Chapter 4

EISPACK
A PACKAGE FOR SOLVING MATRIX EIGENVALUE PROBLEMS

J. J. Dongarra *
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, Illinois 60439

C. B. Moler †
Department of Computer Science
University of New Mexico
Albuquerque, New Mexico 87131

INTRODUCTION

EISPACK is a collection of Fortran subroutines that compute the eigenvalues and eigenvectors of matrices and matrix systems. The package can determine the eigensystem of complex general, complex Hermitian, real general, real symmetric, real symmetric band, real symmetric tridiagonal, and special real tridiagonal matrices, and generalized real and generalized real symmetric matrix systems. In addition, there are two routines that compute the singular value decomposition, useful in solving certain least squares problems.

The subroutines are based mainly on the Algol procedures published in the *Handbook* series of Springer-Verlag by Wilkinson and Reinsch [1971] and the *QZ* algorithm of Moler and Stewart [1973]. The algorithms have been adapted to Fortran and thoroughly tested on a wide range of different computers. The software has been certified and is supported by the developers.

The software for EISPACK can be obtained from either

IMSL
Sixth Floor, NBC Building
7500 Bellaire Boulevard
Houston, Texas 77036
(Phone: 713-772-1927)
Cost: $75.00 (Tape included)

or

* This work was supported in part by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under Contract W-31-109-Eng-38.
† This work was supported in part by the Computer Science Section of the National Science Foundation under Contract MCS 7603297.

National Energy Software Center (NESC)
Argonne National Laboratory
9700 South Cass Avenue
Argonne, Illinois 60439
(Phone: 312-972-7250)
Cost: Determined by NESC policy.

Requesters in European Organization for Economic Cooperation and Development (OECD) countries may obtain the software from

NEA Data Bank
B.P. No. 9 (Bat. 45)
F-91191 Gif-sur-Yvette
France
Cost: Free.

The documentation for the codes is contained in the following books:

B.T. Smith, J.M. Boyle, J.J. Dongarra, B.S. Garbow,
Y. Ikebe, V.C. Klema, and C.B. Moler.
Matrix Eigensystem Routines — EISPACK Guide,
Lecture Notes in Computer Science, Volume 6, 2nd Edition,
Springer-Verlag (1976)
Price: $21.00

B.S. Garbow, J.M. Boyle, J.J. Dongarra, and C.B. Moler.
Matrix Eigensystem Routines-EISPACK Guide Extension,
Lecture Notes in Computer Science, Volume 51,
Springer-Verlag (1977)
Price: $20.00.

HISTORY

EISPACK is primarily based on Algol procedures developed in the 1960's by nineteen different authors and published in the journal *Numerische Mathematik.* J. H. Wilkinson and C. Reinsch edited a collection of these procedures, together with some background material, into a volume entitled *Linear Algebra* in the *Handbook for Automatic Computation* series. This volume was not designed to cover every possible method of solution. Algorithms were chosen on the basis of their generality, elegance, accuracy, speed, or economy of storage.

Prior to the actual publication of the *Handbook,* as the Wilkinson-Reinsch collection has come to be known, Virginia Klema and others at Argonne National Laboratory had begun translating many of the Algol procedures into Fortran.

In early 1970 a group of researchers from Argonne, the University of Texas, and Stanford University proposed to the National Science Foundation (NSF) a project to "explore the methodology, costs, and resources required to produce, test, and disseminate high-quality mathematical software and to test, certify, disseminate, and support packages of mathematical software in certain problem areas." The project, funded by NSF in 1971, came to be known as the NATS project. The initials originally stood for NSF, Argonne, Texas, and Stanford, but later for the National Activity to Test Software.

The participants decided very early in the project not to produce a comprehensive library of mathematical software but to concentrate on certain fundamental areas. The two areas that were selected — matrix eigensystems and functional approximation — were natural to the participating individuals and organizations (see Chapters 1 and 3).

During 1971 at a University of Michigan Summer Conference, Wilkinson presented material on the methods used to carry out the calculations involved in the eigenvalue problem, and Cleve Moler discussed the actual implementation of the algorithms into software. By this time the NATS project had developed Fortran versions of selected Algol procedures from the *Handbook.* These Fortran routines were sent to people who had an interest in this kind of software and had agreed to test the programs.

There were about 20 test sites representing universities, industrial organizations, and government laboratories and covering a wide range of computers and operating systems. The test sites were responsible for compiling and testing the EISPACK routines and making them available to site users. Reports on performance were sent to Argonne, which served as the NATS project center.

