

CITED REFERENCES AND FURTHER READING:

- Golub, G.H., and Van Loan, C.F. 1989, *Matrix Computations*, 2nd ed. (Baltimore: Johns Hopkins University Press), §5.1. [1]
- Smith, B.T., et al. 1976, *Matrix Eigensystem Routines — EISPACK Guide*, 2nd ed., vol. 6 of *Lecture Notes in Computer Science* (New York: Springer-Verlag).
- Wilkinson, J.H., and Reinsch, C. 1971, *Linear Algebra*, vol. II of *Handbook for Automatic Computation* (New York: Springer-Verlag). [2]

11.3 Eigenvalues and Eigenvectors of a Tridiagonal Matrix

Evaluation of the Characteristic Polynomial

Once our original, real, symmetric matrix has been reduced to tridiagonal form, one possible way to determine its eigenvalues is to find the roots of the characteristic polynomial $p_n(\lambda)$ directly. The characteristic polynomial of a tridiagonal matrix can be evaluated for any trial value of λ by an efficient recursion relation (see [1], for example). The polynomials of lower degree produced during the recurrence form a Sturmian sequence that can be used to localize the eigenvalues to intervals on the real axis. A root-finding method such as bisection or Newton's method can then be employed to refine the intervals. The corresponding eigenvectors can then be found by inverse iteration (see §11.7).

Procedures based on these ideas can be found in [2,3]. If, however, more than a small fraction of all the eigenvalues and eigenvectors are required, then the factorization method next considered is much more efficient.

The QR and QL Algorithms

The basic idea behind the *QR* algorithm is that any real matrix can be decomposed in the form

$$\mathbf{A} = \mathbf{Q} \cdot \mathbf{R} \quad (11.3.1)$$

where \mathbf{Q} is orthogonal and \mathbf{R} is upper triangular. For a general matrix, the decomposition is constructed by applying Householder transformations to annihilate successive columns of \mathbf{A} below the diagonal (see §2.10).

Now consider the matrix formed by writing the factors in (11.3.1) in the opposite order:

$$\mathbf{A}' = \mathbf{R} \cdot \mathbf{Q} \quad (11.3.2)$$

Since \mathbf{Q} is orthogonal, equation (11.3.1) gives $\mathbf{R} = \mathbf{Q}^T \cdot \mathbf{A}$. Thus equation (11.3.2) becomes

$$\mathbf{A}' = \mathbf{Q}^T \cdot \mathbf{A} \cdot \mathbf{Q} \quad (11.3.3)$$

We see that \mathbf{A}' is an orthogonal transformation of \mathbf{A} .

You can verify that a QR transformation preserves the following properties of a matrix: symmetry, tridiagonal form, and Hessenberg form (to be defined in §11.5).

There is nothing special about choosing one of the factors of \mathbf{A} to be upper triangular; one could equally well make it lower triangular. This is called the QL algorithm, since

$$\mathbf{A} = \mathbf{Q} \cdot \mathbf{L} \quad (11.3.4)$$

where \mathbf{L} is lower triangular. (The standard, but confusing, nomenclature R and L stands for whether the *right* or *left* of the matrix is nonzero.)

Recall that in the Householder reduction to tridiagonal form in §11.2, we started in the n th (last) column of the original matrix. To minimize roundoff, we then exhorted you to put the biggest elements of the matrix in the lower right-hand corner, if you can. If we now wish to diagonalize the resulting tridiagonal matrix, the QL algorithm will have smaller roundoff than the QR algorithm, so we shall use QL henceforth.

