```
      program sphgric
c program for the prediction of spherical harmonic coefficients
c using fast Least-Squares Collocation (LSC) from data gridded
c equidistantly in longitude. Error estimates are also calculated and
c error-correlations may be calculated optionally, see Ref. given below.
c Programmed 2000-10-31 by cct, update 2006-11-20.
c
c For one given latitude the data must be of the same kind, be in
c the same altitude and have the same error. However it is possible to
c have two or more kinds of data associated with the same parallel.
c Presently 9 kinds of data can be used, identified by an integer code:
c Data type                          code     units
c anomalous potential (T)             10     m**2/s**2
c geoid height                        11     m
c gravity disturbance                 12     mgal
c gravity anomaly                     13     mgal
c radial gravity gradient (Trr)       15     E.U.
c derivative in nothern direction, Tn 16    mgal
c 2. order derivatives Tnr,           20     E.U.
c 2.order derivative   Tnn,           22     E.U.
c 2. order derivative  Tee            24     E.U.
c All functionals are of the kind which when applied on a spherical
c harmonic expansion leaves the terms dependent on longitude unchanged,
c (expect for the sign for Tee).
c
c Data may be point or mean values. Means are calculated as the mean
c of nstep*nstep values. Actual value of nstep is found in a parameter
c statement and must be an odd number.
c
c The data may be generated using a spherical harmonic expansion,
c input from a file containing data generated by an earlier run
c of the program or observed values. This may also be used to test
c the program, i.e. the input and output coefficients should be
c close (seen in relation to the error estimate).
c
c Note that data-types may be mixed and in different altitudes.
c In this case  either geodetic or geocentric coordinates may
c be used.
c
c The program may be run in spherical approximation, i.e. so that
c the distance from the origin is calculated as the sum of the mean
c earth-radius and the altitude. (Not recommended in general).
c
c The covariance function used may either be represented by a finite Legendre
c series or as the sum of a Legendre series and a closed expression.
c See the subroutines covax, covbx and covcx and the references in these
c modules.
c
c Various testing possibilities have been implemented. The most important
c is that a full LSC solution may be calculated when the number of data is
c small.  Furthermore an alternative method for calculating associated
c Legendre functions has been implemented, and the results for the
c routine (lecurs) may be compared to the simpler spharm0 routine
c which is based on standard recursion algorithms. Problems are
c to be expected for high degree and high latitude.
c
c Input may be done interactively or by preparing an input file
c for batch processing. Input has the following structure in the
c most simple case:
c
c ------------------------ INPUT (1) ------------------------
c use spherical approximation ? (t/f)                      F
c ------------------------ INPUT (2) ------------------------
c Test of program ? (T/F)                                   F
c ------------------------ INPUT (3) ------------------------
c Is the grid equidistant in latitude ? (T/F)              T
c   If the answer here is F (=false), the data and error specifications
c   must be given for each parallel, see INPUT (3.2.1) - (3.2.4).
c ------------------------ INPUT (3.1.1) ------------------------
```



```
c input gravsoft grid label (latmin,max,lonmin,max,dlat,dlon)
c using geodetic latitude.
c                              -80.0 80.0 0.0 340.0 20.0 20.0
c ------------------------ INPUT (3.1.2) ------------------------
c input altitude of points (m)                          300000.0
c ------------------------ INPUT (3.1.3) ------------------------
c input functional type (10: anomalous potential, 11: geoid,
c      13: gravity, 15: Trr   13
c ------------------------ INPUT (3.1.4) ------------------------
c input common standard deviation of noise                0.005
c ------------------------ INPUT (3.1.5) ------------------------
c are all data mean values (T/F) ?                          F
c ------------------------ INPUT (4) ------------------------
c input degree of spherical harmonic expansion
c to be used for test or comparison purpose              180
c ------------------------ INPUT (5) ------------------------
c input name of file with coefficients
c used for coparison or data generation                  pcoeff
c ------------------------ INPUT (6) ------------------------
c input name of file to hold calculated coefficients     ccoeff
c ------------------------ INPUT (7) ------------------------
c read formatted (T/F) ?                                    T
c ------------------------ INPUT (7.1) ------------------------
c input format e.g. (2I4,2d19.12)                (2I4,2d20.12)
c ------------------------ INPUT (8) ------------------------
c Use of closed expressions ?                               F
c   If the answer here is T, input of specifications of the closed
c   expression, see subroutine INCOV, input (6) - (7D).
c ------------------------ INPUT (8.2.1) ------------------------
c input minimum and maximal degree of degree-variances    2 180
c ------------------------ INPUT (8.2.2) ------------------------
c input name of file with degree-variances (units mgal**2)  egm96.edg
c ------------------------ INPUT (9) ------------------------
c input maximal degree for prediction                       8
c ------------------------ INPUT (10) ------------------------
c output of error-estimate for max. degree to file ?        T
c ------------------------ INPUT (10.1) ------------------------
c input name of file to hold error-estimates             eco8.dat
c ------------------------ INPUT (10.2) ------------------------
c input t if error-correlations are to be computed          F
c ------------------------ INPUT (11) ------------------------
c Input observations from file(s) ? (T/F)                     F
c   See Input (11.1.1) - (11.1.4.1) if the answer is T:
c   IF T, then input of  number of files and for each file
C   file-name, number of data-items, and
c   the number in the data list of the data to be used
c   followed (on a new line) t if data are geodetic coordinates and
c   f if they are geocentric.
c   data kind identifier (10, 11, 12, 13, 15, 16, 20, 22, 24)
c   If F, data will be generated by the program from the
c   coefficients.
c ------------------------ INPUT (12) ------------------------
c Output of observations to file (T/F)                      F
c   If the answer is T, input of files names to hold observations.
c ------------------------ INPUT (13) ------------------------
c Will covariances be input from file (T/F) ?               F
c   If T, input of file-names (13.1), otherwise the coefficients are
c   output to  one or more files, the name of which must be input.
c   This can be used if the same data points are used again,
C   with a new standard deviation of the error or with new values.
c ------------------------ INPUT (14) ------------------------
c Output of coefficients and differences to current output ? (T/F)          T
c   If the answer is F, only output to the file with name given in
c   Input (6).
c
c If a more complex input is needed, please see the instructions
c contained below in the program.
c
c Output consist of the input parameters, predicted and observed
```

```
c coefficients, their standard deviation and contingently the error
c covariances.
c A summary of the results are given. If both full LSC and fast LSC  are used
c a comparison of the results is made.
c The covariance functions used in fast LSC are stored on a file 'covsph'
c an overwritten in the next run if not renamed. It may be rather large.
c
c Reference:
c Sanso', F. and C.C.Tscherning: Fast Spherical Collocation - Theory and
c Examples. J. of Geodesy, Vol. 77, pp. 101-112, DOI 10.1007/s00190-002-0310-5,
c 2003.
c
c Pending updates: Automatic generation of grids symmetic around Equator.
c For the moment (2004-07-02) implemented in gengri.f
C Update 2006-01-29: Change if covariance functions with finite
c sums are used, using implicitly a Bjerhammar sphere with radius A.
c
      implicit none
      integer maxgrid,maxcof,maxd,maxobs,nstepi,nc,izero,i,j1,
     *nsteps,irec,nnarr,ks,j,kj,m,n,iha,iko,maxdeg,ih,
     *molda,nolda,ih1,kt,idim,iwkdim,idim2,nchlis,nwarn,k,maxcc,
     *n1,ikp,maxc,nd2,nstep,nn1,ncof,mout,maxpre,kc1,kc2,k4,
     *maxd2,ki0,num,ki,idm,jdm,iolda,jolda,k0,i0,k2,k3,ikold,ikpold,
     *ik1old,ik,k1,ik1,i1,j4,issd,m1,n3,n4,ii,ka,kb,kc,n1s,i5,icout,
     *ip,iq,mindeg,n2,ipk,nerror,j5,n11,m4,m3,ma,nout,idata,ndata
     *,KIX,NX1,NX2,imax,iimax,IIDEG,JJDEG,idif,iiold,jold,jm,im,ia,
     * ikn,ikt(8),nofile,noldb,moldb,idvc,maxpar,maxmer,m1n2,nrec,
     *itcoun,maxbl,nbl,nrel,maxfil,nbli,maxbls,maxrec
     *,ktest
c maxbl is maximal number of blocks to hold covariances, nbl is the
c current block number and nrel is the relative record number.
c     parameter (iimax=20000,maxpar=360,maxmer=720)
c     parameter (iimax=2000,maxpar=720,maxmer=720)
C values of maxpar, maxmer and maxobs changed 2005-06-12.
      parameter (iimax=2000,maxpar=1500,maxmer=1000,maxfil=10)
c     parameter (maxgrid=4050,maxcof=360,maxd=1800,maxobs=720*500,
c    *nstepi=3)
      parameter (maxgrid=4050,maxcof=360,maxd=1800,
     *maxobs=1500*1000,nstepi=3,maxbls=2**28)
c nstepi is the number of steps in the numerical integration of mean
c values.
      real*8 D0,D1,D2,D3,D4,D5,RE,PI,GM,fact,cov,
     *slat,a,clat,pim0,root,sq2,degv,cs,cc,cof,rlatg,rlatold,
     *rlon,sslon,cclon,disto,rlat,rlatc,hh,ra,ddlon,sdlon,rlatm,rlat0,
     *sno,snoise,h,dlon,dlod,dlad,rlonmad,rlonmid,
     *rlatmid,rem,fc,gra,pi4,e2,re2,ycc,yc,dyc,gjk,
     *dlfac,rlat1,rlon1,ss,rh,pnmr,pnm,fac,rj,dycmax,datin,dat0,
     *hinp,rloni,rlati,ssdat,sdat,g,s,si,pnm1,deg0,rlonmax,rlonmin,
     *dlat,rlatmin,sscv0,sscv,cv,cvmax,cvmin,cofr,err0,err2,
     *suaa,rc,cc0,rji,ycs,c0,pred,var,ssdegx,s1,t,rlonm,slatj,
     *clatj,c,fac2,fac1,rai,rlonj,slati,clati,ddyc,rlatmad,radeg,
     *degn2,sdeg,ssdeg,covf,g0mij,covff,sua,ssua,enlon,ci,error,
     *ersum,dsum,ddsum,cdif,cvari,d10,spharm0,ccmean,y,slon,clon,z,
     *raddeg,dist2,xy2,xy,x,wrk,cnm,snm,ecnm,esnm,fl,emean,hmean,
     *rb,sm,CIX,CRX,SIGMA0,SIGMA,HMAX,
     *CVX,D,HQ,HP,root0,SIGMAP,slop,sloq,clop,cloq,gc(3),
     *SUMIJ,CCCIJ,ys,yc1,vv,v1,gs,gc1,dds,ddc,degx,sn2,ecov,pnmk,
     *cosi,sini,cosdi,sindi,csdi,scsdi
     *,ctest(2000)
c rb is the Bjerhammar-sphere radius, a semi-major axis.
c
      logical ltest,lt,lf,lsphap,lforma,lnocc,lident,loutcc,lt00,
     *loout,loinp,lerout,lmean,lmean1,lincov,lchlis,legsym,lallme,
     *ltestc,lsin,lgeod,ldiout,linico,LOCAL,LSUM,LSPOUT,
     *LSPHAR,LTSPH,lclose,likn,lercov
c
C corrected 2004-08-13: RE substituted by REM.
C this is probably not changed correctly - 2005-05-10.
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,RE,RADDEG,PI,GM,ITCOUN,
```



```
     *LTEST,LF,LT
      common /rrsphc/root(0:maxd*2+2)
      common  /rrsph/root0(iimax)
c root is a precomputed square root-table (root(0)=0 !).
      COMMON /EUCL/X,Y,Z,XY,XY2,DISTO,DIST2,rlatc
c resulting cartesian coordinates and geocentric latitude.
      character *72 cofile,ifile,forma,obsfil,erout,covfil(maxfil),
     *ccoff
      character *24 udate
      COMMON /ITRANC/A,E2
      common /ww/cof((maxcof+1)**2),cofr(31**2,2),
c cof holds coefficients and cofr coefficients and error-estimates
c predicted by "full" collocation.
     *rlat(maxgrid,nstepi),rlat0(maxgrid,0:nstepi),
c rlat = geocentric latitude, rlat0 = geodetic latitude,
     *clat(maxgrid,nstepi),slat(maxgrid,nstepi),rlon(maxgrid),
     *clon(nstepi,maxgrid),slon(nstepi,maxgrid),
c clat,slat=cos and sin of latitude, rlon=longitudes,
     *ra(maxgrid,nstepi),hh(maxgrid),
c ra= length of radius-vector, in spherical approximation = R+hh.
     *g(maxgrid,maxgrid),
c g contains observations.
     *covf(maxgrid),c(maxgrid*(maxgrid+1)/2),rh(maxobs),izero(maxgrid),
c the array c will initially contain the covariances and the observations.
c it is an upper triangular matrix. rh holds right hand side of equations,
c (initially it is used to hold heights). izero position of first non-
c zero element in a column (here always = 1), see PRONLL.
     *degv(maxd),deg0(maxd),
c deg0 holds potential degree-varaince, degv the actual degree-variances.
     *sno(maxobs),iko(maxobs),rc(maxobs),
     *pnm(0:maxcof),pnm1(0:maxcof),sdeg(0:maxcof,8),
     *pnmr(maxgrid,23,nstepi),ersum(-maxcof:maxcof),
c pnmr are recursion elements used to evaluate spherical harmonics
c and their derivatives.
     *ssdeg(0:maxcof,8),fc(0:25),sscv(0:25),cv(0:25),nc(0:25),
     *iha(0:25),nolda(maxgrid,nstepi),molda(maxgrid,nstepi),
c control elements of degree and order in recursion.
     *lmean1(maxgrid)
c ra holds radius for each latitude.lmean1 holds values true if data
c on one parallel are mean values.
      COMMON /CMCOV/CIX(24),CRX(56),SIGMA0(1200),SIGMA(1200),HMAX,
     *CVX(2,2),D(36),KIX(37),NX1,NX2,LOCAL,LSUM
      COMMON /PDEGV/SIGMAP(2001),slop,sloq,clop,cloq,IIDEG,JJDEG,LSPOUT
c parameters used in covax, covbx and covcx. IIDEG and JJDEG are the degree
c and order of the coefficient to be predicted.
      common /con3/SUMIJ(32761),CCCIJ(32761),
     *sq2,ys,yc1,vv,v1,gs(3),gc1(3),dds(3,3),
     *ddc(3,3),iiold,jold,LSPHAR,LTSPH
      dimension wrk(1320),nnarr(2),dat0(10)
      dimension cclon(maxgrid*nstepi),sslon(maxgrid*nstepi),
     *noldb(maxgrid,nstepi),moldb(maxgrid,nstepi)
      dimension pnmk(maxpar),ecov(1000),cosi(0:maxmer),
     *sini(0:maxmer),cosdi(0:maxmer),sindi(0:maxmer)
c     *sini(0:maxmer),cosdi(0:2*maxcof),sindi(0:2*maxcof)
c cos and sin of j*delta longitude.
      DIMENSION sm(2001)
C THE ARRAY sm IS USED TO STORE THE DEGREE-VARIANCES WHEN THE LOGICAL
C VARIABLE LSUM IS TRUE. IN CASE THE SUBSCRIPT LIMIT IS CHANGED IT IS
C NECESSARY TO CHANGE THE VALUE OF THE VARIABLE N2 ACCORDINGLY.
C
      equivalence (clon(1,1),cclon(1))
      equivalence (slon(1,1),sslon(1))
c
c Files:   UNIT        used for                          name
c           2       Storage of Legendre functions        pnmfile
c         30+i      Covariance functions                 input each name
c           4       Grid definition file                 sphgrid0
c          18       Correlations of data on k'th
c                   parallel with Pnm                    scratch
```

```
c         20       error-covariances                    ercov.txt
c          9       In/out-put observations              input
c         12       Input of sphericl harmonic coeff.    input
c         15       gravity degree-variances             input
c         16       covariance table at equator.         covsphtab
c         21       output of error-estimates            input
c note that the covariances may be stored in several files.
c
c initializing dimension-variables for fft. (Not implemented).
      idim=51200
      iwkdim=1320
      idim2=idim*2
C
c initialisation of variables in common DCON.
      lt=.true.
      lf=.false.
      lallme=lt
      ltest=lf
      lchlis=lt
      lercov=lf
      nchlis=0
      ikpold=0
      ikn=0
      icout=0
      idvc=0
      lmean=lf
      lt00=lt
      lclose=lf
      d0=0.0d0
      d1=1.0d0
      d2=2.0d0
      d3=3.0d0
      d4=4.0d0
      d5=5.0d0
c GRS80 constants.
      A=6378137.0d0
c change 2005-0511 - try. 2005-05-13. Back.
c     RE=A
      REM=6371000.0d0
      RE=REM
      RE2=RE*RE
      GM=3.986005D14
c     E2=6.69438D-4
C erroneous value used until 2003-06-10. fl is the inverse of the flattening.
      fl=298.2572204d0
      fl=(d2-d1/fl)/fl
      E2=fl
      sq2=sqrt(d2)
c
      pi4=atan(d1)
      pi=d4*pi4
      radeg=180.0d0/pi
      raddeg=180.0d0/pi
      emean=d0
      hmean=d0
      rlatold=-180.0d0
c scale factor for gravity
      gra=GM/A**2
c     gra=GM/REM**2
c initializing array with conversion factors.
      fc(0)=d1
      fc(10)=GM/A
c     fc(10)=GM/REM
c conversion to m**2/s**2 for anomalous potential T.
      iha(10)=1
      iha(0)=0
      fc(12)=-GM/A**2*1.0d5
c     fc(12)=-GM/REM**2*1.0d5
      iha(12)=2
```



```
      fc(13)=-gra*1.0d5
c mgal units used.
      iha(13)=2
      fc(16)=fc(12)
      iha(16)=iha(12)
C converting to mgal.
      fc(11)=A
c     fc(11)=REM
      iha(11)=1
c converting to m.
      fc(15)=1.0d9*GM/(A**3)
c     fc(15)=1.0d9*GM/(REM**3)
c new change ...
      iha(15)=3
      fc(20)=fc(15)
      iha(20)=iha(15)
      fc(22)=fc(15)
      iha(22)=iha(15)
      fc(24)=fc(15)
      iha(24)=iha(15)
c converting to EU  (10**-9 s**-2).
      nwarn=0
c
c establish a square-root table.
      if (maxd.lt.maxcof) then
       write(*,*)' maxd < ',maxcof,' (maxcof) '
       stop
      end if
      do k=0,(maxd+1)*2
       root(k)=sqrt(dfloat(k))
      end do
c iniatializing square-root table to be used in COVCX.
      do k=1,iimax
       root0(k)=sqrt(k-d1)
      end do
c
      maxcc=maxgrid*(maxgrid+1)/2
c since the matrix will be full, we set the elements of the
c array izero to zero. (See subroutine PRONLL).
      do i=1,maxgrid
       izero(i)=1
      end do
c
      write(*,*)
     *' Spherical Harmonic Prediction in Grid, v. 10.3, 2005-06-29.'
      call fdate(udate)
      write(*,*)udate
c
c ------------------ INPUT (1) ----------------------------------
      write(*,*)' use spherical approximation ? (t/f) '
      read(*,*)lsphap
      if (lsphap) then
       write(*,*)' Spherical approximation in use '
c
       write(*,28)rem
  28   format(' Mean radius set to ',f10.1,' m.')
      else
       write(*,*)' Spherical approximation NOT in use. '
      end if
c
c ------------------ INPUT (2) ----------------------------------
      write(*,*)' Test of program ? (T/F) '
      read(*,*)ltest
      if (ltest) write(*,*)' Program runs in TEST mode '
c
c ------------------ INPUT (3) ----------------------------------
      write(*,*)' Is the grid equidistant in latitude ? (T/F) '
      read(*,*)lident
      if (lident) then
```

```
c
c ------------------ INPUT (3.1.1) ---------------------------------
      write(*,*)
     * ' input gravsoft grid label (latmin,max,lonmin,max,dlat,dlon)'
      write(*,*)' using geodetic latitude. '
      read(*,*)rlatmid,rlatmad,rlonmid,rlonmad,dlad,dlod
      write(*,51)rlatmid,rlatmad,rlonmid,rlonmad,dlad,dlod
  51  format(' Grid: ',6f8.3)
      n=(rlonmad-rlonmid)/dlod+1
      if (n.gt.maxmer) then
       write(*,*)' number of meridians ',n, ' exceeds ',maxmer
       stop
      end if
      n1=(rlatmad-rlatmid)/dlad+1
      if (n1.gt.maxpar) then
       write(*,*)' number of parallels ',n1,' exceeds ',maxpar
       stop
      end if
      if (abs(rlonmad-rlonmid-(n-1)*dlod).gt.1.0d-8
     *   .or.abs(rlonmad-rlonmid+dlod-360.0d0).gt.1.0d-3) then
c change 2003-08-19.
       write(*,*)' grid does not match ',rlonmid,rlonmad,dlod
       STOP
      end if
      rlatmin=rlatmid/radeg
      dlat=dlad/radeg
      rlonmin=rlonmid/radeg
      dlon=dlod/radeg
c
c ------------------ INPUT (3.1.2) ---------------------------------
      write(*,*)' input altitude of points (m) '
      read(*,*)h
c
c ------------------ INPUT (3.1.3) ---------------------------------
      write(*,61)
  61  format(' input functional type (10: anomalous potential',
     * ' 11: geoid, 12: gravity, disturbance,'/
     * ' 13: gravity anomaly, 15: Trr, 16: Tn, 20: Tnr, 22: Tnn, ',
     * ' 24: Tee')
      read(*,*)ikp
c for data types Tn, Tee, Tnr and Tnn closed expressions must be used
c (COVAX, COVBX and COVCX).
      lclose=lclose.or.ikp.eq.16.or.ikp.eq.20.or.ikp.eq.22.or.
     * ikp.eq.24
      lt00=ikp.eq.0
      if (.not.(ikp.eq.0.or.ikp.eq.10.or.ikp.eq.11.or.ikp.eq.12
     * .or.ikp.eq.13.or.ikp.eq.15.or.ikp.eq.16.or.ikp.eq.20
     * .or.ikp.eq.22.or.ikp.eq.24)) then
       write(*,*)' wrong data-type, stop '
       stop
      end if
c
c ------------------ INPUT (3.1.4) ---------------------------------
      write(*,*)' input common standard deviation of noise '
      read(*,*)snoise
C
c ------------------ INPUT (3.1.5) ---------------------------------
      write(*,*)' are all data mean values (T/F) ? '
      read(*,*)lmean
c
      write(*,66)h,ikp,snoise,lmean
  66  format(' height ',f12.1,' m , functional type ',i3,
     * ' data noise ', d15.4,/,' data mean values ',l2)
C
c this is an aid to create input-files with non-uniform spacing in latitude.
      open(4,file='sphgrid0')
      do k=1,n1
       write(4,53)rlatmid+(k-1)*dlad,h,ikp,snoise,lmean
  53   format(f10.5,f10.1,i4,d10.3,l2)
```



```
      end do
      close(4)
      write(*,*)' Grid definition output to file sphgrid0 '
      lgeod=.true.
     else
c ------------------ INPUT (3.2.1) ---------------------------------
      write(*,67)
  67  format(' Is grid given using geodetic latitude and ',
     * ' ellipsoidal height input T and ',/,
     * ' if geocentric latitude and r input F ')
      read(*,*)lgeod
      write(*,*)lgeod
c ------------------ INPUT (3.2.2) ---------------------------------
      write(*,*)
     * ' input minimal and maximal longitude and grid spacing (deg.) '
      read(*,*)rlonmin,rlonmax,dlod
      n=(rlonmax-rlonmin)/dlod+1
      if (n.gt.maxmer) then
       write(*,*)' number of meridians ',n,' exceeds ',maxmer
       stop
      end if
      if (abs(rlonmax-rlonmin-(n-1)*dlod).gt.1.0d-8) then
       write(*,*)' grid does not match ',n,rlonmax,rlonmin,dlod
       STOP
      else
       write(*,86)rlonmin,rlonmax,dlod
  86   format(' Longitude grid definition: ',2f10.4,f8.4)
      end if
c converting to radians.
      rlonmin=rlonmin/radeg
      dlon=dlod/radeg
c
c ------------------ INPUT (3.2.3) ---------------------------------
      write(*,*) ' input number of parallels '
      read(*,*)N1
      if (n1.gt.maxpar) then
       write(*,*)' number of parallels ',n1,' exceeds ',maxpar
       stop
      end if
c
c ------------------ INPUT (3.2.3) ---------------------------------
      write(*,*)' input table with values of: '
      if (lgeod) then
       write(*,*)
     *  ' latitude (deg.), height (m), data-type code, S.D.-observa. '
       write(*,*)' and logical, true if data are mean values '
      else
       write(*,*)
     *  ' geocentric latitude (deg.),  radius (m), data-type code,'
       write(*,*)
     *  ' S.D. observ. and logical, true if data are mean values '
      end if

     end if
     maxc=n*n1*(n*n1+1)/2
     nd2=n/2
c if the first meridian is different from the zero-meridian, it must
c be equal to half the longitude-spacing.
     if (abs(dlon/d2-rlonmin).gt.1.0d-4.and.rlonmin.gt.1.0d-8) then
      write(*,*)' Warning dlon, lonmin ',dlon,rlonmin
      nwarn=nwarn+1
     end if
c
     do k=1,n1
c rlat0 holds the geodetic latitude.
      if (lident) then
       sno(k)=snoise
       rlat0(k,0)=rlatmin+(k-1)*dlat
       lmean1(k)=lmean
```

```
        iko(k)=ikp
        ikn=1
        ikt(1)=ikp
       else
c ------------------ INPUT (3.2.4) ---------------------------------
        read(*,*)rlat0(k,0),h,iko(k),sno(k),lmean1(k)
        ikp=iko(k)
        if (.not.(ikp.eq.0.or.ikp.eq.10.or.ikp.eq.11.or.ikp.eq.12
     *  .or.ikp.eq.13.or.ikp.eq.15.or.ikp.eq.16.or.ikp.eq.20
     *  .or.ikp.eq.22.or.ikp.eq.24)) then
         write(*,*)' wrong data-type',ikp,' stop '
         stop
        end if
c added 2004-04-06.
        if (rlat0(k,0).gt.90.0.or.rlat0(k,0).lt.-90.0
     *  .or.h.lt.d0.or.h.gt.1.0d7) then
         write(*,*)' Wrong latitude or height ',rlat0(k,0),h
         stop
        end if
        if (sno(k).lt.d0) then
         write(*,*)' negative error estimate ',sno(k)
         stop
        end if
C added 2006-02-06 to check height or radial distance:
        if (lgeod) then
         if (h.lt.-20000.0d0.or.h.gt.1000000.0d0) then
          write(*,*)' h must be wrong ',h
          stop
         end if
        else
         if (h.lt.6300000.0d0.or.h.gt.7300000.0d0) then
          write(*,*)' r must be wrong ',h
          stop
         end if
        end if
c
c for data types Tn, Tnr and Tee closed expressions must be used
c (COVAX, COVBX and COVCX).
        lclose=lclose.or.ikp.eq.16.or.ikp.eq.20.or.ikp.eq.22
c here we count number of different functionals.
        if (ikpold.ne.ikp) then
c new feature 2003-12-08.
         likn=lf
         do ks=1,ikn
          likn=likn.or.ikp.eq.ikt(ks)
         end do
         if (.not.likn) then
          ikn=ikn+1
          ikt(ikn)=ikp
          ikpold=ikp
         end if
        end if
        emean=emean+sno(k)
        hmean=hmean+h
        if (.not.lgeod) then
c added 2003-01-31.
c here h is the radius vector length.
         x=h*cos(rlat0(k,0)/raddeg)
         y=d0
         z=h*sin(rlat0(k,0)/raddeg)
c conversion to geodetic latitude and ellipsoidal height.
         call TRANS(rlat0(k,0),rlatg,h)
         if (abs(rlatg*raddeg-rlat0(k,0)).gt.0.5d0.and.h.lt.1.0d4)
     *   write(*,*)' new lat ',rlatg*raddeg,rlat0(k,0),h
         rlat0(k,0)=rlatg*raddeg
c now h is the ellipsoidal height.
        end if
c
        lmean=lmean.or.lmean1(k)
```



```
         lt00=iko(k).eq.0
         rlat0(k,0)=rlat0(k,0)/radeg
        end if
        lallme=lallme.and.lmean1(k)
        ks=1
        rlatm=rlat0(k,0)
        if (lmean) then
         nstep=nstepi
         sdlon=dlon*(d1+nstepi)/(d2*nstepi)
         ddlon=dlon/nstepi
        else
         nstep=1
         sdlon=d0
         ddlon=d0
        end if
c
        do ks=1,nstep
c generate latitude for grid points.
         ra(k,ks)=h
         hh(k)=h
         rlat0(k,ks)=rlatm-sdlon+ks*ddlon
         clat(k,ks)=cos(rlat0(k,ks))
         slat(k,ks)=sin(rlat0(k,ks))
c correction 2004-02-03.
c calculating geocentric latitude (rlatc).
         call  EUCLID(clat(k,ks),slat(k,ks),d1,d0,H,E2,A)
c variables transferred through common-block /EUCL/
c now using geocentric latitude:
         if (abs(rlatc-rlat0(k,ks)).gt.d2*pi/180.0) then
          write(*,*)' WARNING ',rlatc*180.0/pi,rlat0(k,ks)*180.0/pi
          nwarn=nwarn+1
         end if
         clat(k,ks)=cos(rlatc)
         slat(k,ks)=sin(rlatc)
         rlat(k,ks)=rlatc
         ra(k,ks)=DISTO
        end do
        ks=1
       end do
       if (ikn.gt.1) write(*,171)ikn
 171   format(i4,' datatypes defined ')
       if (.not.lident) write(*,435)hmean/n1,emean/n1
 435   format(' mean altitude = ',f12.1,' mean error = ',f10.5)
c
       leqsym=lt
       do i=1,n1/2
        leqsym=leqsym.and.(abs(rlat(i,1)+rlat(n1-i+1,nstep)).lt.0.001
     * .and.iko(i).eq.iko(n1-i+1).and.(abs(ra(i,1)-ra(n1-i+1,nstep)).lt.
     *  0.1))
        if (lf) write(*,901)i,leqsym,rlat(i,1),rlat(n1-i+1,1)
 901    format(' i,lsy,rlat,rlat1 ',i3,l2,2f8.4)
       end do
       if (leqsym) write(*,*)' Data symmetric around Equator '
c
       if (lallme) write(*,*)' All values are mean values '
       if (lmean) then
c establishing tables of sines and cosines for mean covariance computation.
        do i=1,n*nstepi
         cclon(i)=cos((i-1)*ddlon)
         sslon(i)=sin((i-1)*ddlon)
        end do
       end if
c
       write(*,*)' grid contains ',N1,N,' points '
       if (n.gt.maxgrid.or.n1.gt.maxgrid) then
        write(*,*)' grid too large '
        stop
       end if
       nn1=n*n1
```

```
c
      do k=1,n
       rlon(k)=rlonmin+(k-1)*dlon
      end do
c initializing tables of cos and sin m*longitude and dlon, 2004-04-06.
      do i=0,n-1
       cosi(i)=cos(i*dlon)
       sini(i)=sin(i*dlon)
       cosdi(i)=cos(i*rlonmin)
       sindi(i)=sin(i*rlonmin)
      end do
c
c ------------------ INPUT (4) ----------------------------------
      write(*,*)' input degree of spherical harmonic expansion '
      write(*,*)' to be used for test-or comparison purpose. '
      read(*,*)ncof
      if (ncof.gt.maxcof) then
       write(*,*)' WARNING ',ncof,' > ',maxcof,' ncof changed '
       nwarn=nwarn+1
       ncof=maxcof
      end if
c
c ------------------ INPUT (5) ----------------------------------
      write(*,*)' Input name of file to hold calculated coefficients '
      read(*,('a'))ccoff
      write(*,*)ccoff
      write(*,*)' calculated coefficients will be output to ',ccoff
      open(4,file=ccoff)
c
c ------------------ INPUT (6) ----------------------------------
      write(*,*)
     *' input name of file with coefficients used for comparison '
      write(*,*)' or for the generation of data. '

      read(*,'(a)')cofile
      write(*,*)' coefficients in ',cofile
c
c ------------------ INPUT (7) ----------------------------------
      write(*,*)' read formatted (T/F) ? '
      read(*,*)lforma
      if (lforma) then
c
c ------------------ INPUT (7.1) --------------------------------
       write(*,*)' input format e.g. (2I4,2d19.12) '
       read(*,75)forma
  75   format(a72)
      end if
c
      open(12,file=cofile)
c initializing coefficient array.
      do i=1,(ncof+1)**2
       cof(i)=d0
      end do
c input of (unitless) fully normalized coefficients of anomalous potential.
      write(*,*)'i  j   c(j)   s(j)'
 100  if (lforma) then
c
c ------------------ INPUT (7.1.1) ------------------------------
       read(12,forma)i,j,cc,cs
      else
c ------------------ INPUT (7.2) --------------------------------
       read(12,*)i,j,cc,cs
      end if
      if (i.lt.5) write(*,20)i,j,cc,cs
  20  format(2i4,2d16.8)
      if (j.eq.0) then
       cof(i**2+1)=cc
      else
       cof(i**2+2*j)=cc
```



```
       cof(i**2+2*j+1)=cs
      end if
      if (i.ne.ncof.or.j.ne.ncof) go to 100
c
      write(*,26)ncof
  26  format(' Coefficients to degree ',i5,' input.')
      close(12)
c
c here we have the possibility for using closed expressions.
      if (.not.lclose) then
c ------------------ INPUT (8) ----------------------------------
       write(*,*)' Use of closed covariance expressiones ? (T/F) '
       read(*,*)linico
      else
       linico=lclose
      end if
      if (linico) then
       write(*,*)' Closed covariance functions in use. '
       KIX(6)=3
       KIX(7)=3
       CRX(2)=D0
       CRX(3)=D0
c ------------------ INPUT (8.1) --------------------------------
c all input instructions are found in the subroutine INCOV.
       call INCOV(lt,rb,imax)
       KIX(6)=17
       KIX(7)=17
       CALL COVBX(sm,lt,0)
       maxdeg=maxd
       write(*,*)' maxdeg= ',maxdeg
c degreevariances refer to the mean earth radius, REM.
       write(*,30)(SIGMAP(k),k=1,32)
       open(8,file='sigmap.dgv')
       write(8,30)(SIGMAP(k),k=1,721)
       close(8)
       write(*,*)' Degree-variances output to sigmap.dgv 0-720. '
C COVCX uses potential degree-variances in units of (m**2/s**2)**2,
c while here are used unitless quantities.
       do k=1,maxdeg
c       deg0(k)=(d2*(k-1)+d1)*SIGMAP(k)*(RE/GM)**2
        deg0(k)=(d2*(k-1)+d1)*SIGMAP(k)*(A/GM)**2
c       deg0(k)=(d2*(k-1)+d1)*SIGMAP(k)*(REM/GM)**2
       end do
       write(*,*)' Unitless potential coefficient-variances '
       write(*,30)(deg0(k),k=1,32)
      else
       write(*,*)' Closed covariance functions NOT in use. '
c initializing array with degree-variances.
       do k=1,maxd
        degv(k)=d0
       end do
c
c ------------------ INPUT (8.2.1) ------------------------------
       write(*,*)
     * ' input minimum and maximal degree of degree-variances '
       read(*,*)mindeg,maxdeg
       write(*,*)mindeg,maxdeg
 48    format(' input gravity anomaly degree variances (',i4,
     * ' - ',i4,')')
       if (maxdeg.ge.maxd) then
        write(*,*)' ERROR ',maxdeg,' > ',maxd
        STOP
       end if
C
c ------------------ INPUT (8.2.2) ------------------------------
       write(*,*)
     * ' input name of file with degree-variances (units mgal**2) '
       read(*,'(a)')ifile
       write(*,*)' degree-variances input from: ',ifile
```

```
      open(15,file=ifile)
      read(15,*)(degv(k),k=mindeg+1,maxdeg+1)
      mout=maxdeg+1
      if (maxdeg.gt.23) mout=24
      write(*,*)mout,' first degree-variances (mgal**2) '
      write(*,30)(degv(k),k=1,mout)
 30   format(6d12.5)
      cc=d0
      do K=1,maxdeg+1
       cc=cc+degv(k)
      end do
c
      write(*,96)cc,sqrt(cc)
 96   format(' Variance= ',f9.4,' mgal**2,st.dev. = ',
     * f9.4,' mgal (if data type ne 0). ',/)
c
c converting to potential degree-variances (unitless).
c the factor 1.0d-10 converts from mgal**2.
      deg0(1)=degv(1)
      deg0(2)=degv(1)
      do k=3,maxdeg+1
       if (lident.and.ikp.eq.0) then
        deg0(k)=degv(k)
       else
        deg0(k)= 1.0d-10*(REM**2/GM)**2*degv(k)/((k-d2)**2)
c       write(*,*)k,deg0(k)
       end if
      end do
      write(*,30)(deg0(k),k=1,32)
     end if
c
c ------------------ INPUT (9) ---------------------------------
     write(*,*)' input maximal degree for prediction '
     read(*,*)maxpre
     if (maxpre.gt.maxcof) then
      write(*,*)' max-prd too big ',maxpre,maxcof
      stop
     end if
     if (maxpre.gt.n/2) then
      write(*,*)' WARNING: above Nyquist-frequency'
      nwarn=nwarn+1
      if (maxpre.ge.n) then
       write(*,*)' WARNING: error if order .ge. ',n
       write(*,*)' results from full LSC solution will be output.'
       loutcc=lt
      end if
     end if
c
c The following possibility is only useful in order to create a table
c of error-estimates in a form suited for creating a graph.
c ------------------ INPUT (10) ---------------------------------
     write(*,*)
     *' output of error-estimate for max. degree to file ? '
      read(*,*)lerout
      if (lerout) then
c ------------------ INPUT (10.1) ---------------------------------
       write(*,*)
     * 'input name of file to hold error-estimates '
       read(*,'(a)')erout
       open(21,file=erout)
       write(*,*)
     * ' error estimates for degree ',maxpre,' output to ',erout
      end if
c
c ------------------ INPUT (10.2) ---------------------------------
     write(*,*)' Input t if error-correlations must be computed '
     read(*,*)lercov
     if (lercov) then
      write(*,*)
```



```
     * ' Error-covariances will be computed, and non-zero values '
       write(*,*)' output to file ercov.txt '
       open(18,access='direct',form='unformatted',
     * status='scratch',RECL=8*n1)
       open(20,file='ercov.txt')
      else
       write(*,*)' Error-covariances will NOT be computed '
      end if
c
c open file to hold values of Legendre functions for test purposes only,
c using the subroutine lecurs.
      maxd2=ncof+1
c
      if (maxpre.gt.ncof) maxd2=maxpre+1
      if (ltest.and.(.not.lmean)) then
       OPEN(2,ACCESS='DIRECT',FORM='UNFORMATTED',
     * file='pnmfile',status='unknown',RECL=(maxd+1)*16)
c     * STATUS='SCRATCH',RECL=(maxd+1)*16)
c
       do j=0,maxd2
        pnm1(j)=d0
       end do
      end if
c
c k counts the parallels.
      do k=1,n1
c ikp is type of observation (0, 10, 11, 12, 13, 15) and i+ih is the power
c of (R/r) with which the Legendre function of degree i has to be multiplied.
c
       ikp=iko(k)
       ih=iha(ikp)
c
       if (ltest.and.(.not.lmean)) then
        do j=1,maxd2
         write(2,rec=(k-1)*maxd2+j)pnm1
        end do
       end if
c
       if (lsphap) then
        rlatc=rlat0(k,ks)
        s=REM/(REM+hh(k))
       else
        rlatc=rlat(k,ks)
        if (lf) write(*,902)k,ks,ra(k,ks)
 902    format(' k,ks,ra ',2i4,f11.1)
c change 2005-05-10, RE -> A. and back d. 13
c       s=A/ra(k,ks)
        s=RE/ra(k,ks)
       end if
       if (s.gt.d2.or.s.lt.d0) then
        write(*,*)' error in height, s= ',s
        nwarn=nwarn+1
       end if
c
       if (ltest.and.(.not.lmean)) then
        si=s**(ih-1)
        do i=0,maxd2-1
         si=si*s
         if (i.eq.0) then
          pnm(0)=d1
         else
          call lecurs(pnm,i,pi/d2-rlatc)
         end if
c
c j is the order.
         do j=0,i
          read(2,rec=(k-1)*maxd2+j+1)pnm1
c         if (ltest) write(*,29)i,j,(pnm1(i0),i0=0,4)
          pnm(j)=si*pnm(j)*(-1)**j
```

```
          if (j.gt.0) pnm(j)=pnm(j)*sq2
          pnm1(i)=pnm(j)
          write(2,rec=(k-1)*maxd2+j+1)pnm1
c         if (ltest) write(*,29)i,j,(pnm1(i0),i0=0,4)
c 29      format(2i3,5f8.5)
         end do
        end do
       end if
      end do
c
c chort checks for othogonality of cosine/sine series.
      if (lf) call chort(maxgrid,n,rlon)
      if (n.ge.maxgrid.and.ltest) STOP
c
      write(*,*)
      write(*,*)' Observations '
c
c initializing observation array.
      do k=1,n1
       do j=1,n
        g(j,k)=d0
       end do
      end do
c
c ------------------ INPUT (11) ----------------------------------
      write(*,*)' Input observations from file(s) ? (T/F) '
      read(*,*)loinp
      if (loinp) then
       do k=1,n1
        do j=1,n
         g(j,k)=9999.9d0
        end do
       end do
c ------------------ INPUT (11.1.1) ----------------------------------
       write(*,*)' Input number of files '
       read(*,*)nofile
       do i=1,nofile
c ------------------ INPUT (11.1.2) ----------------------------------
        write(*,*)' Input file name '
        read(*,'(a)')obsfil
        open(9,file=obsfil)
        write(*,*)' obs. input from ', obsfil
c
c ------------------ INPUT (11.1.3) ----------------------------------
        write(*,*)
     *  ' Input number of data items and data item to be used '
        read(*,*)ndata,idata
c
c ------------------ INPUT (11.1.4) ----------------------------------
        write(*,*)
     *  ' Are coordinates geodetic or geocentric (T/F) '
        read(*,*)lgeod
        write(*,*)lgeod
        if (ndata.gt.10) then
         write(*,*)' number of data items too large ! ',ndata
         stop
        end if
        if (.not.lgeod) write(*,*)' Geocentric coordinates expected '
c
c ------------------ INPUT (11.1.4.1) ----------------------------------
        write(*,*)' Input data kind identifier '
        read(*,*)ikp
        if (ikp.lt.0.or.ikp.gt.23) then
         write(*,*)' illegal data kind ',ikp
         stop
        end if
        k=0
        sdat=d0
        ssdat=d0
```



```
  63    read(9,*,end=64)num,rlati,rloni,hinp,(dat0(i1),i1=1,ndata)
        if (rlati.lt.-90.0d0.or.rlati.gt.90.0d0) then
         write(*,*)' wrong latitude ',rlati
         stop
        end if
C added 2006-02-06 to check height or radial distance:
        if (lgeod) then
         if (hinp.lt.-20000.0d0.or.hinp.gt.1000000.0d0) then
          write(*,*)' h must be wrong ',hinp
          stop
         end if
        else
         if (hinp.lt.6300000.0d0.or.hinp.gt.7300000.0d0) then
          write(*,*)' r must be wrong ',hinp
          stop
         end if
        end if
c
        ki0=1
c correction 2004-02-03.
        if (rloni.lt.d0) rloni=rloni+360.0d0
        datin=dat0(idata)
c added 2003-01-31.
        if (.not.lgeod) then
c conversion to geodetic coordinates.
c here hinp is the radius vector length.
         x=hinp*cos(rlati/raddeg)
         y=d0
         z=hinp*sin(rlati/raddeg)
c conversion to geodetic latitude and ellipsoidal height.
         call TRANS(rlati,rlatg,h)
         if (abs(rlatg*raddeg-rlati).gt.0.03d0.and.hinp.lt.1.0d3)
     *   write(*,*)' new lat ',rlatg*raddeg,rlati
c now h is the ellipsoidal height.
         rlati=rlatg*raddeg
        end if
        lchlis=lt
  59    if (abs(rlati-rlat0(ki0,0)*radeg).lt.1.0d-4.and.
     *  ikp.eq.iko(ki0)) then
         ki=ki0
        else
         ki0=ki0+1
         if (ki0.gt.n1) then
          if (lchlis.and.abs(rlatold-rlati).gt.10.0) write(*,68)rlati
  68      format(' Latitude not in list ',f10.5)
          lchlis=lf
          rlatold=rlati
          nchlis=nchlis+1
          go to  63
         end if
         go to 59
        end if
c
c added 2004-02-08.
        kj=rloni/dlod+1
        if (abs(rloni-rlon(kj)*raddeg).gt.1.0d-5) then
         write(*,*)' Longitude grid error ',kj,rloni,rlon(kj)
         stop
        end if
        sdat=sdat+datin
        ssdat=ssdat+datin**2
        g(kj,ki)=datin
        k=k+1
        go to 63
c
  64    ssdat=(ssdat-sdat**2/k)/(k-1)
        if (ssdat.gt.d0) ssdat=sqrt(ssdat)
        sdat=sdat/k
        write(*,69)k,obsfil,sdat,ssdat
```

```
  69    format(i6,' observations input  from file ',a72,
     *  /,', mean= ',d12.4,', stdv.=',d12.4)
        close(9)
       end do
c       if (nchlis.gt.0) write(*,*)nchlis,' observations not used '
c       write(*,6000)((g(k,k1),k=1,n),k1=1,n1)
c6000   format(8f9.5)
c this is to insure that all values are initialized. 2004-02-04.
        do k=1,n1
         do j=1,n
          if (g(j,k).gt.9999.0d0) then
           write(*,6001)j,k,g(j,k)
 6001      format(' error in g(',i5,i5,')=',f10.2)
c stop statement included 2005-06-12.
            stop
          end if
         end do
        end do
      else
c
c initializing variables used to check Legendre functions.
      idm=-1
      jdm=-1
      dycmax=d0
c
c Generation of test data.
      write(*,*)' Test data will be generated by program. '
      do k= 1,n1
       nstep=1
       if (lmean1(k)) then
        nstep=nstepi
c calculating normalized area of mean value block.
        rj=d0
        do ks=1,nstep
         rj=rj+clat(k,ks)
        end do
       end if
       ikp=iko(k)
       fac=fc(ikp)
       ih=iha(ikp)
c
       do ks=1,nstep
        if (lf) write(*,*)' k,ikp,fac,ih,ra(k,ks) ',
     *  k,ikp,fac,ih,ra(k,ks)
C change 2005-05-10, re -> A    and back d. 13
c       s=A/ra(k,ks)
        s=RE/ra(k,ks)
        si=s**(ih-1)
c this may not be correct ! spharm multiplies with s !!
        iolda=-1
        jolda=-1
        iiold=-1
        jold=-1
        pim0=D1
        pnmr(k,2,ks)=d1
        pnmr(k,21,ks)=d1
        k0=0
c j is the order.
        do j=0,ncof
         if (ltest.and.(.not.lmean)) read(2,rec=(k-1)*maxd2+j+1)pnm1
         if (lf) write(*,*)' j,pnm1 ',j,(pnm1(i0),i0=0,4)
c i is the degree.
         do i=j,ncof
          im=i
          jm=j
          ycc=spharm0(slat(k,ks),clat(k,ks),ra(k,ks),i,j,lt,
     *    pnmr(k,1,ks),pnmr(k,2,ks),pnmr(k,3,ks),pnmr(k,4,ks),
     *    iolda,jolda)
c idif is the order of differentiation.
```



```
          idif=iha(ikp)-1
          call spharm(slat(k,ks),clat(k,ks),d0,d1,ra(k,ks),im,jm,
     *    idif,.true.,.false.,pnmr(k,20,ks),pnmr(k,21,ks),pnmr(k,22,ks),
     *    pnmr(k,23,ks),
     *    pnmr(k,6,ks),pnmr(k,7,ks),pnmr(k,8,ks),pnmr(k,9,ks),
     *    pnmr(k,10,ks),pnmr(k,11,ks),pnmr(k,12,ks),pnmr(k,13,ks),
     *    pnmr(k,14,ks),pnmr(k,15,ks),pnmr(k,16,ks),pnmr(k,17,ks),
     *    pnmr(k,18,ks),pnmr(k,19,ks))
          if (abs(yc1-ycc).gt.1.0d-8) write(*,191)k,ks,jm,jold,yc1,ycc
 190      format(' spharm ',5i3,2d15.6)
 191      format(' spharm1 ',4i3,2d15.6)
          if (idif.gt.0) then
           if (ikp.eq.15) then
c           yc1=ycc*(i+d1)*(i+d2)*si
            yc1=ycc*(i+d1)*(i+d2)*(a/ra(k,ks))**(ih-1)
            if (abs(yc1-ddc(3,3)).gt.1.0d-8) write(*,191)k,ks,jm,
     *      jold,yc1,ddc(3,3)
           end if
           if (ikp.eq.12) then
            yc1=-ycc*(i+d1)*si
            if (abs(yc1-gc1(3)).gt.1.0d-8) write(*,191)k,ks,jm,jold,
     *      yc1,gc1(3)
           end if
          end if
          ycc=ycc*si
          if (ikp.eq.16)ycc=-gc1(2)
          if (ikp.eq.20)ycc=ddc(2,3)
          if (ikp.eq.22)ycc=ddc(2,2)
          if (ikp.eq.24)ycc=ddc(1,1)
          if (.not.lsphap.and.ltest.and.(.not.lmean)) then
           yc=pnm1(i)*root(2*i+1)
c control of agreement between spharm and lecurs results.
           dyc=yc-ycc
           ddyc=abs(yc)
           if (abs(dyc).gt.1.0d-07*ddyc.and.ddyc.gt.1.0d-10
     *     .and.(.not.linico)) then
            write(*,*)' error in Pij ',i,j,yc,ycc
            nwarn=nwarn+1
            if (abs(dycmax).lt.abs(dyc)) then
             dycmax=dyc
             idm=i
             jdm=j
            end if
           end if
          end if
          if (ikp.eq.11.or.ikp.eq.10.or.ikp.eq.16.or.ikp.eq.20
     *    .or.ikp.eq.22.or.ikp.eq.24) ycc=ycc*fac
          if (ikp.eq.12) ycc=ycc*fac*(i+d1)
          if (ikp.eq.13) ycc=ycc*fac*(i-d1)
          if (ikp.eq.15) ycc=ycc*fac*(i+d1)*(i+d2)
          if (lf) write(*,*)' ycc ',ycc,i,ikp
          k0=k0+1
          do j1=1,n
           if (j.eq.0) then
            k2=i**2+1
            gjk=ycc*cof(k2)
           else
            k2=i**2+2*j
            gjk=ycc*(cos(j*rlon(j1))*cof(k2)+
     *          sin(j*rlon(j1))*cof(k2+1))
           end if
           if (lmean1(k)) then
            dlfac=d0
            do kt=1,nstepi
c error detected 2001-07-28.
             dlfac=dlfac+cos(j*(ddlon*kt-sdlon))
            end do
            if (lf.and.k.eq.1.and.j1.eq.1) write(*,*)i,j,gjk,dlfac,
     *      g(j1,k)
```

```
            g(j1,k)=g(j1,k)+gjk*clat(k,ks)*dlfac/(rj*nstepi)
           else
            if (lf.and.k.eq.1.and.j1.eq.1) write(*,*)i,j,gjk,g(j1,k)
            g(j1,k)=g(j1,k)+gjk
           end if
          end do
         end do
        end do
       end do
       if (ltest) then
        write(*,10)k,(g(j1,k),j1=1,n)
 10     format(' row no ',i3,/,50(6d12.4,/))
       end if
       k3=k0
       if (k0.gt.24) k3=24
c
c converting into spherical approximation.
       if (lsphap) then
        clat(k,ks)=cos(rlat0(k,ks))
        slat(k,ks)=sin(rlat0(k,ks))
        ra(k,ks)=REM+hh(k)
        fc(10)=(GM/REM)
C converting to m**2/s**2
        fc(12)=GM/REM**2*1.0d5
        fc(16)=GM/REM**2*1.0d5
        fc(13)=GM/REM**2*1.0d5
c converting to mgal
        fc(11)=RE
c converting to m.
        fc(15)=1.0d9*GM/(REM**3)
        fc(20)=1.0d9*GM/(REM**3)
        fc(22)=1.0d9*GM/(REM**3)
        fc(24)=1.0d9*GM/(REM**3)
c converting to EU.
       end if
      end do
      ks=1
c
      if (.not.lsphap.and.(idm.ge.0.or.jdm.ge.0)) then
       write(*,*)' maximal difference between Pij: i,j,dif= ',
     * idm,jdm, dycmax
      end if
c
      end if
c
      if (.not.loinp) then
c ------------------ INPUT (12) ----------------------------------
       write(*,*)' Output of observations to file(s) (T/F) '
       read(*,*)loout
       if (loout) then
c data of each kind are output to separate files.
        write(*,*)' Input ',ikn,' file name(s) '
        do m=1,ikn
c ------------------ INPUT (12.1) ----------------------------------
         read(*,'(a)')obsfil
         open(19,file=obsfil)
         do k=1,n1
          rlat1=rlat0(k,0)*radeg
          ikp=iko(k)
          if (ikp.eq.ikt(m)) then
           do j=1,n
c output coordinates are geodetic latitude, longitude and ellipsoidal
c height.
            rlon1=(rlonmin+(j-1)*dlon)*radeg
            i=(k-1)*n1+j
            write(19,58)i,rlat1,rlon1,hh(k),g(j,k)
c format change i6 -> i8 2005-07-21.
