```
PROGRAM GEOCOL19
! $Id: geocol19.f90,v 1.362 2014/04/09 09:41:50 trier Exp $
! ifort geocol19.f90 *.o -assume byterecl -heap-arrays -openmp -traceback -o geo
col19
! PROGRAMMED BY C.C.TSCHERNING, NIELS BOHR INSTITUTE, UNIVERSITY
! OF COPENHAGEN, DENMARK.
! LAST UPDATE: 2014-02-14 BY CCT.  UNIX VERSION.
! THE PROGRAM IS COPYRIGHT BY THE AUTHOR  FIRST TIME 1975,
! AND LATEST 2014. IT MAY BE COPIED AND TRANSFERRED TO
! NON-COMMERCIAL USERS ON THE CONDITION THAT THE AUTHOR IS NOTIFIED.
! THIS WILL ASSURE THAT USERS ARE INFORMED IF FATAL ERRORS ARE FOUND.
! COMMERCIAL USERS MUST OBTAIN A LICENCE FROM THE COPYRIGHT OWNER
! BEFORE USING THE PROGRAM OR PARTS OF THE PROGRAM.
! A FILE HTTP://CCT/GFY.KU.DK/GEOCOL19.LOG CONTAINS A RECORD OF SUSPECTED
! AND CORRECTED ERRORS, UPDATES AND PLANNED UPDATES.
!
! THE PRIMARY FUNCTION OF THE PROGRAM IS TO COMPUTE AN APPROXIMATION TO
! THE ANOMALOUS POTENTIAL OF THE EARTH USING STEPWISE LEAST SQUARES
! COLLOCATION, CF. REF (B) AND THE DETERMINATION OF RELATED PARAMETERS
! SUCH AS DATUM-SHIFTS, SEE REF. (E).
! WHEN THE APPROXIMATION HAS BEEN DETERMINED, IT MAY BE USED TO PREDICT
! VARIOUS QUANTITIES AND ESTIMATE ERROR AND ERROR CORRELATIONS, SEE TABLE 1.
! SECONDARY FUNCTIONS ARE (A) THE REMOVAL OR SUBTRACTION OF THE
! CONTRIBUTION FROM A SPHERICAL HARMONIC EXPANSION (SHE) AND (B)
! THE COMPUTATION OF EFFECTS OF A DATUM SHIFT.
! THE COLLOCATION METHOD REQUIRES THE SPECIFICATION OF
! (1) ONE OR TWO (AND IN A SPECIAL CASE) THREE SETS OF OBSERVED
! QUANTITIES WITH KNOWN STANDARD DEVIATIONS AND
! (2) ONE OR TWO COVARIANCE FUNCTIONS, CF. REF (A).
!
! THE COVARIANCE FUNCTIONS USED ARE ISOTROPIC. THEY ARE SPECIFIED
! BY A SET OF EMPIRICAL ANOMALY DEGREE-VARIANCES OF DEGREE LESS
! THAN AN INTEGER VARIABLE IMAX, AND BY AN ANOMALY DEGREE-VARIANCE
! MODEL FOR THE DEGREE-VARIANCES OF DEGREE GREATHER THAN IMAX.
!
! THE OBSERVATIONS MAY BE POTENTIAL COEFFICIENTS, MEAN OR POINT
! GRAVITY ANOMALIES, HEIGHT ANOMALIES, VALUES OF THE ANOMALOUS
! POTENTIAL, DEFLECTIONS OF THE VERTICAL, GRAVITY GRADIENTS AND
! DENSITY CONTRASTS. THE ABSOLUTE QUANTITIES, I.E. THE SAME ASSOCIATED
! FUNCTIONALS APPIED ON THE FULL POTENTIAL (V OR W) MAY ALSO
! BE USED IN CERTAIN CIRCUMSTANCES.  A FILTERING TAKES
! PLACE SIMULTANEOUSLY WITH THE DETERMINATION OF THE ANOMALOUS
! POTENTIAL.
!
! THE OBSERVATIONS MAY BE GIVEN IN A LOCAL (E.G. INSTRUMENTAL) FRAME
! IN WHICH CASE ATTITUDE INFORMATION MUST BE INPUT. THEY MAY ALSO
! HAVE CORRELATED ERRORS, IN WHICH CASE THE ERROR-COVARIANCE FUNCTION
! MUST BE DEFINED.
!
! THE DETERMINATION IS MADE IN A NUMBER OF STEPS EQUAL TO THE NUMBER OF
! SETS OF OBSERVATIONS. WHEN POTENTIAL COEFFICIENTS ARE USED, WILL THE-
! ESE FORM A SEPARATE SET. CONTRIBUTIONS FROM A TERRAIN POTENTIAL MAY
! NOT BE COMPUTED BY THIS VERSION, BUT MAY BE INPUT AND WILL THEN BE
! ADDED TO OR SUBTRACTED FROM THE VARIOUS QUANTITIES.
! EACH DATASET (EACH STEP) WILL DETERMINE A HARMONIC FUNCTION, AND THE
! ANOMALOUS POTENTIAL WILL BE EQUAL TO THE SUM OF THEESE (MAXIMALLY
! FOUR) FUNCTIONS.
!
! POTENTIAL COEFFICIENTS WILL DETERMINE A FUNCTION EQUAL TO THE COEFFI-
! CIENTS MULTIPLIED BY THE CORRESPONDING SOLID SPHERICAL HARMONICS. THE
! UP TO TWO SETS OF DATA DIFFERENT FROM POTENTIAL COEFFICIENTS WILL
! EACH BE USED TO DETERMINE CONSTANTS B(I). THE CORRESPONDING HARMONIC
! FUNCTIONS ARE THEN EQUAL TO THEESE CONSTANTS MULTIPLIED BY THE COVA-
! RIANCE BETWEEN THE OBSERVATIONS AND THE VALUE OF THE ANOMALOUS POT-
! ENTIAL IN A POINT, P.
!
! THE PROGRAM  WILL COMPUTE, THE CONSTANTS B(I) AND PREDICT QUANTITIES
! ZETA, KSI, ETA, DELTA G, GRAVITY GRADIENTS, DENSITY CONTRASTS IN POINTS Q,
! AND THE ERRORS OF PREDICTION.
```



```
!
! CORRECTIONS TO A SET OF SPHERICAL HARMONICS MAY ALSO BE COMPUTED,
! CF. REF (H).
!
! DATUM-SHIFT PARAMETERS MAY BE DETERMINED BY THE PROGRAM IN THE
! FORM OF THE CHANGE IN THE LONGITUDE AND LATITUDE COMPONENTS OF THE DE-
! DEFLECTION OF THE VERTICAL AND OF THE HEIGHT-ANOMALY IN A POINT WITH
! GIVEN LATITUDE AND LONGITUDE, CF. REF(E), OR ONE ORE MORE OF THE
! PARAMETERS OF A 7-PARAMETER DATUM SHIFT. IN THIS CASE OBSERVATIONS
! OF THE DIFFERENCE BETWEEN GEOCENTRIC AND LOCAL GEODETIC COORDINATES
! MAY BE USED.
!
! BIAS, TILT AND SCALE-FACTOR PARAMETERS MAY ALSO BE DETERMINED.
! FOURIER COEFFICIENTS ARE BEING IMPLEMENTED, APRIL 2006.
!
! THE DATA USED TO CREATE ONE SOLUTION MAY BE PRESERVED AND USED IN
! ORDER TO REESTABLISH THE SOLUTION OR AS A BUILDING STONE FOR A
! NEW SOLUTION. IN THE FIRST CASE A LOGICAL VARIABLE LWRSOL MUST BE
! TRUE AND IN THE SECOND CASE MUST THE VARIABLE LRESOL BE TRUE.
! THIS VERSION OF THE PROGRAM IS WRITTEN IN FORTRAN95, and split into
! MODULES: MAIN PROGRAM geocol19.f90, and modules m_cholsol.f90
! FOR CHOLESKY REDUCTION USING CHUNKS, AND m_data.f90, m_params.f90.
! ----------------------------------------------------------------------
!   TABLE 1. DATA-KIND CODES AND UNITS USED IN THIS VERSION:
!    DATA-KIND                                CODES:     UNITS:
!  HEIGHT-ANOMALY OR GEOID UNDULATION (ZETA)   1  11     METERS
!  (USE 11 FOR SATELLITE ALTIMETRY)
!  ANOMALOUS POTENTIAL (T) OR POTENTIAL DIFFERENCES
!  (GRACE). CHANGED 2013-09-14.                   51         M**2/S**2
!  GRAVITY DISTURBANCE (-DT/DR)= G-GAMMA          12     MGAL
!  AND DT/DR                                      52     MGAL
!  GRAVITY ANOMALY                              2  13     MGAL
!                                              43  53
!  VERTICAL GRAVITY ANOMALY GRADIENT              14     E.U.
!  VERTICAL GRAVITY DISTURBANCE GRADIENT, TZZ 55  15     E.U.
!  DEFLECTION OF THE VERTICAL, MERIDIAN COM.   3  16     ARCSEC
!                                              43  56
!       -     -   -     -    , PRIME VERTI.    4  17     ARCSEC
!                                              44  57
!  GRAVITY ANOMALY GRADIENT, MERIDIAN COMP.       18     E.U.
!      -      -       -    , PRIME VERT. CO.      19     E.U.
!  GRAVITY DISTURBANCE GRADIENT, MERIDIAN CO-
!  PONENT, TYZ.                                 60  20     E.U.
!     -         -         -    , PRIME VERT.
!  COMPONENt, TXZ                              61  21     E.U.
!  SECOND ORDER DERIVATIVE IN NORTHERN DIRECTION
!  TYY                                         62  22     E.U.
!    MIXED SECOND ORDER DERIVATIVE OF T, TXY  63  23     E.U.
!  2*MIXED SECOND ORDER DERIVATIVE OF T,2TXY  63  23     E.U.
! CHANGED 2013-03-05.
!  SECOND ORDER DERIVATIVE IN EASTERN DIRECTION
!  TXX                                         64  24     E.U.
!  DIFFERENCE BETWEEN SECOND ORDER DERIVATIVES
!  IN PRIME VERTICAL AND MERIDIAN PLANES,TXX-TYY  25     E.U.
!  PAIR OF DEFLECTIONS OF THE VERTICAL      5  26 96 ARCSEC
!                                              45  66
!  PAIR OF HORIZONTAL GRAVITY ANOMALY GRAD.       28 68  E.U.
!  PAIR OF HORIZONTAL GRAVITY DISTURB. GRAD.      30 70  E.U.
!  PAIR OF KIND (25,23)                           35 75  E.U.
!  SECOND ORDER DERIVATIVES (15, 30, 35), ONLY
!  PERMITTED WHEN COLLOCATION IS NOT USED         37    E.U.
!  FULLY NORMALIZED SPHERICAL HARMONIC COEFF.     27
!  ELLIPSOIDAL HEIGHT DIFFERENCE OLD MINUS
!  NEW DATUM VALUES                                6     METERS
!  LATITUDE AND COS(LATITUDE)*LONGITUDE DIFFE-
!  RENCE, NEW MINUS OLD DATUM VALUES.              7     ARCSEC
!  SATELLITE ALTIMETRY CROSS-OVER DIFFERENCE       9     METERS.
!  ANOMALOUS POTENTIAL                             8     M**2/S**2
!  DENSITY CONTRASTS                              10     G/CM**3*
```

```
!                                                                  SCALE FACTOR
! ------------------------------------------------------------------------------
! IF CODE 13 IS USED FOR GRAVITY, SPHERICAL APPROXIMATION IS USED,
! AND IF CODE 2 IS USED, THE POTENTIAL COEFFICIENT SET APPROXIMATION
! IS USED. CODE 13 IS RECOMMENDED IN GENERAL.
! CODES .GT. 40 INDICATE THAT A QUANTITY IS GIVEN IN A LOCAL REFERENCE
! SYSTEM, EAST/NORTH/UP. A LOGICAL VARIABLE LSATP IS PUT TRUE AND AN
! INTEGER ISAT IS INPUT EQUAL TO 1 IF  A ROTATION IN THE HORIZONTAL PLANE
! IS NEEDED, (THEN AZIMUTM MUST BE GIVEN), EQUAL TO 2, 3 OR  4 IF A 3D
! ROTATION IS NEEDE. IF ISAT = 1 THE AZIMUTH MUST BE INPUT.
! IF ISAT = 2 THE AZIMUTH, TILT AND ROLL ANGLES MUST BE INPUT,
! IF ISAT = 3 NO ROTATION IS MADE, AND IF ISAT = 4, THE FULL
! ROTATION MATRIX MUST BE INPUT.
!
! NOTE, THAT IT IS OF ADVANTAGE TO USE OR COMPUTE PAIRS OF QUANTITIES
! BECAUSE COVARIANCES OR CONTRIBUTIONS FROM SPHERICAL HARMONIC EXPAN-
! SIONS MAY BE COMPUTED SIMULTANEOUSLY FOR THESE QUANTITIES.
!
! ------------------------------------------------------------------------------
! TABLE 2:  FILES NEEDED FOR RUNNING THE PROGRAM:
! UNIT NUMBER  USED FOR                          TEMPORARY/PERMANENT
!  5, 6  STANDARD INPUT AND OUTPUT FILES               YES
!  100+X FOR NORMAL EQUATION CHUNKS                        YES IF
!        NUMBER OF FILES AND UNITS SPECIFIED AT INPUT      NEEDED LATER
! INPUT OF ATTITUDE MATRIX ....

! 13     ROTATION MATRIX INPUT FILE                   YES
! 14     DIRECT ACCESS TO STORE ROTATION ELEMENTS     YES
! 15     STORAGE OF PAGED COVARIANCES IF LALLCOV TRUE YES
! 16     DIRECT ACCESS, USED TO STORE OBSERVATION
!        COORDINATES AND THE SOLUTION                 YES
! 17     FORMATTED, USED FOR RESTART FILE OR                 YES
!        RESULT OUTPUT (LWRSOL OR LPUNCH TRUE).
! 18     BINARY, USED FOR STORAGE OF COVARIANCE              YES
!        FUNCTION PARAMETERS ON BINARY FORM.
! 19     BINARY, USED FOR STORAGE OF SOLUTION.             YES
!        OR FOR STORAGE OF COVARIANCES.
! 20     STORAGE OF PREDICTION POINT COORDINATES WHEN
!        ERROR COVARIANCES ARE COMPUTED, ADDED
!        2005-08-09. OR COEFFICIENT PREDICTION RESULTS.    YES
! 21     INPUT OF SHC NEEDED FOR COMPARISON WITH
!        PREDICTED.                                     YES
! 39     USED FOR SHE INPUT IN LOADCS                 YES
! 11     FORMATTED USED FOR OUTPUT OF ERROR IN GRID          YES
!  9     INPUT UNIT FOR POTENTIAL COEFFICIENTS        YES
!        FORMATTED OR BINARY DEPENDING ON "LBIN".
!        USED IN SUBROUTINE LOADCS, AND INPUT UNIT FOR
!        ERROR-DEGREE VARIANCES IN SUBROUTINE INCOV.
!  7     USED TO STORE ERROR COVARIANCES.             YES
!  3     TEMPORARY STORAGE OF POTENTIAL COEFFICIENTS         YES
!        IF DENSITY ANOMALIES ARE USED OR COMPUTED,
!        AND PERMANENT IF LBIPOT IS SET TRUE                  YES
!  2     TEMPORARY STORAGE OF CONTRIBUTIONS FROM      YES
!        DATA ONLY ASSOCIATED WITH PARAMETERS.
!  4 OR DIFFERENT FROM OTHER UNIT NUMBERS, INPUT OF
!        DATA IF LIN4 IS TRUE                         YES
! 12     FILE WITH DETECTED GROSS-ERRORS (LERR,
!        LCOMP, LSTAT MUST BE TRUE).                          YES
!  1     FILE TO HOLD PREDICTED SPHERICAL HARMONIC
!        COEFFICIENT CORRECTIONS NAMED PCOEFF                 YES
! INZ    UNIT FOR DATA INPUT, USE 22-38                       YES
! 99     TRANSFER OF ERROR-ESTIMATES FROM read_single_row
!        TO MAIN PROGRAM.                               YES
! 100    TRANSFER OF SOLUTIONS FROM read_single_column  YES
! > 100  CHUNK-FILES WITH SUFFIX .neq or .err.
! ------------------------------------------------------------------------------
!
! BRIEF SUMMARY OF INPUT SPECIFICATIONS. DETAILS ARE FOUND BELOW
! IN THE MAIN PROGRAM, WITH REFERENCE BACK TO THIS SUMMARY THROUGH
```



```
! THE NUMBERS (1), (2),.. ETC.
!
! (0) INPUT OF LOGICAL VARIABLE (LINTER), TRUE IF INTERACTIVE INPUT.
!     IF FALSE, ALL INPUT INSTRUCTIONS MUST BE STORED IN AN INPUT-FILE.
! (1) INPUT OF LOGICAL VARIABLES DETERMINING THE EXECUTION, AND
!     CONTINGENTLY NAMES OF FILES HOLDING RESTART-FILE, AND OF
!     NORMAL EQUATION FILES.
!     MOST IMPORTANT ARE:
!     LSPHER - THE COMPUTATIONS ARE MADE IN SPHERICAL APPROXIMATION.
!     LTRAN - DATA MAY HAVE TO BE TRANSFORMED FROM A LOCAL DATUM TO
!       GEOCENTRIC SYSTEM NOT ALREADY DEFINED IN SUBROUTINE ICOSYS.
!     LPOT  - FIRST OBSERVATION SET IS POTENTIAL COEFFICIENTS.
!     LPARAM- DATUM OR BIAS PARAMETERS TO DE DETERMINED.
!     LNCOL - NO COLLOCATION SOLUTIONS ARE WANTED.
!     LIOSOL- ESTABLISH OR USE RESTART FILES ON CHARACTER OR BINARY
!       FORM (UNITS 17, 18 AND 19).
!     IF LIOSOL IS TRUE, INPUT OF 5 LOGICAL PARAMETERS:
!     LWRSOL- WRITE RESTART FILE ON UNIT 17.
!     LBIPOT- WRITE BINARY FILE WITH POTENTIAL COEFFICIENTS ON UNIT 3.
!     LBICOV- OUTPUT COVARIANCE FUNCTION PARAMETERS ON BINARY FORM,
!     LBISOL- OUTPUT SOLUTION ON BINARY FORM,
!     LINSOL- INPUT CATALOGUE OF COVARIANCE FUNCTION PARAMETERS,
!       SOLUTIONS AND BOUNDARIES FOR VALIDITY OF SOLUTION ON BINARY
!       FORM. (1986.10.20 ONLY IMPLEMENTED ON RC8000).
!     LMTEST- GENERATES TEST-OUTPUT.
!     INTEGER NPROC, NUMBER OF THREADS TO BE USED.
! WHEN LNCOL IS FALSE:
!     INTEGER NN,NBB: BLOCK SIZE WITHIN CHUNK AND NUMBER OF BLOCKS
!       IN A CHUNK 'ROW'.
!     CHARACTER*128 DNANE(1,1), ROOT NAME FOR FILES.
! (2) INPUT OF PARAMETERS FOR GEOCENTRIC SYSTEM, (IF SYSTEM DEFINITION
!     NOT ALREAD GIVEN IN SUBROUTINE ICOSYS).
! (3) IF LTRAN IS TRUE INPUT OF PARAMETERS FOR SYSTEM IN WHICH DATA
!     CONTINGENTLY ARE GIVEN, AND PARAMETERS FOR TRANSFORMATION TO
!     GEOCENTRIC SYSTEM. THE SYSTEM IDENTIFICATION CODE IS 0 FOR THIS
!     SYSTEM.
! IF LINSOL IS TRUE, JUMP TO (9).
! (4) IF LPOT IS TRUE INPUT OF SPECIFICATIONS FOR POTENTIAL COEFFI-
!     CIENTS, INCLUDING A PARAMETER LFM, WHICH IS TRUE WHEN THE
!     COEFFICIENTS ARE IN THE STANDARD INPUT FILE (UNIT 5) AND
!     FALSE, IF THEY ARE INPUT THROUGH UNIT 9.
! (5) IF LFM IS TRUE INPUT OF COEFFICIENTS.
! IF LNCOL IS TRUE, JUMP TO (15)
! (6) GENERAL SPECIFICATION OF COVARIANCE FUNCTION TYPE. (SUBR. INCOV).
! (7) DETAILS CONCERNING COVARIANCE FUNCTION, AND CONTINGENTLY
!     CONCERNING TABLES USED FOR FAST INTERPOLATION OF VALUES.
! (8) IF LPARM IS TRUE SPECIFICATION OF PARAMETERS TO BE DETERMINED,
!     AND OF A SCRATCH FILE TO BE USED.
! (9) INPUT OF SPECIFICATION (FORMAT AND SEQUENCE OF ELEMENTS) OF
!     DATA SET, INCLUDING VALUE OF LIN4, TRUE IF THE OBSERVATIONS
!     ARE INPUT FROM UNIT INZ (NORMALLY EQUAL TO 4). (SUBR. DEFDAT
!     AND INHEAD).
! (10) IF LIN4 IS FALSE, THEN INPUT OF OBSERVATION RECORDS FROM
!     UNIT 5 ELSE FROM INZ. (SUBR. INP10).
! WHEN LAST RECORD IS ENCOUNTERED:
! (11) INPUT OF LSTOP, TRUE IF THE DATA SET IS THE FINAL ONE CONTRI-
!     BUTING TO THE CURRENT COLLOCATION STEP, AND OF LRESOL, TRUE
!     IF THE SOLUTIONS OR THE REDUCED NORMAL EQUATION MATRIX ARE
!     TO BE INPUT OR RE-USED, RESPECTIVELY.
! IF LSTOP IS FALSE, JUMP TO (9)
! (12) IF LRESOL IS TRUE, INPUT OF LSANEQ AND IFC. LSANEQ IS TRUE
!     IF THE IFC FIRST REDUCED COLUMS OF THE NORMAL EQUATIONS ARE
!     STORE.
! (13) IF IFC IS EQUAL TO THE TOTAL NUMBER OF OBSERVATIONS AND LRESOL
!     IS TRUE, INPUT OF THE SOLUTIONS. (OTHERWISE, THE LAST COLUMNS
!     WILL BE ESTABLISHED, REDUCED, AND THE EQUATIONS SOLVED).
! IF THE FIRST COLLOCATION STEP NOW IS TERMINATED, INPUT OF VARIABLES
! TELLING WHETHER A SECOND STEP SHOULD BE MADE, OTHERWISE JUMP TO (15).
! (14) INPUT OF LCREF, TRUE IF A SECOND STEP IS TO BE MADE AND OF
```

```
!     LPARM, TRUE IF PARAMETERS ARE TO BE DETERMINED IN A SECOND STEP.
! IF LCREF IS TRUE, THEN JUMP BACK TO (7).
! (15) INPUT OF LGRID, TRUE IF PREDICTIONS ARE TO BE MADE IN A GRID,
!     LERNO, TRUE IF ERRORS OF PREDICTION ARE TO BE COMPUTED OR RE-
!     PRODUCED IN OUTPUT (LNCOL TRUE), LCOMP, TRUE IF OBSERVED
!     AND COMPUTED QUANTITIES ARE TO BE COMPARED (DIFFERENCED) AND
!     LSPHAR, TRUE IF CORRECTIONS TO SPHERICAL HARMONIC COEFFICIENTS
!     ARE TO BE DETERMINED.
! IF LSPHAR OR LGRID ARE FALSE, THEN INPUT (9) AND (10), THEN JUMP TO (18).
! (16) INPUT OF GRID SPECIFICATIONS (START POINT, STEPS ETC.).
! (17) IF LCOMP IS TRUE, INPUT OF OBSERVATIONS IN THE GRID POINTS.
! (18) INPUT OF LSTOP. IF IT IS FALSE, JUMP TO (15), OTHERWISE
!     THE PROGRAM WILL TERMINATE.
!
! REFERENCES:
!
!  REF(A): TSCHERNING,C.C: COVARIANCE EXPRESSIONS FOR SECOND AND LOWER
!          ORDER DERIVATIVES OF THE ANOMALOUS POTENTIAL. REPORTS OF
!          THE DEPARTMENT OF GEODETIC SCIENCE NO. 225, THE OHIO STATE
!          UNIVERSITY, COLUMBUS, 1976.
!  REF(B): TSCHERNING,C.C.: A FORTRAN IV PROGRAM FOR THE DETERMINATION
!          OF THE ANOMALOUS POTENTIAL USING STEPWISE LEAST SQUARES
!          COLLOCATION, DEPARTMENT OF GEODETIC SCIENCE, THE OHIO STATE
!          UNIVERSITY, REPORT NO. 212, 1974.
!  REF(C): HEISKANEN W.A. AND H.MORITZ: PHYSICAL GEODESY, 1967.
!  REF(D): TSCHERNING,C.C.: COMPUTATION OF THE SECOND-ORDER
!          DERIVATIVES OF THE NORMAL POTENTIAL BASED ON THE
!          REPRESENTATION BY A LEGENDRE-SERIES. MANUSCRIPTA
!          GEODAETICA, VOL.1, PP. 71-92, 1976.
