NASA TECHNICAL NOTE

NASA TN D-7032
C. 1

NASA TN D-7032

ALLMAT: A TSS/360 FORTRAN IV SUBROUTINE FOR EIGENVALUES AND EIGENVECTORS OF A GENERAL COMPLEX MATRIX

by Gale Fair

Lewis Research Center
Cleveland, Ohio 44135

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • JANUARY 1971

1. Report No. NASA TN D-7032	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle ALLMAT: A TSS/360 FORTRAN IV SUBROUTINE FOR EIGENVALUES AND EIGENVECTORS OF A GENERAL COMPLEX MATRIX		5. Report Date January 1971
		6. Performing Organization Code
7. Author(s) Gale Fair		8. Performing Organization Report No. E-5885
		10. Work Unit No. 129-02
9. Performing Organization Name and Address Lewis Research Center National Aeronautics and Space Administration Cleveland, Ohio 44135		11. Contract or Grant No.
		13. Type of Report and Period Covered Technical Note
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546		14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

Subroutine ALLMAT is described and listed. ALLMAT computes the eigenvalues and eigenvectors of a general (non-Hermitian) complex matrix. The program uses the complex QR algorithm to compute eigenvalues and inverse iteration to compute eigenvectors. The user has the option of computing only the eigenvalues, if desired. An entry point EVDATA is available to provide the user with timing and accuracy information, as well as the number of iterations necessary for each eigenvalue and eigenvector.

17. Key Words (Suggested by Author(s)) Matrix algebra; Computer program, FORTRAN; Eigenvalue; Eigenvector; Hermitian matrix; QR Transformation; Inverst iteration, Wielandt		18. Distribution Statement Unclassified - unlimited	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 29	22. Price* $3.00

*For sale by the National Technical Information Service, Springfield, Virginia 22151

ALLMAT: A TSS/360 FORTRAN IV SUBROUTINE FOR EIGENVALUES AND EIGENVECTORS OF A GENERAL COMPLEX MATRIX

by Gale Fair

Lewis Research Center

SUMMARY

A subroutine is described and listed that computes the eigenvalues and eigenvectors of a general (non-Hermitian) complex matrix. The program, ALLMAT, uses the complex QR algorithm to compute eigenvalues and inverse iteration to compute eigenvectors. The user has the option of computing only the eigenvalues, if desired. An entry point EVDATA is available to provide the user with timing and accuracy information, as well as the number of iterations necessary for each eigenvalue and eigenvector.

INTRODUCTION

Many areas of physics, mathematics, statistics, and engineering require the eigenvalues and eigenvectors of square matrices. This area of numerical analysis, sometimes called the algebraic eigenvalue problem, holds a place that is as important as the more familiar areas such as numerical integration, curve fitting, and numerical integration of differential equations. A general library of subroutines for a computer installation is commonly limited to one such program, and quite often this one subroutine is of only limited applicability.

The ideal subroutine for the algebraic eigenvalue problem should have many features: it should be fast and accurate; it should give a matrix of eigenvectors that are linearly independent; it should be capable of computing only eigenvalues at a corresponding increase in speed; it should have minimal storage requirements; and it must be able to treat all matrices, real or complex, symmetric or nonsymmetric, regardless of the condition of the matrix. Unfortunately, probably no such procedure exists. The usefulness of any subroutine may be judged on the basis of how many of these criteria are fulfilled, as balanced against the needs of the individual user.

There are many different techniques for diagonalizing a square matrix. The treatise by Wilkinson (ref. 1) is evidence for this. Reference 1 describes the state-of-the-art for the algebraic eigenvalue problem as of 1965. Typically, one chooses a particular method because he believes that his matrix has some feature that requires special handling, because a subroutine is conveniently at hand, or because he knows only that one method. Two of the most commonly used procedures are the power method and the Jacobi transformation.

The power method is a special-purpose procedure that computes the largest eigenvalue of a matrix by the formation of a sequence of powers of the matrix acting upon an arbitrary vector. This procedure is useful for the computation of a few eigenvalues (the largest in magnitude) and their eigenvectors. The computation of a full set of eigenvalues and eigenvectors is both time consuming and inaccurate.

The Jacobi transformation, as it applies to a complex Hermitian matrix, consists of a sequence of unitary transformations that diagonalize 2×2 submatrices of the full matrix. This procedure generates the eigenvectors along with the eigenvalues and is particularly useful when the eigenvectors are required to be orthogonal to a high degree of accuracy. The limitations of the Jacobi transformation are that the accuracy of the eigenvectors is usually limited, and as yet no extension to nonsymmetric or non-Hermitian matrices has been made. The most common computer library subroutine for the algebraic eigenvalue problem is a real-symmetric version of the Jacobi method (ref. 2).

For a general (i.e., nonsymmetric or non-Hermitian) matrix, two procedures have been derived to compute the eigenvalues and eigenvectors, respectively. The input matrix is reduced to a Hessenberg form (ref. 1), and the QR transformation of Francis (refs. 3 and 4) is used to compute the eigenvalues. With a knowledge of the eigenvalues, the Wielandt inverse iteration method (ref. 1) generates the eigenvectors. The QR transformation and inverse iteration appear to be the best currently available for their respective tasks (ref. 1) in terms of accuracy and speed. This combined procedure has been coded at the Oak Ridge National Laboratory (ref. 5) for an IBM 360/50 using the H-level FORTRAN compiler and COMPLEX*16 arithmetic. This program was used as the basis for the subroutine to be described in this report.

In order to make the subroutine as general as possible, some modifications and additions were made to the ORNL program, as follows. The fact that for some matrices the Hessenberg form may be decomposed into disjoint submatrices is incorporated in both the QR transformation and the inverse iteration to reduce computational time. A perturbation method is used to obtain linearly independent eigenvectors when eigenvalues are either degenerate or very nearly the same value (meaning that the matrix itself may be ill-conditioned (ref. 1)). An auxiliary entry point is provided to give the user information about the number of iterations required, timing data (measured as central processor or CPU time elapsed in the computation), and error data for the resulting eigenvalues and eigenvectors. Finally, a flag has been provided to allow the user to compute only the eigenvalues, with the use of the QR transformation. The relative contribution made by the present work is seen from the observation that approximately 60 percent of the coding of the current form of the subroutine ALLMAT is the ORNL coding while the remaining 40 percent is new.

The end result of the work described here is a subroutine for the IBM/360 to compute the eigenvalues and eigenvectors of a square matrix. Certainly, this is not meant to be the final work in such procedures, the algebraic eigenvalue problem is an area of extensive research in numerical analysis. On the other hand, this subroutine does satisfy most of the criteria mentioned earlier for the ideal subroutine, at least to some degree. The criterion that is least satisfied is minimal storage. Because ALLMAT is written with COMPLEX*16 arithmetic and has some large scratch-pad arrays, the subroutine uses a large amount of storage. On a TSS/360 system this storage requirement is not a basic limitation on the subroutine, but it does imply that the CPU time is affected.

This brief mention of storage requirements is an opportunity to interpose a slight warning to the prospective user of ALLMAT. If only a small number of the (largest) eigenvalues of a matrix are desired, the power method is more efficient than ALLMAT. For a real, symmetric matrix, a problem that requires eigenvectors along with the eigenvalues would be better suited to a real Jacobi subroutine. On the other hand, for the computation of eigenvalues alone, or for the eigenvalues and eigenvectors of a real, nonsymmetric matrix or for a complex matrix, ALLMAT seems to be the best choice, at this time.