   58       format(i8,2f10.4,f10.1,d13.5)
           end do
```



```
          end if
         end do
         write(*,*)' Observations of type' ,ikt(m),
     *    ' output to file ',obsfil
         close(19)
        end do
       end if
      end if
C
      i=0
      do k=1,n1
       do j=1,n
        i=i+1
        if (i.le.maxobs) then
         rh(i)=g(j,k)
        else
         write(*,*)i,' obs exceeds max,',maxobs
        end if
       end do
      end do
c
      do m=1,ikn
       ikp=ikt(m)
       s=d0
       ss=d0
       i=0
       do k=1,n1
        if (ikp.eq.iko(k)) then
         do j=1,n
          s=s+g(j,k)
          ss=ss+g(j,k)**2
          i=i+1
         end do
        end if
       end do
       write(*,73)i,ikp,s/i,sqrt((ss-s**2/i)/(i-1)),ss/i
   73  format(' mean, stdv. and variance of',i6,
     * ' observations of type ',i3,' = ',/,3d15.5)
      end do
c
      write(*,*)' Observation calculation/input completed '
      call fdate(udate)
      write(*,*)udate
      if (lsphap) then
       RE=REM
      end if
      ikold=-1
      ik1old=-1
c
      lnocc=n*n1.gt.maxgrid
      loutcc=lt
      if (lnocc) then
       write(*,*)' no full collocation solution, n*n1= '
     * ,n*n1
      else
       write(*,*)' Full collocation solution, n*n1= '
     * ,n*n1
      end if
      if (lnocc) go to 1000
c
c setting up the collocation normal-equations with a full matrix.
      K0=1
      i=1
      j=1
      do k=1,n1*n
       if (ltest.and.mod(k,400).eq.0) write(*,*)' k=column no. ',k
       ik=iko(i)
       ih=iha(ik)
       fac=fc(ik)
```

```
c
       clati=clat(i,1)
       slati=slat(i,1)
       rlonj=rlon(j)
       rai=ra(i,1)
       s=A/rai
c      s=RE/rai
       if (linico) THEN
        if (ik.eq.10) then
         KIX(6)=1
        else
         KIX(6)=ik-10
        end if
        CRX(4)=slati
        CRX(6)=clati
        HP=rai-RE
        CRX(2)=HP
        if (ik.eq.11) then
         CRX(10)=GM/rai**2
        else
         CRX(10)=d1
        end if
       end if
c
       j1=1
       m=1
       do k1=1,k
        ik1=iko(j1)
        ih1=iha(ik1)
        fac1=fc(ik1)
c
        if (ik.ne.ikold.or.ik1.ne.ik1old) then
         if (linico) then
          if (ik1.eq.10) then
           KIX(7)=1
          else
           KIX(7)=ik1-10
          end if
          CALL COVBX(sm,lt,0)
         else
c change unitless potential degree-variances to degee-variances
c of actual data-combination.
          ss=d0
          do i1=0,maxdeg
           fac2=d1
           if (ik.eq.12) fac2=fac2*(i1+d1)
           if (ik.eq.13) fac2=fac2*(i1-d1)
           if (ik1.eq.12) fac2=fac2*(i1+d1)
           if (ik1.eq.13) fac2=fac2*(i1-d1)
           if (ik.eq.15) fac2=fac2*(i1+d1)*(i1+d2)
           if (ik1.eq.15) fac2=fac2*(i1+d1)*(i1+d2)
           degv(i1+1)=deg0(i1+1)*fac*fac1*fac2
           ss=ss+degv(i1+1)
          end do
          if (ltest) then
           write(*,70)ik,ik1,ss
 70        format(' degree-variances changed, type= ',2i4,
     *      ', new variance= ',f12.6)
           write(*,30)(degv(j4),j4=1,24)
          end if
c
         end if
         ikold=ik
         ik1old=ik1
        end if
c
        if (linico) then
         if (lmean1(i).or.lmean1(j1)) then
          n1s=n1/2
```



```
          ltestc=ltest.and.icout.lt.2
          c(k0)=ccmean(dlon,maxdeg,ih,ih1,i,j,j1,m,n,ltestc,lt,lsphap,
     *     sm)
          icout=icout+1
         else
          clatj=clat(j1,1)
          slatj=slat(j1,1)
          rai=ra(j1,1)
c change 2004-08-13.
          if (rai.lt.RB) then
           write(*,*)' Radius < Bj.sphere rad. ',rai,rb,' stop '
           stop
          end if
          HQ=rai-RE
          CRX(3)=HQ
          CRX(7)=clatj
          CRX(5)=slatj
          rlonm=rlon(m)
          CRX(8)=sin(rlonj-rlonm)
          CRX(9)=cos(rlonj-rlonm)
          if (ltest) then
           if (abs(cosi(mod(abs(j-m),n))-cos(rlonj-rlonm))
     *     .gt.1.0d-10) then
            nwarn=nwarn+1
            write(*,*)' warning ',j,m
           end if
          end if
          t=slati*slatj+clati*clatj*cosi(mod(abs(j-m),n))
          if (abs(t-1.0d0).lt.1.0d-6.and.k.gt.55.and.k.lt.60)
     *    itcoun=1
          CRX(1)=t
          if (ik1.eq.11) then
           CRX(11)=GM/rai**2
          else
           CRX(11)=d1
          end if
c change 2005-05-17.
          CALL COVCX(sm,c(k0),0,lt,gc,lsphap)
          if (k1.eq.1.and.k.lt.-40.or.abs(c(k0)).gt.1.0d5)
     *    write(*,*)' k,kix67,t,cov ',k,KIX(6),KIX(7),t,c(k0)
         end if
        else
         if (lmean1(i).or.lmean1(j1)) then
          n1s=n1/2
          ltestc=ltest.and.icout.lt.2
          c(k0)=ccmean(dlon,maxdeg,ih,ih1,i,j,j1,m,n,ltestc,lf,lsphap,
     *     sm)
          icout=icout+1
         else
          clatj=clat(j1,1)
          slatj=slat(j1,1)
          rlonm=rlon(m)
          if (lt) then
           if (abs(cosi(mod(abs(j-m),n))-cos(rlonj-rlonm))
     *     .gt.1.0d-10) then
            nwarn=nwarn+1
            write(*,*)' warning ',j,m,mod(abs(j-m),n),
     *      cosi(mod(abs(j-m),n)),cos(rlonj-rlonm)
           end if
          end if
          t=slati*slatj+clati*clatj*cosi(mod(abs(j-m),n))
c         t=slati*slatj+clati*clatj*cos(rlonj-rlonm)
          if (lf) write(*,60)i,j,j1-1,m,t
  60      format(4i3,f10.5)
c WARNING  no Bjerhammar-sphere radius in use !!
c         s1=RE/ra(j1,ks)
          s1=A/ra(j1,ks)
          c(k0)=cov(maxdeg,degv,t,ih,ih1,s,s1)
         end if
```

```
        end if
c
        ssdegx=d0
c correction 2004-01-23.
        do issd=1,maxdeg
         ssdegx=ssdegx+degv(issd)
        end do
        if (k1.eq.k) then
        c(k0)=c(k0)+sno(j1)**2
        if (lf) write(*,72)k0,c(k0),sno(j1)
  72    format(' k0,c,sno ',i4,2d15.7)
c       write(*,*)' ssdeg= ',ssdegx
        end if
        if (m.eq.n) then
         m=1
         j1=j1+1
        else
         m=m+1
        end if
c
        k0=k0+1
       end do
       if (j.eq.n) then
        j=1
        i=i+1
       else
        j=j+1
       end if
      end do
      if (maxc.lt.2011.or.ltest) then
       write(*,*)' Full NEQ '
       write(*,50)(C(K),k=1,maxc)
  50   format(5d14.6)
      else
       write(*,*)' First 20 terms of NEQ '
       write(*,50)(C(K),k=1,20)
      end if
c
      call PRONLL(c,izero,rh,nn1,var,lt,lt,maxcc,maxobs,maxobs)
      write(*,*)
      if (maxc.lt.2011.or.ltest) then
       write(*,*)' Reduced matrix: '
       write(*,50)(C(K),k=1,20)
      end if
      write(*,95)var,ss
  95  format(' Weighted square sua of obs and sq. sua= ',2d14.5,/)
      if (maxc.lt.2011.or.ltest) then
       write(*,*)' Solutions '
       write(*,50)(rh(k),k=1,20)
      end if
C
C check 1 solution by predicting back in point (1,1)
      itcoun=0
      ik1old=-1
      pred=d0
      k=0
      ikp=iko(1)
      fac=fc(ikp)
      ih=iha(ikp)
      s=A/ra(1,ks)
c     s=RE/ra(1,ks)
      ik1old=-1
      if (linico) then
       if (ikp.eq.10) then
        KIX(6)=1
       else
        KIX(6)=ikp-10
       end if
       clati=clat(1,ks)
```



```
       slati=slat(1,ks)
       rlonj=rlon(1)
       rai=ra(1,ks)
       CRX(4)=slati
       CRX(6)=clati
       HP=rai-RE
       CRX(2)=HP
       if (ikp.eq.11) then
        CRX(10)=GM/rai**2
       else
        CRX(10)=d1
       end if
      end if
      do i=1,n1
       ik1=iko(i)
       fac1=fc(ik1)
       ih1=iha(ik1)
       s1=A/ra(i,ks)
c      s1=RE/ra(i,ks)
       do j=1,n
        k=k+1
c
        if (ik1.ne.ik1old) then
         if (linico) then
          if (ik1.eq.10) then
           KIX(7)=1
          else
           KIX(7)=ik1-10
          end if
          CALL COVBX(sm,lt,0)
         else
c change unitless potential degree-variances to degee-variances
c of actual data-combination.
          do i1=0,maxdeg
c WARNING  no Bjerhammar-sphere radius in use !!
           fac2=d1
           if (ikp.eq.12) fac2=fac2*(i1+d1)
           if (ikp.eq.13) fac2=fac2*(i1-d1)
           if (ik1.eq.12) fac2=fac2*(i1+d1)
           if (ik1.eq.13) fac2=fac2*(i1-d1)
           if (ikp.eq.15) fac2=fac2*(i1+d1)*(i1+d2)
           if (ik1.eq.15) fac2=fac2*(i1+d1)*(i1+d2)
           degv(i1+1)=deg0(i1+1)*fac*fac1*fac2
           if (lf) write(*,*)' degv ',i1,degv(i1+1)
          end do
         end if
         ik1old=ik1
        end if
c
        if (linico) then
         if (lmean1(1).or.lmean1(i)) then
          c0=ccmean(dlon,maxdeg,ih,ih1,1,1,i,j,n,lf,lt,lsphap,sm)
         else
          clatj=clat(i,1)
          slatj=slat(i,1)
          rai=ra(i,1)
          HQ=rai-RE
          CRX(3)=HQ
          CRX(7)=clatj
          CRX(5)=slatj
          rlonm=rlon(j)
c         CRX(8)=sin(rlonj-rlonm)
c         CRX(9)=cos(rlonj-rlonm)
          CRX(8)=sini(mod(abs(j-m),n))
          CRX(9)=cosi(mod(abs(j-m),n))
c         t=slati*slatj+clati*clatj*cos(rlonj-rlonm)
          t=slati*slatj+clati*clatj*CRX(9)
          CRX(1)=t
          if (ik1.eq.11) then
```

```
           CRX(11)=GM/rai**2
          else
           CRX(11)=d1
          end if
          CALL COVCX(sm,c0,0,lt,gc,lsphap)
c         if (k1.eq.1.and.k.lt.-40.or.abs(c(k0)).gt.1.0d5)
c    *    write(*,*)' call2: k,kix67,t,cov ',k,KIX(6),KIX(7),t,c(k0)
         end if
        else
         if (lmean1(1).or.lmean1(i)) then
          c0=ccmean(dlon,maxdeg,ih,ih1,1,1,i,j,n,lf,lf,lsphap,sm)
         else
          t=slat(1,1)*slat(i,1)+clat(1,1)*clat(i,1)
     *    *cosi(j-1)
c    *    *cos(rlon(1)-rlon(j))
          c0=cov(maxdeg,degv,t,ih,ih1,s,s1)
         end if
        end if
        if (k.lt.-40) write(*,79)k,i,j,t,c0
  79    format(3i3,2d12.5)
        pred=pred+c0*rh(k)
       end do
      end do
      if (abs(pred-g(1,1)).gt.(1.0d-8+sno(1))) then
       write(*,65)pred,g(1,1),pred-g(1,1)
  65   format(' Error in prediction in point 1: ',3d12.5,/)
       nwarn=nwarn+1
      else
       write(*,*)' Check with prediction in 1 point OK '
      end if
      icout=0
c
      write(*,*)' Full collocation solution completed '
      call fdate(udate)
      write(*,*)udate
C prediction of coefficients
      write(*,*)
     *' Prediction of coefficients and their error-estimates.'
      if (ltest) write(*,*)
     *' degree order predicted  observed      error '
c
      ncof=maxpre
      m1n2=1
c
c initializing for call of spharm0.
      do k=1,n1
       do ks=1,nstep
        pnmr(k,2,ks)=d1
        pnmr(k,21,ks)=d1
        nolda(k,ks)=-1
        molda(k,ks)=-1
        noldb(k,ks)=-1
        moldb(k,ks)=-1
       end do
      end do
c
      do m1=0,ncof
c
       do n2=m1,ncof
        if (linico) then
c change 2005-05-19.
c        sn2=(rb/a)**(2*n2+2)
         sn2=(REM/a)**(2*n2+2)
        else
         sn2=d1
        end if
c WARNING: only correct if one type of observations !
c       degn2=sqrt(degv(n2+1)*deg0(n2+1))
        if (lf) write(*,*)' degn2 ',degn2,n2+1,degv(n2+1),deg0(n2+1)
```



```
        n3=2
        if (m1.eq.0) n3=1
        m=m1
        icout=0
        do n4=1,n3
         if (n4.eq.2) m=-m
C first predicting using (obs*C*-1)*cov(obs,cof).
         pred=d0
         k=0
c loop over all parallels.
         do j=1,n1
          ik=iko(j)
c idif is the order of differentiation.
          idif=iha(ik)-1
          fac=fc(ik)
          ih=iha(ik)
          fac2=d1
          if (.not.linico) then
           if (ik.eq.12) fac2=fac2*(n2+d1)
           if (ik.eq.13) fac2=fac2*(n2-d1)
           if (ik.eq.15) fac2=fac2*((n2+d1)*(n2+d2))
          end if
          degx=degn2
          degn2=deg0(n2+1)*fac*fac2
c         write(*,*)n2,j,ik,fac,fac2,degx,degn2
          if (lf) write(*,*)' degn2 ',degn2,n2+1,degv(n2+1),deg0(n2+1)
          if (ltest.and.(.not.lmean))
     *    read(2,rec=(j-1)*maxd2+m1+1)pnm1
          if (lmean1(j)) then
           nsteps=nstepi
          else
           nsteps=1
          end if
c
          ycs=d0
          if (n4.eq.1) then
           ipk=iko(j)
           ih=iha(ipk)
           rji=d0
           do ks=1,nsteps
c change 2005-05-10, RE -> REM. and back d. 13.
c           s=REM/ra(j,ks)
            s=RE/ra(j,ks)
            si=s**(ih-1)
            rji=rji+clat(j,ks)
            yc=spharm0(slat(j,ks),clat(j,ks),ra(j,ks),n2,m1,lt,
     *      pnmr(j,1,ks),pnmr(j,2,ks),pnmr(j,3,ks),pnmr(j,4,ks)
     *      ,nolda(j,ks),molda(j,ks))
            ycc=yc
            if (lf) write(*,982)n2,m1,ra(j,ks),yc,yc*s/root(2*n2+1)
 982        format(' sph0 ',2i3,3d15.6)
            im=n2
            jm=m1
            iiold=noldb(j,ks)
            jold=moldb(j,ks)
            call spharm(slat(j,ks),clat(j,ks),d0,d1,ra(j,ks),im,jm,
     *      idif,.true.,.false.,pnmr(j,20,ks),pnmr(j,21,ks),
     *      pnmr(j,22,ks),pnmr(j,23,ks),
     *      pnmr(j,6,ks),pnmr(j,7,ks),pnmr(j,8,ks),pnmr(j,9,ks),
     *      pnmr(j,10,ks),pnmr(j,11,ks),pnmr(j,12,ks),pnmr(j,13,ks),
     *      pnmr(j,14,ks),pnmr(j,15,ks),pnmr(j,16,ks),pnmr(j,17,ks),
     *      pnmr(j,18,ks),pnmr(j,19,ks))
            noldb(j,ks)=iiold
            moldb(j,ks)=jold
            if (abs(yc1-ycc).gt.1.0d-8)
     *      write(*,190)ipk,-j,ks,jm,jold,yc1,ycc
            if (idif.gt.0) then
             if (ipk.eq.15) then
c change 2005-05-17.
```

```
              yc1=ycc*(n2+d1)*(n2+d2)*(A/ra(j,ks))**(ih-1)
c             yc1=ycc*(n2+d1)*(n2+d2)*si
              if (abs(yc1-ddc(3,3)).gt.1.0d-8) write(*,190)
     *         ipk,j,-ks,jm,jold,yc1,ddc(3,3)
             end if
             if (ipk.eq.12) then
              yc1=-ycc*(n2+d1)*si
              if (abs(yc1-gc1(3)).gt.1.0d-8)
     *         write(*,190)ipk,j,ks,jm,jold,
     *         yc1,gc1(3)
             end if
            end if
c           if (n2.lt.4.and.m1.lt.4.and.lt)
c    *      write(*,982)n2,m1,ra(j,ks),yc,yc*s/root(2*n2+1)
            if (linico) then
             if (ipk.eq.10.or.ipk.eq.11)  yc=ycc
             if (ipk.eq.12) yc=-gc1(3)
             if (ipk.eq.16) yc=-gc1(2)
             if (ipk.eq.15) yc=ddc(3,3)
             if (ipk.eq.20) yc=ddc(2,3)
             if (ipk.eq.22) yc=ddc(2,2)
             if (ipk.eq.24) yc=ddc(1,1)
             yc=yc/root(2*n2+1)
            else
             yc=yc*si/root(2*n2+1)
            end if
            if (ltest.and.(.not.lmean)) then
             ycc=pnm1(n2)
             if (abs(yc-ycc).gt.1.0d-10) write(*,74)n2,m,ih,s,yc,ycc
   74        format(' error 0 ',3i3,3f12.7)
            end if
            if (lmean1(j)) then
C weighting with respect to cosine latitude for mean values.
             ycs=ycs+yc*clat(j,ks)
             rji=rji+clat(j,ks)
            else
             ycs=yc
            end if
           end do
           pnmr(j,5,1)=ycs
          else
           ycs=pnmr(j,5,1)
          end if
c
c loop over all meridians.
          do i=1,n
           k=k+1
           if (lmean1(j)) then
            dlfac=d0
            do kt=1,nstepi
             dlfac=dlfac+cos(m1*(ddlon*kt-sdlon))
            end do
            nsteps=nstepi
           else
            dlfac=d1
            nsteps=1
           end if
c          if (n2.eq.3.and.m1.eq.3.and.k.lt.6)
c    *     write(*,*)' X ',degn2,n2,pnm1(n2),root(2*n2+1),rlon(i)
c    *       ,yc
            if (m.ge.0) then
c            cc0=pnm1(n2)*cos(m*rlon(i))*degn2/root(2*n2+1)
             if (lmean1(j)) then
C weighting with respect to cosine latitude for mean values.
              cc0=ycs/rji*cos(m*rlon(i))*degn2/root(2*n2+1)*
     *         dlfac/nsteps
             else
              csdi=cosi(mod(abs(m*(i-1)),n))*cosdi(m)
     *        -sini(mod(abs(m*(i-1)),n))*sindi(m)
```



```
              if (abs(csdi-cos(m*rlon(i))).gt.1.0d-10) then
               write(*,*)i,m,csdi,cos(m*rlon(i))
               nwarn=nwarn+1
              end if
              cc0=ycs*csdi*degn2/root(2*n2+1)
             end if
c            if (n2.eq.3.and.m1.eq.3.and.k.lt.6)
c    *       write(*,*)' XX ',j,i,k,ycs,cc0
            else
c            cc0=pnm1(n2)*sin(abs(m)*rlon(i))*degn2/root(2*n2+1)
             if (lmean1(j)) then
C weighting with respect to cosine latitude for mean values.
              cc0=ycs/rji*sin(abs(m)*rlon(i))*degn2/root(2*n2+1)*
     *         dlfac/nsteps
             else
              csdi=sini(mod(abs(m*(i-1)),n))*cosdi(-m)
     *        +cosi(mod(abs(m*(i-1)),n))*sindi(abs(m))
              if (abs(csdi-sin(abs(m)*rlon(i))).gt.1.0d-10) then
               write(*,*)i,m,csdi,sin(m*rlon(i))
               nwarn=nwarn+1
              end if
              cc0=ycs*csdi*degn2/root(2*n2+1)
c             cc0=ycs*sin(abs(m)*rlon(i))*degn2/root(2*n2+1)
             end if
c            if (n2.eq.3.and.m1.eq.3.and.k.lt.6)
c    *        write(*,*)' YY ',j,i,k,ycs,cc0
            end if
c
C change 2006-11-20.
            rc(k)=cc0*sn2
 35         format(' 35 ',4i4,6d12.4)
            pred=pred+cc0*sn2*rh(k)
            if (ltest) write(*,35)k,j,i,m,cc0,rh(k),pred
           end do
          end do
c
c storage of covariances between P(m1,n2) and data on j'th parallel.
         if (lercov.and.n4.ne.2) then
          write(18,rec=m1n2)(pnmr(kc,5,1)*degn2*sn2
     *    /root(2*n2+1),kc=1,n1)
          if (m1.eq.-8) write(*,92)m1n2,pnmr(1,5,1)*degn2*sn2/
     *    root(2*n2+1),m1,n2
 92       format(' y ',i3,d15.6,2i3)
          if (m1n2.gt.50.and.ltest)
     *    write(*,*)' rec ',m1n2,' written to unit 18 '
          m1n2=m1n2+1
         end if
c
c error-estimation.
        call PRONLL(c,izero,rc,nn1,var,lf,lt,maxcc,maxobs,maxobs)
        if (ltest.and.m1.eq.8.and.n2.eq.9) then
         write(*,91)m1,n2,(rc(ia),ia=1,10)
 91     format(' x ',2i3,20(5d11.5,/))
        end if
c predicting using (cov(obs,cof))*C**-1)*obs.
        suaa=d0
        ii=1
        do kb=1,n1
         do ka=1,n
          suaa=suaa+rc(ii)*g(ka,kb)
c         if (abs(rc(ii)).gt.0.1d-40) write(*,91)ka,kb,rc(ii)
          ii=ii+1
         end do
        end do
c
        if (abs(pred-suaa).gt.1.0d-13) then
         write(*,81)pred,suaa
         nwarn=nwarn+1
 81      format(' inconsistency pred,suaa= ',2d15.7)
```

```
        end if
c
        err0=sn2*deg0(n2+1)/(d2*n2+d1)
        err2=err0-var
        if (err2.gt.d0) err2=sqrt(err2)
c
c finding subscript of coefficient in array cof.
        if (m.eq.0) then
         k3=n2**2+1
        else
         if (m.gt.0) then
          k3=n2**2+2*m
         else
          k3=n2**2-2*m+1
         end if
        end if
c
        if (lf) write(*,15)n2,m,pred,cof(k3),err2,sqrt(err0)
  15    format(2i4,4D14.6)
        if (k3.lt.31**2) then
         cofr(k3,1)=pred
         cofr(k3,2)=err2
        end if
c
        if (lercov.and.n3.eq.1) then
         nrec=0
         k3=1
         k4=1
         do i=0,m1
c i is the order
          kc2=ncof
          if (i.eq.m1) kc2=n2-1
          do j=i,kc2
c j is the degree.
           nrec=nrec+1
           read(18,rec=nrec)(pnmk(ii),ii=1,n1)
           if (m1.eq.8.and.nrec.gt.50.and.ltest)
     *     write(*,*)' rec ', nrec,' read '
           if (m1.eq.-8) write(*,94)nrec,pnmk(1),i,j,m1,kc2,n2
 94        format(' z ',i3,d15.6,5i3)
           if (i.eq.0) then
            kc1=1
           else
            kc1=2
           end if
           do kc=1,kc1
            suaa=d0
            ii=1
c kb runs over parallels
            do kb=1,n1
c ka runs over meridians
             do ka=1,n
              if (kc.eq.1) then
               csdi=cosi(mod(abs(i*(ka-1)),n))*cosdi(i)
     *         -sini(mod(abs(i*(ka-1)),n))*sindi(i)
              else
               csdi=sini(mod(abs(i*(ka-1)),n))*cosdi(i)
     *         +cosi(mod(abs(i*(ka-1)),n))*sindi(abs(i))
              end if
              suaa=suaa+rc(ii)*pnmk(kb)*csdi
c             if (i.eq.8.and.j.eq.kc2.and.kb.eq.1.and.k3.le.3.and.
c    *        n2.eq.9.and.m1.eq.8.and.ii.lt.10)
              if ( n2.eq.9.and.m1.eq.8.and.nrec.eq.53.and.lf)
     *        write(*,82)ii,kb,i,ka,rc(ii),pnmk(kb),csdi,suaa
 82           format(4i4,4d14.6)
              ii=ii+1
             end do
            end do
            ecov(k3)=suaa
```



```
            if ( n2.eq.9.and.m1.eq.8.and.nrec.eq.53.and.lf) then
             write(*,*)i,j,k3,suaa
            end if
            k3=k3+1
           end do
          end do
         end do
         if (ltest) then
          write(*,*)n2,m
 84       format(' error-covariances ',2i4)
          write(*,80)(ecov(k),k=1,k3-1)
 80       format(8d10.3)
         end if
c
        end if
        end do
c
       end do
      end do
c
      write(*,*)
     *' Full collocation prediction and error-estimation completed. '
c
      write(*,*)
      call fdate(udate)
      write(*,*)udate
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
 1000 ikold=-1
      ik1old=-1
      icout=0
      cvmin=1.0d10
      cvmax=d0
c
      if (leqsym) then
c if we have symmetry around equator, the covariances will be
c identical for the same symmetric latitude pairs. 2001-07-18.
       n1s=n1/2
       if (n1s*2.ne.n1)n1s=n1s+1
       write(*,*)n1s,' parallels used '
      else
       n1s=n1
      end if
c
c calculate number of files needed for covariance storage.
c added 2005-06-17.
      maxbl=n1*n1s*(nd2+1)/maxbls+1
      if (maxbl.gt.maxfil) then
       write(*,*)' increase maxfil ',maxfil
       stop
      else
       maxrec=maxbls/(nd2+1)-1
       write(*,*)' maximal no. of records per file= ',maxrec
       write(*,*)' total number of records = ',n1*n1s
      end if
c
c If covariances have been computed in a previous run, using the same
c observation points and types, they can be re-used. This can be used
c to repeat the calculations with new data values or new error-estimates.
c ------------------------ INPUT (13) ------------------------
      write(*,*)' Will covariances be input from file (T/F) ?'
      read(*,*)lincov
c ------------------------ INPUT (13.1) ------------------------
      write(*,*)' input name of ',maxbl,' file(s) '
      do nbli=1,maxbl
       read(*,'(a)')covfil(nbli)
       if (lincov) then
        write(*,*)' covariances input from ',covfil(nbli)
```

```
       else
        write(*,*)' covariances will be stored in file ',covfil(nbli)
       end if
c the file will contain the covariances.
       OPEN(30+nbli,ACCESS='DIRECT',FORM='UNFORMATTED',
     * file=covfil(nbli),status='unknown',RECL=(nd2+1)*8)
c    * file=covfil(nbli),status='unknown',RECL=(nd2+1)*16)
c record-length counted in bytes.
      end do
      write(*,*)maxbl,' file(s) opened with recl= ',(nd2+1)*8
      nbl=0
c
      do k=1,n1
c initializing variables for re-use of spharm0.
       do ks=1,nstepi
        pnmr(k,2,ks)=d1
        nolda(k,ks)=-1
        molda(k,ks)=-1
c addition 2003-10-15.
        pnmr(k,21,ks)=d1
        noldb(k,ks)=-1
        moldb(k,ks)=-1
       end do
      end do
c
c variables to register variances initialized.
      do k=0,25
       cv(k)=d0
       sscv(k)=d0
       nc(k)=0
      end do
c
      if (lincov) go to 1600
      do k=1,n1s
       ik=iko(k)
       ih=iha(ik)
       fac=fc(ik)
       s=A/ra(k,1)
c      s=RE/ra(k,1)
       if (linico) THEN
        slati=slat(k,1)
        clati=clat(k,1)
        CRX(4)=slat(k,1)
        CRX(6)=clat(k,1)
        HP=ra(k,1)-RE
        CRX(2)=HP
        if (ik.eq.11) then
         CRX(10)=GM/ra(k,1)**2
        else
         CRX(10)=d1
        end if
       end if
c
       do k1=1,n1
        ik1=iko(k1)
        ih1=iha(ik1)
        fac1=fc(ik1)
c       s1=RE/ra(k1,1)
        s1=A/ra(k1,1)
        do i=0,nd2
         if (ik.ne.ikold.or.ik1.ne.ik1old) then
c change unitless potential degree-variances to degee-variances
c of actual data-combination.
          if (linico) then
           if (ik.eq.10) then
            KIX(6)=1
           else
            KIX(6)=ik-10
           end if
```



```
           if (ik1.eq.10) then
            KIX(7)=1
           else
            KIX(7)=ik1-10
           end if
           CALL COVBX(sm,lt,0)
          else
           if (idvc.lt.3) then
            write(*,*)' degree-variances changed ',ik,ik1,maxdeg
            idvc=idvc+1
           end if
           ss=d0
           do i1=0,maxdeg
            fac2=d1
            if (.not.linico) then
             if (ik.eq.12) fac2=fac2*(i1+d1)
             if (ik.eq.13) fac2=fac2*(i1-d1)
             if (ik1.eq.12) fac2=fac2*(i1+d1)
             if (ik1.eq.13) fac2=fac2*(i1-d1)
             if (ik.eq.15) fac2=fac2*(i1+d1)*(i1+d2)
             if (ik1.eq.15) fac2=fac2*(i1+d1)*(i1+d2)
            end if
            degv(i1+1)=deg0(i1+1)*fac*fac1*fac2
            ss=ss+degv(i1+1)
           end do
          end if
          ikold=ik
          ik1old=ik1
          if ((.not.linico).and.idvc.lt.4) then
           write(*,70)ik,ik1,ss
           write(*,30)(degv(j4),j4=1,24)
           idvc=idvc+1
          end if
         end if
c
         ltestc=ltest.and.leqsym.and.k.eq.n1s.and.k1.eq.n1s.and.i.lt.3
         if (linico) then
          clatj=clat(k1,1)
          slatj=slat(k1,1)
          rai=ra(k1,1)
          HQ=rai-RE
c may be - REM !!
          CRX(3)=HQ
          CRX(7)=clatj
          CRX(5)=slatj
          rlonm=d0
          rlonj=dlon*i
c         CRX(8)=sin(rlonj-rlonm)
c         CRX(9)=cos(rlonj-rlonm)
          CRX(8)=sini(i)
          CRX(9)=cosi(i)
          t=slati*slatj+clati*clatj*CRX(9)
          CRX(1)=t
          if (ik1.eq.11) then
           CRX(11)=GM/rai**2
          else
           CRX(11)=d1
          end if
          CALL COVCX(sm,c0,0,lt,gc,lsphap)
c changed back 2005-05-13. and 14/5 back.
c         CALL COVCX(sm,c0,0,lf,gc,lsphap)
          if (lf) write(*,905)kix(6),kix(7),t,hp,hq,c0
 905      format('cvocx ',2i3,f7.4,2f9.1,d12.3)
          if (k1.eq.1.and.k.lt.-40.or.abs(c0).gt.1.0d5)
c correction 2005-06-16.
     *    write(*,*)' call3: k,kix67,t,cov ',k,KIX(6),KIX(7),t,c0
          if (i.gt.((nd2+1)**2)/2) write(*,*)i
          c(i+1)=c0
          if (lf) write(*,903)s,s1,ih,ih1,t,c(i+1)
```

```
         else
         if (lmean1(k).or.lmean1(k1)) then
c addition 2001-07-25.
          c(i+1)=ccmean(dlon,maxdeg,ih,ih1,k,1,k1,i+1,n,ltestc,lf,
     *     lsphap,sm)
         else
C changed 2001-05-15.
c t is cosine of the spherical distance.
          t=slat(k,1)*slat(k1,1)+clat(k,1)*clat(k1,1)*cosi(i)
          if (ltest.and.i.eq.0.and.ik.eq.1.and.ik1.eq.1)
     *    write(*,903)s,s1,ih,ih1,t
 903      format(' ss1ihih1,t,c ',2f9.6,2i4,f10.6,d15.7)
          if (lf.and.k.eq.1.and.k1.eq.1.and.i.lt.2)
     *    write(*,*)' t ',slat(k,1),slat(k1,1),clat(k,1),
     *    clat(k1,1),cos(dlon*i)
c
          c(i+1)=cov(maxdeg,degv,t,ih,ih1,s,s1)
          if (ltest) write(*,903)s,s1,ih,ih1,t,c(i+1)
         end if
         end if
c
         if (i.eq.0.and.k.eq.k1) then
c creating statistics of data variances.
          nc(ik1)=nc(ik1)+1
          cv(ik1)=cv(ik1)+c(i+1)
          sscv(ik1)=sscv(ik1)+c(i+1)**2
          if (cvmin.gt.c(i+1)) cvmin=c(i+1)
          if (cvmax.lt.c(i+1)) cvmax=c(i+1)
c         write(*,*)'cvmin,max,c ',cvmin,cvmax,c(i+1)
         end if
        end do
c
        if (leqsym.and.k.eq.n1s.and.k1.eq.n1s) then
c output of covariance function close to Equator. 2001-07-25.
         open(16,file='sphcovtab')
         do i5=0,nd2
          write(16,52)i5*dlod,c(i5+1)
  52      format(f8.2,d16.5)
         end do
         close(15)
         write(*,*)' covariances output to sphcovtab '
        end if
c
c change 2005-06-29.
        irec=(k-1)*n1+k1
        nbl=(irec-1)/maxrec
        write(31+nbl,rec=irec-nbl*maxrec)(c(i5+1),i5=0,nd2)
        if (k1.eq.-1) write(*,*)
     *  ' record ',(k-1)*n1+k1,' written to file ',31+nbl
c
        if (lf.and.k.lt.4) write(*,47)k,k1, (c(i5),i5=1,5)
  47    format(' cov1 ',2i4,5f12.6)
c
        nnarr(1)=n
        nnarr(2)=0
        if (ltest) then
c       call fourt(c,nnarr,1,-1,0,wrk,idim2,iwkdim)
        if (lf) write(*,43)k,k1,(c(i),i=1,5)
  43    format(' k,k1,c1-5 ',2i3,/,50(5d14.6,/))
        end if
       end do
       if (ltest) write(*,*)' cov stored ',k
      end do
c
      write(*,*)' Calculation of covariances and storage completed '
      call fdate(udate)
      write(*,*)udate
      write(*,*)
      write(*,*)' Covariance type, number, mean and standard-deviation '
```



```
      do i=0,25
       if (nc(i).gt.0) then
        if (nc(i).gt.2) then
         sscv0=sqrt((sscv(i)-cv(i)**2/nc(i))/(nc(i)-1))
         if (sscv0.gt.1.0d10)write(*,*)' error in sscv0 '
        else
         write(*,*)' too few data ',nc(i)
         sscv0=-1.0
        end if
        write(*,44)i,nc(i),cv(i)/nc(i),sscv0
  44    format(2i4,2d12.4)
       end if
      end do
      write(*,99)cvmin,cvmax
   99 format(' min, max variance= ',2d12.4)
 1600 write(*,*)' '
c
      write(*,*)
     *' Prediction of coefficients using compact formulae.'
c ------------------------ INPUT (14) ------------------------
      write(*,*)
     *' Output coefficients and differences to current output ? (T/F) '
      read(*,*)ldiout
      if (ldiout) then
       write(*,*)
     * ' i  j  prediction  error-est. observed coll.est.  coll.err.'
      end  if
      write(*,*)' Coefficients output to ',ccoff,' format(2i4,4d20.12) '
c
c initializing variables to hold summary of prediction result.
      ncof=maxpre
      do ip=0,ncof
       do iq=1,6
        sdeg(ip,iq)=d0
        ssdeg(ip,iq)=d0
       end do
c initialized to hold max and min of differences obs-pred. and
c error estimates.
       sdeg(ip,7)=1.0d10
       sdeg(ip,8)=-1.0d10
       ssdeg(ip,7)=1.0d10
       ssdeg(ip,8)=-1.0d10
      end do
      nerror=0
c
      do k=1,n1
c initializing variables for re-use of spharm0.
       do ks=1,nstepi
        pnmr(k,2,ks)=d1
        nolda(k,ks)=-1
        molda(k,ks)=-1
c addition 2003-10-15.
        pnmr(k,21,ks)=d1
        noldb(k,ks)=-1
        moldb(k,ks)=-1
       end do
      end do
      iiold=-1
      jold=-1
      ikold=-1
      m1n2=1
c
c m is the order.
      do m=0,ncof
c
       k=0
       nbl=0
c i counts the parallels.
       do i=1,n1
```

```
        if (maxd2.lt.60.and.ltest) read(2,rec=(i-1)*maxd2+m+1)pnm1
        do j=1,i
c change 2001-07-19.
         if (leqsym.and.i.gt.n1s) then
          irec=(n1-i)*n1+n1-j+1
         else
          irec=(i-1)*n1+j
         end if
c
C change 2005-06-29. Change of file if maxrec is exceeded.
         nbl=(irec-1)/maxrec
         read(31+nbl,rec=irec-nbl*maxrec)(covf(j5+1),j5=0,nd2)
         if (j.eq.-1)
     *   write(*,*)' record ',irec,' read from file ',31+nbl
         if (lf) write(*,*)' cov2 ',i,j, (c(i5),i5=1,5)
c        if (m.le.2) write(*,*)' read 3 ',i,j
c evaluating Ref, eq. (36), thereby calculating Gamma-tilde, m, ij.
         g0mij=d0
         do i5=0,n-1
          if (i5.le.nd2) then
           covff=covf(i5+1)
          else
           covff=covf(n-i5+1)
          end if
c         g0mij=g0mij+covff*cos(dlon*i5*m)
          g0mij=g0mij+covff*cosi(mod(abs(i5*m),n))
         end do
         g0mij=d2*g0mij/n
         if (m.eq.0) g0mij=g0mij/d2
c
         c(k+1)=g0mij
         if (i.eq.j) then
          if (m.eq.0) then
c adding noise variance contribution to normal equations, cf. Ref.
c eq. (26).
           c(k+1)=c(k+1)+sno(i)**2/n
          else
           c(k+1)=c(k+1)+d2*sno(i)**2/n
          end if
         end if
         ktest=998*999/2
         if (k.ge.ktest.and.k.lt.ktest+1999) then
          ctest(k-ktest+1)=c(k+1)
         end if
         k=k+1
        end do
       end do
       if (lf) write(*,93)ctest
  93   format(6d12.5)
       if (ltest.and.m.le.1) write(*,90)k,(c(k1),k1=1,5)
  90   format(i6,/,800(6d10.4,/))
       do i=1,n1
        rh(i)=d0
       end do
c
       n11=n1
c one set of equations is solved for each order, m.
c calculating the reduced Cholesky matrix of Ref. eq. (22).
       call PRONLL(c,izero,rh,n11,var,lt,lf,maxcc,maxobs,maxobs)
       if (ltest) then
        write(*,*)' Reduced matrix (1-5) '
        write(*,97)(c(k1),k1=1,5)
       end if
c
       do n2=m,ncof
        if (linico) then
c change 2005-05-19. The factor moves the degree-variances from the
c mean earth sphere to a sphere with radius a.
c        sn2=(rb/a)**(2*n2+2)
```



```
         sn2=(REM/a)**(2*n2+2)
        else
         sn2=d1
        end if
        do i=1,n1
c i counts the parallels.
         if (ltest.and.(.not.lmean)) then
          read(2,rec=(i-1)*maxd2+m+1)pnm1
          ycc=pnm1(n2)
         end if
         ipk=iko(i)
         ih=iha(ipk)
C change (addition) 2003-06-17. Corrected 2003-10-08. (ikp -> ipk).
         if (ipk.ne.ikold.or.ipk.ne.ik1old.or.lt) then
          fac1=fc(ipk)
          fac2=d1
          if (.not.linico) then
c error corrected (ik -> ipk) 2004-02-05.
           if (ipk.eq.12) fac2=fac2*(n2+d1)
           if (ipk.eq.13) fac2=fac2*(n2-d1)
           if (ipk.eq.15) fac2=fac2*((n2+d1)*(n2+d2))
          end if
          degn2=deg0(n2+1)*fac1*fac2
          do i1=0,maxdeg-1
           fac2=d1
           if (ipk.eq.12) fac2=fac2*(i1+d1)**2
           if (ipk.eq.13) fac2=fac2*(i1-d1)**2
           if (ipk.eq.15) fac2=fac2*((i1+d1)*(i1+d2))**2
           degv(i1+1)=deg0(i1+1)*(fac1**2)*fac2
          end do
          ikold=ipk
          ik1old=ipk
         end if
         degx=sqrt(deg0(n2+1)*degv(n2+1))
c        write(*,*)' n2jikfac1fac2degxdegn2 ',
c    *   n2,j,ik,fac1,fac2,degx,degn2
         if (lmean1(i)) then
          nsteps=nstepi
         else
          nsteps=1
         end if
c
c        if (leqsym.and.n2.gt.n1s) then
c preparation for the use of Pnn(sinphi)=Pnm(-sinphi) n even and
c = -Pnm(sinphi) n odd.
c        end if
c
         ycs=d0
         rji=d0
         do ks=1,nsteps
          rji=rji+clat(i,ks)
          yc=spharm0(slat(i,ks),clat(i,ks),ra(i,ks),n2,m,lt,
     *    pnmr(i,1,ks),pnmr(i,2,ks),pnmr(i,3,ks),pnmr(i,4,ks),
     *    nolda(i,ks),molda(i,ks))
c addition 2003-10-15.
          idif=ih-1
          im=n2
          jm=m
          iiold=noldb(i,ks)
          jold=moldb(i,ks)
          call spharm(slat(i,ks),clat(i,ks),d0,d1,ra(i,ks),im,jm,
     *    idif,.true.,.false.,pnmr(i,20,ks),pnmr(i,21,ks),
     *    pnmr(i,22,ks),pnmr(i,23,ks),
     *    pnmr(i,6,ks),pnmr(i,7,ks),pnmr(i,8,ks),pnmr(i,9,ks),
     *    pnmr(i,10,ks),pnmr(i,11,ks),pnmr(i,12,ks),pnmr(i,13,ks),
     *    pnmr(i,14,ks),pnmr(i,15,ks),pnmr(i,16,ks),pnmr(i,17,ks),
     *    pnmr(i,18,ks),pnmr(i,19,ks))
          noldb(i,ks)=iiold
          moldb(i,ks)=jold
```

```
c change 2005-05-10, RE -> A. and a-< rem.nd 16/5.
c         s=REM/ra(i,ks)
c         s=RE/ra(i,ks)
          s=A/ra(i,ks)
c         si=s**ih
c and 2005-05-12. and back 2005-05-13.
c         si=s**(ih-1)*REM/A
          si=s**(ih-1)
          ycc=yc
          if (abs(yc1-ycc).gt.1.0d-8)
     *    write(*,190)ikp,-i,ks,jm,jold,yc1,ycc
          if (idif.gt.0) then
           if (ipk.eq.15) then
            yc1=ycc*(n2+d1)*(n2+d2)*si
            if (abs(yc1-ddc(3,3)).gt.1.0d-8)
     *      write(*,190)ikp,i,-ks,jm,jold,yc1,ddc(3,3)
           end if
           if (ipk.eq.12) then
            yc1=-ycc*(n2+d1)*si
            if (abs(yc1-gc1(3)).gt.1.0d-8)
     *      write(*,190)ikp,i,ks,jm,jold,
     *      yc1,gc1(3)
           end if
          end if
          if (linico) then
           if (ipk.eq.10.or.ipk.eq.11)  yc=yc1
           if (ipk.eq.12) yc=-gc1(3)
           if (ipk.eq.16) yc=-gc1(2)
           if (ipk.eq.15) yc=ddc(3,3)
           if (ipk.eq.20) yc=ddc(2,3)
           if (ipk.eq.22) yc=ddc(2,2)
           if (ipk.eq.24) yc=ddc(1,1)
c change 2005-05-18.
c          yc=yc*(A/REM)
c          yc=yc*(REM/A)**(ih-1)
          else
           yc=yc*si
          end if
c covariance between coefficients and observations, cf. Ref. eq. (6).
          ycs=ycs+yc*clat(i,ks)
         end do
         if (lf.and.(.not.lmean).and.
     *   abs(yc/root(2*n2+1)-ycc).gt.1.0d-10)
     *   write(*,*)' error 1 ',n2,m,yc,ycc
c multiplication with sn2 2003-11-28.
         rh(i)=ycs*sn2*degn2/((d2*n2+d1)*rji)
         rc(i)=rh(i)
        end do
        if (ltest) write(*,97)(rh(k1),k1=1,5)
        if (lercov) then
         if (ltest) write(*,*)' Record ',m1n2,' written to unit 18.'
         write(18,rec=m1n2)(rh(k1),k1=1,n1)
         if (m.eq.-8) write(*,92)m1n2,rh(1),m,n2
         m1n2=m1n2+1
        end if
c
c Cholesky reduction of Ref. eq. (22), right hand side.
       call PRONLL(c,izero,rh,n1,var,lf,lt,maxcc,maxobs,maxobs)
c the quantities delta of Ref. eq. (22) are now stored in the array rh.
        if (lf) then
         write(*,98)n2,m
  98     format(' Solutions delta(',i4,',',i4,')')
         write(*,97)(rh(k1),k1=1,n1)
  97     format(' rh1-5 ',6d10.4)
        end if
c
        m4=2
        if (m.eq.0) m4=1
c
```



```
        fact=d1/n
        if (m.ne.0)  fact=fact*d2
c
        do m3=1,m4
         lsin=m3.eq.2
         sua=d0
         ssua=d0
         do j=1,n1
c j runs over parallels.
          do i=1,n
           if (lsin) then
c           enlon=sin(m*rlon(i))
            enlon=sini(mod(abs(m*(i-1)),n))*cosdi(m)
     *      +cosi(mod(abs(m*(i-1)),n))*sindi(abs(m))
            if (ltest) then
             if (abs(enlon-sin(abs(m)*rlon(i))).gt.1.0d-10) then
              write(*,*)i,m,enlon,sin(m*rlon(i))
              nwarn=nwarn+1
             end if
            end if
           else
c           enlon=cos(m*rlon(i))
            enlon=cosi(mod(abs(m*(i-1)),n))*cosdi(m)
     *      -sini(mod(abs(m*(i-1)),n))*sindi(m)
            if (ltest) then
             if (abs(enlon-cos(m*rlon(i))).gt.1.0d-10) then
              write(*,*)i,m,enlon,cos(m*rlon(i))
              nwarn=nwarn+1
             end if
            end if
           end if
c          if (lmean1(i)) then
c error detected 2003-07-30.
           if (lmean1(j)) then
            dlfac=d0
            do kt=1,nstepi
             dlfac=dlfac+cos(m*(ddlon*kt-sdlon))
            end do
            nsteps=nstepi
           else
            dlfac=d1
            nsteps=1
           end if
c cf. Ref. eq. (23).
           ci=rc(j)*enlon*dlfac/nsteps
           s1=rh(j)*enlon*fact*dlfac/nsteps
           if (ltest.and.i.eq.9.and.nwarn.lt.4.and.n2.eq.m)
     *     write(*,87)j,nsteps,s1,g(i,j),dlfac
  87       format(' jnst, rh*enlon*fact, g(i,j) dlfac ',2i3,3d15.4)
c calculating the sums of Ref. eq. (24) and (25).
           sua=sua+s1*g(i,j)
           ssua=ssua+s1*ci
          end do
         end do
c
         ma=m
         if (lsin) ma=-m
         if (m.eq.0) then
          k3=n2**2+1
          cnm=sua
c cf. Ref. eq. (25).
          error=sn2*deg0(n2+1)/(2*n2+d1)-ssua
c change 2003-11-29 and back 2005-05-10. 17/5 and 18/5.
c         error=deg0(n2+1)/(2*n2+d1)-ssua
          if (error.ge.d0) then
           ecnm=sqrt(error)
          else
           ecnm=error
           if (error.lt.-0.1D-20.and.deg0(n2+1).gt.1.0d-26) then
```

```
            write(*,443) n2,m,ssua,error,sn2
 443        format(' WARNING: error wrong ',2i3,3d15.5)
            nwarn=nwarn+1
           end if
          end if
          snm=d0
          esnm=d0
          write(4,444)n2,m,cnm,snm,ecnm,esnm
 444      format(2i4,4d20.12)
         else
          if (ma.ge.0) then
           k3=n2**2+2*m
           cnm=sua
           error=sn2*deg0(n2+1)/(2*n2+d1)-ssua
c change 2003-11-29 and back 2005-05-10. 18/5.
c          error=deg0(n2+1)/(2*n2+d1)-ssua
           if (error.gt.d0) then
            ecnm=sqrt(error)
           else
            ecnm=error
            if (error.lt.-0.1D-30.and.deg0(n2+1).gt.1.0d-26) then
             write(*,443) n2,m,ssua,error,sn2
             nwarn=nwarn+1
            end if
           end if
          else
           k3=n2**2-2*ma+1
           snm=sua
c change 2003-11-29. and back 2005-05-10. 17/5. 18/5.
           error=sn2*deg0(n2+1)/(2*n2+d1)-ssua
c          error=deg0(n2+1)/(2*n2+d1)-ssua
           if (error.ge.d0) then
            esnm=sqrt(error)
           else
            if (error.lt.-0.1D-30.and.deg0(n2+1).gt.1.0d-26) then
             write(*,443) n2,m,ssua,error,sn2
             nwarn=nwarn+1
            end if
           end if
           write(4,444)n2,-ma,cnm,snm,ecnm,esnm
          end if
         end if
c
         error=sn2*deg0(n2+1)/(2*n2+d1)-ssua
c change 2003-11-29. and back 2005-05-10. 18/5.
c        error=deg0(n2+1)/(2*n2+d1)-ssua
         if (error.gt.0.1d-45) then
          error=sqrt(error)
         else
          if (deg0(n2+1).gt.1.0d-25)
     *    write(*,*)' error2 ',error,deg0(n2+1)/(2*n2+d1),ssua,sn2
          error=d0
         end if
         if (lerout.and.n2.eq.maxpre) ersum(ma)=error
c
         if (k3.lt.31**2.and.(.not.lnocc)) then
          dsum=abs(sua)+abs(cofr(k3,1))
          ddsum=abs(error)+abs(cofr(k3,2))
          if (abs(sua-cofr(k3,1)).gt.dsum*1.0d-4.and.