By May 1972 the collection of Fortran routines was ready for public use. Arrangements were made with the Argonne Code Center (now called the National Energy Software Center) to distribute the collection. The package was available in five versions: IBM 370-360, CDC 6600-7600, Univac 1108, Honeywell 635, and PDP-10. The package was said to be *certified* in the sense that information on testing was available and reports of poor or incorrect performance on the machines and operating systems it was tested on "would gain immediate attention from the developers."

In early 1974 the documentation for the first release of the software was completed. This documentation describes the usage of the various routines and gives detailed timing information on a wide variety of computers. In addition, the user guide describes in detail the alternative paths a user can choose to gain the most efficient solution for various problems.

Work had already started on extensions to the package in 1972. By 1976 the extensions had been tested and were ready for public distribution. The new package, which consisted of 70 routines, offered the capability of handling the generalized eigenvalue problem directly. An additional user guide was subsequently prepared and published to cover the new material in the second release of EISPACK, and the previous guide was updated.

To simplify the solution of the standard and generalized eigenproblems, the control program EISPAC was created. EISPAC is a program written in Fortran and IBM OS 360/370 assembly language and developed to aid users on IBM machines in solving their eigenvalue problems. The control program proved to be valuable for users on IBM machines during the first release of the package. But, because of the highly machine-dependent nature of the program and the addition of driver subroutines to EISPACK, interest in the program has diminished.

Since EISPACK was first made available in 1972, over 1000 copies of the collection have been distributed worldwide.

Recently (1983) some minor changes have been made to eliminate machine-dependent constants, reduce the possibilities of overflow and underflow, and incorporate the modifications of the Algol procedures recommended by Hammarling *et al.* [1981]. This new version, called Edition 3, is described at the end of this chapter.

The cost of developing the package is hard to measure. There are many facets of the project that cannot be assigned precise dollar values. The following figures give very rough estimates of costs [Cowell and Fosdick, 1977]:

EISPACK Edition 1
Size: 34 routines with 6,000 source cards
Duration: 34 months
Total effort: 112 man-months
Cost: $528,000

EISPACK Edition 2
Size: 70 routines with 11,130 source cards
Duration: 41 months (16 months simultaneously with Edition 1)

Total effort: 92 man-months
Cost: $371,000

EISPACK Edition 3
Size: 78 routines with 11,769 source cards
Duration: 12 months
Total effort: 10 man-months
Cost: $100,000.

By these estimates, the total cost of EISPACK is about one million dollars.

ORGANIZATION

The organization of EISPACK can be described from several different points of view. The simplest is that of a "casual" user — someone encountering EISPACK for the first time or someone wanting a quick and easy solution to a matrix eigenvalue problem. Such a user can concentrate on the driver subroutines.

There are 13 drivers, intended for matrices of different forms. Twelve of the drivers provide two options: compute all eigenvalues, or compute all eigenvalues and eigenvectors. One of the drivers provides for all the eigenvalues and some of the eigenvectors of a symmetric matrix. Seven of the drivers are for the "standard" eigenvalue problem involving a single real matrix, A, of various forms:

Driver	Problem	Matrix A
RG	$Ax = \lambda x$	general
RS	$Ax = \lambda x$	symmetric
RSM	$Ax = \lambda x$	symmetric; all values, some vectors
RSB	$Ax = \lambda x$	symmetric band
RSP	$Ax = \lambda x$	symmetric, packed*
RST	$Ax = \lambda x$	symmetric tridiagonal
RT	$Ax = \lambda x$	sign-symmetric tridiagonal†

Two of the drivers solve the standard eigenvalue problem for complex matrices:

* A packed array stores the lower triangle of an $n \times n$ real symmetric or complex Hermitian matrix in a one-dimensional array with only $n(n+1)/2$ elements.

† A sign-symmetric matrix is a real tridiagonal matrix whose offdiagonal elements have matching signs; that is, for all i, $sign(a_{i,i+1}) = sign(a_{i+1,i})$.

Driver	Problem	Matrix A
CG	$Ax = \lambda x$	general
CH	$Ax = \lambda x$	Hermitian

Four of the drivers solve the "generalized" eigenvalue problem involving two real matrices, A and B, without directly inverting either matrix:

Driver	Problem	Matrix A	Matrix B
RGG	$Ax = \lambda Bx$	general	general
RSG	$Ax = \lambda Bx$	symmetric	positive definite
RSGAB	$ABx = \lambda x$	symmetric	positive definite
RSGBA	$BAx = \lambda x$	symmetric	positive definite

The driver subroutines provide easy access to many of EISPACK's capabilities. The user who is satisfied with these capabilities, and whose problems do not make heavy demands on computer time or storage, need not be concerned with any further details of EISPACK organization.