The next section of this report describes schematically the construction of the subroutine ALLMAT. This includes the information necessary for a programmer to use ALLMAT. Also included are brief descriptions of the mathematical procedures used in ALLMAT. The following section discusses the special features that have been incorporated in ALLMAT, including a description of the subsidiary ENTRY EVDATA that provides timing and accuracy information for the user. Finally, a number of test matrices are used as examples for ALLMAT. These examples give an indication of running times and accuracy obtainable with the program, even with some ill-conditioned input matrices. A FORTRAN listing of ALLMAT is given in the appendix.

This report is intended to be used as a user's manual for the subroutine ALLMAT, and as such it described the call vector for the subroutine and the rules for usage. In addition, enough information is provided the prospective user to allow an intelligent application of this program to his particular problem. The prospective user should not apply this program to his problem without some understanding of the numerical methods involved and of the construction of the subroutine.

GENERAL CONSTRUCTION

Usage

The information to be discussed in this section is aimed at explaining the program as a FORTRAN subroutine, along with a description of the ENTRY EVDATA.

The user's access is through the statement (see the appendix for the complete FORTRAN listing of the subroutine):

CALL ALLMAT (AA, LAMBDA, M, MM, EVECT, NCAL)

where

AA	input COMPLEX*16 matrix, of dimension $M \leq MM$. Upon return from ALLMAT, i^{th} column of AA is i^{th} eigenvector, corresponding to i^{th} eigenvalue.
LAMBDA	COMPLEX*16 vector of length M that contains eigenvalues upon return from ALLMAT.
M	actual dimension of input matrix AA.
MM	dimension of AA as it appears in a dimension statement in the calling program. MM is the upper bound for the size of matrices used. As ALLMAT is currently written, MM must be no greater than 50.
EVECT	a logical switch. If EVECT = .TRUE., the eigenvectors of AA are calculated, and returned in the matrix AA. If EVECT = .FALSE., no eigenvectors are calculated and AA contains no useful information upon return from ALLMAT.
NCAL	number of eigenvalues successfully computed by ALLMAT. If NCAL $<$ M some attempts of the QR transformation did not converge within 10 iterations. The value of the element of LAMBDA that corresponds to this eigenvalue has been set to zero by ALLMAT.

In addition to the primary entry point, a secondary ENTRY EVDATA is available to give the user information on the CPU time taken for the eigenvalue and eigenvector procedures. Also available are the number of QR iterations required for each eigenvalue, the number of inverse iterations required for each eigenvector, and the Euclidean norms of the residual vectors. A more complete description of these quantities is given later. The usage for this optional entry point is

```
CALL EVDATA (ITS,KTS,NCO,MCO,RNORM)
```

where

ITS	elapsed time for QR transformation for eigenvalues, including time to reduce to upper Hessenberg form. ITS is an integer, in microminutes.
KTS	elapsed time for inverse iteration for eigenvectors. Does not include time represented by ITS. Also an integer in microminutes.
NCO	an integer vector of dimension MM that has as its i^{th} element the number of QR iterations for the i^{th} eigenvalue. NCO $(i) \leq 10$. If NCO $(i) = 0$, this eigenvalue was obtained along with another, no separate QR iteration was required. If NCO $(i) < 0$, no convergence was obtained for this eigenvalue within ten QR iterations.
MCO	integer vector of dimension MM that has as its i^{th} element the number of inverse iterations necessary to obtain the i^{th} eigenvector. MCO $(i) \leq 10$.
RNORM	REAL*8 vector of the norms of the residual vectors of AA. See section SPECIAL FEATURES OF ALLMAT for a more complete description. RNORM also has a dimension MM.

As an example of the usage of ALLMAT, consider a 6×6 complex matrix AA that is to be diagonalized. Let us assume that the TYPE statement in the calling program that specifies the dimensions of AA and LAMBDA has the form

```
COMPLEX*16 AA (10,10), LAMBDA(10)
```

The arrays have been overdimensioned for more generality. Let us further assume that eigenvectors are desired from ALLMAT, so that EVECT has been assigned a value .TRUE.. Then the call to ALLMAT is

```
CALL ALLMAT (AA,LAMBDA,6,10,EVECT,NCAL)
```

Upon return from ALLMAT the integer variable NCAL contains the number of eigenvalues that have been successfully computed by ALLMAT. The i^{th} column of AA (I.E.

AA (1, 1) to AA (6, 1)) contains the ith eigenvector, corresponding to the eigenvalue LAMBDA (1).

If the timing and error information provided by EVDATA are desired by the user, then the statement

```
CALL EVDATA (ITS,KTS,NCO,MCO,RNORM)
```

is used, where NCO, MCO, and RNORM have been dimensioned at least six in the calling program. The conversion from ITS or KTS (in microminutes) to milliseconds is obtained by multiplying either integer by 0.06 and assigning the result to a floating-point variable.

QR TRANSFORMATION

The basis of the QR transformation is a theorem by Francis that states any nonsingular matrix A has a unique decomposition into the product of a unitary matrix Q and an upper triangular matrix R (ref. 3), or

$$A = QR$$

The QR algorithm consists of forming a sequence of matrices similar to A $(=A_{(1)})$ such that

$$A_{(K)} = Q_{(K)} R_{(K)}$$

and then

$$A_{(K+1)} = R_{(K)} Q_{(K)}$$

where $A_{(K)}$ is the form of the matrix after the Kth decomposition. Francis (ref. 3) shows that this sequence of matrices has as its limit an upper triangular matrix, the diagonal elements of which are the eigenvalues of the original matrix A. Furthermore, even if the original matrix is singular, the algorithm still gives convergence to a unique triangular matrix, even though some of the intermediate Q and R may not be unique.

A full description of the QR transformation is certainly not relevant to this report. A detailed discussion of the convergence properties and the error analysis of the QR algorithm is given in references 1, 3, and 4. It is sufficient to note for our purpose that the QR algorithm is an extremely stable, rapidly converging procedure to calculate the eigenvalues of a general matrix (ref. 1). The version of the QR transformation that is part of ALLMAT, one that includes origin shifts to accelerate convergence, is powerful enough to satisfy nearly all of the needs of the average user.

There is one unusual feature of the standard way in which the QR algorithm is employed that the prospective user should be aware of. A preliminary step in any implementation of the QR transformation is the reduction of the input matrix to Hessenberg form. An upper Hessenberg form (i.e., $A_{ij} = 0$ if $i > j + 1$) is used in ALLMAT. The reduction is accomplished by a sequence of elementary transformations (ref. 1). The elements of these elementary transformations are stored in the unused portion of A (the lower subtriangle of A) and in the integer vector JNT. This information is used at the end of the inverse iteration to recover the eigenvectors of the original matrix from the eigenvectors of the Hessenberg matrix. The point of caution for the user is that the working matrix for the subroutine is the Hessenberg form, which in general bears no simple relation to the input matrix. Thus, if the user attempts to debug this subroutine at an intermediate stage, the relation between the Hessenberg form and the original form must ge kept in mind.

The advantage of using the Hessenberg form is apparent in the time needed to complete the computation of the eigenvalues. Most methods that operate on the entire input matrix, such as the Jacobi method, require a number of operations that is approximately $30N^3$ (ref. 2), where N is the order of the matrix. The reduction to Hessenberg form is a one-pass operation and requires αN^3 operations, where α is of order unity. The QR algorithm applied to the Hessenberg form only requires something of the order of N^2 operations. One interesting result of this is the observation (ref. 5) that under many conditions the QR transform produces eigenvalues in less time than the Jacobi transformation.