!  REF(E): TSCHERNING,C.C.: DETERMINATION OF DATUM-SHIFT PARAMETERS
!          USING LEAST SQUARES COLLOCATION, BOLL.GEODESIA SC. AFF.,
!          ANN. XXXV, NO. 2, 1976.
!  REF(F): TSCHERNING,C.C: IMPLEMENTATION OF ALGOL-PROCEDURES FOR
!          COVARIANCE COMPUTATION ON THE RC 4000-COMPUTER. THE
!          DANISH GEODETIC INSTITUTE INTERNAL REPORT NO. 12, 1976.
!  REF(G): SANSO, F. AND W.-D. SCHUH: FINITE COVARIANCE FUNCTIONS.
!          BULLETIN GEODESIQUE, VOL. 61, PP. 331-347, 1987.
!  REF(H): TSCHERNING, C.C.: PREDICTION OF SPHERICAL HARMONIC
!          COEFFICIENTS USING LEAST-SQUARES COLLOCATION. JOG, 2001.
!  REF(I): TSCHERNING, C.C.: LOCAL GRAVITY FIELD APPROXIMATION,
!          PROC. BEIJING INT. SUMMER SCHOOL, PP. 277-261, 1984.
!  REF(J): Kaas. E., B. Soerensen, C. C. Tscherning, M. Veichert:
!          Multi-Processing Least Squares Collocation Applications to
!          Gravity Field Analysis. J. of Geodetic Science,p. Vol. 3,
!          no. 3, 219-223, 2013.  DOI  10.2478/JGOS-2013-0025 .
!
!-----------------------------------------------------------------!

 USE m_params,      ONLY : NDIMC,NISIZE,NCRW,NNBL,MAXO                 ! PARAMETE
RS GIVING THE SIZE OF THE ARRAYS HOLDING POTENTIAL
 USE m_params,      ONLY : NCOEFF,NROOT,NNSU,NEQFIM                    ! COEFFICI
ENTS FOR MAX=(NNSU/10) (REALS), NNSU/5 (INTEGERS). THEY ARE ALSO FOUND IN SETCM,
 LOADCS, GEOCOLH, GPOTDR AND CXPARM. NCOEFF IS (2201)**2, TO HOLD EGM08.
 USE m_params,      ONLY : NSAT,MXPAR,MAXCX,NMAP,NIPT,NIPCAT,&
                           INBLP,NSPHAR,MAXSA,IIMAX

! MOST OF THE USE VARIABLES HAVE BEFORE 2013-04-20 BEEN IN COMMON BLOCKS.

 USE m_geocol_data, ONLY : C,NCAT,ISZE,NBL,MAXBL,ISIZE,MAXCM,MI1,MI2,MMAXB,&
                           MAXFIL,NEQFI,NEQFMA,MAXBNE,DNANE,LNBL1,ROOT0,FMIOLD
 USE m_geocol_data, ONLY : SR11A,SR12A,SR13A,SR22A,COSAZA,SINAZA
 USE m_geocol_data, ONLY : SR11 ,SR12 ,SR13 ,SR22 ,COSAZ ,SINAZ, SATROT
 USE m_geocol_data, ONLY : SIGMAP,SLOP,SLOQ,CLOP,CLOQ,IIDEG,JJORD
 USE m_geocol_data, ONLY : IDLAT,MLAT,SLAT,IDLON,MLON,SLON,NOX,LFORM,CLATD
 USE m_geocol_data, ONLY : NFILTE,STEPE,STEPN,COSSTE,SINSTE,COST2P,SINT2P,&
                           COSSTN,SINSTN,FILTER,NFOURI,SCFRDD
 USE m_geocol_data, ONLY : LFOUR,NFOUR,SCFACT,RDD
 USE m_cholsol,     ONLY : NN,NBB,file_root_name,file_root_name_old,NPROC,ltest2
 USE m_cholsol,     ONLY : copy_files, time_stamp
```



```
 USE m_data,        ONLY : LSPOUT,NDSET,LCZERO,LLCOEE,LFOURI
 USE m_geocol_data, ONLY : PW2,E21,GM1,AX1,E21,F1,UREF,GREF,SM,KP,IPC,LSMAL,&
                           LADBTE,LNGR,LKSIP,LNCOL,LTNB,LTEB,LOE1,LOE2,LE
 USE m_geocol_data, ONLY : OLDN,NAI,NLA

 USE m_geocol_data, ONLY : FG,FJ,LP,OMEGA2,IORDER
!COMMON /GRAVCC/FG(30),FJ(30),LP(2),OMEGA2,IORDER
! COMMON VARIABLES USED IN GRAVC AND RGRAV, HOLDING I.E. COEFFICI-
! ENTS OF LEGENDRE SERIES OF NORMAL POTENTIAL AND NORMAL GRAVITY
! FORMULA.

 USE m_data,        ONLY : LOPEN4,LDENOL,RLOMAX,RLAMIN,LCLU7,LOPEN7, &
                           LTCOV,LERNO,LIN4,NO,LCOMP,LMDD,LWLONG,    &
                           LBIPOT,LBICOV,LBISOL,LINSOL,LONEQ,LTIME
 USE m_geocol_data, ONLY : EE0,DSHIF0,MODEC0
 USE m_data,        ONLY : ITRACE,LCOERR,LLCOER

 USE m_geocol_data, ONLY : ITRACK,CTIME,LCTIME,NERCOV
!COMMON /CCTIME/CTIME(NIPCAT),ITRACE(NIPCAT),ITMODE,ITM0,ITMOD,ITRGAP,ITRACK,&
! ITOLD,NERCOV,LCTIME,LCOERR,LLCOER
! SEE SUBROUTINE PARCAT FOR DESCRIPTION OF CPARM. THE COMMON BLOCK
! IS ALSO IN CXPARM, GEOCOLH, WRPAR, BLOCK DATA AND PRED.
! ADDED 1997-07-15 AND CHANGED 2005-03-20: ITRACE IS USED
! TO IDENTIFY CORRELATED OBSERVATIONS. IF ITRACE(N)=ITACE(M) AND < 0 THEN
! THE OBS N AND M HAVE CORRELATED ERRORS. THE GLOBAL VARIABLE LCOERR MUST
! BE TRUE.

 USE m_geocol_data, ONLY : ALLREF,ALLGG,ALLCOL,ALLPRE,ALLTRA,ALLPR1,ALLERR,&
                           ALLVAR,ALLIN,LALLCO
! VARIABLES USED WHEN COMPUTATING ALL COMPONENTS OF THE
! DERIVATIVE OF THE  POTENTIAL 2005-09-21. T IS
! STORED IN (1,1), DG IN (1,1),1,2),(1,3) AND THE GRAVITY
! GRADIENTS IN ALL ENTRIES (EAST, NORTH, UP).

 USE m_geocol_data, ONLY : SUMIJ,CCCIJ,SQ2,YS,YC,VV,V1,GS,GC,DDS,DDC,SN2,AXS,&
                           GMS,IIOLD,JOLD,IR,LSPHAR,LTSPH
!COMMON /CON3/SUMIJ((NSPHAR+1)**2),CCCIJ((NSPHAR+1)**2),SQ2,YS,YC,VV,V1,GS(3),GC
(3),&
! DDS(3,3),SN2(0:NSPHAR),AXS,GMS, DDC(3,3),IIOLD,JOLD,IR,LSPHAR,LTSPH
! HOLDS VARIABLES USED IN SPHAR CALC. 1999-05-17.

 USE m_data,        ONLY : OBS
!COMMON /OBSER/OBS(22)

 USE m_data,        ONLY : LC1,LC2,LCREF,LNEWD,LRESOL,LGRID
!COMMON /DAT/ LNEWD,LRESOL,LGRID
! /DAT/ TRANSFERS LOGICAL VARIABLES TO THE SUBROUTINE ITRAN.

 USE m_data,        ONLY : BSIZE,ANDEX,PRETAP,PREDP,HCZERO,NCZERO

 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
! COMMON CONSTANTS D0=0.0D0  ETC.
 USE m_data,        ONLY : LDIF
 USE m_geocol_data, ONLY : HQD,RLATD,SINLATD,COSLATD,RLONGD,SINLOND,COSLOND
! VARIABLES USED WITH DIFFERENCES (GRACE),

 USE m_geocol_data, ONLY : B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,  &
                           WOBS,ISAT,SINLOP,COSLOP,BSIZEN,BSIZEE,COSLAP, &
                           SINLAP,RLONGP,RP,CAZP,SAZP,HP,RLATP,ICZERO,   &
                           NI,NR,IKP,ISATP,NOBLK,LONECO,LNKSIP,LNETAP,   &
                           LDEFVP,LOBSST,LSTART,CFA

 USE m_geocol_data, ONLY : INUMR,NO1,K2,K3,K3P2=>K2P3,K4,IU,K21,IU1,IANG,LPUNCH,
&
                           LTERMA,LTERMO,LSTNO=>LTERM, LOUTC,LTRAN,LNERNO,LK30,L
K31,&
                           LWRSOL,LSTOP,LK2EQ4,LNUOUT
!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
! IN OUTC ARE STORED SUBSCRIPTS OF VARIABLES TO BE OUTPUT AND LIMITS
! FOR DO-LOOPS IN OUTPUT. NOTE THAT OUTC OCCURS IN SUBROUTINES
! HEAD, COUT, CXPARM AND THE BLOCK DATA MODULE.

 USE m_data,        ONLY : OLDT,OLDR,LFIRST
!COMMON /GPOTC1/OLDT,OLDR,CFA,IGQ(12),LFIRST,HP9000
! COMMON VARIABLES USED IN GPOTDR, SETCM ,LOADCM, PRED AND CXPARM.

 USE m_data,        ONLY : OLDB,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
!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

#ifdef _MPI
 USE m_MPI,         ONLY : MPI_initialize,MPI_finish,MPI_stop, &! subroutines/fu
nctions
                           MPI_numprocs,MPI_pid,MPI_ierr        ! variables
 USE m_input,       ONLY : open_job
#endif
!
! THE FIRST 9 QUANTITIES ARE PAGED TO  UNIT 16, IF THE NUMBER OF OBS EXCEEDS MAX
O.
! B    THE CONSTANTS (SOLUTIONS),
! HQ   THE HEIGHT OF THE OBSERVATION POINT (Q),
! RLAT,SINLAT,COSLAT THE LATITUDE (RADIANS) AND COS AND SIN.
! RLONG,SINLON,COSLON, LONGITUDE (RADIANS), SIN AND COS,
! WOBS THE OBSERVATION ERRORS
! COSAZ, SINAZ: COS AND SIN OF AZIMUTH (E.G. FOR TRACK),
! SINLOP, COSLOP, SIN AND COS OF LONGITUDE OF PREDICTION POINT P,
! BSIZE, BLOCKSIZE, BSIZEN, BSIZEE BLOCK SIZE IN NORTH AND EAST,
! COSLAP, SINLAP COS AND SIN OF LATITUDE OF P,
! RLONGP LONGITUDE OF P (RADIANS),
! RP   THE DISTANCE OF P FROM THE ORIGIN,
! CAZP, SAZP COS AND SIN OF AZIMUTH OF P,
! HP, RLATP HEIGHT AND LATITUDE OF P,
! PRETAP, PREDP PREDICTED VALUE OF 2 COMPONENT (E.G. ETA) AND OF P.
! HCZERO,ICZERO,NCZERO
! NI, NR COUNTERS OF OBS AND OBS POINT (THERE MAY BE TWO OBS
!     PER POINT)
! ANDEX  THE CATALOGUE OF THE OBSERVATIONS (ANDEX),
! IKP THE OBSERVATION TYPE,
! ISAT   CATALOGUE OF OBSERVATIONS ATTITUDE DEPENDENCE (POINT,
!        HORIZONTAL PLANE ROTATION, 3-D ROTATION).
! ISATP  ATTITUDE DEPENDENCE OF P.
! NOBLK  CURRENT NUMBER OF BLOCK WHERE DATA ARE STORED ON UNIT 16.
! LOBSST TRUE IF DATA ARE STORED ON UNIT 16.
! THE OTHER LOGICAL VARIABLES ARE USED TO DISTINGUISH
! BETWEEN THE DIFFERENT PREDICTION SITUATIONS.
!
!COMMON /CMCOV/CCI(24),CCR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HCMAX,CCV(2,2),DC(36),KCI(37),NC1,NC2,LOCAL,LSUM
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,DC,HCMAX,KCI,KSAT
 USE m_geocol_data, ONLY : CCR,SIGMAX,CCV,NC1,NC2,LOCAL,LSUM
!
!COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFX,KSAT(17,2),LSATPP,LSATQ,NDX1(5),NDX2(5)
,NDP,NDQ,NWAR,LY,LX(7,5),LNX(7,5),LTESTS
 USE m_data,        ONLY : LTESTS,NWAR,LY
 USE m_geocol_data, ONLY :      CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
                           NDP,NDQ,LX,LNX
!COMMON /CMEAQ/STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE, COST2Q,SINT2Q
! STEPSIZES USED WHEN CALCULATING MEAN VALUES.
 USE m_geocol_data, ONLY : STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE, COST2Q,SINT2
Q
!COMMON /CPARM/SFACT(NIPCAT),FPERIO(10,2),IPTYPE(NIPT),IPACAT(3*NIPT),ITIME(NIPC
AT), ITIME0(NIPT),ITROLD,ICODE(10),NPARM,NPARM1,MAXPAR,MP,IPA,NCXLAS
 USE m_data,        ONLY : IPTYPE,ITIME,ITIME0,NPARM,NPARM1
```



```
 USE m_geocol_data, ONLY : SFACT,FPERIO,IPACAT,ITROLD,ICODE,MAXPAR,MP,&
                           ipa,ncXLAS,DXX,NUM,VARI,SCALE,SCALE2,INN,INV,&
                           COFF,AZP,BETP,TAUP,JR0
!COMMON /GPOTC3/COFF(NCOEFF)
!COMMON /CPARM/SFACT(NIPCAT),FPERIO(10,2),IPTYPE(NIPT),IPACAT(3*NIPT),ITIME(NIPC
AT),&
! ITIME0(NIPT),ITROLD,ICODE(10),NPARM,NPARM1,MAXPAR,MP,IPA,NCXLAS
! FPERIO IS USED WHEN ESTIMATING FOURIER COEFFICIENTS OF PERIODIC
! PHENOMENA, LIKE OCCURRING TWICE PER REVOLUTION FOR SATELLITE DATA.
! FPERIO(I,1) HOLDS PERIOD AND FPERIO(I,2) HOLDS PHASE IN RADIANS FOR
! I .LT.11 AND I.GT.3. 2006.03-20.
! QUASI NORMALIZED SPHERICAL HARMONIC COEFFICIENTS, UNITLESS.
!COMMON /COM2/DXX,NUM(70),VARI(32),SCALE,SCALE2,INN,INV
! USED BY COMPA, COMPARING OBSERVED AND PREDICTED QUANTITIES.
 USE m_geocol_data, ONLY : IMAX1,IMAX1R
 USE m_data,        ONLY : OLDCOV,S,SR,AAI,AAR,IS,IPX,LCO1,NBOLD
!COMMON /BIPARC/ OLDCOV(2),S,SR,AAI,AAR,NBOLD,IS,IPX, IMAX1,IMAX1R,LCO1
! DATA USED WHEN STORING SOLUTIONS OR COVARIANCE FUNCTION ON
! BINARY FORM. (CHANGE MADE NOV 1986).
 USE m_data,        ONLY : 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 /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
  USE m_geocol_data, ONLY : VARNO,PPA,PPS,VG,HPK,DM,DA,WM,REJLEV,SGRE,ROTFIL,&
                            ERNAME,DNAME,FMT,NSTEP,NSTEPE,IDSAT,ILA,ILO,IKPREF,&
                            INZ,IO2,NPOBS0,NOUSE,ILAST,IPAMAX,NGR,NGRE,ICSYS, &
                            LMEAN,LSA,LADMU,LSTAT,LAREA,LZETA,LGRADI,LMENSI,&
                            LSIMH,LMEAN1,LINTRA,LGIGRS,LMEGR,LUSNGS,LSATAC,&
                            LSATP,LGRERR,LCOD,LEQP,LALLP,LGRP,LDEN,LPOTSD,&
                            LEQANG,LFILTE,LBIN,LSKIPL,LGRERS,LTILT,LSCALE,LINERT
!COMMON /CDEFCA/VARNO,PPA,PPS,VG,HPK,DM,DA,WM,REJLEV, SGRE(10), ROTFIL,ERNAME,&
! DNAME(2),FMT(9),NSTEP,NSTEPE,IDSAT, ILA,ILO,IKPREF,&
! INZ,IO2,NPOBS0,NOUSE,ILAST, IPAMAX,NGR,NGRE(10),ICSYS, LMEAN,&
! LSA,LADMU,LSTAT,LAREA,LZETA,LGRADI, LMENSI,LSIMH,   &
! LMEAN1,LINTRA,LGIGRS,LMEGR,LUSNGS, LSATAC,LSATP,LGRERR,LCOD,LEQP,LALLP,
! LGRP,LDEN,LPOTSD, LEQANG,LFILTE,LBIN,LSKIPL,LGRERS,LTILT,LSCALE,LINERT
! TRANSFERS VARIABLES FROM DEFDAT.

  USE m_geocol_data, ONLY : X,Y,Z,XY,XY2,DISTO,DIST2
!COMMON /EUCL/X,Y,Z,XY,XY2,DISTO,DIST2, EUCLIDIAN COORDINATES OF A POINT, THE
! DISTANCE AND THE SQUARE OF THE DISTANCE FROM THE Z-AXIS XY, XY2 AND
! THE DISTANCE AND THE SQUARE OF THE DISTANCE FROM THE ORIGIN DIST0
! AND DIST2.

 USE m_geocol_data, ONLY : DZERO,ROOT
!COMMON /SQROOT/DZERO,ROOT(NROOT), SQUARE-ROOT TABLE USED IN GPOTDR.

 USE m_geocol_data, ONLY : C20IN,G1,G2,CM3,CMM2,CM1
!COMMON /GPOTC0/ C20IN,G1(3),G2(3,3),CM3,CMM2,CM1
! C20IN HOLDS C20, G1 THE FIRST DERIVATIVES, G2 THE SECOND DERIVATIVES.

 USE m_geocol_data, ONLY : SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS3,EPS2,EPS1,DL,AX2,E
22
!COMMON /ITRANC/SINLA0,COSLA0,RLONG0, DX,DY,DZ,EPS3,EPS2,EPS1,DL,AX2,E22
! DATUM SHIFT PARAMETERS.

 USE m_timing,      ONLY : TIMER,PRINT_TIMES
!
IMPLICIT NONE

 CHARACTER(LEN=60) :: cvs_id='$Id: geocol19.f90,v 1.362 2014/04/09 09:41:50 trie
r Exp $'

 INTEGER ::      omp_get_num_threads
 INTEGER ::      omp_get_thread_num
 INTEGER ::      omp_get_num_procs
#ifndef _MPI
 INTEGER, PARAMETER :: MPI_pid = 0
```

```
#endif

 integer, dimension(8)                  :: T1,T2,T0

! I/O status (IOS) included (2014-03-25)
 INTEGER ::      MAXO6,MAXO9,ICHAR,ICSYS0,NJ,KL,  N2,   ICREL,NREL,&
                 NPAOLD,NPOBS,NBL2,IXS,STOPFILE,&
! STOPFILE INTRODUCED 2012-12-17.
                 JJOR,IIDEGM,JJORDM,IIDEG2,NII,MII,IHH,JXS,              &
                      IDEG21,MAXDOU,IHX,IHH2,IHH21,IHJ,&
                 NLO,JIMAX,JI,NOI,MAXCXB,ITO1,     &
                 NOI1,IPOS,NGRERR,   ITRAC0,INDG,&
                 NO2,NPNO,KK,NREL0,IPR,NLAST,NFIRST,JJE,NGRR,IGRR,      &
                 IGG,K2P3,MAXSA9,NERRM,NPRED,&
                 NPRED1,I61,I62,I63,NPOINT,NCXP,&
                 IBSS,MAXOBS,IFC,NERR,    JJOR1,IIDEG1,JJ1,&
                 I,J,K,      N19,N20,N21,NI0,M

 LOGICAL      :: LMTEST,LNOUSE,lopen20,LMAP7,LMAP7E,&
                 LFOR77,LOPEN,LEXIST,&
! LOPEN AND LEXIST  ADDED 2012-10-01 TO BE USED in INQUIRE.
                 LERCOV,LNDAT,  LCHANG,&
                 LFM,       LMAP,LNPOT,&
                 LNFORM,LREPEC,LINVDE,LADDBC,&
                 LNEWDA,       LADDBP,LADBA,LCOLLO,LLEG,&
                 LIOSOL,  LDPR,LTEST,LUNIX,&
                 LFOUND,LINTER,LOK,L386,&
                 LSPHER,LFULLO,&
                 LOUTCO,LEROUT,LWAIT,LINT
! LALLCO IS TRUE IF ALL DERIVATIVES ARE COMPUTED SIMULTANEOUSLY.

 REAL(KIND=8) :: RCBASE,CPU0,GG,F2,GM2,UREF0,&
                 DKSI0,DETA0,DZETA0,COSDLO,SINDLO,&
                 W2,W,RN,RM,X1,DGVAR,SUS,SS,SSC,SIGD,SSOBS,CPU1,&
                 CCII,CCJJ,SII,SSII,SOERR,TMEAN,TSTDV,TVARI,SSCO,&
                 RLATP1,      OERR,OERR1,TCOBS,DIFII,SI,SSI,SCO,DIFI,&
! OERR! ADDED 2013-04-09.
                 SLAC,SLOC,DEGRAD,SINB,SINT,COST,RRE,&
                 GLA,GLO,H,H0,PI4,PW,&
                 COSB,RLATS,RLONGS,SINLA,COSLO,COSLA,SINLO,REF,REF1,REFI,REF2,&
                 REF3,COSLA1,REF0,POT,GP,DUDX,DUDY,POT00,&
                 OB1,RB,HPP,DLATP,OB2,    CPU2,CPU5,VAR,SYTIME,&
                 SU1,CU,RGRAV,RLAT0
! THE VARIABLES WILL HOLD ALL DERIVATIVES OF THE REFERENCE POTENTIAL,
! THE SPHERICAL HARMONIC MODEL, THE COLLOCATION COMPUTATION AND
! THE PREDICTION. ADDED 2005-09-22.

 CHARACTER(LEN=128) :: UDATE, PCOEF
 CHARACTER(LEN=128) :: CCFILE,DCOVA,DERCOV,POSFIL

 REAL(KIND=8) :: TIMEARRAY(2)

 INTEGER, DIMENSION(NMAP)               :: IMAP

 REAL(KIND=8), DIMENSION(16)            :: PREDCO
 REAL(KIND=8), DIMENSION(3)             :: VREF
 REAL(KIND=8), DIMENSION(10)            :: PRV
 REAL(KIND=8), DIMENSION(22)            :: OBI
 REAL(KIND=8), DIMENSION(MAXCX)         :: CX
 REAL(KIND=8), DIMENSION(3,3)           :: RG
 REAL(KIND=8), DIMENSION(0:NSPHAR,4)    :: TOERR
 REAL(KIND=8), DIMENSION((NSPHAR+1)**2) :: TCOEFF
 CHARACTER(LEN=128), DIMENSION(2)       :: PNAME,PNA0,CNANE
 CHARACTER(LEN=128), DIMENSION(300)     :: SNAME
!MUST BE USED IF ISYS0 IS ACTIVATED

 REAL(KIND=8), DIMENSION(NNSU)          :: SU8

 REAL*16,      DIMENSION(NNSU)          :: SU
```



```
! THIS QUADRUPLE PRECISION ARRAY IS USED WHEN EVALUATING SPHERICAL
! HARMONIC SERIES OF HIGH DEGREE.
! ONE OR MORE OF THE FOLLOWING PARAMETERS ARE ALSO FOUND IN THE
! SUBROUTINES GEOCOLH, INCOV, BLKDTA000, PRED. THEY ARE ALSO FOUND IN THE MODULE
S
! INCOV, ETC.

 REAL(KIND=8), DIMENSION(0:(NSPHAR+1)**2) :: PRCOEF
! THE ARRAYS HOLDS PREDICTED COEFFICIENTS AND ERROR-ESTIMATES OF
! THE SAME DEGREE. CHANGED 2011-08-19 SO THAT ERCOEF HOLDS  ALL VARIANCES.

 CHARACTER(LEN=128) :: testline
!-------------------------------------------------------------------------------
!MPI
#ifdef _MPI
  call MPI_initialize
  if ( MPI_pid == 0 ) then
    inquire(file='runfile',EXIST=LEXIST)
    if (LEXIST) go to 9999
    open(91,file='runfile',STATUS='NEW')
    STOPFILE=91
    print *,'number of MPI procs: ',MPI_numprocs
  end if
  print *,'MPI_pid: ',MPI_pid
  call open_job(MPI_pid)
#else
  inquire(file='runfile',EXIST=LEXIST)
  if (LEXIST) go to 9999
  open(91,file='runfile',STATUS='NEW')
  STOPFILE=91
#endif
  call timer('Total',1)
!-----------------------------------------------------------------------!