The drivers are actually just "shell" subroutines which call from one to five other EISPACK subroutines to do the computations. Several of these other subroutines are used by more than one driver. For example, TQLRAT — a subroutine that computes all the eigenvalues of a real, symmetric, tridiagonal matrix — is used by several of the drivers. On the other hand, there are some subroutines that are not used by any of the drivers. These routines provide alternative methods for doing some of the computations, as well as specialized capabilities not covered by the drivers.

In addition to the drivers, there are 58 subroutines in EISPACK. This modular organization greatly reduces the amount of both source and object code that must be handled. It also provides opportunities for using EISPACK facilities in computations not envisioned during the original development. But it means that the user who desires to access these facilities is faced with a formidable list of subroutines. Since most of EISPACK consists of Fortran translations of the Wilkinson-Rensch collection, the names of these subroutines are the ones chosen by the original authors of the Algol procedures.

The efficient and accurate solution of a matrix eigenvalue problem usually involves at least two of the following steps:

Initial scaling: An operation known as "balancing" that is applied to

nonsymmetric matrices to reduce the roundoff errors in subsequent calculations.

Reduction: Similarity transformation to a matrix with zeros below the subdiagonal. The reduced matrix is known as a Hessenberg matrix. A Hessenberg matrix that is also symmetric has zeros above the first superdiagonal and thus is tridiagonal.

Eigenvalue iteration: Any of several iterative processes to compute the eigenvalues of the reduced matrix (which are the eigenvalues of the original matrix).

Eigenvector calculation: Any of several different methods to find eigenvectors of the reduced matrix.

Back transformation: Application of the inverse of the original reduction transformation to the matrix of eigenvectors.

Back scaling: Application of the inverse of the balancing transformation to the matrix of eigenvectors.

Table I summarizes the capabilities of the 58 EISPACK subroutines that are not drivers. The table is not intended to be a complete description, just a brief overview.

Table I

EISPACK Subroutines

Subroutine	Matrix form	Capability
BAKVEC	sign-symmetric	back transformation
BALANC	real	scaling
BALBAK	real	back scaling
BANDR	symmetric band	reduction
BANDV	symmetric band	some eigenvectors
BISECT	symmetric tridiagonal	some eigenvalues
BQR	symmetric band	some eigenvalues
CBABK2	complex	back scaling
CBAL	complex	scaling
CINVIT	complex Hessenberg	some eigenvectors
COMBAK	complex	back transformation
COMHES	complex	reduction

COMLR	complex Hessenberg	all eigenvalues
COMLR2	complex Hessenberg	all eigenvalues and vectors
COMQR	complex Hessenberg	all eigenvalues
COMQR2	complex Hessenberg	all eigenvalues and vectors
CORTB	complex	back transformation
CORTH	complex	reduction
ELMBAK	real	back transformation
ELMHES	real	reduction
ELTRAN	real	reduction
FIGI	sign-symmetric	reduction
FIGI2	sign-symmetric	reduction
HQR	real Hessenberg	all eigenvalues
HQR2	real Hessenberg	all eigenvalues and vectors
HTRIB3	Hermitian packed	back transformation
HTRIBK	Hermitian	back transformation
HTRID3	Hermitian packed	reduction
HTRIDI	Hermitian	reduction
IMTQL1	symmetric tridiagonal	all eigenvalues
IMTQL2	symmetric tridiagonal	all eigenvalues and vectors
IMTQLV	symmetric tridiagonal	all eigenvalues
INVIT	real Hessenberg	some eigenvectors
MINFIT	real	singular value decomposition
ORTBAK	real	back transformation
ORTHES	real	reduction
ORTRAN	real	reduction
QZHES	generalized	reduction
QZIT	generalized	reduction
QZVAL	generalized	all eigenvalues
QZVEC	generalized	all eigenvectors
RATQR	symmetric tridiagonal	some eigenvalues
REBAK	symmetric generalized	back transformation
REBAKB	symmetric generalized	back transformation
REDUC	symmetric generalized	reduction
REDUC2	symmetric generalized	reduction
SVD	real	singular value decomposition
TINVIT	symmetric tridiagonal	some eigenvectors
TQL1	symmetric tridiagonal	all eigenvalues
TQL2	symmetric tridiagonal	all eigenvalues and vectors
TQLRAT	symmetric tridiagonal	all eigenvalues
TRBAK1	symmetric	back transformation
TRBAK3	packed symmetric	back transformation
TRED1	symmetric	reduction
TRED2	symmetric	reduction
TRED3	packed symmetric	reduction

Any matrix can be triangularized by a unitary similarity transformation.