Inverse Iteration

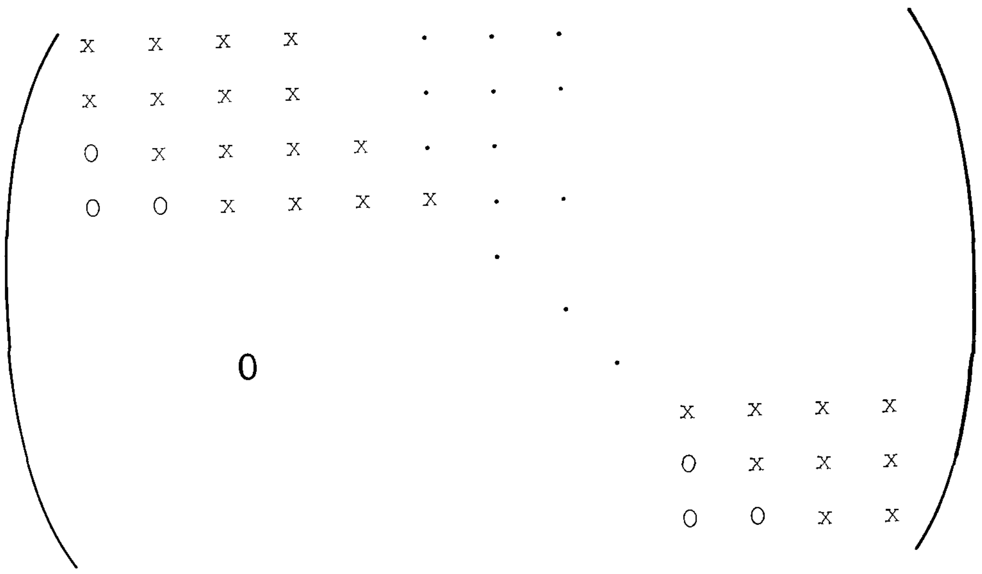
The basis of the inverse iteration procedure is the observation that, if λ is an eigenvalue of the matrix A, the quantity $(A - \lambda I)$, where I is the unit matrix, will be singular. Thus, if λ is a good approximation to an eigenvalue of A, the matrix $(A - \lambda I)^{-1}$ may be iterated to obtain an approximation Y to the eigenvector X. The iteration process is carried out until after the K^{th} iteration the norm of the iterated vector, $(A - \lambda I)^{-1} Y_K$ is greater than some preselected value (see the appendix). This procedure is equivalent to the power method, but in inverse powers of the matrix $(A - \lambda I)$. The speed with which this iteration produces an eigenvector depends on the accuracy of the estimate for the eigenvalue, but rarely does this procedure, combined with the QR algorithm, require more than 2 iterations to produce eigenvectors to at least six or seven place accuracy. Again, the interested user is referred to Wilkinson (ref. 1) for a complete description of the method and the error analysis.

SPECIAL FEATURES OF ALLMAT

As mentioned in the introduction, the basic elements of ALLMAT, the reduction to Hessenberg form, the QR transformation, and the inverse iteration, are taken from an ORNL subroutine (ref. 5). There are several features that have been added to this basic program to either add effectiveness to the program or provide timing and accuracy information to the user. These special features will be discussed in this section, more or less in the order that they appear in the program.

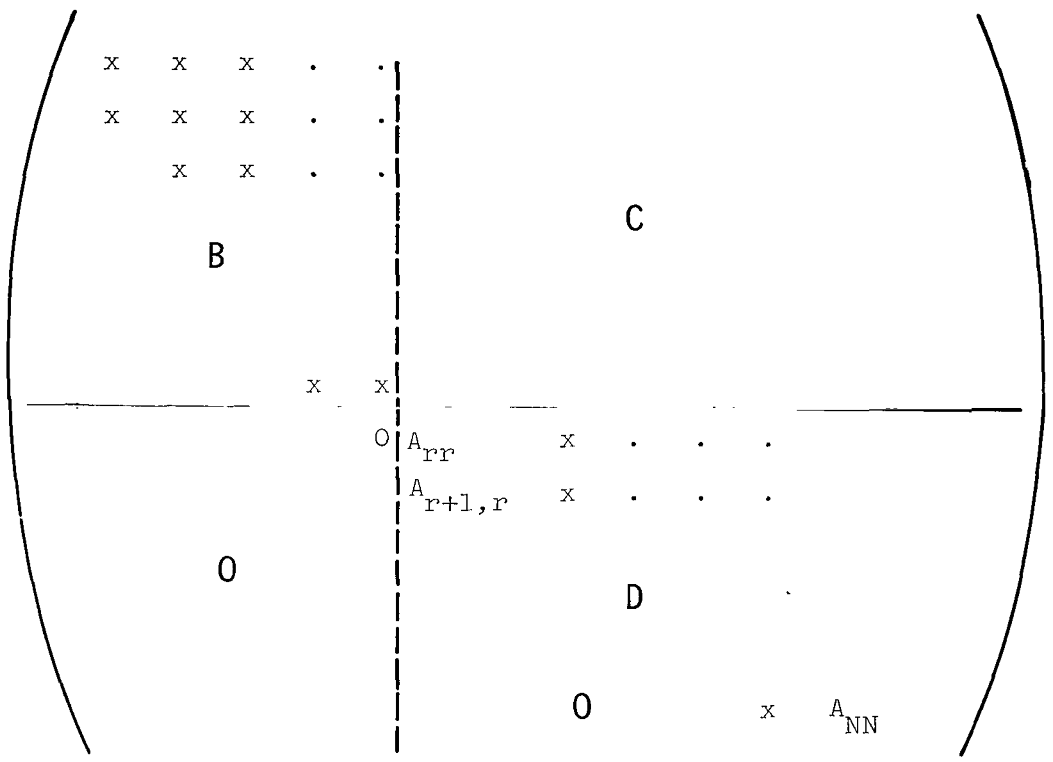
Decomposed Hessenberg Form

The reduction of the original matrix to Hessenberg form is a procedure that decreases the number of operations necessary for the QR algorithm. In a large number of cases the nature of the Hessenberg form allows further simplifications. To illustrate this, sketch (a) shows an upper Hessenberg matrix, of order N. The X's in the sketch



(a)

indicate matrix elements, generally nonzero, whose values are unimportant. Now let one of the subdiagonal elements vanish, for example $A(R, R - 1) = 0$. Then the Hessenberg matrix may be decomposed into four submatrices as shown in sketch (b).



(b)

The submatrices B and D are upper Hessenberg matrices of order $R - 1$ and $N - R + 1$, respectively. Submatrix C is a nonzero matrix with $N - R + 1$ columns and $R - 1$ rows. The remaining submatrix of this partition of A is entirely filled with zeroes.

The result of this decomposition is that the problem of finding the eigenvalues of B and D becomes entire disjoint; that is, the eigenvalues of B and D, collectively, are the eigenvalues of A. The submatrix C plays no part in the eigenvalue problem. Thus, instead of the solution of a single matrix of order N, the problem has been reduced to the solution of two matrices, of order $N - R + 1$ and $R - 1$. Since $N^2 > (N - R + 1)^2 + (R - 1)^2$ for $N \geq 3$ and R that is not trivial, this decomposition implies a significant reduction in the total number of operations in the QR transformation. For many input matrices, particularly those matrices that are sparse, a number of such decompositions may be performed and the gain in machine time is important.

