ALGORITHM 589
SICEDR: A FORTRAN Subroutine for Improving the Accuracy of Computed Matrix Eigenvalues

JACK J. DONGARRA
Argonne National Laboratory

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*eigenvalues*, G.m [**Mathematics of Computing**] Miscellaneous—*FORTRAN*

General Terms: Algorithms

Additional Key Words and Phrases: Matrix eigensystems, iterative method; eigensystem improvement

1. INTRODUCTION

SICEDR is a FORTRAN subroutine for improving the accuracy of a computed real eigenvalue and improving or computing the associated eigenvector. It is first used to generate information during the determination of the eigenvalues by the Schur decomposition technique. In particular, the Schur decomposition technique results in an orthogonal matrix Q and an upper quasi-triangular matrix T, such that

$$A = QTQ^{\mathrm{T}}. \tag{1.1}$$

Matrices A, Q, and T and the approximate eigenvalue, say λ, are then used in the improvement phase. **SICEDR** uses an iterative method similar to iterative improvement for linear systems to improve the accuracy of λ and improve or compute the eigenvector x in $O(n^2)$ work, where n is the order of the matrix A. The method used in **SICEDR** is described in [1, 5].

2. USAGE

For a description of the calling sequence, see the listing presented at the end of this paper.

SICEDR factors the matrix into its Schur decomposition, and this is termed the pre-**SICE** phase. A modification of the EISPACK routine **HQR2** is used for this purpose (see [4, pp. 100–101] for details).

Received 15 October 1980, revised 14 July 1982, accepted 20 July 1982
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 and in part at the University of New Mexico by the Computer Science Section of the National Science Foundation under Contract MCS 7603297.
Author's address· Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439.

During the improvement phase (or **SICE** phase), **SICEDR** is called with matrices A, T, and Q and the approximate eigenvalue W. On return from **SICEDR** the improved eigenvalue is stored in W and the improved or computed eigenvector in X. W and X contain the corrected eigenpair produced by the method at the next to last iteration. A still more accurate eigenpair can be formed if on return from **SICEDR** the user adds W to CW and X to CX in double precision and saves the results in double precision, where CW and CX contain the corrections to W and X, respectively, at the final iteration when the method terminates.

The routine **SICEDR** is provided as a driver to simplify usage. **SICEDR** performs both the pre-**SICE** phase (reduction to upper form) and the **SICE** phase (improvement).

In addition to the EISPACK package, the LINPACK [2] routine **STRSL** and the BLAS [3] package are used to perform fundamental operations.

To produce a double-precision version of **SICEDR** in addition to making the obvious changes, such as changing variables to double precision and using double-precision intrinsic functions, it is necessary to replace routine **STRSL** by **DTRSL** and use the double-precision version of the BLAS. In addition, there are two critical parts of the calculation which *must* be performed in an extended precision. In the single-precision version, the routine **SDDDOT** and **SDADD** perform double-precision accumulations. In a double-precision version, they must be replaced by, say **DQQDOT** and **DQADD**, the extended precision counterparts. These extended precision routines would accumulate in quadruple precision, a nonstandard FORTRAN construct. Such routines are not included here, but can be constructed from the BLAS routines **DQDOTI** and **DQDOTA**.

3. SUMMARY OF THE ALGORITHM

We begin with a brief discussion of the basic algorithm as described in [1].

If λ, x is an approximate eigenpair, and $\lambda + \mu$, $x + \tilde{y}$ is a neighboring eigenpair, then

$$A(x + \tilde{y}) = (\lambda + \mu)(x + \tilde{y}), \tag{3.1}$$

this relation being exact. We assume that x is normalized so that $\| x \|_\infty = 1 = x_s$, and we remove the degree of arbitrariness in $\tilde{y}$ by requiring that $\tilde{y}_s = 0$. Rearranging this equation, we have

$$(A - \lambda I)\tilde{y} - \mu x = \lambda x - Ax + \mu\tilde{y}, \tag{3.2}$$

where the last term on the right will be of second order in the errors of λ, x. The equation above may be simplified by the introduction of a vector y defined by

$$y^{\mathrm{T}} = (y_1, y_2, \ldots, y_{s-1}, \mu, y_{s+1}, \ldots, y_n), \tag{3.3}$$

so that y gives the full information on both μ and $\tilde{y}$. The above equation then becomes

$$By = r + y_s\tilde{y}, \tag{3.4}$$

where $r = \lambda x - Ax$ is the residual vector corresponding to λ, x and B is the matrix $A - \lambda I$ with column s replaced by $-x$.

