the techniques used in " hqr2 " [8], we obtain ratios of the three nonzero elements of the first column of $(AB^{-1}-k_1I)(AB^{-1}-k_2I)$ directly from formulas which involve only the differences of diagonal elements. This insures that small, but non-negligible, offdiagonal elements are not lost in the shift calculation. The formulas are

$(m = n-1)$

$$a_{10} = \left[\left(\frac{a_{mm}}{b_{mm}} - \frac{a_{11}}{b_{11}}\right)\left(\frac{a_{nn}}{b_{nn}} - \frac{a_{11}}{b_{11}}\right) - \left(\frac{a_{mn}}{b_{nn}}\right)\left(\frac{a_{nm}}{b_{mm}}\right) + \left(\frac{a_{nm}}{b_{mm}}\right)\left(\frac{b_{mn}}{b_{nn}}\right)\left(\frac{a_{11}}{b_{11}}\right)\right]\cdot\left(\frac{b_{11}}{a_{21}}\right) + \frac{a_{12}}{b_{22}} - \left(\frac{a_{11}}{b_{11}}\right)\left(\frac{b_{12}}{b_{22}}\right)$$

(4.1)

$$a_{20} = \left(\frac{a_{22}}{b_{22}} - \frac{a_{11}}{b_{11}}\right) - \left(\frac{a_{21}}{b_{11}}\right)\left(\frac{b_{12}}{b_{22}}\right) - \left(\frac{a_{mm}}{b_{mm}} - \frac{a_{11}}{b_{11}}\right) - \left(\frac{a_{nn}}{b_{nn}} - \frac{a_{11}}{b_{11}}\right) + \left(\frac{a_{nm}}{b_{mm}}\right)\left(\frac{b_{mn}}{b_{nn}}\right)$$

$$a_{30} = \frac{a_{32}}{b_{22}}$$

We are now in a position to summarize the double implicit shift method. It is understood that A and B are to be overwritten by the transformed matrices as they are generated.

1) Compute a_{10}, a_{20}, and a_{30} by (4.1).

2) For $k = 1,2,\ldots,n-2$

   a) Determine $Q_k \in \mathcal{N}_3(k)$ to annihilate $a_{k+1,k-1}$ and $a_{k+2,k-1}$.

   b) Determine $Z_k' \in \mathcal{N}_3(k)$ to annihilate $b_{k+2,k+1}$ and $b_{k+2,k}$.

   c) Determine $Z_k'' \in \mathcal{N}_2(k)$ to annihilate $b_{k+1,k}$.

3) Determine $Q_{n-1} \in \mathcal{N}_2(n-1)$ to annihilate $a_{n,n-2}$.

4) Determine $Z_{n-1} \in \mathcal{N}_2(n-1)$ to annihilate $b_{n,n-1}$.

For each k, determination of Q_k requires a few multiplications and one square root. Application of Q_k to both A and B requires about $10(n-k)$ multiplications. The work involved with each Z_k' is the same. Application of Z_k'' requires only about $6(n-k)$ multiplications. The number of additions is about the same. Summing these for k from 1 to $n-1$ gives a total of about $13n^2$ multiplications, $13n^2$ additions and $3n$ square roots per double iteration.

By way of comparison, for the double shift QR algorithm as implemented in " hqr ", Z_k' becomes simply Q_k^T and Z_k'' is not used. Furthermore, the transformations are carried out on only one matrix. Consequently, each double iteration requires about $5n^2$ multiplications, $5n^2$ additions and n square roots. Thus the QZ algorithm applied on two matrices can be expected to require roughly 2.6 times as much work per iteration as the QR algorithm on a single matrix.

In order to obtain eigenvectors, the Q's are ignored and the Z's accumulated. This requires about $8n^2$ more multiplications and $8n^2$ more additions per double iteration.

There is one difficulty. The formulas for a_{10}, a_{20}, and a_{30} are not defined when b_{11} and b_{22} are zero. Moreover, as b_{11} and b_{22} approach zero the terms that determine the shift (terms involving a_{nn}, b_{nn}, etc.) become negligible compared to the other terms, so that the effect of the shift is felt only weakly.

Part of the solution to this difficulty is to deflate from the top. If b_{11} is negligible it may be set to zero to give the forms for A and B $(n = 4)$

$$\begin{matrix} x & x & x & x \\ x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \end{matrix} \qquad \begin{matrix} 0 & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{matrix}$$

A Q in $\mathcal{V}_2(1)$ can then be used to annihilate the (2,1) element of A, which deflates the problem.

The rest of the solution lies in recognizing that there is not much of a problem. If b_{11} and b_{22} are small then the problem has large eigenvalues. We have already observed that the larger eigenvalues tend to emerge at the upper left, and the larger the eigenvalue, the swifter its emergence. Moreover the speed will not be affected by a small shift. This means that whenever the implicit shift is diluted by a small b_{11} or b_{22}, the algorithm is none the less profitably employed in finding a large eigenvalue.

5. Further Reduction of the Quasi-Triangular Form

The result of the algorithm described so far is in an upper triangular matrix B and a quasi-upper triangular matrix A in which no two consecutive subdiagonal elements are nonzero. This means that the original problem decomposes into one by one and two by two subproblems. The eigenvalues of the one by one problems are the ratios of the corresponding diagonal elements of A and B. The eigenvalues of the two by two problems might be calculated as the roots of a quadratic equation, and may be complex even for real A and B.

There are two good reasons for not using the quadratic directly, but instead reducing the two by two problems. First, when A and B are real, the calculation of eigenvectors is greatly facilitated if all the real eigenvalues are contained in one by one problems. A more important second reason is that the one by one problems contain more information then the eigenvalues alone. For example, if a_{11} and b_{11} are small then the eigenvalue $\lambda_1 = a_{11}/b_{11}$ is ill conditioned, however reasonable it may appear. This reason obviously applies to complex eigenvalues as well as real ones. Accordingly, we recommend that the two by two problems be reduced to one by one problems and that the diagonal elements, rather than the eigenvalues, be reported.

Without loss of generality we may consider the problem of reducing two by two matrices A and B simultaneously to upper triangular form by unitary equivalences. For our purposes we may assume that B is upper triangular.

Two special cases may be disposed of immediately. If b_{11} is zero, then a $Q_{23}(1)$ may be chosen to reduce a_{21} to zero. The zero elements

of QB are not disturbed. Similarly, if b_{22} is zero, a $Z \epsilon \mathcal{U}_2(1)$ may be chosen to reduce a_{21} to zero without disturbing b_{21}.

In the general two-by-two case, it is not difficult to write down formulas for the elements of $A' = QAZ$ and $B' = QBZ$ for any Q and Z. Moreover, these formulas can be arranged so that numerically one of a'_{21} or b'_{21} is effectively zero. It is not obvious, however, that the other element is numerically zero, and the effect of assuming that it is by setting it to zero could be disastrous. Consequently, we must consider a somewhat more complicated procedure.