This decomposition is not as important for the calculation of the eigenvectors, although some improvement is made. The eigenvectors corresponding to eigenvalues of D (see sketch (b)) depend upon the submatrices B and C, so that the entire matrix must be used in the inverse iteration procedure. The eigenvectors corresponding to the eigenvalues of B, on the other hand, do not require matrices C or D, so that only B is used in the inverse iteration. Thus, some advantage is gained from the decomposition for the calculation of the eigenvectors. On the whole, though, the main advantage of the decomposition enters in the QR transformation.

Perturbation of Close Eigenvalues

One difficulty with the inverse iteration method arises when two or more eigenvalues are very nearly the same. Since every calculated eigenvalue differs from the "true" eigenvalue by an amount that depends on many factors, these eigenvalues may not produce linearly independent eigenvectors. The way chosen to resolve this accidental degeneracy was to perturb each successive close eigenvalue by an amount small enough to not disturb the convergence of the iterative procedure, but large enough to resolve the eigenvectors into linearly independent vectors (ref. 1). The choice of the perturbation, EPSIL, is arbitrary and a better choice could be made for particular types of matrices.

Since the existence of close but distinct eigenvalues implies that the matrix may be ill-conditioned (ref. 1), the accuracy of the calculated eigenvectors will be in doubt. In this sense, the use of a perturbation to separate the eigenvalues is an attempt to recover some useful information from a badly posed problem. Thus, for most matrices encountered, the existence of close but distinct eigenvalues should be rare. The occurrence of multiple eigenvalues is more common.

Multiple Eigenvalues

The existence of a set of multiple eigenvalue is a not uncommon occurrence in physical problems. The existence of such a set implies that there is a subspace of eigenvectors that one desires the basis vectors of. In this situation the perturbation is of some help. If, by the process of perturbing the degenerate eigenvalues within the inverse iteration process, one can obtain a set of distinct eigenvectors, even if they are not linearly independent, then there is a standard solution to the problem of determining the basis vectors. For this purpose ALLMAT takes the set of distinct eigenvectors produced by the perturbation technique just discussed and uses a Gram-Schmidt (ref. 1) orthogonalization procedure to give a set of linearly independent eigenvectors. Since the Gram-Schmidt process involves taking the differences of nearly equal numbers in many cases, the accuracy of such a procedure is less than the accuracy of an inverse iteration vector for a distinct eigenvalue. Again, however, this represents an attempt to salvage as much information as one can from an undesirable situation. In practice, as shall be seen in the section TESTS, the results of this perturbation and orthogonalization procedure are good.

ENTRY EVDATA

The remaining special feature of ALLMAT is represented by the secondary entry point, EVDATA, as discussed in general construction. A typical user of an installation-supplied mathematical subroutine is usually blissfully unaware of any error considerations for his problem. Since the accuracy of any matrix eigenvalue evaluation strongly depends upon the properties of the input matrix, ignoring error information is equivalent to shutting one's eyes to avoid an oncoming truck. Additionally, since some eigenvalues and eigenvectors may in fact be absent due to nonconvergence either in QR or inverse iteration, the information provided by EVDATA is important to a user. The use of the TSS/FORTRAN multiple-data set capability means that this information is readily available to the user, without so much as the disturbance of an artistic output format.

The information available in EVDATA includes the number of iterations, the CPU time elapsed for the eigenvalue and the eigenvector computations, and an error estimate for each eigenvalue-eigenvector pair. The timing and counting variable provided in EVDATA were discussed sufficiently under usage, but the error information requires some further comment.

If λ and X are an exact eigenvalue and an exact eigenvector of the matrix A, then the vector $AX - \lambda X$ will be identically zero. Since neither λ nor X can ever be computed exactly, this vector $(AX - \lambda X)$, called the residual vector, will be nonzero. The magnitude of this vector is then a measure of the error in λ and X. The length of a vector, as used in ALLMAT, is the Euclidean norm, $||X|| = \left(\sum |x(i)|^2\right)^{1/2}$. The vector RNORM of EVDATA contains the norm of the residual vector for each eigenvalue-eigenvector pair, scaled to the Euclidean norm of the input matrix.

The data entry point EVDATA may be used even if no eigenvectors are computed (i.e., if EVECT = .FALSE.) In this case only ITS and NCO contain meaningful values.

TESTS

Seven matrices were chosen as examples for ALLMAT. The dimensions of these matrices vary from four to 19. All but two matrices are real but not symmetric, one of the remaining matrices is Hermitian, and the final example matrix is complex, but not Hermitian. Some of these matrices were chosen to illustrate ill-conditioning of one type or another. Since the numerical values of the eigenvectors are not of general use, they are not displayed.

Matrix 1

$$A_1 = \begin{pmatrix} 0.1 & -0.7 & -0.4 & -0.5 \\ -0.5 & 0.2 & -0.1 & -0.2 \\ 0.4 & 0.5 & 0.5 & 0.7 \\ 0.1 & 0.2 & 0.5 & 0.4 \end{pmatrix}$$

This real, but unsymmetric, matrix of order four has the exact eigenvalues 0.9, 0.6, -0.3, and 0. In addition, the computed eigenvalue corresponding to 0. is 0.14E-16. The information available from EVDATA on this test includes:

Eigenvalue	Number of QR iterations	Number of inverse iterations	RNORM
-0.3	8	3	0.65E-16
.14E-16	1	3	.43E-16
.6	1	3	.16E-16
.9	1	3	.91E-16

The total time for the QR transformation, including the initial reduction to Hessenberg form was 0.053 second, and the time for the inverse iteration was 0.046 second.

Matrix 2

$$A_2 = \begin{pmatrix} 0.25000025 & -1.0 & -0.49999975 & -0.99999975 \\ -0.50000050 & 0.5 & 0.24999950 & 0.25000025 \\ 1.00000025 & 1.25 & 1.00000025 & 1.25000025 \\ -0.4999975 & -0.25 & 0.25000025 & 0.50000025 \end{pmatrix}$$

This matrix has eigenvectors identical to those of matrix 1, but has a different set of eigenvalues. The exact eigenvalues of A_2 are given in the following table:

Eigenvalue	Number of QR iterations	Number of inverse iterations	RNORM
1.5	2	3	0.68E-16
.75000075	4	3	.10E-15
.75	0	3	.46E-14
-.75	1	3	.64E-15

The closeness of the second and third eigenvalues hint at some error problems with the eigenvectors. The time for QR transformation was 0.020 second, and the time for the inverse iteration was 0.090 second. The difficulties anticipated from the closeness of the eigenvalues are evidenced in the degradation of the third value of RNORM in the table.

Matrix 3

$$A_3 = \begin{pmatrix} 0.009 & 5.00101 & -8.999 & 3.999 \\ -0.001 & 5.01101 & -8.999 & 3.999 \\ -0.001 & 4.91101 & -8.899 & 3.999 \\ -0.001 & 4.96101 & -8.999 & 4.049 \end{pmatrix}$$

This matrix A_3 is an example of an ill conditioned matrix, in contrast with the previous example. Here, A_2 had two nearly alike eigenvalues even though the matrix is not mathematically ill conditioned (ref. 1).

Eigenvalue	Number of QR iterations	Number of inverse iterations	RNORM
0.01	3	3	0.25E-16
.01001	3	3	.14E-16
.1	3	3	.18E-16
.05	1	3	.23E-16

The time for QR transformation was 0.038 second; that for inverse iteration was 0.025 second. Apparently, the ill conditioning did not effect the inverse iterations, as all values of RNORM are satisfactory.