     *    abs(error-cofr(k3,2)).gt.ddsum*1.0d-4) then
           write(*,14)n2,ma,sua,error,cof(k3),cofr(k3,1),cofr(k3,2)
  14       format(i4,i5,5d13.5)
           nerror=nerror+1
          else
           write(*,16)n2,ma,sua,error,cof(k3)
  16       format(i4,i5,3d13.5, ' agrees ')
          end if
         else
          if (ldiout)
```



```
     *    write(*,14)n2,ma,sua,cof(k3),sua-cof(k3),error
         end if
         cdif=sua-cof(k3)
         sdeg(n2,1)=sdeg(n2,1)+sua
         sdeg(n2,2)=sdeg(n2,2)+cdif
         sdeg(n2,3)=sdeg(n2,3)+error
         ssdeg(n2,1)=ssdeg(n2,1)+sua**2
         ssdeg(n2,2)=ssdeg(n2,2)+cdif**2
         ssdeg(n2,3)=ssdeg(n2,3)+error**2
         if (.not.lnocc) then
          sdeg(n2,4)=sdeg(n2,4)+sua-cofr(k3,1)
          ssdeg(n2,4)=ssdeg(n2,4)+(sua-cofr(k3,1))**2
          sdeg(n2,5)=sdeg(n2,5)+cofr(k3,1)-cof(k3)
          ssdeg(n2,5)=ssdeg(n2,5)+(cofr(k3,1)-cof(k3))**2
         end if
         if (sdeg(n2,7).gt.cdif) sdeg(n2,7)=cdif
         if (sdeg(n2,8).lt.cdif) sdeg(n2,8)=cdif
         if (ssdeg(n2,7).gt.error) ssdeg(n2,7)=error
         if (ssdeg(n2,8).lt.error) ssdeg(n2,8)=error
c
         if (lercov.and.(.not.lsin)) then
          nrec=0
          k4=1
          do i=0,m
c i is the order
           kc2=ncof
           if (i.eq.m) kc2=n2
           do j=i,kc2
c j is the degree.
            nrec=nrec+1
            if (i.eq.m) then
             read(18,rec=nrec)(pnmk(ii),ii=1,n1)
             if ((i.eq.8.or.nrec.gt.50).and.ltest)
     *       write(*,*)' rec ', nrec,' read '
             suaa=d0
c kb runs over parallels, and we use ref. eq. B9, B10 to cancel
c the value of "fact".
             do kb=1,n1
              suaa=suaa+pnmk(kb)*rh(kb)*dlfac/nsteps
             end do
             if (ltest) write(*,85)j,i,suaa
 85          format(2i5,d16.5)
             ecov(k4)=-suaa
             k4=k4+1
            end if
           end do
           if (k4.gt.1) then
            write(20,189)n2,ma
 189        format(2i5)
c change 2005-11-08 error estimate in output.
            write(20,88)(ecov(k),k=1,k4-2),error**2,
     *      sn2*deg0(n2+1)/(d2*n2+d1)
 88         format(6d12.4)
           end if
          end do
         end if
c end error correlations.
c
        end do
c end m3-loop (cos and sin).
       end do
      end do
c
      write(*,*)
      write(*,*)' Comparison observed-predicted '
      if (lnocc) then
       if (lt00) then
        write(*,*)' degree mean      S.D.       S.D.  mean error '
       else
```

```
        write(*,*)' multiplied by 1.0d10 '
        write(*,76)
   76   format(' deg . mean     min     max    S.D. '
     *  ,' mean error min    max    sqrt(dgv) ')
       end if
       nout=3
      else
       write(*,*)
     * '      Compact          Tra        Tra.-Comp.  '
       write(*,71)
   71  format(' deg    mean       S.D.       S.D. ',
     * '    S.D.     mean err.  deg. stdv. ')
       nout=5
      end if
c
      do n2=2,ncof
       do j=1,nout
        ssdeg(n2,j)=((ssdeg(n2,j)-sdeg(n2,j)**2/(2*n2))/(2*n2+1))
        if (ssdeg(n2,j).gt.1.0d-55) ssdeg(n2,j)=sqrt(ssdeg(n2,j))
        sdeg(n2,j)=sdeg(n2,j)/(2*n2+1)
       end do
       cvari=sqrt(deg0(n2+1)/(2*n2+1))
       if (lnocc) then
        if (lt00) then
         write(*,89)n2,sdeg(n2,2),ssdeg(n2,2),sdeg(n2,3),cvari
   89    format(i4,6d12.4)
        else
         d10=1.0d10
         write(*,77)n2,sdeg(n2,2)*d10,sdeg(n2,7)*d10,
     *   sdeg(n2,8)*d10,ssdeg(n2,2)*d10,sdeg(n2,3)*d10,
     *   ssdeg(n2,7)*d10,ssdeg(n2,8)*d10,cvari*d10
   77    format(i4,f8.4,f9.4,5f8.4,f9.4)
        end if
       else
        write(*,89)n2,sdeg(n2,2),ssdeg(n2,2),ssdeg(n2,5),
     *  ssdeg(n2,4),sdeg(n2,3),cvari
       end if
      end do
c
      if (.not.lnocc) write(*,*)' number of disagreements = ',nerror
c
      if (nwarn.gt.0) then
       write(*,*)' number of warnings = ', nwarn
      end if
c
      if (lerout) then
       do k=-ncof,ncof
        write(21,78)k,ersum(k)
   78   format(i5,d16.7)
       end do
       close(21)
      end if
c
      close(2)
      do nbli=1,maxbl
       close(30+nbli)
      end do
      close(4)
      close(15)
      if (lercov) then
       close(20)
       close(18)
      end if
c
      write(*,*)' End of program sphgric '
      call fdate(udate)
      write(*,*)udate
c
      stop
```



```
      end
c
      double precision function cov(maxdeg,degv,t,ih,ih1,s,s1)
c covariance function evaluation.
c maxdeg = maximal degree of Legendre expansion of covariance function.
c degv   = array holding degree-variances.
c t      = cosine (spherical distance)
c ih     = the degree+ih is the power of the square of the ratio
c          between the Bjerhammar sphere and the radial distance.
c r      = the square of the ratio of the Bjerhammar-sphere radius and the
c          radial distance.
C
c     implicit real*8 (a-h,o-z),logical(l)
      implicit none
      logical lf,lt,ltest
      integer maxd,ih,ih1,n,maxdeg,itcoun
      parameter (maxd=1800)
      real*8 D0,D1,D2,D3,D4,D5,RE,RADDEG,PI,GM,s,s1,t,
     *r,s2,st,x2,x1,dn,c1,c2,x0,degv
c
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,RE,RADDEG,PI,GM,itcoun,
     *LTEST,LF,LT
      dimension degv(maxd)
c
      R=s*s1
      S2=R**2
      ST=R*T
      X2=D0
      X1=D0
c using clenshaw summation.
      DO 40, N=MAXDEG,0,-1
       DN=DBLE(N)
       C1=-(D2*DN+D1)/(DN+D1)*ST
       C2=S2*(DN+D1)/(DN+D2)
       X0=-C1*X1-C2*X2+degv(N+1)
       X2=X1
       X1=X0
  40  CONTINUE
      X0=X0*s**ih*s1**ih1
c
      cov=X0
      return
      end
c
      double precision function ccmean(dlon,maxdeg,ih,ih1,i,j,j1,m,n,
     *ltestc,linico,lsphap,sm)
c the function calculates covariances of mean values of equal-area
c blocks with side-length dlon (radians).
c programmed 2001-05-17 by C.C.Tscherning. Last update: 2003-10-23.
c
      implicit none
      integer maxgrid,maxcof,maxd,maxobs,nstepi,nc,izero,i,j1,
     *nsteps,nstept,ks,j,kj,km,m,nm,n,iha,iko,maxdeg,ih,
     *molda,nolda,ih1,kt,nm0,KIX,NX1,NX2,itcoun
c     parameter (maxgrid=4050,maxcof=360,maxd=1800,maxobs=720*360,
c    *nstepi=3)
      parameter (maxgrid=4050,maxcof=360,maxd=1800,
     *maxobs=1500*1000,nstepi=3)
c mstepi is the number of steps in the numerical integration of mean
c values.
c
      logical ltest,ltestc,lt,lf,lmean1,lmean,linico
     *,LOCAL,LSUM,lsphap
      real*8 D0,D1,D2,D3,D4,D5,RE,RADDEG,PI,GM,
     *cofr,rlat,rlat0,clat,slat,rlon,clon,slon,
     *ra,hh,g,covf,c,rh,degv,deg0,sno,rc,pnm,pnm1,sdeg,
     *pnmr,ersum,ssdeg,fc,sscv,cv,cof,sslon,cov,rlonm,cclon,
     *t,cdlon,s1,slatj,clatj,rlonj,s,rai,slati,clati,
```

```
     *c0,rji,ddlonj,sdlonj,ddloni,dlon,sdloni,gc(3),
     *CIX,CRX,SIGMA0,SIGMA,HMAX,CVX,D,HP,HQ,ck0,sm(1201),a
c
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,RE,RADDEG,PI,GM,itcoun,
     *LTEST,LF,LT
      common /ww/cof((maxcof+1)**2),cofr(31**2,2),
     *rlat(maxgrid,nstepi),rlat0(maxgrid,0:nstepi),
     *clat(maxgrid,nstepi),slat(maxgrid,nstepi),rlon(maxgrid),
     *clon(nstepi,maxgrid),slon(nstepi,maxgrid),
     *ra(maxgrid,nstepi),hh(maxgrid),
     *g(maxgrid,maxgrid),
     *covf(maxgrid),c(maxgrid*(maxgrid+1)/2),rh(maxobs),izero(maxgrid),
     *degv(maxd),deg0(maxd),
     *sno(maxobs),iko(maxobs),rc(maxobs),
     *pnm(0:maxcof),pnm1(0:maxcof),sdeg(0:maxcof,8),
     *pnmr(maxgrid,23,nstepi),ersum(-maxcof:maxcof),
     *ssdeg(0:maxcof,8),fc(0:25),sscv(0:25),cv(0:25),nc(0:25),
     *iha(0:25),nolda(maxgrid,nstepi),molda(maxgrid,nstepi),
     *lmean1(maxgrid)
c
      COMMON /CMCOV/CIX(24),CRX(56),SIGMA0(1200),SIGMA(1200),HMAX,
     *CVX(2,2),D(36),KIX(37),NX1,NX2,LOCAL,LSUM
      dimension cclon(maxgrid*nstepi),sslon(maxgrid*nstepi)
      equivalence (clon(1,1),cclon(1))
      equivalence (slon(1,1),sslon(1))
c
      A=6378137.0d0
c
      lmean=lmean1(i).and.lmean1(j1)
      if (lmean1(i)) then
       nsteps=nstepi
       sdloni=dlon*(d1+nstepi)/(d2*nstepi)
       ddloni=dlon/nstepi
      else
       nsteps=1
       sdloni=d0
       ddloni=d0
      end if
      if (lmean1(j1)) then
       nstept=nstepi
       sdlonj=dlon*(d1+nstepi)/(d2*nstepi)
       ddlonj=dlon/nstepi
      else
       nstept=1
       sdlonj=d0
       ddlonj=d0
      end if
      rji=d0
c
      c0=d0
      do ks=1,nsteps
       clati=clat(i,ks)
       slati=slat(i,ks)
       rai=ra(i,ks)
c      write(*,*)' rai ',i,ks,rai
c      s=RE/rai
       s=A/rai
       rlonj=rlon(j)-ddlonj
       if (linico) THEN
        CRX(4)=slati
        CRX(6)=clati
        HP=rai-RE
        CRX(2)=HP
       end if
       do kj=1,nsteps
        rlonj=rlon(j)-sdloni+kj*ddloni
c
        do kt=1,nstept
         clatj=clat(j1,kt)
```



```
         slatj=slat(j1,kt)
         s1=A/ra(j1,kt)
c        s1=RE/ra(j1,kt)
         do km=1,nstept
          if (lmean) then
           nm0=(m-1)*nstept+km-(j-1)*nsteps-kj
           nm=abs(mod(nm0,nstepi*n))+1
           cdlon=cclon(nm)
          else
           rlonm=rlon(m)-sdlonj+km*ddlonj
           cdlon=cos(rlonj-rlonm)
          end if
          t=slati*slatj+clati*clatj*cdlon
          if (linico) then
           HQ=ra(j1,kt)-RE
           CRX(3)=HQ
           CRX(7)=clatj
           CRX(5)=slatj
           rlonm=rlon(m)
           CRX(8)=sin(rlonj-rlonm)
           CRX(9)=cos(rlonj-rlonm)
           CRX(1)=t
           CALL COVCX(sm,ck0,0,lf,gc,lsphap)
           c0=c0+ck0
          else
           if (abs(cdlon-cclon(nm)).gt.1.0d-8.and.
     *     m.lt.3.and.j.lt.3.and.lf) then
            write(*,10)j,kj,m,km,nm0,nm,cclon(nm),cdlon
  10        format(6i4,3f12.5)
           end if
c
           c0=c0+cov(maxdeg,degv,t,ih,ih1,s,s1)*clati*clatj
          end if
c change 2001-07-26 - earlier rji was incorrect.
          rji=rji+clati*clatj
          if (ltestc) write(*,907)kt,km,t,c0
 907      format(' ktmtc0 ',2i3,f10.5,d15.8)
         end do
        end do
c
       end do
      end do
c change 2003-02-16
      ccmean=c0/(rji)
      if (ltestc) write(*,908)i,j,j1,m,ccmean,rji
 908  format(' cmean ',4i3,2d15.8)
      if (abs(ccmean).gt.1.0d5 ) then
       write(*,*)' **** error in ccmean '
       write(*,10)j,kj,m,km,nm0,nm,cclon(nm),cdlon,ccmean
       stop
      end if
      return
      end
C
      SUBROUTINE PRONLL(AN,INUL,H,NT,VAR,LRED,LBS,IANT,INULT,IHT)
C
C     THIS SUBROUTINE USES A CHOLESKY ALGORITHME FOR REDUCING
C     AND SOLVING THE SYSTEM OF LINEAR EQUATIONS
C                (AT*A)*X=AT*Y
C     WHERE (AT*A) IS SYMMETRICAL POSITIV DEFINITE MATRIX OF
C     DIMENSION NT*NT, AND (AT*Y) IS A VECTOR OF DIMENSION NT.
C
C     CONTEND OF ARRAYES:
C             AN(.)         THE UPPER PART OF (AT*A), AND RETURNS
C                           WITH LT, WHERE L*LT=(AT*A), IF LRED =
C                           .TRUE.
C             INUL(.)       INDEX OF THE FIRST NON-ZERO ELEMENT
C                           OF EACH ROW.
C             H(.)          THE RIGHT-HANDSIDE (AT*Y), AND RETURNS
```

```
C                           WITH X ,IF LBS = .TRUE., ELSE WITH
C                           (L-1)*(AT*Y).
C              VAR          THE PSEUDO DIAGONAL ELEMENT OF (L-1)*
C                           (AT*Y).
C
C
C     PROGRAMMED BY
C                       PER KNUDSEN
C                       GEODETIC INST.
C                       DK-2920                       12.07.85.
C
C     IMPLICIT REAL*8 (A-H,O-Z)
      implicit none
      integer iht,inult,nt,ir,irt,inul,iant,is,ist,ii,irr,iit
      real*8 an,sum,h,var
      DIMENSION AN(IANT),INUL(INULT),H(IHT)
      LOGICAL LRED,LBS
C
C***  THE UPPER PART OF A IS REDUCED INTO LT IF LRED IS TRUE.
C
      IF(.NOT.LRED) GO TO 50
      DO 25 IS=1,NT
      IST=(IS*(IS-1))/2
      SUM=0.0D0
      DO 10 IR=INUL(IS),(IS-1)
      IRT=(IR*(IR-1))/2
      SUM=0.0D0
      DO 5 II=MAX0(INUL(IS),INUL(IR)),(IR-1)
      SUM=SUM+(AN(IRT+II)*AN(IST+II))
    5 CONTINUE
      AN(IST+IR)=(AN(IST+IR)-SUM)/AN(IRT+IR)
   10 CONTINUE
      SUM=0.0D0
      DO 15 II=INUL(IS),(IS-1)
      SUM=SUM+AN(IST+II)**2
   15 CONTINUE
c     write(*,*)ist,ii,an(ist+ii),sum
c change 2005-06-19.
      IF ((AN(IST+IS)-sum).LE.1.0d-15) THEN
      WRITE(*,102)IS
  102 FORMAT(' NUMERICAL SINGULARITY IN ROW ',I4)
      write(*,*)ist,is,an(ist+is),sum
c test2005-06-21
      write(*,93)(an(ist+is+II),II=-1999,0)
   93 format(6d12.5)
      NT=IS-1
      AN(IST+IS)=0.0D0
c     stop
      GO TO 50
      else
      AN(IST+IS)=SQRT(AN(IST+IS)-SUM)
      end if
C
   25 CONTINUE
C
C
   50 CONTINUE
C***  SOLVE L-1*H
C
      DO 100 IR=1,NT
      IRT=(IR*(IR-1))/2
      SUM=0.0D0
      DO 90 II=INUL(IR),(IR-1)
      SUM=SUM+(AN(IRT+II)*H(II))
   90 CONTINUE
      H(IR)=(H(IR)-SUM)/AN(IRT+IR)
  100 CONTINUE
      SUM=0.0D0
      DO 101 II=1,NT
```



```
      SUM=SUM+H(II)**2
  101 CONTINUE
      VAR=SUM
C
C***  THE SOLUTION IS FOUND BY BACK SUBSTITUTUION IF LBS IS TRUE.
C
      IF(.NOT.LBS) RETURN
C
      DO 150 IRR=1,NT
      IR=NT+1-IRR
      IRT=(IR*(IR-1))/2
      SUM=0.0D0
      DO 140 II=(IR+1),NT
      IIT=(II*(II-1))/2
      SUM=SUM+(AN(IIT+IR)*H(II))
  140 CONTINUE
      H(IR)=(H(IR)-SUM)/AN(IRT+IR)
  150 CONTINUE
C
      RETURN
      END
      SUBROUTINE EUCLID(COSLAP,SINLAP,COSLOP,SINLOP,H,E2,AX)
C PROGRAMMED BY C.C.TSCHERNING, GEODETIC INSTITUTE OF DENMARK, 1974.
C UPDATES: 2001-01-25 by cct.
C COMPUTATION OF EUCLIDIAN COORDINATES X,Y,Z , DISTANCE AND SQUARE OF
C DISTANCE FROM Z-AXIS XY, XY2 AND DISTANCE AND SQUARE OF DISTANCE FROM
C THE ORIGIN DISTO AND DIST2 FROM GEODETIC COORDINATES REFERING TO AN
C ELLIPSOID HAVING SEMI-MAJOR AXIS EQUAL TO AX AND SECOND EXCENTRICITY
C E2.
C     IMPLICIT INTEGER(I,J,K,M,N), REAL *8(A-H,O-Z)
      REAL*8 X,Y,Z,XY,XY2,DISTO,DIST2,rlatc,H,E2,AX,COSLAP,SINLAP,
     *COSLOP,SINLOP,DN
      COMMON /EUCL/X,Y,Z,XY,XY2,DISTO,DIST2,rlatc
C X,Y,Z ARE THE CARTESIAN COORDINATES, XY2 SQUARE OF DISTANCE FROM
C Z-AXIS, DISTO DISTANCE FROM ORIGIN, rlatc the geocentric latitude.
      DN = AX/ SQRT(1.0D0-E2*SINLAP**2)
      Z = ((1.0D0-E2)*DN+H)*SINLAP
      XY = (DN+H)*COSLAP
      XY2 = XY*XY
      DIST2 = XY2+Z*Z
      DISTO =  SQRT(DIST2)
      X = XY* COSLOP
      Y = XY* SINLOP
      rlatc=atan2(Z,XY)
      RETURN
      END
c ---------------------------------------------------------------------
      subroutine lecurs(plm,ll,thet)
c --   One dim. recursion of Legendre column degree ll > 0.
c --   Two central half integral columns degree ll-1/2 also calc.
c --   symmetric left and right Edmonds recursion followed by
c --   partial binomial recursion onto Legendre column degree ll,
c --   repeated from column l=1 (initiated directly ) to degree ll,
c --     input :    ll   degree of  Legendre column wanted.
c --                thet colatitude (radians)
c --     output:    plm(m) Legendre value order m=(0:ll)
c --                d(i,1), i=0,2*ll Legendre column d-matrix degree ll
c --                (central column in d*matrix degree ll).
c --                dd(i,0) & dd(i,2), i=0,2*ll-1 degree ll-1/2 central
c --                (central columns in d-matrix degree ll).
c --                d & dd in common, equivalenced.
c --   Program by Torben Risbo, Geophyscal Institute, Copenhagen Univ.
c --   20 Feb 1992, last version 27 Apr 1992. Changed 2001-01-08 by cct.
c     implicit real*8 (a-h,o-z)
      implicit none
      integer maxd,L,LL,I,l2
      real*8 d,dd,plm,sqt,sq2,thet,q,pc,qc,p,fl
      parameter  (maxd=1800)
      dimension d(0:2*maxd,0:2),dd(0:2*maxd,0:2)
```

```
c     dimension plm(0:maxd),sqt(0:maxd*2)
      dimension plm(0:maxd)
c     logical bool
c     save sqt,sq2,bool
      equivalence (d(0,0),dd(0,0))
      common d
      common /rrsphc/sqt(0:maxd*2+2)
c     data bool /.false. /
c     if (bool) goto 5
      sq2 = sqrt(2.0d0)
c     bool = .true.
c     do 1 i=0,maxd*2
c   1 sqt(i)= sqrt(float(i))
c   5 p = sin(thet/2.0d0)
      p = sin(thet/2.0d0)
      q = cos(thet/2.0d0)
      pc= p
      qc= q
      plm(0) = 1.0d0
      if(ll .eq. 0) return
c ---- initialize column l=1 ----
      d(0,1)  =   sq2*q*p
      d(1,1)  =  -p*pc+q*qc
      d(2,1)  =  -sq2*pc*qc
      if (ll .eq. 1) then
      plm(0) = d(1,1)
      plm(1) = d(2,1)
      return
      endif
c -----  main loop --------------
      do 210 l=1,ll-1
      l2 = 2*l
      do 10 i=0,l2+1
      dd(i,0) = 0.0d0
      dd(i,2) = 0.0d0
   10 continue
c --  recursion from d to dd, after Edmonds scheme to left and right.
      do 110 i=0,l2
      dd(i,0)    = dd(i,0)    +    sqt(l2+1-i)/sqt(l+1)*d(i,1)*q
      dd(i+1,0)  = dd(i+1,0)  -    sqt(i+1)/sqt(l+1)*d(i,1)*pc
      dd(i,2)    = dd(i,2)    +    sqt(l2+1-i)/sqt(l+1)*d(i,1)*p
      dd(i+1,2)  = dd(i+1,2)  +    sqt(i+1)/sqt(l+1)*d(i,1)*q
  110 continue
  100 continue
c ---  null d column
      do 30 i=0,l2+2
      d(i,1) = 0.
   30 continue
c --  binom recursion from dd to central column d --
      fl = float(l2+2)
      do 120 i=0,l2+1
      d(i,1)    = d(i,1)   +    sqt(l2+2-i)*sqt(l+1)/fl*dd(i,2)*q
      d(i+1,1)  = d(i+1,1) -    sqt(i+1)*sqt(l+1)/fl*dd(i,2)*pc
      d(i,1)    = d(i,1)   +    sqt(l2+2-i)*sqt(l+1)/fl*dd(i,0)*p
      d(i+1,1)  = d(i+1,1) +    sqt(i+1)*sqt(l+1)/fl*dd(i,0)*qc
  120 continue
  210 continue
c --- output Legendre values in plm(m) (m=0:l) -----
      do 75 i=0,ll
      plm(i) = d(i+ll,1)
   75 continue
      return
      end
c-------------------------------------------------------------
      subroutine chort(maxgrid,n,rlon)
c subroutine to check orthogonality of sin and cosine.
c programmed 2000-12-01 by C.C.Tscherning.
c     implicit real*8 (a-h,o-z)
      implicit none
```



```
      integer k,n,maxgrid,k1,k2,j
      real*8 d0,d1,rlon,scqn,ssqn,sumc,sums,sucs
c rlon holds the longitudes, n the number of points.
      dimension rlon(maxgrid),scqn(1444),ssqn(1444)
c
      d0=0.0d0
      d1=1.0d1
      write(*,*)' k,  cos(k-1*dlon), sin(k-1*dlon) '
      do k=1,n
       write(*,67)k,cos(rlon(k)),sin(rlon(k))
  67   format(i3,2f10.6)
      end do
c
C check ortogonality.
      write(*,*)' i, j, sum c(i*l)*c(j*l), s*s, cos*sin '
      do k=0, n
       do  k1=0,k
        sumc=d0
        sums=d0
        sucs=d0
        do j=1,n
         sumc=sumc+cos(k*rlon(j))*cos(k1*rlon(j))
         sums=sums+sin(k*rlon(j))*sin(k1*rlon(j))
         sucs=sucs+sin(k*rlon(j))*cos(k1*rlon(j))
        end do
        do k2=1,n
         scqn(k2)=d0
         ssqn(k2)=d0
         do j=1,n
          scqn(k2)=
     *    scqn(k2)+cos(k*rlon(j))*cos(k1*(rlon(j)-rlon(k2)))
           ssqn(k2)=
     *    ssqn(k2)+sin(k*rlon(j))*cos(k1*(rlon(j)-rlon(k2)))
         end do
        end do
c
        write(*,76)k,k1,sumc,sums,sucs
 76     format(2i5,3f10.6)
        write(*,77)(scqn(k2),k2=1,n)
        write(*,77)(ssqn(k2),k2=1,n)
 77     format(' sqn ',8f8.4)
       end do
      end do
      return
      end
      double precision function spharm0(slat,clat,r,i0,j0,lfull
     *,pii,pim0,pim1,pim2,iold,jold)
c Calculation of the values of solid spherical harmonic functions
c Y(i0,j0)(lat,lon,r) using recursion based on Y(i0-1,j0), Y(i0-2,j0)
c when j0 .ne. j0. Otherwise the recursion is based on Y(i0-1,j0-1) and
c Y(i0-2,j0-1).
c Reference:
c Tscherning, C.C. and K.Poder: Some Geodetic applications of Clenshaw
c Summation. Bolletino di Geodesia e Scienze Affini, Vol. XLI, no. 4,
c pp. 349-375, 1982.
c
c variables at call: slat, clat: sine and cosine of latitude, r the
c size of the radius vector,
c lfull a logical variable true if fully normalized functions
c are used.
c
c call and return variables:
c pii, pim0, pim1, pim2 values used in last reqursion step for the
c computation of associated Legendre functions.
c at the very first step (i0=j0=0) pim0 := d1.
c
c Last change 2005-05-10 by cct.
c
c     implicit real*8 (a-h,o-z), logical (l)
```

```
      implicit none
      integer maxd,i,j,j1,jold,j0,i0,iold,itcoun
      parameter (maxd=1800)
      logical lfull,LTEST,LF,LT
      real*8 D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GM,pim1,pim2,fact,v,pm,
     *b,slat,a,clat,rq,pim0,pii,r,q,root,sq2,ax
c
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GM,itcoun,
     *LTEST,LF,LT
      common /rrsphc/root(0:maxd*2+2)
c root is a precomputed square root-table (root(0)=0 !).
      sq2=root(2)
c change 2005-05-10, re -> ax. and back d. 13 and 14/5.
      ax=6378137.0d0
c     q=re/r
      q=ax/r
      j=j0
      i=i0
      j1=j+1
      if (i.eq.j) then
       if (j.ne.(jold+1)) write(*,*)' WARNING j, jold ',j,jold
       pim2=d0
       pim1=d0
       if (i.eq.0) then
        pii=q
        pim0=pii
       else
        rq=root(2*i-1)/root(2*i)*q
        pim0=pii*clat*rq
        pii=pim0
       end if
      else
       if (j.ne.jold.or.i.ne.(iold+1)) write(*,*)' WARNING i,j ',i,j
       a=(d2*i-d1)/(root(i+j)*root(i-j))*q
       pim2=pim1
       pim1=pim0
       pim0=a*slat*pim1
       if (i.gt.j) then
        b=-q**2*root(i-j-1)*root(i+j-1)/(root(i-j)*root(i+j))
        pm= b*pim2
        pim0=pim0+pm
       end if
      end if
      v=pim0
      iold=i
      jold=j
      if (lfull) then
c normalisation.
       if (j.eq.0) then
        fact= root(2*i+1)
       else
        fact= root(2*i+1)*sq2
       end if
       v=v*fact
      end if
      spharm0=v
      if (abs(spharm0).gt.1.0d200) write(*,*)
     *' Error  in spharm0 ',i,j,spharm0
c
      return
      end
      SUBROUTINE TRANS(RLATS,RLATP,HP)
C ORIGINAL VERSION PROGRAMMED IN 1974 BY C.C.TSCHERNING, GEODAETISK
C INSTITUT. LATEST UPDATE 2003-01-31.
c added 2003-01-31, changed 2004-02-07.
C
C THE SUBROUTINE TRANSFORMS THE CARTESIAN COORDINATES
C TO GEODETIC LATITUDE AND ELLIPSOIDAL HEIGHT, HP.
C
```



```
      implicit none
      integer n,itcoun
      logical ltest,lf,lt
      real*8 rlatp,rlatc,D0,D1,D2,D3,D4,D5,RE,RADDEG,PI,GM,
     *X,Y,Z,XY,XY2,DISTO,DIST2,ZC,DISTO0,XY0,X0,Y0,Z0,
     *XY20,DIST20,AX2,E22,DH,s,rlat1,cosla,rlat,hp,sinlap,dn,xyc,
     *dist2c,rlatcc,xy2c,distoc,rlats
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,RE,RADDEG,PI,GM,itcoun,
     *LTEST,LF,LT
      COMMON /EUCL/X,Y,Z,XY,XY2,DISTO,DIST2,rlatc
c resulting cartesian coordinates and geocentric latitude.
      COMMON /ITRANC/AX2,E22
      X0=X
      Y0=Y
      Z0=Z
      XY20= X0*X0+Y0*Y0
      XY0 =  SQRT(XY20)
      DIST20 = XY20+Z0*Z0
      DISTO0 =  SQRT(DIST20)
C
C  COMPUTATION OF THE NEW GEODETIC LATITUDE, CF REF(C) PAGE 183.
      SINLAP=SIN(RLATC)
      S = AX2/ SQRT(D1-E22*SINLAP**2)
      N=0
      DH = DISTO-AX2
      RLAT1 = RLATC
      COSLA=COS(RLATC)
   70 RLAT = RLAT1
C
      RLAT1 =  ATAN2(Z0,XY0-E22*S*COSLA)
      COSLA =  COS(RLAT1)
      S = AX2/ SQRT(D1-E22*(D1-COSLA**2))
      DH = XY0/COSLA-S
      n=n+1
      if (n.gt.15) then
       write(*,*)' number of iterations in trans= ',n
       go to 71
      end if
      IF ( ABS(RLAT1-RLAT).GT.1.0D-16) GO TO 70
C
   71  RLATP = RLAT1
      HP=DH
C
      SINLAP=SIN(RLATP)
      DN = AX2/ SQRT(1.0D0-E22*SINLAP**2)
      ZC = ((1.0D0-E22)*DN+DH)*SINLAP
      XYC = (DN+DH)*COSLA
      XY2C = XYC*XYC
      DIST2C = XY2C+ZC*ZC
      DISTOC=  SQRT(DIST2C)
      rlatcc=atan2(ZC,XYC)
      if (abs(DISTOC-DISTO0).GT.1.0) write(*,*)' WARNING ',
     *DISTOC,DISTO0
      if (abs(rlats*pi/180.0d0-rlatcc).gt.1.0d-8) write(*,*)' WARNIN ',
     *rlats,rlatcc*180.0d0/pi
      RETURN
      END
      SUBROUTINE COVAX(SM,IS)
C ORIGINAL VERSION PROGRAMMED JULY 1975 BY C.C.TSCHERNING. LATEST
C MODIFICATION 1999-02-14.
C
C THIS SUBROUTINE PREPARES CONSTANTS USED FOR COVARIANCE FUNCTION EVALU-
C ATION, WHICH IS EXECUTED USING THE SUBROUTINES COVBX AND COVCX.
C
C THE COVARIANCE FUNCTION USED IS DEFINED ACCORDING TO A DEGREE-VARIANCE
C MODEL AND A SET OF EMPIRICAL (POTENTIAL) DEGREE-VARIANCES. THE DEGREE-
C VARIANCE MODEL IS SPECIFIED THROUGH THE VALUES OF KI(1)-KI(5),CI(8)-
C CI(10) AND THE PARAMETERS N1 AND LOCAL OCCURRING IN THE COMMON BLOCK
C /CMCOV/. EMPIRICAL ANOMALY DEGREE-VARIANCES WILL HAVE TO BE STORED IN
```

```
C SIGMA0 WHEN LOCAL IS FALSE, AND ARE USED FOR THE COMPUTATION OF RESI-
C DUAL POTENTIAL DEGREE-VARIANCES, (SEE REF(A), EQ.(16)).
C
C BY THE CALL OF COVAX, THE KIND OF COVARIANCE FUNCTION TO BE USED IS
C DETERMINED. THE VALUE OF KI(5) WILL DETERMINE THE DEGREE-VARI-
C ANCE MODEL (1,2 OR 3, CF.REF(A),EQ.(17)) THAT WILL BE USED. THE QUAN-
C TITIES K(2),K(3) MUST BE STORED IN KI(3),KI(4), AND BE EQUAL TO ZERO
C WHEN NOT USED (EG.,KI(3),KI(4) BOTH ZERO WHEN KI(5)=1). THE QUANTITY
C A(I) MUST BE STORED IN CI(8) IN UNITS OF (M/SEC)**4, AND THE SQUARE OF
C THE RATIO BETWEEN THE RADIUS OF THE BJERHAMMAR-SPHERE (RB) AND THE
C MEAN RADIUS OF THE EARTH (RE) MUST BE STORED IN CI(10).
C
C THERE ARE THEN THREE POSSIBILITIES:
C (1) ONE OF THE DEGREE-VARIANCE MODELS IS USED WITHOUT MODIFICATIONS.
C     THE SUMMATION LIMIT P OF REF.(A),EQ.(20) IS THEN FIXED TO 3.
C     BECAUSE THIS IS EQUIVALENT TO REQUIRING THE FIRST 3 DEGREE-VARIAN-
C     AREA /CMCOV/ MUST BE EQUAL TO 3 AND .TRUE., RESPECTIVELY.
C     CES TO BE ZERO, THE VARIABLES N1 AND LOCAL STORED IN THE COMMON
C (2) A NUMBER (N1) OF THE ANOMALY DEGREE-VARIANCES (DEGREE ZERO TO
C     N1-1) ARE PUT EQUAL TO EMPIRICAL DETERMINED QUANTITIES. THE ANO-
C     MALY DEGREE-VARIANCE OF DEGREE K WILL HAVE TO BE STORED IN
C     SIGMA0(IS+K+1) IN UNITS OF MGAL**2 WHEN CALLING COVAX. LOCAL MUST
C     BE EQUAL TO FALSE. COVAX WILL CONVERT THE ANOMALY DEGRE5-VARIANCES
C     INTO POTENTIAL DEGREE-VARIANCES. THE POINTER IS MUST BE POSITIVE.
C (3) THE N1 FIRST DEGREE-VARIANCES (DEGREE 0 - N1-1) ARE EQUAL TO ZERO.
C     THIS MEANS, THAT THE VALUES OF A (N1-1)-ORDER LOCAL COVARIANCE
C     FUNCTION WILL BE COMPUTED. LOCAL MUST HAVE THE VALUE .TRUE..
C IN ALL CASES N1 MUST BE LESS THAN 300.
C
C THE COVARIANCES WILL GENERALLY BE COMPUTED BY CLOSED EXPRESSIONS, BUT
C THEY MAY IN CERTAIN CASES BE USELESS IN BIG ALTITUDES OF NUMERICAL
C REASONS, CF. REF(A), SECTION 4. IN THEESE CASES MUST THE LOGICAL VARI-
C ABLE LSUM BE TRUE AND THE VARIABLE HMAX MUST HAVE ASSIGNED A VALUE
C EQUAL TO THE CRITICAL ALTITUDE. WHEN LSUM IS TRUE AND THE HEIGHT OF
C P OR Q IS GREATHER THAN HMAX, WILL THE SERIES REF(A), EQ.(16), ABBRE-
C VIATED TO DEGREE N2-1 BE USED FOR THE COMPUTATION OF THE COVARIANCES.
C THE VALUES OF LSUM, N2 AND HMAX WILL (IN THE SAME WAY AS FOR THE PARA-
C METERS SPECIFYING THE DEGREE-VARIANCE MODEL) BE TRANSFERRED TO COVAX
C THROUGH THE COMMON AREA /CMCOV/, BUT AN ARRAY SM IS TRANSFERRED AS A
C PARAMETER IN THE CALL IN ORDER TO ENABLE VARIABLE DIMENSIONING (SPECI-
C FIED BY THE VARIABLE N2 IN /CMCOV/).
C
C THE CALL OF COVAX WILL ALSO INITIALIZE CERTAIN VARIABLES USED IN
C SUBSEQUENT COMPUTATIONS.
C
C REFERENCES:
C (A) TSCHERNING,C.C.: COVARIANCE EXPRESSIONS FOR SECOND AND LOWER ORDER
C     DERIVATIVES OF THE ANOMALOUS POTENTIAL, REPORTS OF THE DEP. OF
C     GEODETIC SCIENCE NO. 225,1976.
C (B) TSCHERNING,C.C. AND R.H.RAPP: CLOSED COVARIANCE EXPRESSIONS
C     FOR GRAVITY ANOMALIES, GEOID UNDULATIONS, AND DEFLECTIONS OF
C     THE VERTICAL IMPLIED BY ANOMALY DEGREE-VARIANCE MODELS. DEP-
C     ARTMENT OF GEODETIC SCIENCE, THE OHIO STATE UNIVERSITY,
C     REPORT NO. 208, 1974.
C (C) KRARUP, T. AND C.C.TSCHERNING: EVALUATION OF ISOTROPIC COVARIANCE
C     FUNCTIONS OF TORSION BALANCE OBSERVATIONS. BULLETIN GEOD-
C     DESIQUE, VOL. 58, NO. 2, PP. 180-192, 1984.
C (D) TSCHERNING,C.C.: IMPLEMENTATION OF ALGOL-PROCEDURES FOR COV-
C     ARIANCE COMPUTATION ON THE RC 4000-COMPUTER. THE DANISH
C     GEODETIC INSTITUTE INTERNAL REPORT NO. 12, 1976.
C (H) TSCHERNING, C.C.: PREDICTION OF SPHERICAL HARMONIC
C     COEFFICIENTS USING LEAST-SQUARES COLLOCATION. SEPT. 1999.
C (I) Tscherning, C.C.: Computation of covariances of derivatives of the
C     anomalous gravity potential in a rotated reference frame.
C     Manuscripta Geodaetica, Vol. 18, no. 3, pp. 115-123, 1993.
C
      implicit none
C
      REAL*8 CI,CR,SIGMA0,SIGMA,HMAX,CV,D,D0,D1,D2,D3,D4,D5,RE,REX,
```



```
     *RADDEG,PI,GM,A,S,RB2,T,B,SIGMAP,slop,sloq,clop,cloq,RB,RE2,SM
      INTEGER KI,N1,N2,ITCOUN,KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,NR,ND1,
     *ND2,IIDEG,JJDEG,IS
      LOGICAL LOCAL,LSUM,LF,LT,LSPOUT,LTEST
C
      COMMON /CMCOV/CI(24),CR(56),SIGMA0(1200),SIGMA(1200),HMAX,
     *CV(2,2),D(36),KI(37),N1,N2,LOCAL,LSUM
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,REX,RADDEG,PI,GM,itcoun,
     *LTEST,LF,LT
      COMMON /DDY/A,S,RB2,T,B,KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,NR,ND1,ND2
      COMMON /PDEGV/SIGMAP(2001),slop,sloq,clop,cloq,IIDEG,JJDEG,LSPOUT
C THE COMMON BLOCK CONTAINS THE VALUES OF PARAMETERS USED FOR THE COM-
C PUTATIONS AND RETURN VALUES OF FUNCTIONS AND CONSTANTS, WHICH HAVE
C BEEN USED IN THE COMPUTATIONS.
C PARAMETERS USED FOR THE COMPUTATIONS:
C   CI(8) = THE CONSTANT A(I) OF REF.(A), EQ.(17) IN UNITS OF (M/SEC)**4
C   CI(10) THE SQUARE OF THE RATIO BETWEEN THE BJERHAMMAR-SPHERE RADIUS
C   (RB) AND THE MEAN RADIUS OF THE EARTH (RE), OR IF NEGATIVE RB-RE,
C   (CHANGE MADE 3 JULY 1985).
C   SIGMA0(IS+1)-SIGMA0(IS+N1) MUST CONTAIN THE EMPIRICAL ANOMALY
C   DEGREE VARIANCES IN UNITS OF MGAL**2.
C   KI(3) = K(2) OF DEG.VAR. MODEL 2 OR 3,
C   KI(4) = K(3) OF DEG.VAR. MODEL 3, CF. REF.(A), EQ.(17).
C   KI(5) = THE DEG.VAR. MODEL NUMBER, (EQUAL TO 1, 2 OR 3),
C   N1 = THE NUMBER OF EMPIRICAL DEGREE-VARIANCES USED (LOCAL =.FALSE.)
C   OR (ORDER+1) OF THE LOCAL COVARIANCE FUNCTION USED (LOCAL=.TRUE.).
C   HMAX, N2, LSUM. HMAX IS THE HEIGHT ABOVE WHICH THE LEGENDRE SERIES
C   OF MAXIMAL DEGREE N2-1 WILL BE USED FOR THE COMPUTATION OF THE CO-
C   VARIANCES WHEN LSUM IS TRUE. N2 MUST BE GREATHER THAN 2 AS WELL AS
C   GREATHER THAN N1.
C RETURN VALUES:
C   CI(10) RB-RE, A NEGATIVE VALUE (MODIFICATION 3 JULY 1985).
C   CI(9) = RB**2.
C
      DIMENSION SM(2001)
C THE ARRAY SM IS USED TO STORE THE DEGREE-VARIANCES WHEN THE LOGICAL
C VARIABLE LSUM IS TRUE. IN CASE THE SUBSCRIPT LIMIT IS CHANGED IS IT
C NECESSARY TO CHANGE THE VALUE OF THE VARIABLE N2 ACCORDINGLY.
C
      KT = KI(5)
      KT1 = KT+1
      IF (KT.GE.3) GO TO 15
      DO 16 K = KT, 2
   16 KI(K+2) = D0
   15 KI(1) = -2
      KI(2) = -1
C
      IF ((KT.LT.3).OR.(KT.EQ.3.AND.KI(4).GT.KI(3))) GO TO 17
C ASSURING, THAT KI(4).GT.KI(3), BECAUSE THIS FACT IS USED IN SUB-
C SEQUENT COMPUTATIONS.
      K = KI(3)
      KI(3) = KI(4)
      KI(4) = K
   17 II = KI(3)
      JJ = KI(4)
      SM(1) = D0
      SM(2) = D0
C     N3 = N1
      A = CI(8)
      S = CI(10)
      RE=6371000.0d0
      IF (S.GT.D0) GO TO 40
C S IS HERE RB-RE, A NEGATIVE VALUE. (MODIFICATION 3 JULY 1985).
      RB=RE+S
      RB2=RB*RB
      RE2=RE*RE
      S=RB2/RE2
      GO TO 41
   40 RB2 = S*RE2
```

```
      CI(10)=RE*( SQRT(S)-D1)
   41 CI(9) = RB2
      RB2 = RB2*1.0D-10
      T = D0
C
      SIGMA0(IS+1) = D0
      SIGMA0(IS+2) = D0
      IF (LOCAL) THEN
      SIGMA0(IS+3) = D0
      ELSE
      SIGMA0(IS+3) = SIGMA0(IS+3)*RB2/S**4
      END IF
      DO 13 K = 4, N1
      GO TO (10,11,12),KT
   10 KK = 1
      GO TO 14
   11 KK = K+II-1
      GO TO 14
   12 KK = (K+II-1)*(K+JJ-1)
   14 IF (K.LE.N1) THEN
C CONVERSION FROM MGAL**2 TO M**2/SEC**2. (AND COMMENT 2006-02-07:
C MOVING FROM MEAN EARTH TO BJERHAMMAR SPHERE).
      IF (.NOT.LOCAL) T = SIGMA0(IS+K)*S**(-K-1)*RB2
      SIGMA0(IS+K) = (T-A*(K-2)/((K-3)*KK))/(K-2)**2
      END IF
   13 CONTINUE
      RETURN
      END
C ---------------------------------------------------------------------
      SUBROUTINE COVBX(SM,LSAT,IS)
C ORIGINAL VERSION PROGRAMMED JULY 1975 BY C.C.TSCHERNING AS A SUB-
C ENTRY OF COVAX. NEW VERSION CREATED SEP 1987 BY CCT.
C NEW VERSION JUNE 4, 1991. LAST UPDATE 2005-05-18 BY CCT.
C
C THE CALL OF COVBX WILL FIX CERTAIN CONSTANTS USED FOR THE COMPUTA-
C TIONS, WHICH ARE INDEPENDENT OF THE POINTS P AND Q. WHEN COVBX IS CAL-
C LED, THE KIND OF QUANTITIES BETWEEN WHICH THE COVARIANCE IS TO BE
C COMPUTED MUST BE SPECIFIED.  THIS IS DONE BY STORING IN KI(6) AND
C KI(7) INTEGERS EQUAL TO THE EQUATION NUMBERS OF REF.A, EQ.(1) - (9)
C (12) AND (14), AND 10, 11, 13, 15 CORRESPONDING TO REF.(C), EQ.
C (3) - (6). HOWEVER, THE QUANTITY OF KIND 2 IS NOW THE GRAVITY
C DISTURBANCE (CHANGED FROM THE SAME QUANTITY DIVIDED BY R).
C ADDED 1999.02.12 IS (17), FOR COEFFICIENTS OF SPHERICAL HARMONICS.
C
C REFERENCES (A) - (I): SEE COVAX.
C
      implicit none
      REAL*8 CI,CR,SIGMA0,SIGMA,HMAX,CV,D,D0,D1,D2,D3,D4,D5,RE,
     *RADSEC,PI,GM,A,S,RB2,T,B,SIGMAP,slop,sloq,clop,cloq,
     *COVX,CIX,CFA,SM,SIGMAX,RE2,SNN,BB0,RKP,REM,C11
      INTEGER KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,NR,ND1,ND2,IIDEG,JJDEG,
     *KSAT,NDX1,NDX2,NDP,NDQ,NWAR,KI,N1,N2,ITCOUN,K7,K9,K11,K13,
     *K15,K17,K19,K21,K23,K8,J2,I3,I4,M,MK,IS,I,NDT,NDTOT,NDY,KU
C
      LOGICAL LOCAL,LSUM,LN,L,LF,LT,LSAT,LNX,LTEST,LTESTS,LSPOUT,LX,
     *LSPHAR
C
      COMMON /CMCOV/CI(24),CR(56),SIGMA0(1200),SIGMA(1200),HMAX,
     *CV(2,2),D(36),KI(37),N1,N2,LOCAL,LSUM
      COMMON /DCONc/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GM,ITCOUN,
     *LTEST,LF,LT
      COMMON /DDX/K7(17),K9(17),K11(17),K13(17),K15(17),K17(17),
     *K19(17),K21(17),K23(17),K8(17),C11(17),J2(2),I3(2),I4(2),
     *LN(7),L(7)
      COMMON /DDY/A,S,RB2,T,B,KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,NR,ND1,ND2
      COMMON /PDEGV/SIGMAP(2001),slop,sloq,clop,cloq,IIDEG,JJDEG,LSPOUT
      COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFA,KSAT(17,2),
     *NDX1(5),NDX2(5),NDP,NDQ,NWAR,LX(7,5),LNX(7,5),LTESTS
C THE COMMON BLOCK CONTAINS THE VALUES OF PARAMETERS USED FOR THE COM-
```



```
C PUTATIONS AND RETURN VALUES OF FUNCTIONS AND CONSTANTS, WHICH HAVE
C BEEN USED IN THE COMPUTATIONS.
C PARAMETERS USED FOR THE COMPUTATIONS:
C   CI(8) = THE CONSTANT A(I) OF REF.(A), EQ.(17) IN UNITS OF (M/SEC)**4
C   CI(10) THE SQUARE OF THE RATIO BETWEEN THE BJERHAMMAR-SPHERE RADIUS
C   (RB) AND THE MEAN RADIUS OF THE EARTH (RE), OR IF NEGATIVE RB-RE,
C   (CHANGE MADE 3 JULY 1985).
C   CI(13) USER DEFINED VALUE OF CI(11). CI(14), CI(15) USER DEFINED
C   VALUES OF CI(21) - CI(24).