 if ( MPI_pid == 0 ) WRITE(6,*) cvs_id

 file_root_name_old = 'a'
 RCBASE = D0
 CPU0   = SYTIME(RCBASE,TIMEARRAY)
 NERRM  = 0
 NERR   = 0

! HEADINGS AND DEFINING CONSTANTS.
!
! LFOR77  IS TRUE IF RUN USING A FORTRAN 77 OR  95 COMPILER,
 LMTEST=.FALSE.
 LUNIX=.TRUE.
 LFOR77=LUNIX

! THE LOGICAL VARIABLE LDPR IS TRUE, WHEN DOUBLE PRECISION IS USED.
! THIS IS USED, TO BREAK SOME LONG INTEGER STATION NUMBERS IN TWO.
 LDPR=.TRUE.
 LFILTE=LF
 NFILTE=5
 LSPHAR=LF
 NPRED=0
 LTSPH=LF
 LCZERO=LF
 LFULLO=LF
 LALLCO=LF
 LNBL1=LF
 LSATAC=LF
! THIS VARIABLE IS TRUE FIRST TIME UNIT 20 IS OPENED. 2012-06-05.
 LOPEN20=LF

! NEXT STATEMENTS ADDED 2004-11-05 AND 2012-11-27.
 LGRERR=LF
 LGRERS=LF
 LNOUSE=LF
```

```
 LMAP7E=LF
 LEROUT=LF

! FULL PRECISION PI ADDED 2003-06-01.
 PI4=ATAN(1.0D0)
 PI=4.0D0*PI4
 DEGRAD=PI/180.0D0
 SQ2=SQRT(D2)

 DO I=1, IIMAX
  ROOT0(I)= SQRT(DFLOAT(I-1))
 END DO

! if ( MPI_pid == 0 ) WRITE(6,'(a,a)')&
!  ' GEODETIC COLLOCATION, VERSION 2014-02-14  RELEASE', ' 19 '
 if ( MPI_pid == 0 ) then
 WRITE(6,104)
  104 FORMAT(' GEODETIC COLLOCATION, VERSION 2014-02-14  RELEASE',&
 ' 19 ')

 end if
! *** THE CALL OF ATIME MUST BE DELETED IN A NON UNIX ENVIRONMENT ***
! APP IF (LUNIX)I=ATIME(UDATE)
  call DATE_AND_TIME(VALUES = T1)
  T0=T1
  call time_stamp(T1,T2,0)
  T1=T2
 IF (LUNIX) THEN
  CALL FDATE(UDATE)
  WRITE(6,*)UDATE
 END IF
 if ( MPI_pid == 0 ) then
 WRITE(*,112)
  112 FORMAT(/' NOTE THAT IF SPHERICAL APPROXIMATION IS USED',&
 /' MEAN RADIUS = RE = 6371 KM AND MEAN GRAVITY 981 KGAL ',&
 'USED.',/)
 WRITE(6,113)MAXO,MXPAR,NIPCAT,NDIMC,NCOEFF
  113 FORMAT(/,&
 ' MAX NUMBER OF OBS PER RECORD =',I9,&
 ', MAX NUMBER OF PARAMETERS=',I5,/,&
 ' MAX NUMBER OF DATA DEPENDING ON TILT ',&
 ' OR SCALE FACTOR-PARAMETERS ',I7,/,&
 ' SIZE OF NORMAL EQ. BLOCKS=',I10,', SIZE OF POT.COFF. BLOCK=',I8)
!IF (MAXOD.NE.9*MAXO) WRITE(*,*)' **** WARNING0, MAXOD= ',MAXOD
  end if
!
! *************** INPUT (0) *********************************
!
 WRITE(6,*)' INTERACTIVE INPUT (T/F)'
 READ(5,*)LINTER
 IF (LUNIX) THEN
  MAXO6=MAXO*6*8
  MAXO9=MAXO*9*8
  MAXCXB=MAXCX*8
  MAXSA9=MAXO*6*8
 ELSE
  MAXO6=MAXO*6*2
  MAXO9=MAXO*9*2
  MAXCXB=MAXCX*2
  MAXSA9=MAXO*6*2
 END IF
 if ( MPI_pid == 0 ) WRITE(*,*)' BUFFER SIZE MAXO9 = ',MAXO9
!
! *************** INPUT (1) *********************************
!
! INPUT OF 8 LOGICAL VARIABLES:
! LSPHER  = TRUE IF CALCULATIONS ARE DONE IN SPHERICAL APPROXIMATION.
! LTRAN   = INPUT VALUES FOR USER DEFINED DATUM (CODE 0).
! LPOT    = POTENTIAL COEFFICIENTS ARE TO BE USED AS FIRST SET OF
```



```
!           OBSERVATIONS.
! LTEST   = TEST OUTPUT NEEDED ACCORDING TO SPECIFICATIONS IN INPUT
!           1D.
! LLEG    = OUTPUT LEGEND OF TABLES ON UNIT 6.
! LPARAM  = PARAMETERS ARE TO BE DETERMINED IN FIRST COLLOCATION
!           STEP.
! LNCOL   = COLLOCATION STEPS WILL NOT BE EXECUTED (I.E. NO INPUT OF
!           OF COVARIANCE FUNCTION PARAMETERS).
! LIOSOL = WRITE INPUT PARAMETERS, OBSERVATIONS AND SOLUTION ON
!          UNIT 17, SO THAT THE SOLUTION MAY BE RETRIEVED, OR WRITE
!          OR READ COVARIANCE FUNCTION AND SOLUTION ON BINARY FORM.
!
! IN ORDER TO BE ABLE TO USE UNIT 17, WE WILL SUPPOSE THAT IT IS A
! NON-PRINTER LIKE DEVICE, WHICH DO NOT USE THE FIRST CHARACTER IN
! AN OUTPUT RECORD AS A CONTROL CHARACTER.
!
! IF LIOSOL IS TRUE, INPUT OF THE FOLLOWING LOGICAL VARIABLES:
! LWRSOL = WRITE SOLUTION ON UNIT 17, (RESTART FILE).
! LBIPOT = WRITE POTENTIAL COEFFICIENTS ON UNIT 3.
! LBICOV = WRITE COVARIANCE FUNCTION PARAMETERS ON UNIT 18, BINARY,
! LBISOL = WRITE SOLUTION PARAMETERS ON UNIT 19, BINARY,
! LINSOL = READ CATALOGUE OF BINARY FILES OF COVARIANCE FUNCTION
!          AND SOLUTION PARAMETERS WITH AREA OF VALIDITY BOUNDARIES,
!          SEE SUBROUTINE INSOL.
!
! FOLLOWING THE INPUT OF THE LOGICAL VARIABLES ARE THE NAMES OF THE
! FILES USED TO STORE VARIOUS TYPES OF DATA:
! LWRSOL= TRUE: NAME OF RESTART FILE,                 (UNIT 17).
! LBIPOT= TRUE: NAME OF FILE TO HOLD POTENTIAL COEFFICIENTS (UNIT 3).
! LBICOV= TRUE: NAME OF FILE TO HOLD COVARIANCE FUNCTION PARAMETERS.
! LBISOL= TRUE: NAME OF FILE TO HOLD SOLUTION PARAMETERS.
! LINSOL= TRUE: NAME OF FILE HOLDING CATALOGUE OF SOLUTIONS.
! LNCOL TRUE, OR LINSOL IS TRUE, BUT SOLUTIONS ARE NOT YET COMPUTED,
!    NAME OF FILE HOLDING NORMAL EQUATIONS (UNIT 8).
!
! --------------- INPUT (1A) ---------------------------------
!
! 1100 IF (LINTER) WRITE(6,1101)
 1100 IF (LINTER.AND.MPI_pid==0) WRITE(6,1101)
 1101 FORMAT( &
 ' INPUT: LSPHER, TRUE IF SPHERICAL APPROXIMATION IS USED.',/&
 '        LTRAN, TRUE IF NON-STANDARD REF. SYSTEM IS USED',/&
 '        LPOT,  TRUE IF SPHERICAL HARMONIC EXPANSION IS USED',/&
 '        LTEST, TRUE IF TEST-OUTPUT IS NEEDED',/&
 '        LLEG,  TRUE IF LEGEND IS TO BE OUTPUT',/&
 '        LPARAM,TRUE IF PARAMETERS ARE TO BE DETERMINED',/&
 '        LNCOL, TRUE IF COLLOCATION IS NOT USED',/&
 '        LIOSOL,TRUE IF SOLUTION IS STORED OR RECOVERED')
 READ(5,*)LSPHER,LTRAN,LPOT,LTEST,LLEG,LPARAM,LNCOL,LIOSOL
 105 FORMAT(8L2)
 LCOLLO=.NOT.LNCOL
 LWRSOL=LF
 IF (LSPHER) THEN
  WRITE(*,*)' SPHERICAL APPROXIMATION IN USE. '
 ELSE
  WRITE(*,*)' SPHERICAL APPROXIMATION NOT  IN USE. '
 END IF
!
! --------------- INPUT (1B) ---------------------------------
!
 IF (LIOSOL.AND.LINTER)WRITE(6,1102)
 1102 FORMAT(                                                 &
 ' INPUT: LWRSOL, TRUE IF SOLUTION IS OUTPUT ON UNIT 17     ',/&
 '        LBIPOT, LBICOV,LBISOL, TRUE IF POTENTIAL COEFF.   ',/&
 '        COVARIANCE FCT. TABLE OR SOLUTION IS OUTPUT BINARY',/&
 '        LIOSOL, TRUE IF BINARY SOLUTION IS USED'              )
 IF (LIOSOL) READ(5,*)LWRSOL,LBIPOT,LBICOV,LBISOL,LINSOL
 ICHAR=1
! ICHAR IS EQUAL TO THE MAXIMAL NUMBER OF ELEMENTS USED IN DNAME OR
```

```
! DNANE.
!
! --------------- INPUT (1C) ---------------------------------
!
! INPUT OF FILE NAMES:
 IF (LWRSOL) THEN
  IF (LINTER)WRITE(6,*)' INPUT NAME OF FILE TO HOLD SOLUTION'
  READ(5,2103)DNAME(1)
 2103  FORMAT(A128)
  WRITE(6,162)(DNAME(I),I=1,ICHAR)
  162  FORMAT(' NAME OF RESTART FILE=',2A128)
  OPEN(17,FILE=DNAME(1),STATUS='UNKNOWN',FORM='FORMATTED')
 END IF
!
 IF (LBIPOT) THEN
!
! INPUT OF NAME OF FILE TO HOLD POTENTIAL COEFFICIENTS ON BINARY FORM (UNIT 3).
  IF (LINTER)WRITE(6,*)' INPUT NAME OF FILE TO HOLD COEFF.'
  READ(5,2103)PNA0(1)
 END IF
!
 IF (LBICOV) THEN
! INPUT OF NAME OF FILE TO HOLD COVARIANCE FUNCTION PARAMETERS ON BINARY FORM.
  IF (LINTER) WRITE(6,*)' INPUT NAME OF FILE TO HOLD COVFCT'
  READ(5,2103)CNANE(1)
 END IF
!
 IF (LBISOL.OR.LINSOL)  THEN
! INPUT OF NAME OF FILE TO HOLD SOLUTION PARAMETERS ON BINARY FORM.
  IF (LINTER) WRITE(6,*)' INPUT NAME OF FILE WITH BINARY SOL.'
  READ(5,2103)SNAME(1)
!
!      IF (LINSOL) ICSYS0=INITSO(NBLO,NBLP,SNAME,PNAME,BOUNDS)
 END IF
!
 IF (.NOT.LCOLLO) THEN
#ifdef _OPENMP
  NPROC = OMP_GET_NUM_PROCS()
#endif
!
! FOR MULTIPROCESSING WE MUST INPUT NUMBER OF PROCESSES TO BE USED.
  WRITE(*,*) ' Number of available threads ',NPROC
  IF (LINTER) WRITE(*,*) " Insert number of threads "
  READ(5,*) NPROC
  WRITE(*,*)' Threads used ', NPROC
#ifdef _OPENMP
  CALL omp_set_num_threads(NPROC)
#endif
  LNEQ = LF
  LE=LF
 ELSE
! CHANGE 2004-11-08.
  LNEQ=LT
  LE=LT
  MAXFIL=0
  MAXBNE=2.0D0**23/NCRW+1
  WRITE(*,*)MAXBNE,' BLOCKS IN EACH FILE NEEDED '
!
! --------------- INPUT (1D) ---------------------------------
!
  NEQFMA = 1
! ADDED 2004-06-21.
#ifdef _OPENMP
  NPROC = OMP_GET_NUM_PROCS()
#endif
!
! FOR MULTIPROCESSING WE MUST INPUT NUMBER OF PROCESSES TO BE USED.
! SEE REF(J).
  PRINT *,''
```



```
  WRITE(*,*) ' Number of available threads ',NPROC
  IF (LINTER) WRITE(*,*) " Insert number of threads "
  READ(5,*) NPROC
  WRITE(*,*)' Threads used ', NPROC
#ifdef _OPENMP
  CALL omp_set_num_threads(NPROC)
  if (ltest) then
  PRINT *,''
!$OMP PARALLEL
  PRINT *,'   I am thread number ',omp_get_thread_num()
!$OMP BARRIER
!$OMP END PARALLEL
  end if

  PRINT *,''
  PRINT *,'Number of threads in this region: ',omp_get_num_threads()
  PRINT *,''
#endif

  IF (LINTER) WRITE(*,*)' INPUT NUMBER OF COLUMNS PER BLOCK AND BLOCKS PER CHUNK
 '
  READ(5,*)NEQFI(1,2),NBB
  IBSS=NEQFI(1,2)
  NN = IBSS
! MAXO CHANGED SO IT CORRESPONDS TO DATA USED IN FORMING ONE CHUNK SOLUMN. 2012-
05-22.
! MAXO=NN*IBSS
  WRITE(*,*)' NUMBER OF COLUMNS PER (FILLED) BLOCK AND BLOCKS PER CHUNK ',IBSS,N
BB
! SUB-BLOCK SIZE IS THE SAME FOR ALL BLOCKS.
! FILESIZE IN COLUMNS PR. BLOCK
  MI1     = 0
  NBL(0)  = 0
  MAXOBS  = (2*NCRW+IBSS**2+IBSS)/(2*IBSS+1)
  WRITE(*,*)' MAXIMAL NUMBER OF OBSERVATIONS WHEN USING COLLOCATION = ',MAXOBS
  I = MAXOBS/IBSS+1
  IF (I.GT.NNBL) WRITE(*,*) ' WARNING1 *** TOO FEW BLOCKS PERMITTED ',NNBL,I
!
! INPUT OF NAME OF FILE HOLDING NORMAL EQUATIONS.
  IF (LINTER)WRITE(6,*)' INPUT ROOT-NAME OF FILES WITH NORMAL EQ.'
  READ(5,2103)DNANE(1,1)
  file_root_name = DNANE(1,1)
  WRITE(6,163)(DNANE(J,1),J=1,ICHAR)
 163    FORMAT(' ROOT-NAME OF FILE HOLDING NORMAL EQUATIONS=',2A128)
  MMAXB = INT(40000000000.0/(8*(IBSS**2)))
  FMIOLD=1
! WRITE(*,*) ' THIS MAKES ROOM FOR 100000 OBS WITH BLOCK SIZE 400*400. '
! WRITE(*,*)' MAXIMAL NUMBER OF BLOCKS PER FILE = ',MMAXB
 END IF

 IF(LWRSOL) THEN
  IF (LUNIX) WRITE(17,805)LSPHER,LTRAN,LPOT,LPARAM
  805  FORMAT('F',/,3L2,' F F',L2,' F F')
  WRITE(17,*)NEQFMA
  IF (.NOT.LNCOL) THEN
   WRITE(17,2103)(DNANE(J,1),J=1,ICHAR)
  END IF
 END IF
!
! ------------------------- INPUT (1E) ----------------------------
!
! IF LTEST IS TRUE INPUT OF THE FOLLOWING LOGICAL VARIABLES:
! LONEQ   = OUTPUT OF COEFFICIENTS OF NORMAL EQUATIONS TO UNIT 6.
! LTIME   = OUTPUT OF USED CPU TIME (UNIX).
! LTCOV   = OUTPUT OF TEST-DATA FROM COVARIANCE FUNCTION ROUTINES.
! LCZERO  = TRUE, IF MULTIPROCESSING SHOUL NOT BE USED IN PRED.
! LCOERR  = TRUE IF ERRORS ARE CORRELATED.
! LFULLO  = TRUE IF ALL COMPONENTS OF GRAVITY VECTOR OR GRAVITY GRADIENT
!           PLUS ROTATION PARAMETERS ARE OUTPUT ON CURRENT OUTPUT.
```

```
! LMTEST  = TESTOUTPUT FROM cholsol.
 LCOERR=LF
 IF (LINTER.AND.LTEST) WRITE(6,1103)
 1103 FORMAT(                                                  &
 ' INPUT: LONEQ, TRUE IF COEFFICIENTS ARE OUTPUT,              '/&
 '        LTIME: TRUE, IF TIMING IS MADE (ONLY UNIX)           '/&
 '        LTCOV: TRUE, IF OUTPUT FROM COV. CALCULATION         '/&
 '        LCZERO: TRUE, IF NO MULTIPROCESSING IN PRED          '/&
 '        LCOERR: TRUE, IF DATA ERRORS ARE CORRELATED.         '/&
 '        LFULLO: TRUE, IF V, ALL COM. OF DG OR DDG ARE OUTPUT '/&
 '        LMTEST: TRUE IF TESTOUTPUT FROM cholsol              '  )
 IF (LTEST) READ(5,*)LONEQ,LTIME,LTCOV,LCZERO,LCOERR,LFULLO,LMTEST
      if (LCZERO) WRITE(*,*)' NO MULTIPROCESSING IN PRED '
      ltest2=LMTEST
! ADDED 2012-06-17 FOR TRANSFER to m_cholsol.f90.
  IF (LMTEST) THEN
! ADDED 2011-02-11.
   OPEN(10,FILE='mtest.txt')
   WRITE(*,*)' UNIT 10 OPENED FOR TEST-OUTPUT IN mtest.txt '
  END IF

 LTIME=LTIME.AND.LUNIX
 IF (LINTER) THEN
  WRITE(6,*)' ARE ALL PARAMETERS OK ?'
  READ(5,*)LOK
  IF (.NOT.LOK) THEN
   WRITE(*,*)' REPEAT INPUT PARAMETERS '
   GO TO 1100
  END IF
 END IF
!
 LFORM=LF
 LNFORM = LT
 LSMAL=LF
 LNERNO = LF
 LPRED = LNCOL
 LNPOT = .NOT.LPOT
!
 IF (LLEG) WRITE(6,114)
  114 FORMAT(/' LEGEND OF TABLES OF OBSERVATIONS AND PREDICTIONS:',/,&
 ' OBS = OBSERVED VALUE (WHEN AN OBSERVATION IS A VECTOR ',/,&
 '       QUANTITY, THEN ONE BELOW THE OTHER)',/,&
 ' DIF = DIFFERENCE BETWEEN OBSERVED AND PREDICTED VALUE',/,&
 '       WHEN PREDICTION ARE COMPUTED AND ELSE THE RESIDUAL',/,&
 '       OBSERVATION.',/,&
 ' ERR = STANDARD DEVIATION OF OBSERVATION OR ESTIMATE OF',/,&
 '       PREDICTION ERROR.',/,&
 ' TRA = CONTRIBUTION FROM DATUM TRANSFORMATION.'/,&
 ' TERR= CONTRIBUTION FROM TERRAIN',/,&
 ' POT = CONTRIBUTION FROM POTENTIAL COEFFICIENTS.',/,&
 ' COLL= CONTRIBUTION FROM COLLOCATION DETERMINED PART.',/,&
 ' COLL1=CONTRIBUTION FROM FIRST SET OF OBSERVATIONS.',/,&
 ' COLL2=CONTRIBUTION FROM SECOND SET OF OBSERVATIONS.',/,&
 ' PRED= PREDICTED VALUE IN GEOCENTRIC SYSTEM.',/,&
 ' PRED-TRA= PREDICTED VALUE IN SELECTED COORDINATE SYSTEM.')
!
! *************** INPUT (2) ********************************
!
! INPUT OF DATA FOR GEOCENTRIC REFERENCE SYSTEM: (A) FIRST INPUT
! OF CODE FOR SYSTEM DEFINITION, ICSYS, EQUAL TO 0 IF PARAMETERS
! ARE INPUT FROM UNIT 5 AND EQUAL TO 4 FOR GRS1967, 5 FOR GRS1980,
! 7 FOR BEST CURRENT SYSTEM, AND MORE, SEE BELOW.
!
! ---------------- INPUT (2A) --------------------------------
 IF ((.NOT.LINSOL).AND.LINTER)WRITE(6,1105)
 1105 FORMAT(                                                  &
 ' INPUT CODE FOR BASIC REFERENCE SYSTEM:                     '/&
 ' 0: USER DEFINED, 1: ED50 NORTH SEA, 2: ED50/EDOC,          '/&
 ' 3: NAD1927 /NEW MEXICO, 4: GRS67, 5:  GRS80, 6: NWL9D,     '/&
```



```
 ' 7: EGM08, TIDEFREE, 8: BEST CUR. FAROE ISL, 9: ED50 FOR SF,     '/&
 ' 10: IAG-75, 11: KRASSOWSKY, DDR, 12: GERMAN DHDN, BESS.      '/&
 ' 13: ENGLAND/WALES SHIFT, 14: REP. IRELAND SHIFT OF GPS/LEV . '/&
 ' 15: GRIM, 16: TOPEX, 17: WGS84, 18: WGS84 rev. 1.            '  )
! CHANGED 2011-06-08.
 IF (.NOT.LINSOL) READ(5,*)ICSYS0
 102  FORMAT(I5)
 IF (LWRSOL) WRITE(17,102)ICSYS0
!
 WRITE(6,106)
  106 FORMAT(/' REFERENCE SYSTEM:')
!
 IF (ICSYS0.EQ.0) THEN
!
! ---------------- INPUT (2B) --------------------------------
!  INPUT OF TEXT DESCRIBING REFERENCE SYSTEM (MAX.128 CHARACTERS).
  IF (LINTER)WRITE(6,*)' INPUT NAME OF USER DEFINED SYSTEM'
  READ(5,103)FMT(1)
  103  FORMAT(A128)
! IN FORTRAN77 WE SOPPOSE FMT TO BE OF CHARACTER TYPE.
  IF (LWRSOL) WRITE(17,103)FMT(1)
  803  FORMAT(1X,A128)
  WRITE(6,803)FMT(1)
!
! ------------------ INPUT (2C) ----------------------------------
! IF ICSYS=0 INPUT OF PARAMETER MODEC0 SPECIFYING IN WHICH WAY THE
! THE NORMAL POTENTIAL IS DEFINED, AND THEN THE 3 PARAMETERS
! DEFINING THE SYSTEM:
! MODEC0     EE0(1)     EE0(2)    EE0(3)
!   1         GM         AX        J2 (=-C(2,0))
!   2         GM         AX        E2
!   3         GM         AX        1/F
!   4,5     GAMMA        AX        1/F.
! WHERE GM IS THE PRODUCT OF THE MASS AND THE GRAVITY CONSTANT IN
! M**3/S**2, GAMMA THE EQUATORIAL NORMAL GRAVITY IN M/S**2, AX
! THE SEMI-MAJOR AXIS OF THE REFERENCE ELLIPSOID IN M, J2 THE
! THE (NOT NORMALIZED) 2.ORDER ZONAL HARMONIC, E2 THE SECOND
! EXCENTRICITY (AX**2-BX**2)/AX**2), AND F THE FLATTENING (AX-BX)/AX,
! WHERE BX IS THE SEMI-MINOR AXIS. MODEC0=5 GIVES THE INTERNATIONAL
! ELLIPSOID (1928) AND THE CASSINIS GRAVITY FORMULA, AND SUPPOSES A
! POTSDAM CORRECTION OF 13.7 MGAL MUST BE APPLIED TO MEASURED
! GRAVITY VALUES (LPOTSD=.TRUE.).
  IF (LINTER)WRITE(6,1106)
 1106  FORMAT(' INPUT MODECO (1,..,5) AND',/&
  ' FOR 1: GM, SEMI-MAJOR AXIS (M), AND J2',/&
  '     2:  -      -              ,  -  EXCENTRICITY**2',/&
  '     3:  -      -              ,  -  1/FLATTENING',/&
  '     4,5 EQUATORIAL GRAVITY, SEMI-MAJOR AX, 1/FLATTENING',/&
  '     WHERE 5: POTSDAM SYSTEM FOR GRAVITY')
  120  FORMAT(I2,3E16.9)
  READ(5,*)MODEC0,EE0(1),EE0(2),EE0(3)
  IF (LWRSOL) WRITE(17,120)MODEC0,EE0(1),EE0(2),EE0(3)
 END IF
!
 CALL ICOSYS(ICSYS0,15,GM2,AX2,E22,F2,UREF0,GREF)
!
