

## Chapter B11. Eigensystems

```

SUBROUTINE jacobi(a,d,v,nrot)
USE nrtype; USE nrutil, ONLY : assert_eq,get_diag,nrerror,unit_matrix,&
    upper_triangle
IMPLICIT NONE
INTEGER(I4B), INTENT(OUT) :: nrot
REAL(SP), DIMENSION(:), INTENT(OUT) :: d
REAL(SP), DIMENSION(:,:), INTENT(INOUT) :: a
REAL(SP), DIMENSION(:,:), INTENT(OUT) :: v
    Computes all eigenvalues and eigenvectors of a real symmetric  $N \times N$  matrix a. On output,
    elements of a above the diagonal are destroyed. d is a vector of length  $N$  that returns the
    eigenvalues of a. v is an  $N \times N$  matrix whose columns contain, on output, the normalized
    eigenvectors of a. nrot returns the number of Jacobi rotations that were required.
INTEGER(I4B) :: i,ip,iq,n
REAL(SP) :: c,g,h,s,sm,t,tau,theta,tresh
REAL(SP), DIMENSION(size(d)) :: b,z
n=assert_eq((/size(a,1),size(a,2),size(d),size(v,1),size(v,2)/),'jacobi')
call unit_matrix(v(:,:))          Initialize v to the identity matrix.
b(:)=get_diag(a(:,:))             Initialize b and d to the diagonal of
d(:)=b(:)                          a.
z(:)=0.0                           This vector will accumulate terms of
nrot=0                             the form  $ta_{pq}$  as in eq. (11.1.14).
do i=1,50
    sm=sum(abs(a),mask=upper_triangle(n,n))    Sum off-diagonal elements.
    if (sm == 0.0) RETURN
    The normal return, which relies on quadratic convergence to machine underflow.
    tresh=merge(0.2_sp*sm/n**2,0.0_sp, i < 4 )
    On the first three sweeps, we will rotate only if tresh exceeded.
    do ip=1,n-1
        do iq=ip+1,n
            g=100.0_sp*abs(a(ip,iq))
            After four sweeps, skip the rotation if the off-diagonal element is small.
            if ((i > 4) .and. (abs(d(ip))+g == abs(d(ip))) &
                .and. (abs(d(iq))+g == abs(d(iq)))) then
                a(ip,iq)=0.0
            else if (abs(a(ip,iq)) > tresh) then
                h=d(iq)-d(ip)
                if (abs(h)+g == abs(h)) then
                    t=a(ip,iq)/h                     $t = 1/(2\theta)$ 
                else
                    theta=0.5_sp*h/a(ip,iq)          Equation (11.1.10).
                    t=1.0_sp/(abs(theta)+sqrt(1.0_sp+theta**2))
                    if (theta < 0.0) t=-t
                end if
                c=1.0_sp/sqrt(1+t**2)
                s=t*c
                tau=s/(1.0_sp+c)
                h=t*a(ip,iq)
                z(ip)=z(ip)-h
                z(iq)=z(iq)+h
                d(ip)=d(ip)-h
                d(iq)=d(iq)+h
                a(ip,iq)=0.0
            end do
        end do
    end do

```


Sample page from NUMERICAL RECIPES IN FORTRAN 90: The Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
 Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 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](mailto:trade@cup.cam.ac.uk) (outside North America).


```

      call jrotate(a(1:ip-1,ip),a(1:ip-1,iq))
      Case of rotations  $1 \leq j < p$ .
      call jrotate(a(ip,ip+1:iq-1),a(ip+1:iq-1,iq))
      Case of rotations  $p < j < q$ .
      call jrotate(a(ip,iq+1:n),a(iq,iq+1:n))
      Case of rotations  $q < j \leq n$ .
      call jrotate(v(:,ip),v(:,iq))
      nrot=nrot+1
    end if
  end do
end do
b(:)=b(:)+z(:)
d(:)=b(:)
z(:)=0.0
                                Update d with the sum of  $ta_{pq}$ ,
                                and reinitialize z.
call nrerror('too many iterations in jacobi')
CONTAINS

SUBROUTINE jrotate(a1,a2)
REAL(SP), DIMENSION(:), INTENT(INOUT) :: a1,a2
REAL(SP), DIMENSION(size(a1)) :: wk1
wk1(:)=a1(:)
a1(:)=a1(:)-s*(a2(:)+a1(:)*tau)
a2(:)=a2(:)+s*(wk1(:)-a2(:)*tau)
END SUBROUTINE jrotate
END SUBROUTINE jacobi

