

```

c+++++
+
c boxer package
+++++
c+++++
+
c*****
*
c*SUB EPCNTR
*****
c*****
*
      subroutine epcntr(n,xla,xlo,fint,ival,xlaep,xloep,xlaerd,xloerd,
1xlaer,xloer,fin0,finm,npe,xlat,xlon,iout,locali,ifileout4)
c
c      Compute macroseismic epicenter as the baycentre of highest
intensities
c      and epicentral intensity as the largest intensity observed in two
c      localities at least (see Gasperini et al., 1999)
c
c      Input:
c
c      n: number of intensity data points (IDPs)
c      xla: latitudes of IDPs (vector)
c      xlo: longitudes of IDPs (vector)
c      fint: intensity of IDPs (vector)
c      ival: 3 (minimum number of IDPs to compute epicenter)
c      iout: printout information level
c      locali: name of locality (character vector)
c      ifileout4: Fortran unit number of output file
c
c      Output:
c
c      xlaep: latitude of epicenter
c      xloep: longitude of epicenter
c      xlaerd: latitude uncertainty (in degrees)
c      xloerd: longitude uncertainty (in degrees)
c      xlaer: latitude uncertainty (in km)
c      xloer: longitude uncertainty (in km)
c      fin0: epicentral intensity
c      finm: maximum intensity
c      npe: number of IDPs used to compute the epicenter
c      xlat: latitudes used to compute the epicenter (vector)
c      xlon: longitudes used to compute the epicenter (vector)
c
c      Subroutine and functions:
c
c      provided: maxv, trimn, tristd1, sort
c
c
c      implicit doubleprecision (a-h,o-z)
c      parameter(npts=3000)
c      dimension xla(*),xlo(*),fint(*),xlat(npts),xlon(npts)
c      dimension work(npts),fin(npts)
c      doubleprecision maxv
c      character*20 locali(*),loca(npts)
c      data dtor/ 1.7453292519943296d-2/,gradla/111.195d0/
c      finmax=maxv(n,fint)
c      if(dmod(finmax,1.d0).eq.0.d0)then

```

```

c      integer imax
c
c          finmin=finmax
else
c
c      half integer imax
c
c          finmin=finmax-.5d0
endif
c
c      imax
c
c      m=0
do i=1,n
    if(fint(i).le.finmax.and.fint(i).ge.finmin)then
        m=m+1
        xlat(m)=xla(i)
        xlon(m)=xlo(i)
        fin(m)=fint(i)
        loca(m)=locali(i)
    endif
enddo
mm=m
if(m.lt.ival)then
c
c      imax-1
c
c          m1=m
do i=1,n
    if(fint(i).lt.finmin.and.fint(i).ge.finmin-1.d0)then
        m1=m1+1
        xlat(m1)=xla(i)
        xlon(m1)=xlo(i)
        fin(m1)=fint(i)
        loca(m1)=locali(i)
    endif
enddo
mm=m1
endif
if(iout.ge.2)then
    write(ifileout4,'(/,1x,a,/)' )'Used data to compute epicenter:'
    write(ifileout4,'(1x,a)' )'      Lat      Lon      I      locality'
do i=1,mm
    write(ifileout4,'(1x,2f8.3,1x,f5.1,1x,a)' )xlat(i),xlon(i),
1      fin(i),loca(i)
enddo
    write(ifileout4,'(1x)' )
endif
xlaep=trimn(mm,xlat,work)
xlaerd=tristdl(mm,xlaep,xlat,work,ntr)
if(xlaerd.lt.998.d0)then
c      9/9/2009 P.G. standard error of the mean
        xlaerd=xlaerd/dsqrt(dble(ntr))
        xlaer=xlaerd*gradla
else
        xlaer=xlaerd
endif
xloep=trimn(mm,xlon,work)
xloerd=tristdl(mm,xloep,xlon,work,ntr)
c      9/9/2009 P.G. standard error of the mean

```

```

        if(xloerd.lt.998.d0)then
c      9/9/2009 P.G. standard error of the mean
          xloerd=xloerd/dsqrt(dble(ntr))
          xloer=xloerd*gradla*dcos(xlaep*dtor)
        else
          xloer=xloerd
        endif
        npe=mm
c
c      imax-2
c
c      m1=mm
      do i=1,n
        if(fint(i).lt.finmin-1.d0.and.fint(i).ge.finmin-2.d0)then
          m1=m1+1
          xlat(m1)=xla(i)
          xlon(m1)=xlo(i)
          fin(m1)=fint(i)
        endif
      enddo
      mmm=m1
      finm=maxv(mmm,fin)
      fin0=finm
      nmax=0
      fin0m=0.d0
      do i=1,mmm
        if(fin(i).eq.fin0)then
          nmax=nmax+1
        else
          fin0m=max(fin0m,fin(i))
        endif
      enddo
      if(n.ne.1)then
        if(nmax.le.1)then
          fin0=max(fin0m,fin0-1.d0)
        endif
      endif
      return
    end
c*****
*
c*SUB EPCSEA
*****
c*****
*
      subroutine epcsea(nincin,ninc,n,xla,xlo,fint,xlaep,xloep,
1xlaer,xloer,fin0,finm,npe,iout,locali,rms,
1gapmax,ddismin,nmlocb,fin0er,amg,bmg,aattcoe,battcoe,depref,
1xdepthout,xdeper,ncldis,var,ermax,ermed,ermin,azmax,azmed,azmin,
1plmax,plmed,plmin,ecc,ermax2d,azmax2d,ermin2d,azmin2d,ecc2d,
1lis,aatt,aatter,batt,batter,amm,ammer,ammtr,ammtrer,met,iselp,
1aai,bbi,cci,azi1,azi2,ranlat,ranlon,depmax,ifileout4)
c
c      compute macroseismic epicenter, depth, intensity at epicenter and
attenuation
c      coefficients using Pasolini et al. (2008b) attenuation function
c
c      Input:
c
c      nincin: location methods (1-6)

```

```

c      n: number of intensity data points (IDPs)
c      xla: latitudes of IDPs (vector)
c      xlo: longitudes of IDPs (vector)
c      fint: intensity of IDPs (vector)
c      nmlocb: minimum number of IDPs to compute epicenter (10)
c      fin0: epicentral intensity (computed by epcntr)
c      iout: printout information level
c      locali: names of localities (character vector)
c      amg: intercept of M-Io relationship (Pasolini et al., 2008b)
c      bmg: coefficient of M-Io relationship (Pasolini et al., 2008b)
c      aattcoe: coefficient a (linear term) of attenuation equation
(Pasolini et al., 2008b)
c      battcoe: coefficient b (log term) of attenuation equation (Pasolini
et al., 2008b)
c      depref: initial depth (km) (Pasolini et al., 2008b)
c      ncldis: 1 (number of classes for distance weighting)
c      met: 1 (optimization method)
c      iselp: 1 discard IDPs at long distances (Standard attenuation law)
c              2 include IDPs at long distances (Extended attenuation law)
c      aai,bbi,cci: coefficients of bilinear attenuation equation for data
selection
c      (0.53, 0.055, 0.022) (Pasolini et al., 2008a)
c      ifileout4: Fortran unit number of output file
c
c      Output:
c
c      ninc: number of free parameters
c      xlaep: latitude of epicenter
c      xloep: longitude of epicenter
c      xlaer: latitude uncertainty (in km)
c      xloer: longitude uncertainty (in km)
c      fin0: intensity at epicenter (Ie)
c      finm: maximum intensity
c      npe: number of IDPs used to compute the epicenter
c      rms: root mean square of intensity residuals
c      gapmax: maximum azimuthal gap (degrees) among IDPs
c      ddismin: minimum epicentral distance of IDPs (km)
c      fin0er: uncertainty of intensity at epicenter
c      xdepthout: computed depth (km)
c      xdeper: uncertainty of depth (km)
c      var: variance/covariance matrix (7x7 matrix)
c      ermax, ermed, ermin: amplitudes (km) of semiaxes of uncertainty
ellipsoid for hypocenter
c      azmax, azmed, azmin: azimuths (degrees) of semiaxes of uncertainty
ellipsoid for hypocenter
c      plmax, plmed, plmin: plunges (degrees) of uncertainty ellipsoid for
hypocenter
c      ecc: eccentricity of min/max cross-section of uncertainty ellipsoid
for hypocenter
c      ermax2d,ermin2d: amplitudes (km) of semiaxes of uncertainty ellipse
for epicenter
c      azmax2d,azmin2d: azimuths (degrees) of semiaxes of uncertainty
ellipse for epicenter
c      ecc2d: eccentricity of uncertainty ellipse for epicenter
c      is: location status (0: unsuccessful, 2: successful)
c      aatt: coefficient a (linear term) of attenuation equation
c      aatter: uncertainty of coefficient a (linear term) of attenuation
equation
c      batt: coefficient b (log term) of attenuation equation

```

```

c      batter: uncertainty of coefficient b (log term) of attenuation
equation
c
c      Subroutine and functions:
c
c      provided:
wgtdis,maxlik,errors,ellips,sortp,sort,trimn,tristd,tristdl,diex,diex_bil
i,
c
diex6,diexm6,matrixwr,likel_simi,likel_sim6,fepc_sim6,fepc_met6,fepc_simi
,
c      fepc_meti,azimul
c
c      not provided (available from IMSL Library)
c
c      dbconf (Minimize a function of N variables subject to simple bounds
using a quasi-Newton
c      method and finite-difference gradients)
c      du4inf (Utility routine to set parameters for dbconf)
c      dlinrg (Inverse of a general real matrix)
c      df2hes (Approximate the Hessian using forward differences and
function values)
c      devcsf (Compute all of the eigenvalues and eigenvectors of a real
symmetric matrix)
c
c      implicit doubleprecision (a-h,o-z)
parameter(nep=7)
parameter(npts=3000,lp=7)
common/datepc/xlat(npts),xlon(npts),di(npts),wgt(npts),npun
common/atten/aa,bb,cc,fintl,xdeptha,fint0
common/metlik/metc
dimension xla(*),xlo(*),fint(*),iperm(npts),resv(npts)
dimension work(npts),fin(npts),var(lp,lp)
dimension x(lp),g(lp),aziv(npts),ddisv(npts),ama(npts)
doubleprecision maxv
character *20 locali(*),cloc(npts),localimin
logical iexit
character *11 lab(lp),lab1(lp)*5
data lab/'          Lat','          Lon','          Ie','          Dep',
1'          a','          b','          sig'/
data lab1/'Lat ','Lon ','Ie ','Dep ','a ','b ','sig '/
data dtor/ 1.7453292519943296d-2/,gradla/111.195d0/
1,twopi/6.283185307d0/
aa=aai
bb=bbi
cc=cci
metc=met
is=0
ninc=nincin
iexit=.false.
npun=0
write(ifileout4,('' Attenuation function location method '''))
if(nincin.eq.1)then
    write(ifileout4,('' 2 free parameters (lat,lon) '''))
    ninc=2
else if(nincin.eq.2)then
    write(ifileout4,('' 3 free parameters (lat,lon,dep) '''))
    ninc=3
else if(nincin.eq.3)then
    write(ifileout4,('' 3 free parameters (lat,lon,Ie) '''))

