ORNL/TM--11669

DE91 002523

Engineering Physics and Mathematics Division

Mathematical Sciences Section

COMPUTING THE EIGENVALUES AND EIGENVECTORS OF A GENERAL MATRIX BY REDUCTION TO GENERAL TRIDIAGONAL FORM

J. J. Dongarra
G. A. Geist
C. H. Romine

Mathematical Sciences Section
Oak Ridge National Laboratory
P.O. Box 2009, Bldg. 9207-A
Oak Ridge, TN 37831-8083

Date Published: September 1990

Research supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
operated by
Martin Marietta Energy Systems, Inc.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-84OR21400

Contents

COMPUTING THE EIGENVALUES AND EIGENVECTORS OF A GENERAL MATRIX BY REDUCTION TO GENERAL TRIDIAGONAL FORM

J. J. Dongarra

G. A. Geist

C. H. Romine

Abstract

This paper describes programs to reduce a nonsymmetric matrix to tridiagonal form, compute the eigenvalues of the tridiagonal matrix, improve the accuracy of an eigenvalue, and compute the corresponding eigenvector.

1. Introduction and Objectives

A standard approach in computing the eigenvalues of a general square matrix is to reduce the matrix first to Hessenberg form by a sequence of orthogonal transformations, and then to determine the eigenvalues of the Hessenberg matrix through an iterative process referred to as the QR algorithm [2]. The reduction to Hessenberg form requires $O(n^3)$ operations, where n is the order of the matrix, and the iterative portion typically requires $O(n^3)$ operations. The subroutine package EISPACK [8] uses this scheme to compute the eigenvalues and eigenvectors of a general matrix.

If the original matrix is symmetric, then that symmetry can be preserved in the initial reduction, so that the result is tridiagonal. Although the reduction to tridiagonal form costs $O(n^3)$ operations, the subsequent iterations preserve the tridiagonal form and are much less expensive, so that the total cost of the iterative phase is reduced to $O(n^2)$ operations. Again, standard software is available in EISPACK for implementing this two-phase approach for the symmetric case.

The attractively low operation count of iterating with a tridiagonal matrix suggests that the tridiagonal form would be extremely beneficial in the nonsymmetric case as well. Such an approach presents two difficulties, however. First, QR iteration does not preserve the structure of a nonsymmetric tridiagonal matrix. However, this problem can be overcome by using LR iteration [7] instead, which preserves the tridiagonal form. Second, it is difficult to reduce a nonsymmetric matrix to tridiagonal form by similarity transformations in a numerically stable manner. This second problem has been addressed in a paper by Geist [3]. Here, we describe the software available to reduce the matrix to tridiagonal form and to compute the eigenvalues and eigenvectors of the resulting tridiagonal matrix.

2. Initial Approximation to Eigenvalues

2.1. Reduction to Tridiagonal Form

The basic algorithm used in the reduction to tridiagonal form can be found in [4]. For each column from $k = 1$ to $n - 2$, this algorithm first applies the permutation that minimizes the maximum element in $N_r^{-1} N_c$, where $N_r^{-1} N_c A N_c^{-1} N_r$ reduces column k and then row k to the desired form by elementary similarity transformations. Column k and row k are then reduced by applying these similarity transformations. The implementation here differs from the original algorithm in two ways.

First, unlike the original algorithm, the transformations used in reducing each column and row are saved in the locations made available by the eliminations at each step. These transformations are needed for the calculation of the eigenvectors during the eigenvalue refinement.

Second, the reduction algorithm may encounter a zero (or unacceptably small) pivot regardless of permutation. When this occurs, the original algorithm applies one of two recovery methods. However, the first of these recovery methods interferes with the efficient in-place storage of the transformations. Hence, in this implementation, which is called ATOTRI, only the second of these recovery methods is used. The routine, called NEWSTR, applies a random Householder similarity transformation to the original matrix.

