```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       Questo programma risolve l'equazione di Schrodinger
c       utilizzando il metodo alle differenze finite per le
c       derivate. L'equazione si trasforma nel calcolo degli
c       autovalori e autovettori di una matrice simmetrica.
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

        program schrodinger

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       dichiarazione parametri e variabili
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

        parameter (k=160)  ! dimensione dei vettori (numero di punti in r)
        parameter (lda=k,ldevec=k)    ! dimensione matrice
        double precision   dr,r(k),u(k),a(lda,k),elle,rmin,rmax
        double precision   eval(k),evec(ldevec,k)
        double precision omega
        double precision fv1(k), fv2(k)
        integer   i,j,matz,dim

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       passo di integrazione e dimensione della scatola
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

        open(13,file='U.dat', status='unknown')
        rmin=-6.
        rmax=6.
        dr=(rmax-rmin)/float(k+1)
        do i=1,k
           r(i)=rmin+dr*float(i)
        end do

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       potenziale (a scelta)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        dim=1    ! dimensionalita' :  1 per 1D ,  3 per 3D
        elle=0.

        alpha=1.! omega
        beta=1.
        mass=1.

        if (dim.eq.3) then
           write(5,11)
11         format('  momento angolare : ',$)
           read(5,*) elle
        end if

        do i=1,k
           u(i)=0.
```

```
c           u(i)= mass*alpha*(r(i)**2-beta**2)**2
c            U(i)= (beta**2*(16*alpha**2*r(i)**2*(-1 + r(i)**2)**2 -
c      &     (2*(8*alpha*r(i)**2 + 4*alpha*(-1 + r(i)**2)))/beta))/4.

c           u(i)= 10.*(r(i)**2-1.**2)**2

            U(i) = 0.5*mass*alpha**2*(r(i))**2

            write(13,*) r(i), U(i)
            if (dim.eq.3) u(i)=u(i)+elle*(elle+1.)/r(i)**2
         end do

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       matrice da diagonalizzare
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         do j=1,k
            do i=1,k
                 a(j,i)=0.
            end do
         end do
         a(1,1)=u(1)+2./dr**2/2.
         a(1,2)=-1./dr**2/2.
         do j=2,k-1
            a(j,j-1)=-1./dr**2/2.
            a(j,j)=2./dr**2/2.+u(j)
            a(j,j+1)=-1./dr**2/2.
         end do
         a(k,k-1)=-1./dr**2/2.
         a(k,k)=2./dr**2/2.+u(k)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       chiama la routine per autovalori e autovettori
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         print *,' '
         print *,' ... sto calcolando .........'
         matz=1    ! (opzione unica: autovalori e autovettori)
         call rs (k,k,a, eval,matz,evec,fv1,fv2,ierr)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       risultati
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
         print *,' '
         print *,' ....... fatto ! '
         print *,' '
         print *,' primi 6  autovalori :'
         do i=1,6
            print *,i,'  -----> ',eval(i)
         end do
         print *,' '
         print *,'  autovalori in  EIGENVALUES.DAT '
         print *,'  autovettori in EIGENVECTORS.DAT '
         print *,' '
         open(unit=1,file='eigenvectors.dat',status='unknown')
         do i=1,k
           if(dim.eq.3) then
```

```
            do j=1,5
               evec(i,j)=evec(i,j)/r(i)
            end do
          end if
          write(1,46) r(i),evec(i,1),evec(i,2),
     &  evec(i,3),evec(i,4),evec(i,5)
46        format(f8.4,x,5(f10.5,x))
        end do
        close(1)
        open(unit=1,file='eigenvalues.dat',status='unknown')
        do i=1,k
          write(1,47) r(i),u(i),((max(u(i),eval(j))),j=1,15)
47        format(17(f10.5,x))
        end do
        close(1)
        stop
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccc  fine programma principale ccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       diagonalizzazione di matrici simmetriche
c
      subroutine rs(nm,n,a,w,matz,z,fv1,fv2,ierr)
c
      integer n,nm,ierr,matz
      double precision a(nm,n),w(n),z(nm,n),fv1(n),fv2(n)
c
c     this subroutine calls the recommended sequence of
c     subroutines from the eigensystem subroutine package (eispack)
c     to find the eigenvalues and eigenvectors (if desired)
c     of a real symmetric matrix.