The theoretical procedure for reducing A to triangular form may be described as follows. Let λ be an eigenvalue of the problem and form the matrix $E = A - \lambda B$. Choose a $Z \epsilon \mathcal{U}_2(1)$ to annihilate either e_{11} or e_{21}. Since the rows of E are parallel, it follows that whichever of e_{11} or e_{21} is annihilated the other must also be annihilated. Now choose $Q \epsilon \mathcal{U}_2(1)$ so that either QAZ or QBZ is upper triangular. Since the first column of QEZ is zero and $QEZ = QAZ - \lambda QBZ$, it follows that, however Q is chosen, both QAZ and QBZ must be upper triangular.

In the presence of rounding error the method of computing λ and the choice of Z and Q are critical to the stability of the process. A rigorous rounding error analysis will show that, under a reasonable assumption concerning the computed λ, the process described below is stable. However, to avoid excessive detail, we only outline the analysis. We assume that all computations are done in floating point arithmetic with t base β digits and that the problem has been so scaled that underflows and overflows do not occur. We further assume that a_{21} is not negligible in the sense that $|a_{21}| < \beta^{-t}\|A\|$, where $\|\cdot\|$ denotes, say, the row sum norm.

The algorithm for computing λ amounts to making an appropriate origin shift and computing an eigenvalue from the characteristic equation. It goes as follows.

$$\mu = \frac{a_{11}}{b_{11}}$$

$$\bar{a}_{12} = a_{12} - \mu b_{12}$$

$$\bar{a}_{22} = a_{22} - \mu b_{22}$$

$$p = \frac{1}{2}\left(\frac{\bar{a}_{22}}{b_{22}} - \frac{b_{12}a_{21}}{b_{11}b_{22}} \right)$$

$$q = \frac{a_{21}\bar{a}_{12}}{b_{11}b_{22}}$$

$$r = p^2 + q$$

$$\lambda = \mu + p + \text{sign}(p) \cdot \sqrt{r} \qquad \text{(complex if } r < 0 \text{)} \tag{5.1}$$

We must now assume that the computed λ satisfies the equation

$$\det(A' - \lambda B') = 0 \ ,$$

where $\|A-A'\| \leq \sigma_A \|A\|$ and $\|B-B'\| \leq \sigma_B \|B\|$ with σ_A and σ_B small constants of order β^{-t}. Define

$$E' = A' - \lambda B'$$

and let E denote the computed value

$$E = \text{fl}(A - \lambda B) \ .$$

Then

$$E' = E + H$$

with $\|H\| \leq \sigma \max\{\|A\|, |\lambda| \ \|B\|\}$ with σ of order β^{-t}.

We claim that, approximately,

$$\|E\| \geq \beta^{-t} \max\{\|A\|, |\lambda| \ \|B\|\} \ . \tag{5.2}$$

First we note that

$$(5.3) \qquad \|E\| \geq |e_{21}| = |a_{21}| \geq \beta^{-t}\|A\| ,$$

by the assumption that a_{21} is significant. Now assume that $\|E\| < \beta^{-t}|\lambda|\,\|B\|$. Then subtractive cancellation must occur in the computation of e_{11}, e_{12}, and e_{22}. Thus $a_{11} \approx \lambda b_{11}$, $a_{12} \approx \lambda b_{12}$ and $a_{22} \approx \lambda b_{22}$. Hence we have $\|A\| \geq |\lambda|\,\|B\|$, and, from (5.3), $\|E\| \geq \beta^{-t}|\lambda|\,\|B\|$, a contraction.

Now

$$0 = \det(E') = \det(E) + (e_{11}+h_{11})h_{22} - (e_{12}+h_{12})h_{21} + h_{11}e_{22} - h_{12}e_{21} ,$$

Hence

$$|\det(E)| \leq \rho_1\|E\| \max\{\|A\|, |\lambda|\,\|B\|\} + \rho_2[\max\{\|A\|, |\lambda|\,\|B\|\}]^2$$

where ρ_1 and ρ_2 are of order β^{-t}. From (5.2) it then follows that

$$|\det(E)| \leq \rho\|E\| \max\{\|A\|, |\lambda|\,\|B\|\}$$

where ρ is of order β^{-t}.

Now consider the determination of Z. Assume that the second row of E is larger than the first. Then $Z \epsilon \mathcal{U}_2(1)$ is chosen to annihilate e_{21}. Let $F = EZ$. Then f_{21} is essentially zero. Furthermore, since Z is unitary

$$|f_{11}f_{22}| = |\det(E)| \leq \rho\|E\| \max\{\|A\|, |\lambda|\|B\|\} .$$

But $|f_{22}| \cong \|e_2\|$ and, since e_2 was assumed to be the larger row, $\|e_2\| = \|E\|$. Hence we have approximately

$$|f_{11}| \leq \rho \max\{\|A\|, |\lambda|\|B\|\} .$$

To choose Q, let

$$C = AZ \quad , \quad D = BZ ,$$

and let f_1, c_1, and d_1 be the first columns of F, C, and D. Let q_2^T denote the second row of Q. If $\|A\| \geq |\lambda|\|B\|$, we choose Q to annihilate d_{21}. Numerically this means that

$$|q_2^T d_1| \leq \sigma\|B\| ,$$

where σ is a constant on the order of β^{-t}. We must show that $q_2^T c_1$ is negligible. But

$$\begin{aligned} |q_2^T c_1| &= |q_2^T f_1 + \lambda q_c^T d_1| \\ &\leq \|f_1\| + |\lambda| \|q_c^T d_1\| \\ &\leq \rho \max\{\|A\|, |\lambda|\|B\|\} + \sigma|\lambda|\|B\| \\ &\leq (\rho + \sigma)\|A\| . \end{aligned}$$

If, on the other hand, $|\lambda|\|B\| > \|A\|$, we choose Q so that

$$|q_2^T c_1| \leq \sigma\|A\| .$$

It then follows that

$$\begin{aligned} |q_2^T d_1| &= |q_1^T f - q_2^T c_1| / |\lambda| \\ &\leq \rho|\lambda|^{-1} \max\{\|A\|, |\lambda|\|B\|\} + \sigma|\lambda|^{-1}\|A\| \\ &\leq (\rho + \sigma)\|B\| . \end{aligned}$$

In summary, λ is computed using (5.1), Z is chosen to annihilate the first element of the larger of the two rows of $A-\lambda B$ and Q is chosen to annihilate the $(2,1)$ element of the smaller of the two matrices AZ and λBZ. In this way, we can be sure that the computed $(2,1)$ elements of both QAZ and QBZ are negligible.

In practice with matrices of any order, if the transformations are real, they are applied to the entire matrices. If the transformations are

complex, they are used to compute the diagonal elements that would result, but are not actually applied. We thus obtain a quasi-triangular problem in which each two-by-two block is known to correspond to a pair of complex eigenvalues.

The generalized eigenvectors of this reduced problem can be found by a back-substitution process which is a straightforward extension of the method used in " hqr2 " [8]. The vectors of the original problem are then found by applying the accumulated Z's .

6. Some Numerical Results

The entire process described above has been implemented in a Fortran program [7]. There are four main subroutines: the initial reduction to Hessenberg-triangular form, the iteration itself, the computation of the final diagonal elements, and the computation of the eigenvectors. The complete program contains about 600 Fortran statements, although this could be reduced somewhat at the expense of some clarity.