2.2. Eigenvalues of a Tridiagonal Matrix

One of the most efficient methoas of calculating all the eigenvalues of a nonsymmetric tridiagonal matrix is LR iteration. An implementation of the LR iteration has been developed that is specifically tailored to the tridiagonal structure.

In this so-called TLR implementation, the user supplies the tridiagonal matrix as three vectors. In the first step the superdiagonal elements are scaled to one. Since this scaling is preserved by the LR iteration, it decreases the operation count. Moreover, it frees up one vector for use as a working array.

Most of the improvements that have been incorporated into the QR iteration over the years can also be used in the context of the LR iteration. In particular, implicit double-shift iterations, deflation, and arbitrary shifts are used in TLR.

Double-shift iterations and deflation are implemented just as they are in EISPACK for HQR, with the exception that two consecutive small subdiagonals do not trigger a deflation as they do in HQR (although this can be added to TLR.)

Arbitrary shifts are invoked in two different contexts in TLR. First, if an eigenvalue has not converged in 20 iterations, then the iteration is assumed to be stuck in a cycle, and one arbitrary (random) double-shift is applied. Second, if the LR iteration, which does not pivot, encounters a zero (small) diagonal element. then the iteration breaks down, and one arbitrary shift is applied to change the values of the diagonal elements.

Up to 10 consecutive arbitrary shifts will be tried if the breakdown condition persists, after which the algorithm aborts with an error condition. However, a single arbitrary shift proved sufficient during all our tests.

Because of the potential for breakdown and the need to restart an iteration with a different shift, a copy of the matrix is made before the start of each iteration. This requires at most $2n$ storage. One n vector must be supplied by the user for this purpose. A second n vector, which initially holds the superdiagonal, is also used.

3. Improving the Accuracy of an Eigenvalue and Computing its Eigenvector

Approximations to the eigenvalues of A are obtained by reducing the matrix to tridiagonal form T (with ATOTRI) and then calculating the eigenvalues of T (with TLR). In many cases, particularly for small matrices, these computed eigenvalues closely approximate the eigenvalues of A. However, for larger matrices, or for matrices whose eigenvalues are ill conditioned, the rounding errors introduced during the reduction of A to tridiagonal form, coupled with the errors introduced by LR iteration, can induce significant errors in the computed eigenvalues. Hence, we assume that the reduction to tridiagonal form T and the subsequent calculation of the eigenvalues of T yield approximations to the eigenvalues of A that are then improved in a subsequent phase of the computation.

3.1. Inverse Iteration with Rayleigh Quotients

One standard technique for improving the accuracy of an eigenvalue and at the same time computing the associated eigenvector is to apply inverse iteration coupled with calculating the Rayleigh quotient. If only a few eigenpairs are desired, then inverse iteration is fairly attractive, since it is accurate and reasonably rapid. However, if the complete eigensystem of a dense matrix is required, inverse iteration becomes quite costly, since a (different) full linear system must be solved for each eigenpair, for each iteration, amounting to $O(n^4)$ operations. Such an operation count is prohibitive, particularly since the EISPACK routine HQR2 is highly accurate and requires only $O(n^3)$ operations for the full eigensystem.

Another alternative is to apply inverse iteration with Rayleigh quotients to the tridiagonal matrix T obtained from A by **ATOTRI**. Again, the solution of a different linear system for each iteration is required, but the linear systems now have a tridiagonal coefficient matrix and therefore can be solved in only $O(n)$ steps. Thus, inverse iteration with Rayleigh quotients applied to the matrix T is a very fast means of obtaining accurate approximations to the eigensystem of T. Unfortunately, to obtain the eigenvectors of the original matrix A, one must apply the inverse of the transformations that reduced A to tridiagonal form to the computed eigenvectors of T, and the eigenvectors of A may suffer from any resulting roundoff error. Moreover, the eigenvalues of T may differ from those of A for the same reason. The results given in Section 4 indicate the degree of inaccuracy stemming from these roundoff errors.