```

```

else if(nincin.eq.4)then
    write(ifileout4, '(' 4 free parameters (lat,lon,Ie,dep) ')')
else if(nincin.eq.5)then
    write(ifileout4, '(' 5 free param.(lat,lon,Ie,dep,lin) ')')
else if(nincin.eq.6)then
    write(ifileout4, '(' 6 free param.(lat,lon,Ie,dep,lin,log) ')')
1    )
endif
write(ifileout4,*)
do i=1,n
    if(dmod(fint(i),0.5d0).lt.1.d-4)then
        ddis=distgeo(xlaep,xloep,xla(i),xlo(i))
        ddi=diex_bili(ddis)
c
c    discard IDPs at long distances if iselp.ne.2
c
        if(iselp.eq.2.or.fin0-ddi.ge.4.d0)then
            npun=npun+1
            xlat(npun)=xla(i)
            xlon(npun)=xlo(i)
            cloc(npun)=locali(i)
            fin(npun)=fint(i)
            ddisv(npun)=ddis
        endif
    endif
enddo
c
xdepthout=999.d0
aicc=999.d0
bic=999.d0
rms=999.d0
xdepthin=depref
xdeptha=depref
c
c    compute initial Ie, for nincin <2 Ie can be computed (icomp=1) or
imposed (icomp=0)
c
    icomp=1
    if(nincin.gt.2.or.icomp.eq.1)then
        summ=0.d0
        do nn=1,npun
            summ=summ+fin(nn)+diex(ddisv(nn),xdeptha)
        enddo
        if(npun.gt.0)then
            fin0=summ/npun
        endif
    endif
c
c    compute intensity differences
c
    do nn=1,npun
        di(nn)=(fin0-fin(nn))
    enddo
    fint0=fin0
c
    fint1=fin0
    if(npun.lt.nmlocb)go to 125
c
c    set up distance weights
c

```

```

call wgtdis(ncldis,npun,ddisv,wgt)
c
c maximize likelihood
c
call maxlik(nincin,ninc,xlaep,xloep,xdeptha,fin0,
1aattcoe,battcoe,x,fvalue,ranlat,ranlon,depmax)
c
c compute variance, correlation, errors, etc.
c
call errors(nincin,ninc,iout,ifileout4,x,var,xlaer,xloer,
1fin0er,xdeper,aatter,batter)
c
c compute error ellipsoid axes 3D
c
xlaep=x(1)
xloep=x(2)
c
if(xlaer.ge.0.d0.and.xloer.ge.0.d0.and.fin0er.ge.0.d0.and.
1xdeper.ge.0.d0)then
    call ellips(nincin,ninc,iout,ifileout4,var,xlaep,
1    ermax,azmax,plmax,ermed,azmed,plmed,ermin,azmin,plmin)
endif
if(ermin.le.0.d0.or.ermanx.le.0.d0)then
    ecc=999.d0
else
    ecc=dsqrt(1.d0-(ermin**2.d0/ermanx**2.d0))
endif
c
c compute error ellipsoid axes 2D
c
if(xlaer.ge.0.d0.and.xloer.ge.0.d0.and.fin0er.ge.0.d0.and.
1xdeper.ge.0.d0)then
    nincinX2D=3
    nincX2D=3
    call ellips(nincinX2D,nincX2D,iout,ifileout4,var,xlaep,
1    ermax2d,azmax2d,aan,aan,aan,aan,ermin2d,azmin2d,aan)
endif
if(ermin2d.le.0.d0.or.ermanx2d.le.0.d0)then
    ecc2d=999.d0
else
    ecc2d=dsqrt(1.d0-(ermin2d**2.d0/ermanx2d**2.d0))
endif
c
if(ninc.ge.4)then
    xdeptha=x(4)
else if(nincin.eq.2)then
    xdeptha=x(3)
endif
xdepthout=xdeptha
aatt=aattcoe
batt=battcoe
if(nincin.gt.2)then
    fin0=x(3)
    if(ninc.ge.5)then
        aatt=x(5)
        if(ninc.ge.6)then
            batt=x(6)
        endif
    endif
endif
endif

```

c

```
if(met.eq.0)then
  if(npun.gt.ninc) then
    ssr=exp(fvalue*2.d0/npun)/npun
    rms=dsqrt(ssr*npun/(npun-ninc))
  endif
else
  ninc1=ninc+1
  if(npun.gt.ninc1)
1    rms=dsqrt(x(ninc1)*npun/(npun-ninc1))
    if(npun.gt.ninc1+1)then
      aicc=-fvalue-ninc1-ninc1*(ninc1+1)/(npun-ninc1-1)
      bic=-fvalue-ninc1/2.d0*dlog(npun/twopi)
    endif
  endif
endif
```

c

```
if(iout.ge.2)then
  if(met.eq.1)then
    write(ifileout4,
1      '(' AICC                               : ',f15.4)')aicc
    write(ifileout4,
1      '(' BIC                               : ',f15.4,/)')bic
  endif
endif
```

c

```
is=2
if(iout.ge.4)then
  write(ifileout4,*)'Data used to locate epi/hypocenter'
  write(ifileout4,*)
  write(ifileout4,*)'Locality           Dist Azi  '//
1  'Lat      Lon      Int  Res  Mag  Wgt'
endif
ddismin=100000.d0
sumres=0.d0
do npu=1,npun
  ddisv(npu)=distgeo(xlaep,xloep,xlat(npu),xlon(npu))
  aziv(npu)=azimul(xlaep,xloep,xlat(npu),xlon(npu),ddisv(npu))
  ddis=ddisv(npu)
  dit=diex6(ddis,xdepthout,aatt,batt)
  resv(npu)=dit-(di(npu)-fint1+fin0)
  if(ddis.lt.ddismin)then
    ddismin=ddis
    localimin=cloc(npu)
    fintdmin=fin(npu)
  endif
  sumres=sumres+resv(npu)
enddo
do np=1,npun
  iperm(np)=np
enddo
call sortp(npun,ddisv,work,iperm)
sumres=0.d0
sumama=0.d0
sumfin0=0.d0
sumwgt=0.d0
sum=0.d0
```

c

c

c

```
set up distance weights
```

```
call wgtdis(ncldis,npun,ddisv,wgt)
```



```

do np=1,npun
  npu=np
  fin01=fin(npu)+diexm6(ddisv(npu),xdepthout,aatt,batt,depref)
  ama(npu)=amg+bm*fin01
  sumwgt=sumwgt+wgt(npu)
  sumama=sumama+ama(npu)*wgt(npu)
  sumfin0=sumfin0+fin01*wgt(npu)
  sumres=sumres+resv(npu)
  if(iout.ge.4)then
    write(ifileout4,'(1x,a20,1x,F6.1,1x,i3,2(1x,f7.3),1x,f5.1,
1      2(1x,f5.2),2f7.3)')cloc(npu),ddisv(npu),
1      int(aziv(npu)+.5d0),xlat(npu),xlon(npu),fin(npu),
1      resv(npu),ama(npu),wgt(npu)
    endif
  enddo
  amm=sumama
  ammer=std(npun,amm,ama)
  ammtr=trimn(npun,ama,work)
  ammrer=tristdl(npun,ammtr,ama,work,npun1)
  if(iout.ge.4)then
    write(ifileout4,'(1x,A20,1x,6x,1x,3x,2(1x,7x),1x,5x,
1      2(1x,f5.2),2f7.3)') 'Total ',sumres/npun,sumama,sumwgt

    write(ifileout4,*)
  endif
  call sort(npun,aziv,aziv)
  aziv(npun+1)=aziv(1)+360.d0
  gapmax=0.d0
  do i=1,npun
    delaz=aziv(i+1)-aziv(i)
    if(delaz.gt.gapmax)then
      gapmax=delaz
      azil=aziv(i)
      azi2=dmod(aziv(i+1),360.d0)
    endif
  enddo
  enddo
  npe=npun
124  return
125  write(ifileout4,'(/,' N data low           : ',i5,
1'' iteration abandoned',/)' )npun
127  is=0
c
  return
end
c*****
*
c*SUB WGTDIS
*****
c*****
*

subroutine wgtdis(ncl,nn,ddisv,wgt)
implicit doubleprecision (a-h,o-z)
dimension icnt(50),ddisv(*),wgt(*)
dismax=0.d0
dismin=1000000.d0
do i=1,nn
  dismax=max(dismax,ddisv(i))
  dismin=min(dismin,ddisv(i))
enddo
ded=(dismax-dismin)/ncl

```

```

do id=1,ncl
    icnt(id)=0
enddo
do m=1,nn
    idv=min(dble(ncl),(ddisv(m)-dismin)/ded+1)
    icnt(idv)=icnt(idv)+1.d0
enddo
np=0
do id=1,ncl
    if(icnt(id).ne.0.d0) np=np+1
enddo
do m=1,nn
    idv=min(dble(ncl),(ddisv(m)-dismin)/ded+1)
    wgt(m)=1.d0/icnt(idv)/np
enddo
end
C*****
*
C*SUB MAXLIK
*****
C*****
*
    subroutine maxlik(nincin,ninc,xlaep,xloep,xdeptha,fin0,
1aattcoe,battcoe,x,fvalue,ranlat,ranlon,depmax)
    implicit doubleprecision (a-h,o-z)
C
C    maximize likelihood of attenuation equation with respect to unknown
parameters
C
    parameter(lp=7)
    parameter(lwk=lp+(2*lp+8))
    common/metlik/met
    dimension xguess(lp),xlb(lp),xub(lp),xscale(lp),iparam(7)
1,rparam(7),x(lp),wk(lwk),iwk(lp)
    integer ifilein1
    external likel_simi,likel_sim6
    data iparam/7*0/,rparam/7*0.d0/,xscale/lp*1.d0/
C
    xguess(1)=xlaep
    xguess(2)=xloep
    xlb(1)=xguess(1)-ranlat
    xlb(2)=xguess(2)-ranlon
    xub(1)=xguess(1)+ranlat
    xub(2)=xguess(2)+ranlon
C
    if(nincin.ge.3) then
        xguess(3)=fin0
        xlb(3)=-1.d0
        xub(3)=14.d0
    else
        xguess(3)=xdeptha
        xlb(3)=1.0d0
        xub(3)=depmax
    endif
    if(ninc.ge.4) then
        xguess(4)=xdeptha
        xlb(4)=1.0d0
        xub(4)=depmax
    endif
    if(ninc.ge.5) then