 GG=GREF*1.0D5
 WRITE(6,122)AX2,F2,GM2,GG,UREF0
  122 FORMAT(/' A   =',F11.2,' M'/,&
 ' 1/F =',F12.7/,&
 ' GM=',E17.10,/,&
 ' REF.GRAVITY AT EQUATOR =',F14.4,' MGAL'/,&
 ' POTENTIAL AT REF.ELL.  =',F14.4,' M**2/SEC**2'/)
 IF (LTEST.AND.LNCOL) WRITE(6,9122)(FG(I),I=16,30),(FJ(J),J=16,30)
 9122 FORMAT(' CONTENTS OF FG, FJ',/10(3E17.10/))
!
 IF (LTRAN) THEN
!
! *************** INPUT (3) ********************************
```

```
!
! INPUT OF MODEC0, AND 3 PARAMETERS DEFINING THE NORMAL POTENTIAL
! AS FOR THE GEOCENTRIC DATUM, FOLLOWED BY 7 DATUM SHIFT PARAMETERS,
! DX,DY,DZ,EPS3,EPS2,EPS1,DL AND THE VALUE OF A LOGICAL
! VARIABLE LCHANG, WHICH IS TRUE, WHEN AN ADDITIONAL DATUM SHIFT IS
! GIVEN AS A CHANGE IN THE DEFLECTIONS OF THE VERTICAL AND THE HEIGHT
! ANOMALY IN A POINT WITH COORDINATES (LAT0, LONG0). LCHANG MUST BE TRUE
! IN CASE DATUM-SHIFT PARAMETERS OF THIS KIND ARE TO BE ESTIMATED
! BY THE PROGRAM (IE. WHEN LPARAM IS TRUE).
! THE COORDINATES MUST BE INPUT IN DEGREES, MINUTES AND SECONDS, FOL-
! LOWED BY THE TRANSFORMATION ELEMENTS IN KSI, ETA AND ZETA (DKSI0,
! DETA0,DZETA0) IN ARCSEC AND METERS.
!
  IF (LINTER)WRITE(6,1106)
  IF (LINTER)WRITE(6,1107)
 1107  FORMAT(                                          &
  ' INPUT 7 DATUM SHIFT PARAMETERS: DX,DY,DZ,E3,E2,E1,AND  ',/&
  ' LCHANG, TRUE IF CHANGE IN KSI,ETA,ZETA AT ORIGIN ALSO  '   )
  131  FORMAT(I2,3E16.9/3F8.2,3F6.2,E10.2,L2)
  READ(5,*)MODEC0,EE0(1),EE0(2),EE0(3)
  READ(5,*)(DSHIF0(I),I=1,7),LCHANG
  IF (LWRSOL) WRITE(17,131)MODEC0,EE0(1),EE0(2),EE0(3),&
  (DSHIF0(I),I=1,7),LCHANG
  WRITE(6,136)
  136  FORMAT(' PARAMETERS FOR')
  CALL ICOSYS(0,0,GM1,AX1,E21,F1,UREF,GREF)
  WRITE(6,132)AX1,GM1,F1,DSHIF0(7),DX,DY,DZ,DSHIF0(4),&
  DSHIF0(5),DSHIF0(6)
  132  FORMAT(/'  NEW A      NEW GM         NEW 1/F'/,&
  F10.1,E15.7,F10.5,//&
  '    DL       DX     DY     DZ  ',/,E10.2,3F7.1,//,&
  '  EPS3  EPS2  EPS1',/,3F6.2)
  GG=GREF*1.0D5
  WRITE(6,135)GG,UREF
  135  FORMAT(/'  NEW REF. GRAVITY AT EQUATOR=',F12.2,' MGAL',/&
  '  NEW POTENTIAL AT ELLIPSOID =',F11.1,' M**2/SEC**2',/)
  IF (LCHANG) THEN
!
   IF (LINTER) WRITE(6,1108)
 1108   FORMAT(' INPUT LAT.,LON. OF ORIGIN (DD MM SS.S), AND 3 SHIFTS')
  133   FORMAT(2I3,F6.2,2I3,F6.2,3F6.2)
   READ(5,*)IDLAT,MLAT,SLAT,IDLON,MLON,SLON,DKSI0,DETA0,DZETA0
   IF (LWRSOL) WRITE(17,133)IDLAT,MLAT,SLAT,IDLON,MLON,SLON,&
   DKSI0,DETA0,DZETA0
   WRITE(6,134)IDLAT,MLAT,SLAT,IDLON,MLON,SLON,DKSI0,DETA0,DZETA0
  134   FORMAT(' ADDITIONAL DATUM-SHIFT COMPONENTS OR INITIAL PARAME',&
   'TERS FOR DATUM SHIFT',/,' DETERMINATION, GIVEN IN:',/,&
   '   LATITUDE      LONGITUDE BY DKSI  DETA  DZETA',/,&
   I4,I3,F6.2,I4,I3,F6.2,3F7.2)
   CALL RAD(IDLAT,MLAT,SLAT,RLAT0,1)
   CALL RAD(IDLON,MLON,SLON,RLONG0,1)
   SINLA0 =  SIN(RLAT0)
   COSLA0 =  COS(RLAT0)
   COSDLO =  COS(RLONG0)
   SINDLO =  SIN(RLONG0)
   W2 = D1-E22*SINLA0**2
   W  =  SQRT(W2)
   RN = AX2/W
   RM = AX2*(D1-E22)/(W*W2)
   X = -SINLA0*COSDLO*DKSI0*RM/RADSEC-SINDLO*DETA0*RN/RADSEC-COSLA0*COSDLO*DZETA
0
   Y = -SINLA0*SINDLO*DKSI0*RM/RADSEC+COSDLO*DETA0*RN/RADSEC-COSLA0*SINDLO*DZETA
0
   Z =  COSLA0*RM/RADSEC*DKSI0-SINLA0*DZETA0
   X1 = X*X+Y*Y+Z*Z
   IF (X1.GT.0.1D-2) WRITE(6,700)X,Y,Z
  700   FORMAT(' THE CHANGE OF DEFLECTIONS AND HEIGHT ANOMALY CORRES',&
   'POND TO A',/,' TRANSLATION VECTOR: (DX,DY,DZ) =(',F7.2,',',&
   F7.2,',',F7.2,'), (METERS).',/)
```



```
   DSHIF0(1)=DX-X
   DSHIF0(2)=DY-Y
   DSHIF0(3)=DZ-Z
  END IF
!
 ELSE
  E21 = E22
  AX1 = AX2
 END IF
!
 IF (LPOT) THEN
  IF (.NOT.LINSOL) THEN
!
! *************** INPUT (4) *********************************
!
! INPUT IS DONE IN STEPS (A) - (D):
! --------------- INPUT (4A) --------------------------------
! INPUT OF TEXT DESCRIBING SOURCE OF THE POTENTIAL COEFFICIENTS (MAX.
! 128 CHARACTERS).
   IF (LINTER)WRITE(6,*)' INPUT NAME OF POT.COEFF. SET'
   READ(5,103)FMT(1)
   WRITE(6,130)
  130   FORMAT(/' SOURCE OF THE POTENTIAL COEFFICIENTS USED:')
   WRITE(6,803)FMT(1)
   IF (LWRSOL) WRITE(17,103)FMT(1)
!
! --------------- INPUT (4B) --------------------------------
!  GM   IN (METERS**3/SEC**2), ASSOCIATED WITH THE COEFFICIENT SET,
!  A    SEMI-MAJOR AXIS (M),      -       -    -       -       - ,
!  COFF(5) THE COEFFICIENT C(2,0) MULTIPLIED BY 1.0D6, IF LFM IS
!       TRUE, SEE BELOW. (THIS IS BECAUSE C(2,0) DOES NOT FIT INTO
!       A STANDARD INPUT FORMAT FOR THE OTHER COEFFICIENTS). IF
!       LFM IS FALSE, A DUMMY VARIABLE MUST BE INPUT.
!  NMAX THE MAXIMAL DEGREE,
! 5 LOGICAL VARIABLES:
! LFM   TRUE IF THE COEFFICIENTS ARE INPUT WITH A FIXED NUMBER ON
!       ON EACH RECORD, IN THE SEQUENCE C(2,1),S(2,1),C(2,2) ETC.
!       THE COEFFICIENTS MUST BE FULLY NORMALIZED, AND MULTIPLIED
!       BY 1.0D6. IF LFM IS FALSE, INPUT OF COEFFICIENTS FROM UNIT
!       9, SEE THE SUBROUTINE LOADCS.
! LBIN  TRUE, IF THE COEFFICIENTS ARE INPUT FROM UNIT 9 ON BINARY
!       FORM.
! LFORM TRUE, IF A RUN-TIME FORMAT FOR THE COEFFICIENTS ARE USED,
!       IN WHICH CASE THE FORMAT IS INPUT BELOW (4C).
! LINT  TRUE, IF THE COEFFICIENTS ARE STORED AS INTEGER VARIABLES
!       IN THE ARRAY IICC. (LFM MUST BE FALSE). DELETED 2012-02-08.
! LSKIPL, TRUE IF DUMMY LINES IN FRONT OF FILE
!
   IF (LINTER) WRITE(6,1109)

 1109   FORMAT(                                          &
   ' INPUT: GM, SEMI-MAJOR AXIS (M), C(2,0), MAX. DEGREE '/&
   '  LFM, TRUE IF COEFF. IN INPUT STREEM AND *1.0D6     '/&
   '  LBIN, TRUE IF ON BINARY FORM                       '/&
   '  LFORM, TRUE IF FORMAT IS INPUT                     '/&
   '  LINT, TRUE IF STORED AS INTEGERS (NOT PERMITTED).                  '/&
   '  LSKIPL, TRUE IF DUMMY LINES IN FRONT OF FILE       '  )

   READ(5,*)GMP,AX,COFF(5),NMAX,LFM,LBIN,LFORM,LINT,LSKIPL
   LNFORM=.NOT.LFORM
! IF LFM IS FALSE, COFF(5) MAY BE OVERRIDDEN BY SETCS.
  137   FORMAT(E15.8,F11.1,F10.4,I4,5L2)
   IF (LWRSOL) WRITE(17,137)GMP,AX,COFF(5),NMAX,LFM,LBIN,LFORM,LINT,LSKIPL
   IF (LINT)  THEN
    WRITE(6,702)
  702   FORMAT(' COEFFICIENTS CANT  BE STORED AS INTEGERS.')
    STOP
   END IF
   WRITE(6,138)GMP,AX,COFF(5),NMAX
```

```
  138   FORMAT(/'     GM             A       COFF(5)  MAX.DEGREE',/&
   E15.8,F11.1,F10.4,I5)
   IF (NMAX.GT.2200) THEN
    WRITE(6,140)
  140    FORMAT(' NMAX TOO BIG.')
    STOP
   END IF
!
   N2 = (NMAX+1)**2
   L386=N2.GT.NCOEFF
!
! --------------- INPUT (4C) --------------------------------
! HERE INPUT OF FORMAT OF COEFF.
   IF (LINTER.AND.LFORM) WRITE(6,*) ' INPUT FORMAT (2I4,2D18.0) F.EX.'
   IF (LFORM) READ(5,103)FMT(1)
   IF (LFORM.AND.LWRSOL) WRITE(17,103)FMT(1)
   COFF(1)=1.0D6
   IF (.NOT.LFM) COFF(1)=D1
! IF COEFFICIENTS ARE INPUT FROM UNIT 5 THEY ARE SUPPOSED TO BE
! MULTIPLIED BY 1.0D6. IF THEY ARE INPUT FROM UNIT 9, A VALUE OF
! COFF(1) DIFFERENT FROM 1.0D0 IS SUPPOSED TO BE THE SCALE FACTOR.
   IF (.NOT.LFM) THEN
!
! --------------- INPUT (4D) --------------------------------
! INPUT OF NAME OF FILE HOLDING COEFFICIENTS.
    IF (LINTER) WRITE(6,*)' INPUT NAME OF FILE HOLDING COEFF.'
    READ(5,2103)PNAME(1)
    WRITE(6,161)(PNAME(I),I=1,ICHAR)
  161    FORMAT(' NAME OF FILE HOLDING COEFFICIENTS: ',2A128)
    IF (LWRSOL) WRITE(17,2103)(PNAME(J),J=1,ICHAR)
    CALL LOADCS(PNAME,FMT,NMAX,LFORM,LBIN,LSKIPL)
! IF LSKIPL IS TRUE, LOADCS WILL INPUT NUMBER OF LINES TO BE SKIPPED.
! REMARK 2010-11-15 BY CCT.
   ELSE
!
! *************** INPUT (5) **********************************
!
! INPUT OF POTENTIAL COEFFICIENTS STARTING WITH C(2,1). NOTE THAT
! PROBLEM MAY OCCUR IN FREE-FORMAT INPUT, IF DATA IS LINE NUMBERED.
! IN THIS CASE CHANGE TO FIXED FORMAT INPUT.
    IF (LINTER) WRITE(6,*)' INPUT COEFF. FROM C(2,1)'
    IF (LNFORM.AND.(.NOT.LFOR77)) READ(5,99)(COFF(I), I = 6, N2)
    IF (LNFORM.AND.LFOR77) READ(5,*)(COFF(I),I=6,N2)
   99    FORMAT(9F8.4)
    IF (LFORM) READ(5,FMT)(COFF(I), I = 6, N2)
    IF (LWRSOL) WRITE(17,99)(COFF(I),I = 6,N2)
   END IF
   IF (.NOT.L386) THEN
    DO I = 1, 3
     COFF(I+1) = D0
! THIS ASSURES THAT C(1,0),C(1,1) AND S(1,1) ARE ALL ZERO.
    END DO
   END IF
  ELSE
!
   CALL LOADCS(PNAME,FMT,NMAX,LF,LT,LSKIPL)
  END IF
  N2 = (NMAX+1)**2
  CALL SETCM(NMAX,LBIN)
! SETCM QUASI-NORMALIZES THE COEFFICIENTS AND SETS TABLE ROOT.
  IF ((.NOT.LINT).AND.(.NOT.L386)) COFF(1)=D1
  CM3=GMP
  CMM2=AX
  CM1=OMEGA2
!
  IF (LBIPOT) THEN
   OPEN(3,FILE=PNA0(1),STATUS='UNKNOWN',FORM='UNFORMATTED')
!    *  RECL=4) CHANGE 1998.07.04 BY CCT.
! MUST BE 8 IF REAL*8 IS USED:
```



```
!  IF (LINT) THEN
    WRITE(*,*)' COEFFICIENTS TO BE STORED ON UNIT 3'
! CHANGE 1998.=/.06 BY CCT. (READING OF 8 FIRST SKIPPED).
!        DO 9002 I=1,N2
!9002    WRITE(3)COFF(I)
! WARNING: OUTPUT SHOULD ONLY BE TO N2. BUT IF COFF IS USED TO
! STORE COEFFICIENTS ON INTEGER FORM, THEN THIS WILL NOT WORK.
! REMARK 2004-12-09.
    WRITE(3)COFF
!  END IF
   WRITE(6,174)PNA0
  174   FORMAT(' POTENTIAL COEFFICIENTS OUTPUT TO FILE ',2A128)
   CLOSE(3)
  END IF
!
  IF (LTEST.AND.LPOT.AND.LNCOL) THEN
! COMPUTATION OF DEGREE-VARIANCES ADDED 1989.02.27 BY CCT.
   II=1
   DGVAR=D0
   DO I=2,NMAX+1
    SUS=D0
    SS=D0
    DO J=1,2*I-1
     II=II+1
     SSC=COFF(II)
     IF (J.EQ.1.AND.I.LT.12) SSC=SSC-FJ(I+15)
     IF (J.NE.1) THEN
      SUS=SUS+SSC/SQ2
      SS=SS+SSC**2/D2
     ELSE
      SUS=SUS+SSC
      SS=SS+SSC**2
     END IF
    END DO
!
    SIGMA0(I)=(GMC/RE)**2*SS*(I-2)**2/(RE**2*(2*I-1))*1.0D10
    SIGD=((SS-SUS**2/(2*I-1))/(2*I-2))
    WRITE(*,1197) I-1,SUS/(2*I-1),SQRT((SS-SUS**2/(2*I-1))/(2*I-2)),SQRT(SS/(2*I
-1)),SIGD,SS/(2*I-1)
 1197    FORMAT(I5,6D11.3)
    DGVAR=DGVAR+SIGMA0(I)
   END DO
   WRITE(6,728)NMAX,DGVAR
  728   FORMAT(' DEGREE-VARIANCES FROM DEG. 2 TO ',I5,  ' IN MGAL**2'/&
              ' WITH GRAVITY VARIANCE SUM =     ',F9.2,' MGAL**2   ')
    WRITE(6,*)(SIGMA0(I),I=3,NMAX+1)
  END IF
  IF (LNCOL) WRITE(6,173)
  173  FORMAT(2X)
 END IF
!
! ******************************************************************
! COLLOCATION SECTION: INITIALIZATION OF VARIABLES.
!
 N = 0
 ICREL=0
 ISATP=0
 NREL=0
 NERCOV=0
 NOBLK=0
 LOBSST=LF
 LSTART=LT
 LSMAL=LF
!
 IF (LCOLLO) THEN
  WRITE(6,109)
  109  FORMAT(/' START OF COLLOCATION I:')
!
  LNERNO=LF
```

```
  LERNO=LT
 1000  CONTINUE
!
! ************************ INPUT (6) AND (7) **************************
! INPUT OF PARAMETERS DEFINING COVARIANCE FUNCTION.
  IF (.NOT.LINSOL) THEN
   CALL INCOV(LINTER,RB)
  END IF
  SSOBS = D0
!
  MAXPAR=MXPAR
  LINSOL=LF
  LNDAT=.NOT.LPARAM
  IF (LTIME) THEN
   CPU1=SYTIME(RCBASE,TIMEARRAY)
   WRITE(6,7470)TIMEARRAY(1),CPU1
  END IF
 7470  FORMAT(/' TIME USED=',F15.5,' SEC, ELAPSED TIME =',F15.5,' SEC')
!
  IF (.NOT.LPARAM) THEN
! POSSIBLE ERROR: 2003-04-07. VALUE MAY INFLUENCE SUBROUTINE NES.
   NPARM1=0
  ELSE
!
! *************** INPUT (8) **********************************
!
! INPUT OF VARIABLES SPECIFYING DETAILS CONCERNING PARAMETERS.
! LALLP   - TRUE IF ALL PARAMETERS ARE DEFINED AT THE BEGINNING
!           OF A COLLOCATION STEP AND FALSE, IF THEY ARE SPECI-
!           FIED IMPLICITLY THROUGH THE DATA. (FOR SATELLITE ALTI-
!           METRY THROUGT THE REVOLUTION NUMBER, FOR EXAMPLE).
! IF LALLP IS TRUE, THEN NUMBER OF PARAMETERS, AND PARAMETER
! IDENTIFICATION CODES MUST BE INPUT SUBSEQUENTLY. SEE THE SUB-
! ROUTINE WRPAR FOR A SURVEY OF CURRENTLY USED CODES.
   IF (LINTER) WRITE(6,*)' INPUT LALLP, TRUE IF PARAMETERS GLOBAL'
   READ(5,*)LALLP
   NPARM=0
   NPAOLD=-1
   NPOBS=0
   IF (LWRSOL) WRITE(17,105)LALLP
   IPA = 0
   IPAMAX=0
! CHANGE 2002-04-14.
! MAXPAR IS THE MAXIMAL NUMBER OF PARAMETERS CURRENTLY PERMITTED.
!
! WE NEED UNIT 99 TO HOLD PARAMETER ERRORS:
   WRITE(*,*) ' FILE-NAME FOR STORAGE OF ERROR-ESTIMATES OF PARAMETERS. '
!  READ(5,2103)DERCOV
!  OPEN(99,FORM='UNFORMATTED',FILE=DERCOV,STATUS='UNKNOWN')
   call filenamegenerator(99)
   OPEN(99,FORM='UNFORMATTED',FILE=DNANE(1,99))

! OPEN SCRATCH FILE HOLDING CONTRIBUTIONS TO NORMAL
! EQUATIONS PROM PARAMETERS, SEE REF (I), EQ. (6.7), DENOTED "A".
   OPEN(2,ACCESS='DIRECT',FORM='UNFORMATTED',&
   STATUS='SCRATCH',RECL=MAXCXB)
   NCXLAS = 0
   NBL2= ((MAXPAR+1)*(MAXPAR+2)/2)/MAXCX+1
   WRITE(6,*)' NUMBER OF BLOCKS NEEDED FOR CX =',NBL2
!
   DO I = 1, MAXCX
    CX(I) = D0
   END DO
   DO I = 1, NBL2
    WRITE(2,REC=I)CX
   END DO
!
   IF (LALLP) THEN
    IF (LINTER) WRITE(6,*)' INPUT NUMBER OF FIXED PARAMETERS'
```



```
    READ(5,*)NPARM
    IF (LWRSOL) WRITE(17,102)NPARM
    IF (NPARM.GT.MAXPAR) THEN
     WRITE(6,160)
  160     FORMAT('TOO LARGE. CHANGE DIMENSION OF IPTYPE. STOP. ')
     STOP
    END IF
    IF (LINTER) WRITE(6,*)' INPUT PARAMETER CODES'
    READ(5,*)(IPTYPE(I),I=1,NPARM)
  150    FORMAT(12I6)
    IF (LWRSOL) WRITE(17,150)(IPTYPE(I),I=1,NPARM)
    CALL WRPAR
! THE SUBROUTINE WRPAR LISTS THE PARAMETERS.
   END IF
  END IF
!
! INPUT OF ONE OR MORE SETS OF OBSERVATIONS.
  WRITE(6,225)
  225  FORMAT(/'  OBSERVATIONS:'/)
!
  SSOBS=D0
  NCXP=0
  LSTOP=LF
  LMEAN1=LF
! obs changed 24.Sep. 2008:
  LMEAN=LF
! end change
  LGRERR=LF
  LMAP = LF
  LSTAT= LF
  LMEGR = LF
  LCOD = LF
! LCOD WILL BE TRUE IF OBSERVATIONS ARE COORDINATE DIFFERENCES, I.E.
! IKP = 6,7,8 or 9.
  LAREA=LF
  LADMU=LF
  LFORM=LF
  LNFORM=LT
  LIN4=LF
  LSIMH=LT
  LMENSI=LF
  LMAP7=LF
  LNUOUT=LF
  LSATP=LF
  LSATPP=LF
  BSIZEE=D0
  BSIZEN=D0
  NSTEP=1
  NSTEPE=1
! STEPE=D1 TO ASSURE CALL OF COMEAN PUTS LMEAQ1 FALSE. 1996.10.08.
  STEPE=D1
  ISATP=0
  NO1=0
  DM = D1
  DA = D0
! ADDED 2000-07-04 BY CCT.
  LKM=LF
! ANDEX(2)=1
! DELETED 2012-0930.
!
! ================================================================
! RETURN POINT WHEN MORE INPUT DATA SETS ARE NEEDED.
! *************** INPUT (9) ****************************************
 2006  CALL DEFDAT(LCOLLO,LPRED,LPARAM,LFOR77,LE,LINTER,LSMAL,LFORM,LP,ICHAR,NMA
X,ITRAC0,RRE,NGRERR)
  NPOINT=0
  LSATPP=LSATP
  IF (LPRED) THEN
! added 2012-10-05.
```

```
   MAXC=0
  END IF
!
  CALL INHEAD(LCOLLO,LREPEC,PW,LNFORM,LADDBP,LINVDE,LADBA,LADDBC)
!
! *************** INPUT (10) **********************************
! RETURN POINT FOR EACH NEW OBSERVATION RECORD,
 2023  CALL INP10(LINTER,LFOR77,LGRID,LFULLO,LTEST,LNOUSE,&
  NWAR,NO2,ITRAC0,OBI,COSB,SINB,COST,SINT)
! NWAR,NO2,ITRAC0,OBI,COSB,SINB,COST,SINT,TAUP,BETP,AZP)
! INPUT OF COORDINATES OF OBSERVATIONS POINTS AND
! THE OBSERVED QUANTITIES AND CONTINGENTLY THEIR STANDARD DEVIATIONS.
! THIS IS FOLLOWED BY INPUT OF PARAMETER IDENTIFICATION CODES IF
! LPARM IS TRUE AND LEQP IS FALSE, AND THE OBSERVATIONS ARE NOT
! SATELLITE ALTIMETRY OR CROSS-OVER DIFFERENCES (IKP = 11 OR 9).
!
! IF AN OBSERVATION IS NOT USED WE MAY STILL HAVE TO READ THE
! ASSOCIATED PARAMETERS OR TRACK IDENTIFIERS.
  IF (LPARAM.AND.(.NOT.LEQP).AND.IKP.NE.9.AND.IKP.NE.11.AND.IKP.NE.13.AND.(.NOT.
LGRADI).AND.MP.NE.0) THEN
   READ(INZ,*)(IPACAT(IPA+I),I=1,MP)
  END IF
  IF (LNOUSE) GO TO 2023
!
! NO LESS THAN 0 SIGNALS END OF FILE.
