```
C     RADIX-2 FFT  -  PLUS PEAK LISTING SUBROUTINES
CC    COMPRESSES HIGH RESOLUTION DATA INTO +/- MPAR/2 RANGE
      parameter (NPAR=256,NPA=NPAR-1,NP=100)
      byte nt(24,9),nh(-NP:NP,-NP:NP),HH,HK,HL
      real*8 arg,pi2,px
      integer*2 ih,ik,il,jh,jk,jl,ns,ne,nph,nm(0:NPA)
      dimension r(-1:NPAR,-1:NPAR,3),t(4,4),tx(24,3)
      complex f(0:NP,-NP:NP,-NP:NP),cx(0:NPA),fx(0:NPA),fhkl,cxl,fs
      complex p(0:NPA,0:NPA)
      COMMON /WORK/ fx,cx,n1,NX
      COMMON /PKS/ r,t,peak,MPAR,MPA,MPAX,IHORZ,IVERT,ISECT,NATOM
      data npeaks,num,mh,mk,ml,mmk,mml,rlast,peak/5*0,2*99,2*0.0/
      data t/18.,4*-10.,11.,2*4.,-10.,4.,11.,4.,-10.,2*4.,11./
      write(*,40)
   40 format(' Type 1 for difference map ',$)
      read(*,*)ntype
      t0=secnds(0.0)
      do 19 i=1,4
      do 19 j=1,4
   19 t(i,j)=t(i,j)/42.
      px=-1.0
      open(unit=1,file='fft.dat',status='old',form='formatted')
      READ(1,*)MPAR,MPAX,IHORZ,IVERT,ISECT,NATOM,LIST,NPKS
      if(LIST.ne.0) write(2)MPAR,MPAX
      READ(1,*)nsym,VOL
      do 16 i=1,nsym
      READ(1,17)tx(i,1),HH,HK,HL,tx(i,2),KH,KK,KL,tx(i,3),LH,LK,LL
   17 format(3(f10.4,3i5))
      IDET = HH*KK*LL+HK*KL*LH+KH*LK*HL-LH*KK*HL-KH*HK*LL-HH*LK*KL
      nt(i,1) = (KK*LL-LK*KL)/IDET
      nt(i,2) =-(KH*LL-LH*KL)/IDET
      nt(i,3) = (KH*LK-KK*LH)/IDET
      nt(i,4) =-(HK*LL-HL*LK)/IDET
      nt(i,5) = (HH*LL-LH*HL)/IDET
      nt(i,6) =-(HH*LK-LH*HK)/IDET
      nt(i,7) = (HK*KL-KK*HL)/IDET
      nt(i,8) =-(HH*KL-HL*KH)/IDET
      nt(i,9) = (HH*KK-HK*KH)/IDET
   16 continue
      close(unit=1)
      MPA=MPAR-1
      MP2=MPAR+MPA
      pi2=2.0*dacos(px)
      pie2=pi2
      px=pi2/MPAR
C     initialize exp(-twopi*x/MPAR), remember negative sign
      do 20 i=0,MPA
      arg=-i*px
   20 cx(i)=dcmplx(dcos(arg),dsin(arg))
C     zero data arrays
      do 21 il=-NP,NP
      do 21 ik=-NP,NP
      nh(ik,il) = 0
      do 21 ih=  0,NP
   21 f(ih,ik,il)=0.