```

```

        xguess(5)=aattcoe
c      T=.2, Vs=3,
c      Q=T*Vs/2pi/ln(2)/b=10000
        xlb(5)=1.d-5
c      Q=T*Vs/2pi/ln(2)/b=0.9
        xub(5)=0.15d0
    endif
    if(ninc.ge.6)then
        xguess(6)=battcoe
c      spreading exp n=0.17
        xlb(6)=0.25d0
c      spreading exp n=ln(2)*b=2.8
        xub(6)=4.00d0
    endif
    ibtype=0
    fscale=1.d0
c
c      IPARAM, RPARAM
c
    do i=1,lp
        iparam(i)=0
        rparam(i)=0.d0
    enddo
c
    call du4inf(iparam,rparam)
c
c      IPARAM(1) Initialization flag.
        iparam(1)=1
c      IPARAM(2) Number of good digits in the function. Default: Machine
dependent
c      iparam(2)=0
c      IPARAM(3) Maximum number of iterations. Default: 100.
        iparam(3)=500
c      IPARAM(4) Maximum number of function evaluations. Default: 400.
        iparam(4)=800
c      IPARAM(5) Maximum number of gradient evaluations. Default: 400.
        iparam(5)=800
c
    if(met.eq.0)then
        ninc1=ninc
    else
        ninc1=ninc+1
        xlb(ninc1)=1.d-1
        xub(ninc1)=4.d0
        xguess(ninc1)=1.d0
    endif
    if(nincin.ge.3)then
        call dbconf(likel_sim6,ninc1,xguess,ibtype,xlb,xub,
1      xscale,fscale,iparam,rparam,x,fvalue)
    else
        call dbconf(likel_simi,ninc1,xguess,ibtype,xlb,xub,
1      xscale,fscale,iparam,rparam,x,fvalue)
    endif
c
    icod=iercd()
    iertp=nlrty(1)
c
    return
end

```

```

C*****
*
C*SUB ERRORS
*****
C*****
*
      subroutine errors(nincin,ninc,iout,ifileout4,x,var,xlaer,
1xloer,fin0er,xdeper,aatter,batter)
C
C      compute variance, correlation, errors, etc.
C
      implicit doubleprecision (a-h,o-z)
      parameter(lp=7)
      common/metlik/met
      common/attlaw/amgv,bmgv,aattcoev,battcoev,deprefv
      dimension h(lp,lp),var(lp,lp),wk1(lp),wk2(lp),xscale(lp)
1,x(lp),cor(lp,lp)
      external likel_simi,likel_sim6
      character *11 lab(lp),lab1(lp)*5
      data lab/'          Lat','          Lon','          Ie','          Dep',
1'          a','          b','          sig'/
      data lab1/'Lat ','Lon ','Ie ','Dep ','a ','b ','sig '/
      data xscale/lp*1.d0/
      data dtor/ 1.7453292519943296d-2/,gradla/111.195d0/
C
      if(met.eq.0)then
          ninc1=ninc
      else
          ninc1=ninc+1
      endif
      do i=1,ninc1
          do j=1,ninc1
              h(i,j)=0.d0
          enddo
      enddo
      eps=0.d0
      if(nincin.ge.3)then
          call likel_sim6(ninc1,x,alicpt)
          call df2hes(likel_sim6,ninc1,x,xscale,alicpt,eps,h,lp,wk1,wk2)
      else
          call likel_simi(ninc1,x,alicpt)
          call df2hes(likel_simi,ninc1,x,xscale,alicpt,eps,h,lp,wk1,wk2)
      endif
      do i=1,ninc1
          do j=i+1,ninc1
              h(i,j)=h(j,i)
          enddo
      enddo
      call dlinrg(ninc1,h,lp,var,lp)
C
      if(var(1,1).ge.0.d0)then
          xlaer=dsqrt(var(1,1))*gradla
      else
          xlaer=-1.d0
      endif
      if(var(2,2).ge.0.d0)then
          xloer=dsqrt(var(2,2))*gradla*dcos(x(1)*dtor)
      else
          xloer=-1.d0
      endif

```

```

if(nincin.ge.3)then
  if(var(3,3).ge.0.d0)then
    fin0er=dsqrt(var(3,3))
  else
    fin0er=-1.d0
  endif
  xdeper=999.d0
else
  if(var(3,3).ge.0.d0)then
    xdeper=dsqrt(var(3,3))
  else
    xdeper=-1.d0
  endif
  fin0er=999.d0
endif
if(ninc.ge.4)then
  if(var(4,4).ge.0.d0)then
    xdeper=dsqrt(var(4,4))
  else
    xdeper=-1.d0
  endif
else if(nincin.ne.2)then
  xdeper=999.d0
endif
if(ninc.ge.5)then
  if(var(5,5).ge.0.d0)then
    aatter=dsqrt(var(5,5))
  else
    aatter=-1.d0
  endif
endif
if(ninc.ge.6)then
  if(var(6,6).ge.0.d0)then
    batter=dsqrt(var(6,6))
  else
    batter=-1.d0
  endif
endif
if(met.eq.1)then
  if(var(ninc+1,ninc+1).ge.0.d0)then
    siger=dsqrt(var(ninc+1,ninc+1))
  else
    siger=-1.d0
  endif
endif
if(xlaer.lt.0.d0.or.xloer.lt.0.d0.or.fin0er.lt.0.d0.or.
1xdeper.lt.0.d0)then
  if(iout.ge.2)write(ifileout4,('' Hessian matrix error  : '''))
endif

c
c   reset values out of range
c
if(xlaer.lt.0.d0.or.xlaer.gt.998.d0)xlaer=999.d0
if(xloer.lt.0.d0.or.xloer.gt.998.d0)xloer=999.d0
if(xdeper.lt.0.d0.or.xdeper.gt.998.d0)xdeper=999.d0
if(fin0er.lt.0.d0.or.fin0er.gt.998.d0)fin0er=999.d0
c
if(iout.ge.2)then
  if(iout.ge.4)then
    write(ifileout4,'(/,' ' Hessian matrix''))

```

```

        call matrixwr(nincin,ninc,nincl,lab,lab1,h,ifileout4)
C
        write(ifileout4,'(/, '' Variance/covariance matrix'')')
        call matrixwr(nincin,ninc,nincl,lab,lab1,var,ifileout4)
        nmet=ninc
        if (met.eq.1) nmet=nincl
        if (var(1,1).ge.0.d0.and.var(2,2).ge.0.d0.and.
1      (ninc.le.2.and.var(nmet,nmet).ge.0.d0).and.
1      (ninc.le.3.and.var(nmet,nmet).ge.0.d0).and.
1      (ninc.le.4.and.var(nmet,nmet).ge.0.d0).and.
1      (ninc.le.5.and.var(nmet,nmet).ge.0.d0).and.
1      (ninc.le.6.and.var(nmet,nmet).ge.0.d0)) then
            do j=1,nincl
                do i=1,nincl
                    cor(i,j)=var(i,j)/dsqrt(var(i,i)*var(j,j))
                enddo
            enddo
C
            write(ifileout4,'(/, '' Correlation matrix'')')
            call matrixwr(nincin,ninc,nincl,lab,lab1,cor,ifileout4)
C
            endif
            write(ifileout4,*)
        endif
    endif
C
    return
end
C*****
*
C*SUB ELLIPS
*****
C*****
*
    subroutine ellips(nincin,ninc,iout,ifileout4,var,xlaep,ermax,azmax
1,plmax,ermed,azmed,plmed,ermin,azmin,plmin)
C
C    compute error ellipse/ellipsoid semiaxes
C
    implicit doubleprecision (a-h,o-z)
    parameter(lp=7)
    dimension var(lp,lp),var1(lp,lp),eval(lp),evec(lp,lp)
    character *11 lab(lp),lab1(lp)*5
    data lab/'          Lat','          Lon','          Ie','          Dep',
1'      'a','          b','          sig'/
    data lab1/'Lat ','Lon ','Ie ','Dep ','a ','b ','sig '/
    data dtor/ 1.7453292519943296d-2/,gradla/111.195d0/
C
    var1(1,1)=var(1,1)*gradla**2.d0
    var1(1,2)=var(1,2)*gradla**2.d0*dcos(xlaep*dtor)
    if(nincin.ge.3) then
        var1(1,3)=var(1,4)*gradla
    else
        var1(1,3)=var(1,3)*gradla
    endif
    var1(2,1)=var1(1,2)
    var1(2,2)=var(2,2)*(gradla*dcos(xlaep*dtor))**2.d0
    if(nincin.ge.3) then
        var1(2,3)=var(2,4)*(gradla*dcos(xlaep*dtor))
    else