c
c     on input
c
c        nm  must be set to the row dimension of the two-dimensional
c        array parameters as declared in the calling program
c        dimension statement.
c
c        n  is the order of the matrix  a.
c
c        a  contains the real symmetric matrix.
c
c        matz  is an integer variable set equal to zero if
c        only eigenvalues are desired.  otherwise it is set to
c        any non-zero integer for both eigenvalues and eigenvectors.
c
c     on output
c
c        w  contains the eigenvalues in ascending order.
c
c        z  contains the eigenvectors if matz is not zero.
c
```

```
c        ierr  is an integer output variable set equal to an error
c           completion code described in the documentation for tqlrat
c           and tql2.  the normal completion code is zero.
c
c        fv1  and  fv2  are temporary storage arrays.
c
c     questions and comments should be directed to burton s. garbow,
c     mathematics and computer science div, argonne national laboratory
c
c     this version dated august 1983.
c
c     ------------------------------------------------------------------
c
      if (n .le. nm) go to 10
      ierr = 10 * n
      go to 50
c
   10 if (matz .ne. 0) go to 20
c     ........... find eigenvalues only ......         (enabled !!)
c     call  tred1(nm,n,a,w,fv1,fv2)
c     tqlrat encounters catastrophic underflow on the Vax
c     call  tqlrat(n,w,fv2,ierr)
c     call  tql1(n,w,fv1,ierr)
      go to 50
c     ........... find both eigenvalues and eigenvectors ..........
   20 call  tred2(nm,n,a,w,fv1,z)
      call  tql2(nm,n,w,fv1,z,ierr)
   50 return
      end
      subroutine tred2(nm,n,a,d,e,z)
c
      integer i,j,k,l,n,ii,nm,jp1
      double precision a(nm,n),d(n),e(n),z(nm,n)
      double precision f,g,h,hh,scale
c
c     this subroutine is a translation of the algol procedure tred2,
c     num. math. 11, 181-195(1968) by martin, reinsch, and wilkinson.
c     handbook for auto. comp., vol.ii-linear algebra, 212-226(1971).
c
c     this subroutine reduces a real symmetric matrix to a
c     symmetric tridiagonal matrix using and accumulating
c     orthogonal similarity transformations.
c
c     on input
c
c        nm must be set to the row dimension of two-dimensional
c          array parameters as declared in the calling program
c          dimension statement.
c
c        n is the order of the matrix.
c
c        a contains the real symmetric input matrix.  only the
c          lower triangle of the matrix need be supplied.
c
c     on output
```

```
c
c        d contains the diagonal elements of the tridiagonal matrix.
c
c        e contains the subdiagonal elements of the tridiagonal
c          matrix in its last n-1 positions.  e(1) is set to zero.
c
c        z contains the orthogonal transformation matrix
c          produced in the reduction.
c
c        a and z may coincide.  if distinct, a is unaltered.
c
c     questions and comments should be directed to burton s. garbow,
c     mathematics and computer science div, argonne national laboratory
c
c     this version dated august 1983.
c
c     ------------------------------------------------------------------
c
      do 100 i = 1, n
c
         do 80 j = i, n
   80    z(j,i) = a(j,i)
c
         d(i) = a(n,i)
  100 continue
c
      if (n .eq. 1) go to 510
c     .......... for i=n step -1 until 2 do -- ..........
      do 300 ii = 2, n
         i = n + 2 - ii
         l = i - 1
         h = 0.0d0
         scale = 0.0d0
         if (l .lt. 2) go to 130
c     .......... scale row (algol tol then not needed) ..........
         do 120 k = 1, l
  120    scale = scale + dabs(d(k))
c
         if (scale .ne. 0.0d0) go to 140
  130    e(i) = d(l)
c
         do 135 j = 1, l
            d(j) = z(l,j)
            z(i,j) = 0.0d0
            z(j,i) = 0.0d0
  135    continue
c
         go to 290
c
  140    do 150 k = 1, l
            d(k) = d(k) / scale
            h = h + d(k) * d(k)
  150    continue
c
         f = d(l)
```

```
         g = -dsign(dsqrt(h),f)
         e(i) = scale * g
         h = h - f * g
         d(l) = f - g
c     .......... form a*u ..........