```

 As discussed in Volume 1, `jacobi` is generally not competitive with `tqli` in terms of efficiency. However, `jacobi` can be parallelized whereas `tqli` uses an intrinsically serial algorithm. The version of `jacobi` implemented here is likely to be adequate for a small-scale parallel (SSP) machine, but is probably still not competitive with `tqli`. For a massively multiprocessor (MMP) machine, the order of the rotations needs to be chosen in a more complicated pattern than here so that the rotations can be executed in parallel. In this case the Jacobi algorithm may well turn out to be the method of choice. Parallel replacements for `tqli` based on a divide and conquer algorithm have also been proposed. See the discussion after `tqli` on p. 1229.

 `call unit_matrix...b(:)=get_diag...` These routines in `nrutil` both require access to the diagonal of a matrix, an operation that is not conveniently provided for in Fortran 90. We have split them off into `nrutil` in case your compiler provides parallel library routines so you can replace our standard versions.

`sm=sum(abs(a),mask=upper_triangle(n,n))` The `upper_triangle` function in `nrutil` returns an upper triangular logical mask. As used here, the mask is true everywhere in the upper triangle of an  $n \times n$  matrix, excluding the diagonal. An optional integer argument `extra` allows additional diagonals to be set to true. With `extra=1` the upper triangle including the diagonal would be true. By using the mask, we can conveniently sum over the desired matrix elements in parallel.

`SUBROUTINE jrotate(a1,a2)` This internal subroutine also uses the values of `s` and `tau` from the calling subroutine `jacobi`. Variables in the calling routine are visible to an internal subprogram, but you should be circumspect in making use of this fact. It is easy to overwrite a value in the calling program inadvertently, and it is

Sample page from NUMERICAL RECIPES IN FORTRAN 90: The Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
 Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 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](mailto:trade@cup.cam.ac.uk) (outside North America).

often difficult to figure out the logic of an internal routine if not all its variables are declared explicitly. However, `jrotate` is so simple that there is no danger here.

★   ★   ★

```
SUBROUTINE eigsrt(d,v)
USE nrtype; USE nrutil, ONLY : assert_eq,imaxloc,swap
IMPLICIT NONE
REAL(SP), DIMENSION(:), INTENT(INOUT) :: d
REAL(SP), DIMENSION(:,:), INTENT(INOUT) :: v
    Given the eigenvalues d and eigenvectors v as output from jacobi (§11.1) or tqli (§11.3),
    this routine sorts the eigenvalues into descending order, and rearranges the columns of v
    correspondingly. The method is straight insertion.
INTEGER(I4B) :: i,j,n
n=assert_eq(size(d),size(v,1),size(v,2),'eigsrt')
do i=1,n-1
    j=imaxloc(d(i:n))+i-1
    if (j /= i) then
        call swap(d(i),d(j))
        call swap(v(:,i),v(:,j))
    end if
end do
END SUBROUTINE eigsrt
```



`j=imaxloc...` See discussion of `imaxloc` on p. 1017.

`call swap...` See discussion of overloaded versions of `swap` after `amoeba` on p. 1210.