```

```

    var1(2,3)=var(2,3)*(gradla*dcos(xlaep*dtor))
endif
var1(3,1)=var1(1,3)
var1(3,2)=var1(2,3)
if(nincin.ge.3)then
    var1(3,3)=var(4,4)
else
    var1(3,3)=var(3,3)
endif
numaut=min(3,ninc-1,nincin)
if(nincin.eq.1) numaut=2
if(nincin.eq.2) numaut=3
if(iout.ge.4)then
    write(ifileout4,'(/,' Ellipse Variance/covariance matrix''))
    if(numaut.le.2)then
        write(ifileout4,'(6a11)') lab(1),lab(2)
    else
        write(ifileout4,'(6a11)') lab(1),lab(2),lab(4)
    endif
    do k=1,2
        write(ifileout4,'(a5,6(1x,g10.4))') lab1(k),(var(k,j)
1      ,j=1,numaut)
    enddo
    if(numaut.eq.3)write(ifileout4,'(a5,6(1x,g10.4))')
1    lab1(4),(var(k,j),j=1,numaut)
    write(ifileout4,*)
endif
call devcsf(numaut,var1,lp,eval,vec,lp)
if(numaut.eq.2)then
    if(eval(1).ge.0.d0)then
        ermax=dsqrt(eval(1))
        azmax=dmod(450.d0-datan2(vec(1,1),vec(2,1))/dtor,360.d0)
        azmax=dmod(azmax+360.d0,180.d0)
        plmax=0.d0
    else
        ermax=-1.d0
        azmax=0.d0
        plmax=0.d0
    endif
    if(eval(2).ge.0.d0)then
        ermin=dsqrt(eval(2))
        azmin=dmod(450.d0-datan2(vec(1,2),vec(2,2))/dtor,360.d0)
        azmin=dmod(azmin+360.d0,180.d0)
        plmin=0.d0
    else
        ermin=-1.d0
        azmin=0.d0
        plmin=0.d0
    endif
    ermed=-1.d0
    azmed=0.d0
    plmed=0.d0
else if(numaut.eq.3)then
    if(eval(1).ge.0.d0)then
        if(vec(3,1).gt.0.d0)then
            vec(1,1)=-vec(1,1)
            vec(2,1)=-vec(2,1)
            vec(3,1)=-vec(3,1)
        endif
        ermax=dsqrt(eval(1))

```

```

        azmax=dmod(450.d0-datan2(evec(1,1),evec(2,1))/dtr,360.d0)
        azmax=dmod(azmax+360.d0,180.d0)
        plmax=dmod(360.d0+dasin(-evec(3,1))/dtr,360.d0)
    else
        ermax=-1.d0
        azmax=0.d0
        plmax=0.d0
    endif
    if(eval(2).ge.0.d0)then
        if(evec(3,2).gt.0.d0)then
            evec(1,2)=-evec(1,2)
            evec(2,2)=-evec(2,2)
            evec(3,2)=-evec(3,2)
        endif
        ermed=dsqrt(eval(2))
        azmed=dmod(450.d0-datan2(evec(1,2),evec(2,2))/dtr,360.d0)
        azmed=dmod(azmed+360.d0,180.d0)
        plmed=dmod(360.d0+dasin(-evec(3,2))/dtr,360.d0)
    else
        ermed=-1.d0
        azmed=0.d0
        plmed=0.d0
    endif
    if(eval(3).ge.0.d0)then
        if(evec(3,3).gt.0.d0)then
            evec(1,3)=-evec(1,3)
            evec(2,3)=-evec(2,3)
            evec(3,3)=-evec(3,3)
        endif
        ermin=dsqrt(eval(3))
        azmin=dmod(450.d0-datan2(evec(1,3),evec(2,3))/dtr,360.d0)
        azmin=dmod(azmin+360.d0,180.d0)
        plmin=dmod(360.d0+dasin(-evec(3,3))/dtr,360.d0)
    else
        ermin=-1.d0
        azmin=0.d0
        plmin=0.d0
    endif
endif
c
    end
c*****
*
c*SUB DIEX_BILI
*****
c*****
*
    doubleprecision function diex_bili(ddisi)
c
c    compute expected intensity decrement using bilinear law (Gasperini,
2001)
c
    implicit doubleprecision (a-h,o-z)
    common/atten/aa,bb,cc,fint1,xdeptha
c
    ddis=dsqrt(ddisi**2+100.d0)
    if(ddis.lt.45.d0)then
        diex_bili=aa+bb*ddis
    else
        diex_bili=aa+bb*45.d0+cc*(ddis-45.d0)
    endif
endfunction diex_bili

```



```

endif
C
    return
end
C*****
*
C*SUB DIEX
*****
C*****
*
    doubleprecision function diex(ddis,xdepth)
C
C    compute expected intensity decrement using log linear law
(pasolini, 2007)
C
    implicit doubleprecision (a-h,o-z)
    common/attcoe/aattcoe,battcoe,depref
C
    ddis=dsqrt(ddis**2+xdepth**2)
    diex=aattcoe*(ddis-xdepth)+battcoe*(dlog(ddis)-dlog(xdepth))
C
    return
end
C*****
*
C*SUB DIEX6
*****
C*****
*
    doubleprecision function diex6(ddis,xdepth,a,b)
C
C    compute expected intensity decrement using log linear law
(pasolini, 2007)
C
    implicit doubleprecision (a-h,o-z)
C
    ddis=dsqrt(ddis**2+xdepth**2)
    diex6=a*(ddis-xdepth)+b*(dlog(ddis)-dlog(xdepth))
C
    return
end
C*****
*
C*SUB DIEXM6
*****
C*****
*
    doubleprecision function diexm6(ddis,xdepth,aatt,batt,depref)
C
C    compute expected intensity decrement using log linear law
(pasolini, 2007)
C
    implicit doubleprecision (a-h,o-z)
C
    ddis=dsqrt(ddis**2+xdepth**2)
    diexm6=aatt*(ddis-depref)+batt*(dlog(ddis)-dlog(depref))
C
    return
end

```

```

C*****
*
C*SUB LIKEL_SIM6
*****
C*****
*
      subroutine likel_sim6(n,x,f)
C
C      compute log-likelihood function
C
      implicit doubleprecision (a-h,o-z)
      parameter(npts=3000)
      dimension x(n)
      common/metlik/met
      common/datepc/xlat(npts),xlon(npts),di(npts),wgt(npts),npun
C
      if(met.eq.0)then
          call fepc_sim6(n,x,ff)
          f=npun/2.d0*dlog(ff)
      else
          call fepc_met6(n,x,f)
      endif
C
      return
      end
C*****
*
C*SUB LIKEL_SIMI
*****
C*****
*
      subroutine likel_simi(n,x,f)
C
C      compute log-likelihood function
C
      implicit doubleprecision (a-h,o-z)
      parameter(npts=3000)
      dimension x(n)
      common/metlik/met
      common/datepc/xlat(npts),xlon(npts),di(npts),wgt(npts),npun
C
      if(met.eq.0)then
          call fepc_simi(n,x,ff)
          f=npun/2.d0*dlog(ff)
      else
          call fepc_meti(n,x,f)
      endif
C
      return
      end
C*****
*
C*SUB FEPC_MET6 *****
C*****
*
      subroutine fepc_met6(n1,x,f)
C
C      compute likelihood function
C
      implicit doubleprecision (a-h,o-z)

```

```

dimension x(n1)
parameter(npts=3000)
common/datepc/xlat(npts),xlon(npts),di(npts),wgt(npts),npun
common/attcoe/aattcoe,battcoe,depref
common/atten/aa,bb,cc,fint1,xdeptha
data dtor/ 1.7453292519943296d-2/,gradla/111.195d0/

```

c

```

f=0.d0
n=n1-1
sigma=x(n1)
fint0=x(3)
if(n.ge.4) then
  xdeptha=x(4)
  if(n.ge.5) then
    a=x(5)
    if(n.ge.6) then
      b=x(6)
    else
      b=battcoe
    endif
  else
    a=aattcoe
    b=battcoe
  endif
else
  a=aattcoe
  b=battcoe
endif
do i=1,npun
  aioss=fint1-di(i)
  iosssu=aioss+0.5d0+0.01d0
  iossl=aioss+0.01d0
  ddis=distgeo(x(1),x(2),xlat(i),xlon(i))
  dit=diex6(ddis,xdeptha,a,b)
  aiopr=fint0-dit
  anoru=(iossu-aiopr)/sigma
  anorl=(iossl-aiopr)/sigma

```

c

c

c

c

```

f= Ioss-Ipred= (Io-delIoss)-(Ie-delIpred)
               fint1-di(i)-(fint0-dit)

```

```

anou=dnordf(anoru+0.5d0)-dnordf(anoru-0.5d0)
anol=dnordf(anorl+0.5d0)-dnordf(anorl-0.5d0)
if(anou.gt.0.d0) then
  f=f+dlog(anou)*.5d0
else
  f=f-1.d300
endif
if(anol.gt.0.d0) then
  f=f+dlog(anol)*.5d0
else
  f=f-1.d300
endif
enddo
f=-(f-npun*dlog(sigma))
return
end

```

c*****
 *

```

C*SUB FEPC METI
*****
C*****
*
      subroutine fepc_meti(n1,x,f)
C
C      compute sum of squares of intensity residuals (used for
minimization)
C
      implicit doubleprecision (a-h,o-z)
      dimension x(n1)
      parameter(npts=3000)
      common/datepc/xlat(npts),xlon(npts),di(npts),wgt(npts),npun
      common/atten/aa,bb,cc,fint1,xdeptha
      common/attcoe/aattcoe,battcoe,depref
      data dtor/ 1.7453292519943296d-2/,gradla/111.195d0/
      f=0.d0
      n=n1-1
      sigma=x(n1)
      fint0=fint1
      if(n.gt.2)xdeptha=x(3)
      do i=1,npun
         ddis=distgeo(x(1),x(2),xlat(i),xlon(i))
         dit=diex(ddis,xdeptha)
         aioss=fint1-di(i)
         iossu=aioss+0.5d0+0.01d0
         iossl=aioss+0.01d0
         aiopr=fint0-dit
         anoru=(iossu-aiopr)/sigma
         anorl=(iossl-aiopr)/sigma
C
C         f= Ioss-Ipred= (Io-delIoss)-(Ie-delIpred)
C                               fint1-di(i)-(fint0-dit)
C
C         f=f+(dit-di(i)+fint1-fint0)**2*wgt(i)
C
C         f= Ioss-Ipred= (Io-delIoss)-(Ie-delIpred)
C                               fint1-di(i)-(fint1-dit)
C         that is  Ie=Io
C
         anou=dnordf(anoru+0.5d0)-dnordf(anoru-0.5d0)
         anol=dnordf(anorl+0.5d0)-dnordf(anorl-0.5d0)
         if(anou.gt.0.d0) then
            f=f+dlog(anou)*.5d0
         else
            f=f-1.d300
         endif
         if(anol.gt.0.d0) then
            f=f+dlog(anol)*.5d0
         else
            f=f-1.d300
         endif
         if(f.lt.-1.d300) f=-1.d10
      enddo
      f=-(f-npun*dlog(sigma))
      return
      end
C*****
*