C     set up FFT arrays for bit reversal on first pass
      n1=MPAR-1
      nm(0)=0
      NX = alog(float(MPAR))/alog(2.0) - 0.9
      mm=1
```

```
      do 23 i=1,NX+1
      do 24 j=0,mm-1
      nm(j)=2*nm(j)
   24 nm(j+mm) = nm(j)+1
      mm=mm+mm
   23 continue
C     read reflection data
      open(unit=1,file='fourr.tmp',status='old',form='unformatted')
      do 26 i=1,999999
      read(1,end=27)ih,ik,il,ns,ne,ff,ph,fcal,nph
      if(ntype.ne.0) ff = ff-fcal
      num=num+1
      do 28 j=1,nsym
      jh = ih*nt(j,1) + ik*nt(j,2) + il*nt(j,3)
      jk = ih*nt(j,4) + ik*nt(j,5) + il*nt(j,6)
      jl = ih*nt(j,7) + ik*nt(j,8) + il*nt(j,9)
      sh = ih*tx(j,1) + ik*tx(j,2) + il*tx(j,3)
      sh = mod(sh,1.0)
      ph=0.001*nph + sh*pi2
      ph=mod(ph,pie2)
      call SWAP(jh,jk,jl,IHORZ,IVERT,ISECT)
      if(jh) 32,30,29
   30 if(jk) 32,31,29
   31 if(jl) 32,29,29
   32 jh=-jh
      jk=-jk
      jl=-jl
      ph=-ph
   29 mh=max(jh,mh)
      mk=max(jk,mk)
      ml=max(jl,ml)
      mmk=min(jk,mmk)
      mml=min(jl,mml)
      nh(jk,jl)=1
   28 f(jh,jk,jl) = 2.*ff*cmplx(cos(ph),sin(ph))/VOL
   26 continue
   27 close(unit=1)
      t1=secnds(t0)
C     calculate FOURIER
      do 1 ix=-1,MPAX+1
      ixx = ix+1
      ixx = 1+mod(ixx,3)
      do 3 ik = mmk,mk
      do 4 iz=0,MPA
    4 fx(iz)=0.
      do 5 il = mml,ml
      if(nh(ik,il).eq.0) goto 5
      jl=nm(il.and.MPA)
      fs=0.
      do 2 ih=0,mh
      fhkl=f(ih,ik,il)
      if(fhkl.eq.0.0) goto 2
      cxl = cx((ih*ix).and.MPA)
      fs=fs +fhkl*cxl
    2 continue
CC    SUM the (IL+n*MPAR) components into the IL subspace
      fx(jl)=fx(jl)+fs
C     fx(jl)=fs
    5 continue
      call FFT(MPAR)
```

```
      jk=ik.and.MPA
      do 6 iz=0,MPA
CC    SUM the(IK+n*MPAR) results into the IK subspace
    6 p(jk,iz) = p(jk,iz) +fx(iz)
C   6 p(jk,iz) = fx(iz)
    3 continue
      do 7 iz=0,MPA
      do 8 ik=0,MPA
    8 fx(ik)=0.
CC    Scan subspace of p(jk,iz)
      do 9 ik = 0,MPA
C     do 9 ik = mmk,mk
      jk=ik.and.MPA
      kj=nm(jk)
      fx(kj)=p(jk,iz)
CC    Zero P(jk,iz) array before calculating next section on x
    9 p(jk,iz)=0
C   9 continue
      call FFT(MPAR)
      do 10 iy=0,MPA
   10 r(iy,iz,ixx)=fx(iy)
    7 continue
      if(NPKS.ne.0) goto 1
      if(ix.eq.(MPAX+1)) mid=-mid
      if(ix.ge.1) call pkpick(ix-1,mid,npeaks,rlast)
      mid=ixx
      if(LIST.ne.0) write(2)((r(iy,iz,ixx),iy=0,MPA),iz=0,MPA)
    1 continue
      if(npeaks.lt.NATOM) write(*,41) npeaks
   41 format(' only ',i3,' peaks noted in map ')
      t2= (secnds(t0) -t1)
      write(*,*)num,t1,t2
      end
C
      SUBROUTINE FFT(N)
      parameter (NPAR=256,NPA=NPAR-1)
      complex f(0:NPA),cx(0:NPA),a,b
      COMMON /WORK/ f,cx,n1,NX
      do 1 i1 = 0,n1,2
      a= f(i1+1)
      f(i1+1)=f(i1)-a
    1 f(i1)=f(i1)+a
      i1=2
      m2=N/4
      do 2 j0 = 1,NX
      i0=i1
      i2=i1-1
      i1=i1+i1
      do 3 i=0,n1,i1