C   SIGMA0(IS+1)-SIGMA0(IS+N1) MUST CONTAIN THE POTENTIAL ANOMALY
C   DEGREE-VARIANCE CORRECTIONS, CF. REF.(A), EQ.16.
C   KI(3) = K(2) OF DEG.VAR. MODEL 2 OR 3,
C   KI(4) = K(3) OF DEG.VAR. MODEL 3, CF. REF.(A), EQ.(17).
C   KI(5) = THE DEG.VAR. MODEL NUMBER, (EQUAL TO 1, 2 OR 3),
C   KI(6),KI(7) THE INTEGER SPECIFYING THE KIND OF QUANTITY WHICH IS
C   ASSOCIATED WITH P, Q, RESPECTIVELY,
C   KI(26) - KI(34) USER SPECIFIED VALUES FOR KI(10) - KI(23).
C   KI(35) - KI(37) USED BY SUBROUTINE COVCG FOR STATISTICAL PURPOSES.
C   N1 = THE NUMBER OF EMPIRICAL DEGREE-VARIANCES USED (LOCAL =.FALSE.)
C   OR (ORDER+1) OF THE LOCAL COVARIANCE FUNCTION USED (LOCAL=.TRUE.).
C   HMAX, N2, LSUM. HMAX IS THE HEIGHT ABOVE WHICH THE LEGENDRE SERIES
C   OF MAXIMAL DEGREE N2-1 WILL BE USED FOR THE COMPUTATION OF THE CO-
C   GREATHER THAN N1.
C RETURN VALUES:
C   CI(1)-CI(7), THE QUANTITIES C(J,Q) OF REF.(A), EQ.(47), WITH
C   CI(1) - CI(KI(5)+1) = C(J,Q), CI(5) = C(KI(5)+2,Q),
C   CI(6) = C(KI(5)+3,Q), CI(7) = C(KI(5)+4,Q),
C   CI(11),CI(12) QUANTITIES USED TO GIVE THE COMPUTED
C   COVARIANCES THE PROPER UNITS.
C   CI(21) - CI(24) THE QUANTITIES M(1) - M(4) OF REF.(A) EQ. (26) -
C   (29). (CHANGE MADE 1986.10.20).
C   SIGMA(IS+4) - SIGMA(IS+N1), THE POTENTIAL DEGREE-VARIANCES MULTI-
C   PLIED BY THE FACTORS GIVEN IN REF.(A), TABLE 1.
C   SIGMA(IS+1) - SIGMA(IS+3), THE DEGREE-VARIANCES OF DEGREE 0,1,2
C   MINUS TERMS OF THE SAME DEGREES ACQUIRED FROM REF.(A), EQ.(34),(35),
C   (41) AND (42).
C   KI(8),KI(9) THE NUMBER OF DIFFERENTIATIONS IN RADIAL DIRECTION AND
C   WITH RESPECT TO T = COS(SPHERICAL DIST.) TO BE PERFORMED.
C   KI(10) - KI(15) THE CONSTANTS I,K,J,M,J1,M1 OF REF.(A), SECTION 2.
C   KI(16) - KI(19) THE QUANTITIES M(1) - M(4) OF REF.(A), EQ.(26)-(29).
C   KI(20),KI(21) THE EXPONENT OF THE REFERENCE GRAVITY,
C   KI(22),KI(23) THE EXPONENT OF THE RADIAL DISTANCE AND
C   KI(24),KI(25) SUBSCRIPTS OF THE RESULT STORED IN CV (COMMON CMCOV).
C
      DIMENSION SM(2001),SIGMAX(400,5)
C THE ARRAY SM IS USED TO STORE THE DEGREE-VARIANCES WHEN THE LOGICAL
C VARIABLE LSUM IS TRUE. IN CASE THE SUBSCRIPT LIMIT IS CHANGED IS IT
C NECESSARY TO CHANGE THE VALUE OF THE VARIABLE N2 ACCORDINGLY.
      EQUIVALENCE (SIGMAX(1,1),SIGMA0(401))
C SIGMAX IS USED TO HOLD DEGREE-VARIANCES OF RADIAL DERIVATIVES
C UP TO ORDER 2 IN P AND Q. (CHANGE MAY 1991).
C
C THE ARRAYS K7 - K23 CONTAINS TABLES OF QUANTITIES RELATED TO THE KIND
C OF COVARIANCES (1 - 14) WHICH MAY BE COMPUTED. THEIR ACTUAL VA-
C LUES WILL AFTER CALL OF COVBX BE STORED IN THE ELEMENTS OF THE ARRAY
C KI HAVING SUBSCRIPTS 8 - 25.
C K7 CONTAINS THE ORDER OF DIFFERENTIATION WITH RESPECT TO T,K8 THE
C ORDER OF DIFFERENTIATION WITH RESPECT TO THE RADIUS, CF.REF(A),TABLE
C 1. K9,K11,K13 THE KIND OF DIFFERENTIATIONS TO BE COMPUTED WITH RESPECT
C TO THE LATITUDE (2) AND THE LONGITUDE (3), CF.REF(A),SECTION 3. K15
C AND K17 CONTAINS AN INTEGER, WHICH WILL BE ADDED TO THE DEGREE. THE
C SUM WILL THEN BE MULTIPLIED WITH THE DEGREE-VARIANCE OF THE CORRESPON-
C DING DEGREE WHEN A FIRST AND/OR SECOND DIFFERENTIATION WITH RESPECT
C TO THE RADIAL DISTANCE HAS TAKEN PLACE.
C C11 CONTAIN QUANTITIES USED TO GIVE THE COVARIANCES THE PROPER UNITS.
C
C     LTEST=LTESTS
      LSPHAR=.FALSE.
      ITCOUN=0
```

```
      RB2 = CI(9)
      REM=6371000.0d0
      RE2=REM**2
      S=RB2/RE2
      A = CI(8)
      II=KI(3)
      JJ=KI(4)
      KT=KI(5)
      KT1=KT+1
      N3=N1
      CI(11) = D1
      KI(8)=0
      KI(9)=0
      IF (KI(6).GT.17.OR.KI(7).GT.17) GO TO 19
C
      DO 20 M = 1, 2
      K = KI(M+5)
C FOR M = 1, K IS EQUAL TO THE KIND EVALUATED IN P AND FOR M = 2 EQUAL
C TO THE KIND EVALUATED IN Q.
C
      IF (K.EQ.0.OR.K.GE.16) GO TO 42
      KI(M+9) = K9(K)
      KI(M+11) = K11(K)
      KI(M+13) = K13(K)
      CI(M+20) = K15(K)
      CI(M+22) = K17(K)
      KI(M+19) = K19(K)
      KI(M+21) = K21(K)
      KI(M+23) = K23(K)
C
      CI(11) = CI(11)*C11(K)
C     write(*,*)' k ',K,CI(21),CI(22),CI(23),CI(24)
      KI(8)=KI(8)+K7(K)
      KI(9)=KI(9)+K8(K)
      GO TO 20
C
C USER DEFINED VALUES OF KI AND CI. MAY BE USER FOR DENSITY CONTRAST
C COVARIANCES, CF. REF.(D), SECTION 3.
   42 IF (K.NE.17) THEN
      DO 43 MK=1,8
   43 KI(M+MK*2+7)=KI(MK+25)
      CI(11) = CI(11)*CI(13)
      CI(M+20) = CI(14)
      CI(M+22) = CI(15)
      KI(8)=KI(8)+KI(29)
      KI(9)=KI(9)+KI(30)
c     LSPHAR=.FALSE.
c     write(*,*)' LSPHAR=F '
      ELSE
      LSPHAR=.TRUE.
      CI(21)=D0
      CI(22)=D0
      CI(23)=D0
      CI(24)=D0
c     write(*,*)' LSPHAR=T '
c     WRITE(*,*)' N1,N3 ',N1,N3
      END IF
   20 CONTINUE
C
      KQ = K
      KP = KI(6)
   19 ND = KI(8)
c     WRITE(*,*)' COVBX: ND= ',ND
      NR = KI(9)
C
      NDP=K7(KP)+K8(KP)
      NDQ=K7(KQ)+K8(KQ)
c     WRITE(*,*)' COVBX: ND,NDP,NDQ= ',ND,NDP,NDQ
C ND AND NR ARE THE NUMBER OF DIFFERENTIATIONS WITH RESPECT TO T AND
```



```
C THE RADIAL DISTANCES, RESPECTIVELY. NDP, NDQ ARE THE TOTAL NMBER OF
C DERIVATIVES IN P, Q, REPECTIVELY.
C
      IF (LSAT.AND.(.NOT.LSPHAR)) GO TO 100
C UPDATING THE DEGREE-VARIANCES, CF. REF(A), TABLE 1.
      SIGMA(IS+1) = D0
      SIGMA(IS+2) = D0
      SIGMAP(IS+1)= D0
      SIGMAP(IS+2)= D0
      IF (LSUM) N1 = N2
      IF (N1.GE.600) WRITE(*,*)' WARNING N1.gt.600 '
      SNN=S**3
      DO 21 M = 3, 600
      B = D1
      DO 22 I = 1, 4
   22 IF ( ABS(CI(I+20)).GT.0.0) B = B*(M+CI(I+20)-1)
      BB0=B
      IF (M.LE.N3) SIGMA(IS+M) = SIGMA0(IS+M)*B
      IF (.NOT.(LSUM.OR.LSPHAR).OR.M.EQ.3) GO TO 21
      DO 48 K = 1, KT1
   48 B = B/(M+KI(K)-1)
C STORING THE MODIFIED DEGREE-VARIANCES OF DEGREE M-1 IN SM(M) AND AD-
C DING THE DEGREE-VARIANCE CORRECTIONS FOR M .LE. N3.
      SM(M) = B*A
      SNN=SNN*S
      IF (M.LE.N3) THEN
      SM(M) = SM(M)+SIGMA(IS+M)
C     SIGMAP(M)=SM(M)*SNN
      SIGMAP(M)=SM(M)*SNN/BB0
      ELSE
C     SIGMAP(M)=B*A*SNN
      SIGMAP(M)=A*SNN*B/BB0
      END IF
C
C CF. REF(H), EQ. (4).
      SIGMAP(M)=SIGMAP(M)/(D2*M-D1)
c change 2005-05-18 to assure positive degree-variances.
      if (sigmap(m).lt.0.0d0) sigmap(m)=d0
   21 CONTINUE
      IF (N1.GT.2) THEN
      SM(3) = SIGMA(IS+3)
C     SIGMAP(3)=SM(3)*(S**3)/5.0d0
      SIGMAP(3)=SM(3)*(S**3)/(BB0*5.0d0)
      ELSE
      SIGMAP(3)=0.0D0
      END IF
C
      IF (LSPHAR.AND.LSPOUT) THEN
      LSPOUT=.FALSE.
      WRITE(*,*)' GRAVITY ANOMALY AND POTENTIAL DEG.VAR. DEG 3-32 '
      WRITE(*,249)
     *(SIGMAP(K)*(2*K-1)*(K-2)**2*1.0D10/RE2,K=3+IS,32+IS)
      WRITE(*,249)(SIGMAP(K),K=3+IS,32+IS)
  249 FORMAT(8F9.4)
      END IF
      IF (LSUM) N1 = N3
C
C EVALUATION OF THE QUANTITIES C(J,NR), CF.REF(A), TABLE 2.
      DO 23 K = 1, 7
   23 CI(K) = D0
C
      DO 25 K = 1, KT1
      CI(K) = D1
      DO 25 KQ = 1, KT1
   25 IF (K.NE.KQ) CI(K) = CI(K)/(KI(KQ)-KI(K))
C CF.,EQ.(19). WE WILL THEN COMPUTE THE QUANTITIES GIVEN IN REF(A)
C REF(A), TABLE 2.
      IF (NR.LT.2) GO TO 29
      RKP = CI(21)+CI(22)+CI(23)+CI(24)
```

```
      IF (NR.EQ.4) REM = CI(21)*(CI(22)+CI(23)+CI(24))+CI(22)
     *                 *(CI(23)+CI(24))+CI(23)*CI(24)
C
      GO TO (26,27,28),KT
   26 CI(NR+3) = D1
      IF (NR.GT.2) CI(NR+2) = RKP+3
      IF (NR.EQ.4) CI(NR+1) = REM+3*RKP+7
      GO TO 29
   27 IF (NR.GT.2) CI(NR+2) = D1
      IF (NR.EQ.4) CI(NR+1) =-KI(3)+3+RKP
      GO TO 29
   28 IF (NR.EQ.4) CI(NR+1) = D1
   29 IF (NR.EQ.0) GO TO 31
C
      DO 30 KP = 1, 4
      DO 30 K  = 1, KT1
   30 IF ( ABS(CI(KP+20)).NE.0.0) CI(K) = CI(K)*(CI(KP+20)-KI(K))
C
C THE LOGICAL ARRAYS L AND LN REGISTER WHICH TERMS THAT WILL HAVE TO
C BE EVALUATED , RESPECTIVELY NOT EVALUATED IN REF.(A), EQ. (47).
   31 DO 38 K = 1, 7
      L(K) =  ABS(CI(K)).GT.1.0E-15
   38 LN(K) = .NOT.(L(K))
C
      DO 32  K = 3, 7
      DO 32  M = 1, 3
      IF (M.EQ.1.AND.K.GT.5.OR.(M+KI(K)-1).EQ.0.AND.K.LT.5.OR.LN(K))
     *GO TO 32
      GO TO (34,34,35,35,34,36,37),K
   34 B = D1
      GO TO 33
   35 B = D1/(M+KI(K) -1)
      GO TO 33
   36 B = (M-1)
      GO TO 33
   37 B = (M-1)*(M-1)
   33 SIGMA(IS+M) = SIGMA(IS+M)-A*CI(K)*B
   32 CONTINUE
      SIGMA(IS+3) = SIGMA(IS+3)-A*CI(2)
      IF (LTEST) write(*,2)(SIGMA(I),I=1,6)
    2 FORMAT(6E12.6,I3)
C
      ND1 = ND+1
      ND2 = ND1+1
      RETURN
C
  100 DO 109 M=1,7
      DO 109 NDT=1,5
  109 LNX(M,NDT)=LT
      NDTOT=NDP+NDQ+1
      ND=NDTOT-1
      ND1=ND+1
      ND2=ND1+1
C
      DO 101 NDT=1,NDTOT
      DO 110 M=1,4
  110 CI(M+20)=D0
      M=1
      IF (NDT.GT.1) THEN
       CI(21)=D1
       M=2
      END IF
      IF (NDT.GT.2) THEN
      IF (NDP.EQ.1.AND.NDQ.EQ.1.AND.NDTOT.EQ.3) THEN
      CI(22)=D1
      ELSE
      CI(22)=D2
      END IF
      M=3
```



```
      END IF
      IF (NDT.GT.3) THEN
      CI(23)=D1
      M=M+1
      IF (NDT.EQ.5)THEN
      CI(24)=D2
      M=M+1
      END IF
      END IF
      NR=M-1
      NDY=NDTOT-M
      IF (LTEST)WRITE(6,*)NDT,CI(21),CI(22),CI(23),CI(24)
C UPDATING THE DEGREE-VARIANCES, CF. REF(A), TABLE 1.
      SIGMAX(1,NDT) = D0
      SIGMAX(2,NDT) = D0
      DO 121 M = 3, N1
      B = D1
      DO 122 I = 2, NDT
  122 B = B*(M+CI(I+19)-1)
C noget galt her.
  121 IF (M.LE.N3) SIGMAX(M,NDT) = SIGMA0(IS+M)*B
C
C EVALUATION OF THE QUANTITIES C(J,NR), CF.REF(A), TABLE 2.
      DO 123 K = 1, 7
  123 CI(K) = D0
C
      DO 125 K = 1, KT1
      CI(K) = D1
      DO 125 KU = 1, KT1
  125 IF (K.NE.KU) CI(K) = CI(K)/(KI(KU)-KI(K))
C CF.,EQ.(19). WE WILL THEN COMPUTE THE QUANTITIES GIVEN IN REF(A)
C REF(A), TABLE 2.
      IF (NR.LT.2) GO TO 129
      RKP = CI(21)+CI(22)+CI(23)+CI(24)
      IF (NR.EQ.4) REM = CI(21)*(CI(22)+CI(23)+CI(24))+CI(22)
     *                 *(CI(23)+CI(24))+CI(23)*CI(24)
C
      GO TO (126,127,128),KT
  126 CI(NR+3) = D1
      IF (NR.GT.2) CI(NR+2) = RKP+3
      IF (NR.EQ.4) CI(NR+1) = REM+3*RKP+7
      GO TO 129
  127 IF (NR.GT.2) CI(NR+2) = D1
      IF (NR.EQ.4) CI(NR+1) =-KI(3)+3+RKP
      GO TO 129
  128 IF (NR.EQ.4) CI(NR+1) = D1
  129 IF (NR.EQ.0) GO TO 131
C
      DO 130 KU = 1, 4
      DO 130 K  = 1, KT1
  130 IF ( ABS(CI(KU+20)).NE.0.0) CI(K) = CI(K)*(CI(KU+20)-KI(K))
  131 DO 106 K=1,7
  106 CIX(K,NDT)=CI(K)
C
C THE LOGICAL ARRAYS L AND LN REGISTER WHICH TERMS THAT WILL HAVE TO
C BE EVALUATED , RESPECTIVELY NOT EVALUATED IN REF.(A), EQ. (47).
      DO 138 K = 1, 7
      IF (NDT.EQ.1) L(K)=LF
      LNX(K,NDT)= ABS(CI(K)).LE.1.0D-10
      L(K) =  ABS(CI(K)).GT.1.0E-10.OR.L(K)
  138 LN(K)=.NOT.(L(K))
      IF (LTEST) WRITE(6,*)'NDT,LN',NDT,(LNX(K,NDT),K=1,7)
C
      DO 132  K = 3, 7
      DO 132  M = 1, 3
      IF (M.EQ.1.AND.K.GT.5.OR.(M+KI(K)-1).EQ.0.AND.K.LT.5.OR.
     *LNX(K,NDT)) GO TO 132
      GO TO (134,134,135,135,134,136,137),K
  134 B = D1
```

```
      GO TO 133
  135 B = D1/(M+KI(K) -1)
      GO TO 133
  136 B = (M-1)
      GO TO 133
  137 B = (M-1)*(M-1)
  133 SIGMAX(M,NDT) = SIGMAX(M,NDT)-A*CI(K)*B
  132 CONTINUE
      SIGMAX(3,NDT) = SIGMAX(3,NDT)-A*CI(2)
      IF (LTEST) write(*,2)(SIGMAX(I,NDT),I=1,6),NDT
C
      NDX1(NDT) = NDY+1
      NDX2(NDT) = NDY+2
  101 CONTINUE
      RETURN
      END
C ---------------------------------------------------------------------
      SUBROUTINE COVCX(SM,COV,IS,LSAT,gcx,lsphap)
C ORIGINALLY PROGRAMMED JULY 1975 BY C.C.TSCHERNING AS A SUB-
C ENTRY TO COVAX. SEPARATE SUBROUTINE CREATED SEPT 1987 BY CCT.
C LATEST MODIFICATION APR 01, 2003 BY CCT.
C
C COMPUTATION OF THE COVARIANCE IN A SPECIFIC PAIR OF POINTS, OR
C BETWEEN A FUNCTIONAL ASSOCIATED WITH A POINT AND A SPHERICAL-HARMONIC
C COEFFICIENT. THE  VALUE IS RETURNED THROUGH THE PARAMETER COV.
C THE COVARIANCES COMPUTED WILL BE IN UNITS CORRESPONDING TO THE KIND
C OF QUANTITIES, I.E. FOR KIND (1) METERS, (2) EOTVOS (E), (3) MGAL,
C (4),(5) E, (6),(7) ARCSECONDS, (8) - (14) E, (17) UNITLESS.
C THE FOLLOWING QUANTITIES MUST BE STORED IN THE ELEMENTS OF THE ARRAY
C CR WHEN COVCX IS CALLED: (1) COSINE TO THE SPHERICAL DISTANCE BET-
C WEEN P AND Q, (2),(3) THE HEIGHT OF P, Q RESPECTIVELY, (4),(5) SINE
C OF THE LATITUDE THE  OF P, Q, RESPECTIVELY, (6),(7) COSINE OF THE
C LATITUDE OF P, Q, RESPECTIVELY, (8),(9) SINE AND COSINE OF THE
C LONGITUDE DIFFERENCE. THE REFERENCE GRAVITY WILL HAVE TO BE STORED
C IN CR(10),CR(11) FOR P, Q RESPECTIVELY (WHEN USED, OTHERWISE STORE
C 1.0). FOR KIND 17, COS AND SIN OF LONGITUDES MUST BE STORED IN THE
C COMMON BLOCK /PDEGV/.
C
C THE CALL OF COVCX WILL RESULT IN THE COMPUTATION OF THE COVARIANCE ,
C WHICH IS TRANSFERRED TO THE CALLING PROGRAM THROUGH THE VARIABLE COV.
C THE RESULT WILL ALSO BE TRANSFERRED IN THE COMMON CMCOV, BY THE ARRAY
C CV(2,2). IN CASE IT IS POSSIBLE TO COMPUTE MORE THAN ONE QUANTITY AT
C A TIME (I.E. WHEN DERIVATIVES WITH RESPECT TO T=COS(SPHERICAL DIST-
C TANCE) ARE COMPUTED, KINDS 6 - 11, 13 AND 15), THE COVARIANCE
C OF TYPE 6, 8, 10 AND 23 WILL BE STORED IN THE ELEMENT WITH SUBSCRIPT
C 2 AND OTHERWISE IN THE ELEMENT WITH SUBSCRIPT 1. THE KIND OF THE
C FUNCTIONALS IN P WILL DETERMINE THE VALUE OF THE FIRST SUBSCRIPT
C WHILE THE KIND OF THE FUNCTIONALS IN Q WILL DETERMINE THE SECOND
C SUBSCRIPT. EXAMPLE: KIND 6 IN P AND KIND 1 IN Q WILL DELIVER
C THE COVARIANCE BETWEEN THE PRIME-VERTICAL VERTICAL DEFLECTION AND
C AND THE HEIGHT ANOMALY IN CV(1,1), BETWEEN THE MERIDIAN VERTICAL
C DEFLECTIAN AND THE HEIGHT ANOMALY IN CV(2,1).
C
C WHEN LSAT IS TRUE, THE 4D ARRAY COVX HOLDS THE VECTORS OR MATRICES
C OF COVARIANCES BETWEEN ALL 0, 1 OR 2 DERIVATIVES.
C
      implicit none
      REAL*8 CI,CR,SIGMA0,SIGMA,HMAX,CV,D
      INTEGER KI,N1,N2,ITCOUN,IIMAX,IMAX1,I21,I,ILAST,JMAX1,J,JKK,IKK,
     *NCASE,KPQ,IDIF,KKC,KKD,M,K1,K2,I1,I2,NDTOT,NDT,IS,J1,M1,IJ,KM,IX,
     *IIX,IIY,JX,IX1,JX1,K6,M6 ,KZ
c     *,id,jd
      REAL*8 D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GM,C11,CN,CY,R2PQ,
     *A,S,RB2,T,B,HP,HQ,SP,SQ,CP,CQ,SD,CD,RP,RQ,RE2,CLAT,SLAT,CLON,SLON,
     *RH,GAMM,COV,CJLO,SJLO,WWC,GGC,WWS,GGS,COVC,WW,DDDC,DDDS,
     *GG,DDD,SJL1,SC,CS,SCC,CC,CCS,COVS,CSC,CPSD,CQSD,CPCD,CQCD,SS,C,
     *S2,ST,T2,P2,P3,CX,GI,GJ,SI,SM,DC,SIGMAX,DCN,RL,RL2,R,RL1,RN,
     *RNL,P,RL3,RL5,S3,RL4,RL7,S4,S5,RL6,U,RM,V,Q,G,SS1,SS2,RP2,
     *RQ2,RPQ,FAK5,RP2Q,CNX,DD,RPQ2,D3132,D313,CN23,CN33,D37,D27,
```



```
     *CF,CZ,C11P,C11Q
      INTEGER KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,NR,ND1,ND2,
     *KSAT,NDX1,NDX2,NDP,NDQ,NWAR,IIOLD,JOLD
     *,NFU,KEYH,NINTH,NTABH,NHE,NSTART,IIDEG,JJDEG
      REAL*8 COVX,CIX,CFA,RRC,HTA,TMAX,SIZEI,SIGMAP,slop,sloq,
     *clop,cloq,root0,SUMIJ,CCCIJ,
     *sq2,ys,yc,vv,v1,gs,gc,dds,ddc,pii,pim0,pim1,pim2,dlp,
     *dlp0,dlp1,dlp2,dap,dap0,
     *dap1,dap2,ddap,ddap0,ddap1,ddap2,ddal0,ddal1,ax,rex
      INTEGER K7,K9,K11,K13,K15,K17,K19,K21X,K23,K8,J2,I3,I4

      parameter (iimax=2000)
C
      LOGICAL LOCAL,LSUM,LSUMC,LOLDP,LOLDQ,LN,L,LF,LT,LTEST,LTABH
     *,LTA,LSAT,LTESTS,LDGP,LDGQ,LSPHAR,LSPOUT,LTSPH,LCOS
     *,LSPHP,LSPHQ,LX,LNX,lsphap,ltes0
c
      COMMON /CMCOV/CI(24),CR(56),SIGMA0(1200),SIGMA(1200),HMAX,
     *CV(2,2),D(36),KI(37),N1,N2,LOCAL,LSUM
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,REX,RADSEC,PI,GM,ITCOUN,
     *LTEST,LF,LT
      COMMON /DDX/K7(17),K9(17),K11(17),K13(17),K15(17),K17(17),
     *K19(17),K21X(17),K23(17),K8(17),C11(17),J2(2),I3(2),I4(2),
     *LN(7),L(7)
C     COMMON /DDX/K7(15),K9(15),K11(15),K13(15),K15(15),K17(15),K19(15),
C     *K21(15),K23(15),K8(15),C11(15),J2(2),I3(2),I4(2),LN(7),L(7)
      COMMON /DDY/A,S,RB2,T,B,KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,NR,ND1,ND2
      COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFA,KSAT(17,2),
     *NDX1(5),NDX2(5),NDP,NDQ,NWAR,LX(7,5),LNX(7,5),LTESTS
      COMMON /CTABH/RRC(1200),HTA(5),TMAX,SIZEI,NFU(5),KEYH(5,5),
     *NINTH,NTABH(15,5,5),NHE,NSTART,LTABH
      COMMON /PDEGV/SIGMAP(2001),slop,sloq,clop,cloq,IIDEG,JJDEG,LSPOUT
      common  /rrsph/root0(iimax)
      common /con3/SUMIJ(32761),CCCIJ(32761),
     *sq2,ys,yc,vv,v1,gs(3),gc(3),dds(3,3),
     *ddc(3,3),iiold,jold,LSPHAR,LTSPH
c     common /sphold/pii,pim0,pim1,pim2,dlp,dlp0,dlp1,dlp2,dap,dap0,
c     *dap1,dap2,ddap,ddap0,ddap1,ddap2,ddal0,ddal1,vi
C THE COMMON BLOCK CONTAINS THE VALUES OF PARAMETERS USED FOR THE COM-
C PUTATIONS AND RETURN VALUES OF FUNCTIONS AND CONSTANTS, WHICH HAVE
C BEEN USED IN THE COMPUTATIONS.
C PARAMETERS USED FOR THE COMPUTATIONS:
C   CI(8) = THE CONSTANT A(I) OF REF.(A), EQ.(17) IN UNITS OF (M/SEC)**4
C   CI(10) THE SQUARE OF THE RATIO BETWEEN THE BJERHAMMAR-SPHERE RADIUS
C   (RB) AND THE MEAN RADIUS OF THE EARTH (RE), OR IF NEGATIVE RB-RE,
C   (CHANGE MADE 3 JULY 1985).
C   CI(13) USER DEFINED VALUE OF CI(11). CI(14), CI(15) USER DEFINED
C   VALUES OF CI(21) - CI(24).
C   NEW VARIABLES ADDED MAY 1, 1986 AND NOV 1986:
C   CI(16) - CI(24), WHERE CI(20)=0.0 IF PRECISE FORMULAE FOR DERIVATIVES
C   MAY BE USED. IN THIS CASE IS CI(16)=SIN(LONGITUDE DIFFERENCE/2)**2,
C   CI(17)=SIN(LATITUDE DIFFERENCE/2), CI(18)=COS(LATITUDE DIFFERENCE),
C   CI(19)=COS(LATITUDE DIFFERENCE/2). OTHERWISE CI(20)=1.0.
C   CR(2),CR(3) THE HEIGHT OF P, Q, RESPECTIVELY, (UNITS METERS),
C   CR(4),CR(5) SINE OF THE LATITUDE OF P, Q, RESPECTIVELY,
C   CR(6),CR(7) COSINE OF THE LATITUDE OF P, Q, RESPECTIVELY,
C   CR(8),CR(9) SINE AND COSINE OF THE LONGITUDE DIFFERENCE,
C   CR(10),CR(11) THE REFERENCE GRAVITY IN P, Q, RESPECTIVELY (WHEN
C   USED, OTHERWISE STORE 1.0E0), (UNITS M/SEC**2).
C   KI(3) = K(2) OF DEG.VAR. MODEL 2 OR 3,
C   KI(4) = K(3) OF DEG.VAR. MODEL 3, CF. REF.(A), EQ.(17).
C   KI(5) = THE DEG.VAR. MODEL NUMBER, (EQUAL TO 1, 2 OR 3),
C   KI(6),KI(7) THE INTEGER SPECIFYING THE KIND OF QUANTITY WHICH IS
C   ASSOCIATED WITH P, Q, RESPECTIVELY,
C   KI(26) - KI(34) USER SPECIFIED VALUES FOR KI(10) - KI(23).
C   KI(35) - KI(37) USED BY SUBROUTINE COVCG FOR STATISTICAL PURPOSES.
C   N1 = THE NUMBER OF EMPIRICAL DEGREE-VARIANCES USED (LOCAL =.FALSE.)
C   OR (ORDER+1) OF THE LOCAL COVARIANCE FUNCTION USED (LOCAL=.TRUE.).
C   HMAX, N2, LSUM. HMAX IS THE HEIGHT ABOVE WHICH THE LEGENDRE SERIES
```

```
C   OF MAXIMAL DEGREE N2-1 WILL BE USED FOR THE COMPUTATION OF THE CO-
C   VARIANCES WHEN LSUM IS TRUE. N2 MUST BE GREATHER THAN 2 AS WELL AS
C   GREATHER THAN N1.
C RETURN VALUES:
C   CR(ND*8+12), THE VALUES OF THE ND'TH DERIVATIVE OF THE SUM OF THE
C   FINITE LEGENDRE-SERIES, CF.REF.(A), EQ.(20),(48) AND (52).
C   CR(ND*8+13) - CR(ND*8+19), THE VALUES OF THE ND'TH DERIVATIVES OF
C   THE FUNCTIONS F(-2), F(-1), F(KI(3)), F(KI(4)), S0, S1, S2, CF. REF.
C   (A), EQ. (42), (41), (39), (39), (30), (34) AND (35).
C   SIGMA0(IS+1) - SIGMA0(IS+N1) THE POTENTIAL DEGREE-VARIANCE
C   CORRECTIONS, CF. REF.(A), EQ.(16), (AFTER THE CALL OF COVAX).
C   SIGMA(IS+4) - SIGMA(IS+N1), THE POTENTIAL DEGREE-VARIANCES MULTI-
C   PLIED BY THE FACTORS GIVEN IN REF.(A), TABLE 1.
C   SIGMA(IS+1) - SIGMA(IS+3), THE DEGREE-VARIANCES OF DEGREE 0,1,2
C   MINUS TERMS OF THE SAME DEGREES ACQUIRED FROM REF.(A), EQ.(34),(35),
C   (41) AND (42).
C   KI(8),KI(9) THE NUMBER OF DIFFERENTIATIONS IN RADIAL DIRECTION AND
C   WITH RESPECT TO T = COS(SPHERICAL DIST.) TO BE PERFORMED.
C   KI(10) - KI(15) THE CONSTANTS I,K,J,M,J1,M1 OF REF.(A), SECTION 2.
C   KI(16) - KI(19) THE QUANTITIES M(1) - M(4) OF REF.(A), EQ.(26)-(29).
C   KI(20),KI(21) THE EXPONENT OF THE REFERENCE GRAVITY,
C   KI(22),KI(23) THE EXPONENT OF THE RADIAL DISTANCE AND
C   KI(24),KI(25) SUBSCRIPTS OF THE RESULT STORED IN CV (COMMON CMCOV).
C
C ARRAYS CN, DCN, SIGMAX, DD ADDED MAY 1991.
C
C REFERENCES (A)-(I) SEE COVAX.
C
      DIMENSION SM(2001),CX(6,8),DC(6),SIGMAX(400,5),CN(8,5),DCN(8,5),
     *C(6),V(6),U(6),G(6),P(6),R(6),SS1(4),CZ(5),RM(6),Q(6),DD(6,6)
C THE ARRAY SM IS USED TO STORE THE DEGREE-VARIANCES WHEN THE LOGICAL
C VARIABLE LSUM IS TRUE. IN CASE THE SUBSCRIPT LIMIT IS CHANGED IS IT
C NECESSARY TO CHANGE THE VALUE OF THE VARIABLE N2 ACCORDINGLY.
      DIMENSION GG(3),DDD(3,3),GGC(3),GGS(3),DDDC(3,3),DDDS(3,3)
      real*8    gcx(3)
C
      EQUIVALENCE (CX(1,1),C(1)),(CX(1,2),V(1)),(CX(1,3),U(1)),
     *(CX(1,4),G(1)),(CX(1,5),P(1)),(CX(1,6),R(1)),(CX(1,7),SS1(1)),
     *(CX(2,8),SS2),(SIGMAX(1,1),SIGMA0(401)),(D(1),DD(1,1))
C K7 CONTAINS THE ORDER OF DIFFERENTIATION WITH RESPECT TO T,K8 THE
C ORDER OF DIFFERENTIATION WITH RESPECT TO THE RADIUS, CF.REF(A),TABLE
C 1. K9,K11,K13 THE KIND OF DIFFERENTIATIONS TO BE COMPUTED WITH RESPECT
C TO THE LATITUDE (2) AND THE LONGITUDE (3), CF.REF(A),SECTION 3. K15
C AND K17 CONTAINS AN INTEGER, WHICH WILL BE ADDED TO THE DEGREE. THE
C SUM WILL THEN BE MULTIPLIED WITH THE DEGREE-VARIANCE OF THE CORRESPON-
C DING DEGREE WHEN A FIRST AND/OR SECOND DIFFERENTIATION WITH RESPECT
C TO THE RADIAL DISTANCE HAS TAKEN PLACE.
C C11 CONTAIN QUANTITIES USED TO GIVE THE COVARIANCES THE PROPER UNITS.
C
      LTEST=LSPHAR.AND.ITCOUN.LT.5.AND.LTESTS
      IF (LTEST) THEN
      KI(35)=KI(35)+1
      END IF
      ITCOUN=ITCOUN+1
      LSPHP=LF
      LSPHQ=LF
      T = CR(1)
      HP = CR(2)
      HQ = CR(3)
      SP = CR(4)
      SQ = CR(5)
      CP = CR(6)
      CQ = CR(7)
      SD = CR(8)
      CD = CR(9)
c here RE is the mean radius of the Earth. Change 2004-08-13.
      RE=6371000.0d0
      RP = RE+HP
      RQ = RE+HQ
```



```
      RE2= RE**2
      ax=6378137.0d0
C
      KP=KI(6)
      KQ=KI(7)
C
C CHANGE 2003-03-22.
      LDGP=KP.EQ.3
      LDGQ=KQ.EQ.3
      IF (KP.EQ.17.OR.KQ.EQ.17) THEN
C     write(*,*)'10385 CX, KP,KQ,LSAT= ',KP,KQ,LSAT
C
      IF (KP.NE.17.AND.KQ.EQ.17) THEN
      LSPHQ=LT
      KPQ=KP
c     write(*,*)' LSAT ',LSAT,' KPQ=KP= ',KPQ
      CLAT=CP
      SLAT=SP
      SLON=SLOP
      CLON=CLOP
      if (lsphap) then
      RH=RE+HP
      else
      RH=ax+HP
      RE=ax
      RE2= RE**2
      end if
      GAMM=CR(11)
      END IF
C
      IF (KQ.NE.17.AND.KP.EQ.17) THEN
      LSPHP=LT
      KPQ=KQ
c     write(*,*)' LSAT ',LSAT,' KPQ=KP= ',KPQ
      CLAT=CQ
      SLAT=SQ
      SLON=SLOQ
      CLON=CLOQ
      if (lsphap) then
      RH=RE+HQ
      else
      RH=ax+HQ
      RE=ax
      RE2= RE**2
      end if
      GAMM=CR(10)
      END IF
C
      sq2=sqrt(d2)
      imax1=iideg+1
      i21=2*(iideg+1)
      if (iimax.lt.i21) write(*,*) ' imax too large '
C
      DO 501 I=1, i21
      ROOT0(I)= SQRT(DFLOAT(I-1))
  501 CONTINUE
C
      ILAST=(iMAX1)**2+1
C
      IF (KP.EQ.17.AND.KQ.EQ.17) THEN
C COV IS THE VARIANCE OF THE (I,J)'TH COEFFICIENT.
      COV=SIGMAP(IIDEG+1)
      if (LTSPH.AND.(IIDEG.GT.7.AND.IIDEG.LT.13))
     *write(*,*)' IIDEG+1, COV= ',IIDEG+1,COV
      ELSE
C
C SETTING ORDER OF DIFFERENTIATION.
C     IF  (KPQ.GE.1.OR.KPQ.LE.5) THEN   **  ERRONEOUS  **
C ERROR DETECTED 2000-03-27 BY CCT.
```

```
      IF  (KPQ.GE.1.AND.KPQ.LE.5) THEN
      IF (LSAT) THEN
      idif=1
      if (KPQ.eq.5) idif=2
      else
      idif=0
      end if
      ELSE
c     IF  (KPQ.GE.6.OR.KPQ.LE.11) THEN   ** ERRONEOUS  **
C ERROR DETECTED 2000-03-27 BY CCT.
      IF  (KPQ.GE.6.AND.KPQ.LE.9) THEN
      idif=1
      else
      idif=2
      end if
      end if
c
      CFA=D1
      iiold=-1
      jold=-1
      pim0=D1
      cjlo=d1
      sjlo=d0
      jmax1=abs(jjdeg)+1
      LCOS=jjdeg.ge.0
c
C SEE REF(H) EQ. (6).
      do 1002, j=1, jmax1
      do 1001, i=j, imax1
      call spharm(slat,clat,sjlo,cjlo,rh,i-1,j-1,idif,.true.,
     *lsphap,pii,pim0,pim1,pim2,dlp,dlp0,dlp1,dlp2,dap,dap0,
     *dap1,dap2,ddap,ddap0,ddap1,ddap2,ddal0,ddal1)
c
      if (i.gt.8.and.i.lt.13.and.j.eq.1.AND.LTSPH.and.i.eq.imax1)
     *then
      write(*,*)'i, sigmap, yc, KPQ ',i, sigmap(i),yc,KPQ
      end if
c     write(*,*)' ddc/s11 ',ddc(1,1),dds(1,1)
c
      wwc=sigmap(i)*yc
C UNITS OF M.
      if (idif.gt.0) then
      ggc(1)=sigmap(i)*gc(1)/RE
      ggc(2)=sigmap(i)*gc(2)/RE
      ggc(3)=sigmap(i)*gc(3)/RE
      gcx(3)=gc(3)
c     if (i.eq.4.and.j.eq.4.and.(abs(sjlo).lt.1.0d-7).and.(j.eq.jmax1)
c    *.and.(i.eq.imax1)) write(*,982)i,j,gcx(3),yc,sigmap(i)
c982  format(' spha ',2i3,3d15.6)
      IF (.not.LSAT) THEN
      ggc(1)=-ggc(1)*RADSEC/GAMM
      ggc(2)=-ggc(2)*RADSEC/GAMM
C UNITS OF ARCSEC.
C GRAVITY DISTURBANCE:
      ggc(3)=-ggc(3)*1.0D5
C UNITS OF MGAL USED.
C GRAVITY ANOMALY MISSING !!
      end if
      if (idif.gt.1) then
      do 990,kkc=1,3
      do 990,kkd=1,3
  990 dddc(kkc,kkd)=sigmap(i)*ddc(kkc,kkd)*1.0D9/RE2
C EU USED.
      end if
c
      end if
c
      wws=sigmap(i)*ys
      if (idif.gt.0) then
```



```
      ggs(1)=sigmap(i)*gs(1)/RE
      ggs(2)=sigmap(i)*gs(2)/RE
      ggs(3)=sigmap(i)*gs(3)/RE
      gcx(2)=gs(3)
      IF (.NOT.LSAT) THEN
C GRAVITY ANOMALY MISSING !! (SEE BELOW AT LABEL 1013).
      ggs(1)=-ggs(1)*RADSEC/GAMM
      ggs(2)=-ggs(2)*RADSEC/GAMM
C UNITS OF ARCSEC.
      ggs(3)=ggs(3)*1.0D5
C UNITS OF MGAL USED.
      end if
      if (idif.gt.1) then
      do 991,kkc=1,3
      do 991,kkd=1,3
  991 ddds(kkc,kkd)=sigmap(i)*dds(kkc,kkd)*1.0D9/RE2
c     write(*,*)' dddc/s11 ',dddc(1,1),ddds(1,1)
C EU USED.
      end if
c
      end if
c
      IF (.NOT.LSAT) THEN
       GO TO (1011,1012,1013,1014,1015,1016,1017,1018,1019,1020
     * ,1021,1024,1023,1022,1025),KPQ
C HEIGHT ANOMALY (M).
 1011  covc=wwc/GAMM
       covs=wws/GAMM
       GO TO 1126
C GRAVITY DISTURBANCE (MGAL).
 1012  covc= wwc*i/rh*1.0d5
       covs= wws*i/rh*1.0d5
       GO TO 1126
C GRAVITY ANOMALY (MGAL).
 1013  covc= wwc*(i-2)/rh*1.0d5
       covs= wws*(i-2)/rh*1.0d5
C
       if (i.gt.8.and.i.lt.13.and.j.eq.jmax1.AND.LTSPH
     * .and.i.eq.imax1) then
        write(*,1092) i,j,covc,ww,yc
 1092   format(' i,j,covc,ww,yc= ',2i3,3d14.6)
        LTSPH=LF
       end if
C
      GO TO 1126
C RADIAL DER. OF GRAVITY ANOMALY (EU).
 1014  covc=wwc*(i-2)*(i+1)/(rh*rh)*1.0d9
       covs=wws*(i-2)*(i+1)/(rh*rh)*1.0d9
       GO TO 1126
C VERTICAL GRAVITY GRADIENT.
 1015  covc=wwc*i*(i+1)/(rh*rh)*1.0d9
       covs=wws*i*(i+1)/(rh*rh)*1.0d9
C
       if (i.gt.8.and.i.lt.13.and.j.eq.jmax1.AND.LTSPH
     * .and.i.eq.imax1) then
        write(*,1091) i,i*(i+1),j,ww,covc
 1091   FORMAT(' i,i*(i+1),j,ww,covc= ',2i3,i4,2d14.6)
        LTSPH=LF
       end if
C
       GO TO 1126
C DEFLECTION, MERIDIAN COMP.
 1016  covc= ggc(2)
       covs= ggs(2)
       IF (LTEST) THEN
        write(*,*)' ggc,ggs1 ', covc,covs
       END IF
       GO TO 1126
C DEFLECTION, PRIME VERTICAL COMP.
```

```
 1017  covc= ggc(1)
       covs= ggs(1)
       IF (LTEST) THEN
        write(*,*)' ggc,ggs1 ', covc,covs
       END IF
       GO TO 1126
C PRIME VERTICAL DER. OF GRAVITY ANOMALY
C ERROR HERE !!!!!
 1019  covc=-ggc(2)*i/rh
       covs=-ggs(2)*i/rh
       GO TO 1126
C MERIDIAN DER. OF GRAVITY ANOMALY
 1018  covc=-ggc(1)*i/rh
       covs=-ggs(1)*i/rh
       GO TO 1126
C MERIDIAN DER. OF GRAVITY DISTURBANCE. CORR. 2000-03-27 BY CCT.
 1020  covc=-dddc(3,2)
       covs=-ddds(3,2)
       GO TO 1126
C PRIME VERTICAL DER. OF GRAVITY DISTURBANCE.
 1021  covc=-dddc(3,1)
       covs=-ddds(3,1)
       GO TO 1126
C 2. ORDER PRIME VERTICAL DER.
 1022  covc=dddc(1,1)
       covs=ddds(1,1)
       GO TO 1126
C MIXED PRIME VERTICAL & MERIDIAN DER. * 2. (TORSION BALANCE).
 1023  covc= dddc(1,2)*d2
       covs= ddds(1,2)*d2
       GO TO 1126
C 2. ORDER MERIDIAN COMP.
 1024  covc=dddc(2,2)
       covs=ddds(2,2)
       GO TO 1126
C DIFFERENCE 2. ORDER HORIZONTAL DER. (TORSION BALANCE).
 1025  covc=(dddc(1,1)-dddc(2,2))
       covs=(ddds(1,1)-ddds(2,2))
c
 1126  CONTINUE
       IF (LCOS) THEN
        covx(1,1,1,1)=covc
        cov=covc
       ELSE
        covx(1,1,1,1)=covs
        cov=covs
       END IF
C
      ELSE
       IF (LCOS) THEN
        ww=wwc
        DO 1875,IKK=1,3
         gg(IKK)=ggc(IKK)*1.0D5
         DO 1876,JKK=1,3
          ddd(IKK,JKK)=dddc(IKK,JKK)
 1876     CONTINUE
 1875    CONTINUE
       ELSE
        ww=wws
        DO 1877,IKK=1,3
         gg(IKK)=ggs(IKK)*1.0D5
         DO 1878,JKK=1,3
          ddd(IKK,JKK)=ddds(IKK,JKK)
 1878     CONTINUE
 1877    CONTINUE
       END IF
C
       NCASE=NDP+1+NDQ*3
C      write(*,*)NDP,NDQ,NCASE
```



```
       GO TO (1801,1802,1803,1804,1810,1810,1807,
     * 1810,1810),NCASE
C NO DERIVATIVES IN P OR Q.
1801   COVX(1,1,1,1)=ww/GAMM
       GO TO 1810
C 1 DERIVATIVE IN P, NONE IN Q.
1802   COVX(1,1,1,1)=gg(1)
       COVX(2,1,1,1)=gg(2)
       COVX(3,1,1,1)=gg(3)
C GRAVITY ANOMALY WITH GEOID. ADDED 1992.09.07.
       IF (LDGP) COVX(3,1,1,1)=gg(3)+D2*1.0d5*ww/RP
c     if (i.eq.4.and.j.eq.4.and.(abs(sjlo).lt.1.0d-7).and.(j.eq.jmax1))
c    *write(*,*)' g3111 ',gg(3),covx(3,1,1,1)
       GO TO 1810
C 2 DERIVATIVES IN P, NONE IN Q.
1803  COVX(1,1,1,1)= ddd(1,1)
      COVX(2,1,1,1)= ddd(2,1)
      COVX(1,2,1,1)=COVX(2,1,1,1)
      COVX(3,1,1,1)= ddd(3,1)
      COVX(1,3,1,1)=COVX(3,1,1,1)
      COVX(2,2,1,1)=ddd(2,2)
      COVX(2,3,1,1)= ddd(2,3)
      COVX(3,2,1,1)=COVX(2,3,1,1)
      COVX(3,3,1,1)= ddd(3,3)
      GO TO 1810
C NO DERIVATIVE IN P, 1 IN Q.
 1804 COVX(1,1,1,1)=gg(1)
      COVX(1,1,2,1)=gg(2)
      COVX(1,1,3,1)=gg(3)
C GRAVITY ANOMALY WITH GEOID. ADDED 1999.09.07, CORR 000.04.28.
      IF (LDGQ) COVX(1,1,3,1)=gg(3)+D2*1.0d5*ww/RQ
c     if (i.eq.4.and.j.eq.4.and.(abs(sjlo).lt.1.0d-7).and.(j.eq.jmax1))
c    * write(*,*)' g1131 ',gg(3),covx(1,1,3,1)
      GO TO 1810
C NO DERIVATIVE IN P, TWO IN Q.
 1807 COVX(1,1,1,1)= ddd(1,1)
      COVX(1,1,2,1)= ddd(2,1)
      COVX(1,1,1,2)=COVX(1,1,2,1)
      COVX(1,1,3,1)= ddd(3,1)
      COVX(1,1,1,3)=COVX(1,1,3,1)
      COVX(1,1,2,2)= ddd(2,2)
      COVX(1,1,3,2)= ddd(3,2)
      COVX(1,1,2,3)=COVX(1,1,3,2)
      COVX(1,1,3,3)= ddd(3,3)
c
 1810 CONTINUE
      COV=COVX(KSAT(KP,1),KSAT(KP,2),KSAT(KQ,1),KSAT(KQ,2))
      IF (LTEST.AND.ITCOUN.LT.20) THEN
       WRITE(*,*)' KSAT ',KSAT(KP,1),KSAT(KP,2),KSAT(KQ,1),
     * KSAT(KQ,2),' COV ',COV,'NDP,NDQ ',NDP,NDQ,' IDIF ',IDIF
      END IF
C
C this permits test of LSAT prediction of coefficients for
c LGRID = F. 2000-04-17.
      IF (j.eq.1) THEN
      IF (idif.eq.0) then
      covc=wwc
      else
      if (idif.eq.1) then
      covc=ggc(KSAT(KQ,1))*1.0D5
      else
      covc=dddc(KSAT(KQ,1),KSAT(KQ,2))
      end if
      end if
c
      else
      IF (idif.eq.0) then
      covc=wwc
      covs=wws
```

```
      else
      if (idif.eq.1) then
      covc=ggc(KSAT(KQ,1))*1.0D5
      covs=ggs(KSAT(KQ,1))*1.0D5
      else
      covc=dddc(KSAT(KQ,1),KSAT(KQ,2))
      covs=ddds(KSAT(KQ,1),KSAT(KQ,2))
      end if
      end if
      END IF
c
      end if
C
      IF (j.eq.1) THEN
      CCCIJ((i-1)**2+1)=COVC
      ELSE
      CCCIJ((i-1)**2+2*(j-1)+1)=COVS
c     ELSE
      CCCIJ((i-1)**2+2*(j-1))=COVC
      END IF
 1001 if (sigmap(i).gt.1.0d-10.and.LTEST)
     *write(*,511)i,j,covc,covc/sigmap(i),clat,cjlo,kp,kq,idif
  511 format(' i,j,covc,LLY,clat,cjlo,kp,kq= ',2i3,2d15.5,2f6.3,3i3)
C for each order of differentiation a different storage !
C
      sjl1=sjlo
      sjlo=sjlo*clon+cjlo*slon
      cjlo=cjlo*clon-sjl1*slon
 1002 continue
c
      end if
C
C CHANGE HERE TO TAKE CARE OF KSI, ETA 2000-05-02
      GO TO (2011,2011,2011,2011,2011,2016,2016,2018,2018,2020
     *,2020,2022,2024,2022,2024),KPQ
C EV, DERIVATIVES  Z, dgZ, ZZ.
