プログラム pca3.f
入力ファイル：段階消磁データ(test.txt)
複数サンプルも可能。ただし消磁段階数をそろえる。
｛サンプル名(数字)　消磁レベル　磁化強度　偏角　伏角｝
8811 0 2.05E-04 182.0 -20.9
8811 50 1.21E-04 270.1 -46.5
8811 100 6.15E-05 289.1 -39.8
8811 150 3.26E-05 292.9 -34.7
8811 200 2.21E-05 301.6 -34.0
8811 250 1.67E-05 305.6 -33.2
8811 300 1.04E-05 301.6 -31.4
8811 350 7.26E-06 312.2 -32.3
8811 400 5.56E-06 276.5 -37.3
8811 450 4.54E-06 295.7 -48.4
8811 500 4.18E-06 288.4 -23.2

キー入力

```
write(6,*) 'How many samples ?'
read(5,*) ismp   ←　サンプル数（ここでは 1）
write(6,*) 'How many steps for demagnetization ?'
read(5,*) kstp   ←　消磁ステップ数（ここでは 11）
write(6,*) kkk, ' PCA from [beg] to [end] steps '
write(6,*) ' Input ibeg and iend ex.5 8'
read(5,*) istart, iend   ←　消磁ステップの何番目から何番目を使うか？
```

100Oe から 400Oe のデータを使うのであれば、３□９[Enter]

出力ファイル：result.txt
｛サンプル名(数字)　使用点数　開始消磁レベル　終了消磁レベル　MAD　偏角　伏角｝
8811　　9　50.00　450.00　　8.3　270.0　−46.3

プログラム

```
c*******************************************************
      program pca
c     use msflib
c
c     calculate principal components of data set
```

```
c
      dimension dat(5,50),cm(3)
      real mad
      character*10 s
c
      open(10,file='test.txt',status='old')
      open(20,file='result.txt')
c
      pi=2.0*asin(1.0)
      rad=pi/180
c nmax is the maximum number of data points - increase if necessary
      ita=0
      nmax=50
c
      write(6,*) 'How many samples ?'
      read(5,*) ismp
      write(6,*) 'How many steps for demagnetization ?'
      read(5,*) kstp

      mmax=ismp*kstp

      if(kstp.gt.nmax) then
        write(6,*) 'stop!'
        go to 999
      end if
c
      do 10 l1=1,mmax, kstp
      kkk=int(l1/kstp)+1
      write(6,*) kkk, ' PCA from [beg] to [end] steps '
      write(6,*) ' Input ibeg and iend ex.5 8'
      read(5,*) istart, iend
c             s=sample name
c             dat: demag of nrm, n=number of demag data
c
c       read in data and put in a dat array with dat(1,*)=tr
c       dat(2,*)=int, dat(3,*)=dec, dat(4,*)=inc
```

```
c
      do 20 k=1,kstp
      read(10,*, end=99) s, dat1, dat2, dat3, dat4
        dat(1,k)=dat1
        dat(2,k)=dat2
        dat(3,k)=dat3
        dat(4,k)=dat4
   20 continue
   99 continue
        icnt=iend-istart+1
         call dopca(dat,istart,iend,p,t,cm,mad)
        write(6,100)s,icnt,dat(1,istart),dat(1,iend),mad,p/rad,90-t/rad
      write(20,100)s,icnt,dat(1,istart),dat(1,iend),mad,p/rad,90-t/rad
  100   format(a,i3,2(f7.2,1x),3(f6.1,1x))
   10  continue
      close(10)
      close(20)
  999  stop
        end
c_____________________________________
        subroutine dopca(dat,istart,iend,pm,tm,cm,mad)
        dimension dat(5,*),cm(3),x(3,50)
        double precision d(3),e(3),t(3,3)
        real mad
        pi=2.0*asin(1.0)
        rad=pi/180
c
c       calculate center of mass
c
        do 10 i=1,3
         cm(i)=0
  10    continue
        n=iend-istart+1
        icnt=0
        do 200 i=istart,iend
        icnt=icnt+1
```

```
            theta=(90-dat(4,i))*rad
            p=dat(3,i)*rad
            r=dat(2,i)
            call dotpr_xyz(theta,p,r,x(1,icnt),x(2,icnt),x(3,icnt))
            do 20 j=1,3
              cm(j)=cm(j)+x(j,icnt)
 20         continue
 200      continue
c
c         now transfer origin to cm
c
          do 50 i=1,icnt
            do 40 j=1,3
              x(j,i)=x(j,i)-cm(j)/float(n)
 40       continue
 50       continue
c
c         put into a t matrix
c
          call tmatrix(icnt,x,t)
c
c         calculate eigenparameters
c
          call ql(3,3,t,d,e,ierr)
c
c         put into pd and pi, calculate MAD
c
          s1=sqrt(d(3))
          v2=d(2)
          v3=d(1)
          mad=atan(sqrt(v2+v3)/s1)/rad
c
c         first check if right direction
c
          a=acos(t(1,3)*x(1,1)+t(2,3)*x(2,1)+t(3,3)*x(3,1))
          if(a.gt.(pi/2))then