      a=f(i+i0)
      f(i+i0)=f(i)-a
      f(i)=f(i)+a
      i4=0
      do 4 k=i+1,i+i2
      i4=i4+m2
      a = f(k+i0)*cx(i4)
      f(k+i0)=f(k)-a
    4 f(k)=f(k)+a
    3 continue
      m2=m2/2
```

```
    2 continue
      return
      end
C
      SUBROUTINE PKPICK(ixm1,MID,npeaks,rlast)
      parameter (NPAR=256,NPA=NPAR-1,NP=100)
      real*8 a,b,c,d,e,f,g,ah,ak,al,a1,a2,a3,q1,q2,q3,q4,det
      real*8 t11,t12,t13,t22,t23,t33
      dimension r(-1:NPAR,-1:NPAR,3),t(4,4),r27(27)
      dimension px(2000,4),rh(2000),ax(3),nh(27),nk(3)
      COMMON /PKS/ r,t,peak,MPAR,MPA,MPAX,IHORZ,IVERT,ISECT,NATOM
      COMMON /SRT/ px,rh,rholast
      data good,bad/'   G','   B'/
      data nh/0,1,0,1,1,1,0,1,0,4*1,0,4*1,0,1,0,1,1,1,0,1,0/
      nstop=mid/iabs(mid)
      mid=iabs(mid)
      ntop=mid+1
      if(ntop.eq.4) ntop=1
      nbot=6-mid-ntop
C     fill out edges of the map
      do 1 ix=1,3
      do 2 iz=0,MPA
      r(-1,iz,ix) = r(MPA,iz,ix)
    2 r(MPAR,iz,ix) = r(0,iz,ix)
      do 3 iy=0,MPA
      r(iy,-1,ix) = r(iy,MPA,ix)
    3 r(iy,MPAR,ix) = r(iy,0,ix)
      r(-1,-1,ix) = r(MPA,MPA,ix)
      r(-1,MPAR,ix) = r(MPA,0,ix)
      r(MPAR,-1,ix) = r(0,MPA,ix)
      r(MPAR,MPAR,ix) = r(0,0,ix)
    1 continue
      if(npeaks.gt.10) goto 12
      pick=0.
      do 4 iz=0,MPA
      do 4 iy=0,MPA
      xx = r(iy,iz,mid)
    4 if(xx.gt.pick) pick=xx
      pick = 0.15*pick
      if(pick.gt.peak) peak=pick
   12 N = npeaks
      rholast=rlast
      if(N.eq.0)open(1,file='FRAG',status='unknown',form='formatted'
)
      PAR = 1.0/FLOAT(MPAR)
C     find peaks surrounded by 19 positive points
      ix=mid
      ixm=nbot
      ixp=ntop
      nk(1)=nbot
      nk(2)=mid
      nk(3)=ntop
      do 5 iz=0,MPA
      izm=iz-1
      izp=iz+1
      do 6 iy=0,MPA
      iym=iy-1
      iyp=iy+1
      xx=r(iy,iz,ix)
      zz = 0.00001
```

```
      if(xx.lt.peak) goto 6
C     IS THE PEAK A LOCAL MAXIMUM?
      i4=0
      do 50 i3=1,3
      imid=nk(i3)
      do 50 i2=-1,1
      do 50 i1=-1,1
      i4=i4+1
      r27(i4)=r(iy+i1,iz+i2,imid)
   50 if(nh(i4).eq.1.and.xx.lt.r27(i4)) goto 6
C     are the surrounding points all positive densities
      do 51 i1=1,27
   51 if(nh(i1).eq.1.and.zz.ge.r27(i1)) goto 7
C     perform Rollet's ellipsoidal fit procedure
      oqq = log(r27(10))
      qqo = log(r27(4))
      oqo = log(r27(13))
      pqo = log(r27(22))
      oqp = log(r27(16))
      qoq = log(r27(2))
      ooq = log(r27(11))
      poq = log(r27(20))
      qoo = log(r27(5))
      ooo = log(xx)
      poo = log(r27(23))
      qop = log(r27(8))
      oop = log(r27(17))
      pop = log(r27(26))
      opq = log(r27(12))
      qpo = log(r27(6))
      opo = log(r27(15))
      ppo = log(r27(24))
      opp = log(r27(18))
      b = (poo+ppo+pqo+pop+poq -qoo-qqo-qpo-qoq-qop)/10.
      c = (opo+ppo+qpo+opp+opq -oqo-qqo-pqo-oqq-oqp)/10.
      d = (oop+opp+oqp+pop+qop -ooq-oqq-opq-qoq-poq)/10.