 2011 IF (LCOS) THEN
      cv(1,1)=covc
      ELSE
      cv(1,1)=covs
      END IF
      GO TO 2026
C
C KSI, ETA.
 2016 IF (LCOS) THEN
      cv(1,1)=ggc(1)
      IF (LSPHP) THEN
      cv(1,2)=ggc(2)
      ELSE
      cv(2,1)=ggc(2)
      END IF
      ELSE
      cv(1,1)=ggs(1)
      IF (LSPHP) THEN
      cv(1,2)=ggs(2)
      ELSE
      cv(2,1)=ggs(2)
      END IF
      END IF
      GO TO 2026
C
C DELTAG, X, Y.
 2018 IF (LCOS) THEN
      cv(1,1)=-ggc(1)*i/rh
      IF (LSPHP) THEN
      cv(1,2)=-ggc(2)*i/rh
      ELSE
      cv(2,1)=-ggc(2)*i/rh
      END IF
```



```
      ELSE
      cv(1,1)=-ggs(1)*i/rh
      IF (LSPHP) THEN
      cv(1,2)=-ggs(2)*i/rh
      ELSE
      cv(2,1)=-ggs(2)*i/rh
      END IF
      END IF
      GO TO 2026
C
C XZ AND YZ.
 2020 IF (LCOS) THEN
      cv(1,1)=-dddc(3,1)
      IF (LSPHP) THEN
      cv(1,2)=-dddc(3,2)
      ELSE
      cv(2,1)=-dddc(3,2)
      END IF
      ELSE
      cv(1,1)=-ddds(3,1)
      IF (LSPHP) THEN
      cv(1,2)=-ddds(3,2)
      ELSE
      cv(2,1)=-ddds(3,2)
      END IF
      END IF
      GO TO 2026
C
C XX AND YY
 2022 IF (LCOS) THEN
      cv(1,1)=dddc(1,1)
      IF (LSPHP) THEN
      cv(1,2)=dddc(2,2)
      ELSE
      cv(2,1)=dddc(2,2)
      END IF
      ELSE
      cv(1,1)=ddds(1,1)
      IF (LSPHP) THEN
      cv(1,2)=ddds(2,2)
      ELSE
      cv(2,1)=ddds(2,2)
      END IF
      END IF
      GO TO 2026
C
C 2*XY and YY-XX.
 2024 IF (LCOS) THEN
      cv(1,1)=dddc(2,1)*d2
      IF (LSPHP) THEN
      cv(1,2)=dddc(2,2)-dddc(1,1)
      ELSE
      cv(2,1)=dddc(2,2)-dddc(1,1)
      END IF
      ELSE
      cv(1,1)=ddds(2,1)*d2
      IF (LSPHP) THEN
      cv(1,2)=ddds(2,2)-ddds(1,1)
      ELSE
      cv(2,1)=ddds(2,2)-ddds(1,1)
      END IF
      END IF
C
 2026 RETURN
      END IF
C
C IN HEIGH ALTITUDES AND WHEN LSUM IS TRUE WILL THE COVARIANCE BE COM-
C PUTED BY A SUMMATION OF THE LEGENDRE-SERIES ABBREVIATED TO DEGREE
C N2-1.
```

```
      LSUMC = LSUM .AND. (HP.GT.HMAX .OR. HQ.GT.HMAX)
C COMPUTATION OF THE CONSTANT USED TO CONVERT THE COVARIANCE INTO
C PROPER UNITS.
      CI(12) = CI(11)/(RP**KI(22)*RQ**KI(23)
     **CR(11)**KI(21)*CR(10)**KI(20))
C
      S = RB2/(RP*RQ)
C     IF(CI(10).LT.D0) S=D1-(RE*(HP+HQ+D2*(RE-CI(10)))+HP*HQ
C    *- (RE-CI(10))**2)/(RP*RQ)
      LOLDP = (KI(6).EQ.12) .OR. (KI(6).EQ.14) .OR. LSAT
      LOLDQ = (KI(7).EQ.12) .OR. (KI(7).EQ.14) .OR. LSAT
      IF (LSUMC) N1 = N3
C
C COMPUTATION OF THE QUANTITIES D(1)-D(36),CF.REF(A),SECTION 3.
C (MODIFIED ACCORDING TO REF.(C)).
C     if (.true.)write(*,*)' covcx nd=',ND
      DO I=1,36
       D(I)=D0
      END DO
      IF (ND.EQ.0) GO TO 55
C
      D(1) = D1
      CS = CP*SQ
      SC = SP*CQ
      SCC = SC*CD
      CC = CP*CQ
      CCS = CC*SD
      CSC = CS*CD
      IF (CI(20).GT.0.5) GO TO 201
C CF. REF.(D), EQ. (7) AND (8).
C ERROR 2002-10-06. CHANGE OF SIGN ON CI(17)*CI(19).
      D(2)= D2*(CI(17)*CI(19)+SP*CQ*CI(16))
      D(7)= D2*(-CI(17)*CI(19)+SQ*CP*CI(16))
      IF (ABS(D(2)-CS+SCC).GT.1.0D-6 .OR.
     * ABS(D(7)-SC+CSC).GT.1.0D-6) THEN
       WRITE(*,*)
     *' WARNING D(2) ',D(2),(CS-SCC)
       WRITE(*,*)
     *' WARNING D(7) ',D(7),(SC-CSC)
       write(*,*)ci(16),ci(17),ci(19)
      END IF
      GO TO 202
 201  D(2) = CS-SCC
      D(7) = SC-CSC
 202  CPSD = CP*SD
      CPCD = CP*CD
      CQSD = CQ*SD
      CQCD = CQ*CD
      D(3) = CQSD
      D(13)=-CPSD
C
      IF (ND.EQ.1) GO TO 55
      SS = SP*SQ
      D(8) = CC+SS*CD
C CF. REF.(D). EQ.(9).
      IF(CI(20).LT.0.5) THEN
       D(8)=CI(18)-D2*SP*SQ*CI(16)
       IF (ABS(D(8)-(CC+SS*CD)).GT.1.0D-6) THEN
        WRITE(*,*)' D(8) ',D(8),(cc+SS*CD)
        D(8)=-D(8)
       END IF
      END IF
      D(9) = -SQ*SD
      D(14)= SP*SD
      D(15)= CD
      IF (LOLDP) GO TO 91
      D(4) = D(2)+D(3)
      D(6) = D(3)-D(2)
      GO TO 92
```



```
   91 D(4) = -T
      D(6) = -CQCD/CP
   92 IF (LOLDQ) GO TO 93
      D(19)= D(13)+D(7)
      D(31)= D(13)-D(7)
      GO TO 94
   93 D(19)= -T
      D(31)= -CPCD/CQ
C
   94 IF (ND.EQ.2) GO TO 55
      IF (LOLDP) GO TO 95
      D(10) = D(9)+D(8)
      D(12) = D(9)-D(8)
      D(16) = D(15)+D(14)
      D(18) = D(15)-D(14)
      GO TO 96
   95 D(10) = -D(7)
      D(12) = SQ*CD/CP
      D(16) = CPSD
      D(18) = SD/CP
   96 IF (LOLDQ) GO TO 97
      D(20) = D(14)+D(8)
      D(32) = D(14)-D(8)
      D(21) = D(15)+D(9)
      D(33) = D(15)-D(9)
      GO TO 98
   97 D(20) = -D(2)
      D(21) = -CQSD
      D(32) = SP*CD/CQ
      D(33) = -SD/CQ
C
   98 IF (ND.EQ.3) GO TO 55
      IF (.NOT.(LOLDP.AND.LOLDQ)) GO TO 99
      D(22) = T
      D(24) = CQCD/CP
      D(34) = CPCD/CQ
      D(36) = CD/CC
      GO TO 55
   99 IF (.NOT.LOLDQ) GO TO 100
      D(22) = D(21)+D(20)
      D(24) = D(21)-D(20)
      D(34) = D(33)+D(32)
      D(36) = D(33)-D(32)
      GO TO 55
  100 D(22) = D(16)+D(10)
      D(34) = D(16)-D(10)
      D(24) = D(18)+D(12)
      D(36) = D(18)-D(12)
   55 CONTINUE
      if (.false.) write(*,1555)(D1-T),CI(20),CI(17),CR(8)
 1555 format(' t1,ci20,17,cr8',4d14.5)
C
      IF (LTABH) CALL TABH(C,LTA)
      IF (LTA.AND.LTABH) GO TO 204
      S2 = S*S
      ST = S*T
      T2 = T*T
      P2 = (D3*T2-D1)/D2
      P3 = (D3*ST+D1)/D2
C
C INITIALIZING ARRAY ELEMENTS. NOTE THE USE OF THE EQUIVALENCING.
      DO 50  K = 1, 8
      DO 50 M = 1, ND2
   50 CX(M,K) = D0
      DO 51 K = 1, ND2
      C(K) = D0
   51 DC(K) = D0
      DO 52 K = 1, 40
   52 CR(K+11) = D0
```

```
      Q(1)=D0
      RM(1)=D0
C
      IF (.NOT.LSAT) THEN
C
C SUMMATION AND DIFFERENTIATION OF THE LEGENDRE SERIES, CF.REF(A),EQ.
C (49) AND (51).
      IF (LSUMC) N1 = N2
      K1 = N1
      K2 = N1+1
      K = N1-1
      DO 54 M = 1, N1
      GI = (D2*K+D1)*S/K1
      GJ = -K1*S2/K2
      K2 = K1
      K1 = K
      K = K-1
      IF (.NOT.LSUMC) SI = SIGMA(IS+K2)
      IF (LSUMC) SI = SM(K2)
      I2 = 0
      I1 = 1
      DO 53 I = 2, ND2
      B = DC(I)
      DC(I) = C(I)
      C(I) = GI*(DC(I)*T+I2*DC(I1))+GJ*B+SI
      SI = D0
      I2 = I1
   53 I1 = I
   54 CONTINUE
      ltes0=(abs(t-1.0d0).lt.1.0d-6).and.ltest
      if (ltes0)  write(*,*)' C2 ',C(2),S
      IF (LSUMC) N1 = N3
C
C     IF (LSUMC) GO TO 75
      ELSE
       KP=KI(6)
       KQ=KI(7)
       LDGP=KP.EQ.3
       LDGQ=KQ.EQ.3
C
C INITIALIZING ARRAY ELEMENTS. NOTE THE USE OF THE EQUIVALENCING.
       DO 150  K = 1, 8
       DO 150 M = 1, 6
  150  CX(M,K) = D0
       NDTOT=NDP+NDQ+1
       DO 151 K = 1, 8
       DO 151 NDT=1,NDTOT
        CN(K,NDT) = D0
  151  DCN(K,NDT) = D0
C
C SUMMATION AND DIFFERENTIATION OF THE LEGENDRE SERIES, CF.REF(A),EQ.
C (49) AND (51).
       K1 = N1
       K2 = N1+1
       K = N1-1
       DO 154 M = 1, N1
        GI = (D2*K+D1)*S/K1
        GJ = -K1*S2/K2
        K2 = K1
        K1 = K
        K = K-1
        DO 154 NDT=1,5
         SI = SIGMAX(K2,NDT)
         I2 = 0
         I1 = 1
         DO 153 I = 2, NDX2(NDT)
          B = DCN(I,NDT)
          DCN(I,NDT) = CN(I,NDT)
          CN(I,NDT) = GI*(DCN(I,NDT)*T+I2*DCN(I1,NDT))+GJ*B+SI
```



```
          SI = D0
          I2 = I1
  153    I1 = I
  154   CONTINUE
C
       END IF
       IF (LSUMC) GO TO 75
C COMPUTATION OF THE FUNCTIONS L=R(1), N=1/RN, M=RM(2), F0=P(2), CF.
C REF.(A), EQ. (31)-(33),(40) AND (77A).
       RL2 = D1-D2*ST+S2
       RL =  SQRT(RL2)
       R(1) = RL
       RL1 = D1/RL
       RN = D1/(D1+RL-ST)
       RL2 = D1/RL2
       RNL = RN*RL1
       RM(2) = D1-RL-ST
       P(2) = S*DLOG(D2*RN)
       RL3 = RL2*RL1
       RL5 = RL3*RL2
       S3 = S2*S
       R(2) = -S*RL1
       IF (ND.EQ.0) GO TO 56
C
C COMPUTATION OF THE DERIVATIVES WITH RESPECT TO T.
C CF. REF.(A), EQ. (77B),(69A),(57).
       R(3) = -S2*RL3
       RM(3) = -R(2)-S
       P(3) = S2*(RNL+RN)
       IF (ND.EQ.1) GO TO 56
C
C CF. REF.(A), EQ. (77C),(69B),(58).
       R(4) = -D3*S3*RL5
       RM(4) = -R(3)
       P(4) = S3*(RL3+(D1+(D2+RL1)*RL1)*RN)*RN
       IF (ND.EQ.2) GO TO 56
C
C CF. REF.(A), EQ. (77D),(69C),(59).
       RL4 = RL2*RL2
       RL7 = RL5*RL2
       S4 = S2*S2
       R(5) = -15.0E0*S4*RL7
       RM(5) = -R(4)
       P(5) = S4*(D3*RL5+((D3+D3*RL1)*RL3+D2*(D1+(D3+(D3+RL1)*RL1)*RL1)
     * *RN)*RN)*RN
       IF (ND.EQ.3) GO TO 56
C
C CF. REF.(A), EQ. (69D),(60).
       S5 = S4*S
       RL6 = RL4*RL2
       RM(6) = -R(5)
       P(6) = D3*S5*((D5*RL7+((D4+D5*RL1)*RL5+((D4+(8.0E0
     * +D4*RL1)*RL1)*RL3+(D2+(8.0E0+(12.0E0+(8.0E0+D2*RL1)*RL1)
     * *RL1)*RL1)*RN)*RN)*RN)*RN)
C
   56  IF (LN(2)) GO TO 58
C COMPUTATION OF THE FUNCTION F-1 AND ITS DERIVATIVES, CF. REF.(A),
C EQ. (41) AND (61) - (65).
       U(2) = S*(RM(2)+T*P(2))
       IF (ND2.LT.3) GO TO 58
       DO 57 K = 3, ND2
   57  U(K) = S*(RM(K)+T*P(K)+(K-2)*P(K-1))
C
   58  IF (LN(1)) GO TO 60
C COMPUTATION OF THE FUNCTION F-2 AND ITS DERIVATIVES, CF. REF.(A) EQ.
C (42), AND (65)- (68).
       DO 59 K = 2, ND2
        GO TO (61,61,62,63,64,65),K
   61   CY = S*(D1-T2)/4.0E0
```

```
        GO TO 59
   62   CY = -ST/D2
        GO TO 59
   63   CY = D3*P(2)-S/D2
        GO TO 59
   64   CY = 9.0E0*P(3)
        GO TO 59
   65   CY = 18.0E0*P(4)
   59  V(K) = S*(RM(K)*P3+S*((K-2)*D3*RM(K-1)/D2+P2*P(K)+D3*T*P(K-1)*
     * (K-2)+CY))
C
   60  IF (LN(3)) GO TO 73
C COMPUTATION OF THE FUNCTION F1 AND ITS DERIVATIVES, CF. REF.(A) EQ.
C (36), REF.(B), EQ.(101) AND REF.(A), EQ.(70),(71).
       Q(2) = DLOG(D1+D2*S/(D1-S+RL))
       IF (ND.EQ.0) GO TO 66
       Q(3) = S2*RNL
       IF (ND.EQ.1) GO TO 66
       Q(4) = S3*((RL1+D1)*RN+RL2)*RNL
       IF (ND.EQ.2) GO TO 66
       Q(5) = S4*(D3*RL4+((D2+D3*RL1)*RL2+(D2 +(D4+D2*RL1)*RL1)*RN)
     * *RN)*RNL
       IF (ND.EQ.3) GO TO 66
       Q(6) = D3*S5*(D5*RL6+((D3+D5*RL1)*RL4+((D2+(6.0E0+D4*RL1)
     * *RL1)*RL2+(D2+(6.0E0+(6.0E0+D2*RL1)*RL1)*RL1)*RN)*RN)*RN)*RNL
C
C COMPUTATION OF THE FUNCTION F2 AND ITS DERIVATIVES, CF. REF.(A), EQ.
C (3),(72)-(75).
   66  P(2) = (RL-D1+T*Q(2))/S
       IF (ND.EQ.0) GO TO 68
       DO 67  K = 3, ND2
   67  P(K) = (R(K-1)+T*Q(K)+(K-2)*Q(K-1))/S
   68  I1 = II-1
       K1 = 1
       J1 = I1
       IF (I1.GE.2) GO TO 149
       DO 49 M = 2, ND2
        IF (I1.EQ.0) G(M) = Q(M)
        IF (I1.EQ.1) G(M) = P(M)
   49  CONTINUE
  149  IF (L(4)) J1 = JJ-1
       IF (J1.LE.1) GO TO 73
C
C CF. REF.(A), EQ. (38),(76).
       DO 71  K = 2, J1
        DO 69  M = 2, ND2
         B = Q(M)
         Q(M) = P(M)
   69   P(M) = (R(M-1)+(2*K-1)*((M-2)*Q(M-1)+T*Q(M))-K1/S*B)/(K*S)
        IF (K.NE.I1) GO TO 71
        DO 70 M = 2, ND2
   70   G(M) = P(M)
   71  K1 = K
C
   73  IF (LN(6)) GO TO 72
C CF. REF.(A), EQ. (34),(55).
       SS1(2) = S2*(T-S)*RL3
       IF (ND.GT.0) SS1(3) = S2*(RL3+D3*(T-S)*S*RL5)
C
C CF. REF.(A), EQ. (35).
   72  IF (L(7)) SS2= S2*((T+S)*RL3+D3*S*(T2-D1)*RL5)
C
   75  IF (.NOT.LSAT) THEN
C ADDING THE DIFFERENT TERMS, CF. REF.(A), EQ. (22),(47).
C TIPLIED BY RB**2 IN UNITS OF MGAL**2, THE INTEGERS K(2),K(3) OF EQ.
       DO 79 M = 2, ND2
C CF. REF.(A), EQ. (50),(52).
       C(M) = S*C(M)
       if (ltest.and.itcoun.lt.8)write(*,*)'CM',C(M),M
```



```
       CR(M*8 -4) = C(M)
       DO 78 K = 1, 7
       IF (LN(K)) GO TO 78
C STORING THE TERMS FOR TRANSFER TO THE CALLING PROGRAM USING THE COMMON
C AREA /CMCOV/.
       CR(M*8+K -4) = A*CX(M,K+1)*CI(K)
       IF (K.EQ.5) CR(M*8+K-4) = -CR(M*8+K-4)
       C(M) = C(M)+CR(M*8+K -4)
       if (ltest.and.itcoun.lt.8)
      *write(*,1)CX(M,K+1),CI(K),C(M),k,m,HP,HQ,RH
    1 format('CX,CI,C,K,NDT ',3E14.6,2I4,3f12.1,L2)
   78 CONTINUE
   79 CR(M+50)=C(M)
C
       ELSE
C
C FOR THIS SECTION SEE REF.(I) FOR ALL EQUATIONS.
        RP2=RP*RP
        RQ2=RQ*RQ
        RPQ=RQ*RP
        DO 178 NDT=1,5
        DO 178 M = 2, NDX2(NDT)
         CN(M,NDT)=CN(M,NDT)*S
         if (lf)
     *   write(*,*)'CM1',CN(M,NDT),M,NDT,S,LSAT
         DO 179 K = 1, 7
          IF (LNX(K,NDT))GO TO 179
          FAK5=D1
          IF (K.EQ.5) FAK5=-D1
          CN(M,NDT)=CN(M,NDT)+A*CX(M,K+1)*CIX(K,NDT)*FAK5
          if (ltest.and.itcoun.lt.8.and.k.eq.1)
     *    write(*,1)CX(M,K+1),CIX(K,NDT),CN(M,NDT),k,NDT
  179   CONTINUE
        CN(M-1,NDT)=CN(M,NDT)*(-1)**(NDT+1)
  178  CONTINUE
       IF (LTEST.AND.ITCOUN.LT.8)
     * WRITE(*,*)'NDP,NDQ,KP,KQ=',NDP,NDQ,KP,KQ
C
C WE NOW CALCULATE THE CROSS-COVARIANCES BETWEEN ALL QUANTI-
C TIES OF THE GIVEN ORDERS.
       NCASE=NDP+1+NDQ*3
       GO TO (801,802,803,804,805,806,807,808,809),NCASE
C NO DERIVATIVES IN P OR Q.
 801  COVX(1,1,1,1)=CN(1,1)
      if (ITCOUN.lt.5.and.ltest) write(*,*)'CN11 ',CN(1,1)
      GO TO 810
C 1 DERIVATIVE IN P, NONE IN Q. REF(I), EQ. (16) AND (17).
 802  COVX(1,1,1,1)=D(3)*CN(2,1)/RP
      COVX(2,1,1,1)=D(2)*CN(2,1)/RP
      COVX(3,1,1,1)=CN(1,2)/RP
C GRAVITY ANOMALY WITH GEOID. ADDED 1992.09.07.
      IF (LDGP) COVX(3,1,1,1)=(-CN(1,2)-D2*CN(1,1))/RP
      GO TO 810
C 2 DERIVATIVES IN P, NONE IN Q. REF(I), EQ. (24)-(28).
 803  COVX(1,1,1,1)=(D(3)*D(3)*CN(3,1)+CN(1,2)-T*CN(2,1))/RP2
      COVX(2,1,1,1)=D(2)*D(3)*CN(3,1)/RP2
      COVX(1,2,1,1)=COVX(2,1,1,1)
      COVX(3,1,1,1)=D(3)*(CN(2,2)-CN(2,1))/RP2
      COVX(1,3,1,1)=COVX(3,1,1,1)
      COVX(2,2,1,1)=(D(2)*D(2)*CN(3,1)-T*CN(2,1)+CN(1,2))/RP2
      COVX(2,3,1,1)=(D(2)*(CN(2,2)-CN(2,1)))/RP2
      COVX(3,2,1,1)=COVX(2,3,1,1)
      COVX(3,3,1,1)=CN(1,3)/RP2
      GO TO 810
C NO DERIVATIVE IN P, 1 IN Q. REF(I), EQ. (18), (19).
  804 COVX(1,1,1,1)=D(13)*CN(2,1)/RQ
      COVX(1,1,2,1)=D(7)*CN(2,1)/RQ
      COVX(1,1,3,1)=CN(1,2)/RQ
C GRAVITY ANOMALY WITH GEOID. ADDED 1992.09.07.
```

```
      IF (LDGQ) COVX(3,1,1,1)=(-CN(1,2)-D2*CN(1,1))/RQ
      GO TO 810
C 1 DERIVATIVE IN BOTH P AND Q. REF(I), EQ. (20)-(23).
  805 COVX(1,1,1,1)=(D(3)*D(13)*CN(3,1)+D(15)*CN(2,1))/RPQ
      COVX(2,1,1,1)=(D(2)*D(13)*CN(3,1)+D(14)*CN(2,1))/RPQ
      COVX(3,1,1,1)=D(13)*CN(2,2)/RPQ
      COVX(1,1,2,1)=(D(3)*D(7)*CN(3,1)+D(9)*CN(2,1))/RPQ
      COVX(2,1,2,1)=(D(2)*D(7)*CN(3,1)+D(8)*CN(2,1))/RPQ
      COVX(3,1,2,1)=D(7)*CN(2,2)/RPQ
      COVX(1,1,3,1)=D(3)*CN(2,2)/RPQ
      COVX(2,1,3,1)=D(2)*CN(2,2)/RPQ
      COVX(3,1,3,1)=CN(1,3)/RPQ
C GRAVITY ANOMALY WITH GRAVITY VECTOR AND GRAVITY. ADDED 1992.09.30.
      IF (LDGP.AND.(.NOT.LDGQ)) THEN
      COVX(3,1,1,1)=D(13)*(-CN(2,2)-D2*CN(2,1))/RPQ
      COVX(3,1,2,1)=D(7)*(-CN(2,2)-D2*CN(2,1))/RPQ
      COVX(3,1,3,1)=(-CN(1,3)-D2*CN(1,2))/RPQ
      END IF
      IF ((.NOT.LDGP.AND.LDGQ)) THEN
      COVX(1,1,3,1)=D(3)*(-CN(2,2)-D2*CN(2,1))/RPQ
      COVX(2,1,3,1)=D(2)*(-CN(2,2)-D2*CN(2,1))/RPQ
      COVX(3,1,3,1)=(-CN(1,3)-D2*CN(1,2))/RPQ
      END IF
      IF (LDGP.AND.LDGQ)
     *COVX(3,1,3,1)=(CN(1,3)+D4*(CN(1,2)+CN(1,1)))/RPQ
      GO TO 810
C 2 DERIVATIVES IN P, ONE IN Q. REF(I), EQ. (29)-(33).
  806 RP2Q=RP2*RQ
      CNX=CN(2,2)-T*CN(3,1)+D(3)*D(3)*CN(4,1)-CN(2,1)
      COVX(1,1,1,1)=(D(13)*CNX+D2*DD(3,3)*D(3)*CN(3,1))/RP2Q
      COVX(1,1,2,1)=(D(7)*CNX+D2*DD(3,2)*D(3)*CN(3,1))/RP2Q
      COVX(1,1,3,1)=(CN(1,3)+CN(1,2)+D(3)*D(3)*CN(3,2)-T*CN(2,2))/RP2Q
c     COVX(2,1,1,1)=(D(2)*D(3)*D(13)*CN(4,1)+D(17)*CN(2,1)
c    *     +(D(2)*D(15)+D(3)*D(14)+D(13)*D(7))*CN(3,1))/RP2Q
      COVX(2,1,1,1)=(D(2)*D(3)*D(13)*CN(4,1)
     *     +(D(2)*D(15)+D(3)*D(14))*CN(3,1))/RP2Q
c POSSIBLE ERROR 2002-10-29
      COVX(2,1,2,1)=(DD(2,2)*DD(3,1)*CN(3,1)+DD(2,1)*(DD(3,2)*CN(3,1)
     *     +DD(1,2)*DD(3,1)*CN(4,1)))/RP2Q
      COVX(2,1,3,1)=D(2)*D(3)*CN(3,2)/RP2Q
      COVX(3,1,1,1)=(DD(1,3)*DD(3,1)*(CN(3,2)-CN(3,1))+DD(3,3)
     *     *(CN(2,2)-CN(2,1)))/RP2Q
      COVX(3,1,2,1)=(DD(1,2)*DD(3,1)*(CN(3,2)-CN(3,1))+DD(3,2)
     *     *(CN(2,2)-CN(1,2)))/RP2Q
      COVX(3,1,3,1)=DD(3,1)*CN(2,3)/RP2Q
c     COVX(3,1,3,1)=DD(1,3)*CN(2,3)/RP2Q
      COVX(1,2,1,1)=COVX(2,1,1,1)
      COVX(1,2,2,1)=COVX(2,1,2,1)
      COVX(1,2,3,1)=COVX(2,1,3,1)
      CNX=CN(2,2)-T*CN(3,1)+D(2)*D(2)*CN(4,1)-CN(2,1)
      COVX(2,2,1,1)=(DD(1,3)*CNX+D2*D(2)*DD(2,3)*CN(3,1))/RP2Q
      COVX(2,2,2,1)=(DD(1,2)*CNX+D2*D(2)*DD(2,2)*CN(3,1))/RP2Q
      COVX(2,2,3,1)=(CN(1,3)+CN(1,2)+D(2)*D(2)*CN(3,2)-T*CN(2,2))/RP2Q
      CNX=DD(2,1)*(CN(3,2)-CN(3,1))
      COVX(2,3,1,1)=(DD(1,3)*CNX+DD(2,3)*(CN(2,2)-CN(2,1)))/RP2Q
      COVX(2,3,2,1)=(DD(1,2)*CNX+DD(2,2)*(CN(2,2)-CN(2,1)))/RP2Q
      COVX(2,3,3,1)=DD(2,1)*CN(2,3)/RP2Q
      COVX(1,3,1,1)=COVX(3,1,1,1)
      COVX(1,3,2,1)=COVX(3,1,2,1)
      COVX(1,3,3,1)=COVX(3,1,3,1)
      COVX(3,2,1,1)=COVX(2,3,1,1)
      COVX(3,2,2,1)=COVX(2,3,2,1)
      COVX(3,2,3,1)=COVX(2,3,3,1)
      COVX(3,3,1,1)=DD(1,3)*CN(2,3)/RP2Q
      COVX(3,3,2,1)=DD(1,2)*CN(2,3)/RP2Q
      COVX(3,3,3,1)=CN(1,4)/RP2Q
C GRAVITY ANOMALY ADDED 1992.09.30.
      IF (LDGQ) THEN
      COVX(1,1,3,1)=-(CN(1,3)+D3*CN(1,2)+D(3)*D(3)*(CN(3,2)+D2*CN(3,1))
```



```
     *-T*(CN(2,2)+D2*CN(2,1)))/RP2Q
      COVX(2,1,3,1)=-D(2)*D(3)*(CN(3,2)+D2*CN(3,1))/RP2Q
      COVX(3,1,3,1)=-DD(3,1)*(CN(2,3)+D2*CN(2,2))/RP2Q
C     COVX(3,1,3,1)=-DD(1,3)*(CN(2,3)+D2*CN(2,2))/RP2Q
      IF (LTEST) write(*,*)' COVX(3,1,3,1)= ',COVX(3,1,3,1)
      COVX(1,2,3,1)=COVX(2,1,3,1)
      COVX(2,2,3,1)=-(CN(1,3)+D3*CN(1,2)+D(2)*D(2)*(CN(3,2)+D2*CN(3,1))
     *-T*(CN(2,2)+D2*CN(2,1)))/RP2Q
      COVX(2,3,3,1)=-DD(2,1)*(CN(2,3)+D2*CN(2,2))/RP2Q
      COVX(1,3,3,1)=COVX(3,1,3,1)
      COVX(3,2,3,1)=COVX(2,3,3,1)
      COVX(3,3,3,1)=-(CN(1,4)+D2*CN(1,3))/RP2Q
      END IF
      GO TO 810
C NO DERIVATIVE IN P, TWO IN Q. REF(I), EQ. (24)-(28).
  807 COVX(1,1,1,1)=(CN(1,2)+D(13)*D(13)*CN(3,1)-T*CN(2,1))/RQ2
      COVX(1,1,2,1)=D(13)*D(7)*CN(3,1)/RQ2
      COVX(1,1,1,2)=COVX(1,1,2,1)
      COVX(1,1,3,1)=(D(13)*(CN(2,2)-CN(2,1)))/RQ2
C ERROR 2002-11-26.
C     COVX(1,1,3,1)=(D(3)*(CN(2,2)-CN(2,1)))/RQ2
      COVX(1,1,1,3)=COVX(1,1,3,1)
      COVX(1,1,2,2)=(CN(1,2)+D(7)*D(7)*CN(3,1)-T*CN(2,1))/RQ2
      COVX(1,1,3,2)=(D(7)*(CN(2,2)-CN(2,1)))/RQ2
      COVX(1,1,2,3)=COVX(1,1,3,2)
      COVX(1,1,3,3)=CN(1,3)/RQ2
      GO TO 810
C ONE DERIVATIVE IN P, TWO IN Q.  REF(I), EQ. (29)-(33).
  808 RPQ2=RP*RQ2
      CNX=CN(2,2)-T*CN(3,1)+D(13)*D(13)*CN(4,1)-CN(2,1)
      COVX(1,1,1,1)=(D(3)*CNX+D2*DD(3,3)*D(13)*CN(3,1))/RPQ2
      COVX(2,1,1,1)=(D(2)*CNX+D2*DD(2,3)*D(13)*CN(3,1))/RPQ2
      COVX(3,1,1,1)=(CN(1,3)+CN(1,2)+D(13)*D(13)*CN(3,2)
     *    -T*CN(2,2))/RPQ2
C ERROR CORRECTED 1992.09.04 BY CCT.
      COVX(1,1,2,1)=(D(7)*D(13)*D(3)*CN(4,1)+(D(7)*DD(3,3)+D(13)
     *    *DD(3,2))*CN(3,1))/RPQ2
C     COVX(2,1,2,1)=(DD(2,2)*DD(1,3)*CN(3,1)+DD(1,2)*(DD(3,2)*CN(3,1)
C CHANGE 2002-11-01.
      COVX(2,1,2,1)=(DD(2,2)*DD(1,3)*CN(3,1)+DD(1,2)*(DD(2,3)*CN(3,1)
     *    +DD(2,1)*DD(1,3)*CN(4,1)))/RPQ2
      COVX(3,1,2,1)=DD(1,2)*DD(1,3)*CN(3,2)/RPQ2
      COVX(1,1,3,1)=(DD(3,1)*DD(1,3)*(CN(3,2)-CN(3,1))+DD(3,3)
     *    *(CN(2,2)-CN(2,1)))/RPQ2
      COVX(2,1,3,1)=(DD(2,1)*DD(1,3)*(CN(3,2)-CN(3,1))+DD(2,3)
     *    *(CN(2,2)-CN(1,2)))/RPQ2
      COVX(3,1,3,1)=DD(1,3)*CN(2,3)/RPQ2
      COVX(1,1,1,2)=COVX(1,1,2,1)
      COVX(2,1,1,2)=COVX(2,1,2,1)
      COVX(3,1,1,2)=COVX(3,1,2,1)
      CNX=CN(2,2)-T*CN(3,1)+D(7)**2*CN(4,1)-CN(2,1)
      COVX(1,1,2,2)=(D(3)*CNX+D2*D(7)*DD(3,2)*CN(3,1))/RPQ2
      COVX(2,1,2,2)=(D(2)*CNX+D2*D(7)*DD(2,2)*CN(3,1))/RPQ2
      COVX(3,1,2,2)=(CN(1,3)+CN(1,2)+D(7)**2*CN(3,2)
     *    -T*CN(2,2))/RPQ2
      CNX=D(7)*(CN(3,2)-CN(3,1))
      COVX(1,1,3,2)=(D(3)*CNX+DD(3,2)*(CN(2,2)-CN(2,1)))/RPQ2
      COVX(2,1,3,2)=(D(2)*CNX+DD(2,2)*(CN(2,2)-CN(2,1)))/RPQ2
C POSSIBLE ERROR 1992.09.08.
      COVX(3,1,3,2)=D(7)*CN(2,3)/RPQ2
      COVX(1,1,1,3)=COVX(1,1,3,1)
      COVX(2,1,1,3)=COVX(2,1,3,1)
      COVX(3,1,1,3)=COVX(3,1,3,1)
      COVX(1,1,2,3)=COVX(1,1,3,2)
      COVX(2,1,2,3)=COVX(2,1,3,2)
      COVX(3,1,2,3)=COVX(3,1,3,2)
      COVX(1,1,3,3)=D(3)*CN(2,3)/RPQ2
      COVX(2,1,3,3)=D(2)*CN(2,3)/RPQ2
      COVX(3,1,3,3)=CN(1,4)/RPQ2
```

```
C GRAVITY ANOMALY ADDED 1992.09.30.
      IF (LDGP) THEN
       COVX(3,1,1,1)=-(CN(1,3)+D3*CN(1,2)+D(13)*D(13)*(CN(3,2)
     *   +D2*CN(3,1))-T*(CN(2,2)+D2*CN(2,1)))/RPQ2
C 2000-04-03
       COVX(3,1,2,1)=-DD(1,2)*DD(1,3)*(CN(3,2)+D2*CN(3,1))/RPQ2
C      COVX(3,1,3,1)=-DD(3,1)*(CN(2,3)+D2*CN(2,2))/RPQ2
       COVX(3,1,3,1)=-DD(1,3)*(CN(2,3)+D2*CN(2,2))/RPQ2
       COVX(3,1,1,2)=COVX(3,1,2,1)
       COVX(3,1,2,2)=-(CN(1,3)+D3*CN(1,2)+D(7)**2*(CN(3,2)
     *   +D2*CN(3,1))-T*(CN(2,2)+D2*CN(2,1)))/RPQ2
       COVX(3,1,3,2)=-D(7)*(CN(2,3)+D2*CN(2,2))/RPQ2
c      COVX(3,1,3,2)=-D(2)*(CN(2,3)+D2*CN(2,2))/RPQ2 cc 2000-04-05
       COVX(3,1,1,3)=COVX(3,1,3,1)
       COVX(3,1,2,3)=COVX(3,1,3,2)
       COVX(3,1,3,3)=-(CN(1,4)+D2*CN(1,3))/RPQ2
      END IF
      GO TO 810
C TWO DERIVATIVES IN BOTH P AND Q. REF(I), EQ. (34)-(46).
  809 R2PQ=RPQ**2
      D3132=D(3)**2+D(13)**2
      D313=D(3)*D(13)
      COVX(1,1,1,1)=(CN(1,3)+CN(1,2)-D2*T*CN(2,2)+D3132*CN(3,2)
     *  +T*CN(2,1)+CN(3,1)*(D2*(CD**2-D3132)+T2)
     *  -CN(4,1)*(D4*CD*SD**2*CP*CQ+T*D3132)
     *  +CN(5,1)*D313**2)/R2PQ
      COVX(2,1,1,1)=(D(2)*D(3)*(CN(3,2)+D(13)**2*CN(5,1)-T*CN(4,1))
     *  +CN(3,1)*D2*(-D(2)*D(3)+DD(2,3)*DD(3,3))
     *  +CN(4,1)*D2*(D313*DD(2,3)+D(2)*D(13)*DD(3,3)))/R2PQ
      CN23=CN(2,3)-CN(2,2)+CN(2,1)
      COVX(3,1,1,1)=(D(3)*(CN23+D(13)**2*(CN(4,2)-CN(4,1))
     *  +T*(CN(3,1)-CN(3,2)))+D2*D(13)*DD(3,3)*(CN(3,2)-CN(3,1)))/R2PQ
      COVX(1,2,1,1)=COVX(2,1,1,1)
      COVX(2,2,1,1)=(CN(1,3)+CN(1,2)-CN(2,2)*D2*T
     *  +CN(3,2)*(D(13)**2+D(2)**2)+CN(2,1)*T
     *  +CN(3,1)*(D2*(DD(2,3)**2-D(13)**2
     *  -D(2)**2)+T2)+CN(4,1)*(D4*D(2)*D(13)*DD(2,3)-T
     *  *(D(13)**2+D(2)**2))+D(13)**2*D(2)**2*CN(5,1))/R2PQ
      COVX(3,2,1,1)=(D(2)*(CN23
     *  +T*(CN(3,1)-CN(3,2))+D(13)**2*(CN(4,2)-CN(4,1)))
     *  +D2*D(13)*DD(2,3)*(CN(3,2)-CN(3,1)))/R2PQ
c suspected error 2002-10-07
c     *  +T*(CN(3,1)-CN(3,2))+D(13)**2*(CN(4,2)-CN(4,1))
c     *  +D2*D(13)*DD(2,3)*(CN(3,2)-CN(3,1)))/R2PQ
      COVX(1,3,1,1)=COVX(3,1,1,1)
      COVX(2,3,1,1)=COVX(3,2,1,1)
      COVX(3,3,1,1)=(CN(1,4)-T*CN(2,3)+D(13)**2*CN(3,3))/R2PQ
C
      COVX(1,1,2,1)=(D(7)*D(13)*(CN(3,2)+D(3)**2*CN(5,1)-T*CN(4,1))
     *  +CN(3,1)*D2*(-D(7)*D(13)+DD(3,2)*DD(3,3))
     *  +CN(4,1)*D2*(D313*DD(3,2)+D(7)*D(3)*DD(3,3)))/R2PQ
      COVX(2,1,2,1)=(CN(3,1)*(DD(2,3)*DD(3,2)+DD(2,2)*DD(3,3))
     *  +CN(4,1)*(DD(2,3)*D(3)*D(7)+DD(3,3)*D(2)*D(7)
     *  +DD(2,2)*D(3)*D(13)+DD(3,2)*D(2)*D(13))
     *  +CN(5,1)*D(2)*D(3)*D(7)*D(13))/R2PQ
C ERROR 2000-04-05.
      COVX(3,1,2,1)=(D(3)*D(13)*D(7)*(CN(4,2)-CN(4,1))
     *  +(D(13)*DD(3,2)+DD(3,3)*D(7))*(CN(3,2)-CN(3,1)))/R2PQ
C      COVX(3,1,2,1)=(D(3)*D(13)*D(7)*(CN(3,2)-CN(3,1))
C     *  +(D(13)*DD(3,2)+DD(3,3)*D(7))*(CN(2,2)-CN(2,1)))/R2PQ
       COVX(1,2,2,1)=COVX(2,1,2,1)
       COVX(2,2,2,1)=(D(7)*D(13)*(CN(3,2)+D(2)**2*CN(5,1))
     *  +CN(3,1)*D2*(DD(2,3)*DD(2,2)+D(13)
     *  *DD(4,2))+CN(4,1)*(D2*(D(7)*D(2)*DD(2,3)+D(2)*D(13)*DD(2,2))
     *  -D(7)*D(13)*T))/R2PQ
      COVX(3,2,2,1)=((D(8)*D(13)+D(7)*DD(2,3))*(CN(3,2)-CN(3,1))
     *  +D(7)*D(2)*D(13)*(CN(4,2)-CN(4,1)))/R2PQ
       COVX(1,3,2,1)=COVX(3,1,2,1)
       COVX(2,3,2,1)=COVX(3,2,2,1)
```



```
       COVX(3,3,2,1)=D(7)*D(13)*CN(3,3)/R2PQ
C
       COVX(1,1,3,1)= (D(13)*(CN23+D(3)**2*(CN(4,2)-CN(4,1))
     *  +T*(CN(3,1)-CN(3,2)))+D2*D(3)*DD(3,3)*(CN(3,2)-CN(3,1)))/R2PQ
       COVX(2,1,3,1)=((DD(3,3)*D(2)+DD(2,3)*D(3))*(CN(3,2)-CN(3,1))
     *  +D(3)*D(13)*D(2)*(CN(4,2)-CN(4,1)))/R2PQ
c      CN33=CN(3,3)-D2*CN(3,2)+CN(3,1)
       CN33=CN(3,3)-CN(3,2)+CN(3,1)
       COVX(3,1,3,1)=(D(3)*D(13)*CN33+DD(3,3)*CN23)/R2PQ
       COVX(1,2,3,1)=COVX(2,1,3,1)
       COVX(2,2,3,1)=(D(13)*(CN23
     *  +D(2)**2*(CN(4,2)-CN(4,1))
     *  +DD(4,1)*(CN(3,2)-CN(3,1)))
     *  +D2*D(2)*DD(2,3)*(CN(3,2)-CN(3,1)))/R2PQ
       COVX(3,2,3,1)=(DD(2,3)*CN23+D(2)*D(13)*CN33)/R2PQ
       COVX(1,3,3,1)=COVX(3,1,3,1)
       COVX(2,3,3,1)=COVX(3,2,3,1)
       COVX(3,3,3,1)=D(13)*(CN(2,4)-CN(2,3))/R2PQ
C
       COVX(1,1,1,2)=COVX(1,1,2,1)
       COVX(2,1,1,2)=COVX(2,1,2,1)
       COVX(3,1,1,2)=COVX(3,1,2,1)
       COVX(1,2,1,2)=COVX(1,2,2,1)
       COVX(2,2,1,2)=COVX(2,2,2,1)
       COVX(3,2,1,2)=COVX(3,2,2,1)
       COVX(1,3,1,2)=COVX(1,3,2,1)
       COVX(2,3,1,2)=COVX(2,3,2,1)
       COVX(3,3,1,2)=COVX(3,3,2,1)
C
       D37=D(3)**2+D(7)**2
       COVX(1,1,2,2)=(CN(1,3)+CN(1,2)+CN(2,2)*(-D2*T)
     *  +CN(3,2)*D37+CN(2,1)*T
     *  +CN(3,1)*(D2*(DD(3,2)**2-D37)
     *  +T2)+CN(4,1)*(D4*D(7)*D(3)*DD(3,2)-T
     *  *D37)+D(3)**2*D(7)**2*CN(5,1))/R2PQ
       COVX(2,1,2,2)=(D(2)*D(3)*(CN(3,2)+D(7)**2*CN(5,1))
     *  +CN(3,1)*D2*(DD(3,2)*DD(2,2)-D(3)*D(2))
     *  +CN(4,1)*(D2*(D(2)*D(7)*DD(3,2)+D(7)*DD(2,2)*D(3))
     *  +D(2)*D(3)*D(19)))/R2PQ
       COVX(3,1,2,2)=(D(3)*(CN23+D(7)**2*(CN(4,2)-CN(4,1))
     *  +DD(1,4)*(CN(3,2)-CN(3,1)))
     *  +D2*DD(3,2)*D(7)*(CN(3,2)-CN(3,1)))/R2PQ
       COVX(1,2,2,2)=COVX(2,1,2,2)
       D27=D(2)**2+D(7)**2
       COVX(2,2,2,2)=(CN(1,3)+CN(1,2)-D2*T*CN(2,2)+D27*CN(3,2)
     *  +T*CN(2,1)+(T2-D2*(D27-DD(2,2)**2))*CN(3,1)
     *  +(D4*D(8)*D(2)*D(7)-T*D27)*CN(4,1)
     *  +(D(2)*D(7))**2*CN(5,1))/R2PQ
       COVX(3,2,2,2)=(D(2)*(CN23+D(7)**2
     *  *(CN(4,2)-CN(4,1))-T*(CN(3,2)-CN(3,1)))
     *  +D2*D(7)*D(8)*(CN(3,2)-CN(3,1)))/R2PQ
       COVX(1,3,2,2)=COVX(3,1,2,2)
       COVX(2,3,2,2)=COVX(3,2,2,2)
       COVX(3,3,2,2)=(CN(1,4)+D(7)**2*CN(3,3)-T*CN(2,3))/R2PQ
C
       COVX(1,1,3,2)=(D(7)*(CN23
     *  +T*(CN(3,1)-CN(3,2))+D(3)**2*(CN(4,2)-CN(4,1)))
     *  +D2*D(3)*DD(3,2)*(CN(3,2)-CN(3,1)))/R2PQ
       COVX(2,1,3,2)=((D(8)*D(3)+D(2)*DD(3,2))*(CN(3,2)-CN(3,1))
     *  +D(7)*D(2)*D(3)*(CN(4,2)-CN(4,1)))/R2PQ
       COVX(3,1,3,2)=(DD(3,2)*CN23+D(3)*D(7)*CN33)/R2PQ
       COVX(1,2,3,2)=COVX(2,1,3,2)
       COVX(2,2,3,2)=(D(7)*(CN23+D(2)**2
     *  *(CN(4,2)-CN(4,1))-T*(CN(3,2)-CN(3,1)))
     *  +D2*D(2)*D(8)*(CN(3,2)-CN(3,1)))/R2PQ
       COVX(3,2,3,2)=(DD(2,2)*CN23+D(2)*D(7)*CN33)/R2PQ
       COVX(1,3,3,2)=COVX(3,1,3,2)
       COVX(2,3,3,2)=COVX(3,2,3,2)
       COVX(3,3,3,2)=D(7)*(CN(2,4)-CN(2,3))/R2PQ
```

```
C
      COVX(1,1,1,3)=COVX(1,1,3,1)
      COVX(2,1,1,3)=COVX(2,1,3,1)
      COVX(3,1,1,3)=COVX(3,1,3,1)
      COVX(1,2,1,3)=COVX(2,1,3,1)
      COVX(2,2,1,3)=COVX(2,2,3,1)
      COVX(3,2,1,3)=COVX(3,2,3,1)
      COVX(1,3,1,3)=COVX(1,3,3,1)
      COVX(2,3,1,3)=COVX(2,3,3,1)
      COVX(3,3,1,3)=COVX(3,3,3,1)
C
      COVX(1,1,2,3)=COVX(1,1,3,2)
      COVX(2,1,2,3)=COVX(2,1,3,2)
      COVX(3,1,2,3)=COVX(3,1,3,2)
      COVX(1,2,2,3)=COVX(1,2,3,2)
      COVX(2,2,2,3)=COVX(2,2,3,2)
      COVX(3,2,2,3)=COVX(2,3,3,2)
      COVX(1,3,2,3)=COVX(3,1,3,2)
      COVX(2,3,2,3)=COVX(3,2,3,2)
      COVX(3,3,2,3)=COVX(3,3,3,2)
C
      COVX(1,1,3,3)=(CN(1,4)-T*CN(2,3)+D(3)**2*CN(3,3))/R2PQ
      COVX(2,1,3,3)=D(2)*D(3)*CN(3,3)/R2PQ
      COVX(3,1,3,3)=D(3)*(CN(2,4)-CN(2,3))/R2PQ
      COVX(1,2,3,3)=COVX(2,1,3,3)
      COVX(2,2,3,3)=(CN(1,4)+D(2)**2*CN(3,3)-T*CN(2,3))/R2PQ
      COVX(3,2,3,3)=D(2)*(CN(2,4)-CN(2,3))/R2PQ
      COVX(1,3,3,3)=COVX(3,1,3,3)
      COVX(2,3,3,3)=COVX(3,2,3,3)
      COVX(3,3,3,3)=CN(1,5)/R2PQ
      if (lf) write(*,*)' cn15 ',cn(1,5),r2pq,covx(3,3,3,3),rp,rq
  810 END IF
C
  204 IF (.NOT.LSAT) THEN
C INTEGERS SPECIFYING THE KINDS OF DIFFERENTIATION WITH RESPECT TO THE
C LATITUDES AND/OR THE LONGITUDES, CF. REF.(A), SECTION 3.
      I = KI(10)
      J = KI(12)
      K = KI(11)
      M = KI(13)
      J1 = KI(14)
      M1 = KI(15)
      IF (.NOT.(LOLDP.OR.LOLDQ)) GO TO 110
C
      IJ = I+J
      IF (I.GT.3) IJ = 5
      KM = K+M
      IF (K.GT.3) KM = 5
C
C COMPUTATION OF THE DERIVATIVES OF ORDER ND WITH RESPECT TO THE LATI-
C TUDES AND THE LONGITUDES, CF. REF.(A), EQ. (43) - (46).
      GO TO (80,81,82,83,84),ND1
   80 COV = C(2)
      if (abs(cov).gt.1.0d5)write(*,*)' c2= ',cov
      GO TO 85
   81 COV = -C(3)*D(I+6*(K-1))
      GO TO 85
   82 COV = D(I)*D(J1)*D(6*(K-1)+1)*D(6*(M1-1)+1)*C(4)+D(IJ+6*(KM-1))
     **C(3)
      GO TO 85
   83 COV = (-D(IJ+6*(KM-1))*C(3)+(D(IJ)*D(6*(KM-1)+1)+D(I+6*(K-1))
     **D(J1+6*(M1-1))+D(I+6*(M1-1))*D(J1+6*(K-1)))*C(4)
     *+D(I)*D(J1)*D(6*(K-1)+1)*D(6*(M1-1)+1)*C(5))
      GO TO 85
   84 COV  = D(IJ+6*(KM-1))*C(3)+(D(IJ+6*(K-1))*D(6*(M-1)+1)
     *+D(I+6*(KM-1))*D(J)+D(J+6*(KM-1))*D(I)+D(IJ+6*(M-1))
     **D((K-1)*6+1)+D(IJ)*D(6*(KM-1)+1)+D(I+6*(K-1))*D(J+6*(M-1))
     *+D(I+6*(M-1))*D(J+6*(K-1)))*C(4)+(D(IJ)*D(6*(K-1)+1)*D(6*(M-1)+1)
     *+D(I+6*(K-1))*D(J)*D(6*(M-1)+1)+D(I+6*(M-1))*D(J)*D(6*(K-1)+1)
```



```
     *+D(J+6*(K-1))*D(I)*D(6*(M-1)+1)+D(J+6*(M-1))*D(I)*D(6*(K-1)+1)
     *+D(6*(KM-1)+1)*D(I)*D(J))*C(5)+D(I)*D(J)*D(6*(K-1)+1)*D(6*(M-1)
     *+1)*C(6)
C
C GIVING THE COVARIANCE THE PROPER UNITS.