```

```

C*SUB FEPC SIM6
*****
C*****
*
      subroutine fepc_sim6(n,x,f)
C
C      compute sum of squares of intensity residuals (used for
minimization)
C
      implicit doubleprecision (a-h,o-z)
      dimension x(n)
      parameter(npts=3000)
      common/datepc/xlat(npts),xlon(npts),di(npts),wgt(npts),npun
      common/attcoe/aattcoe,battcoe,depref
      common/atten/aa,bb,cc,fintl,xdeptha
      data dtor/ 1.7453292519943296d-2/,gradla/111.195d0/
      f=0.d0
      fint0=x(3)
      if(n.ge.4)then
          xdeptha=x(4)
          if(n.ge.5)then
              a=x(5)
              if(n.ge.6)then
                  b=x(6)
              else
                  b=battcoe
              endif
          else
              a=aattcoe
              b=battcoe
          endif
      else
          a=aattcoe
          b=battcoe
      endif
      do i=1,npun
          ddis=distgeo(x(1),x(2),xlat(i),xlon(i))
          dit=diex6(ddis,xdeptha,a,b)
C
C          f= Ioss-Ipred= (Io-delIoss)-(Ie-delIpred)
C                                fintl-di(i)-(fint0-dit)
C
          f=f+(dit-di(i)+fintl-fint0)**2*wgt(i)
      enddo
      return
      end
C*****
*
C*SUB FEPC SIMI
*****
C*****
*
      subroutine fepc_simi(n,x,f)
C
C      compute sum of squares of intensity residuals (used for
minimization)
C
      implicit doubleprecision (a-h,o-z)
      dimension x(n)
      parameter(npts=3000)

```

```

common/datepc/xlat(npts),xlon(npts),di(npts),wgt(npts),npun
common/atten/aa,bb,cc,fint1,xdeptha
data dtor/ 1.7453292519943296d-2/,gradla/111.195d0/
f=0.d0
if(n.gt.2)xdeptha=x(3)
do i=1,npun
  ddis=distgeo(x(1),x(2),xlat(i),xlon(i))
  dit=diex(ddis,xdeptha)
C
C      f= Ioss-Ipred= (Io-delIoss)-(Ie-delIpred)
C                      fint1-di(i)-(fint0-dit)
C
C      f=f+(dit-di(i)+fint1-fint0)**2*wgt(i)
C
C      f= Ioss-Ipred= (Io-delIoss)-(Ie-delIpred)
C                      fint1-di(i)-(fint1-dit)
C      like assuming Ie=Io
C
C      f=f+(dit-di(i))**2*wgt(i)
enddo
return
end
C*****
*
C*SUB MEAN
*****
C*****
*
      doubleprecision function mean(n,x)
C
C      Computes arithmetic average
C
      implicit doubleprecision (a-h,o-z)
      doubleprecision x(*)
      temp=0.d0
      if(n.ge.1)then
        do i=1,n
          temp=temp+x(i)
        enddo
        mean=temp/n
      else
        mean=999.d0
      endif
      return
      end
C*****
*
C*SUB SORTP
*****
C*****
*
      SUBROUTINE SORTP(N,RA,RB,IPerM)
C
C      REAL VECTOR SORT WITH PERMUTATION VECTOR
C
      INTEGER      N,IPerM(*)
      DOUBLEPRECISION RA(*), RB(*)
      INTEGER      I, IJ, IL(21), IT, ITT, IU(21), J, K, L, M
      DOUBLEPRECISION R, T, TT
      EXTERNAL DCOPY

```

```

C      CALL DCOPY (N, RA, 1, RB, 1)
C                                     CHECK FOR INPUT errors
C      IF (N .LE. 0) THEN
C          write(ifileout4,101)N
101  FORMAT(' THE LENGTH OF VECTORS RA, RB AND ',
1      ' IPerM MUST BE GREATER THAN OR EQUAL TO ONE. ',
2      ' ON INPUT THE LENGTH, N, IS GIVEN AS ',I5)
      STOP 1
      END IF
C
      M = 1
      I = 1
      J = N
      R = .375D0
10  IF (I .EQ. J) GO TO 70
      IF (R .LE. .5898437D0) THEN
          R = R + 3.90625D-2
      ELSE
          R = R - .21875D0
      END IF
20  K = I
C                                     SELECT A CENTRAL ELEMENT OF THE
C                                     ARRAY AND SAVE IT IN LOCATION T
      IJ = I + (J-I)*R
      T = RB(IJ)
      IT = IPerM(IJ)
C                                     IF FIRST ELEMENT OF ARRAY IS GREATER
C                                     THAN T, INTERCHANGE WITH T
      IF (RB(I).gt. T) THEN
          RB(IJ)= RB(I)
          RB(I)= T
          T = RB(IJ)
          IPerM(IJ)= IPerM(I)
          IPerM(I)= IT
          IT = IPerM(IJ)
      END IF
      L = J
C                                     IF LAST ELEMENT OF ARRAY IS LESS THAN
C                                     T, INTERCHANGE WITH T
      IF (RB(J).GE. T) GO TO 40
      RB(IJ)= RB(J)
      RB(J)= T
      T = RB(IJ)
      IPerM(IJ)= IPerM(J)
      IPerM(J)= IT
      IT = IPerM(IJ)
C                                     IF FIRST ELEMENT OF ARRAY IS GREATER
C                                     THAN T, INTERCHANGE WITH T
      IF (RB(I).LE. T) GO TO 40
      RB(IJ)= RB(I)
      RB(I)= T
      T = RB(IJ)
      IPerM(IJ)= IPerM(I)
      IPerM(I)= IT
      IT = IPerM(IJ)
      GO TO 40
30  IF (RB(L).EQ. RB(K)) GO TO 40
      TT = RB(L)
      RB(L)= RB(K)

```

```

        RB(K)= TT
        ITT = IPerM(L)
        IPerM(L)= IPerM(K)
        IPerM(K)= ITT
C
C                                FIND AN ELEMENT IN THE SECOND HALF OF
                                THE ARRAY WHICH IS SMALLer THAN T
40  L = L - 1
    IF (RB(L).gt. T)GO TO 40
C
C                                FIND AN ELEMENT IN THE FIRST HALF OF
                                THE ARRAY WHICH IS GREATER THAN T
50  K = K + 1
    IF (RB(K).LT. T)GO TO 50
C
C                                INTERCHANGE THESE ELEMENTS
    IF (K .LE. L)GO TO 30
C
C                                SAVE UPPER AND LOWER SUBSCRIPTS OF
                                THE ARRAY YET TO BE SORTED
    IF (L-I .LE. J-K)GO TO 60
    IL(M)= I
    IU(M)= L
    I = K
    M = M + 1
    GO TO 80
60  IL(M)= K
    IU(M)= J
    J = L
    M = M + 1
    GO TO 80
C
C                                BEGIN AGAIN ON ANOTHER PORTION OF
                                THE UNSORTED ARRAY
70  M = M - 1
    IF (M .EQ. 0)GO TO 9000
    I = IL(M)
    J = IU(M)
80  IF (J-I .GE. 11)GO TO 20
    IF (I .EQ. 1)GO TO 10
    I = I - 1
90  I = I + 1
    IF (I .EQ. J)GO TO 70
    T = RB(I+1)
    IT = IPerM(I+1)
    IF (RB(I).LE. T)GO TO 90
    K = I
100 RB(K+1)= RB(K)
    IPerM(K+1)= IPerM(K)
    K = K - 1
    IF (T .LT. RB(K))GO TO 100
    RB(K+1)= T
    IPerM(K+1)= IT
    GO TO 90
C
9000 CONTINUE
    RETURN
    END
C*****
*
C*SUB SORT
*****
C*****
*
        SUBROUTINE SORT(N, RA, RB)

```



```

C
C      REAL VECTOR SORT
C
C      INTEGER      N
C      DOUBLEPRECISION RA(*), RB(*)
C      INTEGER      I, IJ, IL(21), IU(21), J, K, L, M
C      DOUBLEPRECISION R, T, TT
C      EXTERNAL DCOPY
C
C
C      CALL DCOPY (N, RA, 1, RB, 1)
C
C      CHECK INPUT ARGUMENT
C      IF (N .LE. 0) THEN
C          write(ifileout4,101)N
101      FORMAT(' N = ',I5,' THE LENGTH OF RA AND RB, N',
1      ', MUST BE GREATER THAN OR EQUAL TO ONE.')
C          STOP 1
C      END IF
C
C      M = 1
C      I = 1
C      J = N
C      R = .375D0
10  IF (I .EQ. J) GO TO 80
20  IF (R .LE. .5898437D0) THEN
C          R = R + 3.90625D-2
C      ELSE
C          R = R - .21875D0
C      END IF
30  K = I
C
C      SELECT A CENTRAL ELEMENT OF THE
C      ARRAY AND SAVE IT IN LOCATION T
C
C      IJ = I + (J-I)*R
C      T = RB(IJ)
C
C      IF FIRST ELEMENT OF ARRAY IS GREATER
C      THAN T, INTERCHANGE WITH T
C
C      IF (RB(I).gt. T) THEN
C          RB(IJ) = RB(I)
C          RB(I) = T
C          T = RB(IJ)
C      END IF
C      L = J
C
C      IF LAST ELEMENT OF ARRAY IS LESS THAN
C      T, INTERCHANGE WITH T
C
C      IF (RB(J).GE. T) GO TO 50
C      RB(IJ) = RB(J)
C      RB(J) = T
C      T = RB(IJ)
C
C      IF FIRST ELEMENT OF ARRAY IS GREATER
C      THAN T, INTERCHANGE WITH T
C
C      IF (RB(I).LE. T) GO TO 50
C      RB(IJ) = RB(I)
C      RB(I) = T
C      T = RB(IJ)
C      GO TO 50
40  IF (RB(L).EQ. RB(K)) GO TO 50
C      TT = RB(L)
C      RB(L) = RB(K)
C      RB(K) = TT
C
C      FIND AN ELEMENT IN THE SECOND HALF OF