★   ★   ★

```
SUBROUTINE tred2(a,d,e,novectors)
USE nrtype; USE nrutil, ONLY : assert_eq,outerprod
IMPLICIT NONE
REAL(SP), DIMENSION(:,:), INTENT(INOUT) :: a
REAL(SP), DIMENSION(:), INTENT(OUT) :: d,e
LOGICAL(LGT), OPTIONAL, INTENT(IN) :: novectors
    Householder reduction of a real, symmetric,  $N \times N$  matrix a. On output, a is replaced
    by the orthogonal matrix Q effecting the transformation. d returns the diagonal elements
    of the tridiagonal matrix, and e the off-diagonal elements, with e(1)=0. If the optional
    argument novectors is present, only eigenvalues are to be found subsequently, in which
    case a contains no useful information on output.
INTEGER(I4B) :: i,j,l,n
REAL(SP) :: f,g,h,hh,scale
REAL(SP), DIMENSION(size(a,1)) :: gg
LOGICAL(LGT), SAVE :: yesvec=.true.
n=assert_eq(size(a,1),size(a,2),size(d),size(e),'tred2')
if (present(novectors)) yesvec=.not. novectors
do i=n,2,-1
    l=i-1
    h=0.0
    if (l > 1) then
        scale=sum(abs(a(i,1:l)))
        if (scale == 0.0) then
            e(i)=a(i,l)
            Skip transformation.
        else
            a(i,1:l)=a(i,1:l)/scale
            h=sum(a(i,1:l)**2)
            Use scaled a's for transformation.
            Form  $\sigma$  in h.
        end if
    end if
end do
```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: The Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 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](mailto:trade@cup.cam.ac.uk) (outside North America).

```

      f=a(i,1)
      g=-sign(sqrt(h),f)
      e(i)=scale*g
      h=h-f*g
      a(i,1)=f-g
      if (yesvec) a(1:1,i)=a(i,1:1)/h
      do j=1,1
        e(j)=(dot_product(a(j,1:j),a(i,1:j)) &
              +dot_product(a(j+1:1,j),a(i,j+1:1)))/h
      end do
      f=dot_product(e(1:1),a(i,1:1))
      hh=f/(h+h)
      e(1:1)=e(1:1)-hh*a(i,1:1)
      do j=1,1
        a(j,1:j)=a(j,1:j)-a(i,j)*e(1:j)-e(j)*a(i,1:j)
      end do
    end if
  else
    e(i)=a(i,1)
  end if
  d(i)=h
end do
if (yesvec) d(1)=0.0
e(1)=0.0
do i=1,n
  if (yesvec) then
    l=i-1
    if (d(i) /= 0.0) then
      This block skipped when i=1. Use u and u/H stored in a to form P · Q.
      gg(1:1)=matmul(a(i,1:1),a(1:1,1:1))
      a(1:1,1:1)=a(1:1,1:1)-outerprod(a(1:1,i),gg(1:1))
    end if
    d(i)=a(i,i)
    a(i,i)=1.0
    a(i,1:1)=0.0
    a(1:1,i)=0.0
  else
    d(i)=a(i,i)
  end if
end do
END SUBROUTINE tred2

```

Now  $h$  is equation (11.2.4).  
 Store  $\mathbf{u}$  in the  $i$ th row of  $\mathbf{a}$ .  
 Store  $\mathbf{u}/H$  in  $i$ th column of  $\mathbf{a}$ .  
 Store elements of  $\mathbf{p}$  in temporarily unused elements of  $\mathbf{e}$ .  
 Form  $K$ , equation (11.2.11).  
 Form  $\mathbf{q}$  and store in  $\mathbf{e}$  overwriting  $\mathbf{p}$ .  
 Reduce  $\mathbf{a}$ , equation (11.2.13).  
 Begin accumulation of transformation matrices.  
 Reset row and column of  $\mathbf{a}$  to identity matrix for next iteration.



This routine gives a nice example of the usefulness of optional arguments. The routine is written under the assumption that usually you will want to find both eigenvalues and eigenvectors. In this case you just supply the arguments `a`, `d`, and `e`. If, however, you want only eigenvalues, you supply the additional logical argument `novectors` with the value `.true.`. The routine then skips the unnecessary computations. Supplying `novectors` with the value `.false.` has the same effect as omitting it.

\* \* \*

```

SUBROUTINE tqli(d,e,z)
USE nrtype; USE nrutil, ONLY : assert_eq,nrerror
USE nr, ONLY : pythag
IMPLICIT NONE
REAL(SP), DIMENSION(:), INTENT(INOUT) :: d,e
REAL(SP), DIMENSION(:,:), OPTIONAL, INTENT(INOUT) :: z
  QL algorithm with implicit shifts, to determine the eigenvalues and eigenvectors of a real,
  symmetric, tridiagonal matrix, or of a real, symmetric matrix previously reduced by tred2

```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: The Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
 Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 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](mailto:trade@cup.cam.ac.uk) (outside North America).