      ah = (opp+oqq-opq-oqp)/4.
      ak = (pop+qoq-poq-qop)/4.
      al = (ppo+qqo-pqo-qpo)/4.
      a1 = ppo+pqo+qpo+qqo
      a2 = pop+poq+qop+qoq
      a3 = opp+opq+oqp+oqq
      q1 = ooo+poo+qoo+opo+oqo+oop+ooq+a1+a2+a3
      q2 = poo+qoo+a1+a2
      q3 = opo+oqo+a1+a3
      q4 = oop+ooq+a2+a3
      a = t(1,1)*q1 + t(1,2)*q2 + t(1,3)*q3 + t(1,4)*q4
      e = t(2,1)*q1 + t(2,2)*q2 + t(2,3)*q3 + t(2,4)*q4
      f = t(3,1)*q1 + t(3,2)*q2 + t(3,3)*q3 + t(3,4)*q4
      g = t(4,1)*q1 + t(4,2)*q2 + t(4,3)*q3 + t(4,4)*q4
      det = -8*e*f*g -2*ah*ak*al + 2*f*ak*ak +2*ah*ah*e +2*g*al*al
      if(dabs(det).lt.0.0001) goto 7
      t11 = (4*f*g-ah*ah)/det
      t12 = (ah*ak-2*g*al)/det
      t13 = (ah*al-2*f*ak)/det
      t22 = (4*e*g-ak*ak)/det
      t23 = (ak*al-2*e*ah)/det
      t33 = (4*e*f-al*al)/det
      x = t11*b + t12*c + t13*d
      y = t12*b + t22*c + t23*d
```

```
      z = t13*b + t23*c + t33*d
      ax(1) = PAR*(float(ixm1)+x)
      if(ax(1).lt.-0.0001.or.ax(1).gt.1.0001) goto 6
      N=N+1
      if(N.gt.NATOM) N=NATOM+1
      rh(N)
=dexp(a+b*x+c*y+d*z+e*x*x+f*y*y+g*z*z+ah*z*y+ak*x*z+al*x*y)
      ax(2) = PAR*(float(iy)+y)
      ax(3) = PAR*(float(iz)+z)
      char = good
      goto 8
    7 N=N+1
      if(N.gt.NATOM) N=NATOM+1
      ax(1) = PAR*float(ixm1)
      ax(2) = PAR*float(iy)
      ax(3) = PAR*float(iz)
      rh(N) = xx
      char = bad
    8 px(N,1)=ax(IHORZ)
      px(N,2)=ax(IVERT)
      px(N,3)=ax(ISECT)
      px(N,4)=char
      if(N.eq.NATOM) call SORT(N)
      if(N.eq.(NATOM+1).and.rh(N).gt.rholast) call INSERT(NATOM)
      npeaks=N
      rlast=rholast
      peak=0.85*rholast
    6 continue
    5 continue
      if(nstop.eq.1) return
      if(N.lt.NATOM.and.N.gt.0) call SORT(N)
      if(N.gt.NATOM) N = NATOM
      do 9 i=1,N
    9 write(1,10) (px(i,j),j=1,3),rh(i),5.0,i,px(i,4)
   10 format(3f9.5,4x,2f8.2,i5,a4)
      return
      end
C
      subroutine SORT(N)
C     sort the peaks in descending order of magnitude
      integer*2 num(2000)
      dimension px(2000,4),rh(2000),q(2000,4)
      COMMON /SRT/ px,rh,rholast
      do 1 i=1,N
    1 num(i)=i
      do 2 i=2,N
      j=i
      hold = rh(1)
      rh(1) = rh(j)
      do while (rh(j) .gt. rh(j-1))
         x=rh(j)
         rh(j)=rh(j-1)
         rh(j-1)=x
         ix=num(j)
         num(j)=num(j-1)
         num(j-1)=ix
         j = j-1
      end do
      rh(1) = hold
      if (rh(2) .gt. rh(1)) then
```

```
            x=rh(2)
            rh(2)=rh(1)
            rh(1)=x
            ix=num(2)
            num(2)=num(1)
            num(1)=ix
          end if
        2 continue
          do 3 i=1,N
          k=num(i)
          q(i,1)=px(k,1)