```

```

C                                     THE ARRAY WHICH IS SMALLer THAN T
50 L = L - 1
   IF (RB(L).gt. T)GO TO 50
C                                     FIND AN ELEMENT IN THE FIRST HALF OF
C                                     THE ARRAY WHICH IS GREATER THAN T
60 K = K + 1
   IF (RB(K).LT. T)GO TO 60
C                                     INTERCHANGE THESE ELEMENTS
   IF (K .LE. L)GO TO 40
C                                     SAVE UPPER AND LOWER SUBSCRIPTS OF
C                                     THE ARRAY YET TO BE SORTED
   IF (L-I .LE. J-K)GO TO 70
   IL(M)= I
   IU(M)= L
   I = K
   M = M + 1
   GO TO 90
70 IL(M)= K
   IU(M)= J
   J = L
   M = M + 1
   GO TO 90
C                                     BEGIN AGAIN ON ANOTHER PORTION OF
C                                     THE UNSORTED ARRAY
80 M = M - 1
   IF (M .EQ. 0)GO TO 9000
   I = IL(M)
   J = IU(M)
90 IF (J-I .GE. 11)GO TO 30
   IF (I .EQ. 1)GO TO 10
   I = I - 1
100 I = I + 1
   IF (I .EQ. J)GO TO 80
   T = RB(I+1)
   IF (RB(I).LE. T)GO TO 100
   K = I
110 RB(K+1)= RB(K)
   K = K - 1
   IF (T .LT. RB(K))GO TO 110
   RB(K+1)= T
   GO TO 100
C
9000 CONTINUE
C
   RETURN
   END
C*****
*
C*SUB STD
*****
C*****
*
doubleprecision function std(n,am,x)
C
implicit doubleprecision (a-h,o-z)
C
compute standard deviation
C
doubleprecision x(*)
temp=0.d0

```

```

        if(n.gt.1)then
            do i=1,n
                temp=temp+(x(i)-am)**2.d0
            enddo
            std=dsqrt(temp/(n-1))
            if(dabs(temp).lt.0.d0)std=0.d0
        else
            std=999.d0
        endif
        return
    end

C*****
*
C*SUB TRIMN
*****
C*****
*
    doubleprecision function trimn(n,x,work)
C
    implicit doubleprecision (a-h,o-z)
C
    compute 20% trimmed mean
C
    doubleprecision x(*),work(*),mean
    if(n.lt.5)then
        trimn=mean(n,x)
        return
    endif
C
C    sort
C
    call sort(n,x,work)
C
C    compute the number of data to discard
C
    isca=n*.20d0+.5d0
    trimn=0.d0
    do i=1+isca,n-isca
        trimn=trimn+work(i)
    enddo
    trimn=trimn/(n-2.d0*isca)
    return
    end
C*****
*
C*SUB TRISTD
*****
C*****
*
    doubleprecision function tristd(n,am,x,work)
    implicit doubleprecision (a-h,o-z)
C
C    compute trimmed standard deviation
C
    doubleprecision x(*),work(*)
    if(n.lt.5)then
        tristd=std(n,am,x)
        return
    endif
C

```

```

c      sort
c
c      call sort(n,x,work)
c
c      compute the number of data to discard
c
c      isca=n*.20d0+.5d0
c      temp=0.d0
c      do i=1+isca,n-isca
c          temp=temp+(x(i)-am)**2.d0
c      enddo
c      tristd=dsqrt(temp/(n-2.d0*isca-1))
c      return
c      end
c*****
c*
c*SUB TRISTD1
c*****
c*****
c*
c      doubleprecision function tristd1(n,am,x,work,ntr)
c      implicit doubleprecision (a-h,o-z)
c
c      compute trimmed standard deviation
c
c      doubleprecision x(*),work(*)
c      if(n.lt.5)then
c          tristd1=std(n,am,x)
c          ntr=n
c          return
c      endif
c
c      sort
c
c      call sort(n,x,work)
c
c      compute the number of data to discard
c
c      isca=n*.20d0+.5d0
c      temp=0.d0
c      do i=1+isca,n-isca
c          temp=temp+(x(i)-am)**2.d0
c      enddo
c      tristd1=dsqrt(temp/(n-2.d0*isca-1))
c      ntr=n-2*isca
c      return
c      end
c*****
c*
c*SUB MAXV
c*****
c*****
c*
c      doubleprecision function maxv(n,x)
c      implicit doubleprecision (a-h,o-z)
c
c      compute max on an array
c
c      doubleprecision x(*)
c      maxv=x(1)

```

```

        do i=2,n
            maxv=max(maxv,x(i))
        enddo
        return
    end
C*****
*
C*SUB DISGTEO
*****
C*****
*
    doubleprecision function distgeo(xlat1,xlon1,xlat2,xlon2)
C
C    compute geodaetic distance(in km)for a spherical earth
C
    implicit doubleprecision (a-h,o-z)
    parameter(ar=6371.d0)
    dtor=datan(1.d0)/45.d0
    arg=dsin(xlat2*dtor)*dsin(xlat1*dtor)+dcos(xlat2*dtor)
    1*dcos(xlat1*dtor)*dcos((xlon1-xlon2)*dtor)
    arg=max(-1.d0,min(1.d0,arg))
    distgeo=dacos(arg)*ar
    return
    end
C*****
*
C*SUB MAGNI1
*****
C*****
*
    subroutine magnil(coef,nrag,rmedn,va,erro,ndat,fintg,amm,ammer,
    1nok,iout,aiv,stda,awme,spes,ifileout4)
C
C    compute macroseismic magnitude using average radii of intensity
classes
C    using sibol et al. (1987) formula new weighting and error scheme
C
C    Input:
C
C    coef: Sibol magnitude coefficients (3 x nrag matrix)
C           coef(1,*): intercept
C           coef(2,*): Log10(FA)^2 coefficient
C           coef(3,*): Io^2 coefficient
C    nrag: number of isoseismal radii
C    rmedn: isoseismal radii (in km) (vector)
C    ndat: number of IDPs for each isoseismal (vector)
C    fintg: epicentral intensity (Io)
C    iout: printout information level (1-10)
C    aiv: intensity lower bound of isoseismal radii (vector)
C    stda: standard deviation of regression for each isoseismal (vector)
C    awme: average weight of each isoseismal
C    ifileout4: Fortran unit number of output file
C
C    Output:
C
C    va: computed magnitude for each isoseismal (vector)
C    erro: magnitude uncertainties for each isoseismal (vector)
C    amm: computed magnitude
C    ammer: type-i magnitude uncertainty
C    nok: number of used radii

```

```

c      spes:  type-ii magnitude uncertainty
c
c
c      Subroutine and functions:
c
c      provided: sortp
c
c
c      implicit doubleprecision (a-h,o-z)
c      parameter (mxva=50)
c      dimension coef(3,*),rmedn(*),ndat(*),aiv(*),stda(*),awme(*)
c      dimension va(mxva),erro(mxva),val(mxva),erro1(mxva),va2(mxva)
c      1,rva(mxva),niq(mxva),iper(mxva)
c      data pi/3.141592741012573d0/
c      iacc=1
c      iwgt=1
c      aminer=1.e30
c      do ll=1,nrag
c
c      zeroes val
c
c      rva(ll)=0.d0
c      va(ll)=0.d0
c      niq(ll)=0
c      erro(ll)=0
c
c      compute magnitude for j-th radius
c
c      if(ndat(ll).ne.0)then
c          xint=fintg
c          niq(ll)=ndat(ll)
c          rva(ll)=rmedn(ll)
c
c      compute weight of the given radius using new weighting
c
c          erro(ll)=stda(ll)/(dsqrt(dble(niq(ll)))/dsqrt(awme(ll)))
c          aminer=min(aminer,erro(ll))
c          va(ll)=+coef(1,ll)
c      1          +coef(2,ll)*dlog10(pi*rmedn(ll)**2)**2
c      1          +coef(3,ll)*xint**2
c      endif
c      enddo
c
c      compute average
c
c      nok=0
c      nmag=0
c      sum=0.d0
c      anmag=0.d0
c      ndat1=0
c      do l=1,nrag
c          if(va(l).ne.0)then
c              nok=nok+1
c              ndat1=ndat(l)
c              val(nok)=va(l)
c              erro1(nok)=erro(l)
c              sum=sum+va(l)*(1.d0+(1.d0/erro(l)**2-1.d0)*iwgt)
c              anmag=anmag+(1+(1.d0/erro(l)**2-1.d0)*iwgt)
c              nmag=nmag+1
c          endif

```

```

        enddo
c
c      warning !!! eliminate single estimates with less than 4 points
c
      if(nok.eq.1.and.ndat1.lt.4)nok=0
      smed=999.d0
      if(nok.ne.0)then
        amean=sum/anmag
        smed=dsqrt(1.d0/anmag)
        iok=0
        if(iout.ge.2)then
          sum=0.d0
          sum1=0.d0
          ner=0
          write(ifileout4,'(/,1x,a,/)')
1        'Iseismal radii to compute magnitude:'
          write(ifileout4,'(1x,a)')
1        '  I      R      area      N  M      Wgt'
          do l=1,nrag
            if(va(l).ne.0.d0)then
              write(ifileout4,'(1x,f5.1,1x,f6.1,1x,f8.0,1x,i4,1x,
1              f4.2,1x'//',f8.4)')aiv(l),rmedn(l),pi*rmedn(l)**2,
1              ndat(l),va(l),1.d0/erro(l)**2
              iok=1
              sum1=sum1+1.d0/stda(l)**2
              sum=sum+1.d0/erro(l)**2
              ner=ner+1
            endif
          enddo
        endif
      endif
c
c      compute mean, median and standard deviations
c
      s=0.d0
c
c      standard deviation
c
      do l=1,nrag
        if(va(l).ne.0.d0)then
          s=s+(amean-va(l))**2*(1.d0+(1.d0/erro(l)**2-1.d0)*iwgt)
        endif
      enddo
      if(nok.gt.1)then
        s=dsqrt(s/anmag/(nmag-1))
      else
        s=999.d0
      endif
c
c      sort
c
      do jm=1,nok
        iper(jm)=jm
      enddo
      call sortp(nok,va1,va2,iper)
c
c      compute median
c
      if(mod(nok,2).eq.0)then
        ame=(va2(nok/2)+va2(nok/2+1))/2.d0
      else