§11.2.  $d$  is a vector of length  $N$ . On input, its elements are the diagonal elements of the tridiagonal matrix. On output, it returns the eigenvalues. The vector  $e$  inputs the subdiagonal elements of the tridiagonal matrix, with  $e(1)$  arbitrary. On output  $e$  is destroyed. When finding only the eigenvalues, the optional argument  $z$  is omitted. If the eigenvectors of a tridiagonal matrix are desired, the  $N \times N$  matrix  $z$  is input as the identity matrix. If the eigenvectors of a matrix that has been reduced by `tred2` are required, then  $z$  is input as the matrix output by `tred2`. In either case, the  $k$ th column of  $z$  returns the normalized eigenvector corresponding to  $d(k)$ .

```

INTEGER(I4B) :: i,iter,l,m,n,ndum
REAL(SP) :: b,c,dd,f,g,p,r,s
REAL(SP), DIMENSION(size(e)) :: ff
n=assert_eq(size(d),size(e),'tqli: n')
if (present(z)) ndum=assert_eq(n,size(z,1),size(z,2),'tqli: ndum')
e(:)=eoshift(e(:),1)          Convenient to renumber the elements of
                                e.
do l=1,n
  iter=0
  iterate: do
    do m=l,n-1
      dd=abs(d(m))+abs(d(m+1))
      if (abs(e(m))+dd == dd) exit
    end do
    if (m == 1) exit iterate
    if (iter == 30) call nrerror('too many iterations in tqli')
    iter=iter+1
    g=(d(l+1)-d(l))/(2.0_sp*e(l))
    r=pythag(g,1.0_sp)
    g=d(m)-d(l)+e(l)/(g+sign(r,g))
    s=1.0
    c=1.0
    p=0.0
    do i=m-1,l,-1
      f=s*e(i)
      b=c*e(i)
      r=pythag(f,g)
      e(i+1)=r
      if (r == 0.0) then
        d(i+1)=d(i+1)-p
        e(m)=0.0
        cycle iterate
      end if
      s=f/r
      c=g/r
      g=d(i+1)-p
      r=(d(i)-g)*s+2.0_sp*c*b
      p=s*r
      d(i+1)=g+p
      g=c*r-b
      if (present(z)) then
        ff(1:n)=z(1:n,i+1)
        z(1:n,i+1)=s*z(1:n,i)+c*ff(1:n)
        z(1:n,i)=c*z(1:n,i)-s*ff(1:n)
      end if
    end do
    d(l)=d(l)-p
    e(l)=g
    e(m)=0.0
  end do iterate
end do
END SUBROUTINE tqli

```

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.



The routine `tqli` is intrinsically serial. A parallel replacement based on a divide and conquer algorithm has been proposed [1,2]. The idea is to split the tridiagonal matrix recursively into two tridiagonal matrices of

Sample page from NUMERICAL RECIPES IN FORTRAN 90: The Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
 Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 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](mailto:trade@cup.cam.ac.uk) (outside North America).

half the size plus a correction. Given the eigensystems of the two smaller tridiagonal matrices, it is possible to join them together and add in the effect of the correction. When some small size of tridiagonal matrix is reached during the recursive splitting, its eigensystem is found directly with a routine like `tqli`. Each of these small problems is independent and can be assigned to an independent processor. The procedures for sewing together can also be done independently. For very large matrices, this algorithm can be an order of magnitude faster than `tqli` even on a serial machine, and no worse than a factor of 2 or 3 slower, depending on the matrix. Unfortunately the parallelism is not well expressed in Fortran 90. Also, the sewing together requires quite involved coding. For an implementation see the LAPACK routine `SSTEDC`. Another parallel strategy for eigensystems uses inverse iteration, where each eigenvalue and eigenvector can be found independently [3].