Matrix 4

$$A_4 = \begin{pmatrix} 6. & 5. & 4. & 3. & 2. & 1. \\ 1. & -2. & 1. & 6. & 3. & 2. \\ 2. & 3. & 2. & -2. & 4. & 3. \\ 3. & 1. & -3. & -1. & 5. & 5. \\ 4. & -4. & 2. & 0. & 1. & 4. \\ 5. & 0. & 1. & 3. & 6. & 6. \end{pmatrix}$$

Unlike the first three test matrices, A_4 has a pair of complex eigenvalues.

Eigenvalue	Number of QR iterations	Number of inverse iterations	RNORM
3.0929	6	3	0.78E-16
.1772+.95E-16i	6	3	.24E-15
.42295+4.3954i	5	3	.11E-15
.42295-4.3954i	4	3	.17E-15
15.247+.11E-14i	1	3	.56E-15
-7.3630	1	3	.47E-15

The time for QR transformation was 0.142 second; that for inverse iteration was 0.314 second. The imaginary part of the sum of the eigenvalues (which should be 0.) is 0.355E-14.

Matrix 5

This test matrix is a 19 by 19 real, unsymmetric matrix given by Francis (ref. 4) to demonstrate the QR transformation. The matrix is too complicated to list here, but the error information is informative. The time to produce the eigenvalues was 2 seconds, and the time to calculate the 19 eigenvectors was 22 seconds. Although this time is large when compared with the previous examples, it is quite reasonable when compared with other methods (ref. 5). Even with a matrix of this order, the residual vectors all had norms less than 1.E-16.

Matrix 6

```
            0.      -.91376103i  0.          0.          0.          0.          0.          0.          0.          0.
       .91376103i    0.          0.          0.          0.          0.          0.          0.          0.          0.
            0.       0.          0.          0.          0.          0.          0.          0.          0.          0.
            0.       0.          0.      -.38821306i     0.          0.      -.14433376i     0.          0.      -.94674090i
            0.       0.      .38821306i  0.              0.      -.14433376i     0.          0.      -.94674090i     0.
A  =        0.       0.          0.          0.          0.          0.          0.          0.          0.          0.
 6          0.       0.          0.      .14433376i      0.          0.      -1.02140683i    0.          0.      .03966810i
            0.       0.      .14433376i  0.              0.      1.02140683i     0.          0.      -.03966810i     0.
            0.       0.          0.      .94674090i      0.          0.      .03966810i      0.          0.      .74705540i
            0.       0.      .94674090i  0.              0.      -.03966810i     0.          0.      -.74705540i     0.
```

This matrix has several features that make it useful as an example. Each nonzero element of A_6 is a purely imaginary number and, in addition, A_6 is Hermitian. Thus, the eigenvalues of A_6 are real and, since the trace of A_6 vanishes, the eigenvalues occur in positive-negative pairs. There is a pair of degenerate eigenvalues with the value 0, so that the orthogonalization procedure must be used to obtain the eigenvectors. Finally, A_6 is sufficiently sparse that the decomposition of the Hessenberg form is effective in reducing the time required for the computations.

Eigenvalue	Number of QR iterations	Number of inverse iterations	RNORM
-0.91376103	3	3	0.26E-16
.91376103	3	3	.16E-16
-1.03872417	7	3	.31E-15
-.38452612	6	3	.96E-16
.38452612	5	3	.55E-15
1.03872417	4	3	.74E-15
-1.53711192	1	3	.15E-14
1.53711192	1	3	.21E-14
0	3	3	0
0	1	3	0

The time for QR transformation was 1.07 second; that for inverse iteration was 1.50 second. The eigenvalues appear in this table in the order in which they are calculated by the QR algorithm. Since the reduction to Hessenberg form and the use of the decomposed Hessenberg form rearrange the matrix, the eigenvalues are not computed in pairs, necessarily. This same effect caused the QR routine to take three iterations to compute a zero eigenvalue. The degenerate eigenvalues caused no loss of accuracy in the computation of the eigenvectors. Furthermore, the sum of the eigenvalues is purely imaginary, and has the magnitude 0.4E-14, reflecting the zero trace of A_6.

Matrix 7

The final example matrix was generated from matrix A_4 by taking each element of this 6 by 6 real, nonsymmetric matrix and multiplying by the imaginary unit i. The result, A_7, is a complex non-Hermitian matrix whose eigenvalues are the eigenvalues

Eigenvalue	Number of QR iterations	Number of inverse iterations	RNORM
3.0929i	6	3	0.77E-16
.95E-16+.1772i	6	3	.28E-15
4.3954+.42295i	5	3	.10E-15
-4.3954+.42295i	4	3	.16E-15
.44E-15-7.3630i	1	3	.48E-15
15.247i	1	3	.52E-15

of A_4 multiplied by i. The Qr transformation time was 0.140 second, and the inverse iteration time was 0.319 second.

A comparison of the results indicated in the preceding table with the results for example A_4 shows that ALLMAT handles the non-Hermitian form with comparable speed, at no loss of accuracy in the eigenvalues and eigenvectors. The sum of the eigenvalues is 0.31E-14+12.i.

These seven examples were chosen to be representative of the application of ALLMAT. Some matrices (A_1, A_4, A_5, and A_7) pose no particular problems, while the remaining (A_2, A_3, and A_6) were included to demonstrate one of more special characteristics of the program. It is seen from the results given above that ALLMAT had no difficulty with any of these test matrices. The norm of the residual vectors is typically less than 1.E-15, and all computed eigenvalues that were also known exactly were in agreement to at least 14 places. At no point did either the QR algorithm or the inverse iteration fail to give convergence within the allotted limit of 10 iterations. In fact, only once did the inverse iteration procedure require more than three iterations to satisfy the convergence criterion.

CONCLUDING REMARKS

This report is intended to be a user's guide for the prospective user of ALLMAT. The information presented here about the construction of ALLMAT should be considered a minimum for the use of this matrix eigenvector program. No program of the complexity of ALLMAT should be used without some understanding of the basic algorithms involved. Certainly, though, most users will apply ALLMAT without consideration of even the simplified discussion presented here. For these users the entry point EVDATA should be required usage as an indicator when ALLMAT does fail on a matrix.

Experience has shown that two inverse iterations are usually enough to give an eigenvector correct to sufficient accuracy. The current version of ALLMAT, however, iterates until the norm of the iterated vector, $(A - \lambda I)^{-1} X$, is greater than 1.E40. If computing time is at a premium, this criterion can be easily changed to a test on the number of iterations. The current limitation on ALLMAT is to matrices of dimension no larger than 50. This restriction may also be changed easily.

ALLMAT was designed to be a general purpose matrix eigenvalue and eigenvector subroutine. Almost any matrix, including the most general case of a complex, non-Hermitian matrix, is amenable to diagonalization by ALLMAT. Furthermore, timing test (ref. 5) indicate that the QR transform may be preferred to the Jacobi method for the eigenvalues of real and symmetric matrices.

Lewis Research Center,
National Aeronautics and Space Administration,
Cleveland, Ohio, September 16, 1970,
129-02.

