ALGORITHM 710
FORTRAN Subroutines for Computing the Eigenvalues and Eigenvectors of a General Matrix by Reduction to General Tridiagonal Form

J. J. DONGARRA, G. A. GEIST, and C. H. ROMINE
Oak Ridge National Laboratory

This paper describes programs to reduce a nonsymmetric matrix to tridiagonal form, to compute the eigenvalues of the tridiagonal matrix, to improve the accuracy of an eigenvalue, and to compute the corresponding eigenvector. The intended purpose of the software is to find a few eigenpairs of a dense nonsymmetric matrix faster and more accurately than previous methods. The performance and accuracy of the new routines are compared to two EISPACK paths: **RG** and **HQR-INVIT**. The results show that the new routines are more accurate and also faster if less than 20 percent of the eigenpairs are needed.

Categories and Subject Descriptors: F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computations on matrices*; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*eigenvalues*; G.4 [**Mathematics of Computing**]: Mathematical Software—*algorithm analysis*

General Terms: Algorithms

Additional Key Words and Phrases: Condensed form, eigenvalues, nonsymmetric, numerical algorithms

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 known as the QR algorithm

This work was supported in part by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy, under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc. The work of C. H. Romine was also supported in part by U.S. Navy Space and Naval Warfare Systems (SPAWAR) contract N00039-89-C-0001.
Authors' address: Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN 37831-8083.

[3]. The reduction to Hessenberg form requires $O(n^3)$ operations, where n is the order of the matrix, and the subsequent iterative phase also requires $O(n^3)$ operations. The subroutine package EISPACK [10] uses this scheme to compute all of 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 reduced matrix is tridiagonal. Although the reduction to tridiagonal form still requires $O(n^3)$ operations, the subsequent iterations preserve the tridiagonal form and, hence, 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 obtained when 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. This problem can be overcome by using LR iteration [9] 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. Methods to improve the stability can be found in [4]. 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

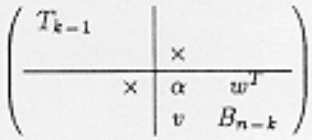
The algorithm used in the direct reduction to tridiagonal form is discussed in detail in [6]. The algorithm alternately eliminates columns and rows of the matrix, preserving the form shown in Figure 1. Retaining this matrix structure allows us to improve the overall stability of the algorithm by pivoting at each step.

At the kth stage (see Figure 1), the algorithm applies the permutation that minimizes the maximum multiplier in the transformation matrix $N_r^{-1}N_c$. Here, N_r and N_c are elementary matrices such that $N_c A N_c^{-1}$ reduces column k and $N_r^{-1}(N_c A N_c^{-1})N_r$ reduces row k. This minimization can be performed in $O(n-k)$ time because of the special structure of $N_r^{-1}N_c$. Details of this minimization algorithm can be found in [5].

The reduction algorithm may encounter a pivot that is zero (or unacceptably small) regardless of the permutation. When this occurs, the original reduction is said to have broken down, and the subprogram **NEWSTR** is called. **NEWSTR** applies a random Householder similarity transformation to the original matrix. The original matrix must therefore be saved, but this is already necessary in order to apply the iterative refinement method described in Section 3.2. **NEWSTR** is only applied once in our scheme. If the reduction of the second matrix also breaks down, then the algorithm returns an error code. This occurrence has not yet been observed in practice.

The transformations used in annihilating each column and row are saved

Fig. 1. Partially reduced matrix.



in the locations made available by the eliminations at each step. These transformations are needed for the calculation of the eigenvectors during the iterative refinement step.

2.2 Eigenvalues of a Tridiagonal Matrix

One of the most efficient methods of calculating all of the eigenvalues of a nonsymmetric tridiagonal matrix is LR iteration. Most of the improvements that have been incorporated into QR iteration over the years [11], such as implicit double-shift iterations, deflation, splitting, and arbitrary shifts, can also be used in the context of LR iteration.

An implementation of LR iteration that is specifically tailored for tridiagonal matrices, called **TLR**, has been developed. The user supplies the tridiagonal matrix as three vectors. In the first step, the matrix is scaled so that its superdiagonal elements are equal to one. This reduces the operation count, since LR iteration preserves the form of such a tridiagonal matrix. Moreover, the superdiagonal vector is now free for use as a working array. Implicit double-shift iterations and splitting are implemented just as they are in EISPACK for **HQR**. Splitting due to either negligible subdiagonal elements or to two consecutive small subdiagonal elements are implemented. The criteria we use for negligible and small entries are the same as in **HQR**.