   85 COV = COV*CI(12)
C
      GO TO 199
  110 CF=CI(12)
      IF (KI(6).EQ.13) CF=CF/D2
      IF (KI(7).EQ.13) CF=CF/D2
      DO 111 IX = 2, ND2
  111 CZ(IX-1) = C(IX)*CF
      CV(1,2) = D0
      CV(2,1) = D0
      CV(2,2) = D0
      GO TO (112, 113, 114, 115, 115), ND1
  112 CV(1,1) = CZ(1)
C  =============================================================
       KZ=1
      GO TO 198
  113 IF (I.EQ.1) GO TO 116
      CV(1,1) = CZ(2)*D(3)
      CV(2,1) = CZ(2)*D(2)
C  =============================================================
       KZ=2
      GO TO 198
  116 CV(1,1) = CZ(2)*D(13)
      CV(1,2) = CZ(2)*D(7)
C  =============================================================
       KZ=3
      GO TO 198
  114 IF (I.GT.1) GO TO 117
      CV(1,2) = CZ(3)*D(19)*D(31)
      CV(1,1) = CZ(3)*D(7)*D(13)*D2
C  =============================================================
      KZ=4
      GO TO 198
  117 IF (K.GT.1) GO TO 118
      CV(2,1) = CZ(3)*D(4)*D(6)
      CV(1,1) = CZ(3)*D(2)*D(3)*D2
C  =============================================================
      KZ=5
      GO TO 198
  118 CV(1,1) = CZ(2)*D(15)+CZ(3)*D(13)*D(3)
      CV(2,2) = CZ(2)*D(8) +CZ(3)*D(2)*D(7)
      CV(1,2) = CZ(2)*D(9) +CZ(3)*D(3)*D(7)
      CV(2,1) = CZ(2)*D(14)+CZ(3)*D(13)*D(2)
C  =============================================================
      KZ=6
C FIRST ORDER HORIZONTAL DERIVATIVES IN BOTH P AND Q.
      GO TO 198
  115 CONTINUE
C
      IIX=2
      DO 119 IX = 1, 2
      IIY=2
      DO 120 JX = 1, 2
      IF (ND.EQ.4) GO TO 121
C SECOND ORDER HORIZONTAL DERIVATIVE IN P OR Q.
      IX1=IX
      JX1=JX
      IF (KI(6) .GE. 12) GO TO 122
      CF = JX
      JX1=IIY
      I = J2(IX)
      J1 = 1
      K = I4(JX)
      M1 = I3(JX)
```

```
      GO TO 123
  122 CF = IX
      IX1=IIX
      I = I4(IX)
      J1 = I3(IX)
      K = J2(JX)
      M1 = 1
  123 K6 = 6*(K-1)
      M6 = 6*(M1-1)
      CV(IX1,JX1) = (CZ(3)*(D(I+K6)*D(J1+M6)+D(J1+K6)*D(I+M6))
     * +CZ(4)*D(I)*D(J1)*D(K6+1)*D(M6+1))*CF
C  ===================================================================
      KZ=7
      GO TO 120
  121 I = I4(IX)
      J = I3(IX)
      K = I4(JX)
      M = I3(JX)
      K6 = 6*(K-1)
      M6 = 6*(M-1)
      CV(IIX,IIY) = (CZ(3)*(D(I+K6)*D(J+M6)+D(I+M6)*D(J+K6))
     *  +CZ(4)*(D(J)*(D(I+K6)*D(M6+1)+D(I+M6)*D(K6+1))
     *        +D(I)*(D(J+K6)*D(M6+1)+D(J+M6)*D(K6+1)))
     *  +CZ(5)*D(I)*D(J)*D(K6+1)*D(M6+1))*IX*JX
C ===================================================================
      KZ=8
  120 IIY=1
  119 IIX=1
  198 COV = CV(KI(24),KI(25))
C ===================================================================
      IF (LTEST.AND.ITCOUN.LT.8.OR.ABS(CV(1,1)).GT.1.0d5)
     *WRITE(6,7788) KZ,I,J,K,M,CV(1,1),CV(1,2),CV(2,1),
     *CV(2,2)
 7788 FORMAT(/' KZ, I, J, K, M, CV(1,1), CV(1,2), ',
     *' CV(2,1) CV(2,2)'/1X,5I4,4F12.4)
  199 RETURN
      ELSE
       COV=COVX(KSAT(KP,1),KSAT(KP,2),KSAT(KQ,1),KSAT(KQ,2))
c change 2005-05-17.
       IF (KP.EQ.1) COV=COV/CR(10)
       IF (KQ.EQ.1) COV=COV/CR(11)
       IF (KP.EQ.15.AND.KQ.NE.15)
     * COV=COV-COVX(2,2,KSAT(KQ,1),KSAT(KQ,2))
C CHANGE, SO THAT UNITS ARE M, MGAL OR EU. 1992.08.26.
       IF (KP.EQ.6.OR.KP.EQ.7) THEN
        C11P=1.0D5
       ELSE
C CHANGE 2003-04-01.
C     C11P=C11(KP)/(CR(10)**K19(KP))
        C11P=C11(KP)
       END IF
       IF (KQ.EQ.6.OR.KQ.EQ.7) THEN
        C11Q=1.0D5
       ELSE
C     C11Q=C11(KQ)/(CR(11)**K19(KQ))
        C11Q=C11(KQ)
       END IF
       CFA=C11P*C11Q
       IF (KP.NE.15.AND.KQ.EQ.15)
     * COV=COV-COVX(KSAT(KP,1),KSAT(KP,2),2,2)
       IF (KP.EQ.15.AND.KQ.EQ.15)
     * COV=COV-COVX(1,1,2,2)-COVX(2,2,1,1)+COVX(2,2,2,2)
       COV=COV*CFA
C 2000-04-04.
       IF (ITCOUN.LT.5.and.ltest)WRITE(*,1791)
     * KSAT(KP,1),KSAT(KP,2),KSAT(KQ,1),
     * KSAT(KQ,2),COV,CFA,KP,KQ
 1791  format(' KSAT ',4i3,' COV ',D17.6,' CFA ',D17.6)
      END IF
```



```
      RETURN
      END
      SUBROUTINE INCOV(LINTER,RB,IMAX)
C PROGRAMMED BY C.C.TSCHERNING, GEOPHYSICAL INSTITUTE, UNIVERSITY
C OF COPENHAGEN, DENMARK.
C LAST UPDATE: 2001-09-21 BY CCT.
C THIS MODULE READS COVARIANCE FUNCTION PARAMETERS, CREATES NECESSARY
C TABELS FOR THE EVALUATION OF THE COVARIANCE FUNCTION.
C
      IMPLICIT NONE
      INTEGER MAXO,NSAT,NCTA,ICSYSL,NO,NAI,NLA,INL,IEM,INZOLD
     *,ICZERO,NCZERO,KCI,NC1,NC2,NI,NR,INDEX,IKP,ISAT,ISATP,NOBLK,
     *MAXB,IX,ICX,ITX,ITX1,ISX,IS1,IZ,IZ1,IZ2,IZ3,IMX,IM1,
     *IM2,IM3,IM4,IM5,IM6,II0,IOLD,NBOLD,IS,IPX,
     *IMAX1,IMAX1R,KTYPE,IK,IK1,I,IMAX,IMIN,MODEL1,NTA,NT1,NT2,
     *NTMAX1,NTMAX,NHE,NINTH,NFU,KEYH,J,NTABH,IOBS2,NSTART,IT,
     *MAXC1,MODEL,IC,IIP1,K2P3,K3,K4,NUM,INN,ITCOUN,IDIMC,IA1,
     *MAXC,IU1,NIR,IOBSR,INV,NT,INUMR,K21,K2,IU,
     *IA,IB,IH,IP,IB1,IP1,IC1,IC11,K1,IOBS1,
     *ITE,ITE1,IITE,IITE1,IIP,IIE,IIE1,INO,NO1,IANG,N,IT1,MAXC2,N1,
     *IOBS,ISO,IDIMCN,MAXBLT,II,NMAX,NFILTE,JR
C
      REAL*8 GM,RLAMAX,RLOMAX,RLAMIN,RLOMIN,
     *B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS,
c     *COSAZ,SINAZ,
     *SINLOP,COSLOP,BSIZE,BSIZEN,BSIZEE,COSLAP,
     *SINLAP,RLONGP,RP,CAZP,SAZP,
     *HP,RLATP,PRETAP,PREDP,HCZERO,CCI,CCR,SIGMA0,SIGMA,HCMAX,
     *CCV,DC,CTA,CTTF,CTSF,SZ,AZ,DXX,VARI,SCALE,SCALE2,
     *D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,
     *STEPN,COSSTN,SINSTN,STEPE,COSSTE,SINSTE,
     *COST2P,SINT2P,FILTER,STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE,
     *COST2Q,SINT2Q,S,SR,AAI,AAR,SM,DRAPP,DGPM2,SATROT,CNR,
     *SUMSIG,R,VARDG2,DR,RB,RB2,CVV,VZERO,A0,SIZEI,HTA,AX,GMP,
     *VG,RTA,TMAX,OLDB,VAR,gcx(3),REX

C
      PARAMETER (MAXO=5600,NSAT=5600,NCTA=1600)
C SMA PARAMETER (MAXO= 100,NSAT= 20,NCTA=100)
C 386 PARAMETER (MAXO= 250,NSAT= 20,NCTA=150)
      LOGICAL LTERRC,LNUOUT,LPOTIN,LINTER,LZERO,LMODEL,LTRAN,
     *LK31,LCLU7,LOPEN7,LTABLE,LTABLR,LOK,LF,LT,LOCAL,
     *LOPEN4,LTABH,LTIME,LTCOV,LOBSST,LMULTF,LCREF,LSUM,LWRSOL,
     *LSTOP,LIN4,LPUNCH,LNDAER,LKM,LC1,LC2,LDEFVP,
     *LONECO,LSTNO,LTERMA,LTERMO,LPARAM,
     *LPOSDA,LDEFF,LERNO,LCOMP,LCOM,LWLONG,
     *LDENOL,LMDD,LOPCOF,LBIPOT,LBICOV,LBISOL,LINSOL,LK30,
     *LNERNO,LOUTC,LNETAP,LNKSIP,LK2EQ4,LINT,LNEQ,
     *LNEWSO,LNEQ8,LCO1,LPRED,LPOT,LONEQ,LTEST
C
      CHARACTER*72 PNAME,OLDN,OLDCOV
C
      COMMON /CON1/OLDN(4)/CON2/GM,RLAMAX,RLOMAX,RLAMIN,RLOMIN,
     *ICSYSL,NO,NAI,NLA,INL,IEM,INZOLD,
     *LNDAER,LPOSDA,LDEFF,LERNO,LCOMP,LCOM,LWLONG,
     *LDENOL,LMDD,LIN4,LOPCOF,LCLU7,LOPEN4,LOPEN7,
     *LBIPOT,LBICOV,LBISOL,LINSOL,LTIME,LTCOV,LONEQ
C THESE VARIABLES HAVE BEEN PLACED IN COMMON, SO THAT THEY MAY BE
C INITIALIZED BY THE BLOCK DATA MODULE.
C
      COMMON/PR/B(MAXO),HQ(MAXO),RLAT(MAXO),SINLAT(MAXO),COSLAT(MAXO),
     *RLONG(MAXO),SINLON(MAXO),COSLON(MAXO),WOBS(MAXO),
C     *COSAZ(NSAT),SINAZ(NSAT),SINLOP,COSLOP,
     *SINLOP,COSLOP,
     *BSIZE(42),BSIZEN,BSIZEE,COSLAP,SINLAP,RLONGP,RP,CAZP,SAZP,
     *HP,RLATP,PRETAP,PREDP,HCZERO,ICZERO,NCZERO,
     *NI,NR,INDEX(42),IKP,ISAT(42),ISATP,NOBLK,
     *LONECO,LNKSIP,LNETAP,LDEFVP,LOBSST
C IN /PR/ IS STORED: THE CONSTANTS B(I), THE CATALOGUE OF THE OBSER-
```

```
C VATIONS (INDEX), LATITUDE, COS AND SIN OF LATITUDE, LONGITUDE AND
C HEIGHT OF OBSERVATION POINTS, THE CORRESPONDING QUANTITIES FOR THE
C PREDICTION POINT. THE LOGICAL VARIABLES ARE USED TO DISTINGUISH
C BETWEEN THE DIFFERENT PREDICTION SITUATIONS. THE COMMON BLOCK IS ALSO
C FOUND IN BLOCK DATA, PRED, OUTSOL AND INSOL.
C FOR A COMPLETE DESCRIPTION, SEE THE MAIN PROGRAM.
C
      COMMON /CMCOV/CCI(24),CCR(56),SIGMA0(1200),SIGMA(1200),HCMAX,
     *CCV(2,2),DC(36),KCI(37),NC1,NC2,LOCAL,LSUM
C COMMON VARIABLES USED IN COVAX.
C
      COMMON /TABELC/CTA(NCTA,16,2),CTTF(800),CTSF(20),SZ(30),AZ(18),
     *MAXB(20),IX(8),ICX,ITX,ITX1,ISX,IS1,IZ,IZ1,IZ2,IZ3,IMX,IM1,
     *IM2,IM3,IM4,IM5,IM6,II0,IOLD
C COMMON VARIABLES USED IN CTABEL AND COVCG.
C
      COMMON /CTABH/RTA(1200),HTA(5),TMAX,SIZEI,NFU(5),KEYH(5,5),
     *NINTH,NTABH(15,5,5),NHE,NSTART,LTABH
C COMMON VARIABLES USED IN INTABH AND TABH.
C
      COMMON/OUTC/INUMR(12),NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,
     *LTERMA,LTERMO,LSTNO,
     *LOUTC,LTRAN,LNERNO,LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT
C IN OUTC ARE STORED SUBSCRIPTS OF VARIABLES TO BE OUTPUT AND LIMITS
C FOR DO-LOOPS IN OUTPUT. NOTE THAT OUTC OCCURS IN SUBROUTINES
C HEAD, COUT, CXPARM AND THE BLOCK DATA MODULE.
C
      COMMON /CHEAD/IA,IB,IH,IP,IT,IA1,IB1,IP1,IT1,IC1,IC11,K1,IOBS1,
     *IOBS2,ITE,ITE1,IITE,IITE1,IIP,IIP1,IIE,IIE1,INO,
     *LPOT,LKM,LTERRC,LPOTIN
      COMMON /CHEAD1/LC1,LC2,LCREF
C IN /OUTC/ AND /CHEAD/ ARE STORED INFORMATION USED TO HANDLE THE DIF-
C FERENT I/O SITUATIONS.
      COMMON /COM2/DXX,NUM(70),VARI(32),SCALE,SCALE2,INN,INV
C USED BY COMPA, COMPARING OBSERVED AND PREDICTED QUANTITIES.
C
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,REX,RADSEC,PI,GMC,ITCOUN,
     *LTEST,LF,LT
C
      COMMON /CMEAN/STEPN,COSSTN,SINSTN,STEPE,COSSTE,SINSTE,
     *COST2P,SINT2P,FILTER(11),NFILTE
      COMMON /CMEAQ/STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE,
     *COST2Q,SINT2Q
C STEPSIZES USED WHEN CALCULATING MEAN VALUES.
C
      COMMON /BIPAR/OLDB(4),CNR,GMP,AX,NMAX,II,IOBS,IOBSR,N1,NIR,
     *MAXC,MAXC1,MAXC2,N,IC,NT,IDIMC,IDIMCN,MAXBLT,JR,ISO,
     *LPARAM,LPRED,LNEQ,LNEQ8,LNEWSO,LINT
      COMMON /BIPARC/OLDCOV(2),S,SR,AAI,AAR,NBOLD,IS,IPX,
     *IMAX1,IMAX1R,LTABLE,LTABLR,LCO1
C DATA USED WHEN STORING SOLUTIONS OR COVARIANCE FUNCTION ON
C BINARY FORM. (CHANGE MADE NOV 1986).
C
      DIMENSION SM(2001),DRAPP(181),DGPM2(201),SATROT(3,3)
      EQUIVALENCE (DRAPP(1),RTA(1)),(DGPM2(1),RTA(182))
C
        RE=6371000.0d0
        IF (LCREF) GO TO 1000
C
C *************** INPUT (6) *********************************
C
C INPUT OF THE INTEGER KTYPE DETERMINING TYPE OF DEGREE-VARIANCE
C MODEL USED FOR DEGREE-VARIANCES OF DEGREE GREATHER THAN IMAX
C (SEE BELOW). KTYPE MAY BE EQUAL TO 1, 2, OR 3, CORRESPONDING
C TO THE DEGREE-VARIANCE MODEL NUMBERS OF REF(A).
      IF (LINTER)WRITE(6,*)' INPUT DEGREE-VARIANCE MODEL NO. (1,2,3)'
  102 FORMAT(I2)
      READ(5,*)KTYPE
      IF (LWRSOL) WRITE(17,102)KTYPE
```



```
      KCI(5)=KTYPE
      IK=0
      IK1=0
      IF (LINTER)WRITE(6,*)' INPUT DENOMINATOR(S) IN MODEL'
      IF (KTYPE.LT.2) GO TO 1036
      IF (KTYPE.EQ.2) READ(5,*)IK
      IF (KTYPE.EQ.3) READ(5,*)IK,IK1
      IF (LWRSOL)WRITE(17,107)IK,IK1
  107 FORMAT(2I4)
      IF (KTYPE.LE.0 .OR. KTYPE.GE.4) STOP
C
 1036 KCI(3)=IK
      KCI(4)=IK1
      WRITE(6,141)
  141 FORMAT(/' THE MODEL ANOMALY DEGREE-VARIANCES ARE EQUAL TO'/,
     *' A*(I-1)')
      GO TO (1038,1039,1037),KTYPE
 1038 WRITE(6,143)
  143 FORMAT('+',8X,'/(I-2).')
      GO TO 1000
 1039 WRITE(6,144)IK
  144 FORMAT('+',8X,'/((I-2)*(I+',I4,')).')
      GO TO 1000
 1037 WRITE(6,142)IK,IK1
  142 FORMAT('+',8X,'/((I-2)*(I-',I4,')*(I-',I4,')).')
C
C
C THIS IS THE RETURN POINT AFTER THE FIRST COLLOCATION STEP IF
C A SECOND STEP IS WANTED. NOTE THAT THE SAME DEGREE-VARIANCE
C MODEL MUST BE USED, BUT R,VARDG2 AND IMAX MAY BE CHANGED.
C
 1000 CNR = D0
      DO 1035 I = 1, 300
 1035 SIGMA(I) = D0
C
      SUMSIG = D0
      MAXC1 = 1
C
C *************** INPUT (7) *********************************
C
C INPUT OF CONSTANTS USED FOR THE FINAL SPECIFICATION OF THE DEGREE-VAR-
C IANCE MODEL:
C R      - RATIO BETWEEN THE BJERHAMMAR-SPHERE RADIUS AND THE
C          MEAN RADIUS OF THE EARTH (RE), IF POSITIVE. IF NEGATIVE IT
C          IS THE DEPTH TO THE BJERHAMMAR SPHERE IN KM.
C VARDG2 - VARIANCE OF GRAVITY ANOMALIES AT ZERO ALTITUDE.
C IMAX   - MAXIMAL DEGREE FOR EMPIRICAL DEGREE-VARIANCES.
C LZERO  - TRUE IF ALL EMPIRICAL DEGREE-VARIANCES ARE ZERO.
C LTABLE - TRUE IF TABLE OF COVARIANCES IS TO BE USED.
C LMODEL - TRUE IF THE DEGREE-VARIANCES ARE (SCALED) ERROR DEGREE
C          VARIANCES OBTAINED FROM A GEOPOTENTIAL MODEL. THE VALUES
C          ARE FOR TWO MODELS FOUND IN THE BLOCK DATA MODULE.
C LTABH  - TRUE, IF COVARIANCE TABLES ARE USED WITH FIXED HEIGHTS
C          AND FUNCTIONALS.
C NOTE THAT LTABH AND LTABLE CAN NOT BE USED SIMULTANEOUSLY.
C THIS IS THEN FOLLOWED BY FURTHER DETAILS:
C (A) IF LMODEL TRUE, SPECIFICATION OF MODEL FOR THE VARIANCES.
C (B) IF LMODEL OR LZERO FALSE, THE EMPERICAL DEGREE-VARIANCES.
C (C) IF LTABLE IS TRUE, THEN TABLE SPECIFICATIONS.
C (D) IF LTABH IS TRUE THE TABLE AND FUNCTIONAL SPECIFICATIONS.
 1111 IF (LINTER)WRITE(6,1110)
 1110 FORMAT(' INPUT PARAMETERS DESCRIBING COV. FCT.',/
     *' R - NEG. DEPTH TO BJ.SPHERE IN KM OR RATIO RB/RE',/
     *' GRAVITY ANOMALY VARIANCE IN MGAL**2',/
     *' MAX. DEGREE OF LEGENDRE FCT. EXPANSION (E.G. 180, 360)',/
     *' LZERO - TRUE IF FIRST COEFF. ALL ARE ZERO',/
     *' LTABLE - TRUE IF COV.FCT. IS TABULATED IN 2D',/
     *' LMODEL - TRUE IF DEGREE-VAR. FROM PREDEFINED MODEL',/
     *' LTABH - TRUE IF 1D TABULATION',/,
```

```
     *' LSUM - IF FINITE LEGENDRE SERIES IS USED.')
      READ(5,*,ERR=1111)R,VARDG2,IMAX,LZERO,LTABLE,LMODEL,LTABH,LSUM
  101 FORMAT(F8.5,1X,F7.2,I4,5L2)
      IF (LWRSOL) WRITE(17,101)R,VARDG2,IMAX,LZERO,LTABLE,LMODEL,
     *LTABH,LSUM
      IF (R.GT.D1.OR.VARDG2.LT.D0.OR.LTABLE.AND.LTABH) STOP
      IMAX1=IMAX+1
      IF (LSUM) THEN
       WRITE(*,*)' INPUT NAME OF FILE WITH GRAVITY DEGREE-VARIANCES '
       READ(*,'(A)')PNAME
       WRITE(6,*)' DEGREE-VARIANCES INPUT FROM FILE ',PNAME
       OPEN(9,FILE=PNAME,STATUS='OLD')
       READ(9,*)(SIGMA0(I+1),I=IMIN,IMAX)
       IF (LWRSOL)WRITE(17,2103)PNAME
       CLOSE(9)
       VARDG2=D0
       DO I=3,IMAX
        VARDG2=VARDG2+SIGMA0(I+1)
       END DO
       SUMSIG=VARDG2
       CVV=D1
      END IF

C
      IF (R.GT.D0) S=RE*(R-D1)
      IF (R.LT.D0) S=R*1.0D3
      CCI(10)=S
      DR=S
      IF (R.LT.D0) R=(RE+S)/RE
      RB=S+RE
      RB2=(S+RE)**2
      AAI=RB2*1.0D-8
      CCI(8) = AAI
      LOCAL=LT
C     LSUM=LF
C CHANGE 2002.10.01
      IF (LSUM) THEN
       HCMAX=-1.0d5
      ELSE
      HCMAX = 1.0D6
C     HCMAX = 1.0D5
      END IF
      WRITE(*,1911)HCMAX
 1911 FORMAT(' HCMAX = ',F10.1,' M. ')
C THIS IMPLIES, THAT THE POSSIBILITY FOR USING THE SUMMATION OF
C THE LEGENDRE SERIES IN COVAX CAN NOT BE USED. IF THIS IS NEEDED
C CHANGE LSUM,HCMAX AND THE DIMENSION OF SM (TO E.G. 2000).
      NC1=IMAX1
      NC2=3
      IF (LSUM) GO TO 1002
      CALL COVAX(SM,IS)
C     CVV=VAR(SM,IS,3,S,AAI,D0,IMAX1,LF)
C ERROR DETECTED 1994.12.20 BY TK.
      CVV=VAR(SM,IS,3,S,AAI,D0,IMAX1,LF,1.0d0,0.0d0,LF,
     *   SATROT)
C
      LOCAL = LZERO
      IF (LZERO) WRITE(6,112)IMAX
  112 FORMAT( I4,' ERROR DEGREE-VARIANCES EQUAL TO ZERO')
      IF (LOCAL) GO TO 1040
      IF (.NOT.LMODEL) GO TO 1041
C
C --------------- INPUT (7A) ---------------------------------
C INPUT OF MODEL NUMBER, FIRST DEGREE TO BE USED AND SCALE FACTOR.
      IF (LINTER) WRITE(6,*)
     *' INPUT MODEL NO., START DEGR. & SCALE FACT.'
      READ(5,*)MODEL,IMIN,VG
C MODEL .LE. 0 INDICATES THAT THE DEGREE-VARIANCES ARE INPUT FROM A
C         FILE (PNAME), AND FILE NAME MUST BE INPUT SUBSEQUENTLY.
```



```
C MODEL 1 IS A MODEL FOR THE ERROR IN RAPP'S 1978 SET
C MODEL 2 IS THE ERROR DEGREE-VARIANCES FOR RAPP'S 1981 SET,
C MODEL 3 IS THE ERROR DEGREE-VARIANCES FOR WENZELS GPM2 SET.
C MODEL 4 IS A LINEAR MODEL IN THE DEGREE, SO THAT FOR VG=1.0 THE
C    THE ERROR DEGREE VARIANCE IS EQUAL TO 1.0 FOR DEGREE 100.
C MODEL 5 IS A SIMILAR, BUT QUADRATIC MODEL.
C FOR MODEL 2 AND 3 THE INITIALIZATION TAKES PLACE IN THE
C BLOCK DATA MODULE. CONSEQUENTLY THESE MODES CAN ONLY BE USED
C WHEN THE VARIABLES, WITH WHICH THEY ARE EQUIVALENCED (RLAT),
C HAVE NOT BEEN USED FOR SOMETHING ELSE ALREADY.
      IF (MODEL.EQ.1.OR.IC.LT.1218) GO TO 1050
C
      WRITE(6,117)
  117 FORMAT(' **** ERROR DEGREE-VARIANCES DESTROYED IN',
     *' FIRST COLLOCATION STEP **** ')
      STOP
C
 1050 IF (LWRSOL) WRITE(17,115)MODEL,IMIN,VG
  115 FORMAT(2I3,F9.6)
      WRITE(6,116)MODEL,IMIN,IMAX,VG
  116 FORMAT(' MODEL ',I3,' USED FROM DEGREE ',I3,' TO ',I3,
     *' WITH SCALE FACTOR= ',F9.6)
C ADDITION 1999-05-17 BY CCT.
      LMULTF=(MODEL.LT.0)
      IF (LMULTF) THEN
       MODEL=0
      END IF
C
      MODEL1=MODEL+1
      DO 1043 I = 2, IMAX
       SIGMA(I+1) = D0
       IF (I.LE.IMIN) GO TO 1043
       GO TO (1043,1051,1052,1053,1054,9955),MODEL1
 1051  SIGMA(I+1) = (2*I+1)*(VG*9.81)**2
       GO TO 1043
 1052  SIGMA(I+1) = VG*DRAPP(I+1)
       GO TO 1043
 1053  SIGMA(I+1) = VG*DGPM2(I+1)
       GO TO 1043
 1054  SIGMA(I+1) = I*1.0D-2*VG
       GO TO 1043
 9955  SIGMA(I+1) = I**2*1.0D-4*VG
C MODES 4 AND 5 ADDED 1988.11.30 BY CCT. MODEL 0, JAN. 1990.
 1043 CONTINUE
C
      IF (MODEL.NE.0) GO TO 1042
      IF (LINTER) WRITE(6,*)' INPUT NAME OF FILE WITH DEGR.VAR.'
      READ(*,'(A)')PNAME
      WRITE(6,*)' DEGREE-VARIANCES INPUT FROM FILE ',PNAME
      OPEN(9,FILE=PNAME,STATUS='OLD')
      READ(9,*)(SIGMA(I+1),I=IMIN,IMAX)
      IF (LWRSOL)WRITE(17,2103)PNAME
 2103 FORMAT(A72)
      CLOSE(9)
C CHANGE 1999-05-17 BY CCT:
      IF (LMULTF) THEN
       WRITE(*,*)' MULTIPLICATIVE FACTOR USED '
       DO I=IMIN,IMAX
        SIGMA(I+1)=SIGMA(I+1)*VG
       END DO
      ELSE
       WRITE(*,*)' INPUT VALUE FOR I=IMIN '
       READ(*,*)VZERO
       WRITE(*,1071)VZERO
 1071  FORMAT(' LINEAR FACTOR =',F8.4,' USED.')
       VZERO=VZERO/VG
       DO I=IMIN,IMAX
        SIGMA(I+1)=SIGMA(I+1)*VG*(VZERO+I/(IMAX-1))
       END DO
```

```
      END IF
C
      GO TO 1042
C
C --------------- INPUT (7B) ---------------------------------
C INPUT OF EMPIRICAL DEGREE-VARIANCES. NOTE, THAT PROBLEM MAY OCCUR
C IF FREE FORMAT IS USED, AND INPUT-DATA IS LINE NUMBERED. IN THIS
C CASE CHANGE TO FORMATTED INPUT.
 1041 CONTINUE
      IF (LINTER) WRITE(6,*)' INPUT DEGR. VARIANCES (MGAL**2)'
      READ(5,*) (SIGMA(I), I = 3, IMAX1)
      IF (LWRSOL) WRITE(17,98) (SIGMA(I), I = 3, IMAX1)
   98 FORMAT(8F8.2)
C NOTE THAT THE DEGREE-VARIANCE OF ORDER I IS STORED IN SIGMA(I+1).
C
      WRITE(6,111)IMAX
  111 FORMAT(I4,' EMPIRICAL ANOMALY DEGREE-VARIANCES FOR DEGREE',
     *' > 1,'/,' IN UNITS OF MGAL**2 : ')
      WRITE(6,98) (SIGMA(I), I = 3, IMAX1)
C
 1042 CONTINUE
      DO 1001 I = 3, IMAX1
      SIGMA0(IS+I)=SIGMA(I)
 1001 SUMSIG = SUMSIG + SIGMA(I)
 1040 IF (IMAX1+IS.LT.1200) GO TO 1002
      WRITE(6,108)
  108 FORMAT(' SUBSCRIPTS OF ARRAY SIGMA EXCEEDS ARRAY LIMIT, STOP.')
      STOP
C
 1002 AAI=(VARDG2-SUMSIG)*RB2*1.0D-8/CVV
      CCI(8)=AAI
      CALL COVAX(SM,IS)
      CALL COVBX(SM,LF,IS)
      CALL COVCX(SM,CVV,IS,LF,gcx,lt)
      IF ( ABS(CVV-VARDG2).GT.0.1) WRITE(6,7464)CVV,VARDG2,SUMSIG
 7464 FORMAT(' ** WARNING ** CVV,VARGD2= ',3E15.8)
C
C THE DEG.VAR. OF THE COVARIANCE FUNCTION OF THE ANOMALOUS POTENTIAL
C ARE STORED IN THE FIRST PART OF SIGMA (SUBSCRIPT 1 TO IMAX1R) FOR
C COLLOCATION I AND IN THE LAST PART (SUBSCRIPT IS=IMAX1R+3 TO
C IS+IMAX1) FOR COLLOCATION II.
C
  110 FORMAT(/' RATIO R/RE                =   ',F9.6,/
     *' DEPTH TO BJERHAMMAR SPHERE (R-RE)    = ',F10.2,' M'/
     *' VARIANCE OF POINT GRAVITY ANOMALIES   = ',F10.2,' MGAL**2'/
     *' THE FACTOR A, DIVEDED BY RE**2 IS     = ',F10.2,' MGAL**2')
      A0 = AAI*1.0D10/RE**2
      WRITE(6,110)R,DR,VARDG2,A0
      IF (LINTER) THEN
       WRITE(6,*)' ARE ALL PARAMETERS OK ?'
       READ(5,*)LOK
       IF (.NOT.LOK) GO TO 1111
      END IF
      IF (.NOT.LTABLE) GO TO 1055
C
C INITIALIZATION OF POINTERS FOR COVARIANCE FUNCTION TABLES.
C IPX HAS BEEN INITIALIZED TO -1 IN THE BLOCK DATA MODULE.
      IPX=IPX+1
      IZ1=3*IPX+1
      IZ2=IZ1+1
      IZ3=IZ2+1
      IM1=IPX*6+1
      IM2=IM1+1
      IM6=IM1+5
C
C --------------- INPUT (7C) ---------------------------------
C
C INPUT OF PARAMETERS DEFINING THE TABLE USED FOR FAST COMPUTATION
C OF COVARIANCES. MAXB(1) NUMBER OF STEPS IN HEIGHT, SZ(1) MINIMUM
```



```
C HEIGHT (M), (GENERALLY ZERO), SZ(2) MAXIMAL HEIGHT (M),
C MAXB(2) NUMBER OF INTERVALS WITH EQUIDISTANT STEPSIZE IN PSI. SZ(3)
C IS ZERO AS STARTING INTERVAL END-POINT, SZ(I+3) IS RIGHTMOST END-
C POINT OF I'TH INTERVAL (ARCSEC). MAXB(I+6) IS NUMBER OF EQUIDISTANT
C INTERVALS IN I'TH INTERVAL.
C PSI IS THE SPHERICAL DISTANCE.
C A TABLE WILL BE GENERATED BY CTABEL, WHICH MAKES A FAST COMPUTATION
C OF COVARIANCES OF TYPE (1,1), (1,6), (1,7), (3,3), (3,6), (3,7)
C POSSIBLE.
      IF (LINTER) WRITE(6,1112)
 1112 FORMAT(
     *' SPECIFY 2D TABULATION OF COVARIANCE FUNCTIONS:',/
     *' INPUT: NUMBER OF STEPS IN ALTITUDE, MIN. & MAX. HEIGHT (M)',/
     *'     NUMBER OF INTERVALS WITH EQUIDISTANT STEPSIZE IN PSI')
      READ(5,*)MAXB(IM1),SZ(IZ1),SZ(IZ2),NTA
      IF (LWRSOL) WRITE(17,50)MAXB(IM1),SZ(IZ1),SZ(IZ2),NTA
      MAXB(IM2)=NTA
C NTA+5 MUST BE LESS THAN 31, (DIMENSION OF SZ).
   50 FORMAT(I4,2F10.3,I4)
C
C INPUT OF RIGHT-MOST END POINT OF EACH INTERVAL IN UNITS OF ARC-
C SECONDS AND OF NUMBER OF EQUIDISTANT SUB-INTERVALS.
      IF (LINTER) WRITE(6,*)
     *' INPUT RIGHT-MOST ENDPOINTS OF EACH INTERVAL (ARCSEC)'
      READ(5,*)(SZ(I+IZ3),I=1,NTA)
   51 FORMAT(6F10.3)
      IF (LWRSOL) WRITE(17,51)(SZ(I+IZ3),I=1,NTA)
      IF (LINTER) WRITE(6,*)
     *' INPUT NUMBER OF SUB-INTERVALS IN EACH INTERVAL'
      READ(5,*)(MAXB(I+IM6),I=1,NTA)
   56 FORMAT(6I10)
      IF (LWRSOL) WRITE(17,56)(MAXB(I+IM6),I=1,NTA)
      SZ(IZ3)=D0
      NT1=1
      NT2=0
      NTMAX1=798
      NTMAX=NCTA
C NTMAX IS CURRENT MAXIMUM OF POINTS IN TABEL, I.E. NTMAX*8 IS
C THE DIMENSION OF CT. NTMAX1 IS THE UPPER LIMIT FOR NUMBER OF
C ENTRIES RELATED TO SPHERICAL DISTANCE (PSI), I.E. THE DIMENSION
C OF CTTF (COMMON BLOCK TABELC).
C
      DO 57 I=1,NTA
   57 NT1=NT1+MAXB(I+IM6)
      NT2=ITX+NT1
      IF (NT2.GT.NTMAX1.OR.(NT2*MAXB(IM1)).GT.NCTA)
     *WRITE(6,53)NT1,NT2
   53 FORMAT(' *** TOO LARGE TABEL REQUIRED *** NT1=',I5,' NT2=',I5)
      WRITE(6,52)MAXB(IM1),SZ(IZ1),SZ(IZ2)
   52 FORMAT(/' TABEL OF COVARIANCES GENERATED USING',/,
     *' NS=',I4,', HMIN=',F10.3,', HMAX=',F10.3,
     */,'  MAX-PSI (ARCSEC)  N-INTERVALS.')
      DO 198 I=1,NTA
  198 WRITE(6,55)SZ(I+IZ3),MAXB(I+IM6)
   55 FORMAT(1X,F10.3,8X,I10)
C
      IX(4+IPX*4)=IS
C IX(4) POINTS AT THE ZERO'TH SUBSCRIPT OF SIGMA, WHICH HERE IS IS.
      CALL CTABEL(IPX,LTCOV)
C
 1055 IF (.NOT.LTABH) GO TO 1056
C
C ------------------------ INPUT (7D) ----------------------------
C
C INPUT OF PARAMETERS SPECIFYING COVARIANCE FUNCTION TABLE USING
C FIXED HEIGHTS AND FUNCTIONALS.
C NHE  - NUMBER OF HEIGHTS (MAX 5)
C NINTH- NUMBER OF INTERVALS
C SIZEI- SIZE OF INTERVALS IN ARCSECONDS.
```

```
      IF (LINTER) WRITE(6,1113)
 1113 FORMAT(' INPUT NUMBER OF HEIGHTS, INTERVALS & SIZE IN ARCSEC')
      READ(5,*)NHE,NINTH,SIZEI
   41 FORMAT(2I4,F8.1)
      IF (LWRSOL) WRITE(17,41)NHE,NINTH,SIZEI
      WRITE(6,42)NINTH,SIZEI
   42 FORMAT(' NUMBER OF INTERVALS=',I4,', SIZE =',F9.1,' ARCSEC.',
     */,' HEIGHT (M)  FUNCTIONAL TYPES')
      IF (NINTH.GT.499) WRITE(6,43)
   43 FORMAT(' *** WARNING ***  NINTH DECREASED TO 499')
      IF (NINTH.GT.499) NINTH=499
      DO 44 I=1,NHE
C INPUT OF HEIGHT IN METERS AND NUMBER OF FIXED FUNCTIONALS.
      IF (LINTER) WRITE(6,1114)
 1114 FORMAT(' INPUT HEIGHTS (M) & NUMBER OF FUNCTIONALS')
      READ(5,*)HTA(I),NFU(I)
   45 FORMAT(F9.1,I3)
      IF (LWRSOL) WRITE(17,45)HTA(I),NFU(I)
C INPUT OF FUNCTIONAL TYPES (AS USED IN COVAX, I.E. IN
C GENERAL MUST 10 BE SUBTRACTED FROM THE NUMBERS USED HERE).
      IF (LINTER) WRITE(6,*)' INPUT FUNCTIONALS CODES-10'
      READ(5,*)(KEYH(J,I),J=1,NFU(I))
      IF (LWRSOL) WRITE(17,46)(KEYH(J,I),J=1,NFU(I))
      WRITE(6,47)HTA(I),(KEYH(J,I),J=1,NFU(I))
   47 FORMAT(F10.1,5I4)
   44 CONTINUE
   46 FORMAT(5I3)
C
      CALL INTABH(SM,IS,LTCOV)
C
 1056 RETURN
      END
      BLOCK DATA
C PROGRAMMED BY C.C.TSCHERNING, GEODETIC INSTITUTE, 1974.
C UPDATED: 2002-02-04 BY CCT.
C THE SUBROUTINE INITIALIZES A NUMBER OF VARIABLES. IT MAY BE
C SUBSTITUTED BY A "BLOCK DATA" CALL ON OTHER COMPUTERS.
C ON ICL-COMPUTERS, IT MUST HAVE A NAME, AND BE DECLARED AS
C AN EXTERNAL.
C
      IMPLICIT NONE
      INTEGER MAXO,NSAT,NCTA,MXPAR,NIPT,NIPCAT,MAXOD
     *,KCI,NC1,NC2,NFU,KEYH,NINTH,NTABH,NHE,IOBS2,NSTART,KSAT,
     *NDX1,NDX2,I4,NDP,IPACAT,NDQ,IT,K3,K4,NUM,INN,ITCOUN,
     *IGP,NBOLD,NWAR,IA1,IKP,IU1,IC11,IMAX1,IMAX1R,INV,ITE1,
     *ITIME0,KK,ITIME,INUMR,IP1,K21,K2,IU,IITE1,IITE,IIP1,IIP,IIE,
     *IIE1,K2P3,IT1,ITE,IP,IC1,IA,IB,NNX,NTABX,IFQ,ISATP,ISAT,
     *IHQ,IHP,INDEX,NR,NI,ICZERO,J2,K8,INZOLD,IEM,K21X,INL,K17,NAI,
     *K15,ICSYSL,K11,NO1,K9,K7,NO,IOBS1,IANG,IH,MP,IPAR,IFP,
     *KFQ,JR,NOBLK,IXX,K13,K19,NCZERO,NLA,INO,IB1,ISO,IPX,IS,
     *JJDEG,IIDEG,K1,IPTYPE,K23,I3,IPA,KFP,NPARM,NPARM1,MAXPAR,
     *II,NMAX,MAXB,IX,NCXLAS
C
      REAL*8 GM,RLOMAX,RLAMAX,RLOMIN,RLAMIN,B,HQ,RLAT,
     *SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS,SINLOP,
c     *COSAZ,SINAZ,
     *COSLOP,BSIZE,BSIZEN,BSIZEE,COSLAP,SINLAP,RLONGP,RP,CAZP,SAZP,
     *CCI,CCR,SIGMA0,SIGMA,HCMAX,CCV,D,OBS,OLDR,SLOQ,CFX,
     *RE,BIPC,CRHT,PREDP,HP,RLATP,BIP,HQOLD,C11,CTA,CTTF,CTSF,
     *SZ,AZ,HTA,TMAX,SIZEI,COVX,CIX,SLOP,D2,CLOP,CLOQ,GMC,PI,DXX,HCZERO,
     *VARI,DGPM2,SCALE,SCALE2,DRAPP,OLDT,RADSEC,CFA,SIGMAP,HPOLD,
     *D5,D0,D1,D3,D4,PRETAP

      PARAMETER (MAXO=5600,NSAT=5600,NCTA=1600,MXPAR=500,NIPT=500,
     *NIPCAT=100002,MAXOD=50400)
C
      LOGICAL L,LN,LOPEN7,LONECO,LNKSIP,LNETAP,LDEFVP,LSTOP,LRESOL,
     *LC1,LC2,LCREF,LKM,LNEQ,LT,LNDAER,LPOSDA,LDEFF,LF,LGRID,LERNO,
     *LDENOL,LNEWD,LPUNCH,LOUTC,LNERNO,LK30,LK31,LIN4,LOPCOF,LCLU7,
```



```
     *LFIRST,LSUM,LOCAL,LWRSOL,LPOT,LMDD,LCOMP,LCOM,LWLONG,LPRED,
     *LPARAM,LTERRC,LPOTIN,LK2EQ4,LNUOUT,LTABLE,LTABLR,LNEQ8,LNEWSO,
     *LINT,LTERMA,LTERMO,LTERM,LCO1,LBIPOT,LBICOV,LBISOL,LINSOL,
     *HP9000,LOPEN4,LTABH,LTIME,LTCOV,LONEQ,LX,LNX,LTESTS,LOBSST
     *,LCOERR,LSPOUT,LTRAN,LLCOER,LTEST
C
      CHARACTER*72 OLDN,OLDCOV
      COMMON /CON1/OLDN(4)/CON2/GM,RLOMAX,RLAMAX,RLOMIN,RLAMIN,
     *ICSYSL,NO,NAI,NLA,INL,IEM,INZOLD,
     *LNDAER,LPOSDA,LDEFF,LERNO,LCOMP,LCOM,LWLONG,
     *LDENOL,LMDD,LIN4,LOPCOF,LCLU7,LOPEN7,LOPEN4,
     *LBIPOT,LBICOV,LBISOL,LINSOL,LTIME,LTCOV,LONEQ
C
      COMMON/PR/B(MAXO),HQ(MAXO),RLAT(MAXO),
     *SINLAT(MAXO),COSLAT(MAXO),RLONG(MAXO),
     *SINLON(MAXO),COSLON(MAXO),WOBS(MAXO),
C    *COSAZ(NSAT),SINAZ(NSAT),SINLOP,COSLOP,BSIZE(42),BSIZEN,BSIZEE,
     *SINLOP,COSLOP,BSIZE(42),BSIZEN,BSIZEE,
     *COSLAP,SINLAP,RLONGP,RP,CAZP,SAZP,
     *HP,RLATP,PRETAP,PREDP,HCZERO,ICZERO,NCZERO,
     *NI,NR,INDEX(42),IKP,ISAT(42),ISATP,NOBLK,
     *LONECO,LNKSIP,LNETAP,LDEFVP,LOBSST
C FOR A COMPLETE DESCRIPTION, SEE THE MAIN PROGRAM.
C
      COMMON /CMCOV/CCI(24),CCR(56),SIGMA0(1200),SIGMA(1200),HCMAX,
     *CCV(2,2),D(36),KCI(37),NC1,NC2,LOCAL,LSUM
      COMMON /DDX/K7(17),K9(17),K11(17),K13(17),K15(17),K17(17),
     *K19(17),K21X(17),K23(17),K8(17),C11(17),J2(2),I3(2),I4(2),
     *LN(7),L(7)
C
      COMMON /TABELC/CTA(NCTA,16,2),CTTF(800),CTSF(20),SZ(30),AZ(18),
     *MAXB(20),IX(8),IXX(18)
      COMMON /CTABH/DRAPP(181),DGPM2(201),CRHT(818),HTA(5),TMAX,
     *SIZEI,NFU(5),KEYH(5,5),NINTH,NTABH(15,5,5),NHE,NSTART,LTABH
      COMMON /CTABH1/HPOLD,HQOLD,IHP,IHQ,KFP,KFQ,IFP,IFQ,NTABX,NNX
C
      COMMON/DAT/LNEWD,LRESOL,LGRID
C
      COMMON/OUTC/INUMR(12),NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,
     *LTERMA,LTERMO,LTERM,
     *LOUTC,LTRAN,LNERNO,LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT
      COMMON /CHEAD/IA,IB,IH,IP,IT,IA1,IB1,IP1,IT1,IC1,IC11,K1,IOBS1,
     *IOBS2,ITE,ITE1,IITE,IITE1,IIP,IIP1,IIE,IIE1,INO,
     *LPOT,LKM,LTERRC,LPOTIN
      COMMON /CHEAD1/LC1,LC2,LCREF
      COMMON /COM2/DXX,NUM(70),VARI(32),SCALE,SCALE2,INN,INV
C
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,
     *LTEST,LF,LT
C
      COMMON /GPOTC1/OLDT,OLDR,CFA,IGP(12),LFIRST,HP9000
      COMMON /OBSER/OBS(22)
C
      COMMON/CPARM/IPTYPE(NIPT),IPACAT(3*NIPT),
     *NPARM,NPARM1,MAXPAR,MP,IPA,NCXLAS
      COMMON/CCTIME/ITIME(NIPCAT),ITIME0(NIPT),LCOERR,LLCOER
C
      COMMON /CCVCG/KK(24)
C
      COMMON /BIPAR/BIP(7),NMAX,II,IPAR(13),JR,ISO,LPARAM,LPRED,
     *LNEQ,LNEQ8,LNEWSO,LINT
      COMMON /BIPARC/OLDCOV(2),BIPC(4),NBOLD,IS,IPX,IMAX1,IMAX1R,LTABLE,
     *LTABLR,LCO1
      COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFX,KSAT(17,2),
     *NDX1(5),NDX2(5),NDP,NDQ,NWAR,LX(7,5),LNX(7,5),LTESTS
      COMMON /PDEGV/SIGMAP(2001),SLOP,SLOQ,CLOP,CLOQ,
     *IIDEG,JJDEG,LSPOUT
C
C KSAT HOLDS THE MAPPING BETWEEN THE DATA CODES AND THE POSITIONS
```

```
c IN THE ARRAY COVCX HOLDING THE COVARIANCES. SEE SUBROUTINE COVCX.
      DATA KSAT/
     *1,3,3,3,3,2,1,2,1,2,1,2,1,1,1,1,1,
     *1,1,1,3,3,1,1,1,1,3,3,2,2,1,1,1,1/
      DATA KK,KFP,HPOLD,HQOLD/1,5,2,5,5,3,4,9*5,1,3,9,11,0,1,4,5,
     *-1,2*-1.0D5/,KCI(26),KCI(27),KCI(28),KCI(29),KCI(30),KCI(31),
     *KCI(32),KCI(33),CCI(14)/1,0,1,0,2,0,-1,1,0.5/
     *,KCI(35),KCI(36),KCI(37)/3*0/
C INITIALIZING FOR DENSITY CONTRAST COMPUTATION.
C
      DATA
     *K7/5*0,6*1,4*2,2*0/,K9/5*1,2,3,2,3,2,3,2,2,3,4,2*0
     */,K11/11*0,2,3,3,6,2*0/,
     *K13/11*1,2,3,3,6,2*0/,K15/0,1,-1,-1,1,0,0,-1,-1,2,2,6*0/,
     *K17/3*0,2,2,12*0/,K19/1,4*0,1,1,10*0/,K21X/0,1,1,2,2,1,1,10*2/,
     *K23/5*1,2,1,2,1,2,1,1,1,2,2,0,0/,
     *K8/0,1,1,2,2,0,0,4*1,6*0/,C11,
     *HCMAX/1.0D0,2*1.0D5,2*1.0D9,2*-206264.806D0,5*1.0D9,2.0D9,
     *2*1.0D9,2*1.0D0,1.0D9
     */,D,BIP,BIPC/47*0.0D0/,J2/3,2/,I3/6,3/,I4/4,2/
C
      DATA RE,GMC,D1,D2,D3,D4,D5,
     *D0,BSIZE,SIGMA,SIGMA0,OLDT,OLDR,PREDP,PRETAP,OBS,
     *RADSEC,PI/6371.0D3,3.98D14,1.0D0,2.0D0,3.0D0,4.0D0,
     *5.0D0,2469*0.0D0,206264.806D0,3.1415926535D0/,LT,LNEQ,LSPOUT,
     *LNDAER,LCO1,LNERNO,LWRSOL,LBIPOT,LBICOV,LBISOL,LINSOL,LTABH,
     *LDENOL,LPOSDA,LFIRST,LCREF,LC1,LC2,LDEFF,LMDD,LIN4,LOPCOF,
     *LF,LGRID,LERNO,LCOMP,LCOM,LWLONG,LPRED,LCLU7,LOPEN7,LRESOL
     *,LTIME,LTCOV,LONEQ
     *,LTERRC,LTABLE,LTABLR,LNEQ8,LOPEN4/6*.TRUE.,34*.FALSE./,RLAMAX,
     *RLOMAX,RLAMIN,RLOMIN,HCZERO,ICSYSL,NCZERO/4*0.0D0,-3.0D8,-2,-1/,
     *ITCOUN,IPAR,NBOLD,ITE,ITE1,INZOLD,
     *IX,NPARM,NO,NAI,NLA,IS,ISO,IGP,IT,IP,INDEX(1),INDEX(2),
     *IA,IA1,INL,IEM,II,JR,NPARM1/48*0,2*9,10,25,2*2,1/
     *,IXX,IPX,NWAR/0,0,1,0,1,0,1,2,3,0,1,2,3,4,5,6,0,-2,-1,0/
C THE FACTOR MUST BE 2*NIPT.