          q(i,2)=px(k,2)
          q(i,3)=px(k,3)
        3 q(i,4)=px(k,4)
          do 4 i=1,N
          px(i,1)=q(i,1)
          px(i,2)=q(i,2)
          px(i,3)=q(i,3)
        4 px(i,4)=q(i,4)
          rholast = rh(N)
          return
          end
C
          subroutine INSERT(NATOM)
C         place newest peak within the sorted stack
          dimension px(2000,4),rh(2000)
          COMMON /SRT/ px,rh,rholast
          NATOM1=NATOM+1
          rho = rh(NATOM1)
          do 1 i=1,NATOM-1
          j=NATOM-i
          j1=j+1
          if(rh(j).gt.rho)  goto 2
          rh(j1)=rh(j)
          do 4 k=1,4
        4 px(j1,k)=px(j,k)
        1 continue
          rh(1)=rho
          do 5 i=1,4
        5 px(1,i)=px(NATOM1,i)
          goto 6
        2 rh(j1)=rh(NATOM1)
          do 3 j=1,4
        3 px(j1,j)=px(NATOM1,j)
        6 rholast = rh(NATOM)
          return
          end
C
          SUBROUTINE SWAP(jh,jk,jl,IHORZ,IVERT,ISECT)
          integer*2 jh,jk,jl,nv(3)
          nv(IHORZ) = jh
          nv(IVERT) = jk
          nv(ISECT) = jl
          jh = nv(1)
          jk = nv(2)
          jl = nv(3)
          return
          end
```

```
      PARAMETER NPAR=512,NPA=NPAR-1
C     1-D TEST TO COMPRESS HIGH RESOLUTION DATA INTO +/- NDIV/2
RANGE
      DIMENSION R(0:NPA),RB(0:NPA)
      COMPLEX DATA(0:NPA),cx(0:NPA),f(0:NPA),dx
      real*8 x,y,rand
      COMMON /WORK/ f,data,cx
C
C     ISEED is a random integer to generate COMPLEX-VALUED F(H) DATA
C     NDIV = 2**N defines the usual minimal grid size computing the
FFT
C     NDIV2 = 2**M is smaller than NDIV and for which a normal FFT
can
C     not be used to calculate the transform of the generated data.
C
      write(*,8)
    8 FORMAT(' FIRST TRY USING ISEED=123, NDIV=64, NDIV2=16 ')
      write(*,5)
    5 FORMAT(' RANDOM INTEGER TO GENERATE F(H) VALUES, ISEED = ',$)
      read(*,*)ISEED
      write(*,4)
    4 FORMAT('{ # GRID DIVISIONS (LE 512) NDIV = 2**N } NDIV = ',$)
      read(*,*)NDIV
      MDIV=NDIV-1
      write(*,7)
    7 FORMAT('{ # GRID DIVISION MPAR2.LT.MPAR = 2**M } MPAR2= : ',$)
      read(*,*)NDIV2
      MDIV2=NDIV2-1
      call srand(iseed)
      PI2=2.0*acos(-1.0)
C     Set up dummy set of complex-valued data for some MPA
C     Note the data define a complex valued transform since
F(h).NE.F(-h)*
      do 9 i=0,MDIV
      x=50.*rand()
      y=PI2*rand()
    9 DATA(i) = x*cmplx(cos(y),sin(y))
C     Set up EXP table for the NDIV grid
      p=PI2/NDIV
      do 10 i=0,MDIV
      a= -i*p
   10 cx(i)=cmplx(cos(a),sin(a))
      write(*,3)
    3 format(/)
C
C     BRUTE FORCE calculation over NDIV**2 points
C
      do 17 i=0,MDIV
      dx=0.