In summary, inverse iteration can give a useful rapid initial approximation to the eigensystem of A. However, if inverse iteration is applied to the original matrix A rather than the tridiagonal matrix T, the cost for computing the complete eigensystem is prohibitive. Finally, if applied to the tridiagonal matrix, there may be inaccuracies introduced by rounding error either in calculating the eigenvalues or in obtaining the eigenvectors of A from the eigenvectors of T.

3.2. Iterative Refinement

It has long been known that Newton's method for the solution of nonlinear systems can be applied to the problem of calculating the eigensystem of a matrix[6]. Moreover, in [1], Dongarra *et al.* describe an algorithm for improving the accuracy of an eigenpair based on Newton's method. The main drawback of their approach is that it is costly, in general. In this section, we describe a less costly variant of the algorithm given in [1] that takes advantage of the tridiagonalization of A while still obtaining a high degree of accuracy. The software implementing this algorithm is described in some detail in Section 5, but a short motivating description is given below.

Assume that (λ, x) is an approximate eigenpair of the matrix A and that $\lambda + \delta\lambda$ and $x + \delta x$ are a near by eigenpair such that the relationship

$$A(x + \delta x) = (\lambda + \delta\lambda)(x + \delta x),$$

is exact. Thus, $\delta\lambda$ and δx represent the errors associated with the computed values λ

and x, respectively.

Rearranging the equation, we have

$$(A - \lambda I)\delta x - \delta\lambda x = \lambda x - Ax + \delta\lambda\delta x,$$

where the last term on the right will be of second order in the errors in λ, x.

If we let $r = \lambda x - Ax$ and assume that the second-order term $\delta\lambda\delta x$ is negligible, we can rewrite the equation in the form

$$\begin{pmatrix} A - \lambda I & -x \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} \delta x \\ \delta\lambda \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix},$$

where $e_s^T \delta x = 0$ is a normalization applied to x such that the s component of x equals one, implying $\delta x_s = 0$ (see [1] for details).

When the original approximate eigenvalue is found by using the reduction to tridiagonal form, this yields a matrix N such that

$$A = N^{-1}TN.$$

Using the transformations from the reduction to tridiagonal form, we have

$$\begin{pmatrix} N & \\ & 1 \end{pmatrix} \begin{pmatrix} A - \lambda I & -x \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} N^{-1} & \\ & 1 \end{pmatrix} \begin{pmatrix} N & \\ & 1 \end{pmatrix} \begin{pmatrix} \delta x \\ \delta\lambda \end{pmatrix} = \begin{pmatrix} N & \\ & 1 \end{pmatrix} \begin{pmatrix} r \\ 0 \end{pmatrix},$$

which can be rewritten as

$$\begin{pmatrix} T - \lambda I & -Nx \\ e_s^T N^{-1} & 0 \end{pmatrix} \begin{pmatrix} \tilde{\delta x} \\ \delta\lambda \end{pmatrix} = \begin{pmatrix} \tilde{r} \\ 0 \end{pmatrix},$$

where $\tilde{r} = Nr$ and $\tilde{\delta x} = N\delta x$. The solution to the resulting linear system will produce approximations to the errors $\delta\lambda$ and δx, yielding new approximations to the eigenpair. The linear system is easily solved by transforming it into a tridiagonal system of equations by a rank-one modification. The software we have implemented applies the Sherman-Morrison formula [5] to solve the system of equations.

The approach described here will not only improve the accuracy of the approximate

Table 1: Comparison of the accuracy of the new routines to the EISPACK routine RG. Residual is $\max ||Ax - \lambda x||_\infty$ and e_λ is $\max |\lambda_i - \bar{\lambda}_i|$. ($\lambda_i$ is the eigenvalue obtained from RG, and $\bar{\lambda}_i$ is the computed eigenvalue.

Accuracy of Routines on dense random matrices					
problem size	ATOTRI-TLR	RG	INVIT	REFINE	
	e_λ	residual	residual	e_λ	residual
10	8.7E-14	1.8E-14	6.9E-14	4.4E-15	4.2E-16
100	7.2E-06	5.3E-12	1.6E-09	2.7E-13	5.1E-13
500	1.2E-02	4.4E-09	3.0E-07	4.3E-12	2.2E-12

eigenvalue λ but will also compute the eigenvector. The convergence theorem for this iterative procedure can be found in [1].