C CHANGE 2002-02-05. DIMENSION OF ITIME NOW NIPCAT.
      DATA IPTYPE,ITIME0/NIPT*0,NIPT*0/ITIME/NIPCAT*0/
     *LCOERR,LLCOER/2*.FALSE./
C
C INITIALIZING VARIABLES IN /PR/ TO HOLD ERROR DEGREE-VARIANCES FOR
C POTENTIAL COEFFICIENT SOLUTIONS RAPP 1981 AND WENZELS GPM2.
      DATA DRAPP/
     *0.0,0.0,0.000006,0.0002455,0.0007523,0.005981,
     *0.007083,0.0264,0.03313,0.07616,0.06944,
     *0.128,0.095,0.135,0.135,0.140,0.177,0.168,0.185,0.203,0.195,
     *0.194,0.222,0.226,0.254,0.277,0.289,0.263,0.301,0.298,0.286,
     *0.280,0.293,0.274,0.283,0.292,0.304,0.183,0.188,0.193,0.198,
     *0.203,0.208,0.213,0.218,0.223,0.228,0.234,0.238,0.243,0.250,
     *0.254,0.259,0.266,0.271,0.276,0.282,0.287,0.291,0.297,0.303,
     *0.307,0.314,0.319,0.323,0.327,0.337,0.341,0.346,0.355,0.357,
     *0.362,0.371,0.376,0.385,0.387,0.392,0.398,0.406,0.410,0.418,
     *0.427,0.440,0.441,0.444,0.448,0.465,0.470,0.471,0.477,0.482,
     *0.496,0.495,0.510,0.511,0.514,0.521,0.532,0.545,0.541,0.555,
     *0.547,0.574,0.595,0.584,0.595,0.603,0.613,0.618,0.634,0.642,
     *0.641,0.650,0.652,0.657,0.702,0.681,0.692,0.694,0.710,0.730,
     *0.711,0.719,0.741,0.768,0.776,0.762,0.802,0.760,0.790,0.811,
     *0.776,0.835,0.844,0.824,0.859,0.869,0.864,0.907,0.884,0.914,
     *0.899,0.887,0.942,0.956,0.943,0.935,0.951,0.981,0.972,0.986,
     *1.009,1.015,1.018,1.021,1.049,1.095,1.060,1.072,1.104,1.117,
     *1.118,1.143,1.134,1.228,1.280,1.184,1.164,1.241,1.234,1.233,
     *1.270,1.230,1.268,1.361,1.320,1.297,1.323,1.352,1.450,1.363/
C
      DATA DGPM2/0.0,0.00,0.000030,0.000139,0.000208,0.000262,
     *0.000307,0.000338,0.000386,0.000422,0.000457,
     *0.000,13*0.001,0.002,0.002,0.002,0.002,0.002,0.003,
     *0.003,0.003,0.003,0.004,0.004,0.004,0.005,0.005,0.005,0.006,
     *0.006,0.007,0.007,0.008,0.008,0.009,0.009,0.010,0.010,0.011,
     *0.012,0.012,0.013,0.014,0.015,0.015,0.016,0.017,0.018,0.019,
```



```
     *0.020,0.020,0.021,0.022,0.023,0.024,0.026,0.027,0.028,0.029,
     *0.030,0.031,0.033,0.034,0.035,0.037,0.038,0.039,0.041,0.042,
     *0.044,0.045,0.047,0.049,0.050,0.052,0.054,0.056,0.057,0.059,
     *0.061,0.063,0.065,0.067,0.069,0.071,0.074,0.076,0.078,0.080,
     *0.083,0.085,0.087,0.090,0.092,0.095,0.097,0.100,0.103,0.105,
     *0.108,0.111,0.114,0.117,0.120,0.123,0.126,0.129,0.132,0.136,
     *0.139,0.142,0.146,0.149,0.153,0.156,0.160,0.164,0.168,0.171,
     *0.175,0.179,0.183,0.187,0.192,0.196,0.200,0.205,0.209,0.213,
     *0.218,0.223,0.227,0.232,0.237,0.242,0.247,0.252,0.258,0.263,
     *0.268,0.274,0.279,0.285,0.291,0.297,0.302,0.308,0.315,0.321,
     *0.327,0.334,0.340,0.347,0.353,0.360,0.367,0.374,0.381,0.389,
     *0.396,0.404,0.411,0.419,0.427,0.435,0.444,0.457,0.465,0.601,
     *0.659,0.549,0.523,0.521,0.526,0.533,0.541,0.549,0.558,0.567,
     *0.577,0.587,0.597,0.607,0.618,0.629,0.640,0.652,0.663,0.675/
C
      END
      subroutine spharm(slat,clat,sjlo,cjlo,r,i0,j0,idif,lfull,
     *lsphap,pii,pim0,pim1,pim2,dlp,dlp0,dlp1,dlp2,dap,dap0,
     *dap1,dap2,ddap,ddap0,ddap1,ddap2,ddal0,ddal1)
c Calculation of the values and the up to 2. order derivatives
c of solid spherical harmonic functions  Y(i0,j0)(lat,lon,r) using
c recursion based on Y(i0-1,j0), Y(i0-2,j0)  when j0 .ne. j0.
c Otherwise the recursion is based on Y(i0-1,j0-1) and
c Y(i0-2,j0-1). The calculation of first order derivatives at the
c poles is done using a recursion formulae, where the cos(lat) then is
c eliminated. The second order derivative with respect to x is at
c the poles calculated using the Laplace equation.
c programmed february 1999  by C.C.Tscherning. Last change 2003-10-08.
c References:
c Tscherning, C.C.: On the Chain-Rule Method for Computing Potential
c Derivatives. Manuscripta Geodaetica, Vol. 1, pp. 125-141, 1976.
c Tscherning, C.C. and K.Poder: Some Geodetic applications of Clenshaw
c Summation. Bolletino di Geodesia e Scienze Affini, Vol. XLI, no. 4,
c pp. 349-375, 1982.
c
c variables at call: slat, clat: sine and cosine of latitude, r the
c size of the radius vector, idif the maximal order of differentiation
c (up to 2),  cjlo, cjlo: cos and sin of j*longitude,
c lfull a logical variable true if fully normalized functions
c are used.
c Last change 2000-04-24 by cct.
c
c     implicit real*8 (a-h,o-z), logical (l)
      implicit none
      REAL*8 D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GM,SUMIJ,CCCIJ,
     *sq2,ys,yc,vv,v1,gs,gc,dds,
     *ddc,root0,pii,pim0,pim1,pim2,dlp,dlp0,dlp1,dlp2,dap,dap0,
     *dap1,dap2,ddap,ddap0,ddap1,ddap2,ddal0,ddal1,r,q,ddal,
     *rq,clat,slat,a,ddal2,b,pm,q2,q3,v,cjlo,sjlo,ddc0,fact,ax
c
      integer i,i0,iimax,j,j0,j1,jold,idif,iiold,k,n,itcoun
c
      parameter (iimax=2000)
      logical lfull,LSPHAR,LTSPH,lf,lt,lsphap,LTEST
c
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GM,ITCOUN,
     *LTEST,LF,LT
      common /con3/SUMIJ(32761),CCCIJ(32761),
     *sq2,ys,yc,vv,v1,gs(3),gc(3),dds(3,3),
     *ddc(3,3),iiold,jold,LSPHAR,LTSPH
      common /rrsph/root0(iimax)
c root is a precomputed square root-table (root0(1)=0 !).
c     common /sphold/pii,pim0,pim1,pim2,dlp,dlp0,dlp1,dlp2,dap,dap0,
c    *dap1,dap2,ddap,ddap0,ddap1,ddap2,ddal0,ddal1,vi
      ax=6378137.0d0
      if (lsphap) then
       q=re/r
      else
c change 2005-05-13. and 14/5.
```

```
       q=ax/r
c      q=re/r
      end if
      j=j0
      i=i0
      j1=j+1
      if (i.eq.j) then
       if (j.ne.(jold+1)) write(*,*)' WARNING j '
       pim2=d0
       pim1=d0
c      write(*,*)r,slat,clat,sjlo,cjlo
       if (idif.gt.0) then
       dlp0=d0
       dlp1=d0
       dlp2=d0
       dap0=d0
       dap1=d0
       if (idif.gt.1) then
        ddal0=d0
        ddal1=d0
        ddap1=d0
        ddap2=d0
       end if
      end if
      if (i.eq.0) then
      pii=q
      pim0=pii
      if (idif.gt.0) then
      dlp=d0
      dlp0=dlp
      dap=d0
      dap0=dap
      if (idif.gt.1) then
      ddap=d0
      ddap0=dap
      ddal=d0
      ddal0=ddal
      end if
      end if
      else
      rq=root0(2*i)/root0(2*i+1)*q
      if (idif.gt.1) then
      ddap=(clat*ddap-d2*slat*dap-clat*pii)*rq
      ddap0=ddap
      ddal=dap*rq
      ddal0=ddal
      end if
      if (idif.gt.0) then
      dap=(-slat*pii+clat*dap)*rq
      dap0=dap
      dlp=pii*rq
      dlp0=dlp
      end if
      pim0=pii*clat*rq
      pii=pim0
      end if
      else
      if (j.ne.jold.or.i.ne.(iiold+1)) write(*,*)' WARNING1 i,j ',
     *i,j,iiold+1,jold
      a=(2*i-1)/(root0(i+j+1)*root0(i-j+1))*q
      if (idif.gt.1) then
      ddap2=ddap1
      ddap1=ddap0
      ddap0=(slat*ddap0+d2*clat*dap0-slat*pim0)*a
      ddal2=ddal1
      ddal1=ddal0
c correction 1999-02-28 by cct - forgottem until 2000-04-25.
c     ddal0=a*(clat*dlp0+slat*ddap0)
      ddal0=a*(clat*dlp0+slat*ddal0)
```



```
      end if
      if (idif.gt.0) then
      dap2=dap1
      dap1=dap0
      dap0=a*(clat*pim0+slat*dap0)
      dlp2=dlp1
      dlp1=dlp0
      dlp0=a*slat*dlp0
      end if
      pim2=pim1
      pim1=pim0
      pim0=a*slat*pim1
      if (i.gt.j) then
      b=-q**2*root0(i-j)*root0(i+j)/(root0(i-j+1)*root0(i+j+1))
      if (idif.gt.1) then
      ddap0=ddap0+b*ddap2
      ddal0=ddal0+b*ddal2
      end if
      if (idif.gt.0) then
      dap0=dap0+b*dap2
      dlp0=dlp0+b*dlp2
      end if
      pm= b*pim2
      pim0=pim0+pm
      end if
      end if
      q2=q*q
      q3=q2*q
      v=pim0
c     yc=v*q*cjlo
c     ys=v*q*sjlo
      yc=v*cjlo
      ys=v*sjlo
      if (idif.gt.0) then
      gs(3)=(-i-1)*q*ys
      gc(3)=(-i-1)*q*yc
      gc(1)=dlp0*q*(-sjlo)*j
      gs(1)=dlp0*q*( cjlo)*j
      gc(2)=dap0*q*cjlo
      gs(2)=dap0*q*sjlo
      if (idif.gt.1) then
      ddc(1,2)=ddal0*q2*(-sjlo)*j
      dds(1,2)=ddal0*q2*( cjlo)*j
      ddc(2,1)=ddc(1,2)
      dds(2,1)=dds(1,2)
      ddc(1,3)=(-i-2)*q2*dlp0*(-sjlo)*j
      dds(1,3)=(-i-2)*q2*dlp0*( cjlo)*j
      ddc(3,1)=ddc(1,3)
      dds(3,1)=dds(1,3)
      ddc(2,2)=(ddap0+(-i-1)*v)*q2*cjlo
      dds(2,2)=(ddap0+(-i-1)*v)*q2*sjlo
      ddc(2,3)=(-i-2)*q2*dap0*cjlo
      dds(2,3)=(-i-2)*q2*dap0*sjlo
      ddc(3,2)=ddc(2,3)
      dds(3,2)=dds(2,3)
      ddc(3,3)=gc(3)*(-i-2)*q
      dds(3,3)=gs(3)*(-i-2)*q
      if (abs(clat).gt.1.0d-10) then
      ddc0=q2*((-i-1)*v-(slat*dap0+v*j**2/clat)/clat)
      ddc(1,1)=ddc0*cjlo
      dds(1,1)=ddc0*sjlo
      else
      ddc(1,1)=-ddc(2,2)-ddc(3,3)
      dds(1,1)=-dds(2,2)-dds(3,3)
      end if
      end if
      end if
      iiold=i
      jold=j
```

```
      if (lfull) then
c normalisation.
      if (j.eq.0) then
      fact= root0(2*i+2)
      else
      fact= root0(2*i+2)*sq2
      end if
      v=v*fact
      yc=yc*fact
      ys=ys*fact
      do 25, k=1,3
C ERROR 2000-05-02 DETECTED.
      gc(k)=gc(k)*fact
      gs(k)=gs(k)*fact
      do 25, n=1,3
      ddc(k,n)=ddc(k,n)*fact
      dds(k,n)=dds(k,n)*fact
  25  continue
c
      end if
      return
      end
      subroutine checkc(npos)
c using the Laplace equation to check to covariances.
c programmed 2002-10-07 by C.C.Tscherning, latest update: 2002-11-26.
c Input:
c  npos - call - used to indicate from where the subroutine is called.
c  COVX - CSAT - holds covariances. Two first subscripts realted
c                to one point (P) and the last two to a second point (Q).
c                index 1: East derivative, 2: North derivative,
c                3: up derivative (radius vector).
c ndp,ndq CSAT - number of derivatives in P, Q, respectively.
c
c Output
c nwar - csat - number of warnings
c if lout is true, output of Laplace equation, sum of absolute value of
c    the 3 terms, the 3 terms.
c
      implicit none
      integer i,j,npos,ndp,ndq,nwar,ksat,ndx1,ndx2,ncase
      real*8 test1,test2,test4,test5,covx,cix,cfx,atest1,atest2
      logical lout,ltests,lx,lnx
      COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFX,KSAT(17,2),
     *NDX1(5),NDX2(5),NDP,NDQ,NWAR,LX(7,5),LNX(7,5),LTESTS
c
      lout=.true.
      j=1
      NCASE=NDP+1+NDQ*3
      go to (810,810,803,810,810,806,807,808,809),ncase
c zero in P, 2 in Q.
 807  test1=covx(1,1,1,1)+covx(1,1,2,2)+covx(1,1,3,3)
      atest1=abs(test1)
      test4=abs(covx(1,1,1,1))+abs(covx(1,1,2,2))+abs(covx(1,1,3,3))
      if (atest1.gt.test4*1.0d-4.and.atest1.gt.1.0d-10) then
       if (lout)
     * write(*,10)npos,i,j,test1,test4,covx(1,1,1,1),covx(1,1,2,2),
     * covx(1,1,3,3)
       nwar=nwar+1
      end if
      go to 810
c
c TWO in P, ONE IN Q.
 806  do I=1,3
       test2=covx(1,1,i,1)+covx(2,2,i,1)+covx(3,3,i,1)
       atest2=abs(test2)
       test5=abs(covx(1,1,i,1))+abs(covx(2,2,i,1))+abs(covx(3,3,i,1))
       if (atest2.gt.test5*1.0d-4.and.atest2.gt.1.0d-20) then
        if (lout)
     *  write(*,10)npos,i,j,test1,test4,covx(i,j,1,1),covx(i,j,2,2),
```



```
     *  covx(i,j,3,3)
        nwar=nwar+1
       end if
      end do
      go to 810
c
 803  test1=covx(1,1,1,1)+covx(2,2,1,1)+covx(3,3,1,1)
      atest1=abs(test1)
      test4=abs(covx(1,1,1,1))+abs(covx(2,2,1,1))+abs(covx(3,3,1,1))
      if (atest1.gt.test4*1.0d-4.and.atest1.gt.1.0d-10) then
       if (lout)
     * write(*,10)npos,i,j,test1,test4,covx(1,1,1,1),covx(2,2,1,1),
     * covx(3,3,1,1)
       nwar=nwar+1
      end if
      go to 810
c
c 1 in P 2 in Q.
 808  do i=1,3
      test1=covx(i,1,1,1)+covx(i,1,2,2)+covx(i,1,3,3)
      atest1=abs(test1)
      test4=abs(covx(i,1,1,1))+abs(covx(i,1,2,2))+abs(covx(i,1,3,3))
       if (atest1.gt.test4*1.0d-4.and.atest1.gt.1.0d-20) then
        if (lout)
     *  write(*,10)npos,i,j,test1,test4,covx(i,1,1,1),covx(i,1,2,2),
     *  covx(i,1,3,3)
        nwar=nwar+1
      end if
      end do
      go to 810
c
c TWO IN BOTH P AND Q.
 809  do i=1,3
      do j=1,3
       test1=covx(i,j,1,1)+covx(i,j,2,2)+covx(i,j,3,3)
       atest1=abs(test1)
       test2=covx(1,1,i,j)+covx(2,2,i,j)+covx(3,3,i,j)
       atest2=abs(test2)
       test4=abs(covx(i,j,1,1))+abs(covx(i,j,2,2))+abs(covx(i,j,3,3))
       test5=abs(covx(1,1,i,j))+abs(covx(2,2,i,j))+abs(covx(3,3,i,j))
       if (atest1.gt.test4*1.0d-4.and.atest1.gt.1.0d-30) then
        if (lout)
     *  write(*,10)npos,i,j,test1,test4,covx(i,j,1,1),covx(i,j,2,2),
     *  covx(i,j,3,3)
   10   format(' WARNING ',i2,2i3,5d12.5)
        nwar=nwar+1
       end if
       if (atest2.gt.test5*1.0d-4.and.atest2.gt.1.0d-30) then
        if (lout)
     *  write(*,11)npos,i,j,test2,test5,covx(1,1,i,j),covx(2,2,i,j),
     *  covx(3,3,i,j)
   11   format(' WARNING5 ',i2,2i3,5d12.5)
        nwar=nwar+1
       end if
      end do
      end do
  810 return
      end
      FUNCTION VAR(SM,IS,KP,DRM,AAI,HP,IMAX1,LMEAN,CP,SP,LSAT,SROT)
C PROGRAMMED FEB 1985 BY C.C.TSCHERNING. UPDATE: OCT.08, 2002.
C THE FUNCTION COMPUTES THE VARIANCE OF A SIGNAL QUANTITY OF TYPE
C KP USING COVBX AND COVCX.
C IF DOUBLE PRECISION IS NEEDED, ACTIVATE THE FOLLOWING STATEMENT:
      implicit none
      LOGICAL LT,LF,LSUM,LOCAL,LMEAN,LSAT,LTESTS,LX,LNX,LTEST
      REAL*8 CI,CR,SIGMA0,SIGMA,HMAX,D,D0,D1,D2,D3,D4,D5,RE,RADSEC,
     *PI,GM,STEPN,COSSTN,SINSTN,STEPE,COSSTE,SINSTE,REX,
     *COST2P,SINT2P,FILTER,STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE,
     *COST2Q,SINT2Q,COVX,CIX,CFA,SM,SROT,AAI,DRM,RP,HP,CVV,STEQQN,
```

```
     *VAR,COMEAN,CP,SP,gcx(3)
      integer ksat,ki,n1,n2,nfilte,ndx1,ndx2,ndp,ndq,nwar,imax1,
     *kp,is,itcoun
C
      COMMON /CMCOV/CI(24),CR(56),SIGMA0(1200),SIGMA(1200),HMAX,
     *D(40),KI(37),N1,N2,LOCAL,LSUM
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,REX,RADSEC,PI,GM,ITCOUN,
     *LTEST,LF,LT
      COMMON /CMEAN/STEPN,COSSTN,SINSTN,STEPE,COSSTE,SINSTE,
     *COST2P,SINT2P,FILTER(11),NFILTE
      COMMON /CMEAQ/STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE,
     *COST2Q,SINT2Q
      COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFA,KSAT(17,2),
     *NDX1(5),NDX2(5),NDP,NDQ,NWAR,LX(7,5),LNX(7,5),LTESTS
      DIMENSION SM(2001),SROT(3,3)
C
      CI(8) = AAI
C CHANGE 2005-05-14.
      RE=6371000.0d0
      CI(9) = (RE+DRM)**2
c     write(*,*)' ci9 ',ci(9),re,drm
      CI(10)= DRM
      CI(20)= D1
      N1 = IMAX1
      KI(6) = KP
      KI(7) = KP
      RP = RE+HP
c     CALL COVBX(SM,LF,IS)
      CALL COVBX(SM,LSAT,IS)
      CR(1) = D1
      CR(2) = HP
      CR(3) = HP
      CR(4) = D0
      CR(5) = D0
      CR(6) = D1
      CR(7) = D1
      CR(8) = D0
      CR(9) = D1
      CR(10)= GM/(RP*RP)
      CR(11) = CR(10)
      IF (.NOT.LMEAN) THEN
      CALL COVCX(SM,CVV,IS,LF,gcx,lt)
c     CALL COVCX(SM,CVV,IS,LSAT,gcx,lt)
      IF (LSAT) THEN
       CALL COVROT(SROT,SROT)
       IF (LTESTS) WRITE(*,101)COVX
 101   FORMAT(6D12.4)
c      CVV=COVX(KSAT(KP,1),KSAT(KP,2),KSAT(KP,1),KSAT(KP,2))
       IF (KP.NE.25) THEN
        CvV=COVX(KSAT(KP,1),KSAT(KP,2),KSAT(KP,1),KSAT(KP,2))
C CHANGE 2002-10-23.
       ELSE
C DDT/DXX-DDT/DYY IN P.
        CVV=
     *  (COVX(KSAT(14,1),KSAT(14,2),KSAT(14,1),KSAT(14,2))
     *  -COVX(KSAT(12,1),KSAT(12,2),KSAT(14,1),KSAT(14,2))
     *  +COVX(KSAT(14,1),KSAT(14,2),KSAT(12,1),KSAT(12,2))
     *  -COVX(KSAT(12,1),KSAT(12,2),KSAT(12,1),KSAT(12,2)))
       END IF
       IF (LTESTS) WRITE(*,100)CVV,KSAT(KP,1),KSAT(KP,2),KP
 100   FORMAT(' CVV, KP ',D14.6,3I3)
      END IF
      ELSE
C CHANGE 2001-07-15.
      STEQQN=STEQN
      STEQN=STEPN
      COSSQN=COSSTN
      SINSQN=SINSTN
      STEQE=STEPE
```



```
      COSSQE=COSSTE
      SINSQE=SINSTE
      COST2Q=COST2P
      SINT2Q=SINT2P
      CVV=COMEAN(SM,IS,0,CP,SP,D1,D0,CP,SP,D1,D0,5,5,LF,LF,LF)
      END IF
C CHANGE 2000-04-11 AND 2002-09-30 BY CCT.
      IF (LSAT) THEN
       IF (KP.EQ.6.OR.KP.EQ.7.OR.KP.EQ.2) THEN
       CVV=CVV*1.0D10
C CONVERSION TO MGAL.
C      CVV=CVV*(CR(10)*1.0D5/RADSEC)**2
       ELSE
        IF (KP.GT.7.OR.KP.EQ.5) THEN
c scaling for 2-order derivatives (to EU**2).
         CVV=CVV*1.0D18
C SCALING FOR 2*TXY. 2002-11-26.
         IF (KP.EQ.13)CVV=CVV*4.0d0
         IF (LTESTS) WRITE(*,*)' KP, CVV ',KP,CVV
        END IF
       END IF
      END IF
      VAR = CVV
C CHANGE 2001-07-15.
      STEQN=STEQQN
      RETURN
      END
      SUBROUTINE INTABH(SM,IS,LTEST)
C PROGRAMMED SEPT 1987 BY C.C.TSCHERNING. LATEST CHANGE 4 DEC 87.
C
C THE SUBROUTINE INITIALIZES TABELS FOR COVARIANCE-FUNCTION
C INTERPOLATION FOR FIXED HEIGHTS AND FUNCTIONALS.
C
C CALL VALUES:
C SM, IS PARAMETERS USED BY COVBX AND COVCX. SM IS A DUMMY
C REAL ARRAY WITH AT LEAST 3 ELEMENTS AND IS IS THE SUBSCRIPT
C OF THE FIRST ELEMENT USED IN SIGMA0 AND SIGMA, WHICH HOLDS
C THE DEGREE-VARIANCE CORRECTIONS. (SEE COVAX).
C
C OTHER CALL VALUES ARE STORED IN COMMON BLOCK CTABH:
C HTA  - HOLD THE UP TO 5 FIXED HEIGHTS IN METERS,
C SIZEI- THE INTERVAL SIZE IN ARCSECONDS (AT CALL),
C NFU  - THE NUNBER OF FIXED FUNCTIONALS FOR EACH FIXED HEIGHT,
C KEYH - THE FUNCTIONAL TYPES (0 - 15), SEE COVAX,
C NINTH- NUMBER OF INTERVALS IN TABLE,
C NHE  - NUMBER OF FIXED HEIGHTS.
C
C RETURN VALUES:
C IN CHTA THE COVARIANCE TABLES,
C TMAX  - THE MAXIMAL VALUE OF T=COS(SPHERICAL DISTANCE)
C NTABH - THE LOGICAL TABLE NUMBER FOR EACH HEIGHT AND
C         AND FUNCTIONAL. THERE WILL NORMALLY BE SEVERAL
C         TABLES FOR EACH FUNCTIONAL, SUCH AS BOTH THE FIRST,
C         SECOND AND THIRD DERIVATIVE WITH RESPECT TO T.
C SIZEI - NOW IN UNITS OF RADIANS.
C
      IMPLICIT NONE
      INTEGER NCTA
      PARAMETER (NCTA=51200)
C SMA PARAMETER (NCTA= 3200)
C 386 PARAMETER (NCTA= 4800)
C IF DOUBLE PRECISION IS NEEDED, ACTIVATE THE FOLLOWING:
      REAL*8 CCI,CR,SIGMA0,SIGMA,HMAX,CCV,D,
     *CHTA,CTTF,CTSF,SZ,AX,RC,HTA,TMAX,SIZEI,A,S,RB2,T,B,
     *D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,SM,TT,COV,HP,HQ,CR50,gcx(3)
C
      INTEGER KI,NC1,N2,MAXB,IX,IIX,IOLD,NFU,KEYH,NINTH,NTABH,
     *NHE,NSTART,KT,KT1,K,I1,JJD,N3,KK,KQ,KP,ND,NR,ND1,ND2,ITCOUN,
     *NTAB,NINTH1,NINTH2,I,JI,J,II,JJ,II1,NTT,IK,NN,IS
```

```
C
      LOGICAL LTABH,LOCAL,LSUM,LOLDFU,LTEST,LTESTS,LT,LF
C
      COMMON /CMCOV/CCI(24),CR(56),SIGMA0(1200),SIGMA(1200),HMAX,
     *CCV(2,2),D(36),KI(37),NC1,N2,LOCAL,LSUM
C CMCOV CONTAINS VARIABLES USED BY COVAX, COVBX AND COVCX.
      COMMON /TABELC/CHTA(NCTA),CTTF(800),CTSF(20),SZ(30),AX(18),
     *MAXB(20),IX(8),IIX(17),IOLD
C TABELC CONTAINS VARIABLES USED BY CTABLE AND COVCG.
      COMMON /CTABH/RC(1200),HTA(5),TMAX,SIZEI,NFU(5),KEYH(5,5),
     *NINTH,NTABH(15,5,5),NHE,NSTART,LTABH
      COMMON /DDY/A,S,RB2,T,B,KT,KT1,K,I1,JJD,N3,KK,KQ,KP,ND,NR,ND1,
     *ND2
C DDY CONTAINS VARIABLES USED BY COVBX AND COVCX. HERE ONLY ND2
C EQUAL TO THE NUMBER OF DERIVATIVES WITH RESPECT TO T IS USED.
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,
     *LTESTS,LF,LT
      DIMENSION SM(2001),TT(500)
C
      LTABH=LF
C CREATION OF TABLES. NTAB COUNTS NUMBER OF TABLES.
      NTAB=0
      SIZEI=SIZEI/RADSEC
      TMAX= COS(NINTH*SIZEI)
      NINTH1=NINTH+1
      NINTH2=NINTH1+1
      DO 210 I=1,NINTH1
  210 TT(I)= COS((I-1)*SIZEI)
C
      DO 300 I=1,NHE
      HP=HTA(I)
      CR(2)=HP
      JI=((I-1)*I)/2
C
      DO 301 J=1, I
      HQ=HTA(J)
      CR(3)=HQ
      DO 301 II=1,NFU(I)
      KI(6)=KEYH(II,I)
C IF THE FUNCTIONAL IS ONE OF A PAIR (LIKE A PAIR OF
C OF DEFLECTIONS) THE TABLE FOR ONLY ONE OF THE TWO IS MADE.
      IF (KI(6).GT.16) KI(6)=KI(6)-10
      IF (KI(6).EQ.13)KEYH(JJ,I)=15
      IF (KI(6).EQ.11)KEYH(JJ,I)=10
      IF (KI(6).EQ.7)KEYH(JJ,I)=6
C
      II1=II
      IF (I.NE.J) II1=NFU(J)
      DO 302 JJ=1,II1
      KI(7)=KEYH(JJ,J)
      IF (KI(7).GT.16) KI(7)=KI(7)-10
      CALL COVBX(SM,.FALSE.,IS)
C
C DEPENDING OF FUNCTIONAL TYPE, THE DERIVATIVES WITH RESPECT TO
C T FROM NSTART-2 TO ND2-2 MYST BE STORED, SEE COVCX.
      NSTART=2
      IF (ND1.EQ.1) GO TO 305
      LOLDFU=KP.EQ.12.OR.KP.EQ.14.OR.KQ.EQ.12.OR.KQ.EQ.14
      NSTART=3
      IF (LOLDFU) GO TO 305
      NSTART=4
      IF (KI(10).GT.1.AND.KI(11).GT.1.OR.ND2.EQ.3) NSTART=3
  305 CONTINUE
      NTABH(JI+J,II,JJ)=NTAB
      NTT=ND2-NSTART+1
C NTT IS TOTAL NUMBER OF DERIVATIVES WRT T, WHICH MUST BE STORED.
C
      DO 303 IK=1, NINTH1
      CR(1)=TT(IK)
```



```
      CALL COVCX(SM,COV,IS,.FALSE.,gcx,lt)
      IF (LTEST)WRITE(6,11)IK,CR(1),COV,(CR(NN+50+NSTART-1),NN=1,NTT)
   11 FORMAT(' IK,T,COV=',I3,F10.7,F9.3,/,5E14.7)
C
C THE CALL OF COVCX GIVES ALL DERIVATIVES, BUT NOT ALL NEED TO
C BE STORED.
      DO 304 NN=1,NTT
      CR50=CR(50+NN+NSTART-1)
      IF (IK.EQ.2) CHTA((NTAB+NN-1)*NINTH2+1)=CR50
  304 CHTA((NTAB+NN-1)*NINTH2+IK+1)= CR50
  303 CONTINUE
      IF (LTEST) WRITE(6,50)NTAB,ND2,NSTART,II,JJ,KI(6),KI(7)
   50 FORMAT(' NTAB,ND2,NSTART,II,JJ,KP,KQ=',7I4)
      NTAB=NTAB+ND2-NSTART+1
  302 CONTINUE
  301 CONTINUE
  300 CONTINUE
      LTABH=.TRUE.
      RETURN
      END
C -----------------------------------------------------------------
      SUBROUTINE TABH(C,LTA)
C PROGRAMMED BY C.C.TSCHERNING, SEPT 1987.
C UPDATE 2002-10-25 (IMPLICIT NONE).
C
C THE SUBROUTINE INTERPOLATES IN A COVARIANCE FUNCTION TABLE
C USING SPLINES, AFTER FIRST HAVING CHECKED WHETHER THE NECESSARY
C TABLES ARE THE ONES CREATED. OTHERWISE COVCX IS CALLED.
C
C CALL VALUES:
C THE TABLE VALUES ARE STORED IN CHTA OF COMMON TABELC.
C THE VARIABLES OF COMMON CTABH AS INITIALIZED BY INTABH.
C
C CALL AND RETURN VARIABLES:
C IF THE SAME FIXED FUNCTIONALS AND HEIGHT AS IN LAST CALL
C ARE USED, THEN TABEL VALUES IN RC ARE USED FOR THE
C THE SPLINE INTERPOLATION. OTHERWISE THEY ARE COMPUTED BY
C ISPCOV.
C THE VARIABLES IN CTABH1 KEEPS TRACK OF THE LAST HEIGHT AND
C FUNCTIONAL TYPES USED.
C
C RETURN VARIABLES:
C C   - ARRAY CONTAINING THE DERIVATIVES WITH RESPECT TO T.
C LTA - LOGICAL, TRUE IF IT WAS POSSIBLE TO USE THE TABLES.
C       OTHERWISE COVCX WILL COMPUTE THE NEEDED QUANTITIES.
C
      implicit none
      INTEGER NCTA,NFU,KEYH,NINTH,NTABH,NHE,NSTART,N,
     *NC1,N2,KI,MAXB,IX,IIX,IOLD,KT,KT1,K,II,JJ,N3,NINTH1,NINTH2,JI,
     *NR,ND1,ND2,KK,KQ,KP,ND,IHP,IHQ,KFP,KFQ,IFP,IFQ,NTAB,NN,ITCOUN
      PARAMETER (NCTA=51200)
C SMA PARAMETER (NCTA= 3200)
C 386 PARAMETER (NCTA= 4800)
      REAL*8 RC,HTA,TMAX,SIZEI,CCI,CR,SIGMAO,SIGMA,HMAX,
     *CCV,D,CHTA,CTTF,CTSF,SZ,AX,A,S,RB2,T,B,HPOLD,HQOLD,
     *D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,C,HP,HQ,T1,PSI,X1,COVS,
     *SPLCOV
C
      LOGICAL LTA,LTABH,LOCAL,LSUM,LSAHP,LSAHQ,LSAFP,LSAFQ,LSWI,
     *LOLDFU,LTEST,LF,LT
C
      COMMON /CTABH/RC(1200),HTA(5),TMAX,SIZEI,NFU(5),KEYH(5,5),
     *NINTH,NTABH(15,5,5),NHE,NSTART,LTABH
C SEE INTABH FOR DESCRIPTION OF VARIABLES.
      COMMON /CMCOV/CCI(24),CR(56),SIGMAO(1200),SIGMA(1200),HMAX,
     *CCV(2,2),D(36),KI(37),NC1,N2,LOCAL,LSUM
C SEE COVAX, COVBX AND COVCX FOR DESCRIPTION OF VARIABLES.
      COMMON /TABELC/CHTA(NCTA),CTTF(800),CTSF(20),SZ(30),AX(18),
     *MAXB(20),IX(8),IIX(17),IOLD
```

```
C CHTA CONTAINS THE TABLES.
      COMMON /DDY/A,S,RB2,T,B,KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,
     *NR,ND1,ND2
C VARIABLES USED IN COVBX AND COVCX.
      COMMON /CTABH1/HPOLD,HQOLD,IHP,IHQ,KFP,KFQ,IFP,IFQ,NTAB,NN
C VARIABLES KEEPS TRACK OF LAST HEIGHTS AND FUNCTIONALS USED.
      COMMON /DCONC/ D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,
     *LTEST,LF,LT
C CONSTANTS INITIALIZED BY BLOCK DATA.
      DIMENSION C(6)
C
      LTEST=LF
      IF (T.LT.TMAX) GO TO 600
      NINTH1=NINTH+1
      NINTH2=NINTH1+1
      HP= CR(2)
      HQ= CR(3)
      KP=KI(6)
      IF (KP.EQ.7.OR.KP.EQ.11)KP=KP-1
      IF (KP.EQ.13)KP=15
      KQ=KI(7)
      IF (KQ.EQ.7.OR.KQ.EQ.11)KQ=KQ-1
      IF (KQ.EQ.13)KQ=15
      LSAHP =  ABS(HP-HPOLD).LT. 0.1
      LSAHQ =  ABS(HQ-HQOLD).LT. 0.1
      LSAFP=KP.EQ.KFP
      LSAFQ=KQ.EQ.KFQ
C
C CHECK, WHETHER WE STILL HAVE THE SAME HEIGHT AND FUNCTIONALS
C AS DURING THE LAST CALL.
      IF (LSAHP.AND.LSAHQ.AND.LSAFP.AND.LSAFQ) GO TO 500
C
C THEN FIND POINTERS TO ACTUAL HEIGHTS AND FUNCTIONALS.
      IF (LSAHP) GO TO 100
      IHP=0
  101 IHP=IHP+1
      IF (IHP.GT.NHE) GO TO 600
      IF ( ABS(HP-HTA(IHP)).GT.0.1) GO TO 101
      HPOLD=HP
C
  100 IF (LSAHQ) GO TO 102
      IHQ=0
  103 IHQ=IHQ+1
      IF (IHQ.GT.NHE) GO TO 600
      IF ( ABS(HQ-HTA(IHQ)).GT. 0.1) GO TO 103
      HQOLD=HQ
C
  102 LSWI=IHQ.GT.IHP
      IF (LSWI) JI=((IHQ-1)*IHQ)/2+IHP
      IF (.NOT.LSWI) JI= ((IHP-1)*IHP)/2+IHQ
C JI POINTS AT TABLE INDEXES.
C
      IFP=0
  104 IFP=IFP+1
      IF (IFP.GT.NFU(IHP)) GO TO 600
      IF (KP.NE.KEYH(IFP,IHP)) GO TO 104
      KFP=KP
      IFQ=0
  105 IFQ=IFQ+1
      IF (IFQ.GT.NFU(IHQ)) GO TO 600
      IF (KQ.NE.KEYH(IFQ,IHQ)) GO TO 105
      KFQ=KQ
C
      IF (LSWI.OR.IHP.EQ.IHQ.AND.IFP.LE.IFQ) NTAB=NTABH(JI,IFQ,IFP)
      IF (.NOT.(LSWI.OR.IHP.EQ.IHQ.AND.IFP.LE.IFQ)) NTAB=
     *                                        NTABH(JI,IFP,IFQ)
C
C FIND WHICH DERIVATIVES WRT T ARE STORED.
      NSTART=2
```



```
      IF (ND1.EQ.1) GO TO 110
      LOLDFU=KP.EQ.12.OR.KP.EQ.14.OR.KQ.EQ.12.OR.KQ.EQ.14
      NSTART=3
      IF (LOLDFU) GO TO 110
      NSTART=4
      IF (KI(10).GT.1.AND.KI(11).GT.1.OR.ND2.EQ.3) NSTART=3
  110 CONTINUE
C
      NN=ND2-NSTART+1
C CREATE TABLES NECESSARY FOR THE SPLINE INTERPOLATION.
      IF (LTEST)
     *WRITE(6,10)IHP,IHQ,IFP,IFQ,KFP,KFQ,NTAB,NSTART,ND2
   10 FORMAT(' IHP,IHQ,IFP,IFQ,KFP,KFQ,NTAB,NSTART,ND2=',9I3)
      DO 130 N=1,NN
  130 CALL ISPCOV(NTAB,N)
      KI(37)=KI(37)+1
      GO TO 502
C
  500 KI(36)=KI(36)+1
C
  502 T1= ABS(D1-T)
      IF (T1.LT.2.0D-10) PSI=0.0D0
      IF (T1.GE.2.0D-10) PSI=PI/2.0D0-ASIN(T)
      IF (PSI.LT.D0) PSI=D0
      X1=PSI/SIZEI+D2
C SPLINE INTERPOLATION OF ALL NEEDED DERIVATIVES WRT T.
      DO 501 N=1,NN
      COVS=SPLCOV(X1,NTAB,N)
  501 C(N+NSTART-1)=COVS
      IF (LTEST) WRITE(6,15)PSI*RADSEC,T,C(NSTART)*CCI(12)
   15 FORMAT(' PSI,T,COV=',F10.1,F13.10,E15.5)
      LTA=.TRUE.
      RETURN
C
  600 LTA=.FALSE.
      KI(35)=KI(35)+1
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE CTABEL(IP,LTEST)
C
COMMENT GI REG.NO. 81024, PROGRAMMED BY C.C.TSCHERNING, SEP. 1981.
C UPDATED JAN. 1983 BY CCT, FORTRAN VERSION JUNE 1985, LATEST UPDATE
C NOV 1991 BY CCT.
C
C FUNCTION:
C THE PROCEDURE WILL GENERATE A TABLE OF COVARIANCE VALUES
C IN A GRID WITH (NT+1)*(NS+1) POINTS. EACH KNOT IS ASSOCIATED WITH
C A VALUE OF T1=1-COS(SPHERICAL DISTANCE) AND S1=1-RB2/(RP*RQ),
C WHERE RB2 IS THE SQUARE OF THE RADIUS OF THE BJERHAMMAR-SPHERE
C AND RP, RQ ARE THE RADIAL DISTANCES OF THE POINTS P, Q FROM THE
C ORIGIN. THE TABLE IS USED TO GENERATE THE COEFFICIENTS OF BICUBIC
C POLYNOMIAL WITH 16 COEFFICIENTS, WHICH ARE STORED IN CTA.
C
C IN ORDER TO MAKE IT POSSIBLE TO HAVE SEVERAL TABELS STORED IN THE SAME
C COMMON AREA (FOR STEPVISE COLLOCATION, FOR EXAMPLE), POINTERS ARE USED
C TO POINT AT THE ZERO'TH ELEMENTS IN THE TABELS: CT - IC, CTSF - IS,
C CTTF - IT, SS - IZ, MAXB - IM, SIGMA AND SIGMA0 - II. THE POINTER
C VARIABLES WITH 1 - 6 ADDED POINTS AT THE 1 - 6 ELEMENTS. THE VALUES OF
C THESE POINTERS ARE KEPT IN COMMON, AND CHANGED IF IOLD IS DIFFERENT
C FROM IP. POINTERS FOR A NEW TABLE ARE RETURNED IN IX(IP*4+K), K=1,2,3
C OR 4.
C
C PARAMETERS:
C CALL VALUES:
C CI, KI, SIGMA0, AS SPECIFIED IN COVBX, SEE GI REG.NO. 76083.
C MAXB (INTEGER DIMENSION, BOUNDS (IM1:AT LEAST 12) WITH MAXB(1)=NS, THE
C NUMBER OF POINTS IN S, MAXB(IM2) THE NUMBER OF EQUIDISTANT INTERVALS
C IN PSI, MAXB(IM6+I) THE NUMBER OF SUBINTERVALS IN THE I'TH INTERVAL.
```

```
C SS   (DIMENSION, BOUNDS (1:MIN 6). THE VALUES OF SS DESCRIBES IN
C _     A CONDENSED FORM THE GRID. WE MUST HAVE:
C _     SS(IZ1) = MINIMUM HEIGHT (M), SS(IZ2) MAXIMAL HEIGHT (M),
C _     SS(IZ3) = 0.0, SS(IZ3+I) THE RIGHT-MOST INTERVAL BOUNDARY OF THE
C _     I'TH INTERVAL. A TYPICAL EXAMPLE IS (WITH IS=IM=0):
C _     SS(1)=0.0, SS(2)=2000.0, SS(4)=60.0, SS(5)= 360.0, SS(6)=1200.0.
C _     MAXB(1)=4, MAXB(2)=3, MAXB(7)=3, MAXB(8)=3, MAXB(9)=10.
C
C RETURN VALUES, (IN COMMON TABELC):
C MAXB(IM3)= NT1= TOTAL NUMBER OF T1 INTERVALS +1.
C CTA  (DIMENSION, BOUND ((NS+1)*NT1),16,2) THE COEFFICIENTS.
C CTSF (DIMENSION, BOUND .GE. NS), S1 VALUES.
C CTTF (DIMENSION, BOUND .GE. MAXB(3)), T1 VALUES.
C IX   (DIMENSION, BOUND .GE.6), NEW POINTERS.
C
      implicit none
      INTEGER NCTA
      PARAMETER (NCTA=1600)
C SMA PARAMETER (NCTA=100)
C 386 PARAMETER (NCTA=150)
      LOGICAL LOCAL,LSUM,LTEST,LTESTS,LF,LT
      REAL*8 CT,SM,RR,U,RS,CX,CTA,CTTF,CTSF,SS,A,CI,CR,SIGMA0,
     *SIGMA,HMAX,CV,D,D0,D1,D2,D3,D4,D5,RE,RG,PI,GM,R5,R9,DB,R,PMIN,
     *TE,P,DP,SN,S0,T0,S1,T1,S,R2,G,G2,COV,RB2,HMIN,DH,H,gcx(3)
      INTEGER KK,MAXB,IX,IC,IT,IT1,IS,IS1,IZ,IZ1,IZ2,IZ3,IM,IM1,IM2,IM3,
     *IM4,IM5,IM6,II,IOLD,KI,N1,N,ITCOUN,IP4,IP,NS,NS1,NT,I,K2,
     *J,K,NT1,KA,K0,K1,I1,J1,I0,J0,M,NN,J3,I5,NS2,N2
C
      DIMENSION CT(3200),SM(2001),RR(200),U(200),RS(8),CX(18)
      COMMON /CCVCG/KK(24)
      COMMON /TABELC/CTA(NCTA,16,2),CTTF(800),CTSF(20),SS(30),A(18),
     *MAXB(20),IX(8),IC,IT,IT1,IS,IS1,IZ,IZ1,IZ2,IZ3,IM,IM1,IM2,IM3,
     *IM4,IM5,IM6,II,IOLD
      COMMON /CMCOV/CI(24),CR(56),SIGMA0(1200),SIGMA(1200),HMAX,
     *CV(2,2),D(36),KI(37),N1,N2,LOCAL,LSUM
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,RE,RG,PI,GM,ITCOUN,
     *LTESTS,LF,LT
C
      IP4=IP*4
      IC=IX(1+IP4)
      IT=IX(2+IP4)
      IT1=IT+1
      IS=IX(3+IP4)
      IS1=IS+1
      IZ=IP*3
      IZ1=IZ+1
      IZ2=IZ+2
      IZ3=IZ+3
      IM=IP*6
      IM1=IM+1
      IM2=IM+2
      IM3=IM+3
      IM4=IM+4
      IM5=IM+5
      IM6=IM+6
      II=IX(4+IP4)
      IOLD=IP
C
      MAXB(IM4)=-1
      MAXB(IM5)=-1
      MAXB(IM6)=-1
      KI(37)=0
      KI(35)=0
      KI(36)=0
C KI(35) - (37) ARE USED TO KEEP TRACK OF HOW MANY TIMES COVCG IS
C CALLED WITH (37) AND WITHOUT (36) HAVING TO CALL BILDEC, AND WHEN IT
C WAS NECESSARY TO CALL COVCX (35).
      R9=1.0D9
      R5=1.0D5
```



```
      NS=IABS(MAXB(IM1))
      NS1=NS+1
      NT= MAXB(IM2)
C
C DB IS THE DIFFERENCE BETWEEN THE EARTH MEAN RADIUS (RE) AND THE RADIUS
C OF THE BJERHAMMAR SPHERE.
      DB=-CI(10)
      RB2=CI(9)
      HMIN= SS(IZ1)
      DH=(SS(IZ2)-HMIN)/NS
      DO 10 I=1,NS1
      H=DH*(I-1)+HMIN
      R=RE+H
      RR(I)=R
   10 CTSF(I+IS)= (D2*RE-DB+H)*(DB+H)/(R*R)
C
      PMIN=D0
      TE=D0
      K2=1
      CTTF(IT1)=D0
      U(1)=D0
      DO 20 J=1,NT
      K=MAXB(J+IM6)
C CONVERSION OF SPHERICAL DISTANCE TO UNITS OF RADIANS.
      P=SS(J+IZ3)/RG
      DP=(P-PMIN)/K
      PMIN=P
C
      DO 21 I=1,K
      K2=K2+1
      TE=TE+DP
      CTTF(K2+IT)=D2* SIN(TE/D2)**2
   21 U(K2)= SIN(TE)
   20 CONTINUE
      NT1=K2
      MAXB(IM3)=NT1
C
      DO 25 KA=1,4
      KI(6)=6
      IF (KA.GT.2) KI(6)=8
      KI(7)= KA
      N= (-1)**(KA+1)
C
      CALL COVBX(SM,.FALSE.,II)
      CR(6)=D1
      CR(11)=D1
      CR(10)=D1
      CR(4)=D0
      CR(8)=D0
      K0=KA
      IF (KA.GT.2)K0=K0+2
      K1= K0+2
      DO 26 I=1,NS1
      CR(2)= RR(I)-RE
      CR(3)= CR(2)
      SN=N
      IF (KA.EQ.2.OR.KA.EQ.4) SN=N/(D1-CTSF(I))
C
      DO 26 J=1,NT1
      CR(9)= D1
      CR(7)= D1-CTTF(J)
      CR(1)= CR(7)
C CR(1) AND CR(7) HOLDS COSINE OF THE SPHERICAL DISTANCE,
C HERE EQUAL TO COSINE OF THE LATITUDE.