Implicit 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 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. (Another obvious method for avoiding a zero diagonal is to pivot inside LR, but this has the major drawback of destroying the tridiagonal structure of the matrix.) Up to 10 consecutive arbitrary shifts are tried if the breakdown condition persists, after which the algorithm aborts with an error condition. However, a single arbitrary shift proved sufficient during all of 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. The second n vector initially holds the superdiagonal, but this space is reclaimed after the matrix is scaled.

3. IMPROVING THE ACCURACY OF THE EIGENVALUE AND COMPUTING THE EIGENVECTOR

Approximations to the eigenvalues of A are obtained by reducing the matrix to tridiagonal form T (with **ATOTRI**) and then by 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 regard the reduction to tridiagonal form T and the subsequent calculation of the eigenvalues of T as yielding approximations to the eigenvalues of A, which 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. The EISPACK routine **INVIT** performs inverse iteration (without Rayleigh quotients) to a Hessenberg matrix. Each iteration requires $O(n^2)$ operations, since solving a linear system with a new right-hand side is required for each iteration. If the complete eigensystem of a dense matrix is required, then the EISPACK routine **RG** is recommended because it is robust, 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 will 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 on T can give a useful rapid initial approximation to the eigensystem of A. But 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 [8]. Moreover, in [2], 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 too costly, in general. In this section

we describe a less costly variant of the algorithm given in [2] that takes advantage of the tridiagonalization of A while still obtaining a high degree of accuracy.

Assume that (λ, x) is an approximate eigenpair of the matrix A and that $\lambda + \delta\lambda$ and $x + \delta x$ are a neighboring 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 and 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 [2] 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

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

where $\bar{r} = Nr$ and $\overline{\delta x} = N\delta x$. The solution to the resulting linear system produces 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 [7] to solve the system of equations.

Given the original matrix A, the tridiagonal matrix T, the transformations N that reduced A to T ($A = N^{-1}TN$), and the approximate eigenvalue λ_0,

the refinement algorithm can be described as follows:

v is an initial guess for the eigenvector;
perform one step of inverse iteration, $(T - \lambda_0 I)v_0 = v$;
$x_0 = Nv_0$;
for $i = 0, 1, 2, \ldots$
$r_i = Ax_i - \lambda_i x_i$;
$g_i^T = (e_s^T N, -1)^T$;
solve $\begin{pmatrix} T - \lambda_i I & -Nx_i \\ g_i^T & 0 \end{pmatrix} \begin{pmatrix} y_i \\ \mu_i \end{pmatrix} = \begin{pmatrix} Nr_i \\ 0 \end{pmatrix}$;
$w_i = Ny_i$;
$x_{i+1} = x_i + w_i$;
$\lambda_{i+1} = \lambda_i + \mu_i$;
check if converged, $\dfrac{\|Ax_{i+1} - \lambda_{i+1}x_{i+1}\|}{10\|A\|\epsilon} \leq 1$;
end.

Note that the eigenpair is refined relative to the original matrix; that is, the residual is computed with the original data A, and the improvement is being made to the eigenvector of A. The tridiagonal matrix T and the transformations N are used solely to simplify solving the system of equations. Hence, the convergence will be to the eigensystem of the original matrix A, not the tridiagonal matrix T.

Because of the relationship with Newton's method, convergence is guaranteed when $\beta\eta\kappa < \frac{1}{2}$, where

$$\beta = \left\| \begin{pmatrix} A - \lambda_i I & -x_i \\ e_s^T & 0 \end{pmatrix} \right\|^{-1},$$

$\eta = \|x_{i+1} - x_i\|$, and $\kappa = 2$ (κ is a bound on the second derivative). As can be seen, the convergence rate and error bound depend on the condition of the matrix (see [1] for additional information).

The approach described here will not only improve the accuracy of the approximate eigenvalue λ but will also compute the eigenvector. The convergence theorem for this iterative procedure can be found in [2].

During the improvement phase, the subprogram named **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 **GTINIT**) 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. Routines from Release 3 of EISPACK, which is currently available, were used in our comparisons. All experiments were executed on an IBM RS/6000 model 530, using the FORTRAN compiler **xlf** with optimization.