During the improvement phase, the subprogram **REFINE** is called, with the original data matrix **A**, the reduced tridiagonal matrix **T**, the transformation **N**, and an approximate eigenvalue **(WR,WI)** as parameters. A single inverse iteration step is performed with the tridiagonal matrix **T** (using **INVIT**) to obtain an initial approximation to the eigenvector associated with the given eigenvalue. On return from **REFINE** the improved eigenvalue is stored in **(WR,WI)** and the improved eigenvector in **(XR,XI)**.

4. Examples and Performance

We present two test suites to illustrate the speed and accuracy of the new algorithms. The performance of **HQR2** is included for comparison. All experiments were executed on an IBM RS/6000 model 530, using the Fortran compiler **xlf** without optimization.

Tables 1 and 2 show the results from three different size random matrices. The entries in each matrix are uniformly distributed on $[-1.0, 1.0]$. Table 1 shows the max-

Table 2: Comparison of the speed of the new routines to the EISPACK routine RG. Time for **INVIT** and **REFINE** are per eigenpair.

Performance of Routines on dense random matrices				
problem size	ATOTRI-TLR	RG (EISPACK)	INVIT (per λ, x)	REFINE (per λ, x)
10	.004	.028	.002	.005
100	2.62	15.27	.056	.277
500	493	2127	1.08	11.15

imum difference between the eigenvalues computed by ATOTRI-TLR and those calculated by HQR. In addition, the maximum difference of the improved eigenvalues is given. Finally, the residual is given for the results from inverse iteration, iterative refinement, and HQR2. Here, inverse iteration is performed by the routine INVIT applied to the tridiagonal matrix T until convergence to the desired eigenpair is achieved. The eigenvectors of A are then obtained by applying the inverse of the transformation matrix N. Table 2 shows the time in seconds to reduce the matrix to tridiagonal form and calculate its eigenvalues. Also shown is the average time per eigenvalue to improve the eigenvalue and calculate the corresponding eigenvector with either inverse iteration or iterative refinement.

The results of running the EISPACK general matrix test suite are shown in Table 3. The accuracy and robustness of the new algorithms are displayed by this test where we compare the residual from HQR2 to INVIT and REFINE.

Table 3: Maximum residual for the three methods of calculating eigenvalue/eigenvector pairs for dense matrices.

EISPACK Test Suite of Real General Matrices			
max $\lVert Ax - \lambda x \rVert_\infty$			
problem number	inverse iteration	iterative refinement	EISPACK (RG)
1	2.5E-12	2.9E-13	1.2E-12
2	9.2E-07	2.1E-07	6.3E-06
3	9.0E-13	1.3E-14	4.6E-06
4	1.8E-14	2.7E-13	1.0E-13
5	1.7E-07	9.4E-09	9.4E-07
6	1.5E-07	1.2E-09	2.4E-08
7	3.8E-08	2.9E-10	8.5E-09
8	0.0E-00	0.0E-00	0.0E-00
9	2.9E-15	1.7E-13	5.3E-09
10	1.2E-10	9.5E-11	1.8E-08
11	1.7E-14	1.3E-14	1.7E-13
12	2.9E-15	1.7E-15	2.4E-14
13	1.7E-13	9.2E-16	1.7E-14
14	3.3E-12	1.9E-16	2.4E-14
15	5.2E-14	4.8E-16	1.6E-14
16	7.5E-15	0.0E-00	1.1E-49
17	4.4E-15	0.0E-00	1.2E-30
18	6.3E-15	0.0E-00	0.0E-00
19	9.0E-15	8.8E-09	2.7E-08
20	1.4E-14	1.0E-15	9.7E-15
21	6.3E-15	2.2E-16	6.0E-15
22	1.0E-13	7.1E-16	2.1E-14
23	2.0E-10	3.2E-17	2.9E-14
24	2.5E-06	6.2E-09	1.1E-02
25	8.7E-07	2.2E-15	6.0E-14
26	4.3E-13	3.6E-14	2.2E-15
27	3.6E-01	9.0E-10	2.6E-06
28	4.8E-14	1.2E-14	5.7E-14
29	2.4E-14	2.8E-14	4.0E-12
30	5.2E-14	2.3E-13	4.2E-13
31	5.7E-14	1.8E-15	5.6E-14
32	1.4E-14	1.4E-05	4.4E-07
33	5.4E-01	1.9E-04	1.1E-08
34	4.4E-02	9.1E-14	1.5E-08
35	1.8E-12	1.8E-05	2.7E-13