```

```

        ame=va2(nok/2+1)
    endif
    if(nok.gt.1.and.iout.ge.2)then
        write(ifileout4,'(/,1x,a,1x,3f5.2)')
1        'Mean and std          : ',amean,s,smed
        write(ifileout4,'(1x,a,1x,f5.2)')
1        'Median              : ',ame
    endif

C
C    trimmed mean with at least 4 estimates
C
        s=999.d0
        if(nok.ge.4)then
            if(iout.ge.2)then
                write(ifileout4,'(/,1x,a,/)' )'Trimmed mean used '
                write(ifileout4,'(1x,a,1x,15f7.2)')
1                'M   ',(va2(l),l=2,nok-1)
                write(ifileout4,'(1x,a,1x,15f7.2)')
1                'Wgt',(1.d0/erro1(iper(l))**2,l=2,nok-1)

                write(ifileout4,*)
            endif
            sum=0.d0
            ntot=0
            antot=0
            antot1=0
            do l=2,nok-1
                sum=sum+va2(l)*(1.d0+(1.d0/erro1(iper(l))**2-1.d0)*iwgt)
                antot=antot+(1.d0+(1.d0/erro1(iper(l))**2-1.d0)*iwgt)
                antot1=antot1+(1.d0+(1.d0-1.d0)*iwgt)
                ntot=ntot+1
            enddo
            sum=sum/(antot)
            s=0.d0

C
C    trimmed standard deviation
C
            do l=2,nok-1
                s=s+(sum-va2(l))**2*(1.d0+
1                (1.d0/erro1(iper(l))**2-1.d0)*iwgt)
            enddo
            if(ntot.le.1)then
                s=999.d0
            else
                s=dsqrt(s/antot/(nok-3))
            endif
            spes=dsqrt(1.d0/antot)
        else

C
C    with nok < 4 use median
C
            spes=smed
            sum=ame
        endif
    else

C
C    computation of magnitude impossible
C
        s=999.d0
        spes=999.d0

```



```

        ame=0.d0
        sum=0.d0
        amean=0.d0
    endif
    amm=sum
    ammer=s
c
    return
end
c*****
*
c*SUB MAGNI2
*****
c*****
*
    subroutine magni2(amg,bmg,aatt,batt,depref,npo,raiw,fint
1,fintg,ammc,ammer,ammctr,ammertr,fint0,nn,nmlocb,xdepthf
1,ncldis,nok2,iout,ifileout4,iselp,aai,bbi,cci)
c
c    Compute alternative macroseismic magnitude using attenuation
function
c
c    Input:
c
c    amg: intercept of M-Io relationship (Pasolini et al., 2008b)
c    bmg: coefficient of M-Io relationship (Pasolini et al., 2008b)
c    aatt: coefficient a (linear term) of attenuation equation (Pasolini
et al., 2008b)
c    batt: coefficient b (log term) of attenuation equation (Pasolini et
al., 2008b)
c    depref: reference depth (km) (Pasolini et al., 2008b)
c    npo: number of intensity data points (IDPs)
c    raiw: epicentral distances of each IDP (in Km) (vector)
c    fint: intensity of each IDP (vector)
c    fintg: epicentral intensity (computed by epcntr)
c    nmlocb: minimum number of IDPs to compute epicenter (10)
c    xdepthf: effective hypocentral depth (km) (computed by epcsea)
c    ncldis: 1 (number of classes for distance weighting)
c    iout: printout information level (1-10)
c    ifileout4: Fortran unit number of output file
c    iselp: 1 discard IDPs at long distances (Standard attenuation law)
c           2 include IDPs at long distances (Extended attenuation law)
c    aai,bbi,cci: coefficients of bilinear attenuation equation for data
selection
c    (0.53, 0.055, 0.022) (Pasolini et al., 2008a)
c
c    Output:
c
c    ammc: computed magnitude
c    ammer: uncertainty of computed magnitude
c    ammctr: trimmed average magnitude
c    ammertr: uncertainty of trimmed average magnitude
c    fint0: intensity at epicenter (Ie)
c    nn: number of select data points (IDPs) by attenuation law
c    nok2: output status (1: succesfull, 0:unsuccessful)
c
c
c    Subroutine and functions:
c
c    provided: wgtdis,trimn,tristdl

```

c
c

```
implicit doubleprecision (a-h,o-z)
common/atten/aa,bb,cc,sfint1,sxdeptha,sfint0
dimension raiw(*),fint(*),aint0v(5000),wgt(5000),fintv(5000),
1raiwv(5000),work(5000)
aa=aai
bb=bbi
cc=cci
if(npo.le.nmlocb)then
    nok2=0
    return
endif
nn=0
do i=1,npo
    if(dmod(fint(i),0.5d0).lt.1.d-4.and.fint(i).gt.2.d0)then
        ddi=diex_bili(raiw(i))
```

c
c
c

```
discard IDPs at long distances if iselp.ne.2

        if(iselp.eq.2.or.fintg-ddi.ge.4.d0)then
            fir=diexm6(raiw(i),xdepthf,aatt,batt,depref)
            nn=nn+1
            aint0v(nn)=(fint(i)+fir)

            fintv(nn)=fint(i)

            raiwv(nn)=raiw(i)
        endif
    endif
enddo
if(nn.le.nmlocb)then
    nok2=0
    go to 100
endif
```

c
c
c

weight over ncldis distance intervals

```
call wgtdis(ncldis,nn,raiw,wgt)
sum=0.d0
do m=1,nn
    sum=sum+aint0v(m)*wgt(m)
enddo
trime=trimn(nn,aint0v,work)
trist=tristd1(nn,trime,aint0v,work,ntr)
fint0=sum
sum2=0.d0
do m=1,nn
    sum2=sum2+(aint0v(m)-fint0)**2*wgt(m)
enddo
err=dsqrt(sum2)
ammc=amg+bm* (fint0)
ammer=bm*err/dsqrt(dfloat(nn-1))
ammctr=amg+bm*trime
ammertr=bm*trist/dsqrt(dfloat(ntr))
nok2=1
if(iout.ge.5)then
write(ifileout4,'(/,a,/)' ) 'Data used to compute magnitude (ATN) '
write(ifileout4,'(a,/)' )
1'      I      Dist      Ie      M      Res      Wgt'
```

```

        do m=1,nn
        amagv=amg+bm*aint0v(m)
        write(ifileout4,'(1x,f5.1,f7.1,f6.2,f5.2,f6.2,f7.3)')
        lfv(m),raiw(m),aint0v(m),amagv,amagv-ammc,wgt(m)
        enddo
        endif
100    return
        end
C*****
*
C*SUB ORIENN
*****
C*****
*
        subroutine orienn(dista,azim,aint,npo,fintg,acram,bcram,al,
        langle,stda,prayl,pkuip,ndat,ierr,nmin,ndecr,iout,locali,ifileout4)
C
C    Compute orientation of macroseismic data distribution
C
C    p.g. 4/9/2009 new std according to Fisher (1993), corrected
Rayleigh and Kuiper test
C
C    Input:
C
C    dista: epicentral distance of each Intensity Data Point (IDP) (in
Km) (vector)
C    azim: axial orientation with respect to epicentre of each IDP (in
degrees) (vector)
C    aint: intensity at each IDP (vector)
C    npo: number of IDPs
C    fintg: maximum intensity
C    acram: a coefficient of CRAM attenuation function (Berardi et al.,
1993)
C            (-0.46, Gasperini et al., 1999)
C    bcram: b coefficient of CRAM attenuation function (Berardi et al.,
1993)
C            (0.93, Gasperini et al., 1999)
C    al: fault length (e.g. computed from magnitude by Wells and
Coppersmith, 1994)
C    nmin: minimum number of IDPs to compute fault orientation (3)
C    ndecr: number of half degrees decrements (4)
C    iout: printout information level
C    locali: locality name of each IDP (character*20 vector)
C    ifileout4: Fortran unit number of output file
C
C    Output:
C
C    angle: axial orientation of fault trace (in degrees)
C    stda: fault orientation uncertainty (in degrees)
C    prayl: significance level (s.l.) of Rayleigh test for uniformity
C    pkuip: significance level (s.l.) of Kuiper test for uniformity
C    ndat: number of IDPs used to compute fault orientation
C    ierr: error status (0: succesfull, 1:unsuccessful)
C
C    Subroutine and functions:
C
C    provided: sort
C
        implicit doubleprecision (a-h,o-z)
        parameter(ktot=3000)

```

```

dimension dista(*),aint(*),azim(*)
dimension deltai(ktot),aziz(ktot),aziz1(ktot),sina(ktot)
1,cosa(ktot),wmod(ktot)
dimension azm(8),azmv(8),dis(8),nnn(8),fr(8),fk(8),fp(8),fs(8)
character *20 locali(*)
dtor=datan(1.d0)/45.d0
finte=fintg
ammxw=0.d0
do kt=1,npo
    deltai(kt)=finte-aint(kt)
    dist=(max(0.d0,deltai(kt))-acram)/bcram
    if(azim(kt).ne.999.d0)then
        wmod(kt)=dista(kt)**(1.d0/3.d0)/dist
        sina(kt)=wmod(kt)*dsin(2.d0*azim(kt)*dtor)
        cosa(kt)=wmod(kt)*dcos(2.d0*azim(kt)*dtor)
        ammxw=max(ammxw,wmod(kt))
    else
        sina(kt)=0.d0
        cosa(kt)=0.d0
        wmod(kt)=0.d0
    endif
enddo
do in=1,ndecr
c
c     ain intensity decrement
c
    ain=(in-1)*.5d0
    kt1=0
    dmed=0
    nmed=0
    sumc=0.d0
    sums=0.d0
    sumdis=0.d0
    azml=200.d0
    azmlb=200.d0
    do kt=1,npo
        if(wmod(kt).ne.0.d0.and.deltai(kt).le.ain)then
            kt1=kt1+1
            sums=sums+sina(kt)
            sumc=sumc+cosa(kt)
            sumdis=sumdis+wmod(kt)
            dmed=dmed+dista(kt)
            nmed=nmed+1
        endif
    enddo
c
c     compute orientation only if no. of data ge nmin
c
    if(kt1.ge.nmin)then
        dmed=dmed/kt1
        if((sumc.ne.0.d0.or.sums.ne.0.d0))then
c
c     compute angular mean and rayleigh tests
c
            sums=sums/kt1
            sumc=sumc/kt1
            sumdis=sumdis/kt1
            azmlb=datan2(sums,sumc)/dtor/2.d0
            vec=dsqrt(sums**2+sumc**2)
            rr=vec/sumdis

