5.1 Eigenvalues and Eigenvectors of a Symmetric Matrix

A. Purpose

Compute the N eigenvalues and eigenvectors of an N × N symmetric matrix A.

B. Usage

B.1 Program Prototype, Single Precision

REAL A(LDA,≥N) [LDA ≥ N], **EVAL**($\geq N$), **WORK**($\geq N$)

INTEGER LDA, N, IERR

Assign values to A(,), LDA, and N.

CALL SSYMQL(A, LDA, N, EVAL, WORK, IERR)

Results are returned in A(,) and EVAL().

B.2 Argument Definitions

A(,) [inout] On entry the locations on and below the diagonal of this array must contain the lower-triangular elements of the N × N symmetric matrix A. On return the eigenvectors of A will be stored as column vectors in the array A(,). These N eigenvectors will be mutually orthogonal and of unit Euclidean length. The eigenvector stored in column J will be associated with the eigenvalue stored in EVAL(J).

LDA [in] Dimension of the first subscript of the array A(,). Require LDA ≥ N.

N [in] Order of the symmetric matrix A. N ≥ 1.

EVAL() [out] Array in which the N eigenvalues of A will be stored by the subroutine. The eigenvalues will be sorted with the algebraically smallest eigenvalue first.

WORK() [scratch] An array of at least N locations used as temporary space.

IERR [out] On exit this is set to 0 if the QL algorithm converges, otherwise see Section E.

B.3 Modifications for Double Precision

Change SSYMQL to DSYMQL, and the REAL type statement to DOUBLE PRECISION.

C. Examples and Remarks

The following symmetric matrix A is given on page 55 of [1].

$$A = \begin{bmatrix} 5 & 4 & 1 & 1 \\ 4 & 5 & 1 & 1 \\ 1 & 1 & 4 & 2 \\ 1 & 1 & 2 & 4 \end{bmatrix}$$

The eigenvalues of this matrix are 1, 2, 5, and 10. Unnormalized eigenvectors associated with these eigenvalues are (1, -1, 0, 0), (0, 0, -1, 1), (-1, -1, 2, 2), and (2, 2, 1, 1), respectively.

The code in DRSSYMQL, given below, computes the eigenvalues and eigenvectors of this matrix. Output from this program is given in the file ODSSYMQL.

Before the call to SSYMQL, the matrix is saved in an array ASAV() in order to compute the relative residual matrix D defined as

$$D = (AW - W\Lambda)/\gamma$$

where W is the matrix whose columns are the computed eigenvectors of A, Λ is the diagonal matrix of eigenvalues, and γ is the maximum-row-sum norm of A.

Recall that if $\mathbf{v}$ is an eigenvector, then so is $\alpha\mathbf{v}$ for any nonzero scalar α. More generally, if an eigenvalue, λ, of a symmetric matrix occurs with multiplicity k, there will be an associated k-dimensional subspace in which every vector is an eigenvector for λ. This subroutine will return eigenvectors constituting an orthogonal basis for such an eigenspace.

D. Functional Description

The implicit-shift QL algorithm implemented in this subroutine is based on the Algol procedure given in [2], pp. 337–383. The code combines slightly modified EISPACK routines TRED2, and IMTQL2, see [3]. Modifications made are minor changes to convert the code to take advantage of Fortran 77; they should not affect results. TRED2 uses Householder orthogonal similarity transformations to transform the matrix A to tridiagonal form. IMTQL2 uses the QL algorithm with implicit shifts to reduce the off-diagonal elements of the tridiagonal matrix to a magnitude of approximately the last bit of the largest element of A.

The resulting diagonal elements are the eigenvalues of A. The matrix of eigenvectors is computed as the product of the orthogonal transformation matrices used in transforming A first to tridiagonal form and then to (almost) diagonal form.

The eigenvalues are sorted in nondecreasing algebraic order and the eigenvectors are permuted as necessary to correspond to the ordered eigenvalues.

References

1. R. T. Gregory and D. L. Karney, **A Collection of Matrices for Testing Computational Algorithms**, J. Wiley and Sons, New York (1969) 153 pages.
2. A. Dubrelle, R. S. Martin, and J. H. Wilkinson, *The implicit QL algorithm*, **Numerische Mathematik 12** (1968).
3. B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, **Matrix Eigensystem Routines — EISPACK Guide**, *Lecture Notes in Computer Science 6*, Springer Verlag, Berlin (1974) 387 pages.