Most of the techniques employed in EISPACK are constructive realizations of variants of Schur's theorem. It is usually not possible to compute Schur's transformation with a finite number of rational arithmetic operations. Instead, the algorithms employ a potentially infinite sequence of similarity transformations

$$A_{k+1} = S_k^{-1} A_k S_k$$

for which A_{k+1} approaches an upper triangular matrix. The sequence is terminated when all of the subdiagonal elements of a particular A_{k+1} are less than the roundoff errors involved in the computation. These elements can then be set to zero without introducing any more perturbations in the eigenvalues than have already been caused by the previous transformations. The diagonal elements of the resulting A_{k+1} are then the desired approximations to the eigenvalues of the original matrix. The corresponding eigenvectors can be readily computed if they have been requested.

It is important for several reasons to have special algorithms that deal with real symmetric matrices:

- Many of the eigenvalue problems encountered in practice involve symmetric matrices.
- The eigenvalues of a symmetric matrix are real.
- The eigenvectors of a symmetric matrix can be chosen to be orthogonal.
- The eigenvalues of symmetric matrices are usually less sensitive to perturbation.
- Algorithms designed specifically for symmetric matrices have better convergence properties.
- Algorithms for symmetric matrices usually require less computer time and storage.

The only similarity transformations that also preserve symmetry are those based on orthogonal matrices. The following example outlines the basic steps used in the algorithms employed by EISPACK's driver subroutine RS, which computes all the eigenvalues and optionally all the eigenvectors of a real symmetric matrix. The input matrix is

Any matrix can be triangularized by a unitary similarity transformation.

Most of the techniques employed in EISPACK are constructive realizations of variants of Schur's theorem. It is usually not possible to compute Schur's transformation with a finite number of rational arithmetic operations. Instead, the algorithms employ a potentially infinite sequence of similarity transformations

$$A_{k+1} = S_k^{-1} A_k S_k$$

for which A_{k+1} approaches an upper triangular matrix. The sequence is terminated when all of the subdiagonal elements of a particular A_{k+1} are less than the roundoff errors involved in the computation. These elements can then be set to zero without introducing any more perturbations in the eigenvalues than have already been caused by the previous transformations. The diagonal elements of the resulting A_{k+1} are then the desired approximations to the eigenvalues of the original matrix. The corresponding eigenvectors can be readily computed if they have been requested.

It is important for several reasons to have special algorithms that deal with real symmetric matrices:

- Many of the eigenvalue problems encountered in practice involve symmetric matrices.
- The eigenvalues of a symmetric matrix are real.
- The eigenvectors of a symmetric matrix can be chosen to be orthogonal.
- The eigenvalues of symmetric matrices are usually less sensitive to perturbation.
- Algorithms designed specifically for symmetric matrices have better convergence properties.
- Algorithms for symmetric matrices usually require less computer time and storage.

The only similarity transformations that also preserve symmetry are those based on orthogonal matrices. The following example outlines the basic steps used in the algorithms employed by EISPACK's driver subroutine RS, which computes all the eigenvalues and optionally all the eigenvectors of a real symmetric matrix. The input matrix is

$$\begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 5 & 4 & 3 & 2 \\ 3 & 4 & 5 & 4 & 3 \\ 2 & 3 & 4 & 5 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}.$$

The initial orthogonal similarity transformations are carried out by subroutine TRED1 or TRED2. An $n \times n$ matrix requires $n-2$ transformations, each of which introduces zeros into a particular row and column of the matrix, while preserving symmetry and preserving the zeros introduced by previous transformations. In the case of our 5×5 example, the result of the first transformation is a matrix that has three zeros in the last row and column. The next transformation introduces two more zeros in the fourth row and column. The final transformation places one more zero in the third row and column:

$$\begin{bmatrix} 0.6594 & -0.1438 & 0 & 0 & 0 \\ -0.1438 & 0.9687 & 0.5678 & 0 & 0 \\ 0 & 0.5678 & 5.3052 & 4.4192 & 0 \\ 0 & 0 & 4.4192 & 13.0667 & -5.4772 \\ 0 & 0 & 0 & -5.4772 & 5.0000 \end{bmatrix}.$$

Since the result of the initial reduction is a symmetric tridiagonal matrix, it can be stored in just two vectors: one with n components for the diagonal and one with $n-1$ components for the offdiagonal. Subroutine TRED1 returns these two vectors only. Subroutine TRED2 also returns the orthogonal matrix that transforms the original matrix to this tridiagonal matrix.