Table I. Comparison of Accuracy of New Routines to EISPACK Routine **RG**: Residual Is $\max \|Ax - \lambda x\|_\infty$, and e_λ Is $\max|\lambda_i - \overline{\lambda}_i|(\lambda_i$ is the eigenvalue obtained from **RG**, and $\overline{\lambda}_i$ is the computed eigenvalue)

Accuracy of routines on dense random matrices					
Problem size	ATOTRI-TLR e_λ	RG (EISPACK) residual	INVIT (EISPACK) residual	REFINE e_λ	REFINE Residual
10	8.7E-14	1.8E-14	2.0E-14	4.4E-15	3.7E-16
100	7.2E-06	5.3E-12	2.9E-12	2.7E-13	5.1E-13
500	1.2E-02	2.5E-09	3.0E-10	4.3E-12	2.3E-12

Table II. Comparison of Execution Times in Seconds of New Routines to EISPACK Routine **RG**: Time for **INVIT** and **REFINE** Are per Eigenpair

Performance of routines on dense random matrices					
Problem size	RG all (λ, x)	ATOTRI TLR	ELMHES HQR	INVIT (per λ, x)	REFINE (per λ, x)
10	0.024	0.004	0.030	0.0004	0.0036
100	1.710	0.293	0.796	0.0053	0.0332
300	78.300	5.560	34.500	0.1490	0.4342
500	459.000	22.200	202.000	0.5911	1.867

Tables I and II show the results for random matrices with the entries uniformly distributed in $[-1.0, 1.0]$. Table I shows the maximum difference between the eigenvalues computed by **ATOTRI-TLR** and those calculated by **RG**. The maximum difference of the improved eigenvalues is also given. Finally, the residual is given for the results from inverse iteration, iterative refinement, and **RG**. The inverse iteration results are obtained by calling the EISPACK routines **ELMHES** and **HQR**, followed by **INVIT** for every eigenpair. The maximum residual over all of the eigenpairs is reported in the table. Similarly, **REFINE** was called for every eigenpair, and the maximum residual is reported. In every case the smallest maximum residual was generated with the new iterative refinement routines.

Table II compares the execution times of three methods of calculating eigenpairs for a nonsymmetric matrix. For reference, the time required for **RG** to calculate all of the eigenpairs is given. **RG** does not allow the user to calculate selected eigenpairs. If selected eigenpairs are desired, then the user can call the EISPACK path **ELMHES**, **HQR**, **INVIT**, **ELMBAK**. The table shows the time to reduce the matrix to Hessenberg form and to calculate all of its eigenvalues. In a separate column, the average time to calculate an eigenpair is given. (The time for **ELMBAK** is divided among the n eigenpairs calculated.) The table also shows the time to reduce the matrix to tridiagonal form and to calculate its eigenvalues. This operation is amazingly fast on a cache-based machine like the RS/6000. The average time per eigenpair for improving the eigenvalue and for calculating the corresponding eigenvector with iterative refinement is about four times more than using **INVIT**. But because the

Table III. Maximum Residual for Three Methods of Calculating Eigenvalue/Eigenvector Pairs for Dense Matrices

EISPACK test suite of real general matrices

$\max\lvert(\lambda - \bar{\lambda}_{RG})/\bar{\lambda}_{RG}\rvert$		$\max\lVert Ax - \lambda x\rVert_\infty$		
Problem number	ATOTRI TLR	Inverse iteration	Iterative refinement	EISPACK (RG)
1	1E-13	2.5E-12	2.9E-13	1.2E-12
2	1E-11	9.2E-07	2.1E-07	6.3E-06
3	8E-07	9.0E-13	1.3E-14	4.6E-06
4	5E-15	1.8E-14	2.7E-13	1.0E-13
5	1E-15	1.7E-07	9.4E-09	9.4E-07
6	1E-13	1.5E-07	1.2E-09	2.4E-08
7	8E-10	3.8E-08	2.9E-10	8.5E-09
8	0E-00	0.0E-00	0.0E-00	0.0E-00
9	2E-06	2.9E-15	1.7E-13	5.3E-09
10	5E-15	1.2E-10	9.5E-11	1.8E-08
11	7E-16	1.7E-14	1.3E-14	1.7E-13
12	3E-00	2.9E-15	1.7E-15	2.4E-14
13	7E-15	1.7E-13	9.2E-16	1.7E-14
14	3E-15	3.3E-12	1.9E-16	2.4E-14
15	7E-16	5.2E-14	4.8E-16	1.6E-14
16	0E-00	7.5E-15	0.0E-00	1.1E-49
17	0E-00	4.4E-15	0.0E-00	1.2E-30
18	0E-00	6.3E-15	0.0E-00	0.0E-00
19	8E-07	9.0E-15	8.8E-09	2.7E-08
20	1E-16	1.4E-14	1.0E-15	9.7E-15
21	1E-16	6.3E-15	2.2E-16	6.0E-15
22	5E-14	1.0E-13	7.1E-16	2.1E-14
23	6E-14	2.0E-10	3.2E-17	2.9E-14
24	1E-10	2.5E-06	6.2E-09	1.1E-02
25	1E-16	8.7E-07	2.2E-15	6.0E-14
26	1E-16	4.3E-13	3.6E-14	2.2E-15
27	6E+07	3.6E-01	9.0E-10	2.6E-06
28	1E-16	4.8E-14	1.2E-14	5.7E-14
29	1E-16	2.4E-14	2.8E-14	4.0E-12
30	1E-13	5.2E-14	2.3E-13	4.2E-13
31	2E-12	5.7E-14	1.8E-15	5.6E-14
32	1E-03	1.4E-14	1.4E-05	4.4E-07
33	1E-16	5.4E-01	1.9E-04	1.1E-08
34	1E-16	4.4E-02	9.1E-14	1.5E-08
35	1E-01	1.8E-12	1.8E-05	2.7E-13