```

```

        if(rr.lt.0.65d0)then
            ak=rr/6.d0*(12.d0+6.d0*rr**2+5.d0*rr**4)
        else
            rr=min(rr,.999999d0)
            ak=1.d0/(2.d0*(1.d0-rr)-(1.d0-rr)**2-(1.d0-rr)**3)
        endif
c
c      p.g. 8/9/2009
c      correct argument of Rayleigh test probability according to
(Fisher, 1993, eq 4.17)
c
c      aka=2.d0*kt1*rr**2
      aka=kt1*rr**2
      p=exp(-aka)*(1.d0+(2.d0*aka-aka**2)/(4.d0*kt1)-
1      (24.d0*aka-132.d0*aka**2+76.d0*aka**3-9.d0*
1      aka**4)/(288.d0*kt1**2))
      p=max(0.d0,p)
      ss=1.d0/dsqrt(kt1*rr*ak)
      sazml=sumdis-vec
      s0=sazml/sumdis
c
c      p.g. 4/9/2009
c      compute sample circular dispersion (Fisher, 1993, eq 2.28)
c
      sinmed=sums/vec
      cosmed=sumc/vec
      angmed=datan2(sinmed,cosmed)
      sumc2=0.d0
      sums=0.d0
      sumc=0.d0
      sumdis=0.d0
      do kt=1,npo
          if(wmod(kt).ne.0.d0.and.deltai(kt).le.ain)then
              sumdis=sumdis+wmod(kt)
              sinang=dsin(2.d0*azim(kt)*dtor)
              cosang=dcos(2.d0*azim(kt)*dtor)
              sindif=-sinmed*cosang+cosmed*sinang
              cosdif=sinmed*sinang+cosmed*cosang
              diff=datan2(sindif,cosdif)
              sums=sums+wmod(kt)*dsin(diff)
              sumc=sumc+wmod(kt)*dcos(diff)
              sumc2=sumc2+wmod(kt)*dcos(2.d0*diff)
          endif
      enddo
      del=(1-sumc2/sumdis)/(2.d0*rr**2)
c
c      compute circular standard error Fisher, 1993, eq 4.21)
c
      std=dsqrt(del/kt1)/dtor/2.d0
      else
          azmlb=200.d0
          s0=0.d0
          rr=0.d0
          aka=0.d0
          p=0.d0
          ss=0.d0
      endif
      endif
      azm(in)=azml
      azmv(in)=azmlb

```

```

        dis(in)=dmed
        nnn(in)=kt1
        fr(in)=rr
        fk(in)=ak
        fp(in)=p
        fs(in)=std
    enddo
    if(azmv(ndecr).eq.200.d0)then
        ierr=1
        go to 999
    endif
    dd=10000.d0
    ii=0
    if(iout.ge.2)then
        write(ifileout4,'(/,1x,a)') '  DI      Dist      N      Azim'
    endif
    do i=1,ndecr
c
c      find average distance closer to half fault length 'al/2'
c
        if(iout.ge.2)then
            if(azmv(i).ge.0.d0)then
                az=azmv(i)
            else
                az=180.d0+azmv(i)
            endif
            if(az.lt.200.d0)then
                write(ifileout4,'(1x,f5.1,1x,f7.1,1x,i4,1x,f6.1)')
1              (i-1)*.5,dis(i),nnn(i),az
            else
                write(ifileout4,'(1x,f5.1,1x,f7.1,1x,i4)')
1              (i-1)*.5,dis(i),nnn(i)
            endif
            if(azmv(i).ne.200.d0.and.dabs(1.d0-dis(i)/(al/2.d0)).
1            le.abs(1.d0-dd/(al/2.d0)))then
                ii=i
                dd=dis(i)
            endif
        enddo
        if(dis(ndecr).lt.al/2.d0)then
            ii=ndecr
            dd=dis(ndecr)
        endif
        if(iout.ge.2)write(ifileout4,'(1x)')
        if(ii.ne.0)then
c
c      distance found, do kuiper tests
c
            kt1=0
            ain=(ii-1)*.5d0
            if(iout.ge.2)then
                write(ifileout4,'(/,1x,a,f7.1)')
1              'Average distance closer to half fault length: ',dd
                write(ifileout4,'(1x,a,f7.1)')
1              'Maximum intensity decrement',ain
                write(ifileout4,'(/,1x,a)')'Used data for azimuth:'
                write(ifileout4,'(/,1x,a)')
1              ' Azim  DI  Dist  Wgt  locality'
            endif

```

```

do kt=1,npo
deltai(kt)=finte-aint(kt)
if(wmod(kt).ne.0.d0.and.deltai(kt).le.ain)then
    kt1=kt1+1
c
c
c
c
p.g. 4/9/2009
    aziz(kt1)=dmod(azim(kt)+360.d0,360.d0)

    aziz(kt1)=dmod(2.d0*azim(kt)+360.d0,360.d0)
    if(iout.ge.2)then
        if(azim(kt).ge.0.d0)then
            azimkt=azim(kt)
        else
            azimkt=azim(kt)+180.d0
        endif
        write(ifileout4,'(1x,f6.1,1x,f3.1,1x,f6.1,1x,f7.4,
1          1x,a20)')azimkt,deltai(kt),dista(kt),wmod(kt)/ammxw,
1          locali(kt)
        endif
    endif
enddo
write(ifileout4,'(1x)')
if(kt1.ne.0)then
    call sort(kt1,aziz,aziz1)
    pos1=0.d0
    fneg1=0.d0
    do kt=1,kt1
        dif1=dfloat(kt)/dfloat(kt1)-aziz1(kt)/360.d0
        dif2=aziz1(kt)/360.d0-dfloat(kt-1)/dfloat(kt1)
        pos1=max(pos1,dif1)
        fneg1=max(fneg1,dif2)
    enddo
    vn1=pos1+fneg1
    vvnn=1.d0/(dsqrt(dfloat(kt1))+0.155d0+0.24d0/
1    dsqrt(dfloat(kt1)))
    if(vn1.ge.vvnn*2.001d0)then
        pkul=.01d0
    else if(vn1.ge.vvnn*1.862d0)then
        pkul=.025d0
    else if(vn1.ge.vvnn*1.747d0)then
        pkul=.05d0
    else if(vn1.ge.vvnn*1.620d0)then
        pkul=.10d0
    else
        pkul=1.d0
    endif
endif
ierr=0
angle=azmv(ii)
if(angle.lt.0.d0)angle=180.d0+angle
stda=fs(ii)
prayl=fp(ii)
pkui=pkul
ndat=nnn(ii)
else
    ierr=1
    angle=200.d0
    stda=0.d0
    prayl=0.d0
    pkui=0.d0

```

```

        ndat=0
    endif
C
999    return
    end
C*****
*
C*SUB MATRIXWR
*****
C*****
*
    subroutine matrixwr(nincin,ninc,nincl,lab,lab1,trix,ifileout4)
C
    implicit real*8 (a-h,o-z)
    parameter(lp=7)
    common/metlik/met
    character *11 lab(lp),lab1(lp)*5
    dimension trix(lp,lp)
C
    if(nincin.ge.3)then
        write(ifileout4,'(7a11)') (lab(j),j=1,ninc),
1      (lab(7),j=ninc,nincl-1)
        do k=1,ninc
            write(ifileout4,'(a5,7(1x,g10.4))')lab1(k),
1      (trix(k,j),j=1,ninc),(trix(k,nincl),j=ninc,nincl-1)
        enddo
        if(met.eq.1)then
            write(ifileout4,'(a5,7(1x,g10.4))')lab1(7),
1      (trix(nincl,j),j=1,ninc),(trix(nincl,nincl),j=ninc,nincl-1)
        endif
    else
        if(nincin.eq.1)then
            write(ifileout4,'(7a11)')lab(1),lab(2),
1      (lab(7),j=ninc,nincl-1)
        else
            write(ifileout4,'(7a11)')lab(1),lab(2),lab(4),
1      (lab(7),j=ninc,nincl-1)
        endif
        do k=1,ninc-1
            write(ifileout4,'(a5,7(1x,g10.4))')lab1(k),
1      (trix(k,j),j=1,ninc),(trix(k,nincl),j=ninc,nincl-1)
        enddo
        write(ifileout4,'(a5,7(1x,g10.4))')lab1(4),
1      (trix(ninc,j),j=1,ninc),(trix(ninc,nincl),j=ninc,nincl-1)
        if(met.eq.1)then
            write(ifileout4,'(a5,7(1x,g10.4))')lab1(7),
1      (trix(nincl,j),j=1,ninc),(trix(nincl,nincl),j=ninc,nincl-1)
        endif
    endif
C
    return
    end
C*****
*
C*SUB AZIMUT
*****
C*****
*
    doubleprecision function azimuth(alatp,alonp,alatx,alonx,dis)
C

```



```

c      compute azimuth (in degrees)of second point with respect to first
one
c      for a spherical earth
c
      implicit doubleprecision (a-h,o-z)
      parameter(ar=6371.d0)
      dtor=datan(1.d0)/45.d0
      dist=dis/ar
      sinaz=dcos(alatx*dtor)*dsin((alonx-alonp)*dtor)/dsin(dist)
      if(dsin(alatx*dtor).ge.dcos(dist)*dsin(alatp*dtor)) then
        azimuth=dasin(min(1.d0,max(-1.d0,sinaz)))/dtor
      else
        azimuth=-dasin(min(1.d0,max(-1.d0,sinaz)))/dtor
      endif
      return
      end
c*****
*
c*SUB AZIMU1
*****
c*****
*
      doubleprecision function azimul(alatp,alonp,alatx,alonx,dis)
c
c      compute azimuth (in degrees)of second point with respect to first
one
c      for a spherical earth
c
      implicit doubleprecision (a-h,o-z)
      parameter(ar=6371.d0)
      dtor=datan(1.d0)/45.d0
      dist=dis/ar
      sinaz=dcos(alatx*dtor)*dsin((alonx-alonp)*dtor)/dsin(dist)
      if(dsin(alatx*dtor).ge.dcos(dist)*dsin(alatp*dtor)) then
        azimul=asin(min(1.d0,max(-1.d0,sinaz)))/dtor
      else
        azimul=180.d0-dasin(min(1.d0,max(-1.d0,sinaz)))/dtor
      endif
      azimul=dmod(azimul+360.d0,360.d0)
      return
      end
c+++++
+
c      end boxer package
+++++
c+++++
+
```