EISPACK includes several routines for computing the eigenvalues of a real, symmetric tridiagonal matrix. Many, but not all, of these routines are variants of the *QR* algorithm, originally published by J. G. F. Francis in 1961 and perfected by Wilkinson, Reinsch, Dubrulle, and other authors of chapters in the *Handbook*. The driver RS uses two variants of the *QR* algorithm: subroutine TQLRAT if the eigenvectors are not requested and subroutine TQL2 if they are. (For technical reasons associated with the scaling of so-called "graded" matrices, the indexing within the algorithm is carried out in the opposite order from the original Francis algorithm; and the resulting version is the *QL*, rather than the *QR*, algorithm. The subroutine names thus reflect details that are of interest to the numerical analysts who were the intended audience of the papers in *Numerische Mathematik,* but that are not important to most users of EISPACK.)

The symmetric, tridiagonal QR algorithm produces a sequence of similar matrices whose offdiagonal elements are decreasing in magnitude and whose diagonal elements are approaching the desired eigenvalues. With the QL variant, the shift calculation is intended to reduce the first offdiagonal element most rapidly. The first three iterates are

$$\begin{bmatrix} 0.5667 & 0.0609 & 0 & 0 & 0 \\ 0.0609 & 0.7645 & 0.0989 & 0 & 0 \\ 0 & 0.0989 & 1.3832 & 0.7501 & 0 \\ 0 & 0 & 0.7501 & 7.0326 & -4.4112 \\ 0 & 0 & 0 & -4.4112 & 15.2530 \end{bmatrix}$$

$$\begin{bmatrix} 0.5484 & -0.0003 & 0 & 0 & 0 \\ -0.0003 & 0.7655 & 0.0285 & 0 & 0 \\ 0 & 0.0285 & 1.2751 & 0.1085 & 0 \\ 0 & 0 & 0.1085 & 5.4084 & -1.4364 \\ 0 & 0 & 0 & -1.4364 & 17.0025 \end{bmatrix}$$

$$\begin{bmatrix} 0.5484 & 0.0000 & 0 & 0 & 0 \\ 0.0000 & 0.7641 & 0.0085 & 0 & 0 \\ 0 & 0.0085 & 1.2737 & 0.0167 & 0 \\ 0 & 0 & 0.0167 & 5.2501 & -0.4103 \\ 0 & 0 & 0 & -0.4103 & 17.1637 \end{bmatrix}.$$

Notice that all the offdiagonal elements have generally decreased with each iteration and that the first offdiagonal element has decreased a great deal. Indeed, the first offdiagonal element has now reached a size that is comparable to the roundoff errors made during the calculation. Thus, setting it to zero can be regarded as simply another roundoff error. The first diagonal element is now an accurate approximation to one of the eigenvalues of the original matrix.

The next two iterations affect only the lower 4×4 submatrix:

$$\begin{bmatrix} 0.5484 & 0 & 0 & 0 & 0 \\ 0 & 0.7639 & -0.0000 & 0 & 0 \\ 0 & -0.0000 & 1.2738 & 0.0003 & 0 \\ 0 & 0 & 0.0003 & 5.2362 & -0.0315 \\ 0 & 0 & 0 & -0.0315 & 17.1777 \end{bmatrix}.$$

The second offdiagonal element is now negligible, and the second diagonal element approximates another eigenvalue. Three more iterations, which we do not show, are needed to reduce the remaining offdiagonal elements to roundoff level. The final diagonal matrix is

$$\begin{bmatrix} 0.5484 & 0 & 0 & 0 & 0 \\ 0 & 0.7639 & 0 & 0 & 0 \\ 0 & 0 & 1.2738 & 0 & 0 \\ 0 & 0 & 0 & 5.2361 & 0 \\ 0 & 0 & 0 & 0 & 17.1778 \end{bmatrix}.$$

In this example, a total of 11 similarity transformations were required, 3 for the initial reduction and 8 for the QR iterations. The later transformations involved considerably less arithmetic than the earlier ones because they were done on submatrices of decreasing order. Let S denote the product of all the orthogonal similarity transformations that are required, and let D denote the final diagonal matrix. Then

$$S^{-1}AS = D,$$

and hence

$$AS = SD.$$

This relation shows that the columns of S are the eigenvectors of A. Moreover, since S is the product of orthogonal matrices, it must also be orthogonal. Subroutines TRED2 and TQL2 accumulate eigenvectors S as a by-product of the original reduction and eigenvalue iteration. Since S is a full matrix, most of the execution time is used in its calculation. If only the eigenvalues are desired, then S need not be computed, and TRED1 and TQLRAT could be used instead.