  IF (NO.GE.0) THEN
   COSLAP =  COS(RLATP)
   SINLAP =  SIN(RLATP)
   COSLOP = COS(RLONGP)
   SINLOP = SIN(RLONGP)
   IF (LINSOL.AND.LNEWSO) THEN
    PW2=VAR(SM,IS,KP,S,AAI,HP,IMAX1,LMENSI,COSLAP,SINLAP,COSLOP,SINLOP,&
    LSATP,SATROT)
   END IF
!
   IF (LMENSI.AND.(.NOT.LMEAN1)) THEN
    RLATP=RLATP+STEPN*D2
! SPHERICAL APPROXIMATION 2001-09-21.
    COSLAP =  COS(RLATP)
    SINLAP =  SIN(RLATP)
   END IF
   IF (LMENSI) THEN
    IF (LMEAN1) THEN
     CALL PAZIM(RLATP,RLONGP,COSLAP,SINLAP,COSLOP,SINLOP,-CAZP,-SAZP,COST2P,SINT
2P,LTEST)
    ELSE
     IF (.NOT.LEQANG) CALL ICMEAN(BSIZEN,STEPE,NSTEPE,COSSTE,SINSTE,COSLAP,SINLA
P,LF,LF)
     RLONGP=RLONGP-STEPE*D2
    END IF
   END IF
   COSLOP = COS(RLONGP)
   SINLOP = SIN(RLONGP)
   IF (LOE1.OR.LOE2) OBS(K2) = OBI(IIE)
!       write(*,*)' LOE1,LOE, K2,OB(IIE) ',LOE1,LOE2,K2,OBI(IIE)
!
   IF (LPARAM.AND.(.NOT.LEQP)) THEN
!
! --------------- INPUT (10A) --------------------------------
! INPUT OF PARAMETER TYPES, IF PARAMETERS ARE GIVEN INDIVIDUALLY
! FOR EACH OBSERVATION. FOR DATA RELATED TO SATELLITE-ALTIMETRY
! WE USE THE REVOLUTION NUMBER(S) WHICH IS IMPLICITLY GIVEN BY THE
! OBSERVATION NUMBER, (CODES IKP=9 FOR CROSS-OVER DIFFERENCES AND
! IKP=11 FOR SEA-SURFACE HEIGHTS TREATED LIKE GEOID UNDULATIONS.)
    CALL INPAR(IKP,NO,ITRACK,IC,N,NOX,NPOBS,NPAOLD,LNOUSE,LINTER,LIN4,LPRED,LDPR
,LWRSOL,LTEST,LONEQ,OBI,IOBS1,NPOINT)
! RETURN TO INPUT (10) IF CROSS-OVER DIFFERENCE COULD NOT BE USED.
    IF (LNOUSE) GO TO 2023
   END IF
```



```
!
   IF (LOE1.OR.LOE2) OBS(K2) = OBI(IIE)
!       write(*,*)' LOE1,LOE, K2,OB(IIE) ',LOE1,LOE2,K2,OBI(IIE)
! LOUTC IS EVALUATED BY THE SUBROUTINE INHEAD, AND IS TRUE IF
! LNEQ AND LCOMP BOTH ARE TRUE.
   IF (LOUTC) THEN
    IF (LOE1.OR.LOE2) OBS(K2) = OBI(IIE)
    IF(LOE2) OBS(K21) = OBI(IIE1)
    IF (K1.EQ.5) OBS(5)=D0
!
    IF (LREPEC.AND.IOBS2.GT.0) OBS(12) = OBI(IOBS2)
    IF (IOBS1.GT.0) OBS(2) = OBI(IOBS1)
! ADDED 2006-08-20.
    IF (LALLCO.AND.LCOMP) THEN
     IF (IORDER.EQ.1) THEN
      ALLIN(1,1)=OBI(IOBS1)
     ELSE
      IF (IORDER.EQ.1.AND.IOBS1.GT.5) THEN
       ALLIN(1,1)=OBI(IOBS1-2)
       ALLIN(1,2)=OBI(IOBS1-1)
       ALLIN(1,3)=OBI(IOBS1)
      ELSE
! WE HERE SUPPOSE THAT GRAVITY GRADIENTS ARE IN THE ORDER XX,YY,ZZ,
! XY,XZ,YZ.
       ALLIN(1,1)=OBI(IOBS1-5)
       ALLIN(2,2)=OBI(IOBS1-4)
       ALLIN(3,3)=OBI(IOBS1-3)
       ALLIN(1,2)=OBI(IOBS1-2)
       ALLIN(1,3)=OBI(IOBS1-1)
       ALLIN(2,3)=OBI(IOBS1)
      END IF
     END IF
    END IF
   END IF
!
   IF (IH.EQ.0) THEN
    OBS(1) = HP
    H=HP
    IF (LMEAN.AND.LSIMH.AND.(.NOT.LWRSOL)) OBS(1) = D0
   ELSE
!       write(*,*)' obi2 ',obs(1)
    H=OBI(1)
    OBS(1)=H
! CORRECTION 2003-04-08.
    IF (LMEAN.AND.LSIMH) THEN
     OBS(1) = HP
     H=HP
    END IF
   END IF
!
! CORRECTING THE OBSERVATION BY AN ADDITIVE AND MULTIPLICATIVE
! CONSTANT.
   IF (LADMU) OBS(2)=OBS(2)*DM+DA
! CORRECTION 2010-11-22.
!  IF (.TRUE.) write(*,*)' 1608 O3,OIIE,IIE ',OBS(3),OBI(IIE),IIE
!       IF (LADMU.AND.(.NOT.LSA)) OBS(3)=OBS(3)*DM+DA
   IF (LKM) H = H*1.0D3
! CONVERSION FROM KM TO M.
! CORRECTION 2004-01-26.
   IF (IKP.GT.10.AND.IH.NE.0) HP = H
   IF (LMEAN.AND.LSIMH) H = D0
   IF (LDEN) THEN
    HP=RRE**2/(RE-HP) - RE
! CONVERSION OF DEPTH TO ARTIFICIAL HEIGHT FOR DENSITY ANOMALIES.
    RP=RE+HP
   END IF
!
   IF (.NOT.(LNGR.AND.(IORDER.NE.2.OR.LNKSIP).AND.(.NOT.LSATP).AND.(.NOT.LMDD)))
 THEN
```

```
!
    RLATS=RLATP
    RLONGS=RLONGP
    COSLA=COSLAP
    SINLA=SINLAP
    COSLO=COSLOP
    SINLO=SINLOP
    REF=D0
! COMPUTATION OF MEAN VALUES IF NSTEP IS .GT. 1.
    DO I=1,NSTEP
     CALL EUCLID(COSLA,SINLA,COSLOP,SINLOP,H,E21,AX1)
     REFI = RGRAV(IPC,IKP,REF1,REF2,REF3,SINLA,H,RG,CU,SU1,&
     LSATP)
     VREF(1)=REF1
     VREF(2)=REF2
     VREF(3)=REF3
!
! CHANGE 1990.10.19 BY CCT CALL OF AXV ADDED .
     IF (LSATP.AND.(.NOT.LGRADI)) THEN
! CALCULATION OF REFERENCE VALUES FOR 1. ORDER DERIVATIVES.
      CALL AXV(SATROT,VREF)
! ADDED 2005-09-21.
      IF (LALLCO) THEN
       ALLREF(1,1)=VREF(1)
       ALLREF(2,1)=VREF(2)
       ALLREF(3,1)=VREF(3)
      END IF
      IF (LGRP) REFI=VREF(3)
      IF (.NOT.LNKSIP) REFI=VREF(2)
      IF (.NOT.LNETAP) REFI=VREF(1)
     END IF
     IF (LMENSI) THEN
      IF (LMEAN1) THEN
! FILTER FACTORS INTRODUCED 1992.11.26 BY CCT.
       REF = REF+REFI*FILTER(I)
       CALL PAZIM(RLATS,RLONGS,COSLA,SINLA,COSLO,SINLO,CAZP,SAZP,COSSTN,SINSTN,L
TEST)
      ELSE
       REF = REF+REFI
       COSLA1=COSLA
       COSLA=COSLA*COSSTN+SINLA*SINSTN
       SINLA=SINLA*COSSTN-COSLA1*SINSTN
      END IF
     ELSE
      REF = REF+REFI
     END IF
    END DO
!
    REF=REF/NSTEP
! COMPUTING THE REFERENCE VALUES.
    IF (LMEGR.AND.(LGRP.OR.(.NOT.LNKSIP).OR.(.NOT.LNETAP)).AND.(.NOT.LMDD)) THEN
! WE SUPPOSE THE FIRST DERIVATIVES ARE IN MGAL WHEN LMEGR IS TRUE.
     OBS(2) = OBS(2)-REF*1.0D5
    END IF
    IF (LMDD.AND.(.NOT.LSATP)) THEN
     OBS(2) = OBS(2)-REF*1.0D9
     IF (LF) WRITE(*,*) ' OB2,REF ',OBS(2),REF
    END IF
    REF0=REF
   END IF
!
   IF (LWLONG.AND.LDEFVP.AND.LREPEC) OBS(12) = - OBS(12)
   IF (LWLONG.AND.LONECO.AND.(.NOT.LNETAP)) OBS(2) = -OBS(2)
!
   OBS(IB) = D0
   POT=D0
   GP=D0
   DUDX=D0
   DUDY=D0
```



```
   IF (LREPEC) OBS(IB1) = D0
   IF (LTERRC) THEN
!
    OBS(ITE)=OBI(IITE)
    IF (LADBTE) OBS(IB)=OBS(ITE)
    IF (LREPEC) THEN
     OBS(ITE1)=OBI(IITE1)
     IF (LADBTE) OBS(IB1)=OBS(ITE1)
    END IF
   END IF
!
   IF (LTRAN.OR.LPOT) THEN
    CALL TRAPOT(LNPOT,LREPEC,LSPHER,LADDBP,LFULLO,POT00,RB,REF,REF0,UREF0,OBI,H,
HPP,RRE,SU,SU8,VREF)
!
    IF (LSCALE) THEN
! CHANGE 2006-04-17. NOW THE VALUE FROM THE SPHERICAL HARMONIC EXPANSION
! WILL BE USED IN THE OBSERVATION EQUATION FOR SCALE-FACTORS (APARM). NOW THE VA
LUE FROM THE SPHERICAL HARMONIC EXPANSION
! WILL BE USED IN THE OBSERVATION EQUATION FOR SCALE-FACTORS (APARM).
     SFACT(N+1)=OBI(IP)
    END IF
   ELSE
! CHANGE 2005-11.12.
    IF (.NOT.LSPHER) THEN
     CALL EUCLID(COSLAP,SINLAP,COSLOP,SINLOP,HP,E22,AX2)
! NO SPHERICAL APPROXIMATION, 2001-09-21.
! CHANGE 2004-08-11.
     IF (.NOT.LSATP.AND.(.NOT.LZETA)) ISATP=0
! CHANGED 2013-05-02 TO INDICATE NO ROTATION.
!    IF (.NOT.LSATP.AND.(.NOT.LZETA)) ISATP=3
! THIS ASSIGNMENT INDICATES FULL ROTATION, USING THE UNIT MATRIX.
     IF (DISTO.LT.RB) THEN
      WRITE(*,*)' POINT INSIDE BJERHAMMAR SPHERE, H::= 10000 M '
      WRITE(*,*)HP,DISTO,RB
      HPP=0.0D0
     ELSE
      HPP=DISTO-RE
! CHANGE 2003-06-02.
      IF (IH.NE.0) HP=HPP
     END IF
!
     COSLAP=XY/DISTO
     SINLAP=Z/DISTO
     RLATP1=ATAN2(Z,XY)
! DLATP IS GEOCENTRIC LATITUDE MINUS GEODETIC LATITUDE. MORE CORRECT
! IS THE ANGLE BETWEEN THE NORMAL GRAVITY FIELD VECTOR, SEE RGRAV.
     DLATP=RLATP1-RLATP
     IF (ABS(DLATP).GT.0.1) THEN
      WRITE(*,*)' ERROR, RLATP,P1 = ',RLATP,RLATP1
     ELSE
! CORRECTION 2003-04-06.
      RLATP=RLATP1
     END IF
     SLAT=RLATP*180.0D0/PI
    ELSE
     HPP=HP
    END IF
   END IF
!
   IF ((.NOT.LRESOL).AND.LCREF) THEN
!
!  write(*,*)' 1689 ',LCOD,IOBSR,NIR,IMAX1R,LSATAC,LPRED
    IF (.NOT.LCOD) THEN
! CHANGE 2005-03-29.
     CALL PRED(SR,AAR,0 ,    0  ,2 ,IOBSR,NIR,IMAX1R,LT   ,LF  ,LF  ,LTCOV,LSATA
C,LF    ,0)
!        PRED(SS,AAI,IS,ISP,ISO,II,IC,   NC, IMAX1, LPRED,LBST,LCST,LTCOV,LSATA
C,LWAIT,NPRED  )
```

```
    ELSE
     PREDP=D0
     PRETAP=D0
    END IF
!
    OBS(IC1) = PREDP
    IF (LADDBC) OBS(IB) = OBS(IB)+OBS(IC1)
!
    IF (LREPEC) THEN
     OBS(IC11) = PRETAP
     IF (LADDBC) OBS(IB1)= OBS(IB1)+OBS(IC11)
    END IF
   END IF
!
   IF (LTNB) OBS(IU) = OBS(IB)-OBS(IT)
   IF (LTEB) OBS(IU) =-OBS(IT)
   IF (LK30) OBS(3) = OBS(2)-OBS(IU)
   IF (LK30) OB1 = OBS(3)
   IF (.NOT.LK30) OB1 = OBS(2)
!
   IF (.NOT.LCOD) THEN
! STORING COORDINATES AND RIGHT-HAND SIDES OF NORMAL-EQ., N COUNTS
! THE COLUMNS AND IC THE STATIONS.
    N = N+1
    IC = IC+1
    IF (LLCOER) THEN
     ITRACE(IC)=-ITRACK
! ADDED 2005-04-04: USE OF CTIME. TO BE CHANGED IF TIME IN DECIMAL
! SEC.
     CTIME(IC)=NO
    END IF
    ICREL=ICREL+1
    NREL=NREL+1
!   CR(N-ISO)= OB1
    B(NREL) = OB1
    IF (LE) THEN
     WOBS(NREL) = OBS(K2)
    ELSE
     WOBS(NREL) = D0
    END IF
    SSOBS = SSOBS+OB1**2/PW2
!
    COSLAT(ICREL) = COSLAP
    SINLAT(ICREL) = SINLAP
    COSLON(ICREL) = COSLOP
    SINLON(ICREL) = SINLOP
    RLONG(ICREL) = RLONGP
    RLAT(ICREL)   = RLATP
    HQ(ICREL)=HPP
!   write(*,*)' 1794 ',ICREL,LMEAN1,LDIF
!
    IF (LSATP.OR.LMEAN1) THEN
     IF (ISATP.EQ.1.OR.LMEAN1) THEN
! ERROR DETECTED (BLOCK MOVED UP) 2003-07-30.
      COSAZ(ICREL)=CAZP
      SINAZ(ICREL)=SAZP
      IF (LDIF) THEN
       SR11(ICREL)=RLATD
       SR12(ICREL)=RLONGD
       SR13(ICREL)=HQD
!      write(*,*)' 1804 ',ICREL,RLATD,RLONGD,HQD
       SINLATD=SIN(RLATD*DEGRAD)
       COSLATD=COS(RLATD*DEGRAD)
       SINLOND=SIN(RLONGD*DEGRAD)
       COSLOND=COS(RLONGD*DEGRAD)
      END IF
     ELSE
      SR11(ICREL)=COSB
      SR12(ICREL)=SINB
```



```
      SR13(ICREL)=COST
      SR22(ICREL)=SINT
      COSAZ(ICREL)=CAZP
      SINAZ(ICREL)=SAZP
     END IF
    END IF
!
    IF (.NOT.LSPHER.AND.(.NOT.LSATP)) THEN
! WE PREPER FOR A ROTATION EQUAL TO THE ANGLE BETWEEN THE
! RADIUS VECTOR AND THE NORMAL GRAVITY FIELD VECTOR AROUND THE 1. AXIS.
! STILL PROBLEM LEFT FOR MEAN-VALUES.
! CHANGE 2001-09-25.
     SATROT(1,1)=D1
     SATROT(2,1)=D0
     SATROT(3,1)=D0
     SATROT(1,2)=D0
     SATROT(2,2)=COS(DLATP)
! SIGN MAY BE WRONG CCCCC
     SATROT(2,3)=-SIN(DLATP)
     SATROT(3,1)=D0
     SATROT(3,2)=-SATROT(2,3)
     SATROT(3,3)=SATROT(2,2)
    END IF
    IF (NREL.GE.MAXO) THEN
!  CHECK OF NUMBER OF OBSERVATIONS NOT EXCEEDING ARRAY LIMIT
     IF (NOBLK.EQ.0) THEN
! ESTABLISHING SCRATCH FILE FOR OBSERVATIONS, SOLUTIONS AND
! ERRORS. CHANGE 1992.07.22 BY CCT.
      WRITE(6,229)
  229      FORMAT(' NUMBER OF OBSERVATIONS REQUIRE STORAGE ON UNIT 14, 15 AND 16
')
!     OPEN(16,ACCESS='DIRECT',FORM='UNFORMATTED',STATUS='SCRATCH',RECL=MAXO9)
      write(*,*)' MAXO9 ',MAXO9
      call filenamegenerator(16)
      OPEN(16,ACCESS='DIRECT',FORM='UNFORMATTED',RECL=MAXO9)
      OPEN(15,ACCESS='DIRECT',FORM='UNFORMATTED',STATUS='SCRATCH',RECL=MAXO6)
! UNIT 15 WILL BE USED TO STORE ALLCOV. CHANGE 2011-02-05.
      OPEN(14,ACCESS='DIRECT',FORM='UNFORMATTED',RECL=MAXO6)
!     OPEN(14,ACCESS='DIRECT',FORM='UNFORMATTED',STATUS='SCRATCH',RECL=MAXO6)
      LOBSST=LT
     END IF
! ADDED 2012-09-19 TO MAKE MULTIPROCESSING START AGAIN WHEN NEW UNIT 16 RECORD
! IS READ.
     IF (NOBLK.GE.0) THEN
      write(*,*)' 1795 JR,ANDEX(JR),ANDEX(JR+1) ',JR,ANDEX(JR),ANDEX(JR+1)
      ANDEX(JR+3)=IKP
      ANDEX(JR)=IC+1
      ISAT(JR)=ISATP
      IF ((.NOT.LSIMH).AND.LMEAN) BSIZE(JR)= BSIZEN
      BSIZE(JR+1)=BSIZEE
      write(*,*)' 1857 ',BSIZEN,BSIZEE,LMEAN,LSIMH
      IF (LCOERR) THEN
       LLCOEE(JR)=LLCOER
       IF (LLCOER) THEN
! HERE WE STORE INFORMATION ON THE ERROR-COVARIANCE FUNCTION. IF LFOUR
! IS TRUE IT IS REPRESENTED BY A FOURIER SERIES AND IF LCTIME IT
! DEPENDS ON TIME DIFFERENCE, WITH TIME STORED IN ICTIME.
        LFOURI(JR)=LFOUR
        LFOURI(JR+1)=LCTIME
        IF (LFOUR) THEN
         JR0=JR
         NFOURI(JR)=NFOUR
! WE THEN NEED TO STORE THE PSD-VALUES.
         WRITE(*,*)' FOURIER COEFFICIENTS READ IN DEFDAT '
!        WRITE(*,*)' NOT IMPLEMENTED '
!        STOP
        ELSE
! HERE IS MISSING PREPARATION FOR FOURIER SERIES REPR. (PSD).
         SCFRDD(JR)=SCFACT
```

```
          SCFRDD(JR+1)=RDD
! CHANGE 2005-03-11.
        END IF
       END IF
      END IF
      JR=JR+2
      NDSET(INT(ISO/10)+1) = NDSET(INT(ISO/10)+1)+1
      write(*,*)' ANDEX UPDATED, JR,ANDEX(JR-2),ANDEX(JR-1) ',&
      JR,ANDEX(JR-2),ANDEX(JR-1),NDSET(INT(ISO/10)+1)
     END IF
     NOBLK=NOBLK+1
     write(*,*)' output to unit 16, noblk= ',noblk
     WRITE(16,REC=NOBLK)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
     IF (LSATAC) THEN
      WRITE(14,REC=NOBLK)SR11,SR12,SR13,SR22,COSAZ,SINAZ
!     WRITE(14,REC=NOBLK)ROTSAT
!     WRITE(*,*)' ROTSATX ',ROTSAT(1),SR11(1)
      WRITE(*,*)' 3 BLOCK ',NOBLK,' WRITTEN, LSATAC= ',LSATAC
     END IF
! HERE IS TO BE ADDED OUTPUT OF PARTIALS FOR PARAMETERS.
     NREL=0
     ICREL=0
    END IF
   END IF
!
   IF (LREPEC) THEN
!
    IF (LTNB) OBS(IU1) = OBS(IB1)-OBS(IT1)
    IF (LTEB) OBS(IU1) = -OBS(IT1)
    IF (LK30) OBS(13) = OBS(12)-OBS(IU1)
    IF (LK30) OB2 = OBS(13)
    IF (.NOT.LK30) OB2 = OBS(12)
    IF (.NOT.LCOD) THEN
     SSOBS = SSOBS+OB2**2/PW2
!    write(*,*)' 1820 SSOBS,OB2,PW2 ', SSOBS,OB2,PW2
     N = N+1
     NREL=NREL+1
     B(NREL) = OB2
!    CR(N-ISO)=OB2
     WOBS(NREL)=D0
     IF (LE) WOBS(NREL) = OBS(K21)
    END IF
!
    IF (NREL.GT.MAXO) THEN
!  CHECK OF NUMBER OF OBSERVATIONS NOT EXCEEDING ARRAY LIMIT
     IF (NOBLK.EQ.0) THEN
      WRITE(6,229)
      OPEN(16,ACCESS='DIRECT',FORM='UNFORMATTED',&
      RECL=MAXO9)
!     STATUS='SCRATCH',RECL=MAXO9)
      OPEN(14,ACCESS='DIRECT',FORM='UNFORMATTED',&
      RECL=MAXO6)
!     STATUS='SCRATCH',RECL=MAXO6)
      LOBSST=LT
     END IF
     NOBLK=NOBLK+1
     write(*,*)' output to unit 16, noblk= ',noblk
     WRITE(16,REC=NOBLK)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
     IF (LSATAC.AND.LOBSST) WRITE(14,REC=NOBLK)SR11,SR12,SR13,SR22,COSAZ,SINAZ
!    IF (LSATAC.AND.LOBSST) WRITE(14,REC=NOBLK)ROTSAT
     NREL=0
     ICREL=0
    END IF
   END IF
!
   CNR =RLATP*IC+CNR
! CNR IS USED AS A KIND OF "CHECKSUM" IN CASE SOLUTIONS ARE
! IN - OR OUTPUT, SECURING THAT THE OBSERVATIONS OCCUR IN THEIR
! PROPER SEQUENCE.
```



```
!
! IF OBS ONLY DEPEND ON PARAMETERS, THEIR CONTRIBUTION IS CALCU-
! LATED AND STORED ON UNIT 2 BY CXPARM.
   IF (LCOD) THEN
    CALL CXPARM(SINLAP,COSLAP,RLONGP,HP,IKP)
    NCXP=NCXP+1
   END IF
!
   IF (LSTAT) CALL COMPA(VG,IKP,LONECO,IU,2)
!
   IF (NO1.EQ.1.AND.LNUOUT) WRITE(6,279)
  279   FORMAT(' ONLY STATION NUMBERS OUTPUT:')
! OUTPUT OF OBSERVATIONS.
   CALL COUT(NO,LONECO,LSMAL,LF,0)
! CHANGE 2002-11-25.
   IF ((LPUNCH.OR.LWRSOL).AND.LSATP.AND.(.NOT.LZETA).AND.(ISATP.EQ.2.OR.ISATP.GT
.3))&
   WRITE(17,282)AZP,BETP,TAUP
  282   FORMAT(3F14.8,i4)
!
   IF (LPARAM.AND.(.NOT.(LEQP.OR.LCOD))) THEN
    IPA = IPA+MP
   END IF
   IF (IPA.GT.IPAMAX) IPAMAX=IPA
!
! RETURN POINT TO INPUT (10).
   IF(.NOT.LSTOP)GO TO 2023
  END IF
!
! *************** INPUT (11) ********************************
!
! IN ORDER TO TERMINATE THE INPUT OF SETS OF OBSERVATIONS, LSTOP MUST
! BE TRUE. IN THIS CASE IT IS POSSIBLE TO READ AN ALREADY COMPUTED SO-
! LUTION (LRESOL=TRUE) AND USE A SET OF ALREADY REDUCED NORMAL EQUA-
! TIONS, (LSANEQ=TRUE).
!
  LNBL1=LF
  IF (LINTER) WRITE(6,*)' INPUT LSTOP, LRESOL _ READ SOLUTION'
  READ(5,*)LSTOP,LRESOL
  IF (LWRSOL) WRITE(17,215)LSTOP,LT
  215  FORMAT(2L2)
!