APPENDIX - FORTRAN LISTING

```
      SUBROUTINE ALLMAT (AA,LAMBDA,M,MM,EVECT,NCAL)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMPLEX*16 AA(MM,MM)
C                                 6
C     IF THE USER REQUIRES DIMENSION LARGER THAN 50, THE
C      DIMENSIONS IN THE 2 LINES FOLLOWING THIS COMMENT MUST
C     BE CHANGED FROM 50 TO A SIZE TO SUIT THE USER.
C
      COMPLEX*16 A(50,50),H(50,50),HL(50,50),LAMBDA(MM)
      COMPLEX*16 VECT(50),MULT(50),SHIFT(3),TEMP,SINT,COST,TEMP1,TEMP2
      COMPLEX*16 EIG,CONJI
      LOGICAL EVECT,INTH(50)
      DIMENSION NCOUNT(50),MCOUNT(50)
      INTEGER JNT(50),R,RP1,RP2
      DO 1000 J=1,M
      DO 1000 I=1,M
 1000 A(I,J) = AA(I,J)
      CALL CPUTIM(ITIM)
      ITSUM = 0
      KTSUM = 0
      CONJI = (0.,-1.)
C
C     THE CONSTANT EPSIL DETERMINES THE CONVERGENCE OF THE
C     QR ALGORITHM, AND ALSO IS THE PERTURBATION PARAMETER
C     FOR THE INVERSE ITERATION.
C
      EPSIL = 1.0D-12
C
C     THE CONSTANT EPSMAX DETERMINES THE CONVERGENCE OF THE
C     INVERSE ITERATION.  THIS NUMBER IS THE LOGARITHM OF
C     NORM OF THE ITERATED EIGENVECTOR THAT IS SUFFICIENT
C     FOR CONVERGENCE.
C
      EPSMAX = 40.
      NSTOP = M
      N = NSTOP
      NSTART = 1
      MN1 = 1
      NCAL = 0
      IF (N.NE.1) GOTO 1
      LAMBDA(1) = A(1,1)
      A(1,1) = 1.0
      GO TO 92
```

```
    1 ICOUNT = 1
      SHIFT(1) = 0.
      IF (N.NE.2) GOTO 4
    2 NSP1 = NSTART + 1
      TEMP = (A(NSTART,NSTART)+A(NSP1,NSP1)+CDSQRT -
     1((A(NSTART,NSTART)+A(NSP1,NSP1))**2-4.*(A(NSP1, -
     1NSP1)*A(NSTART,NSTART)-A(NSP1,NSTART)*A(NSTART, -
     1NSP1))))/2.
      RELTEM = TEMP
      AMGTEM = CONJI*TEMP
      IF (RELTEM.NE.0..OR.AMGTEM.NE.0.) GOTO 3
      LAMBDA(NSTOP) = SHIFT(1)
      LAMBDA(MN1) = A(NSTART,NSTART)+A(NSP1,NSP1)+SHIFT(1)
      NCOUNT(NSTOP) = ICOUNT
      NCOUNT(MN1) = ICOUNT
      GO TO 37
    3 LAMBDA(NSTOP) = TEMP + SHIFT(1)
      LAMBDA(MN1)=(A(NSTART,NSTART)*A(NSP1,NSP1)- -
     1A(NSP1,NSTART)*A(NSTART,NSP1))/(LAMBDA(NSTOP) -
     2-SHIFT(1))+SHIFT(1)
      NCOUNT(NSTOP) = ICOUNT
      NCOUNT(MN1) = ICOUNT
      ICOUNT = 1
      GO TO 37
C
C       REDUCE MATRIX A TO HESSENBERG FORM.
C
    4 NM2 = N-2
      DO 15 R=1,NM2
      RP1 = R+1
      RP2 = R+2
      ABIG = 0.
      JNT(R) = RP1
      DO 5 I=RP1,N
      RELAIR = A(I,R)
      AMGAIR = CONJI*A(I,R)
      ABSSQ = RELAIR**2 + AMGAIR**2
      IF (ABSSQ.LE.ABIG) GOTO 5
      JNT(R) = I
      ABIG = ABSSQ
    5 CONTINUE
      INTER = JNT(R)
      IF (ABIG.EQ.0.) GOTO 15
      IF (INTER.EQ.RP1) GOTO 8
      DO 6 I=R,N
      TEMP = A(RP1,I)
      A(RP1,I) = A(INTER,I)
    6 A(INTER,I) = TEMP
      DO 7 I=1,N
      TEMP = A(I,RP1)
      A(I,RP1) = A(I,INTER)
    7 A(I,INTER) = TEMP
    8 DO 9 I=RP2,N
      MULT(I) = A(I,R)/A(RP1,R)
    9 A(I,R) = MULT(I)
      DO 11 I=1,RP1
      TEMP = 0.
      DO 10 J=RP2,N
   10 TEMP = TEMP + A(I,J)*MULT(J)
   11 A(I,RP1) = A(I,RP1) + TEMP
      DO 13 I=RP2,N
      TEMP = 0.
      DO 12 J=RP2,N
   12 TEMP = TEMP + A(I,J)*MULT(J)
   13 A(I,RP1) = A(I,RP1)+TEMP-MULT(I)*A(RP1,RP1)
      DO 14 I=RP2,N
      DO 14 J=RP2,N
   14 A(I,J) = A(I,J) - MULT(I)*A(RP1,J)
   15 CONTINUE
C
C      CALCULATE EPSILON.
C
      EPS = 0.
      DO 16 I=1,N
   16 EPS = EPS +CDABS(A(1,I))
      DO 18 I=2,N
      SUM = 0.
      IM1 = I - 1
      DO 17 J=IM1,N
   17 SUM = SUM + CDABS(A(I,J))
   18 IF(SUM.GT.EPS) EPS=SUM
      EPS = DSQRT(DFLOAT(N))*EPS*1.D-20
      IF (EPS.EQ.0.) EPS=1.D-20
      EPSIL = DMAX1(EPS,EPSIL)
C
C     SAVE THE HESSENBERG FORM IN THE ARRAY H.
C
   20 DO 19 I=1,N
      DO 19 J=1,N
   19 H(I,J) = A(I,J)
      NSM1 = NSTOP - 1
      IF (NSM1.NE.0) GOTO 100
      R = 1
C
C     START SCANNING FOR ZEROES IN THE SUB-DIAGONAL. THIS
C     DEFINES THE SUB-BLOCKS OF THE DECOMPOSED HESSENBERG
C     FORM.
C
      GO TO 102
  100 DO 101 I=1,NSM1
      R = NSTOP - I + 1
      RELAM1 = A(R,R-1)
      AMGAM1 = CONJI*A(R,R-1)
      IF ((DABS(RELAM1)+DABS(AMGAM1)).LE.EPSIL) GOTO 102
  101 CONTINUE
      R = 1
  102 NSTART = R
C
C     NSTART AND NSTOP ARE THE INDICES OF THE BEGINNING AND
C     END OF A DECOMPOSED HESSENBERG BLOCK.
C
      NS = NSTOP - NSTART + 1
      NC = NS
      MN1 = NSTOP + NSTART - N
  103 IF (NS.NE.1) GOTO 21
      LAMBDA(MN1) = A(NSTART,NSTART) + SHIFT(1)
      NCOUNT(MN1) = ICOUNT
      GO TO 37
   21 IF(NS.EQ.2) GOTO 2
 ✓ 22 RELANN = A(N,N)
 ✓    AMGANN = CONJI*A(N,N)
     -RLNNM1 = A(N,N-1)
     -AMNNM1 = CONJI*A(N,N-1)
      RLNDN1 = A(N,N-1)/A(N,N)