This routine uses `z` as an optional argument that is required only if eigenvectors are being found as well as eigenvalues.

`iterate:` do See discussion of named do loops after `simplx` on p. 1219.

\* \* \*

```

SUBROUTINE balanc(a)
USE nrtype; USE nrutil, ONLY : assert_eq
IMPLICIT NONE
REAL(SP), DIMENSION(:, :), INTENT(INOUT) :: a
REAL(SP), PARAMETER :: RADX=radix(a), SQRADX=RADX**2
    Given an  $N \times N$  matrix a, this routine replaces it by a balanced matrix with identical
    eigenvalues. A symmetric matrix is already balanced and is unaffected by this procedure.
    The parameter RADX is the machine's floating-point radix.
INTEGER(I4B) :: i, last, ndum
REAL(SP) :: c, f, g, r, s
ndum=assert_eq(size(a,1),size(a,2),'balanc')
do
    last=1
    do i=1,size(a,1)
        c=sum(abs(a(:,i)))-a(i,i)
        r=sum(abs(a(i,:)))-a(i,i)
        if (c /= 0.0 .and. r /= 0.0) then
            g=r/RADX
            f=1.0
            s=c+r
            do
                if (c >= g) exit
                f=f*RADX
                c=c*SQRADX
            end do
            g=r*RADX
            do
                if (c <= g) exit
                f=f/RADX
                c=c/SQRADX
            end do
            if ((c+r)/f < 0.95_sp*s) then
                last=0
                g=1.0_sp/f
                a(i,:)=a(i,:)*g
                a(:,i)=a(:,i)*f
            end if
        end if
    end do
    Apply similarity transformation.

```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: The Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
 Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 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](mailto:trade@cup.cam.ac.uk) (outside North America).

```

    end do
    if (last /= 0) exit
end do
END SUBROUTINE balanc

```

**f90** REAL(SP), PARAMETER :: RADX=radix(a)... Fortran 90 provides a nice collection of numeric inquiry intrinsic functions. Here we find the machine's floating-point radix. Note that only the type of the argument *a* affects the returned function value.

★   ★   ★

```

SUBROUTINE elmhes(a)
USE nrtype; USE nrutil, ONLY : assert_eq,imaxloc,outerprod,swap
IMPLICIT NONE
REAL(SP), DIMENSION(:, :), INTENT(INOUT) :: a
    Reduction to Hessenberg form by the elimination method. The real, nonsymmetric,  $N \times N$ 
    matrix a is replaced by an upper Hessenberg matrix with identical eigenvalues. Recom-
    mended, but not required, is that this routine be preceded by balanc. On output, the
    Hessenberg matrix is in elements a(i, j) with  $i \leq j + 1$ . Elements with  $i > j + 1$  are to be
    thought of as zero, but are returned with random values.
INTEGER(I4B) :: i,m,n
REAL(SP) :: x
REAL(SP), DIMENSION(size(a,1)) :: y
n=assert_eq(size(a,1),size(a,2),'elmhes')
do m=2,n-1
    i=imaxloc(abs(a(m:n,m-1)))+m-1
    x=a(i,m-1)
    if (i /= m) then
        call swap(a(i,m-1:n),a(m,m-1:n))
        call swap(a(:,i),a(:,m))
    end if
    if (x /= 0.0) then
        y(m+1:n)=a(m+1:n,m-1)/x
        a(m+1:n,m-1)=y(m+1:n)
        a(m+1:n,m:n)=a(m+1:n,m:n)-outerprod(y(m+1:n),a(m,m:n))
        a(:,m)=a(:,m)+matmul(a(:,m+1:n),y(m+1:n))
    end if
end do
END SUBROUTINE elmhes

```

**f90** *y*(*m*+1:*n*)=... If the four lines of code starting here were all coded for a serial machine in a single do-loop starting with `do i=m+1,n` (see Volume 1), it would pay to test whether *y* was zero because the next three lines could then be skipped for that value of *i*. There is no convenient way to do this here, even with a *where*, since the shape of the arrays on each of the three lines is different. For a parallel machine it is probably best just to do a few unnecessary multiplies and skip the test for zero values of *y*.