The QL algorithm consists of a *sequence* of orthogonal transformations:

$$\begin{aligned} \mathbf{A}_s &= \mathbf{Q}_s \cdot \mathbf{L}_s \\ \mathbf{A}_{s+1} &= \mathbf{L}_s \cdot \mathbf{Q}_s \quad (= \mathbf{Q}_s^T \cdot \mathbf{A}_s \cdot \mathbf{Q}_s) \end{aligned} \quad (11.3.5)$$

The following (nonobvious!) theorem is the basis of the algorithm for a general matrix \mathbf{A} : (i) If \mathbf{A} has eigenvalues of different absolute value $|\lambda_i|$, then $\mathbf{A}_s \rightarrow$ [lower triangular form] as $s \rightarrow \infty$. The eigenvalues appear on the diagonal in increasing order of absolute magnitude. (ii) If \mathbf{A} has an eigenvalue $|\lambda_i|$ of multiplicity p , $\mathbf{A}_s \rightarrow$ [lower triangular form] as $s \rightarrow \infty$, except for a diagonal block matrix of order p , whose eigenvalues $\rightarrow \lambda_i$. The proof of this theorem is fairly lengthy; see, for example, [4].

The workload in the QL algorithm is $O(n^3)$ per iteration for a general matrix, which is prohibitive. However, the workload is only $O(n)$ per iteration for a tridiagonal matrix and $O(n^2)$ for a Hessenberg matrix, which makes it highly efficient on these forms.

In this section we are concerned only with the case where \mathbf{A} is a real, symmetric, tridiagonal matrix. All the eigenvalues λ_i are thus real. According to the theorem, if any λ_i has a multiplicity p , then there must be at least $p - 1$ zeros on the sub- and superdiagonal. Thus the matrix can be split into submatrices that can be diagonalized separately, and the complication of diagonal blocks that can arise in the general case is irrelevant.

In the proof of the theorem quoted above, one finds that in general a super-diagonal element converges to zero like

$$a_{ij}^{(s)} \sim \left(\frac{\lambda_i}{\lambda_j} \right)^s \quad (11.3.6)$$

Although $\lambda_i < \lambda_j$, convergence can be slow if λ_i is close to λ_j . Convergence can be accelerated by the technique of *shifting*: If k is any constant, then $\mathbf{A} - k\mathbf{1}$ has

eigenvalues $\lambda_i - k$. If we decompose

$$\mathbf{A}_s - k_s \mathbf{1} = \mathbf{Q}_s \cdot \mathbf{L}_s \quad (11.3.7)$$

so that

$$\begin{aligned} \mathbf{A}_{s+1} &= \mathbf{L}_s \cdot \mathbf{Q}_s + k_s \mathbf{1} \\ &= \mathbf{Q}_s^T \cdot \mathbf{A}_s \cdot \mathbf{Q}_s \end{aligned} \quad (11.3.8)$$

then the convergence is determined by the ratio

$$\frac{\lambda_i - k_s}{\lambda_j - k_s} \quad (11.3.9)$$

The idea is to choose the shift k_s at each stage to maximize the rate of convergence. A good choice for the shift initially would be k_s close to λ_1 , the smallest eigenvalue. Then the first row of off-diagonal elements would tend rapidly to zero. However, λ_1 is not usually known *a priori*. A very effective strategy in practice (although there is no proof that it is optimal) is to compute the eigenvalues of the leading 2×2 diagonal submatrix of \mathbf{A} . Then set k_s equal to the eigenvalue closer to a_{11} .

More generally, suppose you have already found $r - 1$ eigenvalues of \mathbf{A} . Then you can *deflate* the matrix by crossing out the first $r - 1$ rows and columns, leaving

$$\mathbf{A} = \begin{bmatrix} 0 & & \cdots & \cdots & & 0 \\ & \cdots & & & & \\ & & 0 & & & \\ \vdots & & d_r & e_r & & \vdots \\ \vdots & & e_r & d_{r+1} & & \\ & & & & \cdots & 0 \\ 0 & & & & d_{n-1} & e_{n-1} \\ & & \cdots & 0 & e_{n-1} & d_n \end{bmatrix} \quad (11.3.10)$$

Choose k_s equal to the eigenvalue of the leading 2×2 submatrix that is closer to d_r . One can show that the convergence of the algorithm with this strategy is generally cubic (and at worst quadratic for degenerate eigenvalues). This rapid convergence is what makes the algorithm so attractive.