5. Description of the Software and Programming Details

In this section we describe the software implementing the new algorithms.

```
      SUBROUTINE ATOTRI( LDA, A, N, PIVOTS, INFO )
c  Purpose:
c    This subroutine reduces an n-by-n real general matrix A to
c    tridiagonal form using elementary similarity transformations.
c
c    At each step k the permutation that minimizes the maximum entry
c    in the transformation matrix which reduces column k then row k
c    is applied.
c
c  Arguments:
c   LDA       -integer
c              LDA is the leading dimension of A.
c
c   A         -double precision array of dimension (LDA,N)
c              On entry A contains the matrix being reduced.
c              On exit A is overwritten by its tridiagonal form.
c
c   N         -integer
c              N specifies the order of the matrix A.
c              N must be nonnegative.
c              N is not modified.
c
c   PIVOTS    -integer array of dimension (LDA)
c              On exit pivots contains the pivot sequence used during
c              the reduction (permutation vector).
c
c   INFO      -integer
c              On exit, INFO is set to
```

```
c             0  normal return.
c             1  if NEWSTR should be executed before ATOTRI.
c
      SUBROUTINE NEWSTR( A, LDA, N, W, IFLAG )
c  Purpose:
c     This subroutine generates a random Householder transformation and
c     applies it to the matrix A to scramble it.  The matrix A is
c     assumed to be in dense format.
c
c  Arguments:
c   A        -double precision array of dimension (LDA,N)
c             On entry A contains the original matrix.
c             On exit, A contains QAQ, where Q is defined by W below.
c
c   LDA      -integer
c             LDA is the leading dimension of A.
c
c   N        -integer
c             N specifies the order of the matrix A.
c             N must be nonnegative.
c             N is not modified.
c
c   W        -double precision array of dimension (N)
c             On exit, W contains a random Householder vector defining
c             a Householder transformation Q=I-2WW'.
c
c   IFLAG    -integer
c             On exit, IFLAG is set to one, indicating that NEWSTR
c             has been called.
c
      SUBROUTINE TLR( N, DIAG, SUB, SUP, SAV, INFO )
c  Purpose:
```

```
c    This subroutine determines the eigenvalues of a general
c    tridiagonal matrix by applying implicit double-shift LR iterations.
c
c    The eigenvalues are returned with the real part on the
c    diagonal and the imaginary part on the subdiagonal.
c    INFO equals 1 on exit if TLR is unable to determine all
c    the eigenvalues.
c
c  Arguments:
c   N     -integer
c          N specifies the order of the tridiagonal matrix.
c          N is not modified.
c
c   DIAG  -double precision array of dimension (N)
c          On entry DIAG contains the diagonal of the tridiagonal
c          matrix.
c          On exit DIAG contains the real part of the eigenvalues.
c
c   SUB   -double precision array of dimension (N)
c          On entry SUB contains the sub-diagonal of the
c          tridiagonal matrix.
c          On exit SUB contains the imaginary part of the eigenvalues.
c
c   SUP   -double precision array of dimension (N)
c          On entry SUP contains the super-diagonal of the
c          tridiagonal matrix. It is used as a work array
c          during the iteration.
c
c   SAV   -double precision array of dimension (N)
c          SAV is a work array used along with SUP to save a copy
c          of the previous iteration matrix in case the present
c          iteration breaks down and an arbitrary shift is required.
```

```
c
c  INFO  -integer
c          On exit, INFO is set to
c           0  normal return.