Several of the papers and Algol procedures in the *Handbook* resulted from

research that refined and perfected the *QR* algorithm. A paper by Reinsch, which was published after the *Handbook* was completed, describes the fastest routine, TQLRAT. Many of these procedures were taken over to EISPACK, so that there are 10 different subroutines like TQLRAT and TQL2 which compute the eigenvalues of a symmetric tridiagonal matrix. Eight of the ten are variants of the *QR* algorithm; the other two employ bisection algorithms based on Sturm sequences. Ten or fifteen years ago, it was not clear which of these alternative approaches was preferable. Today, experience gained with EISPACK allows us to make such decisions.

Nonsymmetric matrices involve somewhat different techniques, although the general approach of an initial reduction followed by some *QR*-type iteration is still followed. Since there is no symmetry to be preserved, similarity transformations can be based on nonorthogonal matrices. Algorithms that employ elimination methods require less arithmetic and, hence, are potentially faster than those that use orthogonal transformations. However, such algorithms may produce somewhat less accurate results and, in extreme examples, may be completely unstable because of element growth. Deciding between the two classes of algorithms involves comparing execution speed with numerical reliability. When designing a general-purpose library, such decisions are very difficult to make.

The following example illustrates the usual behavior of the "real, general" driver RG. The input matrix is

$$\begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 9 & 5 & 4 & 3 & 2 \\ 8 & 9 & 5 & 4 & 3 \\ 7 & 8 & 9 & 5 & 4 \\ 6 & 7 & 8 & 9 & 5 \end{bmatrix}.$$

Subroutine ELMHES uses nonorthogonal elimination methods to produce the following Hessenberg matrix:

$$\begin{bmatrix} 5.0000 & 8.8889 & 4.0174 & 4.6055 & 2.0000 \\ 9.0000 & 12.2222 & 6.2902 & 6.4082 & 3.0000 \\ 0 & 16.2963 & 11.7891 & 10.9525 & 7.0000 \\ 0 & 0 & -2.5925 & -2.6545 & -1.9705 \\ 0 & 0 & 0 & 1.3901 & -1.3569 \end{bmatrix}.$$

In this case, there has been very little growth in the size of the elements during the initial reduction, so there has been very little roundoff error magnification.

RG now uses subroutine HQR2 to carry out an "implicit, double-step, Hessenberg" *QR* iteration. The shift calculations are designed to reduce the size of the last subdiagonal element. After two iterations we have:

$$\begin{bmatrix} 24.8549 & 10.9746 & 3.7663 & -13.3527 & 6.1188 \\ -0.2000 & 3.5305 & -1.7923 & 5.3528 & -5.6901 \\ 0 & 0.0538 & -1.2472 & -1.7953 & 0.0899 \\ 0 & 0 & 0.9872 & -0.5516 & -2.6015 \\ 0 & 0 & 0 & 0.2848 & -1.5867 \end{bmatrix}.$$

Notice that the first two subdiagonal elements are actually decreasing more rapidly than the last one, which is the target of the shift calculation. This is an important property of the *QR* algorithm: It effectively "works on" all eigenvalues simultaneously. Five more iterations lead to

$$\begin{bmatrix} 24.7514 & -11.1821 & -3.6155 & 12.9960 & 7.0641 \\ 0 & 3.6275 & -1.3330 & 4.9235 & 6.0036 \\ 0 & 0 & -1.2203 & -1.8985 & -0.4090 \\ 0 & 0 & 1.2977 & -0.6818 & 2.4406 \\ 0 & 0 & 0 & 0 & -1.4768 \end{bmatrix}.$$

Three of the eigenvalues are revealed in diagonal positions 1, 2, and 5. The 2×2 submatrix that includes diagonal positions 3 and 4 is the source of a pair of complex conjugate eigenvalues

$$-0.9511 \pm 1.5463i.$$

The entire computation has proceeded using real arithmetic, even though the final results are complex.

NUMERICAL PROPERTIES

The algorithms used by EISPACK that are based on orthogonal transformations are always numerically stable in the sense that they produce the exact answer to an eigenvalue problem involving a matrix $A + E$, which is a small perturbation of the given matrix A. The norm of the perturbation E is roughly the size of roundoff error when compared to the norm of A. The same can be said for the algorithms based on nonorthogonal transformations if no exceptional growth in the size of the matrix elements occurs during the computation.

One immediate consequence of this numerical stability is that the computed results will produce small residuals. If λ and x are a computed eigenvalue and eigenvector of a given matrix A, then Ax will always be close to λx. More precisely, the size of the *relative residual*

$$\frac{||Ax - \lambda x||}{||A||\,||x||}$$

can always be expected to be roughly equal to the relative accuracy of floating point arithmetic on the computer being used.