```

```
      do 77 j=1,3
       t(j,3)=-t(j,3)
77    continue
      endif
      x1=t(1,3)
      y=t(2,3)
      z=t(3,3)
      call doxyz_tpr(x1,y,z,tm,pm,r)
      return
      end
c_____________________________________
      subroutine dotpr_xyz(t,p,r,x,y,z)
c
c     calls no other routines
c       takes phi, theta, (in radians) and r, converts to x,y,z
c
      x=r*sin(t)*cos(p)
      y=r*sin(t)*sin(p)
      z=r*cos(t)
      return
      end
c_____________________________________________________
      subroutine tmatrix(n,x,t)
      dimension x(3,*)
      double precision t(3,3)
c
c     initialize t matrix
c
      do 10 i=1,3
      do 10 j=1,3
       t(i,j)=0
10    continue
c
c     do sums of squares and products
c
      do 20 i=1,n
```

```
            do 20 j=1,3
              do 20 k=1,3
                t(j,k)=t(j,k)+x(j,i)*x(k,i)
 20         continue
            return
            end
c
        subroutine ql(nm, n, a, d, e, ierr)
          implicit double precision (a-h,o-z)
c$$$$ calls tred2, tql2
c   using   eispack   routines tred2, tql2,   solves the symmetric
c   eigenvalue-eigenvector problem for a real matrix.
c   on input
c   nm     row dimension of the symmetric array   a   in the caller.
c   n      order of the array (<= nm)
c   a      the real symmetric array to be treated
c   e       a working array at least   n   long
c
c   on output
c   d      the array of eigenvalues ascending
c   a      the corresponding array of eigenvectors, with row
c          dimension   nm.   original   a   is overwritten.
c   ierr 0   if all's well.
c          j   if eigenvalue number j and above not found.
c
c
        dimension a(nm,*), d(*), e(*)
        call tred2(nm, n, a, d, e, a)
c
        call tql2(nm, n, d, e, a, ierr)
        return
        end
c_____________________________________________________________________
        subroutine tred2(nm, n, a, d, e, z)
          implicit double precision (a-h,o-z)
c$$$$ calls no other routines
```

```
c
      dimension 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 b. s. garbow,
```

```
c       applied mathematics division, argonne national laboratory
c       ------------------------------------------------------------------
c
        do 100 i = 1, n
c
            do 100 j = 1, i
                z(i,j) = a(i,j)
  100   continue
c
        if (n .eq. 1) go to 320
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 + abs(z(i,k))
c
            if (scale .ne. 0) go to 140
  130       e(i) = z(i,l)
            go to 290
c
  140       do 150 k = 1, l
                z(i,k) = z(i,k) / scale
                h = h + z(i,k) * z(i,k)
  150       continue
c
            f = z(i,l)
           if(h.lt.0)then
           write(14,*)'problem in tred2'
           endif
            g = -sign(sqrt(h),f)
            e(i) = scale * g
```

```
            h = h - f * g
            z(i,l) = f - g
            f = 0.0d0
c
            do 240 j = 1, l
                z(j,i) = z(i,j) / h
                g = 0.0d0
c       :::::::::: form element of a*u ::::::::::
                do 180 k = 1, j
  180            g = g + z(j,k) * z(i,k)
c
                jp1 = j + 1
                if (l .lt. jp1) go to 220
c
                do 200 k = jp1, l
  200            g = g + z(k,j) * z(i,k)
c       :::::::::: form element of p ::::::::::
  220            e(j) = g / h
                f = f + e(j) * z(i,j)
  240       continue
c
            hh = f / (h + h)
c       :::::::::: form reduced a ::::::::::
            do 260 j = 1, l
                f = z(i,j)
                g = e(j) - hh * f
                e(j) = g
c
                do 260 k = 1, j
                    z(j,k) = z(j,k) - f * e(k) - g * z(i,k)
  260       continue
c
  290       d(i) = h
  300 continue
c