routines **ATOTRI** and **TLR** are so fast (**TLR** is $O(n^2)$ as compared to $O(n^3)$ for **HQR** [5]), the total time for calculating up to 20 percent of the eigenpairs is smaller (and the results more accurate) using the new routines.

The results of running the EISPACK general matrix test suite [10] are shown in Table III. This test suite consists of 35 small matrices (none exceeding 20×20) that are designed to be pathological with respect to their eigenvalues and eigenvectors. Most of the matrices are ill-conditioned, some are defective, some are derogatory, and some are all three. The accuracy and robustness of the new algorithms are displayed by this test, where we

compare the residual from **RG** to **GTINIT** and **REFINE**. **GTINIT** applies inverse iteration with Rayleigh quotients 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. For reference, the maximum error (with respect to **RG**) of the initial eigenvalue estimate from **TLR** is given for each matrix.

5. SUMMARY

It is clear from our tests that if all of the eigenpairs are required, then the EISPACK routine **RG** is the recommended approach. However, the new routines presented in this paper are superior in both speed and accuracy to existing methods (i.e., EISPACK) when only a few (up to 20 percent) of the eigenpairs of a dense nonsymmetric matrix are required. The demand for routines to solve such problems is growing rapidly in many areas of computational science, including quantum chemistry and materials science.

REFERENCES

1. Dongarra, J. J. Improving the accuracy of computed matrix eigenvalues. Tech. Rep. ANL-80-84, Argonne National Laboratory, Chicago, Ill., Aug. 1980.
2. Dongarra, J. J., Moler, C. B., and Wilkinson, J. H. Improving the accuracy of computed eigenvalues and eigenvectors. *SIAM J. Numer. Anal. 20*, 1 (Feb. 1983), 23–45.
3. Francis, J. G. F. The QR transformation—Part 2. *Comput. J. 4*, 4 (Oct. 1961), 332–345.
4. Geist, G. A. Reduction of a general matrix to tridiagonal form. Tech. Rep. ORNL/TM-10991, Oak Ridge National Laboratory, Oak Ridge, Tenn., Feb. 1989.
5. Geist, G. A. Reduction of a general matrix to tridiagonal form. *SIAM J. Matrix Anal. Appl. 12*, 2 (Apr. 1991), 362–373.
6. Geist, G. A., Lu, A., and Wachispress, E. L. Stabilized Gaussian reduction of an arbitrary matrix to tridiagonal form. Tech. Rep. ORNL/TM-11089, Oak Ridge National Laboratory, Oak Ridge, Tenn., Feb. 1989.
7. Golub, G. H., and Van Loan, C. F. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Md., 1983.
8. Rall, L. B. *Computational Solution of Nonlinear Operator Equations*. Wiley, New York, 1969.
9. Rutishauser, H. Solution of eigenvalue problems with the LR transformation. *Nat. Bur. Standards Appl. Math. Ser. 49* (1958), 47–81.
10. Smith, B. T., Boyle, J. M., Dongarra, J. J., Garabow, B. S., Ikebe, Y., Klema, V. C., and Moler, C. B. *Matrix Eigensystem Routines—EISPACK Guide*. Springer-Verlag, Heidelberg, 1974.
11. Wilkinson, J. H. *The Algebraic Eigenvalue Problem*. Oxford University Press, Oxford, 1965.

Received November 1990; revised May and October 1991; accepted November 1991