      AMNDN1 = CONJI*(A(N,N-1)/A(N,N))
      IF (RELANN.NE.0..OR.AMGANN.NE.0.) -
    - 1IF (DABS(RLNDN1) + DABS(AMNDN1)-1.D-18) 24,24,23
   23 IF (DABS(RLNNM1)+DABS(AMNNM1).GE.EPS) GOTO 25
   24 LAMBDA(MN1) = A(N,N) + SHIFT(1)
      NCOUNT(MN1) = ICOUNT
      ICOUNT = 1
      N = N - 1
      NS = NS - 1
      MN1 = MN1 + 1
      GO TO 21
C
C      DETERMINE SHIFT
C
   25 SHIFT(2)=(A(N-1,N-1)+A(N,N)+CDSQRT((A(N-1, -
     1N-1)+A(N,N))**2-4.*(A(N,N)*A(N-1,N-1)-A(N,N-1) -
     2*A(N-1,N))))/2.
      RELSHF = SHIFT(2)
      AMGSHF = CONJI*SHIFT(2)
      IF (RELSHF.NE.0..OR.AMGSHF.NE.0.) GOTO 26
      SHIFT(3) = A(N-1,N-1)+A(N,N)
      GO TO 27
   26 SHIFT(3)=(A(N,N)*A(N-1,N-1)-A(N,N-1)*A(N-1,N))/SHIFT(2)
   27 IF(CDABS(SHIFT(2)-A(N,N)).LT.CDABS(SHIFT(3) -
     1-A(N,N))) GO TO 28
      INDEX = 3
      GO TO 29
   28 INDEX = 2
   29 IF(CDABS(A(N-1,N-2)).GE.EPS) GOTO 30
      LAMBDA(MN1) = SHIFT(2) + SHIFT(1)
      LAMBDA(MN1+1) = SHIFT(3) + SHIFT(1)
      NCOUNT(MN1) = ICOUNT
      NCOUNT(MN1+1) = 0
      ICOUNT = 1
      N = N - 2
      NS = NS - 2
      MN1 = MN1 + 2
      GO TO 103
   30 SHIFT(1) = SHIFT(1) + SHIFT(INDEX)
      DO 31 I=NSTART,N
   31 A(I,I) = A(I,I) - SHIFT(INDEX)
C
C      PERFORM GIVENS ROTATIONS, QR ITERATES.
C
```

```
      IF (ICOUNT.LE.10) GOTO 32
      NCOUNT(MN1) = -ICOUNT
      NC = NC - NS
      GO TO 37
   32 NM1 = N - 1
      TEMP1 = A(NSTART,NSTART)
      TEMP2 = A(NSTART+1,NSTART)
      DO 36 R=NSTART,NM1
      NN = R
      RP1 = R + 1
      RELTM1 = TEMP1
      AMGTM1 = CONJI*TEMP1
      RELTM2 = TEMP2
      AMGTM2 = CONJI*TEMP2
      RHO = DSQRT(RELTM1**2+AMGTM1**2+RELTM2**2+AMGTM2**2)
      IF (RHO.EQ.0.) GOTO 36
      COST = TEMP1/RHO
      SINT = TEMP2/RHO
      INDEX = MAX0(NN-1,NSTART)
      DO 33 I=INDEX,N
      TEMP = DCONJG(COST)*A(NN,I)+DCONJG(SINT)*A(RP1,I)
      A(RP1,I) = -SINT*A(NN,I)+COST*A(RP1,I)
   33 A(NN,I) = TEMP
      TEMP1 = A(RP1,RP1)
      TEMP2 = A(NN+2,R+1)
      DO 34 I=NSTART,R
      TEMP=COST*A(I,NN)+SINT*A(I,RP1)
      A(I,RP1) = -DCONJG(SINT)*A(I,NN)+DCONJG(COST)*A(I,RP1)
   34 A(I,NN) = TEMP
      INDEX = MIN0(NN+2,N)
      DO 35 I=RP1,INDEX
      A(I,NN) = SINT*A(I,RP1)
   35 A(I,RP1)=DCONJG(COST)*A(I,RP1)
   36 CONTINUE
      ICOUNT = ICOUNT + 1
      GO TO 22
C
C      CALCULATE VECTORS.
C
   37 IF (.NOT.EVECT) GOTO 64
      CALL CPUTIM(JTIM)
      ITSUM = ITSUM + (JTIM - ITIM)
      IF (NC.EQ.0) GOTO 64
      NPNCAL = NSTART + NC - 1
      N = NSTOP
      NS = NSTOP - NSTART + 1
      NM1 = N - 1
      IF (N.NE.2) GO TO 38
      EPS = DMAX1(CDABS(LAMBDA(1)),CDABS(LAMBDA(2)))*1.D-16
      IF (EPS.EQ.0.) EPS=EPSIL
      H(1,1) = A(1,1)
      H(2,1) = A(2,1)
      H(1,2) = A(1,2)
      H(2,2) = A(2,2)
   38 DO 63 L=NSTART,NPNCAL
      ABIG = 0.
      EIG = LAMBDA(L)
      IF (L.EQ.NSTART) GOTO 40
      LM1 = L - 1
      RELEIG = EIG
      AMGEIG = CONJI*EIG
      DO 39 I=NSTART,LM1
      RELAMI = LAMBDA(I)
      AMGAMI = CONJI*LAMBDA(I)
      IF (DABS(RELEIG-RELAMI).GT.EPSIL) GOTO 39
      IF (DABS(AMGEIG-AMGAMI).GT.EPSIL) GOTO 39
      EIG = EIG + CONJI*EPSIL
   39 CONTINUE
   40 DO 42 I=1,N
      DO 41 J=1,N
   41 HL(J,I) = H(J,I)
   42 HL(I,I) = HL(I,I) - EIG
      DO 46 I=1,NM1
      MULT(I) = 0.
      INTH(I) = .FALSE.
      IP1 = I + 1
      IF (CDABS(HL(I+1,I)).LE.CDABS(HL(I,I))) GO TO 44
      INTH(I) = .TRUE.
      DO 43 J=I,N
      TEMP = HL(I+1,J)
      HL(I+1,J) = HL(I,J)
   43 HL(I,J) = TEMP
   44 RELHII = HL(I,I)
      AMGHII = CONJI*HL(I,I)
      IF (RELHII.EQ.0..AND.AMGHII.EQ.0.) GOTO 46
      MULT(I) = -HL(I+1,I)/HL(I,I)
      DO 45 J=IP1,N
   45 HL(I+1,J)=HL(I+1,J) + MULT(I)*HL(I,J)
   46 CONTINUE
      DO 48 I=1,N
   48 VECT(I) = 1.
      IF (NSTOP.EQ.M) GOTO 110
      NSTP1 = NSTOP + 1
      DO 47 I=NSTP1,M
   47 VECT(I) = 0.
  110 ICOUNT = 1
   49 RELHNN = HL(N,N)
      AMGHNN = CONJI*HL(N,N)
      IF (RELHNN.EQ.0..AND.AMGHNN.EQ.0.) HL(N,N)=EPS
      VECT(N) = VECT(N)/HL(N,N)
      DO 51 I=1,NM1
      K = N-I
      DO 50 J=K,NM1
   50 VECT(K) = VECT(K) - HL(K,J+1)*VECT(J+1)
      RELHKK = HL(K,K)
      AMGHKK = CONJI*HL(K,K)
      IF (RELHKK.EQ.0..AND.AMGHKK.EQ.0.) HL(K,K)=EPS
   51 VECT(K) = VECT(K)/HL(K,K)
      BIG = 0.
      DO 52 I=1,N
      RELVEC = VECT(I)
      AMGVEC = CONJI*VECT(I)
      SUM = DABS(RELVEC)+DABS(AMGVEC)
      IF (SUM.LE.BIG) GOTO 52
      BIG = SUM
      II = I
      RELV = RELVEC
      AMGV = AMGVEC
   52 CONTINUE
      IF (BIG.EQ.0.) GOTO 155
      IF (AMGV.EQ.0.) GOTO 135