★   ★   ★

Sample page from NUMERICAL RECIPES IN FORTRAN 90: The Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
 Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 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](mailto:trade@cup.cam.ac.uk) (outside North America).

```

SUBROUTINE hqr(a,wr,wi)
USE nrtype; USE nrutil, ONLY : assert_eq,diagadd,nrerror,upper_triangle
IMPLICIT NONE
REAL(SP), DIMENSION(:), INTENT(OUT) :: wr,wi
REAL(SP), DIMENSION(:,,:), INTENT(INOUT) :: a
  Finds all eigenvalues of an  $N \times N$  upper Hessenberg matrix a. On input a can be exactly
  as output from elmhes §11.5; on output it is destroyed. The real and imaginary parts of
  the  $N$  eigenvalues are returned in wr and wi, respectively.
INTEGER(I4B) :: i,its,k,l,m,n,nn,mnnk
REAL(SP) :: anorm,p,q,r,s,t,u,v,w,x,y,z
REAL(SP), DIMENSION(size(a,1)) :: pp
n=assert_eq(size(a,1),size(a,2),size(wr),size(wi),'hqr')
anorm=sum(abs(a),mask=upper_triangle(n,n,extra=2))
  Compute matrix norm for possible use in locating single small subdiagonal element.
nn=n
t=0.0
do
  Gets changed only by an exceptional shift.
  Begin search for next eigenvalue: "Do while
  if (nn < 1) exit
  nn >= 1".
  its=0
  iterate: do
    Begin iteration.
    do l=nn,2,-1
      Look for single small subdiagonal element.
      s=abs(a(l-1,l-1))+abs(a(l,l))
      if (s == 0.0) s=anorm
      if (abs(a(l,l-1))+s == s) exit
    end do
    x=a(nn,nn)
    if (l == nn) then
      One root found.
      wr(nn)=x+t
      wi(nn)=0.0
      nn=nn-1
      exit iterate
      Go back for next eigenvalue.
    end if
    y=a(nn-1,nn-1)
    w=a(nn,nn-1)*a(nn-1,nn)
    if (l == nn-1) then
      Two roots found ...
      p=0.5_sp*(y-x)
      q=p**2+w
      z=sqrt(abs(q))
      x=x+t
      if (q >= 0.0) then
        ... a real pair ...
        z=p+sign(z,p)
        wr(nn)=x+z
        wr(nn-1)=wr(nn)
        if (z /= 0.0) wr(nn)=x-w/z
        wi(nn)=0.0
        wi(nn-1)=0.0
      else
        ... a complex pair.
        wr(nn)=x+p
        wr(nn-1)=wr(nn)
        wi(nn)=z
        wi(nn-1)=-z
      end if
      nn=nn-2
      exit iterate
      Go back for next eigenvalue.
    end if
    No roots found. Continue iteration.
    if (its == 30) call nrerror('too many iterations in hqr')
    if (its == 10 .or. its == 20) then
      Form exceptional shift.
      t=t+x
      call diagadd(a(1:nn,1:nn),-x)
      s=abs(a(nn,nn-1))+abs(a(nn-1,nn-2))
      x=0.75_sp*s
      y=x
      w=-0.4375_sp*s**2