         do 170 j = 1, l
  170    e(j) = 0.0d0
c
         do 240 j = 1, l
            f = d(j)
            z(j,i) = f
            g = e(j) + z(j,j) * f
            jp1 = j + 1
            if (l .lt. jp1) go to 220
c
            do 200 k = jp1, l
               g = g + z(k,j) * d(k)
               e(k) = e(k) + z(k,j) * f
  200       continue
c
  220       e(j) = g
  240    continue
c     .......... form p ..........
         f = 0.0d0
c
         do 245 j = 1, l
            e(j) = e(j) / h
            f = f + e(j) * d(j)
  245    continue
c
         hh = f / (h + h)
c     .......... form q ..........
         do 250 j = 1, l
  250    e(j) = e(j) - hh * d(j)
c     .......... form reduced a ..........
         do 280 j = 1, l
            f = d(j)
            g = e(j)
c
            do 260 k = j, l
  260       z(k,j) = z(k,j) - f * e(k) - g * d(k)
c
            d(j) = z(l,j)
            z(i,j) = 0.0d0
  280    continue
c
  290    d(i) = h
  300 continue
c     .......... accumulation of transformation matrices ..........
      do 500 i = 2, n
         l = i - 1
         z(n,l) = z(l,l)
         z(l,l) = 1.0d0
         h = d(i)
         if (h .eq. 0.0d0) go to 380
```

```
c
         do 330 k = 1, l
  330    d(k) = z(k,i) / h
c
         do 360 j = 1, l
            g = 0.0d0
c
            do 340 k = 1, l
  340       g = g + z(k,i) * z(k,j)
c
            do 360 k = 1, l
               z(k,j) = z(k,j) - g * d(k)
  360    continue
c
  380    do 400 k = 1, l
  400    z(k,i) = 0.0d0
c
  500 continue
c
  510 do 520 i = 1, n
         d(i) = z(n,i)
         z(n,i) = 0.0d0
  520 continue
c
      z(n,n) = 1.0d0
      e(1) = 0.0d0
      return
      end
      subroutine tql2(nm,n,d,e,z,ierr)
c
      integer i,j,k,l,m,n,ii,l1,l2,nm,mml,ierr
      double precision d(n),e(n),z(nm,n)
      double precision c,c2,c3,dl1,el1,f,g,h,p,r,s,s2,tst1,tst2,pythag
c
c     this subroutine is a translation of the algol procedure tql2,
c     num. math. 11, 293-306(1968) by bowdler, martin, reinsch, and
c     wilkinson.
c     handbook for auto. comp., vol.ii-linear algebra, 227-240(1971).
c
c     this subroutine finds the eigenvalues and eigenvectors
c     of a symmetric tridiagonal matrix by the ql method.
c     the eigenvectors of a full symmetric matrix can also
c     be found if  tred2  has been used to reduce this
c     full matrix to tridiagonal form.
c
c     on input
c
c        nm must be set to the row dimension of two-dimensional
c          array parameters as declared in the calling program
c          dimension statement.
c
c        n is the order of the matrix.
c       ~
c        d contains the diagonal elements of the input matrix.
c
```

```
c        e contains the subdiagonal elements of the input matrix
c          in its last n-1 positions.  e(1) is arbitrary.
c
c        z contains the transformation matrix produced in the
c          reduction by  tred2, if performed.  if the eigenvectors
c          of the tridiagonal matrix are desired, z must contain
c          the identity matrix.
c
c      on output
c
c        d contains the eigenvalues in ascending order.  if an
c          error exit is made, the eigenvalues are correct but
c          unordered for indices 1,2,...,ierr-1.
c
c        e has been destroyed.
c
c        z contains orthonormal eigenvectors of the symmetric
c          tridiagonal (or full) matrix.  if an error exit is made,
c          z contains the eigenvectors associated with the stored
c          eigenvalues.