E. Error Procedures and Restrictions

If the QL algorithm fails to converge in 30 iterations on the J^{th} eigenvalue the subroutine sets IERR = J. In this case $J-1$ eigenvalues and eigenvectors are computed correctly but the eigenvalues are not ordered. If N $\leq$ 0 on entry, IERR is set to -1. In either case an error message is printed using IERM1 of Chapter 19.2 with an error level of 0, before the return.

F. Supporting Information

The source language is ANSI Fortran 77.

Entry	Required Files
DSYMQL	DIMQL, DSYMQL, ERFIN, ERMSG, IERM1, IERV1
SSYMQL	ERFIN, ERMSG, IERM1, IERV1, SIMQL, SSYMQL

Converted by: F. T. Krogh, JPL, October 1991.

DRSSYMQL

```
c      program DRSSYMQL
c>> 1996-05-28 DRSSYMQL  Krogh Added external statement.
c>> 1994-10-19 DRSSYMQL  Krogh  Changes to use M77CON
c>> 1994-09-22 DRSSYMQL  CLL
c>> 1992-04-23 CLL
c>> 1992-03-04 DRSSYMQL  Krogh Initial version.
c      Demonstrate symmetric eigenvalue/eigenvector subroutine SSYMQL.
c      ------------------------------------------------------------------
c--S replaces "?": DR?SYMQL, ?SYMQL, ?VECP, ?MATP, ?DOT
c      ------------------------------------------------------------------
      integer I, IERR, J, LDA, N
      parameter (LDA = 4)
      real           A(LDA,LDA), ASAV(LDA,LDA), ANORM, D(LDA,LDA)
      external SDOT
      real           SDOT, EVAL(LDA), WORK(LDA)
      data A(1,1)          /  5.0e0 /
      data (A(2,J), J=1,2) /  4.0e0,  5.0e0 /
      data (A(3,J), J=1,3) /  1.0e0,  1.0e0, 4.0e0 /
      data (A(4,J), J=1,4) /  1.0e0,  1.0e0, 2.0e0, 4.0e0 /
      data ANORM / 11.0e0 /
      data N /LDA/
c      ------------------------------------------------------------------
      print*,'DRSSYMQL..  Demo driver for SSYMQL.'
c
c      First copy A() to ASAV() for later residual check.
c
      do 20 I = 1,N
         do 10 J = 1,I
            ASAV(I,J) = A(I,J)
            ASAV(J,I) = ASAV(I,J)
   10    continue
   20 continue
      call SSYMQL(A, LDA, N, EVAL, WORK, IERR)
      if (IERR .eq. 0) then
         call SVECP(EVAL, N, '0 Eigenvalues')
         call SMATP(A, LDA, N, N,  '0 Eigenvectors as column vectors')
```

```
c
c        As a check compute D = (ASAV*EVEC - EVEC*EVAL) / ANORM.
c        The EVEC's are in the array A().
c        Expect D to be close to machine precision.
c
         do 40 J = 1, N
            do 30 I = 1, N
               D(I, J) = (SDOT(N, ASAV(I,1), LDA, A(1,J), 1) -
     *                   A(I,J) * EVAL(J)) / ANORM
   30       continue
   40    continue
         call SMATP(D, LDA, N, N,
     *        '0 Residual matrix D = (A*EVEC - EVEC*EVAL) / ANORM')

      else
         print '(/a,i5)',' Convergence failure in SSYMQL, IERR =',IERR
      end if
      stop
      end
```

ODSSYMQL

```
DRSSYMQL..  Demo driver for SSYMQL.

 Eigenvalues
  1 TO    4     1.000000        1.999998        5.000000        10.00000

 Eigenvectors as column vectors

                  COL    1          COL    2          COL    3          COL    4
ROW    1        0.7071068       3.3527613E-08   0.3162276         0.6324556
ROW    2       -0.7071068      -2.7939677E-08   0.3162276         0.6324557
ROW    3       -2.4333493E-08   0.7071068      -0.6324555         0.3162276
ROW    4        0.000000       -0.7071065      -0.6324557         0.3162278

 Residual matrix D = (A*EVEC - EVEC*EVAL) / ANORM

                  COL    1          COL    2          COL    3          COL    4
ROW    1        2.1674417E-08   2.6415700E-08  -4.3348834E-08   0.000000
ROW    2        1.6255813E-08   3.2172956E-08  -4.3348834E-08  -4.3348834E-08
ROW    3       -1.2178032E-09   1.6255812E-07  -6.5023251E-08   1.5172091E-07
ROW    4        9.9433251E-10  -5.4186042E-08   2.1674417E-08  -4.3348834E-08
```