The numerical properties observed experimentally are consistent with the use of unitary transformations. The eigenvalues are always found to whatever accuracy is justified by their condition. If an eigenvalue and eigenvector are not too "ill-disposed", then they produce a small relative residual.

Similar numerical properties can not generally be expected from any algorithm which inverts B or any submatrix of B. This is even true of 2-by-2 submatrices, as illustrated by the following example due to Wilkinson.

$$A = \begin{pmatrix} .1 & .2 \\ .3 & .4 \end{pmatrix} \qquad B = \begin{pmatrix} .1 & .1 \\ 0 & \mu \end{pmatrix}$$

Here μ is about the square root of the machine precision, that is, μ is not negligible compared to 1, but μ^2 is. There is one eigenvalue near -2. Small relative changes in the elements of the matrices cause only small relative changes in this eigenvalue. The other eigenvalue becomes infinite as μ approaches zero. Great care must be taken in solving this problem so that the mild instability of the one eigenvalue does not cause an inaccurate result for the other, stable eigenvalue.

Of course, the use of unitary transformations makes our technique somewhat slower than others which might be considered. But the added cost is not very great. In testing our program, we solve problems of order 50 regularly. A few problems of orders greater than 100 have been run, but these become somewhat expensive when they are merely tests.

One typical example of order 50 requires 45 seconds on Stanford's IBM 360 model 67. Of this, 13 seconds are spent in the initial reduction 29 seconds are used for the 61 double iterations required, and 3 seconds are needed for the diagonal elements and eigenvectors. If the eigenvectors are not needed and so the transformations not saved, the total time is reduced to 27 seconds. By way of comparison, formation of $B^{-1}A$ a la Peters and Wilkinson [9] and use of Fortran versions [12] of "orthes" [5] and " hqr2 " [8] requires a total of 27 seconds for this example. (All of these times are for code generated by the IBM Fortran IV compiler, H level, with the optimization parameter set to 2 .)

In the examples we have seen so far, the total number of double iterations required is usually about 1.2 or 1.3 times the order of the matrices. This figure is fairly constant, although it is not difficult to find examples which require many fewer or many more iterations. As a rule of thumb, for a matrix of order n the time required on the model 67 is about $.36\ n^3$ milliseconds if vectors are computed, $.22\ n^3$ milliseconds if they are not.

The example in Table 1 is not typical, but it does illustrate several interesting points. It was generated by applying non-orthogonal rank one modifications of the identity to direct sums of companion matrices. The companion matrices were chosen so that the resulting problem has three double roots,

$$\lambda_1 = \lambda_2 = \infty \quad ,$$

$$\lambda_3 = \lambda_5 = \frac{1}{2} + \frac{\sqrt{3}}{2} i \quad ,$$

$$\lambda_4 = \lambda_6 = \frac{1}{2} - \frac{\sqrt{3}}{2} i \quad .$$

The double root at ∞ results from the fact that B has a double zero eigenvalue. All three roots are associated with quadratic elementary divisors; i.e., each root has only one corresponding eigenvector. The computed diagonals of the triangularized matrices are given in the table. Note that the four finite eigenvalues are obtained with a relative accuracy of about 10^{-8}. This is about the square root of the machine precision and is the expected behavior for eigenvalues with quadratic elementary divisors. The singularity of B does not cause any further deterioration in their accuracy. Furthermore, the infinite eigenvalues are obtained from the reciprocals of quantities which are roughly the square root of the machine precision times the norm of B. Consequently we are somewhat justified if we claim to have computed the square root of infinity.*/

*/ This prompts us to recall the limerick which introduces George Gamow's One, Two, Three, Infinity:

There was a young fellow from Trinity
Who tried $\sqrt{\infty}$
But the number of digits
Gave him such fidgits
That he gave up Math for Divinity.

$$A = \begin{matrix} 50 & -60 & 50 & -27 & 6 & 6 \\ 38 & -28 & 27 & -17 & 5 & 5 \\ 27 & -17 & 27 & -17 & 5 & 5 \\ 27 & -28 & 38 & -17 & 5 & 5 \\ 27 & -28 & 27 & -17 & 16 & 5 \\ 27 & -28 & 27 & -17 & 5 & 16 \end{matrix} \qquad B = \begin{matrix} 16 & 5 & 5 & 5 & -6 & 5 \\ 5 & 16 & 5 & 5 & -6 & 5 \\ 5 & 5 & 16 & 5 & -6 & 5 \\ 5 & 5 & 5 & 16 & -6 & 5 \\ 5 & 5 & 5 & 5 & -6 & 16 \\ 6 & 6 & 6 & 6 & -5 & 6 \end{matrix}$$

α	β
25.768670843143	$.2637605112.10^{-6}$
-12.821841071323	$.1312405807.10^{-6}$
5.814535434181 + 10.071071345641 i	11.629071028730
5.800765071150 - 10.047220375909 i	11.601530302268
5.736511506410 + 9.935928843473 i	11.473022854605
5.510879468089 - 9.545122710676 i	11.021758784186

α/β

$0.976972281.10^{8}$

$-0.976972290.10^{8}$

0.49999999310489 + 0.866025439242?1 i

0.49999999310489 - 0.866025439242?1 i

0.50000000689511 + 0.8660253683261? i

0.50000000689511 - 0.86602536832617 i

Table 1

Acknowledgments

Most of Moler's work was done during a visit to Stanford University where he received support from the Computer Science Department, the Stanford Linear Accelerator Center and National Science Foundation grant GJ-1158. Partial support was also obtained at the University of Michigan from the Office of Naval Research, contract NR-044-377. Stewart received support at the University of Texas from NSF grant GP-2[illegible]55.

W. Kahan and J. H. Wilkinson made several helpful comments. Linda Kaufman of Stanford has recently shown how to carry out the corresponding generalization of the LR algorithm and has written a program that accepts general complex matrices.

References

[1] Charles Crawford, "The numerical solution of the generalized eigenvalue problem," Ph.D. Thesis, University of Michigan, 1970. Also to appear in Comm. ACM.

[2] G. Fix and R. Heiberger, "An algorithm for the ill-conditioned generalized eigenvalue problem," to appear in Numer. Math.

[3] J. G. F. Francis, "The QR transformation -- a unitary analogue to the LR transformation," Computer Journal 4, 265-271, 332-345 (1961-62).

[4] A. S. Householder, "Unitary triangularization of a nonsymmetric matrix," J. Assoc. Comput. Mach. 5, 339-342 (1958).

[5] R. S. Martin and J. H. Wilkinson, "Similarity reduction of a general matrix to Hessenberg form," Numer. Math. 12, 349-368 (1968).

[6] R. S. Martin and J. H. Wilkinson, "Reduction of the symmetric eigenproblem $Ax = \lambda Bx$ and related problems to standard form," Numer. Math. 11, 99-110 (1968).

[7] C. B. Moler and G. W. Stewart, "The QZ algorithm for $Ax = \lambda Bx$," see Part II of this report.