  IF (LSTAT) CALL COMPA(VG,IKP,LONECO,IU,3)
!
  IF (LPARAM) THEN
! CHANGE 2005-03-25.
   IF (.NOT.LCOD) THEN
    IPACAT(ILAST) = IC
    IPACAT(ILAST+1)=ABS(MP)
   END IF
   IF (LEQP.AND.(.NOT.LCOD)) IPA = IPA+MP
   IF (IPA.GT.IPAMAX) IPAMAX=IPA
!
   IF (LCOD) THEN
    IF (NOUSE.GT.0) WRITE(6,221)NOUSE
  221    FORMAT(I4,' OBSERVATIONS NOT USED',/)
    NOUSE=0
   END IF
   IF ((.NOT.LSTOP).OR.(.NOT.LPOT).OR.LBIN) THEN
    NCXLAS = 0
   END IF
  END IF
!
  IF (LDEN.AND.LPOT) THEN
! THERE SEEMS TO BE AN INCONSISTENCY HERE. 2004-12-05.
   IF (.NOT.L386) THEN
    REWIND 3
    READ(3)COFF
   END IF
```

```
   CM3=GMP
   CMM2=AX
   CM1=OMEGA2
  END IF
!
  IF (.NOT.LCOD) THEN
! ESTABLISHING A CATALOGUE OF THE OBSERVATIONS,MAXIMALLY 9 SETS
! ALLOWED PER COLLOCATION STEP.
! JR IS INITIALIZED TO 2 IN THE BLOCK DATA MODULE.
   ANDEX(JR) = IC+1
!  write(*,*)' 1929 ANDEX(JR-1,JR),JR ',ANDEX(JR-1),ANDEX(JR),JR
   NDSET(INT(ISO/10)+1) = NDSET(INT(ISO/10)+1)+1
!  write(*,*)' 1919 ISO,NDSET(INT(ISO/10)+1)= ',ISO,NDSET(INT(ISO/10)+1)
!  NDSET COUNTS NUMBER OF DATASETS.
   ISAT(JR)=ISATP
   IF ((.NOT.LSIMH).AND.LMEAN) BSIZE(JR)= BSIZEN
   BSIZE(JR+1)=BSIZEE
! ADDED 1998.03.20 BY CCT.
   IF (LCOERR) THEN
    LLCOEE(JR)=LLCOER
    IF (LLCOER) THEN
! HERE WE STORE INFORMATION ON THE ERROR-COVARIANCE FUNCTION. IF LFOUR
! IS TRUE IT IS REPRESENTED BY A FOURIER SERIES AND IF LCTIME IT
! DEPENDS ON TIME DIFFERENCE, WITH TIME STORED IN ICTIME.
     LFOURI(JR)=LFOUR
     LFOURI(JR+1)=LCTIME
     IF (LFOUR) THEN
      NFOURI(JR)=NFOUR
! WE THEN NEED TO STORE THE PSD-VALUES.
      WRITE(*,*)' NOT FULLY IMPLEMENTED '
     ELSE
! HERE IS MISSING PREPARATION FOR FOURIER SERIES REPR. (PSD).
      SCFRDD(JR)=SCFACT
      SCFRDD(JR+1)=RDD
      write(*,7497)jr,scfact,rdd
 7497 format(' jr,scfact,rdd ',i5,2d16.5)
! CHANGE 2005-03-11.
     END IF
    END IF
   END IF
!
   JR = JR+2
!  write(*,*)' 1958 JR ',JR,ANDEX(JR),ANDEX(JR-1)
   IF (ANDEX(JR-3).EQ.IKP.AND.(.NOT.LMEAN).AND.(.NOT.LOBSST)) THEN
! HERE WE CHECK THAT THE NEW DATA-SET IS OF A DIFFERENT KIND THAN THE
! ONE JUST BEFORE. IF NOT THE COUNTER JR IS DIMINISHED BY 2.
!
    JR = JR-2
    ANDEX(JR-2) = ANDEX(JR)
    NDSET(INT(ISO/10)+1) = NDSET(INT(ISO/10)+1) -1
    WRITE(*,*)' SAME DATA-TYPE ENCOUNTERED, NDSET REDUCED BY 1 ',&
     NDSET(INT(ISO/10)+1)
    WRITE(*,*)' ANDEX(JR),JR= ',ANDEX(JR),JR
   END IF
  END IF
!
  IF (LSTOP .AND.LPARAM) THEN
! PREPARING FOR PARAMETER DETERMINATION IN CATALOGUE.
   ANDEX(JR+1)=100
   ANDEX(JR)=ANDEX(JR-2)+NPARM
   JR = JR+2
   IF (ANDEX(JR-1).NE.ANDEX(JR-3)) THEN
    NDSET(INT(ISO/10)+1) = NDSET(INT(ISO/10)+1)+1
    WRITE(*,*)' ANDEXJR-1,ANDEXJR-3,NDSET ',ANDEX(JR-1),ANDEX(JR-3),NDSET(INT(IS
O/10)+1)
   END IF
   IC = IC+NPARM
   KL = MOD((MAXPAR*(MAXPAR+1)/2),MAXCX)
   READ(2,REC=NBL2)CX
```



```
   IF (.FALSE.) THEN
    WRITE(*,*)' BUNIT 2 READ, BLOCK ',NBL2
    WRITE(*,*)(CX(KL+IGG),IGG=1,NPARM)
   END IF
   DO I = 1, NPARM
    NREL=NREL+1

! *** WARNING ** MUST BE CHANGED - 2007-09-24. 2012-02-06.
!   CR(NREL-ISO)=CX(KL+I)
    B(NREL)=CX(KL+I)
! CORRECTION 1999-11-23 BY CCT.
    IF (NREL.GE.MAXO) THEN
!  CHECK OF NUMBER OF OBSERVATIONS NOT EXCEEDING ARRAY LIMIT
     IF (NOBLK.EQ.0) THEN
      WRITE(6,229)
      OPEN(16,ACCESS='DIRECT',FORM='UNFORMATTED',&
      RECL=MAXO9)
!     STATUS='SCRATCH',RECL=MAXO9)
      LOBSST=LT
     END IF
     NOBLK=NOBLK+1
! ADDED 2012-02-06
     B(NREL+1)=SSOBS
  write(*,*)' 2006 ssobs ',ssobs
     write(*,*)' output to unit 16, noblk= ',noblk
     WRITE(16,REC=NOBLK)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
     IF (LSATAC) WRITE(14,REC=NOBLK)SR11,SR12,SR13,SR22,COSAZ,SINAZ
     WRITE(*,*)' 5 BLOCK ',NOBLK,' WRITTEN '
     NREL=0
     ICREL=0
    END IF
   END DO
! THIS IS IN ORDER TO COMPLETE THE RIGHT HAND SIDE WITH
! CONTINGENT CONTRIBUTIONS FROM COORDINATE DIFFERENCES.
   N = N+NPARM
  END IF
  IF ((JR-II) .GE. 19) THEN
!
   WRITE(6,298)
  298   FORMAT(' OBSERVATIONS ARRANGED TOO COMPLICATED')
   GO TO 9999
! ADDED 2003-10-03 BY CCT.
  END IF
  IF (.NOT.LCOD) WRITE(*,*) ' INPUT DATA CONSIST OF ',NDSET(INT(ISO/10)+1),' DAT
ASETS ',&
  ' NUMBER OF DATA POINTS ',ANDEX(JR-2)-1-ISO,JR
!
  IF (LSATAC.AND.(.NOT.LOBSST)) THEN
   DO I=1,NSAT
    SR11A(I)=SR11(I)
    SR12A(I)=SR12(I)
    SR13A(I)=SR13(I)
    SR22A(I)=SR22(I)
    COSAZA(I)=COSAZ(I)
    SINAZA(I)=SINAZ(I)
   END DO
   IF (LF) WRITE(*,*)' ROT TRANSFERRED TO ROTA ',SR11A(1),SR12A(1),&
   SR12A(1),SR22A(1),COSAZA(1),SINAZA(1)
  END IF
! RETURN TO INPUT (9). ==============================================
  IF (.NOT.LSTOP) GO TO 2006
!
  IF (LPARAM.AND.(.NOT.LALLP)) CALL WRPAR
! CHANGE 2005-11-07 TO PERMIT OUTPUT.
!      IF (LPARAM.AND.LONEQ)WRITE(6,296)(IPACAT(I),I=1,IPA)
  IF (LPARAM)WRITE(6,296)(IPACAT(I),I=1,IPA)
  296  FORMAT(/' PARAMETER CATALOGUE:',/,(9I8))
!
! END OF INPUT OF OBSERVATIONS. N = NUMBER OF OBSERVATIONS, IOBS =
```

```
! NUMBER OF OBSERVATION POINTS.
!
  IOBS = IC-ISO
  N = N-ISO
  N1 = N+1
! ADDED 2012-08-05.
  WRITE(*,*)' DATA-STRUCTURE. NUMBER OF DATA-SETS= ',NDSET(INT(ISO/10)+1),ISO
  WRITE(*,*)' JR  FIRST OBS NO.IN NEXT DATA SET, DATA KIND '
  DO M=1,NDSET(INT(ISO/10)+1)
   JR=2*(M+ISO)
   WRITE(*,*)JR,ANDEX(JR),ANDEX(JR+1)
  END DO
  NREL=NREL+1
  NREL0=MOD(N1+ISO,MAXO)
  IF (NREL.NE.NREL0) WRITE(*,*)' WARNING3 NREL.NE.NREL0',NREL
  B(NREL) = SSOBS
! CHANGE 1992.07.19.
  IF (LOBSST) THEN
   NOBLK=NOBLK+1
     write(*,*)' output to unit 16, noblk= ',noblk
   WRITE(16,REC=NOBLK)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
   IF (LSATAC) WRITE(14,REC=NOBLK)SR11,SR12,SR13,SR22,COSAZ,SINAZ
   WRITE(*,*) NOBLK,' BLOCKS STORED '
  END IF
  IF (LTIME) THEN
   CPU2=SYTIME(RCBASE,TIMEARRAY)
   WRITE(6,7470)TIMEARRAY(1),CPU2
  END IF
!
! IF LWRESOL IS TRUE OUTPUT OF 'T' AND VALUE OF N1, WHICH SUBSEQUENTLY
! MAY BE USED AS INPUT IF LRESOL IS TRUE. IN CASE THE VALUES ARE NOT
! CHANGED, A SOLUTION MAY BE COMPLETELY RE-ESTABLISHED, AND IT MAY
! BE POSSIBLE TO COMPUTE ESTIMATES OF THE ERROR OF PREDICTION, USING
! THE ALREADY REDUCED NORMAL EQUATIONS. IN CASE IFC IS DIFFERENT FROM
! N1, IT WILL BE SUPPOSED, THAT THE FIRST IFC COLUMNS ARE REDUCED,
! AND THE ELEMENTS OF THE N1-IFC LAST COLUMNS WILL BE COMPUTED AND
! REDUCED SUBSEQUENTLY. THIS MAY BE USED TO CORRECT A SOLUTION, WHERE
! A COLUMN IS DELETED OR ADDED, OR WHERE AN OBSERVATION IS CHANGED.
!
  IF (LWRSOL) WRITE(17,216)LT,N1
  216  FORMAT(L2,I7)
!
! GEOCOLH WILL SET UP NORMAL EQUATIONS AND SOLVE THESE. HERE IS ALSO
! FOUND INPUT 12 AND 13 (INPUT OF SOLUTIONS).
  if (ltest) then
   write(*,*)' GEOCOLH start  '
   call time_stamp(T1,T2,0)
   T1=T2
  end if
  call timer('GeocolH',1)
  CALL GEOCOLH(LINTER,TIMEARRAY,RCBASE,LNDAT,SSOBS,LSATAC,IBSS,LMTEST,&
  LUNIX)
  call timer('GeocolH',2)
  IF (NCXP.NE.0) WRITE(*,*)' CXPARM CALLED ',NCXP,' TIMES '
  IF (NERCOV.NE.0) WRITE(*,*)' ERCOV CALLED ',NERCOV,' TIMES. '
  if (ltest) then
   write(*,*)' GEOCOLH end  '
   call time_stamp(T1,T2,0)
   T1=T2
  end if
!
  IF (.NOT.LC1) THEN
!
   LC1 = LT
!
! *************** INPUT (14) ********************************
!
! INPUT OF LCREF, WHICH IS TRUE WHEN ONE MORE SET OF OBSERVATIONS
! SHALL BE INPUT AND USED FOR THE ESTIMATION OF ONE MORE HARMONIC
```



```
! FUNCTION AND OF LPARAM, WHICH IS TRUE, WHEN PARAMETERS ARE TO BE
! DETERMINED FROM THE FOLLOWING SET OF OBSERVATIONS.
   IF (LINTER) WRITE(6,371)
  371   FORMAT(' INPUT LCREF, TRUE IF ANOTHER COLLOCATION SOLUTION',&
   ' IS NEEDED',/,&
   '       LNEWDA, TRUE IF PARAMETERS ARE TO BE DETERMINED ')
   READ(5,*)LCREF,LNEWDA
   IF (LWRSOL) WRITE(17,215)LCREF,LNEWDA
   IF (LCREF) THEN
!
    IF (LPARAM.AND.LNEWDA) WRITE(6,373)
  373 FORMAT(' *** WARNING4 ***  THIS MAY NOT WORK.')
    LPARAM = LNEWDA
!
! STORING AWAY THE NECESSARY CONSTANTS FOR COLLOCATION I.
    SR = S
    IOBSR = IOBS
    AAR=AAI
    IMAX1R = IMAX1
    NIR = N1
!    LTABLR=LTABLE
! INITIALIZING VARIABLES FOR START OF COLLOCATION II.
    CPU0=CPU2
    IS = IMAX1+2
    II = 22
    JR = 22
! CHANGE 1992.08.21. EARLIER IC=NIR+2.
    IC = NIR
    N = IC
    ISO = IC
    NREL= MOD(ISO,MAXO)
    ICREL=NREL
    IF (LOBSST) NOBLK=NOBLK-1
    ANDEX(21) = IC
! ADDED FOR VER. 18, 2007-08-27.
    MI1      = 0
    NBL(0)  = 0
    NDSET(2) = 0
    WRITE(6,345)ISO,NDSET(1),NDSET(2)
  345    FORMAT(/' START OF COLLOCATION II: ISO,NDSET1,2= ',3I6,/)
    GO TO 1000
! RETURN TO COLLOCATION STEP II. ==============================
   END IF
  END IF
!
! INITIALIZING VARIABLES FOR PREDICTION. MAXC1 IS THE SUBSCRIPT OF
! THE FIRST ELEMENT IN THE COLUMN FORMING THE RIGHT-HAND SIDE.
  LPRED = LT
  LLCOER=LF
  LCOERR=LF
  LNEQ = LF
  LE = LF
  LC2 = LCREF
  LRESOL=LF
  MAXC1 = MAXC+1
  IF (LCREF) KK = 40
  IF (.NOT.LCREF) KK = 22
  ANDEX(KK+1) = 0
  JR = KK
  MI2=MI1
! IN MI2 IS STORED THE SUBSCRIPT IN THE ARRAY NBL WHICH POINTS AT THE
! LAST REDUCED COLUMN.
!
  IF (LWRSOL) THEN
!
   LWRSOL = LF
   LOPEN7=LF
   LCLU7 = LT
  390   FORMAT(1X,2A3)
```

```
   CLOSE(17)
   CLOSE(13)
  END IF
  IPA = 0
!
!      IF (LBICOV) CALL OUTCOV(CNANE,NCBL)
!      IF (LBISOL) CALL OUTSOL(SNAME,NSBL)
  IF (LPARAM.AND.LF)WRITE(*,347)IPAMAX,(IPACAT(I),I=1,IPAMAX)
  347  FORMAT(' IPAMAX = ',I5,', IPACAT: ',/,(500(6I8,/)))
  INQUIRE(99,OPENED=LOPEN,EXIST=LEXIST,POS=IPOS)
  WRITE(*,*)' UNIT 99 AT POS ',IPOS
  if (LOPEN) THEN
   close(99)
   WRITE(*,*)' unit 99 closed '
  end if
  WRITE(6,344)
  344  FORMAT('  PREDICTIONS: <'/)
!
 END IF
! ================ END LCOLLO=TRUE. ==================================
!
! *************** INPUT (15) ********************************
!
! INPUT OF LOGICALS DEFINING TYPE OF PREDICTION OUT-PUT.
! LGRID - TRUE WHEN PREDICTIONS SHALL BE MADE IN POINTS OF A UNIFORM GRID
!   OR WHEN LSPHER IS TRUE ALL COEFF. FROM A GIVEN DEG. AND ORDER.
! LERNO - TRUE,WHEN THE ESTIMATED ERROR OF PREDICTION SHALL BE COMPUTED.
! LCOMP - TRUE, WHEN OBSERVED AND PREDICTED VALUES SHALL BE COMPAIRED.
! LERNO - TRUE IN CASE LNCOL IS TRUE AND AN INPUT FILE CONTAINS
!   ERROR ESTIMATES OF OBSERVATIONS. THEY ARE SIMPLY REPRODUCED IN OUTPUT.
! LSPHAR- TRUE IF COEFFICIENTS OF FULLY NORMALIZED SPHERICAL HARMONICS ARE
!          TO BE PREDICTED
 2000 IF (LINTER) WRITE(6,1115)
 1115 FORMAT(                                                    &
 ' INPUT: LGRID - TRUE IF COMPUTATIONS IN A GRID                '/&
 '  OR WHEN LSPHAR IS TRUE ALL COEFF. LE THE DEGREE             '/&
 '  LERR - TRUE IF ERROR ESTIMATES ARE TO BE COMPUTED           '/&
 '            OR REPRODUCED IN OUTPUT                           '/&
 '  LCOMP- TRUE IF COMPUTED VALUES ARE SUBTRACTED FROM OBSERVED '/&
 '  LSPHAR - TRUE IF COEFFICIENTS OF SPHERICAL HARMONICS ARE    '/&
 '            TO BE PREDICTED                                   '  )
! ADDITION 1999-05-17 BY CCT.
 READ(5,*)LGRID,LERNO,LCOMP,LSPHAR
!
! --------------- INPUT (15A) ---------------------------------
!
 LNERNO=.NOT.LERNO
 NI = MAXC1
 LGRERR=LF
 LWAIT=LERNO.AND.(.NOT.LNCOL)
! IFC IS INITIALLY EQUAL TO THE NUMBER OF ALREADY REDUCED COLUMNS, I.E.
! HERE THE NUMBER OF OBSERVATIONS.
 IFC=N1-1
 NPRED=0
 NPRED1=0
 MI1=MI2
 IF (LCOLLO) NBL(MI1-1)=IFC
!
! --------------- INPUT (15B) ---------------------------------
!
 IF (LERNO.AND.LCOLLO) THEN
! LERCOV IS TRUE IF ERROR-COVARIANCES ARE COMPUTED .
! ADDED 2005-02-24.
  IF (LINTER) WRITE(*,*) ' COMPUTATION OF ERROR-COVARIANCES (T/F) '
  READ(5,*)LERCOV
  WRITE(*,*)LERCOV
  NPRED=0
  NPRED1=0
  IF (LUNIX) THEN
```



```
   N19=(N+1)*8
!  N20=13*8
! MADE ROOM FOR OUTPUT OF OBS TO UNIT 20, 2012-05-11.
   N20=(16+23+1)*8
! N21 IS THE SIZE OF A BLOCK OF SPHERICAL HARMONIC VALUES USED IN SPHARMA.
   N21=18*8
  ELSE
   N19=(N+1)*2
   N20=(16+23+1)*2
!  N20=13*2
   N21=18*2
  END IF
! FILE TO HOLD POSITIONS, ADDED 2005-08-09.
  IF (.NOT.LOPEN20.OR.LSPHAR) THEN
! change 2012-08-18.
   IF ((.NOT.LSPHAR).AND.(.NOT.LMAP7E)) THEN
    rewind(20)
!   WRITE(*,*) ' INPUT FILE-NAME FOR STORAGE OF POSITIONS '
!   READ(5,2103)POSFIL
    INQUIRE(20,OPENED=LOPEN,EXIST=LEXIST)
    IF (.NOT.(LEXIST.and.LOPEN)) THEN
    WRITE(*,*) ' FILE-NAME FOR STORAGE OF POSITIONS '
    call filenamegenerator(20)
!   OPEN(20,FILE=POSFIL)
    OPEN(20,FILE=DNANE(1,20),FORM='UNFORMATTED',STATUS='UNKNOWN')
!   OPEN(20,FILE=DNANE(1,20),FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
!   OPEN(20,FORM='UNFORMATTED',FILE=POSFIL,ACCESS='SEQUENTIAL')
!   OPEN(20,FORM='UNFORMATTED',FILE=POSFIL,POSITION='APPEND')
!   WRITE(*,*)' UNIT 20 OPENED ',POSFIL
    ELSE
     WRITE(*,*)' UNIT 20 IS OPEN AND EXIST ',DNANE(1,20)
    END IF
   end if
!  close(99)
!  WRITE(*,*) ' INPUT FILE-NAME FOR STORAGE OF ERROR-ESTIMATES '
!  READ(5,2103)DERCOV
   WRITE(*,*) ' FILE-NAME FOR STORAGE OF ERROR-ESTIMATES '
   call filenamegenerator(99)
!  OPEN(99,FORM='UNFORMATTED',FILE=DERCOV,ACCESS='SEQUENTIAL')
! change 2012-08-15.
!  OPEN(99,FORM='UNFORMATTED',FILE=DERCOV,POSITION='REWIND')
!  OPEN(99,FORM='UNFORMATTED',FILE=DERCOV,POSITION='APPEND',STATUS='UNKNOWN')
   OPEN(99,FORM='UNFORMATTED',FILE=DNANE(1,99))
!  OPEN(99,FORM='UNFORMATTED',FILE=DERCOV)
!  WRITE(*,*)' UNIT 99 OPENED, 2328 ',DNANE(1,99)
!  LOPEN20=LT
  ELSE
   REWIND(unit=20)
   REWIND(unit=99)
!  write(*,*)' rewind, 2260, 20 and 99 '
  END IF
  IF (LERCOV) THEN
   IF (LSPHAR) THEN
    IF (LINTER) WRITE(*,*) ' INPUT FILE-NAME FOR STORAGE OF ERROR-COV '
    READ(5,2103)DERCOV
    WRITE(*,*)' FILE ',DERCOV
   ELSE
    WRITE(*,*) ' INPUT FILE-NAME FOR STORAGE OF ERROR-COV AND POSITIONS '
    READ(5,2103)DERCOV
    READ(5,2103)POSFIL
    WRITE(*,*)' FILES ',DERCOV,POSFIL
   END IF
! OUTPUT IS ERROR-COVARIANCES FOR EACH PREDICTED QUANTITY WITH
! ALL PREDICTED IN THE SAME GROUP, AND THE DATA VARIANCE.
! DIRECT ACCESS FILE TO HOLD COVARIANCES OF PREDICTIONS.
! FILE TO HOLD ERROR COVARIANCES.
   OPEN(7,FILE=DERCOV)
! FILE TO HOLD POSITIONS, ADDED 2005-08-09.
!  IF (.NOT.LSPHAR) OPEN(20,ACCESS='DIRECT',FILE=POSFIL,&
```

```
!  FORM='UNFORMATTED',RECL=N20)
  END IF
 ELSE
  LERCOV=LF
 END IF
!
! THE FOLLOWING PART WHERE SPHERICAL HARMONIC COEFFICIENTS ARE ESTIMATED
! HAS BEEN REDESIGNED 2011-08-17.
 IF (LSPHAR) THEN
! CREATION OF TEMPORARY FILE FOR SPHERICAL HARMONIC STORAGE 2011-08-05.
  IF (LUNIX) THEN
! N21 IS THE SIZE OF A BLOCK OF SPHERICAL HARMONIC VALUES USED IN SPHARMA.
   N21=20*8
  ELSE
   N21=20*2
  END IF
  OPEN(98,ACCESS='DIRECT',FORM='UNFORMATTED',STATUS='SCRATCH',RECL=N21)
  WRITE(*,*)' SCRATCH FILE FOR UNIT 98 CREATED '
! ADDITION 2004-08-10.
  IF (LPARAM) THEN
   LALLP=LT
   MP=0
   IPACAT(3)=0
   IPACAT(2)=MP
   CALL PARCAT(LALLP,NPNO)
  END IF
!
! --------------- INPUT (15C) --------------------------------
!
! ADDITION 2006-11-20.
  IF (LINTER) WRITE(*,*)' INPUT SEMI-MAJOR AXIS AND GM TO BE USED '
  READ(5,*)AXS,GMS
  WRITE(*,2233)AXS,GMS
  IF (ABS(AXS-6378137.0D0).GT.2.0) THEN
   WRITE(*,*)' ERROR IN AXIS '
   STOP
  END IF
 2233  FORMAT(' AXIS= ',F12.2,' M, GM = ',D16.8)
  DO J=0, NSPHAR
   SN2(J)=(RE/AXS)**(2*J+2)
  END DO
!