When the original approximate eigenpairs have been found by Francis' double QR algorithm, we have an orthogonal matrix Q and a quasi upper triangular matrix T, such that

$$A = QTQ^T. \tag{3.5}$$

T is triangular apart from possible 2×2 diagonal blocks corresponding to complex conjugate eigenvalues. We solved a succession of linear systems of the form (3.4) with varying right-hand sides g. It will be convenient to introduce the generic notation $Z - \lambda I = Z_\lambda$ and $(Z - \lambda I)e_s = Z_\lambda e_s = z_{\lambda s}$. We may then rewrite eq. (3.4) in the form

$$[A_\lambda - (x + a_{\lambda s})e_s^T]y = (A_\lambda + ce_s^T)y = g, \tag{3.6}$$

where

$$c = -x - a_{\lambda s}.$$

Using the orthogonal factorization, we have

$$Q[T_\lambda + Q^T ce_s^T Q]Q^T y = g,$$

giving

$$(T_\lambda + df^T)Q^T y = Q^T g, \tag{3.7}$$

where

$$d = Q^T c, \qquad f^T = e_s^T Q \qquad \text{and} \qquad g = r + y_s \tilde{y}.$$

The matrix df^T is a rank one modification of the quasi-triangular matrix T_λ. To solve this system, we need to retriangularize $T_\lambda + df^T$. Accordingly, we premultiply the system by two orthogonal matrices Q_1 and Q_2, giving

$$Q_2 Q_1 (T_\lambda + df^T)Q^T y = Q_2 Q_1 Q^T g, \tag{3.8}$$

where Q_1 and Q_2 are products of elementary plane rotations determined as follows.

The matrix Q_1 is such that

$$Q_1 d = (P_2 P_3 \cdots P_n)d = \gamma e_1 \qquad \text{where} \quad \gamma = \| d \|_2 \tag{3.9}$$

and P_i is a rotation in the $(i - 1, i)$ plane designed to annihilate the ith component of $P_{i+1}P_{i+2} \cdots P_n d$. We have

$$Q_1(T_\lambda + df^T) = Q_1 T_\lambda + \gamma e_1 f^T, \tag{3.10}$$

where $Q_1 T_\lambda$ is upper Hessenberg, while $\gamma e_1 f^T$ is null except in the first row. Hence the right-hand side is also an upper Hessenberg matrix H. H may now be reduced to upper triangular form, $\bar{T}_\lambda$, by premultiplication with Q_2 defined by

$$Q_2 = P_n' \cdots P_3' P_2', \tag{3.11}$$

where the premultiplication by P_i' annihilates the element $(i, i - 1)$ of the current matrix by a rotation in the $(i - 1, i)$ plane. Hence, the triangular system remains to be solved

$$\bar{T}_\lambda Q^{\mathrm{T}} y = Q_2 Q_1 Q^{\mathrm{T}} g. \tag{3.12}$$

A system with the matrix B may thus be solved in $O(n^2)$ operations.

By its nature, eq. (3.2), which leads to eq. (3.12), is mildly nonlinear. Thus we repeat the process with $x + \tilde{y}$ and $\lambda + \mu$ as the new approximations. The convergence theorem for this iterative procedure can be found in [1]. Since an orthogonal triangularization of A is available, it becomes practical to update the matrix B at each stage of the iteration, using the current approximation to the eigenpair. Accordingly, we treat the $(p + 1)$th step of the iteration as though it were the first step in the basic iteration starting with values $\lambda^{(p)}$ and $x^{(p)}$. Since we treat each iterate as if it were the first, the term $\mu\tilde{y}$ in eq. (3.2) is zero. Thus $g = \lambda x - Ax$. The algorithm then becomes

$$(A - \lambda^{(p)} I)\tilde{y}^{(p)} - y_s^{(p)} x^{(p)} = r^{(p)} = (\lambda^{(p)} I - A) x^{(p)} \tag{3.13}$$

where

$$x^{(p+1)} = x^{(p)} + \tilde{y}^{(p)}, \qquad \lambda^{(p+1)} = \lambda^{(p)} + y_s^{(p)}.$$

We may rewrite this as

$$B^{(p)} y^{(p)} = r^{(p)}, \tag{3.14}$$

where

$$B^{(p)} = A - \lambda^{(p)} I + c^{(p)} e_s^{\mathrm{T}}, \qquad c^{(p)} = -x - a_{\lambda s}^{(p)}.$$