      do 18 j=0,MDIV
      ij=(i*j).and.MDIV
   18 dx=dx+DATA(j)*cx(ij)
      r(i)= dx
   17 rb(i)=imag(dx)
C     PRINT OUT REAL & IMAGINARY PARTS OF THE TRANSFORM
      WRITE(*,20)
   20 FORMAT(' BRUTE FORCE FOURIER CALCULATION')
      WRITE(*,1)  ( R(i),i=0,MDIV)
      WRITE(*,2)  (RB(i),i=0,MDIV)
    1 FORMAT('REAL',8F8.2)
```

```
    2 FORMAT('IMAG',8f8.2)
      write(*,3)
C
C     FFT for same calculation for MPAR data and MPAR grid points
C
      CALL FFT(NDIV)
      do 6 iz=0,MDIV
      rb(iz)=imag(f(iz))
    6 r(iz) = f(iz)
C     FFT results agree with brute force calculation on an MPAR grid
      WRITE(*,21)
   21 FORMAT(' PROPERLY DIMENSIONED FFT GIVES SAME RESULT')
      WRITE(*,1)  ( R(i),i=0,MDIV)
      WRITE(*,2)  (RB(i),i=0,MDIV)
      write(*,3)
C
C     reprint old FFT results on the smaller NDIV2 grid
      IT=NDIV/NDIV2
      WRITE(*,22)
   22 FORMAT(' SUB LATTICE OF PREVIOUS MAP')
      WRITE(*,1)  ( R(i),i=0,MDIV,IT)
      WRITE(*,2)  (RB(i),i=0,MDIV,IT)
      write(*,3)
C
C     Set up table of EXP for NDIV2 point grid
      p=PI2/NDIV2
      do 13 i=0,MDIV2
      a= -i*p
   13 cx(i)=cmplx(cos(a),sin(a))
C
C     Try computing the FFT for NDIV terms on a smaller NDIV2 grid
C
      CALL FFT(NDIV2)
      do 14 iz=0,MDIV2
      rb(iz)=imag(f(iz))
   14 r(iz) = f(iz)
C     These results should be clobbered
      WRITE(*,23)
   23 FORMAT(' WHEN THE # OF DATA > # GRID POINTS, GARBAGE RESULTS')
      WRITE(*,1)  ( R(i),i=0,MDIV2)
      WRITE(*,2)  (RB(i),i=0,MDIV2)
      write(*,3)
C
C     To get correct result Sum overtones of the DATA into the inner
shell
C
      do 11 i=0,MDIV
      ii=i.and.MDIV2
   11 if(ii.ne.i) DATA(ii)=DATA(ii) +DATA(i)
C     Then call FFT
      CALL FFT(NDIV2)
      do 12 iz=0,MDIV2
      rb(iz)=imag(f(iz))
   12 r(iz) = f(iz)
      if(MDIV2.eq.0)  r(0) = DATA(0)
      if(MDIV2.eq.0) rb(0) =IMAG(DATA(0))
C     FFT for NDIV data computed on an NDIV2 grid
      WRITE(*,24)
   24 FORMAT(' CORRECT FORMULATION OF FFT FOR REDUCED GRID SPACE')
      WRITE(*,1)  ( R(i),i=0,MDIV2)
```

```
      WRITE(*,2)  (RB(i),i=0,MDIV2)
      END
C
      SUBROUTINE FFT(MPAR)
      PARAMETER NPAR=512,NPA=NPAR-1
      complex f(0:NPA),cx(0:NPA),data(0:NPA),a,b
      dimension nm(0:NPA)
      COMMON /WORK/ f,data,cx
C     set up array for bit-inversion
      nm(0)=0
      mm=1
      NX = alog(float(MPAR))/alog(2.0) - 0.9
      do 6 i=1,NX+1
      do 5 j=0,mm-1
      nm(j)=2*nm(j)
    5 nm(j+mm) = nm(j)+1
    6 mm=mm+mm
      MPA=MPAR-1
C     perform the swap to restore data(I) in f(J)
      do 7 i=0,MPA
      j = nm(i)
    7 f(j)=data(i)
      do 1 i = 0,MPA,2
      a= f(i)
      b= f(i+1)
      f(i)   = a + b
    1 f(i+1) = a - b
      i1=2
      m2=(MPAR)/4
      do 2 i = 1,NX
      i0=i1
      i2=i1-1
      i1=i1+i1
      do 3 j=0,MPA,i1
      i4=-m2
      do 4 k=j,j+i2
      i4=i4+m2
      a = f(k)
      kk=k+i0
      b = f(kk)*cx(i4)
      f(k)=a+b
    4 f(kk)=a-b
    3 continue
      m2=m2/2
    2 continue
      return
      end
```