[8] G. Peters and J. H. Wilkinson, "Eigenvectors of real and complex matrices by LR and QR triangularizations," Numer. Math. 16, 181-204, (1970).

[9] G. Peters and J. H. Wilkinson, "$Ax = \lambda Bx$ and the generalized eigenproblem," SIAM J. Numer. Anal. 7, 479-492 (1970).

[10] G. W. Stewart, "On the sensitivity of the eigenvalue problem $Ax = \lambda Bx$," submitted for publication.

[11] J. H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, 1965.

[12] Fortran translations of the Algol procedures in [5] and [8] may be obtained from the NATS Project, Applied Mathematics Division, Argonne National Laboratories, Argonne, Illinois.

Part II

FORTRAN PROGRAM

```
      SUBROUTINE QZ(ND,N,A,B,EPS,ALFR,ALFI,BETA,ITER,WANTX,X)
      DIMENSION A(ND,ND),B(ND,ND),ALFR(N),ALFI(N),BETA(N),X(ND,ND)
      DIMENSION ITER(N)
      LOGICAL WANTX
C
C   A AND B ARE N-BY-N REAL MATRICES, STORED IN ARRAYS WITH ND ROWS.
C   EPS IS THE RELATIVE PRECISION OF ELEMENTS OF A AND B.
C   FINDS N PAIRS OF SCALARS, (ALFA(M),BETA(M)) SO THAT
C     BETA(M)*A - ALFA(M)*B IS SINGULAR.
C   THE EIGENVALUES OF  A*X - LAMBDA*B*X  CAN BE OBTAINED BY
C     DIVIDING ALFA(M) BY BETA(M), EXCEPT BETA(M) MIGHT BE ZERO.
C   IF (WANTX) ALSO FINDS CORRESPONDING EIGENVECTORS.
C   USES ONLY UNITARY TRANSFORMATIONS, NO INVERSES,
C     SO EITHER A OR B (OR BOTH) MAY BE SINGULAR.
C
C   BETA(M) IS REAL.
C   ALFA(M) IS COMPLEX, REAL AND IMAGINARY PARTS IN ALFR(M) AND ALFI(M).
C   COMPLEX PAIRS OCCUR WITH ALFA(M)/BETA(M) AND ALFA(M+1)/BETA(M+1)
C     COMPLEX CONJUGATES EVEN THOUGH ALFA(M) AND ALFA(M+1) ARE NOT
C     NECESSARILY CONJUGATE.
C   USES ONLY REAL ARITHMETIC.
C   IF A AND B WERE REDUCED TO TRIANGULAR FORM BY UNITARY EQUIVALENCES,
C     ALFA AND BETA WOULD BE THE DIAGONALS.
C   A AND B ARE ACTUALLY REDUCED ONLY TO QUASI-TRIANGULAR FORM WITH
C     1-BY-1 AND 2-BY-2 BLOCKS ON DIAGONAL OF A.
C   IF ALFA(M) IS NOT REAL, THEN BETA(M) IS NOT ZERO.
C   ITER IS TROUBLE INDICATOR AND ITERATION COUNTER.
C     IF (ITER(1).EQ.0) EVERYTHING IS OK.
C     ITER(M) IS NUMBER OF ITERATIONS NEEDED FOR M-TH EIGENVALUE.
C     IF (ITER(1) THRU ITER(M) .EQ. -1) THEN ITERATION FOR M-TH
C       EIGENVALUE DID NOT CONVERGE AND ALFA(1) THRU ALFA(M) AND
C       BETA(1) THRU BETA(M) ARE PROBABLY INACCURATE.
C   IF (WANTX) X(.,M) IS THE M-TH REAL EIGENVECTOR.
C     X(.,M) AND X(.,M+1) ARE THE REAL AND IMAGINARY PARTS
C     OF THE M-TH COMPLEX EIGENVECTOR.
C     X(.,M) AND -X(.,M+1) AND THE REAL AND IMAGINARY PARTS
C     OF THE (M+1)-ST COMPLEX EIGENVECTOR.
C     VECTORS NORMALIZED SO THAT LARGEST COMPONENT IS 1. OR 1.+0.I .
C
C   USES FOUR PRIMARY SUBROUTINES, QZHES, QZIT, QZVAL AND QZVEC.
C   USES FOUR AUXILLIARY SUBROUTINES, HSH3, HSH2, CHSH2 AND CDIV.
C   USES TWO STANDARD FUNCTIONS, SQRT AND ABS.
C   AUTHORS:  C. B. MOLER, STANFORD, AND G. W. STEWART, U. OF TEXAS
C   THIS VERSION DATED 7/19/71.
C
      CALL QZHES(ND,N,A,B,WANTX,X)
      CALL QZIT (ND,N,A,B,EPS,EPSA,EPSB,ITER,WANTX,X)
      CALL QZVAL(ND,N,A,B,EPSA,EPSB,ALFR,ALFI,BETA,WANTX,X)
      IF (WANTX) CALL QZVEC(ND,N,A,B,EPSA,EPSB,ALFR,ALFI,BETA,X)
      RETURN
      END
C
C
```

```
      SUBROUTINE QZHES(ND,N,A,B,WANTX,X)
      DIMENSION A(ND,ND),B(ND,ND),X(ND,ND)
      LOGICAL WANTX
C
C  INITIALIZE X, USED TO SAVE TRANSFORMATIONS
C
      IF (.NOT.WANTX) GO TO 10
      DO 3 I=1,N
         DO 2 J=1,N
            X(I,J) = 0.
    2    CONTINUE
         X(I,I) = 1.
    3 CONTINUE
C
C  REDUCE B TO UPPER TRIANGULAR
C
   10 NM1=N-1
      DO 100 L=1,NM1
         L1 = L+1
         S = 0.
         DO 20 I=L1,N
            IF (ABS(B(I,L)).GT.S) S = ABS(B(I,L))
   20    CONTINUE
         IF (S.EQ.0.) GO TO 100
         IF (ABS(B(L,L)).GT.S) S = ABS(B(L,L))
         R = 0.
         DO 25 I=L,N
            B(I,L) = B(I,L)/S
            R = R + B(I,L)**2
   25    CONTINUE
         R = SQRT(R)
         IF (B(L,L).LT.0.) R = -R
         B(L,L) = B(L,L) + R
         RHO = R*B(L,L)
         DO 50 J=L1,N
            T = 0.
            DO 30 I=L,N
               T = T + B(I,L)*B(I,J)
   30       CONTINUE
            T = -T/RHO
            DO 40 I=L,N
               B(I,J) = B(I,J) + T*B(I,L)
   40       CONTINUE
   50    CONTINUE
         DO 80 J=1,N
            T = 0.
            DO 60 I=L,N
               T = T + B(I,L)*A(I,J)
   60       CONTINUE
            T = -T/RHO
            DO 70 I=L,N
               A(I,J) = A(I,J) + T*B(I,L)
   70       CONTINUE
   80    CONTINUE
         B(L,L) = -S*R
         DO 90 I=L1,N
            B(I,L) = 0.
   90    CONTINUE
```

```
  100 CONTINUE
C