c           1  failure to converge to one or more eigenvalues.
c              User should revert to EISPACK routines in this case.
c
      SUBROUTINE REFINE( N, A, LDA, AORG, WR, WI, XR, XI, IPVT, W,
     $                   IFLAG, WORK, LDWORK )
c  Purpose:
c     This routine uses an iterative refinement technique to
c     improve the accuracy of the eigenvalue approximation
c     (WR,WI) and to compute the corresponding eigenvector
c     (XR,XI).  It is assumed that the user has reduced the
c     matrix to tridiagonal form (see routines ATOTRI and TLR
c     for details).  The matrix A contains information about
c     the reduction to tridiagonal form.  AORG is the orginal
c     matrix, required in the residual computation for the
c     refinement.
c
c  Arguments:
c    N     -integer
c      N specifies the order of the matrix A.
c      N must be nonnegative.
c      N is not modified.
c
c    A     -double precision array of dimension (LDA,N)
c      A contains information about the reduction to
c      tridiagonal form.
c
c    LDA    -integer
c      LDA is the leading dimension of the array A.
```

```
c       LDA >= max(1,N).
c
c     AORG   -double precision array of dimension (LDA,N)
c       AORG contains the original matrix.
c
c     WR      -double precision
c       On entry, WR is the real part of the approximate
c       eigenvalue.
c       On exit, WR is the improved real part of the
c       approximate eigenvalue.
c
c     WI      -double precision
c       On entry, WI is the imaginary part of the
c       approximate eigenvalue.
c       On exit, WI is the improved imaginary part of the
c       approximate eigenvalue.
c
c     XR      -double precision array of dimension (N)
c       The real part of the computed eigenvector.
c
c     XI      -double precision array of dimension (N)
c       The imaginary part of the computed eigenvector.
c
c     IPIV   -integer array of dimension (N)
c       IPIV contains the pivot sequence used during the
c       reduction to tridiagonal form.
c
c     W      -double precision array of dimension (N)
c       W may contain information if a restart was
c       performed in the tridiagonal process as
c       indicated by IFLAG.
c
```

```
c    IFLAG  -integer
c      IFLAG signals if a restart was required during
c            reduction to tridiagonal form.
c            IFLAG = 1 signals a restart was taken.
c
c    WORK   -double precision array of dimension (LDWORK,19)
c      WORK is used for workspace.
c
c    LDWORK -integer
c            LDWORK is the leading dimension of the array WORK.
c            LDWORK >= max(1,N+1).
c
```

6. References

[1] J. J. Dongarra, C. B. Moler, and J. H. Wilkinson. Improving the accuracy of computed eigenvalues and eigenvectors. *SIAM J. Numer. Anal.*, 20:23–45, February 1983.

[2] J. G. F. Francis. The QR transformation – Part 2. *The Computer Journal*, 4:332 – 345. 1961.

[3] G. A. Geist. Reduction of a general matrix to tridiagonal form. Technical report, Oak Ridge National Laboratory, February 1989. ORNL/TM-10991.

[4] G. A. Geist, A. Lu, and E. L. Wachspress. Stabilized Gaussian reduction of an arbitrary matrix to tridiagonal form. Technical report, Oak Ridge National Laboratory, February 1989. ORNL/TM-11089.

[5] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, 1983.

[6] L. B. Rall. *Computational Solution of Nonlinear Operator Equations*. Wiley, 1969.

[7] H. Rutishauser. Solution of eigenvalue problems with the LR transformation. *Nat. Bur. Standards Appl. Math. Ser.*, 49:47–81, 1958.

[8] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garabow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystem Routines - EISPACK Guide.* Springer, Heidelberg, 1974.

- END -

DATE FILMED

11 / 28 / 90