      IF (DABS(AMGV).GT.DABS(RELV)) GOTO 125
      RAT = AMGV/RELV
      DEN = RELV + RAT*AMGV
      DO 120 I=1,N
      IF (I.EQ.II) GOTO 120
      RELVEC = VECT(I)
    * AMGVEC = CONJI*VECT(I)
      RELVC = (RELVEC + RAT*AMGVEC)/DEN
      AMGVC = (AMGVEC - RAT*RELVEC)/DEN
      VECT(I) = DCMPLX(RELVC,AMGVC)
  120 CONTINUE
      VECT(II) = 1.
      GO TO 150
  125 RAT = RELV/AMGV
      DEN = AMGV + RAT*RELV
      DO 130 I=1,N
      IF (I.EQ.II) GOTO 130
      RELVEC = VECT(I)
      AMGVEC = CONJI*VECT(I)
      RELVC = (AMGVEC + RAT*RELVEC)/DEN
      AMGVC = (RAT*AMGVEC - RELVEC)/DEN
      VECT(I) = DCMPLX(RELVC,AMGVC)
  130 CONTINUE
      VECT(II) = 1.
      GO TO 150
  135 DO  53 I=1,N
   53 VECT(I) = VECT(I)/BIG
  150 ABIG = ABIG + DLOG10(BIG)
      IF (ABIG.GT.EPSMAX) GOTO 55
  155 IF(ICOUNT.GE.10) GOTO 55
      DO 54 I=1,NM1
      IF (.NOT.INTH(I)) GOTO 54
      TEMP = VECT(I)
      VECT(I) = VECT(I+1)
      VECT(I+1) = TEMP
   54 VECT(I+1) = VECT(I+1)+MULT(I)*VECT(I)
      ICOUNT = ICOUNT + 1
      GO TO 49
   55 IF (M.LE.2) GOTO 69
      MCOUNT(L) = ICOUNT
      MM2 = M-2
      DO 57 I=1,MM2
      M1I = M-1-I
      MI1 = M-I+1
      DO 56 J=MI1,M
   56 VECT(J)=H(J,M1I)*VECT(M1I+1)+VECT(J)
      INDEX = JNT(M1I)
      TEMP = VECT(M1I+1)
      VECT(M1I+1) = VECT(INDEX)
   57 VECT(INDEX) = TEMP
C
C      NORMALIZE EIGENVECTOR.
C
   69 SUM = 0.
      DO 58 I=1,M
      RELVEC = VECT(I)
      AMGVEC = CONJI*VECT(I)
   58 SUM = SUM + RELVEC*RELVEC + AMGVEC*AMGVEC
      SUM = DSQRT(SUM)
      IF (SUM.EQ.0.) GO TO 60
      DO 59 I=1,M
   59 VECT(I) = VECT(I)/SUM
   60 CONTINUE
      DO 61 I=1,M
   61 A(I,L) = VECT(I)
      CALL CPUTIM(KTIM)
      KTSUM = KTSUM + (KTIM - JTIM)
   63 CONTINUE
      NCAL = NCAL + NC
   64 IF(NSTART.EQ.1) GOTO 70
      SHIFT(1) = 0.
      NSTOP = NSTART - 1
      N = NSTOP
      GO TO 20
   70 DO 80 L=2,M
      DO 79 I=1,M
   79 JNT(I) = 0
      RELAML = LAMBDA(L)
      AMGAML = CONJI*LAMBDA(L)
      LM1 = L - 1
      R = 0
      DO 71 I=1,LM1
      RELAMI = LAMBDA(I)
      AMGAMI = CONJI*LAMBDA(I)
      IF (DABS(RELAML-RELAMI).GT.EPS) GOTO 71
      IF (DABS(AMGAML-AMGAMI).GT.EPS) GOTO 71
      JNT(I) = L
      R = R + 1
   71 CONTINUE
      IF (R.EQ.0) GOTO 80
      DO 72 I=1,M
   72 VECT(I) = 0.
      DO 75 I=1,LM1
      IF (JNT(I).NE.L) GOTO 75
      TEMP = 0.
      DO 73 J=1,M
   73 TEMP = TEMP + DCONJG(A(J,L))*A(J,I)
      DO 74 J=1,M
   74 VECT(J) = VECT(J) + TEMP*A(J,I)
      IF (R.EQ.1) GOTO 76
      R = R - 1
   75 CONTINUE
   76 SUM = 0.
```

```
      DO 77 I=1,M
      A(I,L) = A(I,L) - VECT(I)
   77 SUM = SUM + A(I,L)*DCONJG(A(I,L))
      IF (SUM.EQ.0.) GOTO 80
      SUM = DSQRT(SUM)
      DO 78 I=1,M
   78 A(I,L) = A(I,L)/SUM
   80 CONTINUE
   92 DO 95 J=1,M
      DO 95 I=1,M
      TEMP = A(I,J)
      A(I,J) = AA(I,J)
   95 AA(I,J) = TEMP
      RETURN
      ENTRY EVDATA (ITS,KTS,NCO,MCO,RNORM)
      DIMENSION MCO(1),NCO(1),RNORM(1)
      DO 83 I=1,M
   83 NCO(I) = NCOUNT(I)
      ITS = ITSUM
      IF (.NOT.EVECT) RETURN
      DO 84 I=1,M
   84 MCO(I) = MCOUNT(I)
      ANORM = 0.
      DO 85 I=1,M
      DO 85 J=1,M
   85 ANORM = ANORM + A(J,I)*DCONJG(A(J,I))
      ANORM = DSQRT(ANORM)
      IF (ANORM.EQ.0.) ANORM=1.
      KTS = KTSUM
      DO 90 L=1,M
      VNORM = 0.
      DO 89 I=1,M
      TEMP = 0.
      DO 82 J=1,M
   82 TEMP = TEMP + A(I,J)*AA(J,L)
      TEMP = TEMP - LAMBDA(L)*AA(I,L)
   89 VNORM = VNORM + (CDABS(TEMP))**2
   90 RNORM(L) = DSQRT(VNORM)/ANORM
      RETURN
      END
```

REFERENCES

1. Wilkinson, J. H.: The Algebraic Eigenvalue Problem. Clarendon Press, Oxford, 1965.

2. Greenstadt, John: The Determination of the Characteristic Roots of a Matrix by the Jacobi Method. Mathematical Methods for Digital Computers. Anthony Ralston and Herbert S. Wilf, eds., John Wiley & Sons, Inc., 1960, pp. 84-91.

3. Francis, J. G. F.: The QR Transformation: A Unitary Analogue to the LR Transformation. I. Computer J., vol. 4, Oct. 1961, pp. 265-271.

4. Francis, J. G. F.: The QR Transformation. II. Computer J., vol. 4, 1962, pp. 332- 345.

5. Funderlic, R. E., ed.: The Programmer's Handbook: A Compendium of Numerical Analysis Utility Programs. Rep. K-1729, Union Carbide Nuclear Co., Feb. 9, 1968.

NASA-Langley, 1970 —— 8 E-5885

```
02U 001 33   51 3DS        71012  00903
AIR FORCE WEAPONS LABORATORY /WLOL/
KIRTLAND AFB, NEW MEXICO  87117


ATT E. LOU BOWMAN, CHIEF,TECH. LIBRARY
```

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958