C  REDUCE A TO UPPER HESSENBERG, KEEP B TRIANGULAR
C
      IF (N.LE.2) GO TO 170
      NM2=N-2
      DO 160 K=1,NM2
         K1 = K+1
         NK1 = N-K-1
         DO 150 LB=1,NK1
            L = N-LB
            L1 = L+1
            CALL HSH2(A(L,K),A(L1,K),U1,U2,V1,V2)
            IF (U1.NE.1.) GO TO 125
            DO 110 J=K,N
               T = A(L,J) + U2*A(L1,J)
               A(L,J) = A(L,J) + T*V1
               A(L1,J) = A(L1,J) + T*V2
  110       CONTINUE
            A(L1,K) = 0.
            DO 120 J=L,N
               T = B(L,J) + U2*B(L1,J)
               B(L,J) = B(L,J) + T*V1
               B(L1,J) = B(L1,J) + T*V2
  120       CONTINUE
  125       CALL HSH2(B(L1,L1),B(L1,L),U1,U2,V1,V2)
            IF (U1.NE.1.) GO TO 150
            DO 130 I=1,L1
               T = B(I,L1) + U2*B(I,L)
               B(I,L1) = B(I,L1) + T*V1
               B(I,L) = B(I,L) + T*V2
  130       CONTINUE
            B(L1,L) = 0.
            DO 140 I=1,N
               T = A(I,L1) + U2*A(I,L)
               A(I,L1) = A(I,L1) + T*V1
               A(I,L) = A(I,L) + T*V2
  140       CONTINUE
            IF (.NOT.WANTX) GO TO 150
            DO 145 I=1,N
               T = X(I,L1) + U2*X(I,L)
               X(I,L1) = X(I,L1) + T*V1
               X(I,L) = X(I,L) + T*V2
  145       CONTINUE
  150    CONTINUE
  160 CONTINUE
  170 CONTINUE
      RETURN
      END
C
C
```

```
      SUBROUTINE QZIT (ND,N,A,B,EPS,EPSA,EPSB,ITER,WANTX,X)
      DIMENSION A(ND,ND),B(ND,ND),X(ND,ND)
      DIMENSION ITER(N)
      LOGICAL WANTX,MID
C
C  INITIALIZE ITER, COMPUTE EPSA,EPSB
C
      ANORM = 0.
      BNORM = 0.
      DO 185 I=1,N
         ITER(I) = 0
         ANI = 0.
         IF (I.NE.1) ANI = ABS(A(I,I-1))
         BNI = 0.
         DO 180 J=I,N
            ANI = ANI + ABS(A(I,J))
            BNI = BNI + ABS(B(I,J))
  180    CONTINUE
         IF (ANI.GT.ANORM) ANORM = ANI
         IF (BNI.GT.BNORM) BNORM = BNI
  185 CONTINUE
      EPSA = EPS*ANORM
      EPSB = EPS*BNORM
C
C  REDUCE A TO QUASI-TRIANGULAR, KEEP B TRIANGULAR
C
      M = N
  200 IF (M.LE.2) GO TO 390
C
C     CHECK FOR CONVERGENCE OR REDUCIBILITY
C
      DO 220 LB=1,M
         L = M+1-LB
         IF (L.EQ.1) GO TO 260
         IF (ABS(A(L,L-1)) .LE. EPSA) GO TO 230
  220 CONTINUE
  230 A(L,L-1) = 0.
      IF (L.LT.M-1) GO TO 260
      M = L-1
      GO TO 200
C
C     CHECK FOR SMALL TOP OF B
C
  260 IF (ABS(B(L,L)).GT.EPSB) GO TO 300
      B(L,L) = 0.
      L1 = L+1
      CALL HSH2(A(L,L),A(L1,L),U1,U2,V1,V2)
      IF (U1.NE.1.) GO TO 280
      DO 270 J=L,N
         T = A(L,J) + U2*A(L1,J)
         A(L,J) = A(L,J) + T*V1
         A(L1,J) = A(L1,J) + T*V2
         T = B(L,J) + U2*B(L1,J)
         B(L,J) = B(L,J) + T*V1
         B(L1,J) = B(L1,J) + T*V2
  270 CONTINUE
  280 L = L1
      GO TO 230
```

```
C
C       BEGIN ONE QZ STEP,  ITERATION STRATEGY
C
  300 M1 = M - 1
      L1 = L + 1
      CONST = 0.75
      ITER(M) = ITER(M) + 1
      IF (ITER(M).EQ.1) GO TO 305
      IF (ABS(A(M,M-1)).LT.CONST*OLD1) GO TO 305
      IF (ABS(A(M-1,M-2)).LT.CONST*OLD2) GO TO 305
      IF (ITER(M).EQ.10) GO TO 310
      IF (ITER(M).GT.30) GO TO 380
C
C       ZEROTH COLUMN OF A
C
  305 B11 = B(L,L)
      B22 = B(L1,L1)
      IF (ABS(B22).LT.EPSB) B22 = EPSB
      B33 = B(M1,M1)
      IF (ABS(B33).LT.EPSB) B33 = EPSB
      B44 = B(M,M)
      IF (ABS(B44).LT.EPSB) B44 = EPSB
      A11 = A(L,L)/B11
      A12 = A(L,L1)/B22
      A21 = A(L1,L)/B11
      A22 = A(L1,L1)/B22
      A33 = A(M1,M1)/B33
      A34 = A(M1,M)/B44
      A43 = A(M,M1)/B33
      A44 = A(M,M)/B44
      B12 = B(L,L1)/B22
      B34 = B(M1,M)/B44
      A10 = ( (A33-A11)*(A44-A11) - A34*A43 + A43*B34*A11 )/A21
     1        + A12 - A11*B12
      A20 = (A22-A11-A21*B12) - (A33-A11) - (A44-A11) + A43*B34
      A30 = A(L+2,L1)/B22
      GO TO 315
C
C       AD HOC SHIFT
C
  310 A10 = 0.
      A20 = 0.
      A30 = 1.1605
C
  315 OLD1 = ABS(A(M,M-1))
      OLD2 = ABS(A(M-1,M-2))
      IF (.NOT.WANTX) LOR1 = L
      IF (WANTX) LOR1 = 1
      IF (.NOT.WANTX) MORN = M
      IF (WANTX) MORN = N
C
C       BEGIN MAIN LOOP
C
      DO 360 K=L,M1
         MID = K.NE.M1
         K1 = K+1
         K2 = K+2
         K3 = K+3
```

```
      IF (K3.GT.M) K3 = M
      KM1 = K-1
      IF (KM1.LT.L) KM1 = L
      IF (K.EQ.L) CALL HSH3(A10,A20,A30,U1,U2,U3,V1,V2,V3)
      IF (K.GT.L.AND.K.LT.M1)
     1   CALL HSH3(A(K,XM1),A(K1,KM1),A(K2,KM1),U1,U2,U3,V1,V2,V3)
      IF (K.EQ.M1) CALL HSH2(A(K,KM1),A(K1,KM1),U1,U2,V1,V2)
      IF (U1.NE.1.) GO TO 325
      DO 320 J=KM1,MORN
         T = A(K,J) + U2*A(K1,J)