! ADDITION 2005-04-25.
  IF (LINTER) WRITE(*,*)' MUST COEFFICIENTS BE OUTPUT TO FILE (T/F) ? '
  READ(5,*)LOUTCO
  IF (LOUTCO) THEN
   IF (LINTER) WRITE(*,*)' INPUT NAME OF OUTPUT FILE '
   READ(5,*)PCOEF
   WRITE(*,*)' COEFFICIENTS WILL BE OUTPUT TO ',PCOEF
!  OPEN(22,FILE=PCOEF,IOSTAT=J,FORM='FORMATTED')
!  write(*,*)' IOSTAT= ',J
  ELSE
   IF (LINTER) WRITE(*,*)LOUTCO,' UNIT 10 OPENED TO FILE PCOEF '
! CHANGE 2011-06-08, NEEDED FOR RECALL TO FUNCTION.
   OPEN(1,FILE='PCOEF')
  END IF
!
  IF (LGRID) THEN
! THIS WILL ONLY AFFECT THE CALCULATION OF ERROR-ESTIMATES. ALL
! COEFFICIENTS WILL BE PREDICTED.
   IF (LINTER) WRITE(*,*)' INPUT INITIAL DEGREE AND ORDER '
   READ(5,*)IXS,JXS
   WRITE(*,2113)IXS,JXS
 2113   FORMAT(' INITIAL DEGREE AND ORDER ',2I6)
   IF (IXS.NE.(-JXS)) THEN
    JXS=-IXS
    WRITE(*,*)' START ORDER CHANGED TO ',JXS
   END IF
   IF (LERNO) THEN
```



```
    LWAIT=.NOT.LNCOL
!     LWAIT=.FALSE.
! test 2010-06-06.
    IF (.NOT.LERCOV) THEN
! INPUT OF FILE NAME TO STORE COVARIANCES FOR ERROR-ESTIMATION.
     NPRED1=0
     IF (LINTER) WRITE(*,*)' INPUT NAME OF COVARIANCE FILE '
     READ(5,2103)DCOVA
! ADDED 2012-08-09.
     WRITE(*,*)DCOVA
     OPEN(19,ACCESS='DIRECT',FORM='UNFORMATTED',&
     FILE=DCOVA,RECL=N19)
! FILE TO HOLD ERROR COVARIANCES.
    END IF
   END IF
  ELSE
   IXS=0
   JXS=0
   WRITE(*,*)' PREDICTION STARTS FROM DEGREE=0 '
! CHANGE 2011-11-01.
   LWAIT=.FALSE.
  END IF
!
  WRITE(*,*) ' INPUT MAXIMAL DEGREE & ORDER OF COEFF. TO BE PREDICTED '
  READ(5,*)IIDEGM,JJORDM
! CHANGE 2011-08-14.
  IF (IIDEGM.NE.JJORDM) THEN
   JJORDM = IIDEGM
   WRITE(*,*)' MAXIMAL ORDER PUT EQUAL TO MAXIMAL DEGREE '
  END IF
  IIDEG2 = (IIDEGM+1)**2
  WRITE(*,1110) IIDEGM,JJORDM
 1110  FORMAT(2I6)
!
  WRITE(*,*)' LCOMP - OBS-PRED CALCULATED ',LCOMP
  IF (LCOMP) THEN
   IF (IIDEGM.GE.NSPHAR) THEN
    WRITE(*,*)' STORAGE  EXCEEDED FOR COMPARISON. '
    WRITE(*,*)' MAXIMAL DEGREED REDUCED TO ',NSPHAR
    IIDEGM=NSPHAR
   END IF
!
   WRITE(*,*)' INPUT NAME OF COEFF. FILE FOR COMPARISON '
   READ(5,'(A)')CCFILE
   OPEN(21,FILE=CCFILE)
   WRITE(*,*)' INPUT DATA FORMAT '
   READ(5,103)FMT(1)
   WRITE(*,103)FMT(1)
!
! LOOP READING COEFFICIENTS FOR COMPARISON.
 1996   CONTINUE
   READ(21,FMT,END=1997)NII,MII,CCII,CCJJ
   IF (NII.EQ.MII.AND.LF) write(*,*)' coef. read ',nii,mii
!
   IF (MII.LE.IIDEGM.AND.NII.LE.IIDEGM) THEN
    IF (MII.EQ.0) THEN
     TCOEFF((NII)**2+1)=CCII
    ELSE
     IF (MII.LE.NII) THEN
      TCOEFF((NII)**2+2*MII)=CCII
      TCOEFF((NII)**2+2*MII+1)=CCJJ
     END IF
    END IF
   END IF
!
   GO TO 1996
! END READING LOOP.
 1997   CLOSE(21)
   write(*,*)' coeff read '
```

```
!
   SII=0.0D0
   SSII=SII
   SOERR=D0
! ADDED 2004-08-15
   SSCO=D0
  END IF
!
! PUTTING TO ZERO THE ARRAY WHERE PREDICTIONS ARE ACCUMULATED 2000-01-13.
  DO IHH=1,(NSPHAR+1)**2
   SUMIJ(IHH)=D0
  END DO
  OERR=D0
!
! NEW FEATURE ADDED 1999.09.08 BY CCT. ALL COEFFICIENTS FROM THE
! INITIAL DEGREE AND ORDER (IXS,JXS)  UP TO
! AND INCLUSIVE DEGREE IIDEG ARE PREDICTED.
!
! NEW FEATURE ADDED 2000-12-01 (START FROM IXS).
! CHANGED 2011-08-14. NOW ALWAYS START FROM DEGREE/ORDER ZERO.
  IXS=0
  LTSPH=LF
  RLONGP=D0
  RLATP=D0
  SINLOP=D0
  COSLOP=D1
  HP=D0
  ISATP=0
! IKP=17 DEFINES THAT POTENTIAL COEFFICIENTS ARE TO BE PREDICTED. THE
! VALUE IS TRANSFERRED TO COVCX IN KCI(6).
  IKP=17
  KCI(6)=17
  LNKSIP=LT
  LNETAP=LT
!
  JJE=0
  DO JJOR=0,IIDEGM
! JJORD IS THE ORDER WHICH SWITCHES BETWEEN POSITIVE AND NEGATIVE VALUES.
   IF (JJOR.GT.0) JJE=1
   DO IIDEG=JJOR,IIDEGM
! IIDEG IS THE DEGREE.
    DO J=0,JJE
! THIS CAUSES A SWITCH BETWEEN POSITIVE AND NEGATIVE ORDER.
     IF (J.EQ.0) THEN
      JJORD=JJOR
     ELSE
      JJORD=-JJOR
     END IF
! IDEG21 IS THE SUBSCRIPT OF THE PREDICTED VALUE IN SUMII AND OF THE
! ERROR-ESTIMATE IN CCCIJ.
     IF (JJORD.EQ.0) THEN
      IDEG21=IIDEG**2+1
     ELSE
      IF (JJORD.LT.0) THEN
       IDEG21=IIDEG**2-2*JJORD+1
      ELSE
       IDEG21=IIDEG**2+2*JJORD
      END IF
     END IF
!
     IF (LERNO.AND.JJORD.EQ.0) THEN
      PW2=VAR(SM,IS,17,S,AAI,0.0D0,IMAX1,LMENSI,1.0D0,0.0D0,1.0D0,0.0D0,&
      LF,SATROT)
!       WRITE(*,*)' MAXBLT, NT, IDIMCN ',MAXBLT,NT,IDIMC,
      IF (MAXC2.GT.NDIMC) THEN
       WRITE(*,*)' 1: STOP, NDIMC, MAXC2= ',NDIMC,MAXC2
       STOP
      END IF
      PRCOEF(IIDEG)=PW2
```



```
!     C(MAXC2) = PW2
!     write(*,*)' PW2,JJORD:',PW2,JJORD
     END IF
!
! WHEN LINSOL IS TRUE, WE HAVE TO READ THE LAST BLOCK IN ORDER
! THAT THE CALL OF PRED MAY STORE THE COVARIANCES AT THE RIGHT
! POSITIONS IN THE ARRAY C.
!
!      WRITE(*,*)' PRED 2454 ',IS,IPX,ISO,II,IOBS,N1,IMAX1,NI
!    IF (.NOT.LWAIT.OR.NPRED.EQ.0) THEN
     IF (LPRED.AND.NPRED.EQ.0) THEN
      NI = 1
      N1=1
!     NI = MAXC1
! change 2012-03-16.
     ELSE
      NI=NI0
!      WRITE(*,*)' 2462 NI,N1 ',NI,N1
     END IF
     PW2=PRCOEF(IIDEG)
     CALL PRED(S ,AAI,IS,    ISO,II,IOBS,N1,IMAX1,LT   ,LERNO,LF   ,LTCOV ,LSATA
C,LWAIT,NPRED)
!     PRED(SS,AAI,IS,ISP,ISO,II,IC  ,NC,IMAX1,LPRED,LBST,LCST ,LTCOV,LSATAC,LWAI
T ,NPRED      )
     IF (LWAIT.AND.LERNO) THEN
      MAXC2=NI
      IF (LERCOV) THEN
! CHANGE 2012-03-16.
      DO I63=1,NPRED
       IF (NI.GT.NDIMC) THEN
        WRITE(*,*)' 2:NI= ',NI
       END IF
       C(NI)=D0
       NI=NI+1
      END DO
      END IF
      IF (NPARM.GT.0) THEN
       C(NI)=-PW2
      ELSE
       C(NI)=PW2
      END IF
!    if (NI.lt.5000) write(*,*)' NPRED,NI,PW2 ',NPRED,NI,PW2
      NI=NI+1
     END IF
!         write(*,*)' copred w0: NPRED, NI ',NPRED,NI,PW2
     NI0=NI
!
     IF (LERNO) THEN     ! error estimates
!     IF (.NOT.LWAIT) THEN
!      IF (MAXC2.GT.NDIMC) THEN
!       WRITE(*,*)' 3:NDIMC,MAXC2 ',MAXC2,MAXC2
!       STOP
!      END IF
!      C(MAXC2) = PW2
!      IF (NPARM.GT.0) C(MAXC2) = -C(MAXC2)
!     ELSE
       C(MAXC2) = PW2
       IF (NPARM.GT.0) C(MAXC2) = -C(MAXC2)
!       IF (MOD(NPRED,IBSS).EQ.1) WRITE(*,*)' C(MAXC2), MAXC2 ',C(MAXC2),MAXC2
       NPRED = NPRED+1
       IF (MOD(NPRED,IBSS).EQ.0) THEN
        NBL(MI1)=N+NPRED
        NJ=NBL(MI1-1)+1
         IF (LMTEST) write(*,*)' restore-wait0: MI1,IFC,NPRED,NBL,LERCOV ',&
         MI1,IFC,NPRED,NBL(MI1-1),LERCOV
! CHANGE 2012-03-13. WHEN LERNO=LT, ARE ALL COLLUMNS OF LENGTH N1.
         CALL RESTORE_CH(NPRED-IBSS+1,N,LF,LT,LERCOV,LMTEST)
        MI1=MI1+1
        NI0=1
```

```
        IFC = N+NPRED
       END IF
!     END IF
! STORING THE NEW RIGHT-HAND SIDE, SO THAT THE ERROR OF
! PREDICTION CAN BE COMPUTED.
!
! COMPUTATION OF THE ERROR OF PREDICTION. THE CALL OF NES GIVES CSS-
!  CPT*(C**-1)*CP+APT*(C**-1)*A*EXX*AT*(C**-1)*AP.
      IF (LF) WRITE(*,*) ' NES,N1,N,NT,IDIMCN',N1,N,NT,IDIMCN
      IF (JJORD.EQ.-IIDEG.AND.IIDEG.EQ.IIDEGM) THEN
!change 2012-08-15.
!     IF (.NOT.LWAIT.OR.(JJORD.EQ.-IIDEG.AND.IIDEG.EQ.IIDEGM)) THEN
      IF (NPRED.GT.1.AND.LSPHAR) WRITE(*,*) ' restore - nes: NPRED,N ',NPRED,N
!     IF (NPRED.GT.1.AND.LMTEST) WRITE(*,*) ' restore - nes: NPRED,IFC ',NPRED,I
FC,MI1,MI2
       NBL(MI1)=N1+NPRED-1
!      NJ = NBL(MI1) + 1                 ! ???
! CHANGE 2012-03-18.
       NJ = NPRED-MOD(NPRED,IBSS)+1
! CHANGE 2013-01-17.
!      CALL RESTORE_CH(NJ,0,lf,lt,LERCOV,LMTEST)    ! is this correct also for e
rror of prediction?
      IF (NPRED.GT.1.AND.LSPHAR) WRITE(*,*) ' restore - nes: NPRED,NJ,N ',&
      NPRED,NJ,N
       CALL RESTORE_CH(NJ,N,lf,lt,LERCOV,LMTEST)    ! is this correct also for e
rror of prediction?  NO !
       write(*,*)' nes 1, N1,Nj,N ',N1,NJ,N
! corrected 2012-05-01.
!bso cholsol1
   call cholsol(N,0,NPARM1,NPRED,lf,lf,lt,lf,lf)
   print*,'BSO CHOLSOL1'
! change 2012-03-15.
   IF (LOUTCO) THEN
   OPEN(22,FILE=PCOEF,IOSTAT=MI1,FORM='FORMATTED')
   write(*,*)' IOSTAT= ',MI1
   WRITE(22,*)' i,j,cij,sij,ecij,esij '
   WRITE(*,*)' i,j,cij,sij,ecij,esij '
   END IF
   if (LERNO) THEN
    JJE=0
    inquire (99,OPENED=lopen)
    if (lopen) then
      write(*,*)' unit 99 open= ',lopen
      rewind(99)
      write(*,*)' 2586 rewind 99 '
    else
     OPEN(99,FORM='UNFORMATTED',FILE=DNANE(1,99))
     rewind(99)
    end if
    I=1
    DO JJOR1=0,IIDEGM
! JJORD IS THE ORDER WHICH SWITCHES BETWEEN POSITIVE AND NEGATIVE VALUES.
     IF (JJOR1.GT.0) JJE=1
     DO IIDEG1=JJOR1,IIDEGM
! IIDEG IS THE DEGREE.
      DO JJ1=0,JJE
! THIS CAUSES A SWITCH BETWEEN POSITIVE AND NEGATIVE ORDER.
       IF (JJ1.EQ.0) THEN
        JJORD=JJOR1
       ELSE
        JJORD=-JJOR1
       END IF
! IDEG21 IS THE SUBSCRIPT OF THE PREDICTED VALUE IN SUMII AND OF THE
! ERROR-ESTIMATE IN CCCIJ.
       IF (JJORD.EQ.0) THEN
        IDEG21=IIDEG1**2+1
       ELSE
        IF (JJORD.LT.0) THEN
         IDEG21=IIDEG1**2-2*JJORD+1
```



```
        ELSE
         IDEG21=IIDEG1**2+2*JJORD
        END IF
       END IF
       READ(99)OERR
!       READ(99,REC=I)OERR
       I=I+1
       if (OERR.GT.0.0d0) OERR=SQRT(OERR)
       IF (LMTEST) write(*,'(4i6,d16.8)')IIDEG1,JJORD,IDEG21,I-1,OERR
       CCCIJ(IDEG21)=OERR*AXS/GMS
      end do
     end do
    end do
    REWIND(99)
   END IF
      END IF
      LNBL1=MAXBL.EQ.1
! CORRECTION 2004-08-17 AND 2011-08-14.
      IF (NPRED.EQ.1) THEN
       IF (NPARM.GT.0) THEN
        OERR=-OERR
!        write(*,*)' nparm,oerr ',NPARM,OERR
       END IF
      END IF
     END IF
!
     SUMIJ(IDEG21)=PREDP*AXS/GMS
     IF (LCOMP.AND.LT) THEN
!         IF (LCOMP.AND.LF) THEN
      TCOBS=TCOEFF(IDEG21)
      DIFII=SUMIJ(IDEG21)-TCOBS
!
      IF (LERNO) THEN
       IF (LF) WRITE(*,1132)IIDEG,JJORD,SUMIJ(IDEG21),TCOBS,DIFII,&
       CCCIJ(IDEG21)
      ELSE
       IF (LF) WRITE(*,1132)IIDEG,JJORD,SUMIJ(IDEG21),TCOBS,DIFII
      END IF
     END IF
!
    END DO
!        THIS ENDS THE SWITCH BETWEEN POSITIVE AND NEGATIVE ORDER.
! CHANGE 2012-07-20.
!   IF (LWAIT.AND.LERNO) THEN
!    IF (JJORD.EQ.0) THEN
!     WRITE(1,1132)IIDEG,JJORD,SUMIJ(IDEG21),D0,CCCIJ(IDEG21),D0
!    ELSE
!     IF (JJORD.LT.0) WRITE(1,1132) IIDEG,-JJORD,SUMIJ(IDEG21-1),SUMIJ(IDEG21),C
CCIJ(IDEG21-1),CCCIJ(IDEG21)
!    END IF
!   END IF
   END DO
!       END LOOP FOR IIDEG.
  END DO
!      END LOOP OVER JJDEG.
!
  IF (LERNO.AND.NPRED.GT.1) THEN
   WRITE(*,*)' UNITLESS ERROR-ESTIMATES '
  END IF
  NPRED=0
!
  DO IIDEG=0,IIDEGM
! HERE OUTPUT OF COEFF. AND ERROR-ESTIMATES. 2005-04-27.
   IF (LERNO) THEN
    PW2=PRCOEF(IIDEG)
    IF (IIDEG.NE.0) THEN
     WRITE(*,1173)IIDEG,PRCOEF(IIDEG)
 1173     FORMAT(/' DEG=',I4,' COEFF. VAR.= ',D16.6,' (M**2/S**2)**2 ')
     IF (PW2.GT.0.0D0)TOERR(IIDEG,1)= SQRT(PW2)*AXS/GMS
```

```
     IF (PW2.GT.0.0) WRITE(*,1174) SQRT(PW2)*AXS/GMS
 1174     FORMAT('      COEFF. STDV    = ',D16.6,' UNITLESS')
     IF (LCOMP.AND.LERNO) THEN
      WRITE(*,*) ' DEG ORD   PRED. COEF     OBSERVED       DIFFERENCE    EST. ER
R.'
     ELSE
      IF (LCOMP) THEN
       WRITE(*,*)' DEG ORD    PRED. COEF     OBS.COEFF    DIFF. '
      END IF
     END IF
    END IF
   END IF
   SSII=D0
   SSCO=D0
   SII=SSII
   SOERR=D0
   DO JJORD=0,IIDEG
    IF (JJORD.EQ.0) THEN
     IDEG21=IIDEG**2+1
    ELSE
     IDEG21=IIDEG**2+2*JJORD
    END IF
    IF (LCOMP) THEN
 1132     FORMAT(2I4,4D15.5)
     TCOBS=TCOEFF(IDEG21)
     DIFII=SUMIJ(IDEG21)-TCOBS
     IF (IIDEG.NE.0) THEN
      IF (LERNO) THEN
       WRITE(*,1132)IIDEG,JJORD,SUMIJ(IDEG21),TCOBS,DIFII,&
       CCCIJ(IDEG21)
       SOERR=SOERR+CCCIJ(IDEG21)
      ELSE
       WRITE(*,1132)IIDEG,JJORD,SUMIJ(IDEG21),TCOBS,DIFII
      END IF
     END IF
     SII=SII+DIFII
     SSII=SSII+DIFII**2
     SSCO=SSCO+TCOBS**2
     IF (JJORD.NE.0) THEN
      TCOBS=TCOEFF(IDEG21+1)
      DIFII=SUMIJ(IDEG21+1)-TCOBS
      IF (LERNO) THEN
       WRITE(*,1132)IIDEG,-JJORD,SUMIJ(IDEG21+1),TCOBS,DIFII,&
       CCCIJ(IDEG21+1)
       SOERR=SOERR+CCCIJ(IDEG21+1)
      ELSE
       WRITE(*,1132)IIDEG,-JJORD,SUMIJ(IDEG21+1),TCOBS,DIFII
      END IF
      SII=SII+DIFII
      SSII=SSII+DIFII**2
      SSCO=SSCO+TCOBS**2
     END IF
    END IF
    IF (LOUTCO) THEN
     IF (LERNO) THEN
      IF (JJORD.EQ.0) THEN
!      WRITE(*,1132) IIDEG,JJORD,SUMIJ(IDEG21),D0,CCCIJ(IDEG21),D0
       WRITE(22,1132) IIDEG,JJORD,SUMIJ(IDEG21),D0,CCCIJ(IDEG21),D0
      ELSE
!      WRITE(*,1132) IIDEG,JJORD,SUMIJ(IDEG21),SUMIJ(IDEG21+1),CCCIJ(IDEG21),CCC
IJ(IDEG21+1)
       WRITE(22,1132) IIDEG,JJORD,SUMIJ(IDEG21),SUMIJ(IDEG21+1),CCCIJ(IDEG21),CC
CIJ(IDEG21+1)
      END IF
     ELSE
      IF (JJORD.EQ.0) THEN
       WRITE(22,1132) IIDEG,JJORD,SUMIJ(IDEG21),D0
      ELSE
       WRITE(22,1132) IIDEG,JJORD,SUMIJ(IDEG21),SUMIJ(IDEG21+1)
```



```
      END IF
     END IF
    END IF
   END DO
!
   IF (LGRID.AND.IIDEG.NE.0) THEN
    TMEAN=SII/(2*IIDEG+1)
    TSTDV=SQRT((SSII-SII**2/(2*IIDEG+1))/(2*IIDEG))
    TVARI=SQRT(SSII/(2*IIDEG+1))
    TOERR(IIDEG,2)=TVARI
    SOERR=SOERR/(2*IIDEG+1)
    TOERR(IIDEG,3)=SOERR
    IF (SSCO.GT.D0) SSCO=SQRT(SSCO/(2*IIDEG-1))
    TOERR(IIDEG,4)=SSCO
    WRITE(*,1176)TMEAN,TSTDV,TVARI,SOERR,SSCO
 1176    FORMAT(' MEAN, STDV, RMS= ',3D16.5,/&
     ' MEAN COLL ERR= ',D16.5,' COEFF. STDV. ',D16.5/)
   END IF
!
  END DO
!      END LOOP OVER IIDEG(REE).
   CLOSE(22)
   WRITE(*,*)' UNIT 22 CLOSED ',PCOEF
   IF (LGRID) THEN
    WRITE(*,*)
    IF (LERNO) THEN
     WRITE(*,*) ' DEG,   DEG.STDV.    STDV OBS-PRED    STDV COL.ERR.   SD-C '
    ELSE
     WRITE(*,*) ' DEG,     STDV OBS-PRED    STDV-COEFFICIENTS  '
    END IF
!
    DO ITO1=IXS,IIDEGM
     IF (LERNO) THEN
      WRITE(*,1181)ITO1,TOERR(ITO1,1),TOERR(ITO1,2),TOERR(ITO1,3),&
      TOERR(ITO1,4)
     ELSE
      WRITE(*,1181)ITO1,TOERR(ITO1,2),TOERR(ITO1,4)
     END IF
 1181     FORMAT(I5,4D16.7)
    END DO
   ELSE
!
    MAXDOU= (IIDEGM+1)**2
    IF (LOUTCO) THEN
     CLOSE(1)
    ELSE
     WRITE(*,*)' COEFFICIENT PREDICTION RESULTS: '
     WRITE(*,1177)(SUMIJ(IHH),IHH=1,MAXDOU)
 1177     FORMAT(5D14.7)
    END IF
!
    IF (LCOMP) THEN
     WRITE(*,*)' DEG.  MEAN   STDV(OBS-PRED)    STDV(PRED) ',IIDEG
     IHX=5
     DO IHH=2,IIDEGM
      IHH2=2*IHH
      IHH21=IHH2+1
      SI=D0
      SSI=D0
      SCO=D0
      SSCO=D0
      DO IHJ=1,IHH21
       TCOBS=TCOEFF(IHX)
       DIFI=SUMIJ(IHX)-TCOBS
       SI=SI+DIFI
       SSI=SSI+DIFI**2
       SCO=SCO+TCOBS
       SSCO=SSCO+TCOBS**2
       IHX=IHX+1
```

```
      END DO
      SSI=SQRT((SSI-SI**2/IHH21)/IHH2)
      SI=SI/IHH21
      SSCO=SQRT((SSCO-SCO**2/IHH21)/IHH2)
!  OUTPUT
      WRITE(*,1181)IHH,SI,SSI,SSCO
     END DO
     WRITE(*,*)' TOTAL NUMBER OF COEFFICIENTS COMPARED= ',IHX-1
    ELSE
     IF (LERNO) THEN
      WRITE(*,1132)IIDEG,JJORD,PREDP*AXS/GMS,OERR*RE/GMC
     ELSE
      WRITE(*,1132)IIDEG,JJORD,PREDP*AXS/GMS
     END IF
    END IF
   END IF
! END COEFFICIENT ESTIMATION.
!
  write(*,*)' end coefficient-estimation '
  close(1)
 ELSE
!
  IIDEG=-1
!