But what about accuracy? How close are the computed eigenvalues to the exact eigenvalues? The answers to these questions depend more on the matrix involved than on the particular EISPACK subroutine used to do the computation. To see why, assume that A has a complete, linearly independent set of eigenvectors X, and let D denote the diagonal matrix of eigenvalues. Then

$$X^{-1}AX = D.$$

Suppose that A is perturbed somehow, either by errors in its initial formation or by roundoff errors generated by EISPACK. Then

$$X^{-1}(A + E)X = D + X^{-1}EX.$$

The resulting perturbation to D is not diagonal, but this equation makes it plausible that the damage done by E to the eigenvalues in D could be as large as $||X^{-1}||\,||E||\,||X||$, rather than merely $||E||$. The quantity

$$\kappa(X) = ||X||\,||X^{-1}||,$$

which is the condition number of X, occurs in the perturbation analysis for systems of simultaneous equations. Note that here with the eigenvalue problem, it is the condition of X — the matrix of eigenvectors — and not the condition of A itself that is relevant. If the eigenvector matrix is nearly singular, then the eigenvalues are potentially sensitive to perturbation.

If A is real and symmetric — or more generally, if A commutes with its complex conjugate transpose — then X can be taken to be unitary and $\kappa(X)$ (with respect to the 2-norm) is 1. In this case, a small change in the matrix causes a correspondingly small change in the eigenvalues. In other words, the eigenvalues of such matrices are always well-conditioned.

In the extreme case where A does not have a full set of eigenvectors, so that its Jordan Canonical Form is not diagonal, then $\kappa(X)$ should be regarded as infinite. The eigenvalues are infinitely sensitive to perturbation in the sense that they are no longer analytic functions of that perturbation.

With EISPACK, the consequences of this perturbation theory are the following:

For real symmetric matrices and for complex Hermitian matrices, the eigenvalues are always computed with an accuracy that corresponds to a few units of roundoff error in the largest eigenvalue of the matrix. The small eigenvalues of such matrices will not necessarily be computed to high accuracy relative to themselves, but they will be computed to full accuracy relative to the norm of the matrix. For such matrices, the presence of multiple eigenvalues has little effect on the accuracy of the computed results.

For general, nonsymmetric matrices, the effect of roundoff errors on the computed eigenvalues will increase as the condition number of the eigenvector matrix increases. If a nonsymmetric matrix has multiple eigenvalues or is close to a matrix with multiple eigenvalues, and if its eigenvector matrix has a large condition number, then the computed eigenvalues may be accurate to less than full precision.

As an example, consider the following matrix:

$$A = \begin{bmatrix} -64 & 82 & 21 \\ 144 & -178 & -46 \\ -771 & 962 & 248 \end{bmatrix}.$$

This matrix was constructed in such a way that its exact eigenvalues happen to be 1, 2, and 3 . When the eigenvalues are computed using EISPACK subroutines on a computer with a 24-bit floating point fraction, the results are

$$\begin{matrix} 1.00195 \\ 2.00113 \\ 2.99736 \end{matrix} .$$

Such a computer has a relative floating point accuracy of better than 10^{-6}, so the computed eigenvalues have lost half the available figures.

The difficulties lie with the matrix itself, not with EISPACK. The matrix of computed eigenvectors, renormalized so that the last component of each vector is one, is

$$X = \begin{bmatrix} 0.090922 & 0.075114 & 0.111016 \\ -0.181810 & -0.196554 & -0.166741 \\ 1.000000 & 1.000000 & 1.000000 \end{bmatrix}.$$

The condition number of X is greater than 10^3. Thus a single roundoff error in

A, which on this computer affects the sixth significant figure, could cause changes in the third significant figure of the eigenvalues. EISPACK has computed the eigenvalues as accurately as is possible using floating point arithmetic of this precision.

As another measure of accuracy, let D be the matrix whose diagonal elements are the computed eigenvalues. Then

$$\frac{||AX - XD||}{||A||\ ||X||} = 0.13 \times 10^{-6}.$$

In other words, the relative residuals are on the order of roundoff error, even though the eigenvalues are "accurate" to only three figures. For further information on the perturbation theory for the eigenvalue problem, see Stewart [1973] and Wilkinson [1965].

EISPACK 3

In this section we describe the latest version of the package, EISPACK 3, a limited set of modifications to the second edition of EISPACK. The modifications eliminate the machine-dependent constants and reduce the probability of underflow/overflow difficulties. They also may improve the execution time of a few subroutines. However, they do not introduce any new capabilities, nor do they change any calling sequences. The resulting collection is thus readily portable from machine to machine and somewhat more robust and efficient, but the two EISPACK guides continue to serve as the basic documentation.

Writing or revising code is the easy part of any software project. Proper testing, certification, and distribution were key features of the original EISPACK activity. We have used the same procedures in testing this version as in the past.