         IF (MID) T = T + U3*A(K2,J)
         A(K,J) = A(K,J) + T*V1
         A(K1,J) = A(K1,J) + T*V2
         IF (MID) A(K2,J) = A(K2,J) + T*V3
         T = B(K,J) + U2*B(K1,J)
         IF (MID) T = T + U3*B(K2,J)
         B(K,J) = B(K,J) + T*V1
         B(K1,J) = B(K1,J) + T*V2
         IF (MID) B(K2,J) = B(K2,J) + T*V3
320   CONTINUE
      IF (K.EQ.L) GO TO 325
      A(K1,K-1) = 0.
      IF (MID) A(K2,K-1) = 0.
325   IF (K.EQ.M1) GO TO 340
      CALL HSH3(B(K2,K2),B(K2,K1),B(K2,K),U1,U2,U3,V1,V2,V3)
      IF (U1.NE.1.) GO TO 340
      DO 330 I=LOR1,K3
         T = A(I,K2) + U2*A(I,K1) + U3*A(I,K)
         A(I,K2) = A(I,K2) + T*V1
         A(I,K1) = A(I,K1) + T*V2
         A(I,K) = A(I,K) + T*V3
         T = B(I,K2) + U2*B(I,K1) + U3*B(I,K)
         B(I,K2) = B(I,K2) + T*V1
         B(I,K1) = B(I,K1) + T*V2
         B(I,K) = B(I,K) + T*V3
330   CONTINUE
      B(K2,K) = 0.
      B(K2,K1) = 0.
      IF (.NOT.WANTX) GO TO 340
      DO 335 I=1,N
         T = X(I,K2) + U2*X(I,K1) + U3*X(I,K)
         X(I,K2) = X(I,K2) + T*V1
         X(I,K1) = X(I,K1) + T*V2
         X(I,K) = X(I,K) + T*V3
335   CONTINUE
340   CALL HSH2(B(K1,K1),B(K1,K),U1,U2,V1,V2)
      IF (U1.NE.1.) GO TO 360
      DO 350 I=LOR1,K3
         T = A(I,K1) + U2*A(I,K)
         A(I,K1) = A(I,K1) + T*V1
         A(I,K) = A(I,K) + T*V2
         T = B(I,K1) + U2*B(I,K)
         B(I,K1) = B(I,K1) + T*V1
         B(I,K) = B(I,K) + T*V2
350   CONTINUE
      B(K1,K) = 0.
      IF (.NOT.WANTX) GO TO 360
      DO 355 I=1,N
```

```
                T = X(I,K1) + U2*X(I,K)
                X(I,K1) = X(I,K1) + T*V1
                X(I,K) = X(I,K) + T*V2
  355       CONTINUE
  360   CONTINUE
C
C       END MAIN LOOP
C
        GO TO 200
C
C       END ONE QZ STEP
C
  380   DO 385 I=1,M
           ITER(I) = -1
  385   CONTINUE
  390   CONTINUE
        RETURN
        END
C
C
```

```
      SUBROUTINE QZVAL(ND,N,A,B,EPSA,EPSB,ALFR,ALFI,BETA,WANTX,X)
      DIMENSION A(ND,ND),B(ND,ND),ALFR(N),ALFI(N),BETA(N),X(ND,ND)
      LOGICAL WANTX,FLIP
C
C  FIND EIGENVALUES OF QUASI-TRIANGULAR MATRICES
C
C     DO 400 THRU 490 FOR M = N STEP (-1 OR -2) UNTIL 1
C
      M = N
  400 CONTINUE
         IF (M.EQ.1) GO TO 410
         IF (A(M,M-1).NE.0.) GO TO 420
C
C        ONE-BY-ONE SUBMATRIX, ONE REAL ROOT
C
  410    ALFR(M) = A(M,M)
         BETA(M) = B(M,M)
         ALFI(M) = 0.
         M = M-1
         GO TO 490
C
C        TWO-BY-TWO SUBMATRIX
C
  420    L = M-1
         IF (ABS(B(L,L)).GT.EPSB) GO TO 425
            B(L,L) = 0.
            CALL HSH2(A(L,L),A(M,L),U1,U2,V1,V2)
            GO TO 460
  425    IF (ABS(B(M,M)).GT.EPSB) GO TO 430
            B(M,M) = 0.
            CALL HSH2(A(M,M),A(M,L),U1,U2,V1,V2)
            BN = 0.
            GO TO 435
  430    AN = ABS(A(L,L))+ABS(A(L,M))+ABS(A(M,L))+ABS(A(M,M))
         BN = ABS(B(L,L))+ABS(B(L,M))+ABS(B(M,M))
         A11 = A(L,L)/AN
         A12 = A(L,M)/AN
         A21 = A(M,L)/AN
         A22 = A(M,M)/AN
         B11 = B(L,L)/BN
         B12 = B(L,M)/BN
         B22 = B(M,M)/BN
         C = (A11*B22 + A22*B11 - A21*B12)/2.
         D = (A22*B11 - A11*B22 - A21*B12)**2/4.
     1       + A21*B22*(A12*B11 - A11*B12)
         IF (D.LT.0.) GO TO 480
C
C        TWO REAL ROOTS
C        ZERO BOTH A(M,L) AND B(M,L)
C
         IF (C.GE.0.) E = (C + SQRT(D))/(B11*B22)
         IF (C.LT.0.) E = (C - SQRT(D))/(B11*B22)
         A11 = A11 - E*B11
         A12 = A12 - E*B12
         A22 = A22 - E*B22
         FLIP = (ABS(A11)+ABS(A12)).GE.(ABS(A21)+ABS(A22))
         IF (FLIP) CALL HSH2(A12,A11,U1,U2,V1,V2)
         IF (.NOT.FLIP) CALL HSH2(A22,A21,U1,U2,V1,V2)
```

```
  435     IF (U1.NE.1.) GO TO 450
          DO 440 I=1,M
             T = A(I,M) + U2*A(I,L)
             A(I,M) = A(I,M) + V1*T
             A(I,L) = A(I,L) + V2*T
             T = B(I,M) + U2*B(I,L)
             B(I,M) = B(I,M) + V1*T
             B(I,L) = B(I,L) + V2*T
  440     CONTINUE
          IF (.NOT.WANTX) GO TO 450
          DO 445 I=1,N
             T = X(I,M) + U2*X(I,L)
             X(I,M) = X(I,M) + V1*T
             X(I,L) = X(I,L) + V2*T
  445     CONTINUE
  450     IF (BN.EQ.0.) GO TO 475
          FLIP = AN .GE. ABS(E)*BN
          IF (FLIP) CALL HSH2(B(L,L),B(M,L),U1,U2,V1,V2)
          IF (.NOT.FLIP) CALL HSH2(A(L,L),A(M,L),U1,U2,V1,V2)
  460     IF (U1.NE.1.) GO TO 475
          DO 470 J=L,N
             T = A(L,J) + U2*A(M,J)
             A(L,J) = A(L,J) + V1*T
             A(M,J) = A(M,J) + V2*T
             T = B(L,J) + U2*B(M,J)
             B(L,J) = B(L,J) + V1*T
             B(M,J) = B(M,J) + V2*T
  470     CONTINUE