      CR(5)= U(J)
C
      CALL COVCX(SM,COV,II,.FALSE.,gcx,lt)
C
      CT(K0+8*(NT1*(I-1)+J-1))= CR(52)*SN
```

```
      CT(K1+8*(NT1*(I-1)+J-1))= -CR(53)*SN
C
   26 CONTINUE
   25 CONTINUE
C
C SET POINTERS FOR NEW COVARIANCE FUNCTION TABLES:
      IP4=IP4+4
      IX(IP4+1)=NT1*NS1+IC
      IX(IP4+2)=NT1+IT
      IX(IP4+3)=NS1+IS
C
      NT=NT1-1
      DO 50 KA=1,2
      K=0
      IF (KA.EQ.2)K=4
C
      DO 51 I=1,NS
      DO 51 J=1,NT
      S0=CTSF(I+IS)
      T0=CTTF(J+IT)
      I1=I+1
      J1=J+1
C
      DO 53 I0=I,I1
      DO 53 J0=J,J1
      M=((J0-J)*2+I0-I)+1
      K0=KK(M+16)
C
      DO 54 N=1,4
      NN=N+K+8*((I0-1)*NT1+J0-1)
   54 CX(K0+KK(N+20))=CT(NN)
   53 CONTINUE
C
      S1=CTSF(I+IS1)-S0
      T1=CTTF(J+IT1)-T0
C
      CALL BILDEC(S1,T1,CX,A)
C
      J3=IC+(I-1)*NT1+J
      DO 55 I5=1,16
   55 CTA(J3,I5,KA)=A(I5)
C
   51 CONTINUE
   50 CONTINUE
C
      IF (.NOT.LTEST) GO TO 99
C
      WRITE(6,100)(CTSF(I+IS),I=1,NS1)
  100 FORMAT('0 1-S=',8F7.4,/,6X,8F7.4)
      WRITE(6,101)(CTTF(I+IT),I=1,K2)
  101 FORMAT('0 1-T=',5(1X,F10.9),10(/,6X,5(1X,F10.9)))
      WRITE(6,102)
  102 FORMAT(' (1,1) (1,2) (1,6) (2,6) (3,3) (3,4)',
     *' (3,8) (4,8) ')
C
      NS2=NS1
      IF (MAXB(IM1).LT.0) NS2=1
      K=1
      DO 30 I=1,NS2
      DO 30 J=1,K2
      S= D1-CTSF(I+IS)
      R= RR(I)
      R2= R*R
      G= R2/GM
      G2= G*G
C
      RS(1)=CT(K)*G2
      RS(2)=CT(K+1)*S*R9*G/R2
      RS(3)=CT(K+2)*RG*G2/R*U(J)
```



```
      RS(4)=CT(K+3)*S*R9*RG*G/(R2*R)*U(J)
      RS(5)=CT(K+4)*R5*R5/R2
      RS(6)=CT(K+5)*S*R5*R9/(R2*R)
      RS(7)=CT(K+6)*R9*R5/(R2*R)*U(J)
      RS(8)=CT(K+7)*S*R9*R9/(R2*R2)*U(J)
      K=K+8
      WRITE(6,105)RS
  105 FORMAT(1X,8F9.3)
   30 CONTINUE
   99 RETURN
      END
C -------------------------------------------------------------
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C                     I F R A C                                      C
C                                                                    C
C  SUBROUTINE GIVING TRUE INTEGER PART OF REAL REAL    C
C                                                                    C
C  RF, JUNE 1983                                                     C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      INTEGER FUNCTION IFRAC(R)
C
      implicit none
      REAL*8 R
      IF (R.LT.0.0D0) GO TO 1
        IFRAC = R
      RETURN
 1      IFRAC = R - 0.999999999D0
      RETURN
      END
C -----------------------------------------------------------
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C                     I S P C O V                                    C
C                                                                    C
C  INITIALIZATION PROCEDURE FOR FAST 1-DIMENSIONAL EQUIDISTANT       C
C  SPLINE INTERPOLATION, WITH FREE BOUNDARY END CONDITIONS           C
C  REFERENCE: JOSEF STOER: EINFUHRUNG IN DIE NUMERISCHE MATHEMATIK C
C  I, SPRINGER 1972. MODIFIED FOR COVARIANCE INTERPOLATION.          C
C                                                                    C
C  PARAMETERS (REAL):                                    C
C                                                                    C
C  Y  GIVEN VALUES, Y(1), ..., Y(N)                                  C
C                                                                    C
C  R  SPLINE MOMENTS (1 ... N), TO BE USED BY FUNCTION 'SPLINE'      C
C                                                                    C
C  Q  WORK-ARRAY, DECLARED AT LEAST 1:N                              C
C                                                                    C
C  RENE FORSBERG, JULY 1983, MODIFIED BY C.C.TSCHERNING OCT 87.      C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE ISPCOV(NTAB,NN)
C
      implicit none
      INTEGER NCTA,MAXB,IX,IIX,IOLD,NFU,KEYH,NINTH,NTABH,NHE,NSTART,
     *N,KY0,NTAB,NN,KSTART,KK,K,KY
      REAL*8 Y,CTTF,CTSF,SZ,AX,R,HTA,TMAX,SIZEI,Q,P
      PARAMETER (NCTA=51200)
      LOGICAL LTABH
C
      COMMON /TABELC/Y(NCTA),CTTF(800),CTSF(20),SZ(30),AX(18),
     *MAXB(20),IX(8),IIX(17),IOLD
      COMMON /CTABH/R(1200),HTA(5),TMAX,SIZEI,NFU(5),KEYH(5,5),
     *NINTH,NTABH(15,5,5),NHE,NSTART,LTABH
      DIMENSION Q(1200)
C
```

```
      N=NINTH+2
      KY0=(NTAB+NN-1)*N
      KSTART=N*(NN-1)
      Q(1) = 0.0D0
      R(1+KSTART) = 0.0D0
      DO 11 KK = 2, N-1
        K=KK+KSTART
        KY=KK+KY0
        P = Q(K-1)/2+2
        Q(K) = -0.5E0/P
        R(K) = (3*(Y(KY+1)-2*Y(KY)+Y(KY-1)) - R(K-1)/2)/P
   11 CONTINUE
      R(N+KSTART) = 0.0D0
      DO 12 KK = N-1, 2, -1
        K=KK+KSTART
        R(K) = Q(K)*R(K+1)+R(K)
   12 CONTINUE
      RETURN
      END
C -------------------------------------------------------
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                  C
C                          S P L C O V                             C
C                                                                  C
C  FAST ONE-DIMENSIONAL EQUIDISTANT SPLINE INTERPOLATION FUNCTION. C
C                                                                  C
C  PARAMETERS:                                                     C
C                                                                  C
C  X   INTERPOLATION ARGUMENT (REAL), X = 1 FIRST DATA-POINT,      C
C      X = N LAST DATA-POINT. OUTSIDE THE RANGE LINEAR EXTRA-      C
C      POLATION IS USED.                                           C
C                                                                  C
C  Y   REAL*8 ARRAY, 1 .. N : DATA VALUES                          C
C                                                                  C
C  R   DO: SPLINE MOMENTS CALCULATED BY SUBROUTINE 'ISPCOV'        C
C                                                                  C
C  PROGRAMMER:                                                     C
C  RENE FORSBERG, JUNE 1983, MODIFIED OCT 87 BY C.C.TSCHERNING     C
C                                                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      FUNCTION SPLCOV(X, NTAB, NN)
C
      implicit none
      REAL*8 X,Y,CTTF,CTSF,SZ,AX,R,HTA,TMAX,SIZEI,XX,SPLCOV
      INTEGER NTAB,NN,NINTH,NTABH,NHE,NSTART,MAXB,IX,IIX,IOLD,
     *NFU,KEYH,NCTA,N,KY0,KY1,KSTART,J,IFRAC
      LOGICAL LTABH
      PARAMETER (NCTA=51200)
C
      COMMON /TABELC/Y(NCTA),CTTF(800),CTSF(20),SZ(30),AX(18),
     *MAXB(20),IX(8),IIX(17),IOLD
      COMMON /CTABH/R(1200),HTA(5),TMAX,SIZEI,NFU(5),KEYH(5,5),
     *NINTH, NTABH(15,5,5),NHE,NSTART,LTABH
C
      N=NINTH+2
      KY0=(NTAB+NN-1)*N
      KY1=KY0+1
      KSTART=N*(NN-1)
      IF(X.GE.1.0D0) GO TO 1
        SPLCOV = Y(KY1) + (X-1)*(Y(2+KY0)-Y(KY1)-R(2+KSTART)/6)
      RETURN
    1 IF(X.LE.FLOAT(N)) GO TO 2
        SPLCOV = Y(N+KY0)+(X-N)*(Y(N+KY0)-Y(KY0+N-1)+R(KSTART+N-1)/6)
      RETURN
    2   J = IFRAC(X)
        XX = X - J
        SPLCOV = Y(J+KY0) +
     .   XX * ((Y(J+KY1)-Y(J+KY0)-R(J+KSTART)/3-R(J+1+KSTART)/6) +
```



```
     .     XX * (R(J+KSTART)/2 +
     .     XX * (R(J+KSTART+1)-R(J+KSTART))/6))
      RETURN
      END
      SUBROUTINE ICMEAN
     *(BSIZE,STEP,NSTEP,COSST,SINST,COSLAT,SINLAT,LEQANG,LMEA1)
C PROGRAMMED BY C.C.TSCHERNING, GEODETIC INSTITUTE, NOV 1985.
C THE SUBROUTINE INITIALIZES STEP VARIABLES FOR MEAN VALUE
C COMPUTATION. CHANGED 1996.10.08 BY CCT.
C LEQANG IS TRUE, WHEN WE DEAL WITH EQUAL-ANGULAR BLOCK AVERAGES.
C LMEA1 IS TRUE WHEN WE HAVE 1-D MEANS.
C IF DOUBLE PRECISION IS NEEDED, ACTIVATE THE FOLLOWING STATEMENT:
      implicit none
      LOGICAL LEQANG,LMEA1,LTEST
      REAL*8 BSIZE,STEP,COSST,SINST,COSLAT,SINLAT,BSIZEA
      INTEGER NSTEP,NSTEP1
C
      LTEST=.FALSE.
      NSTEP1=NSTEP-1
      BSIZEA=ABS(BSIZE)
      IF (LEQANG) GO TO 10
      STEP=2*BSIZE/4.0
      BSIZEA=BSIZEA/(COSLAT* COS(STEP)+SINLAT* SIN(STEP))
C CORRECTION 1995.11.21 BY CCT.
   10 IF (LMEA1) THEN
C FOR 1-D MEANS, THE POINTS ARE SUPPOSED TO BE DISTRIBUTED EQUIDISTANTLY
C ON THE INTERVAL OF SIZE BSIZE. FOR 2-D MEANS THEY ARE DISTRIBUTED
C WITH NSTEP POINTS INSIDE THE INTERVAL.
      STEP=BSIZEA/NSTEP1
      ELSE
      STEP=BSIZEA/NSTEP
      END IF
      COSST= COS(ABS(STEP))
      SINST= SIN(ABS(STEP))
      IF (LTEST) WRITE(*,*)' ICMEAN: STEP= ',STEP
      RETURN
      END
      FUNCTION COMEAN(SM,IS,ISP,COSLAP,SINLAP,COSLOP,SINLOP,
     *COSLAQ,SINLAQ,COSLOQ,SINLOQ,NSTEPP,NSTEPQ,LTABLE,
     *LCZERO,LTCOV)
C PROGRAMMED NOV 1985 BY C.C.TSCHERNING, GEODETIC INSTITUTE.
C THE SUBROUTINE COMPUTES MEAN VALUES OF COVARIANCES.
C CHANGED 2002-09-12.
      implicit none
      LOGICAL LOCAL,LSUM,LTABLE,LMEAP1,LMEAQ1,LTEST,LCZERO,LTCOV,
     *LFOURI,LLCOEE,LPARMP,LPARMQ
C
      REAL*8 STEPN,COSSTN,SINSTN,STEPE,COSSTE,SINSTE,
     *COST2P,SINT2P,FILTER,STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE,
     *COST2Q,SINT2Q,CCI,CCR,SI,HCMAX,CCV,DC,SCFRDD,SCFACT,RDD,
     *FOUCOF,SM,COVME,RADEG,RLAT,RJ,SINLAP,SINLAQ,COVM,COLAP,
     *COSLAP,SILAP,COLOP,COSLOP,SILOP,RLAY,COLOQ,COSLOQ,SILOQ,RLAX,
     *COSDLO,T,RLOX,PSI,COV,COLOQ1,COLAQ1,COLOP1,COLAP1,COMEAN,COVCG,
     *COZERO,RLONG,SINLOP,SINLOQ,COLAQ,SILAQ,RLOY,COSLAQ,gcx(3)
      INTEGER KVI,KP,KQ,NFOURI,I,NSTEPE,NSTEQE,NSTEPP,NSTEPQ,MLAP,
     *MLOP,J,IS,ISP,MLAQ,MLOQ,NFILTE
C
      COMMON /CMEAN/STEPN,COSSTN,SINSTN,STEPE,COSSTE,SINSTE,
     *COST2P,SINT2P,FILTER(11),NFILTE
      COMMON /CMEAQ/STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE,
     *COST2Q,SINT2Q
      COMMON /CMCOV/CCI(24),CCR(56),SI(2400),HCMAX,CCV(4),
     *DC(36),KVI(39),LOCAL,LSUM
C     COMMON /CLPARM/SCFRDD(42),SCFACT,RDD,LLCOEE(42)
C CHANGE 2002-09-12.
      COMMON /CLPARM/SCFRDD(42),SCFACT,RDD,FOUCOF(0:21),NFOURI,LFOURI,
     *LLCOEE(42)
C     COMMON /CLPARM/SCFACT,RDD
      COMMON /CLPAR1/KP,KQ,LPARMP,LPARMQ
```

```
      DIMENSION SM(2001),COVME(4)
C
      LTEST=LTCOV
      RADEG=180.0/3.1415926535D0
C CCI(20)=1 INDICATES THAT NOT-SO PRECISE EQUATIONS WILL BE USED IN
C COVCX. 2002.10-30.
      CCI(20)=1.0D0
      RLAT=0.0D0
      RJ = 0.0D0
C     STEQE=5.0d0
      LMEAP1=STEPE.LT.1.0D-8
      LMEAQ1=STEQE.LT.1.0D-8
      IF (LTEST) WRITE(*,*)' STEPE,STEQE ',STEPE,STEQE
      NSTEPE=NSTEPP
      NSTEQE=NSTEPQ
      IF (LMEAP1) NSTEPE=1
      IF (LMEAQ1) NSTEQE=1
      IF (LTEST) write(*,*)'STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE',
     *STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE
      IF ( ABS(SINLAP-SINLAQ).GT.1.0D-8.OR. ABS(SINLOP-SINLOQ)
     *.GT.1.0D-8.OR.NSTEPP.EQ.1) GO TO 2999
      COSSQN=COSSTN
      COSSQE=COSSTE
      SINSQN=SINSTN
      SINSQE=SINSTE
C
 2999 COVM=0.0D0
      IF (LTEST) write(*,*)' LMP,Q,SPEN,SQEN ',LMEAP1,LMEAQ1,
     *STEPE,STEPN,STEQE,STEQN
      DO 3000 I=1,4
 3000 COVME(I)=0.0D0
C
      COLAP=COSLAP
      SILAP=SINLAP
C
      DO 3043 MLAP=1,NSTEPP
      CCR(4)=SILAP
      CCR(6)=COLAP
      IF (MLAP.EQ.1.OR.(.NOT.LMEAP1)) THEN
      COLOP=COSLOP
      SILOP=SINLOP
      END IF
C     IF (MLAP.EQ.1.AND.LMEAP1) THEN
C     CALL PAZIM(RLAT,RLONG,COLAP,SILAP,COLOP,SILOP,
C    *-COSSTE,-SINSTE,COST2P,SINT2P,LTEST)
C     END IF
      IF (LTEST) RLAY=ATAN2(SILAP,COLAP)*RADEG
C
      DO 3044 MLOP=1,NSTEPE
      COLAQ=COSLAQ
      SILAQ=SINLAQ
      IF (LTEST) THEN
      RLOY=ATAN2(SILOP,COLOP)*RADEG
      write(*,*)' LAP,LOP',RLAY,RLOY
      END IF
C
      DO 3045 MLAQ=1,NSTEPQ
      IF (MLAQ.EQ.1.OR.(.NOT.LMEAQ1)) THEN
      COLOQ=COSLOQ
      SILOQ=SINLOQ
      END IF
C     IF (MLAQ.EQ.1.AND.LMEAQ1) THEN
C     CALL PAZIM(RLAT,RLONG,COLAQ,SILAQ,COLOQ,SILOQ,
C    *-COSSQE,-SINSQE,COST2Q,SINT2Q,LTEST)
C     END IF
      CCR(5)=SILAQ
      CCR(7)=COLAQ
      IF (LTEST) RLAX=ATAN2(SILAQ,COLAQ)*RADEG
C
```



```
      DO 3046 MLOQ=1,NSTEQE
      COSDLO=COLOP*COLOQ+SILOP*SILOQ
      T=SILAQ*SILAP+COLAP*COLAQ*COSDLO
      IF (T.GT.1.0D0) T=1.0D0
      CCR(9)=COSDLO
      CCR(8)=-SILOP*COLOQ+COLOP*SILOQ
      CCR(1)=T
      IF (LTEST) THEN
      RLOX=ATAN2(SILOQ,COLOQ)*RADEG
      write(*,*)' LAQ,LOQ, T' ,RLAX,RLOX,T
      END IF
      IF (LCZERO) THEN
C FINITE COVARIANCE FUNCTIONS INTRODUCED MAY, 1996 BY CCT.
      PSI=ACOS(T)
      COV=SCFACT*COZERO(PSI,RDD,1)
      CCV(1)=COV
      ELSE
      IF (LTABLE) THEN
      COV=COVCG(SM,ISP,.FALSE.)
      ELSE
      CALL COVCX(SM,COV,IS,.FALSE.,gcx,.true.)
      IF (LTEST) WRITE(*,*)' COV= ',COV
      END IF
      END IF
C CORRECTION FOR LATITUDE FACTOR MADE DEC. 1996.
      IF (.NOT.LMEAP1.AND.(.NOT.LMEAQ1)) THEN
      DO 3001 I=1,4
 3001 COVME(I)=COVME(I)+CCV(I)*COLAP*COLAQ
      COVM=COVM+COV*COLAP*COLAQ
      RJ=RJ+COLAP*COLAQ
      ELSE
      IF (LMEAP1.AND.LMEAQ1) THEN
      COVM=COVM+COV*FILTER(MLAQ)*FILTER(MLAP)
      ELSE
      IF (LMEAQ1.AND.(.NOT.LMEAP1)) THEN
      COVM=COVM+COV*FILTER(MLAQ)*COLAP
      RJ=RJ+COLAP
      END IF
      IF (LMEAP1.AND.(.NOT.LMEAQ1)) THEN
      COVM=COVM+COV*FILTER(MLAP)*COLAQ
      RJ=RJ+COLAQ
      END IF
      END IF
      END IF
C
      IF (.NOT.LMEAQ1) THEN
      COLOQ1=COLOQ
      COLOQ=COLOQ*COSSQE-SILOQ*SINSQE
      SILOQ=SILOQ*COSSQE+COLOQ1*SINSQE
      END IF
 3046 CONTINUE
C
      IF (LMEAQ1) THEN
      CALL PAZIM(RLAT,RLONG,COLAQ,SILAQ,COLOQ,SILOQ,
     *COSSQE,SINSQE,COSSQN,SINSQN,.FALSE.)
      ELSE
      COLAQ1=COLAQ
      COLAQ=COLAQ*COSSQN+SILAQ*SINSQN
      SILAQ=SILAQ*COSSQN-COLAQ1*SINSQN
      END IF
 3045 CONTINUE
C
      IF (.NOT.LMEAP1) THEN
      COLOP1=COLOP
      COLOP=COLOP*COSSTE-SILOP*SINSTE
      SILOP=SILOP*COSSTE+COLOP1*SINSTE
      END IF
 3044 CONTINUE
C
```

```
      IF (LMEAP1) THEN
      CALL PAZIM(RLAT,RLONG,COLAP,SILAP,COLOP,SILOP,
     *COSSTE,SINSTE,COSSTN,SINSTN,.FALSE.)
      ELSE
      COLAP1=COLAP
      COLAP=COLAP*COSSTN+SILAP*SINSTN
      SILAP=SILAP*COSSTN-COLAP1*SINSTN
      END IF
 3043 CONTINUE
C
      J=(NSTEPP*NSTEPQ*NSTEPE*NSTEQE)
      IF (LMEAP1.AND.LMEAQ1) RJ = J
      COMEAN=COVM/RJ
      IF (.NOT.LMEAP1.AND.(.NOT.LMEAQ1)) THEN
      DO 3003 I=1,4
 3003 CCV(I)=COVME(I)/RJ
      ELSE
      CCV(1)=COVM/RJ
      END IF
      IF (LTEST) WRITE(*,*)' COMEAN, J, RJ ',COMEAN,J,RJ
C
      RETURN
      END
      SUBROUTINE BILDEC(DS,DT,C,A)
COMMENT GI REG.NO. 81020, PROGRAMMED SEPT. 1981 BY C.C.TSCHERNING.
C
C REFERENCE: SUENKEL, HANS: A COVARIANCE APPROXIMATION PROCE-
C DURE, OSU REP. 286, 1979, P. 32.
C
C FUNCTION:
C THE VALUES OF A FUNCTION OF TWO VARIABLES, F(S, T) AND ITS DERI-
C VATIVES FS=DF/DS, FT=DF/DT, FTS=D(2)F/DSDT GIVEN IN THE FOUR
C CORNERS OF A RECTANGLE WITH SIDE LENGTH DS, DT ARE USED TO COM-
C PUTE THE COEFFICIENTS OF A POLYNOMIAL, WHICH MAY BE USED TO
C REPRESENT THE FUNCTION OVER THE RECTANGLE, SEE REF. SECTION 4.
C
C  DS  (CALL VALUE, REAL)   SIDE LENGTH IN S,
C  DT  (  -    -  ,  -  )    -       -       T,
C  C   (  -    -  , DIMENSION) BOUNDS (1:16) HOLDS THE DERIVATIVES:
C    IF K IS RELATED TO THE 4 CORNERS IN THE SEQUENCE (0, 0),
C    (DS, 0), (0, DT), (DS, DT), THEN C(J)=F, C(J+1)=FS, C(J+4)=
C    FT, C(J+5)=FTS, AND J=CASE K OF (1, 3, 9, 11).
C  A  (RETURN VALUES, DIMENSION) THE COEFFICIENTS OF THE POLYNOMIAL
C     IN A(1) - A(16). A(17) = DS, A(18) = DT
C
C IF DOUBLE PRECISION IS NEEDED, ACTIVATE:
      implicit none
      real*8 B,A,C,DS,DT,E
      INTEGER I,J
      DIMENSION B(16),A(18),C(16)
C
      A(17)= DS
      A(18)= DT
C
      I=0
      DO 100 J=1,4
      IF (J.EQ.3)I=8
      I=I+2
      C(I)= C(I)*DS
      C(I+3)= C(I+3)*DT
  100 C(I+4)= C(I+4)*DS*DT
C
      DO 101 I=1,4
      B(2*I-1)= 2.0*(C(I+8)-C(I))
      B(2*I)= 1.5*B(2*I-1)
      E= C(I+4)
      B(2*I+7)= C(I+12)+E
  101 B(2*I+8)= B(2*I+7)+E
C
```



```
      A(1)=C(1)
      A(2)=C(2)
      A(5)=C(5)
      A(6)=C(6)
      A(9)= B(2)-B(10)
      A(13)= B(9)-B(1)
      A(10)= B(4)-B(12)
      A(14)= B(11)-B(3)
      A(4)= C(4)+C(2)-2.0*(C(3)-C(1))
      A(3)= C(3)-C(2)-C(1)-A(4)
      A(8)= C(8)+C(6)-2.0*(C(7)-C(5))
      A(7)= C(7)-C(6)-C(5)-A(8)
      E= B(6)-B(14)-A(9)
      A(12)= B(8)-B(16)+A(10)-2.0*E
      A(11)= E-A(10)-A(12)
      E= B(5)-B(13)+A(13)
      A(16)= 2.0*E-B(7)+B(15)+A(14)
      A(15)= -E-A(14)-A(16)
      RETURN
      END
      FUNCTION POL(I,A,T)
C IF DOUBLE PRECISION IS NEEDED, ACTIVATE:
      implicit none
      integer I,J,K
      REAL*8 A,T,P,POL
      DIMENSION A(18)
      P=A(I+12)
      K=2
      DO 10 J=1,3
      P=P*T+A(I+4*K)
   10 K=K-1
      POL=P
      RETURN
      END
      FUNCTION DPOL(I,A,T2,T32)
C PROGRAMMED JUNE 1985 BY C.C.TSCHERNING. COMPUTES FIRST ORDER
C DERIVATIVE OF POLYNOMIAL USED IN BSFC.
C IF DOUBLE PRECISION IS NEEDED, ACTIVATE:
      IMPLICIT NONE
      REAL*8 A,T2,T32,DPOL
      INTEGER I
C
      DIMENSION A(18)
      DPOL=A(I)+T2*(A(I+4)+T32*A(I+8))
      RETURN
      END
      FUNCTION BSFC(A,IS,IT,S,T)
C
COMMENT GI REG.NO.81021, PROGRAMMED BY C.C.TSCHERNING, SEP. 1981.
C FORTRAN VERSION JUNE 1985.
C
C REF.: SUENKEL, HANS: A COVARIANCE APPROXIMATION PROCEDURE,
C OSU REP. NO. 286, 1979, P. 42.
C
C FUNCTION:
C CALCULATION OF ZERO TO SECOND ORDER DERIVATIVES OF A
C BICUBIC POLYNOMIAL.
C
C PARAMETERS:
C  IS   (CALL VALUE, INTEGER) ORDER OF DERIVATIVE IN S,
C  IT   (  -     -  ,    -    )   -          -        T,
C  S    (  -     -  , REAL) NORMALIZED COORDINATE OF THE POINT,
C  T    (  -     -  ,   - )     -         -              -  ,
C  A    (  -     -  , DIMENSION) COEFFICIENTS OF THE POLYNOMIAL, E.G.
C                   AS PRODUCED BY BILDEC IN A(1) - A(16).
C                   A(17), A(18) HOLDS GRID SIZE IN S AND T.
C
C IF DOUBLE PRECISION IS NEEDED, ACTIVATE:
      implicit none
```

```
      integer IS,IT,IT1,IS1,I,I1
      REAL*8 S,T,A,T2,T32,DS,DT,B,B1,B2,BSFC,POL,DPOL
C
      DIMENSION A(18)
      T2= 2.0D0*T
      T32= T*1.5D0
      DS= A(17)
      DT= A(18)
C
      IT1=IT+1
      IS1=IS+1
      GO TO (10,11,12),IT1
   10 B= POL(4,A,T)
C
      GO TO (20,21,22),IS1
C
   20 I = 3
      DO 30 I1 = 1,3
      B = POL(I,A,T)+B*S
   30 I = I-1
      GO TO 99
C
   21 B= POL(3,A,T)+B*S*1.5E0
      B= (POL(2,A,T)+B*S*2.0D0)/DS
      GO TO 99
C
   22 B= (POL(3,A,T)+B*S*3.0D0)*2.0D0/(DS*DT)
      GO TO 99
C
   11 B= DPOL(8,A,T2,T32)
C
      GO TO (31,32,33),IS1
C
   31 I = 7
      DO 34 I1 = 1, 3
      B = DPOL(I,A,T2,T32)+B*S
   34 I = I-1
      B= B/DT
      GO TO 99
C
   32 B1= DPOL(7,A,T2,T32)+B*S*1.5E0
      B2= DPOL(6,A,T2,T32)+B1*S*2.0D0
      B= B2/(DT*DS)
      GO TO 99
C
   33 B= DPOL(7,A,T2,T32)+B*S*3.0D0
      B= B*2.0D0/(DS*DS*DT)
      GO TO 99
C
   12 B= A(13)+S*(A(14)+S*(A(15)+S*A(16)))
      B= A(9)+S*(A(10)+S*(A(11)+S*A(12)))+B*T*3.0D0
      B= B*2.0D0/(DS*DT)
C
   99 BSFC= B
      RETURN
      END
      SUBROUTINE COVROT(SROTP,SROTQ)
C THE SUBROUTINE WILL COMPUTE THE ROTATED COVARIANCE MATRIV OR VECTOR
C USING THE ROTATION MATRICES SRORP, SROTQ ASSOCIATED WITH THE POINTS
C P, Q, RESPECTIVELY. SEE REF(I), SECTION 3.
C PROGRAMMED BY C.C.TSCHERNING, GEOPHYSICAL INSTITUTE, UNIVERSITY OF
C COPENHAGEN, JUNE, 1991.
C (I) Tscherning, C.C.: Computation of covariances of derivatives of the
C     anomalous gravity potential in a rotated reference frame.
C     Manuscripta Geodaetica, Vol. 18, no. 3, pp. 115-123, 1993.
C LAST UPDATE 2002-10-24.
C
      implicit none
      REAL*8 COVX,CIX,CFA,SROTP,SROTQ,V,A
```



```
      INTEGER KSAT,NDX1,NDX2,NDP,NDQ,NWAR,NCASE,IM,JM,I,J
      LOGICAL LSATS,LNX,LX
C
      COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFA,KSAT(17,2),
     *NDX1(5),NDX2(5),NDP, NDQ,NWAR,LX(7,5),LNX(7,5),LSATS
      DIMENSION SROTP(3,3),SROTQ(3,3),V(3),A(3,3)
      NCASE=NDP+1+NDQ*3
C
      GO TO (801,802,803,804,805,806,807,808,809),NCASE
C 1 DERIV. IN P, NONE IN Q.
  802 DO 831 IM=1,3
  831 V(IM)=COVX(IM,1,1,1)
      CALL AXV(SROTP,V)
      DO 812 IM=1,3
  812 COVX(IM,1,1,1)=V(IM)
      GO TO 801
C
C 2 DERIV. IN P, NONE IN Q.
  803 DO 823 IM=1,3
      DO 823 JM=1,3
  823 A(IM,JM)=COVX(IM,JM,1,1)
      CALL ATBA(SROTP,A,A)
      DO 824 IM=1,3
      DO 824 JM=1,3
  824 COVX(IM,JM,1,1)=A(IM,JM)
      GO TO 801
C
C NO DERIV. IN P, 1 IN Q.
  804 DO 832 IM=1,3
  832 V(IM)=COVX(1,1,IM,1)
      CALL AXV(SROTQ,V)
      DO 833 IM=1,3
  833 COVX(1,1,IM,1)=V(IM)
      GO TO 801
C
C 1 DERIV. IN BOTH P AND Q.
  805 DO 834 IM=1,3
      DO 835 JM=1,3
  835 V(JM)=COVX(JM,1,IM,1)
      CALL AXV(SROTP,V)
      DO 836 JM=1,3
  836 COVX(JM,1,IM,1)=V(JM)
  834 CONTINUE
      DO 844 IM=1,3
      DO 845 JM=1,3
  845 V(JM)=COVX(IM,1,JM,1)
      CALL AXV(SROTQ,V)
      DO 846 JM=1,3
  846 COVX(IM,1,JM,1)=V(JM)
  844 CONTINUE
      GO TO 801
C
C 2 DERIV. IN P, 1 IN Q.
  806 DO 854 I=1,3
      DO 855 IM=1,3
      DO 855 JM=1,3
  855 A(IM,JM)=COVX(IM,JM,I,1)
      CALL ATBA(SROTP,A,A)
      DO 856 IM=1,3
      DO 856 JM=1,3
  856 COVX(IM,JM,I,1)=A(IM,JM)
  854 CONTINUE
      DO 955 IM=1,3
      DO 955 JM=1,3
      DO 954 I=1,3
  954 V(I)=COVX(IM,JM,I,1)
      CALL AXV(SROTQ,V)
      DO 956 I=1,3
  956 COVX(IM,JM,I,1)=V(I)
```

```
  955 CONTINUE
      GO TO 801
C
C NO DERIV. IN P, 2 IN Q.
  807 DO 923 IM=1,3
      DO 923 JM=1,3
  923 A(IM,JM)=COVX(1,1,IM,JM)
      CALL ATBA(SROTQ,A,A)
      DO 924 IM=1,3
      DO 924 JM=1,3
  924 COVX(1,1,IM,JM)=A(IM,JM)
      GO TO 801
C
C ONE DERIV. IN P, 2 IN Q.
  808 DO 754 I=1,3
      DO 755 IM=1,3
      DO 755 JM=1,3
  755 A(IM,JM)=COVX(I,1,IM,JM)
      CALL ATBA(SROTQ,A,A)
      DO 756 IM=1,3
      DO 756 JM=1,3
  756 COVX(I,1,IM,JM)=A(IM,JM)
  754 CONTINUE
      DO 975 IM=1,3
      DO 975 JM=1,3
      DO 974 I=1,3
  974 V(I)=COVX(I,1,IM,JM)
      CALL AXV(SROTP,V)
      DO 976 I=1,3
  976 COVX(I,1,IM,JM)=V(I)
  975 CONTINUE
      GO TO 801
C
C 2 DERIV. IN P AND Q.
  809 DO 540 I=1,3
      DO 540 J=1,3
      DO 555 IM=1,3
      DO 555 JM=1,3
  555 A(IM,JM)=COVX(IM,JM,I,J)
      CALL ATBA(SROTP,A,A)
      DO 556 IM=1,3
      DO 556 JM=1,3
  556 COVX(IM,JM,I,J)=A(IM,JM)
  540 CONTINUE
      DO 541 I=1,3
      DO 541 J=1,3
      DO 565 IM=1,3
      DO 565 JM=1,3
  565 A(IM,JM)=COVX(I,J,IM,JM)
      CALL ATBA(SROTQ,A,A)
      DO 456 IM=1,3
      DO 456 JM=1,3
  456 COVX(I,J,IM,JM)=A(IM,JM)
  541 CONTINUE
C
  801 RETURN
      END
      SUBROUTINE PAZIM(RLATP,RLONGP,COSLAP,SINLAP,COSLOP,SINLOP,
     *CAZP,SAZP,COSDT,SINDT,LTEST)
c THE SUBROUTINE WILL FROM A POINT WITH LATITUDE AND LONGITUDE
C SPECIFIED IN THE CALL PRODUCE THE CORRESPONDING VALUES IN A
C NEW POINT IN DISTANCE DT  AND AZIMUTH GIVEN BY
C COS AND SIN - CAZP, SAZP.
C PROGRAMMED BY C.C.TSCHERNING, OCT. 92. LAST CHANGE: 2002-10-24.
      implicit none
      real*8 rlatp,rlongp,coslap,sinlap,coslop,sinlop,cazp,sazp,
     *cosdt,sindt,sidlon,codlon,dlong,raddeg,dlatp,dlongp
      LOGICAL LTEST
      RLONGP=ATAN2(SINLOP,COSLOP)
```



```
      SINLAP=COSLAP*SINDT*CAZP+SINLAP*COSDT
      COSLAP=SQRT(1.0D0-SINLAP**2)
      SIDLON=SINDT*SAZP/COSLAP
      CODLON=SQRT(1.0D0-SIDLON**2)
      DLONG=ATAN2(SIDLON,CODLON)
      RLONGP=RLONGP+DLONG
      RLATP=ATAN2(SINLAP,COSLAP)
      COSLOP=COS(RLONGP)
      SINLOP=SIN(RLONGP)
      RADDEG=180.0D0/3.1415926535D0
      DLATP=RADDEG*RLATP
      DLONGP=RADDEG*RLONGP
      IF (LTEST) WRITE(*,*)' PAZIM - LAT,LONG=',DLATP,DLONGP
C
      RETURN
      END
      DOUBLE PRECISION FUNCTION COZERO(PSI,R,MODEL)
C THE SUBROUTINE WILL EVALUATE A COVARIANCE FUNCTION WHICH
C IS ZERO AFTER THE DISTANCE (PSI) IS LARGER THAN OR EQUAL TO 2*R.
C PROGRAMMED APRIL 1996 BY C.C.TSCHERNING, GEOPHYSICAL
C DEPARTMENT. LAST CHANGE 2002-10-24.
C     PSI -  SPHERICAL DISTANCE IN RADIANS
C     R   -  2*R DISTANCE IN RADIANS FOR WHICH FUNCTION IS ZERO.
C     MODEL - PARAMETER TO DISTINGUISH BETWEEN DIFFERENT MODELS.
C
      implicit none
      real*8 d0,d2,d3,pi,psi2,psi,psi3,r2,r,r4
      integer model
c
      PI = 3.1415926535D0
      D0=0.0D0
      D2=2.0D0
      D3=3.0D0
      PSI2=PSI**2
      PSI3=PSI**3
      GO TO (10,20,30) MODEL
  10  R2=R*R
      R4=R2*R2
      IF (PSI.LT.D2*R) THEN
      COZERO= R4*PI*(R2/D3-PSI2/D2)
     *+(R2*(R2*PSI+4.0D0/D3*PSI3)-PSI3*PSI2/12.0D0)/D3
     **SQRT(R2-PSI2/4.0D0)
     *+(R2*R2*(PSI2-D2/D3*R2))*ASIN(PSI/(D2*R))
      ELSE
      COZERO=D0
      END IF
      GO TO 30
  20  COZERO=D0
  30  CONTINUE
      RETURN
      END
      FUNCTION COVCG(SM,IP,LTEST)
C
COMMENT GI REG.NO.81025, PROGRAMMED BY C.C.TSCHERNING, SEPT. 81.
C FORTRAN VERSION JUNE 1985, LATES UPDATE NOV 1991 BY CCT, WHERE
C LINEAR INTERPOLATION IS USED INSTEAD OF BICUBIC FOR THE
C CALCULATION OF CROS-COVARIANCE GRAVITY - DEFLECTIONS.
C
C FUNCTION:
C THE PROCEDURE WILL FIND IN CTA (1) THE COEFFICIENTS OF A BICUBIC
C POLYNOMIUM REPRESENTING A COVARIANCE FUNCTION IN A RECTANGLE
C WITH CORNER COORDINATES (CTSF(I), CTTF(J)) (LOWER, LEFTMOST)
C (CTSF(I+1), CTTF(J+1)) (UPPER, RIGHTMOST), IN A COORDINATE
C SYSTEM WITH S1 AS ABSCISSA AND T1 AS ORDINATE, (2) COMPUTE
C THE COVARIANCE FOR A POINT WITHIN THE RECTANGLE.
C
C HERE S1=1-S, S=RB**2/(RP*RQ), T1=1-T=1-COS(SPHERICAL DIST.),
C WHERE RB IS THE RADIUS OF THE BJERHAMMAR-SPHERE, RP, RQ ARE
C THE RADIAL DISTANCES OF TWO POINTS P AND Q, SEE THE PROCEDURE
```

```
C COVCX.
C
C THE COEFFICIENTS ARE STORED IN THE ARRAY A. IN MAXB ARE
C CURRENT VALUES OF I AND J STORED, TOGETHER WITH AN INTEGER K
C INDICATING WHICH TYPE OF COVARIANCE FUNCTION IS STORED (COVA-
C RIANCE(T, T) FOR K=0 AND COV(DELTA G, DELTA G) FOR K=3).
C THIS MAKES THE RECOLLECTION OF THE ELEMENTS OF A UNNECESSARY
C IF THE PROCEDURE IS CALLED IN ORDER TO COMPUTE VALUES WITHIN
C THE SAME RECTANGLE ANOTHER TIME.
C
C IF THE COVARIANCES NEEDED FOR THE INTERPOLATION HAVE NOT BEEN
C TABULATED (IN THE ARRAY CT), THEN COVCX WILL BE CALLED.
C
C PARAMETERS:
C SEE COVCX, COVBX, CTABLE FOR KI, CI, SIGMA, CR, CT AND D.
C A  (DIMENSION, BOUNDS (18)), HOLDS THE COEFFICIENTS (1-16),
C _   AND SIDE LENGTHS IN A(17), A(18).
C
C EXTNAL PROCEDURES USED: COVCX, BSFC.
C
      IMPLICIT NONE
      INTEGER NCTA
      PARAMETER (NCTA=1600)
C SMA PARAMETER (NCTA=100)
C 386 PARAMETER (NCTA=150)
      LOGICAL LFAST,LTESTS,LT,LSUM,LOCAL,LF,LTEST
      INTEGER KI,N1,N2,ITCOUN,KK,MAXB,IX,IC,IT,IT1,IS,IS1,IZ,IZ1,IZ2,
     *IZ3,IM,IM1,IM2,IM3,IM4,IM5,IM6,II,IOLD,IP,IP4,NS,NT,NT1,NS1,
     *J,JJ,JD,NDT,NDS,K,KA,I2,J0,I,KP,KQ,NT0,I0
C
      REAL*8 CI,CR,SIGMA0,SIGMA,HMAX,CVV,D,D0,D1,D2,D3,D4,D5,RE,RG,
     *PI,GM,CTA,CTTF,CTSF,SZ,A,SM,R5,DB,SP,SQ,CP,CQ,CD,T,T1,HP,HQ,
     *RP,RQ,S,S1,S0,T0,DS,DT,C,DCT,DCS,DCSTY,DD00,DD01,DD10,DD11,DCST,
     *SD,SS,DCTX,CS,SC,SCC,CSC,CF,COV,COVCG,BSFC,gcx(3)
C
      COMMON /CMCOV/CI(24),CR(56),SIGMA0(1200),SIGMA(1200),HMAX,
     *CVV(2,2),D(36),KI(37),N1,N2,LOCAL,LSUM
      COMMON /DCONC/D0,D1,D2,D3,D4,D5,RE,RG,PI,GM,ITCOUN,
     *LTESTS,LF,LT
      COMMON /CCVCG/KK(24)
C ELEMENTS OF KK ARE INITIALIZED BY BLOCK DATA ROUTINE.
      COMMON /TABELC/CTA(NCTA,16,2),CTTF(800),CTSF(20),SZ(30),A(18),
     *MAXB(20),IX(8),IC,IT,IT1,IS,IS1,IZ,IZ1,IZ2,IZ3,IM,IM1,IM2,IM3,
     *IM4,IM5,IM6,II,IOLD
      DIMENSION SM(2001)
C
      IF (IOLD.EQ.IP) GO TO 10
      IP4=IP*4
      IC=IX(1+IP4)
      IT=IX(2+IP4)
      IT1=IT+1
      IS=IX(3+IP4)
      IS1=IS+1
      IZ=IP*3
      IZ1=IZ+1
      IZ2=IZ+2
      IZ3=IZ+3
      IM=IP*6
      IM1=IM+1
      IM2=IM+2
      IM3=IM+3
      IM4=IM+4
      IM5=IM+5
      IM6=IM+6
      II=IX(4+IP4)
      IOLD=IP
      MAXB(IM6)=-1
C THIS ASSIGNMENT OF -1 IS DONE TO ASSURE THAT BILDEC IS CALLED WHEN A
C A NEW COVARIANCE FUNCTION TABEL IS USED.
```



```
C
   10 R5=1.0D5
      DB=-CI(10)
      NS=IABS(MAXB(IM1))
      NT=MAXB(IM2)
      NT1=MAXB(IM3)
      NS1=NS+1
C
      SP= CR(4)
      SQ= CR(5)
      CP= CR(6)
      CQ= CR(7)
C
      SS= SP*SQ
      CD=CR(9)
      T= CR(1)
      T1=D1-T
      HP= CR(2)
      HQ= CR(3)
C
      RP= RE+HP
      RQ= RE+HQ
      S= CI(9)/(RP*RQ)
      S1= D1-S
C
      IF (KI(6).EQ.0.OR.KI(7).EQ.0) GO TO 90
      KP=KK(KI(6))
      KQ=KK(KI(7))
      LFAST=(T1.LE.CTTF(NT1+IT)).AND.(S1.GE.CTSF(IS1)).AND.(S1.LE.
     *CTSF(NS1+IS)).AND.(KP.LT.5).AND.(KQ.LT.5).AND.KI(8).LT.2
      IF (.NOT.LFAST) GO TO 90
C
      IF ( ABS(HP-HQ) .GE. 1.0D-5) GO TO 11
      IF ( ABS(HP) .LT. 1.0D-5) S1=DB*(D2-DB/RE)/RE
      IF ( ABS(HP).GT.1.0D-5) S1= (HP+DB)*(2*RE+HP-DB)/RP**2
      GO TO 12
   11 S1= (RE*(2*DB+HP+HQ)+HP*HQ-DB*DB)/(RP*RQ)
   12 S= D1-S1
C
COMMENT FIND INDEX FOR LOWER LEFT CORNER
      I=0
   14 I=I+1
      IF (CTSF(I+IS1) .LT. S1) GO TO 14
      J=0
      IF (.NOT.LTEST) GO TO 19
   13 J=J+1
      IF (CTTF(J+IT1) .LT. T1) GO TO 13
   19 NT0=NT1-1
      JJ=J
      J=1
      IF (T1.LE.D0) GO TO 22
      IF (T1.LT.CTTF(NT0+IT)) GO TO 23
      J=NT0
      GO TO 22
   23 J=NT0/2+1
      JD=NT0
   24 JD=JD/2
      IF (JD.LT.1) JD=1
      IF (T1.GE.CTTF(J+IT)) GO TO 25
      J=J-JD
      GO TO 24
   25 IF (T1.LT.CTTF(J+IT1)) GO TO 22
      J=J+JD
      GO TO 24
   22 IF (J.NE.JJ.AND.LTEST) WRITE(6,98)J,JJ
   98 FORMAT(' J,JJ=',2I4)
      S0= CTSF(I+IS)
      T0= CTTF(J+IT)
C
```

```
      DS= S1-S0
      DT=T1-T0
C
      IF (LTEST) WRITE(6,100)I,J,T1,S1,DT,DS
  100 FORMAT(' I,J,T1,S1,DT,DS=',2I4,4F12.9)
      NDT=0
      IF (KP.GT.2) NDT=1
      IF (KQ.GT.2) NDT=NDT+1
      NDS=0
      IF (KP.EQ.2)NDS=1
      IF (KQ.EQ.2)NDS=NDS+1
      IF (NDS.NE.2) GO TO 15
      NDS= 0
      K= 3
      GO TO 16
   15 K=-1
C
   16 IF (I.EQ.MAXB(IM4).AND.J.EQ.MAXB(IM5).AND.K.EQ.MAXB(IM6)) GO TO 18
      MAXB(IM4)=I
      MAXB(IM5)=J
      MAXB(IM6)=K
      KA=1
      IF (K.EQ.3)KA=2
      I2=IC+(I-1)*NT1+J
      DO 20 J0=1,16
   20 A(J0)=CTA(I2,J0,KA)
C
      S1=CTSF(I+IS1)-S0
      T1=CTTF(J+IT1)-T0
      A(17)=S1
      A(18)=T1
      IF (LTEST) WRITE(6,102)(A(I0),I0=1,16)
  102 FORMAT(' A=',4E15.6,3(/,3X,4E15.5))
      KI(37)= KI(37)+1
      GO TO 21
C
   18 KI(36)= KI(36)+1
C
   21 S1=A(17)
      T1=A(18)
      DS=DS/S1
      DT= DT/T1
C
      C=D0
      DCT=D0
      IF (NDS.EQ.0) GO TO 29
      IF (NDT.GT.0) GO TO 28
C
C COVARIANCE GRAVITY AND HEIGHT ANOMALY.
      C=BSFC(A,0,0,DS,DT)
      DCS= S*BSFC(A, 1, 0, DS, DT)
      C= -(DCS+D2*C)
      CR(52)=C
      GO TO 31
C
C COVARIANCE GRAVITY ANOMALY WITH DEFLECTIONS.
   28 DCTX= BSFC(A, 0, 1, DS, DT)
      IF (LTEST) DCSTY= BSFC(A, 1, 1, DS, DT)
C CHANGE HERE MADE MAY 1988, USING LINEAR INTERPOLATION INSTEAD
C OF BICUBIC INTERPOLATION. THIS MAY NOT BE NECESSARY ON COMPUTERS
C USING DOUBLE PRECISION ARTIHMETIC. DDXY HOLDS THE CORNER VALUES
C OF THE MIXED FIRST ORDER DERIVATIVES IN THE SQUARE WITH SIDE-
C LENGTHS EQUAL TO 1.0 .
      DD00= A(6)/(A(17)*A(18))
      DD01= BSFC(A,1,1,D0,D1)
      DD10= BSFC(A,1,1,D1,D0)
      DD11= BSFC(A,1,1,D1,D1)
      DCST= (DD00*(D1-DS)+DD10*DS)*(D1-DT)+(DD01*(D1-DS)+DD11*DS)*DT
      IF (LTEST) WRITE(6,9765)DD00,DD01,DD10,DD11,DS,DT,DCST,DCSTY
```



```
 9765 FORMAT(' DD00011011 ',4E15.7,/,' DSDTSTSTY ',4E15.7)
      DCT= DCST*S+2*DCTX
      CR(53)=DCT
      GO TO 31
C
   29 IF (NDT.GT.0) GO TO 30
C AUTOCOVARIANCE OF GRAVITY OR OF HEIGHT ANOMALY.
      C=BSFC(A,0,0,DS,DT)
      CR(52)=C
      GO TO 31
C
C CROSS-COVARIANCE HEIGHT ANOMALY WITH DEFLECTIONS.
   30 DCT=-BSFC(A,0,1,DS,DT)
      CR(53)=DCT
   31 CONTINUE
C
      IF (NDT.EQ.0) GO TO 40
      SD= CR(8)
      CS= CP*SQ
      SC= SP*CQ
      SCC= SC*CD
      CSC= CS*CD
      D(3)=CQ*SD
      D(13)= -CP*SD
      D(2)= CS-SCC
      D(7)= SC-CSC
C
   40 IF (LTEST) WRITE(6,110)C,RP,RQ,CI(11),CR(10),CR(11)
  110 FORMAT(' C,RP,RQ,CI11,GP,GQ=',/,
     *E13.6,2F9.1,E13.6,2F9.6)
C
      CF=CI(11)/(RP**KI(22)*RQ**KI(23)*CR(10)**KI(20)
     **CR(11)**KI(21))
      CI(12)=CF
      C=C*CF
      DCT=DCT*CF
      IF (NDT.GT.0) GO TO 71
      CVV(1,1)=C
      GO TO 72
   71 IF (KP.GT.2) GO TO 73
C COVARIANCE WITH DEFLECTION IN Q AND HEIGHT OR GRAVITY ANOMALY IN P.
      CVV(1,1)= DCT*D(13)
      CVV(1,2)= DCT*D(7)
      GO TO 72
C COVARIANCE WITH DEFLECTION IN P AND HEIGHT OR GRAVITY ANOMALY IN Q.
   73 CVV(1,1)=DCT*D(3)
      CVV(2,1)=DCT*D(2)
   72 COVCG=CVV(KI(24),KI(25))
C
      GO TO 99
   90 CALL COVCX(SM,COV,II,.FALSE.,gcx,.true.)
      COVCG=COV
      KI(35)= KI(35)+1
C
   99 RETURN
      END
C -------------------------------------------------------------------
      SUBROUTINE AXV(A,V)
C THE SUBRIUTINE WILL COMPUTE THE PRODUCT OF THE MATRIX A AND THE
C VECTOR V AND RETURN IT IN V. PROGRAMMED 1990.11.03 BY CCT.
      implicit none
      real*8 a,v,y
      integer i,j
      DIMENSION A(3,3),V(3),Y(3)
      DO 10 I=1,3
      Y(I)=V(I)
   10 V(I)=0.0D0
      DO 20 I=1,3
      DO 20 J=1,3
```

```
   20 V(I)=A(I,J)*Y(J)+V(I)
      RETURN
      END
      SUBROUTINE ATBA(A,B,C)
C PROGRAMMED AUG 89 BY C.C.TSCHERNING.
C THE SUBROUTINE WILL COMPUTE THE PRODUCT OF THE 3*3 MATRICES A TRANS-
C POSED, B AND A AND STORE THE RESULT IN C.
      implicit none
      real*8 A,B,C,D,E
      integer j,k,n
      DIMENSION A(3,3),B(3,3),C(3,3),D(3,3),E(3,3)
C A TRANSPOSED TIMES B STORED IN D: :
      DO 30 K=1,3
      DO 30 J=1,3
      D(K,J)=0.0D0
      DO 30 N=1,3
c  30 D(K,J)= A(K,N)*B(N,J)+D(K,J)
   30 D(K,J)= A(N,K)*B(N,J)+D(K,J)
C
C D TIMES A STORED IN E:
      DO 40 K=1,3
      DO 40 J=1,3
      E(K,J)=0.0D0
      DO 40 N=1,3
c  40 E(K,J)=E(K,J)+D(K,N)*A(J,N)
   40 E(K,J)=E(K,J)+D(K,N)*A(N,J)
C
      DO 50 K=1,3
      DO 50 J=1,3
   50 C(K,J)=E(K,J)
      RETURN
      END
      SUBROUTINE CINIT
      implicit none
      logical lc1,lc2,lcref
      COMMON /CHEAD1/LC1,LC2,LCREF
      LCREF=.false.
      RETURN
      END
```