  320 d(1) = 0.0d0
```

```
      e(1) = 0.0d0
c     :::::::::: accumulation of transformation matrices ::::::::::
      do 500 i = 1, n
         l = i - 1
         if (d(i) .eq. 0) go to 380
c
         do 360 j = 1, l
            g = 0.0d0
c
            do 340 k = 1, l
  340        g = g + z(i,k) * z(k,j)
c
            do 360 k = 1, l
               z(k,j) = z(k,j) - g * z(k,i)
  360       continue
c
  380       d(i) = z(i,i)
            z(i,i) = 1.0d0
            if (l .lt. 1) go to 500
c
            do 400 j = 1, l
               z(i,j) = 0.0d0
               z(j,i) = 0.0d0
  400       continue
c
  500 continue
c
      return
      end
c_____________________________________________________________________
      subroutine tql2(nm, n, d, e, z, ierr)
        implicit double precision (a-h,o-z)
c$$$$ calls no other routines
c
      dimension d(n),e(n),z(nm,n)
      double precision b,c,f,g,h,p,r,s,machep
```

```
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         ------------------------------------------------------------------
c
c         :::::::::: machep is a machine dependent parameter specifying
c                     the relative precision of floating point arithmetic.
c                     machep = 16.0d0**(-13) for long form arithmetic
c                     on s360 ::::::::::
          data machep/1.421d-14/
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
          b = 0.0d0
          e(n) = 0.0d0
c
          do 240 l = 1, n
             j = 0
             h = machep * (abs(d(l)) + abs(e(l)))
             if (b .lt. h) b = h
c         :::::::::: look for small sub-diagonal element ::::::::::
             do 110 m = l, n
```

```
                if (abs(e(m)) .le. b) 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
            g = d(l)
            p = (d(l1) - g) / (2.0d0 * e(l))
            r = sqrt(p*p+1.0d0)
            d(l) = e(l) / (p + sign(r,p))
            h = g - d(l)
c
            do 140 i = l1, n
  140       d(i) = d(i) - h
c
            f = f + h
c       :::::::::: ql transformation ::::::::::
            p = d(m)
            c = 1.0d0
            s = 0.0d0
            mml = m - l
c       :::::::::: for i=m-1 step -1 until l do -- ::::::::::
            do 200 ii = 1, mml
               i = m - ii
               g = c * e(i)
               h = c * p
               if (abs(p) .lt. abs(e(i))) go to 150
               c = e(i) / p
               r = sqrt(c*c+1.0d0)
               e(i+1) = s * p * r
               s = c / r
               c = 1.0d0 / r
```

```
                go to 160
  150           c = p / e(i)
                r = sqrt(c*c+1.0d0)
                e(i+1) = s * e(i) * r
                s = 1.0d0 / r
                c = c * s
  160           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
            e(l) = s * p
            d(l) = c * p
            if (abs(e(l)) .gt. b) 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
           write(6,*)'No convergence after 30 iterations'
 1001 return
        end
c_____________________________________________________________
            subroutine doxyz_tpr(x,y,z,t,p,r)
c
c           calls no other routines.
c             takes x,y,z components and returns theta (t) and phi (p)
c             in radians
c
            pi=2.0*asin(1.0)
            r=sqrt(x*x+y*y+z*z)
            t=acos(z/r)
            if (x.eq.0) then
            if (y.lt.0) then
                        p= 3*pi/2
                         else
                         p= pi/2
            endif
            return
            endif
            p = (atan(y/x))
```

```
      if (x.lt.0) then
      p = p + pi
      endif
c     p= (atan2(y,x))
      if (p.lt.0) then
        p = p+2*pi
      endif
      return
      end
```