  475     A(M,L) = 0.
          B(M,L) = 0.
          ALFR(L) = A(L,L)
          ALFR(M) = A(M,M)
          BETA(L) = B(L,L)
          BETA(M) = B(M,M)
          ALFI(M) = 0.
          ALFI(L) = 0.
          M = M-2
          GO TO 490
C
C         TWO COMPLEX ROOTS
C
  490     ER = C/(B11*B22)
          EI = SQRT(-D)/(B11*B22)
          A11R = A11 - ER*B11
          A11I = EI*B11
          A12R = A12 - ER*B12
          A12I = EI*B12
          A21R = A21
          A21I = 0.
          A22R = A22 - ER*B22
          A22I = EI*B22
          FLIP = (ABS(A11R)+ABS(A11I)+ABS(A12R)+ABS(A12I)) .GE.
         1           (ABS(A21R)+ABS(A22R)+ABS(A22I))
          IF (FLIP) CALL CHSH2(A12R,A12I,-A11R,-A11I,CZ,SZR,SZI)
          IF (.NOT.FLIP) CALL CHSH2(A22R,A22I,-A21R,-A21I,CZ,SZR,SZI)
          FLIP = AN .GE. (ABS(ER)+ABS(EI))*BN
          IF (FLIP) CALL CHSH2(CZ*B11+SZR*B12, SZI*B12,
         1              SZR*B22, SZI*B22, CQ, SQR, SQI)
```

```
      IF (.NOT.FLIP) CALL CHSH2(CZ*A11+SZR*A12, SZI*A12,
     1              CZ*A21+SZR*A22, SZI*A22, CQ, SQR, SQI)
      SSR = SQR*SZR + SQI*SZI
      SSI = SQR*SZI - SQI*SZR
      TR = CQ*CZ*A11 + CQ*SZR*A12 + SQR*CZ*A21 + SSR*A22
      TI = CQ*SZI*A12 - SQI*CZ*A21 + SSI*A22
      BDR = CQ*CZ*B11 + CQ*SZR*B12 + SSR*B22
      BDI = CQ*SZI*B12 + SSI*B22
      R = SQRT(BDR*BDR + BDI*BDI)
      BETA(L) = BN*R
      ALFR(L) = AN*(TR*BDR + TI*BDI)/R
      ALFI(L) = AN*(TR*BDI - TI*BDR)/R
      TR = SSR*A11 - SQR*CZ*A12 - CQ*SZR*A21 + CQ*CZ*A22
      TI =  - SSI*A11 - SQI*CZ*A12 + CQ*SZI*A21
      BDR = SSR*B11 - SQR*CZ*B12 + CQ*CZ*B22
      BDI =  - SSI*B11 - SQI*CZ*B12
      R = SQRT(BDR*BDR + BDI*BDI)
      BETA(M) = BN*R
      ALFR(M) = AN*(TR*BDR + TI*BDI)/R
      ALFI(M) = AN*(TR*BDI - TI*BDR)/R
      M = M-2
C
  490 IF (M.GT.0) GO TO 400
      RETURN
      END
C
C
```

```
      SUBROUTINE QZVEC(ND,N,A,B,EPSA,EPSB,ALFR,ALFI,BETA,X)
      DIMENSION A(ND,ND),B(ND,ND),ALFR(N),ALFI(N),BETA(N),X(ND,ND)
      LOGICAL WANTX,FLIP
C
C  FIND EIGENVECTORS OF QUASI-TRIANGULAR MATRICES
C  USE B FOR INTERMEDIATE STORAGE
C
C     DO 500 THRU 590 FOR M = N STEP (-1 OR -2) UNTIL 1
C
      M = N
  500 CONTINUE
         IF (ALFI(M).NE.0.) GO TO 550
C
C        REAL VECTOR
C
         ALFM = ALFR(M)
         BETM = BETA(M)
         IF (ABS(ALFM).LT.EPSA) ALFM = 0.
         IF (ABS(BETM).LT.EPSB) BETM = 0.
         B(M,M) = 1.
C
C        DO 510 THRU 540 FOR L = M-1 STEP (-1 OR -2) UNTIL 1
C
         L = M-1
         IF (L.EQ.0) GO TO 540
  510    CONTINUE
            L1 = L+1
            SL = 0.
            DO 515 J=L1,M
               SL = SL + (BETM*A(L,J)-ALFM*B(L,J))*B(J,M)
  515       CONTINUE
            IF (L.EQ.1) GO TO 520
            IF (A(L,L-1).NE.0.) GO TO 530
  520       D = BETM*A(L,L)-ALFM*B(L,L)
            IF (D.EQ.0.) D = (EPSA+EPSB)/2.
            B(L,M) = -SL/D
            L = L-1
            GO TO 540
C
  530       K = L-1
            SK = 0.
            DO 535 J=L1,M
               SK = SK + (BETM*A(K,J)-ALFM*B(K,J))*B(J,M)
  535       CONTINUE
            TKK = BETM*A(K,K) - ALFM*B(K,K)
            TKL = BETM*A(K,L) - ALFM*B(K,L)
            TLK = BETM*A(L,K)
            TLL = BETM*A(L,L) - ALFM*B(L,L)
            D = TKK*TLL - TKL*TLK
            IF (D.EQ.0.) D = (EPSA+EPSB)/2.
            B(L,M) = (TLK*SK - TKK*SL)/D
            FLIP = ABS(TKK) .GE. ABS(TLK)
            IF (FLIP) B(K,M) = -(SK + TKL*B(L,M))/TKK
            IF (.NOT.FLIP) B(K,M) = -(SL + TLL*B(L,M))/TLK
            L = L-2
  540    IF (L.GT.0) GO TO 510
         M = M-1
         GO TO 590
```

```
C
C          COMPLEX VECTOR
C
  550     ALMR = ALFR(M-1)
          ALMI = ALFI(M-1)
          BETM = BETA(M-1)
          MR = M-1
          MI = M
C
C          NORMALIZE SO THAT M-TH COMPONENT = 0.-1.*I
C          (M-1)ST  = -(BETM*A(M,M)-ALFM*B(M,M))*(M-TH)/(BETM*A(M,M-1))
C
          B(M-1,MR) = ALMI*B(M,M)/(BETM*A(M,M-1))
          B(M-1,MI) = (BETM*A(M,M)-ALMR*B(M,M))/(BETM*A(M,M-1))
          B(M,MR) = 0.
          B(M,MI) = -1.
C
C          DO 560 THRU 585 FOR L = M-2 STEP (-1 OR -2) UNTIL 1
C
          L = M-2
          IF (L.EQ.0) GO TO 585
  560     CONTINUE
             L1 = L+1
             SLR = 0.
             SLI = 0.
             DO 565 J=L1,M
                TR = BETM*A(L,J) - ALMR*B(L,J)
                TI = -ALMI*B(L,J)
                SLR = SLR + TR*B(J,MR) - TI*B(J,MI)
                SLI = SLI + TR*B(J,MI) + TI*B(J,MR)
  565        CONTINUE
             IF (L.EQ.1) GO TO 570
             IF (A(L,L-1).NE.0.) GO TO 575
  570        DR = BETM*A(L,L) - ALMR*B(L,L)