Note that with shifting, the eigenvalues no longer necessarily appear on the diagonal in order of increasing absolute magnitude. The routine `eigsrt` (§11.1) can be used if required.

As we mentioned earlier, the *QL* decomposition of a general matrix is effected by a sequence of Householder transformations. For a tridiagonal matrix, however, it is more efficient to use plane rotations \mathbf{P}_{pq} . One uses the sequence $\mathbf{P}_{12}, \mathbf{P}_{23}, \dots, \mathbf{P}_{n-1,n}$ to annihilate the elements $a_{12}, a_{23}, \dots, a_{n-1,n}$. By symmetry, the subdiagonal elements $a_{21}, a_{32}, \dots, a_{n,n-1}$ will be annihilated too. Thus each \mathbf{Q}_s is a product of plane rotations:

$$\mathbf{Q}_s^T = \mathbf{P}_1^{(s)} \cdot \mathbf{P}_2^{(s)} \cdots \mathbf{P}_{n-1}^{(s)} \quad (11.3.11)$$


```

do 11 i=2,n
    e(i-1)=e(i)
enddo 11
e(n)=0.
do 15 l=1,n
    iter=0
1    do 12 m=l,n-1
        dd=abs(d(m))+abs(d(m+1))
        if (abs(e(m))+dd.eq.dd) goto 2
    enddo 12
    m=n
2    if(m.ne.1)then
        if(iter.eq.30)pause 'too many iterations in tqli'
        iter=iter+1
        g=(d(l+1)-d(l))/(2.*e(l))
        r=pythag(g,1.)
        g=d(m)-d(l)+e(l)/(g+sign(r,g))
        s=1.
        c=1.
        p=0.
        do 14 i=m-1,l,-1
            f=s*e(i)
            b=c*e(i)
            r=pythag(f,g)
            e(i+1)=r
            if(r.eq.0.)then
                d(i+1)=d(i+1)-p
                e(m)=0.
                goto 1
            endif
            s=f/r
            c=g/r
            g=d(i+1)-p
            r=(d(i)-g)*s+2.*c*b
            p=s*r
            d(i+1)=g+p
            g=c*r-b
C    Omit lines from here ...
        do 13 k=1,n
            f=z(k,i+1)
            z(k,i+1)=s*z(k,i)+c*f
            z(k,i)=c*z(k,i)-s*f
        enddo 13
C    ... to here when finding only eigenvalues.
        enddo 14
        d(l)=d(l)-p
        e(l)=g
        e(m)=0.
        goto 1
    endif
enddo 15
return
END

```

Convenient to renumber the elements of e .

Look for a single small subdiagonal element to split the matrix.

Form shift.

This is $d_m - k_s$.

A plane rotation as in the original QL , followed by Givens rotations to restore tridiagonal form.

Recover from underflow.

Form eigenvectors.

CITED REFERENCES AND FURTHER READING:

- Acton, F.S. 1970, *Numerical Methods That Work*; 1990, corrected edition (Washington: Mathematical Association of America), pp. 331–335. [1]
- Wilkinson, J.H., and Reinsch, C. 1971, *Linear Algebra*, vol. II of *Handbook for Automatic Computation* (New York: Springer-Verlag). [2]

World Wide Web sample page from NUMERICAL RECIPES IN FORTRAN 77: THE ART OF SCIENTIFIC COMPUTING (ISBN 0-521-43064-X)
 Copyright (C) 1988-1992 by Cambridge University Press. Programs Copyright (C) 1988-1992 by Numerical Recipes Software.
 Permission is granted for internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books, diskettes, or CDROMs visit website <http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to trade@cup.cam.ac.uk (outside North America).