```

Sample page from NUMERICAL RECIPES IN FORTRAN 90: The Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
 Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 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](mailto:trade@cup.cam.ac.uk) (outside North America).



```

end if
its=its+1
do m=nn-2,1,-1
    z=a(m,m)
    r=x-z
    s=y-z
    p=(r*s-w)/a(m+1,m)+a(m,m+1)
    q=a(m+1,m+1)-z-r-s
    r=a(m+2,m+1)
    s=abs(p)+abs(q)+abs(r)
    p=p/s
    q=q/s
    r=r/s
    if (m == 1) exit
    u=abs(a(m,m-1))*(abs(q)+abs(r))
    v=abs(p)*(abs(a(m-1,m-1))+abs(z)+abs(a(m+1,m+1)))
    if (u+v == v) exit
end do
do i=m+2,nn
    a(i,i-2)=0.0
    if (i /= m+2) a(i,i-3)=0.0
end do
do k=m,nn-1
    if (k /= m) then
        p=a(k,k-1)
        q=a(k+1,k-1)
        r=0.0
        if (k /= nn-1) r=a(k+2,k-1)
        x=abs(p)+abs(q)+abs(r)
        if (x /= 0.0) then
            p=p/x
            q=q/x
            r=r/x
        end if
    end if
    s=sign(sqrt(p**2+q**2+r**2),p)
    if (s /= 0.0) then
        if (k == m) then
            if (1 /= m) a(k,k-1)=-a(k,k-1)
        else
            a(k,k-1)=-s*x
        end if
        p=p+s
        x=p/s
        y=q/s
        z=r/s
        q=q/p
        r=r/p
        pp(k:nn)=a(k,k:nn)+q*a(k+1,k:nn)
        if (k /= nn-1) then
            pp(k:nn)=pp(k:nn)+r*a(k+2,k:nn)
            a(k+2,k:nn)=a(k+2,k:nn)-pp(k:nn)*z
        end if
        a(k+1,k:nn)=a(k+1,k:nn)-pp(k:nn)*y
        a(k,k:nn)=a(k,k:nn)-pp(k:nn)*x
        mnnk=min(nn,k+3)
        pp(1:mnnk)=x*a(1:mnnk,k)+y*a(1:mnnk,k+1)
        if (k /= nn-1) then
            pp(1:mnnk)=pp(1:mnnk)+z*a(1:mnnk,k+2)
            a(1:mnnk,k+2)=a(1:mnnk,k+2)-pp(1:mnnk)*r
        end if
        a(1:mnnk,k+1)=a(1:mnnk,k+1)-pp(1:mnnk)*q
        a(1:mnnk,k)=a(1:mnnk,k)-pp(1:mnnk)
    end if
end if

```

Form shift and then look for 2 consecutive small subdiagonal elements.

Equation (11.6.23).

Scale to prevent overflow or underflow.

Equation (11.6.26).

Double QR step on rows 1 to nn and columns m to nn.

Begin setup of Householder vector.

Scale to prevent overflow or underflow.

Equations (11.6.24).

Ready for row modification.

Column modification.

Sample page from NUMERICAL RECIPES IN FORTRAN 90: The Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
 Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 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](mailto:trade@cup.cam.ac.uk) (outside North America).

```

      end do
    end do iterate
  end do
END SUBROUTINE hqr

```

Go back for next iteration on current eigenvalue.



`anorm=sum(abs(a),mask=upper_triangle(n,n,extra=2)` See the discussion of `upper_triangle` after `jacobi` on p. 1226. Setting `extra=2` here picks out the upper Hessenberg part of the matrix.

`iterate: do` We use a named loop to improve the readability and structuring of the routine. The if-blocks that test for one or two roots end with `exit iterate`, transferring control back to the outermost loop and thus starting a search for the next root.

`call diagadd...` The routines that operate on the diagonal of a matrix are collected in `nrutil` partly so you can write clear code and partly in the hope that compiler writers will provide parallel library routines. Fortran 90 does not provide convenient parallel access to the diagonal of a matrix.

#### CITED REFERENCES AND FURTHER READING:

- Golub, G.H., and Van Loan, C.F. 1989, *Matrix Computations*, 2nd ed. (Baltimore: Johns Hopkins University Press), §8.6 and references therein. [1]  
 Sorensen, D.C., and Tang, P.T.P. 1991, *SIAM Journal on Numerical Analysis*, vol. 28, pp. 1752–1775. [2]  
 Lo, S.-S., Philippe, B., and Sameh, A. 1987, *SIAM Journal on Scientific and Statistical Computing*, vol. 8, pp. s155–s165. [3]

Sample page from NUMERICAL RECIPES IN FORTRAN 90: The Art of PARALLEL Scientific Computing (ISBN 0-521-57439-0)  
 Copyright (C) 1986-1996 by Cambridge University Press. Programs Copyright (C) 1986-1996 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](mailto:trade@cup.cam.ac.uk) (outside North America).