We now write

$$\begin{aligned} B^{(p)} &= Q(T_\lambda^{(p)} + Q^{\mathrm{T}} c^{(p)} e_s^{\mathrm{T}} Q) Q^{\mathrm{T}} \\ &= Q(T_\lambda^{(p)} + d^{(p)} f^{\mathrm{T}}) Q^{\mathrm{T}} \end{aligned} \tag{3.15}$$

and solve (3.14) by the triangular system

$$\bar{T}_\lambda^{(p)} Q^{\mathrm{T}} y^{(p)} = Q_2^{(p)} Q_1^{(p)} Q^{\mathrm{T}} r^{(p)}. \tag{3.16}$$

Convergence is detected by examining successive values for the correction to the eigenvalue. When the previous correction is smaller by a factor of 2 than the current correction, the iteration is stopped.

Note that Q and f will be independent of p if s is not changing from one iteration to the next. The rotations involved in $Q_1^{(p)}$ and $Q_2^{(p)}$, on the other hand, differ from one iteration to the next; but because the number of operations in each retriangularization is $O(n^2)$ and since some $\frac{1}{2}n^2$ multiplication and additions are necessarily involved in the solution of a triangular system, this is quite acceptable.

If we examine the number of operations for the reduction to quasi-triangular form, the *pre*-**SICE** phase, we see that there are roughly $10n^3 + 30n^2$ operations involved (here, an operation means a multiplication followed by an addition). For the **SICE** phase there are approximately $13n^2$ operations per iteration. Assuming 3 iterations to improve an eigenpair, the total count is approximately $39n^2$ operations.

ACKNOWLEDGMENT

I would like to thank the referees for their valuable suggestions, which have improved the quality of the software.

REFERENCES

1. DONGARRA, J.J., MOLER, C B., AND WILKINSON, J.H. Improving the accuracy of computed eigenvalues and eigenvectors. *SIAM J. Numer. Anal.* to appear (ANL-81-43, Argonne National Laboratory, Argonne, Ill. (1981))
2. DONGARRA, J.J., BUNCH, J.R., MOLER, C.B., AND STEWART, G.W *LINPACK Users' Guide.* SIAM Pub., Philadelphia, Pa., 1979.
3. LAWSON, C.L., HANSON, R J., KINCAID, D R., AND KROGH, F T Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math Softw. 5*, 3 (Sept. 1979), 308–323.
4. SMITH, B.T., ET AL. Matrix eigensystem routines—EISPACK guide. In *Lecture Notes in Computer Science*, 2nd ed , vol. 6, Springer-Verlag, Berlin, 1976.
5. SYMM, H.J., AND WILKINSON, J.H. Realistic error bounds for a simple eigenvalue and its associated eigenvector. *Numer Math 35* (1980), 113–126.

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service (see page 409 for order form).]

```
      SUBROUTINE SICEDR(LD, N, A, T, Q, W, X, CW, CX, K, Y1, Y2, R, Z1,
     * WR, WI, JOB, INFO, INIT)
C
      INTEGER I, J, IP1, IERR, LD, N, K, JOB, INFO, INIT
      REAL A(LD,1), T(LD,1), Q(LD,1), WR(1), WI(1)
      REAL W, X(1), CW, CX(1), Y1(1), Y2(1), R(1), Z1(1)
C
C     THIS SUBROUTINE PROVIDES TWO FUNCTIONS DEPENDING ON
C     THE VALUE OF INIT.
C
C           1) TO REDUCE A MATRIX TO QUASI-TRIANGULAR FORM,
C              ACCUMULATING THE ORTHOGONAL TRANSFORMATIONS,
C              AND DETERMINE THE EIGENVALUES OF THE MATRIX.
C
C           2) TO IMPROVE THE ACCURACY OF AN EIGENVALUE AND
C              IMPROVE THE EIGENVECTOR OR COMPUTE AND IMPROVE THE
C              EIGENVECTOR, GIVEN THE ORIGINAL MATRIX, AN
C              APPROXIMATE EIGENVALUE, THE QUASI-TRIANGULAR MATRIX,
C              AND THE ORTHOGONAL MATRIX WHICH PRODUCED THE
C              QUASI-TRIANGULAR FORM.
C
C     THESE FUNCTIONS ARE SIGNALED BY THE PARAMETER INIT.
```