c
c        ierr is set to
c          zero       for normal return,
c          j          if the j-th eigenvalue has not been
c                     determined after 30 iterations.
c
c     calls pythag for  dsqrt(a*a + b*b) .
c
c     questions and comments should be directed to burton s. garbow,
c     mathematics and computer science div, argonne national laboratory
c
c     this version dated august 1983.
c
c     ------------------------------------------------------------------
c
      ierr = 0
      if (n .eq. 1) go to 1001
c
      do 100 i = 2, n
  100 e(i-1) = e(i)
c
      f = 0.0d0
      tst1 = 0.0d0
      e(n) = 0.0d0
c
      do 240 l = 1, n
         j = 0
         h = dabs(d(l)) + dabs(e(l))
         if (tst1 .lt. h) tst1 = h
c     ..........  look for small sub-diagonal element ..........
         do 110 m = l, n
            tst2 = tst1 + dabs(e(m))
            if (tst2 .eq. tst1) go to 120
c     ..........  e(n) is always zero, so there is no exit
c                 through the bottom of the loop ..........
```

```
  110    continue
c
  120    if (m .eq. l) go to 220
  130    if (j .eq. 30) go to 1000
         j = j + 1
c     .......... form shift ..........
         l1 = l + 1
         l2 = l1 + 1
         g = d(l)
         p = (d(l1) - g) / (2.0d0 * e(l))
         r = pythag(p,1.0d0)
         d(l) = e(l) / (p + dsign(r,p))
         d(l1) = e(l) * (p + dsign(r,p))
         dl1 = d(l1)
         h = g - d(l)
         if (l2 .gt. n) go to 145
c
         do 140 i = l2, n
  140    d(i) = d(i) - h
c
  145    f = f + h
c     .......... ql transformation ..........
         p = d(m)
         c = 1.0d0
         c2 = c
         el1 = e(l1)
         s = 0.0d0
         mml = m - l
c     .......... for i=m-1 step -1 until l do -- ..........
         do 200 ii = 1, mml
            c3 = c2
            c2 = c
            s2 = s
            i = m - ii
            g = c * e(i)
            h = c * p
            r = pythag(p,e(i))
            e(i+1) = s * r
            s = e(i) / r
            c = p / r
            p = c * d(i) - s * g
            d(i+1) = h + s * (c * g + s * d(i))
c     .......... form vector ..........
            do 180 k = 1, n
               h = z(k,i+1)
               z(k,i+1) = s * z(k,i) + c * h
               z(k,i) = c * z(k,i) - s * h
  180       continue
c
  200    continue
c
         p = -s * s2 * c3 * el1 * e(l) / dl1
         e(l) = s * p
         d(l) = c * p
         tst2 = tst1 + dabs(e(l))
```

```
         if (tst2 .gt. tst1) go to 130
  220    d(l) = d(l) + f
  240 continue
c     .......... order eigenvalues and eigenvectors ..........
      do 300 ii = 2, n
         i = ii - 1
         k = i
         p = d(i)
c
         do 260 j = ii, n
            if (d(j) .ge. p) go to 260
            k = j
            p = d(j)
  260    continue
c
         if (k .eq. i) go to 300
         d(k) = d(i)
         d(i) = p
c
         do 280 j = 1, n
            p = z(j,i)
            z(j,i) = z(j,k)
            z(j,k) = p
  280    continue
c
  300 continue
c
      go to 1001
c     .......... set error -- no convergence to an
c                eigenvalue after 30 iterations ..........
1000 ierr = l
1001 return
      end
c***********************************************************************
      double precision function pythag(a,b)
      double precision a,b
c
c     finds dsqrt(a**2+b**2) without overflow or destructive underflow
c
      double precision p,r,s,t,u
      p = dmax1(dabs(a),dabs(b))
      if (p .eq. 0.0d0) go to 20
      r = (dmin1(dabs(a),dabs(b))/p)**2
   10 continue
         t = 4.0d0 + r
         if (t .eq. 4.0d0) go to 20
         s = r/t
         u = 1.0d0 + 2.0d0*s
         p = u*p
         r = (s/u)**2 * r
      go to 10
   20 pythag = p
      return
      end
```