Until the current release, there has been no "official" double precision version of EISPACK. Several of the different machine versions are single precision. The IBM version uses the nonstandard declaration REAL*8 because we regard this as the appropriate working precision for such machines and did not want to call it "double." We now give in to generally accepted usage and provide two machine-independent versions, one for single precision and one for double precision.

Three routines which use inverse iteration to find selected eigenvectors of symmetric matrices — BANDV, TINVIT and TSTURM — have been modified in a way that may reduce the size of the groups involved in the reorthogonalization process. This should make the routines significantly faster for matrices where the original grouping criterion produced larger groups than necessary. The size of these groups also affects the orthogonality of the computed eigenvectors,

but we expect that the new versions should still produce acceptable orthogonality. These inverse iteration routines were primarily intended to find just a few selected eigenvectors. But, particularly with this new grouping calculation, in many cases they also provide the *fastest* way to find *all* the eigenvectors.

A new driver for real symmetric matrices, RSM, has been added to the collection. It provides easy access to the recommended routines in the inverse iteration path. It has an additional integer parameter M which must be less than or equal to the order N . It computes all the eigenvalues, and the M eigenvectors associated with the M smallest eigenvalues, of a real symmetric matrix. It should be considered for use even when M = N so that all the eigenvectors are computed. It is difficult to summarize the tradeoffs between RSM and RS, the other real symmetric driver which computes all the eigenvectors. RSM is usually faster, and may be up to 40% faster on some matrices. The computed eigenvalues are usually of comparable accuracy. But the computed eigenvectors provided by RS may be somewhat more accurate in two senses: the residuals may be smaller, and the departure from orthogonality may be smaller.

Fortran standards require storage of two-dimensional arrays by column. Many modern computer systems employ cache or paged memories where access to columns of a matrix is much more efficient than access to rows. Most Algol implementations store two-dimensional arrays by rows. Accordingly, the inner loops of TRED1 and TRED2 access rows of the matrix; and an inner loop in TRED3 contains an IF statement resulting from its one-dimensional subscripting of a symmetric two-dimensional array.

We have rewritten TRED1, TRED2, and TRED3 so that their inner loops involve sequential access to memory. The improvement in efficiency of the new versions of the TREDs will be very dependent on the order of the matrix and the nature of the computer and operating system being used. For small matrices with conventional architecture and memory management, there will be little change. But for large matrices that require the use of virtual memory, the improvement can be significant.

EISPACK is now over 10 years old. It was never designed as a uniform "package" as we now use that term today. The overall organization, the choice of algorithms, the subroutine names, and the structure of the code itself are all inherited from the Algol collection in the *Handbook*.

We believe that it is time for a major new edition of EISPACK. A complete revision should have expanded capabilities, new user interfaces, and uniform naming conventions. Some algorithms should be altered to take account of vector machine architectures and paging operating systems. The programs should be written in Fortran 77 with an eye to future versions of Fortran. Such a project

would require careful planning and extensive resources.

REFERENCES

Cowell, W. R., and L. D. Fosdick [1977]. "Mathematical software production." *Mathematical Software III.* Ed. J. R. Rice. Academic Press, New York, pp. 195-224.

Garbow, B. S., J. M. Boyle, J. J. Dongarra, and C. B. Moler [1977]. *Lecture Notes in Computer Science, Vol. 51, Matrix Eigensystem Routines — EISPACK Guide Extension.* Springer-Verlag, New York.

Hammarling, S. J., P. D. Kenward, and H. J. Symm [1981]. *Amendments to Handbook for Automatic Computation, Volume II.* NPL Report DNACS 41/81.

Moler, C. B., and G. W. Stewart [1973]. "An algorithm for generalized matrix eigenvalues problems." *SIAM J. of Numer. Anal.* 10:241-256.

Schur, I. [1909]. "Uber die characteristischen Wurzeln einer linearen Substitution mit einer Anwendung auf die Theorie der Integral Gleichungen." *Math. Ann.* 66:488-510.

Smith, B. T., J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler [1976]. *Lecture Notes in Computer Science, Vol. 6, 2nd Edition, Matrix Eigensystem Routines — EISPACK Guide.* Springer-Verlag, New York.

Stewart, G. W. [1973]. *Introduction to Matrix Computations.* Academic Press, New York.

Wilkinson, J. H. [1965]. *The Algebraic Eigenvalue Problem.* Oxford University Press (Clarendon), Oxford.

Wilkinson, J. H., and C. Reinsch, eds. [1971]. *Handbook for Automatic Computation, Vol. II, Linear Algebra.* Springer-Verlag, New York.