Smith, B.T., et al. 1976, *Matrix Eigensystem Routines — EISPACK Guide*, 2nd ed., vol. 6 of Lecture Notes in Computer Science (New York: Springer-Verlag). [3]

Stoer, J., and Bulirsch, R. 1980, *Introduction to Numerical Analysis* (New York: Springer-Verlag), §6.6.6. [4]

11.4 Hermitian Matrices

The complex analog of a real, symmetric matrix is a Hermitian matrix, satisfying equation (11.0.4). Jacobi transformations can be used to find eigenvalues and eigenvectors, as also can Householder reduction to tridiagonal form followed by *QL* iteration. Complex versions of the previous routines `jacobi`, `tred2`, and `tqli` are quite analogous to their real counterparts. For working routines, consult [1,2].

An alternative, using the routines in this book, is to convert the Hermitian problem to a real, symmetric one: If $\mathbf{C} = \mathbf{A} + i\mathbf{B}$ is a Hermitian matrix, then the $n \times n$ complex eigenvalue problem

$$(\mathbf{A} + i\mathbf{B}) \cdot (\mathbf{u} + i\mathbf{v}) = \lambda(\mathbf{u} + i\mathbf{v}) \quad (11.4.1)$$

is equivalent to the $2n \times 2n$ real problem

$$\begin{bmatrix} \mathbf{A} & -\mathbf{B} \\ \mathbf{B} & \mathbf{A} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \quad (11.4.2)$$

Note that the $2n \times 2n$ matrix in (11.4.2) is symmetric: $\mathbf{A}^T = \mathbf{A}$ and $\mathbf{B}^T = -\mathbf{B}$ if \mathbf{C} is Hermitian.

Corresponding to a given eigenvalue λ , the vector

$$\begin{bmatrix} -\mathbf{v} \\ \mathbf{u} \end{bmatrix} \quad (11.4.3)$$

is also an eigenvector, as you can verify by writing out the two matrix equations implied by (11.4.2). Thus if $\lambda_1, \lambda_2, \dots, \lambda_n$ are the eigenvalues of \mathbf{C} , then the $2n$ eigenvalues of the augmented problem (11.4.2) are $\lambda_1, \lambda_1, \lambda_2, \lambda_2, \dots, \lambda_n, \lambda_n$; each, in other words, is repeated twice. The eigenvectors are pairs of the form $\mathbf{u} + i\mathbf{v}$ and $i(\mathbf{u} + i\mathbf{v})$; that is, they are the same up to an inessential phase. Thus we solve the augmented problem (11.4.2), and choose one eigenvalue and eigenvector from each pair. These give the eigenvalues and eigenvectors of the original matrix \mathbf{C} .

Working with the augmented matrix requires a factor of 2 more storage than the original complex matrix. In principle, a complex algorithm is also a factor of 2 more efficient in computer time than is the solution of the augmented problem. In practice, most complex implementations do not achieve this factor unless they are written entirely in real arithmetic. (Good library routines always do this.)

CITED REFERENCES AND FURTHER READING:

Wilkinson, J.H., and Reinsch, C. 1971, *Linear Algebra*, vol. II of *Handbook for Automatic Computation* (New York: Springer-Verlag). [1]

Smith, B.T., et al. 1976, *Matrix Eigensystem Routines — EISPACK Guide*, 2nd ed., vol. 6 of Lecture Notes in Computer Science (New York: Springer-Verlag). [2]

World Wide Web sample page from NUMERICAL RECIPES IN FORTRAN 77: THE ART OF SCIENTIFIC COMPUTING (ISBN 0-521-43064-X)
Copyright (C) 1988-1992 by Cambridge University Press. Programs Copyright (C) 1988-1992 by Numerical Recipes Software.
Permission is granted for internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-readable files (including this one), to any server computer, is strictly prohibited. To order Numerical Recipes books, diskettes, or CDROMs visit website <http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to trade@cup.cam.ac.uk (outside North America).