             DI = -ALMI*B(L,L)
             CALL CDIV(-SLR, -SLI, DR, DI, B(L,MR), B(L,MI))
             L = L-1
             GO TO 585
C
  575        K = L-1
             SKR = 0.
             SKI = 0.
             DO 580 J=L1,M
                TR = BETM*A(K,J) - ALMR*B(K,J)
                TI = -ALMI*B(K,J)
                SKR = SKR + TR*B(J,MR) - TI*B(J,MI)
                SKI = SKI + TR*B(J,MI) + TI*B(J,MR)
  580        CONTINUE
             TKKR = BETM*A(K,K) - ALMR*B(K,K)
             TKKI = -ALMI*B(K,K)
             TKLR = BETM*A(K,L) - ALMR*B(K,L)
             TKLI = -ALMI*B(K,L)
             TLKR = BETM*A(L,K)
             TLKI = 0.
             TLLR = BETM*A(L,L) - ALMR*B(L,L)
             TLLI = -ALMI*B(L,L)
             DR = TKKR*TLLR - TKKI*TLLI - TKLR*TLKR
             DI = TKKR*TLLI + TKKI*TLLR - TKLI*TLKR
```

```
            IF(DR.EQ.0. .AND. DI.EQ.0.) DR = (EPSA+EPSB)/2.
            CALL CDIV(TLKR*SKR-TKKR*SLR+TKKI*SLI,
     1                TLKR*SKI-TKKR*SLI-TKKI*SLR,
     2                DR, DI, B(L,MR), B(L,MI))
            FLIP = (ABS(TKKR)+ABS(TKKI)) .GE. ABS(TLKR)
            IF (FLIP) CALL CDIV(-SKR-TKLR*B(L,MR)+TKLI*B(L,MI),
     1                          -SKI-TKLR*B(L,MI)-TKLI*B(L,MR),
     2                          TKKR, TKKI, B(K,MR), B(K,MI))
            IF (.NOT. FLIP) CALL CDIV(-SLR-TLLR*B(L,MR)+TLLI*B(L,MI),
     1                          -SLI-TLLR*B(L,MI)-TLLI*B(L,MR),
     2                          TLKR, TLKI, B(K,MR), B(K,MI))
            L = L-2
  585     IF (L.GT.0) GO TO 560
          M = M-2
  590   IF (M.GT.0) GO TO 500
C
C  TRANSFORM TO ORIGINAL COORDINATE SYSTEM
C
        M = N
  600   CONTINUE
          DO 620 I=1,N
            S = 0.
            DO 610 J=1,M
              S = S + X(I,J)*B(J,M)
  610       CONTINUE
            X(I,M) = S
  620     CONTINUE
          M = M-1
        IF (M.GT.0) GO TO 600
C
C  NORMALIZE SO THAT LARGEST COMPONENT = 1.
C
        M = N
  630   CONTINUE
          S = 0.
          IF (ALFI(M).NE.0.) GO TO 650
          DO 635 I=1,N
            R = ABS(X(I,M))
            IF (R.LT.S) GO TO 635
            S = R
            D = X(I,M)
  635     CONTINUE
          DO 640 I=1,N
            X(I,M) = X(I,M)/D
  640     CONTINUE
          M = M-1
          GO TO 690
C
  650     DO 655 I=1,N
            R = X(I,M-1)**2 + X(I,M)**2
            IF (R.LT.S) GO TO 655
            S = R
            DR = X(I,M-1)
            DI = X(I,M)
  655     CONTINUE
          DO 660 I=1,N
            CALL CDIV(X(I,M-1),X(I,M),DR,DI,X(I,M-1),X(I,M))
  660     CONTINUE
```

```
            M = M-2
  690 IF (M.GT.0) GO TO 630
C
  700 RETURN
      END
C
C
```

```
      SUBROUTINE HSH3(A1,A2,A3,U1,U2,U3,V1,V2,V3)
C
C  FINDS HOUSEHOLDER TRANSFORMATION THAT WILL ZERO A2 AND A3
C  P = I + (V1,V2,V3)*(U1,U2,U3)**T
C
      IF (A2.EQ.0. .AND. A3.EQ.0.) GO TO 10
      S = ABS(A1) + ABS(A2) + ABS(A3)
      U1 = A1/S
      U2 = A2/S
      U3 = A3/S
      R = SQRT(U1*U1+U2*U2+U3*U3)
      IF (U1.LT.0.) R = -R
      V1 = -(U1 + R)/R
      V2 = -U2/R
      V3 = -U3/R
      U1 = 1.
      U2 = V2/V1
      U3 = V3/V1
      RETURN
   10 U1 = 0.
      RETURN
      END
C
C
```

```
      SUBROUTINE HSH2(A1,A2,U1,U2,V1,V2)
C
C  FINDS HOUSEHOLDER TRANSFORMATION THAT WILL ZERO A2
C  P = I + (V1,V2)*(U1,U2)**T
C
      IF (A2.EQ.0.) GO TO 10
      S = ABS(A1) + ABS(A2)
      U1 = A1/S
      U2 = A2/S
      R = SQRT(U1*U1+U2*U2)
      IF (U1.LT.0.) R = -R
      V1 = -(U1 + R)/R
      V2 = -U2/R
      U1 = 1.
      U2 = V2/V1
      RETURN
   10 U1 = 0.
      RETURN
      END
C
C
```

```
      SUBROUTINE CHSH2(A1R,A1I,A2R,A2I,C,SR,SI)
C
C  COMPLEX HOUSEHOLDER THAT WILL ZERO A2
C       (C  S*)
C P =   (S -C ) ,  C REAL,  S COMPLEX
C
      IF (A2R.EQ.0. .AND. A2I.EQ.0.) GO TO 10
      IF (A1R.EQ.0. .AND. A1I.EQ.0.) GO TO 20
      R = SQRT(A1R*A1R+A1I*A1I)
      C = R
      SR = (A1R*A2R+A1I*A2I)/R
      SI = (A1R*A2I-A1I*A2R)/R
      R = SQRT(C*C+SR*SR+SI*SI)
      C = C/R
      SR = SR/R
      SI = SI/R
      RETURN
   10 C = 1.
      SR = 0.
      SI = 0.
      RETURN
   20 C = 0.
      SR = 1.
      SI = 0.
      RETURN
      END
C
C
```

```
      SUBROUTINE CDIV(XR,XI,YR,YI,ZR,ZI)
C
C  COMPLEX DIVIDE.  Z = X/Y
C
      IF (ABS(YR).LT.ABS(YI)) GO TO 10
      WR = XR/YR
      WI = XI/YR
      VI = YI/YR
      D = 1. + VI*VI
      ZR = (WR + WI*VI)/D
      ZI = (WI - WR*VI)/D
      RETURN
   10 WR = XR/YI
      WI = XI/YI
      VR = YR/YI
      D = VR*VR + 1.
      ZR = (WR*VR + WI)/D
      ZI = (WI*VR - WR)/D
      RETURN
      END
```