  SSOBS=D0
  LSTOP=LF
  LMEAN1=LF
  LGRERR=LF
  LGRERS=LF
  IF (LERNO.AND.LRESOL) THEN
   LERNO = LF
   WRITE(6,226)
  226   FORMAT(' *** ERROR WILL NOT BE COMPUTED, REQUIRED NEQ NOT ',&
   'STORED. ***')
  END IF
!
  LNERNO = .NOT.LERNO
  LMAP = LF
  LSTAT= LF
  LMEGR = LF
  LCOD = LF
  LAREA=LF
  LADMU=LF
  LFORM=LF
  LNFORM=LT
  LIN4=LF
  LSIMH=LT
! change 2008-09-24.
  LMEAN = LF
  LMENSI=LF
  LMAP7=LF
  LNUOUT=LF
  LSATP=LF
  LSATPP=LF
  BSIZEE=D0
  BSIZEN=D0
  NSTEP=1
  NSTEPE=1
! STEPE=D1 TO ASSURE CALL OF COMEAN PUTS LMEAQ1 FALSE. 1996.10.08.
  STEPE=D1
  ISATP=0
  NO1=0
  DM = D1
  DA = D0
! ADDED 2000-07-04 BY CCT.
  LKM=LF
!
  IF (LGRID) THEN
!
```



```
! *************** INPUT (16) *********************************
!
! INPUT OF GRID LABEL: MIN, MAX LATITUDE AND LONGITUDE, AND
! GRID SPACING IN LAT, LONG (ALL DECIMAL DEGREES).
! THE DATA TYPE (IKP), THE COORDINATE
! SYSTEM (ICSYS) (.LT.0, GEOCENTRIC, BEST), HP = THE HEIGHT OF
! THE MEAN SPHERE ON WHICH THE POINTS OF PREDICTION ARE SI-
! TUATED, THE VALUE OF LMAP, WHICH IS TRUE, WHEN THE PREDICTIONS
! SHALL BE PRINTED AS A PRIMITIVE MAP, THE VALUE OF LPUNCH, WHICH IS
! TRUE WHEN THE PREDICTIONS SHALL BE OUTPUT TO UNIT 17 AND  THE VALUE
! OF LMEAN, TRUE WHEN THE PREDICTED QUANTITIES ARE MEAN VALUES.
! IF LPUNCH AND LMAP ARE TRUE, THEN THE RESULTS WILL ONLY BE OUTPUT
! TO UNIT 17, ON GI STANDARD GRID FORM.
! THIS IS THEN FOLLOWED BY
! (A) IF LPUNCH INPUT OF FILE NAME CONNECTED TO UNIT 17.
! (B) IF LPARM IS TRUE NUMBER OF PARAMETERS AND CODES,
! (C) IF LCOMP IS TRUE, THE SAMPLING INTERVAL MAGNITUDE, VG.
! (D) IF LMEAN IS TRUE, THEN LSIMH, TRUE IF THE MEAN VALUE FUNCTIONAL
!     IS SIMULATED BY MOVING THE HEIGHT UP TO A CERTAIN ALTITUDE,
!     LEQANG, TRUE FOR EQUAL ANGULAR BLOCKS, FOLLOWED BY THE BLOCK-
!     SIZE IN LATITUDE AND IF LEQANG IS TRUE THE BLOCK SIZE IN LON-
!     GITUDE AND THE MIDDLE LATITUDE OF THE AREA IN DECIMAL DEGREES.
!     OTHERWISE TWO ZERO VALUES MUST BE GIVEN.
! IF IKP=10 (DENSITY ANOMALIES), THEN ALSO INPUT 9J MUST BE USED.
!
   IF (LINTER) WRITE(6,1116)
 1116   FORMAT(' INPUT GRID SPECIFICATION',/&
   '  MIN, MAX LATITUDE, MIN, MAX LONGITUDE, STEP IN LAT AND LONG',/&
   '  FUNCTIONAL TYPE (CODE), NEG. VALUE THEN SPH.EXP. SUBTRACTED',/&
   '  COORD.SYSTEM CODE (-1 THEN GLOBAL SYSTEM)',/&
   '  HEIGHT OF GRID POINTS (M)',/'   LMAP - PRIMITIVE MAP OUTPUT',/&
   '  LPUNCH- OUTPUT TO UNIT 17',/'  LMEAN - MEAN VALUES OUTPUT',/&
   '   LMAP&LPUNCH  SIMULTANEOUS TRUE, ONLY MAP OUTPUT TO UNIT 17')
   READ(5,*)RLAMIN,SLAC,SLOC,RLOMAX,GLA,GLO,IKP,ICSYS,HP,LMAP,LPUNCH,LMEAN
! HANNGE 2008-12-08.
   LRESOL=IKP.LT.0
   IF (LRESOL) IKP = - IKP
   NLO= (RLOMAX-SLOC)/GLO+0.5
   NLA= (SLAC-RLAMIN)/GLA+0.5
   WRITE(*,*)' GRID CONSIST OF ',NLA+1,' * ',NLO+1,' POINTS'
   LGIGRS=IKP.GT.40.AND.IKP.LT.90
   IF (LGIGRS) THEN
    IKP=IKP-40
    WRITE(*,*)' DATA IN 3-D FRAME ORIENTATION '
   END IF
   LGRADI=IKP.EQ.15.OR.IKP.EQ.30.OR.IKP.EQ.35.OR.(IKP.GE.20.AND.IKP.LE.25)
   LSMAL=HP.GT.1.0D4.AND.IKP.NE.11
! ADDITION 1999.12.13 BY CCT.
   LSATP=(IKP.EQ.12.OR.IKP.EQ.16.OR.IKP.EQ.17.OR.IKP.EQ.13.OR.LGRADI.OR.IKP.EQ.1
1).AND.LGIGRS
   LSATPP=LSATP
   IF (LSATP) THEN
    ISATP=3
    AZP=90.0D0
    BETP=D0
    TAUP=D0
    SAZP = SIN(AZP*DEGRAD)
    CAZP = COS(AZP*DEGRAD)
    SINB = SIN(BETP*DEGRAD)
    COSB = COS(BETP*DEGRAD)
    SINT = SIN(TAUP*DEGRAD)
    COST = COS(TAUP*DEGRAD)
    SATROT(1,1) =  SAZP*COSB
    SATROT(1,2) =  CAZP*COST+SINT*SINB*SAZP
    SATROT(1,3) = -CAZP*SINT+COST*SAZP*SINB
    SATROT(2,1) = -CAZP*COSB
    SATROT(2,2) =  SAZP*COST-SINT*SINB*CAZP
    SATROT(2,3) = -SAZP*SINT-COST*SINB*CAZP
    SATROT(3,1) = -SINB
```

```
    SATROT(3,2) =  SINT*COSB
    SATROT(3,3) =  COSB*COST
   ELSE
! CORRECTION 2003-03-22.
    ISATP=0
   END IF
   IF (LPUNCH)  THEN
!
! -------------------- INPUT (16A) ----------------------------
    IF (LINTER) WRITE(6,*)' INPUT NAME OF FILE TO HOLD RESULT'
    READ(5,2103)DNAME(1)
    IF (.NOT.(LOPEN7.AND.OLDN(1).EQ.DNAME(1).AND.OLDN(2).EQ.DNAME(2))) THEN
     IF (LOPEN7) THEN
      WRITE(*,*)' UNIT 17 CLOSED AT LABEL 2085 '
      CLOSE(17)
     END IF
     OPEN(17,FILE=DNAME(1),STATUS='UNKNOWN',FORM='FORMATTED')
     WRITE(6,290)(DNAME(I),I=1,ICHAR)
  290     FORMAT(/' SIMULTANEOUS OUTPUT TO UNIT 17, FILE: ',2A128)
     LOPEN7=LT
     OLDN(1)=DNAME(1)
     OLDN(2)=DNAME(2)
    END IF
   END IF
!
   LMAP7=LMAP.AND.LPUNCH
   IF (IKP.EQ.1.OR.IKP.EQ.11) LSMAL=LF
   LMAP7E=LMAP7.AND.LERNO
   LWAIT=LERNO.AND.(.NOT.LNCOL)
! ADDITION 1994.02.04 BY CCT.
   IF (LMAP7E) THEN
    IF (LINTER) WRITE(6,*)' INPUT NAME OF FILE TO HOLD ERROR'
    READ(5,2103)DNAME(1)
    OPEN(11,FILE=DNAME(1),FORM='FORMATTED')
    INQUIRE(11,OPENED=LOPEN)
    WRITE(*,*)' UNIT 11 OPEN= ',LOPEN
    WRITE(6,291)(DNAME(I),I=1,ICHAR)
  291    FORMAT(/' ERROR SIMULTANEOUS OUTPUT TO UNIT 11 FILE: ',2A128)
! LWAIT INDICATES THAT ERROR-ESTIMATES ARE COMPUTED WHEN THE
! LAST GRID VALUE IS COMPUTED.
    NPRED=0
   END IF
   IF (LMAP7)WRITE(17,392)RLAMIN,SLAC,SLOC,RLOMAX,GLA,GLO
   IF (LMAP7E)WRITE(11,392)RLAMIN,SLAC,SLOC,RLOMAX,GLA,GLO
   IF (LMAP7) LMAP=LF
!
   LNEWD=ICSYS.LT.0
   LTRAN=.NOT.LNEWD
   RP= RE+HP
   JIMAX = (NLA+1)*(NLO+1)
   IF (LCOMP.AND.JIMAX.GT.NMAP) JI=NMAP
   LMAP=LMAP.AND.JIMAX.LE.NMAP
   LWLONG=LF
   LSTAT=LCOMP
   LGRP=IKP.EQ.2.OR.IKP.EQ.13.OR.IKP.EQ.12
   LDEN=IKP.EQ.10
   IF (LDEN.AND.L386.AND.LPOT) THEN
    WRITE(6,*)' DENSITY COMPUTATION REQUIRES COEFF. IN CORE'
    STOP
   END IF
   LZETA=IKP.EQ.1.OR.IKP.EQ.11
!
   IF (LPARAM) THEN
    IPA=2
    LEQP=LT
!
! --------------- INPUT (16B) --------------------------------
! INPUT OF NUMBER OF PARAMETERS AND PARAMETER IDENTIFICATION CODES.
    IF (LINTER) WRITE(6,*)' INPUT NUMBER OF PARAMETERS & CODES '
```



```
    READ(5,*)MP,(IPACAT(I+2),I=1,MP)
    IF (MP.GT.0.AND.MP.LT.4) WRITE(6,170)MP,(IPACAT(I+2),I=1,MP)
  170   FORMAT(/' OBSERVATIONS CONTRIBUTE TO/DEPEND ON ',I3,&
   ' PARAMETERS',3I6)
    IF (MP.GT.3) WRITE(6,171)MP,(IPACAT(I+2),I=1,MP)
  171   FORMAT(/' OBSERVATIONS CONTRIBUTE TO/DEPEND ON ',I3,&
   ' PARAMETERS',3I6,(/,12I6))
    IPACAT(2)=MP
    CALL PARCAT(LALLP,NPNO)
   END IF
!
   IOBS2 = 0
   IH = 0
!
! --------------- INPUT (16C) --------------------------------
   IF (LINTER.AND.LCOMP) WRITE(6,*)' INPUT HISTOGRAM BIN-WIDTH'
   IF (LCOMP) READ(5,*)VG
!
! --------------- INPUT (16D) --------------------------------
   IF (LINTER.AND.LMEAN)WRITE(6,1117)
 1117   FORMAT(' INPUT PARAMETERS DEFINING TYPE AND SIZE OF MEAN VALUE',/&
   ' LSIMH - TRUE IF WE USE EQUIVALENT HEIGHT REPR. OF MEAN',/&
   ' LEQANG- TRUE IF EQUAL ANGULAR OR 1-D BLOCK',/&
   ' BLOCK SIDE LENGTH IN LAT. & LONG. (MIN) OR (LENGTH .0 IF 1D)',/&
   ' LATITUDE OF TOTAL AREA MEAN')
  231   FORMAT(2L2,3F10.2)
   IF (LMEAN) READ(5,*)LSIMH,LEQANG,BSIZEN,BSIZEE,&
   RLATP
   LMENSI=.NOT.LSIMH
   LFILTE=LF
! LMEAN1 IS TRUE IF MEAN VALUES ARE 1D, ALONG A SATELLITE OR
! AIRCRAFT TRACK, FOR EXAMPLE BUT IS NOT MEANINGFULL FOR A GRID.
   LMEAN1=.NOT.LSIMH.AND.(.NOT.LEQANG).AND.ABS(BSIZEE).LT.1.0D-8
!  write(*,*)' 3208 ',LMEAN1
   IF (LMEAN1) CALL MEAN1(FILTER,NFILTE,SAZP,CAZP,LFILTE,LGRID,LINTER)
!
! *************** INPUT (17) ********************************
!
   IF (LINTER.AND.LCOMP)WRITE(6,*)' INPUT OBSERVED GRID VALUES'
   IF (LCOMP) READ(5,*)(IMAP(I),I=1,JI)
! INPUT OF OBSERVATIONS IN GRID-FORM. FIRST VALUE IN NORTH-WEST
! CORNER. MAXIMALLY NMAP VALUES CAN BE USED.
!
   H = D0
   IF (.NOT.(LMEAN.AND.LSIMH)) H = HP
! MEAN VALUES ARE SUPPOSED TO BE CALCULATED AT HEIGHT ZERO,
! AND COVARIANCES ARE COMPUTED AT HEIGHT H, IF LSIMH IS TRUE.
   LKM = H.GE.1.0D6
   H0 = H
! H0 SAVES THE INITIAL VALUE OF H, WHICH MAY BE CHANGED FOR
! GRAVITY ANOMALIES.
   OBS(1) = H
!
   IF (LKM) THEN
! H IN KM IS STORED FOR OUTPUT.
    OBS(1) = H*1.0D-3
   END IF
   write(*,*) ' H, HP  2997 ',H,HP
!
   LSTOP = LT
   IANG = 3
   NO = 0
   IOBS1 = 5
!
   DO NAI=0,NLA
    SLAT = -NAI*GLA+SLAC
    IF (SLON.GT.360.0) SLON = SLON-360.0
    IF (SLON.LT.(-360.0)) SLON = SLON+360.0
    IF (LCOMP.AND.NO.LE.400) OBS(2) = IMAP(NO)/100.0
```

```
    H = H0
!
! ---------------- INPUT (17A) -------------------------------
! THIS CALL ONLY TAKES PLACE FOR THE FIRST POINT IN THE GRID.
    IF (NAI.EQ.0)  THEN
     IF (LDEN) CALL DENDEF(NMAX,LINTER,LWRSOL,LPARAM,LPOT,LBIPOT,LBIN,LINSOL,LDE
NOL,LSKIPL,RRE)
! INPUT OF EXPONENT OF WEIGHT FACTOR ON HARMONIC DENSITY,
! RADIUS OF SPHERE WITHIN WICH MASSES ARE LOCATED IN M. CF. REF(F),
! SECTION 3,SCALE FACTOR AND VALUE OF LOGICAL VARIABLE LNPOT
! TRUE, IF NEW SET OF COEFFICIENTS ARE TO BE USED FOR THE DENSITY
! COMPUTATIONS, (DIFFERENCES BETWEEN THE ORIGINAL COEFFICIENTS AND
! COEFFICIENTS OF A TOPOGRAPHIC-ISOSTATIC REDUCTION POTENTIAL).
! IF POTENTIAL COEFFICIENTS NOT ALREADY STORED ON BINARY FORM
! ALSO THE NAME OF THE FILE TO BE CONNECTED TO UNIT 3.
! THIS ONLY APPLIES IF LPOT IS TRUE.
!
! ==============================================================
     CALL INHEAD(LCOLLO,LREPEC,PW,LNFORM,LADDBP,LINVDE,LADBA,LADDBC)
    END IF
!
    DO NOI=0,NLO
     NOI1=NOI
     NO = NO+1
!    IF (LERNO) NPRED=NPRED+1
! CHNNGE 2013-10-01.
     NPRED=NPRED+1
     SLON = NOI*GLO+SLOC
     CALL RAD(IDLAT,MLAT,SLAT,RLATP,3)
     CALL RAD(IDLON,MLON,SLON,RLONGP,3)
!
     COSLAP =  COS(RLATP)
     SINLAP =  SIN(RLATP)
     COSLOP = COS(RLONGP)
     SINLOP = SIN(RLONGP)
     IF (LINSOL.AND.LNEWSO) THEN
      PW2=VAR(SM,IS,KP,S,AAI,HP,IMAX1,LMENSI,COSLAP,SINLAP,COSLOP,SINLOP,&
      LSATP,SATROT)
     END IF
!
     IF (LMENSI.AND.(.NOT.LMEAN1)) RLATP=RLATP+STEPN*D2
! SPHERICAL APPROXIMATION 2001-09-21.
     COSLAP =  COS(RLATP)
     SINLAP =  SIN(RLATP)
     IF (LMENSI) THEN
      IF (LMEAN1) THEN
       CALL PAZIM(RLATP,RLONGP,COSLAP,SINLAP,COSLOP,SINLOP,-CAZP,-SAZP,COST2P,SI
NT2P,LTEST)
      ELSE
       IF (.NOT.LEQANG) CALL ICMEAN(BSIZEN,STEPE,NSTEPE,COSSTE,SINSTE,COSLAP,SIN
LAP,LF,LF)
       RLONGP=RLONGP-STEPE*D2
      END IF
     END IF
     COSLOP = COS(RLONGP)
     SINLOP = SIN(RLONGP)
!
     IF (.NOT.(LNGR.AND.(IORDER.NE.2.OR.LNKSIP).AND.(.NOT.LSATP).AND.(.NOT.LMDD)
)) THEN
      RLATS=RLATP
      RLONGS=RLONGP
      COSLA=COSLAP
      SINLA=SINLAP
      COSLO=COSLOP
      SINLO=SINLOP
      REF=D0
      DO I=1,NSTEP
       CALL EUCLID(COSLA,SINLA,COSLOP,SINLOP,H,E21,AX1)
       REFI = RGRAV(IPC,IKP,REF1,REF2,REF3,SINLA,H,RG,CU,SU1,LSATP)
```



```
       VREF(1)=REF1
       VREF(2)=REF2
       VREF(3)=REF3
!
! CHANGE 1990.10.19 BY CCT CALL OF AXV ADDED .
       IF (LSATP.AND.(.NOT.LGRADI)) THEN
        CALL AXV(SATROT,VREF)
        IF (LGRP) REFI=VREF(3)
        IF (.NOT.LNKSIP) REFI=VREF(2)
        IF (.NOT.LNETAP) REFI=VREF(1)
       END IF
       IF (LMENSI) THEN
        IF (LMEAN1) THEN
! FILTER FACTORS INTRODUCED 1992.11.26 BY CCT.
         REF = REF+REFI*FILTER(I)
         CALL PAZIM(RLATS,RLONGS,COSLA,SINLA,COSLO,SINLO,CAZP,SAZP,COSSTN,SINSTN
,LTEST)
        ELSE
         REF = REF+REFI
         COSLA1=COSLA
         COSLA=COSLA*COSSTN+SINLA*SINSTN
         SINLA=SINLA*COSSTN-COSLA1*SINSTN
        END IF
       ELSE
        REF = REF+REFI
       END IF
      END DO
      REF=REF/NSTEP
! COMPUTING THE REFERENCE VALUES.
      IF (LMEGR.AND.(LGRP.OR.(.NOT.LNKSIP).OR.(.NOT.LNETAP)).AND.(.NOT.LMDD)) TH
EN
! WE SUPPOSE THE FIRST DERIVATIVES ARE IN MGAL WHEN LMEGR IS TRUE.
       OBS(2) = OBS(2)-REF*1.0D5
      END IF
      IF (LMDD.AND.(.NOT.LSATP)) THEN
       OBS(2) = OBS(2)-REF*1.0D9
       IF (LF) WRITE(*,*)' OB2,REF ',OBS(2),REF
      END IF
      REF0=REF
     END IF
!
     IF (LWLONG.AND.LDEFVP.AND.LREPEC) OBS(12) = - OBS(12)
     IF (LWLONG.AND.LONECO.AND.(.NOT.LNETAP)) OBS(2) = -OBS(2)
!
     OBS(IB) = D0
     POT=D0
     GP=D0
     DUDX=D0
     DUDY=D0
     IF (LREPEC) OBS(IB1) = D0
     IF (LTERRC) THEN
!
      OBS(ITE)=OBI(IITE)
      IF (LADBTE) OBS(IB)=OBS(ITE)
      IF (LREPEC) THEN
       OBS(ITE1)=OBI(IITE1)
       IF (LADBTE) OBS(IB1)=OBS(ITE1)
      END IF
     END IF
!
     IF (LTRAN.OR.LPOT) THEN
      CALL TRAPOT(LNPOT,LREPEC,LSPHER,LADDBP,LFULLO,POT00,RB,REF,REF0,UREF0,OBI,
H,HPP,RRE,SU,SU8,VREF)
     ELSE
! CHANGE 2004-07-09.
      IF (.NOT.LSPHER) THEN
       CALL EUCLID(COSLAP,SINLAP,COSLOP,SINLOP,HP,E22,AX2)
!
! NO SPHERICAL APPROXIMATION, 2001-09-21.
```

```
! CHANGE 2004-08-11.
       IF (.NOT.LSATP.AND.(.NOT.LZETA)) ISATP=1
! THIS ASSIGNMENT INDICATES NO ROTATION 2013-05-02.
!      IF (.NOT.LSATP.AND.(.NOT.LZETA)) ISATP=3
! THIS ASSIGNMENT INDICATES FULL ROTATION, USING THE UNIT MATRIX.
       IF (DISTO.LT.RB) THEN
        WRITE(*,*)' POINT INSIDE BJERHAMMAR SPHERE, H::= 10000 M '
        WRITE(*,*)HP,DISTO,RB
        HPP=0.0D0
       ELSE
        HPP=DISTO-RE
! CHANGE 2003-06-02.
        IF (IH.NE.0) HP=HPP
       END IF
!
       COSLAP=XY/DISTO
       SINLAP=Z/DISTO
       RLATP1=ATAN2(Z,XY)
! DLATP IS GEOCENTRIC LATITUDE MINUS GEODETIC LATITUDE. MORE CORRECT
! IS THE ANGLE BETWEEN THE NORMAL GRAVITY FIELD VECTOR, SEE RGRAV.
       DLATP=RLATP1-RLATP
       IF (ABS(DLATP).GT.0.1) THEN
        WRITE(*,*)' ERROR, RLATP,P1 = ',RLATP,RLATP1
       ELSE
! CORRECTION 2003-04-06.
        RLATP=RLATP1
       END IF
       IF (IANG.EQ.6) SLAT=RLATP*180.0D0/PI
      ELSE
       HPP=HP
      END IF
     END IF
!
     IF (LCREF) THEN
!     write(*,*)' 3308 SRAAR,IOBSR,NIR,IMAX1R ',SR,AAR,IOBSR,NIR,IMAX1R
      CALL PRED(SR,AAR,0 ,    0  ,2 ,IOBSR,NIR,IMAX1R,LT   ,LF  ,LF  ,LTCOV,LSAT
AC,LF   ,0    )
          !PRED(SS,AAI,IS,ISP,ISO,II,IC   ,NC ,IMAX1 ,LPRED,LBST,LCST,LTCOV,LSAT
AC,LWAIT,NPRED)
!
!     write(*,*)' pred called 3308 '
      OBS(IC1) = PREDP
      IF (LADDBC) OBS(IB) = OBS(IB)+OBS(IC1)
!
      IF (LREPEC) THEN
       OBS(IC11) = PRETAP
       IF (LADDBC) OBS(IB1)= OBS(IB1)+OBS(IC11)
      END IF
     END IF
!
     IF (LTNB) OBS(IU) = OBS(IB)-OBS(IT)
     IF (LTEB) OBS(IU) =-OBS(IT)
     IF (LK30) OBS(3) = OBS(2)-OBS(IU)
     IF (LK30) OB1 = OBS(3)
     IF (.NOT.LK30) OB1 = OBS(2)
!
     IF (LCOLLO) THEN
      IF (.NOT.LWAIT.OR.NPRED.EQ.0) THEN
       NI = MAXC1
      ELSE
       IF (NPRED.GE.1) THEN
        IF (MOD(NPRED,IBSS).EQ.1) THEN
         NI=1
        ELSE
         NI=N1*((NPRED-1)-INT((NPRED-1)/IBSS)*IBSS)+1
        END IF
       END IF
!      WRITE(*,*)' 3443 NI,NPRED,N1 ',NI,NPRED,N1
      END IF
```



```
!     WRITE(*,*)' 3445 NI,NPRED,N1 ',NI,NPRED,N1
      LEROUT=LMAP7E.AND.(NAI.EQ.NLA).AND.NOI.EQ.NLO
! THIS VARIABLE HAS NO USE AFTER 2012-05-12 SINCE ERROR COMPUTATION IS
! MADE AFTER ALL ESTIMATES HAVE BEEN COMPUTED.
      LWAIT=.NOT.LNCOL
      CALL COPRED(PREDCO,PW2,OBI,WM,SM,&
      KP,NPARM,                                NPRED,NPRED1,&
      LERNO,       LREPEC,LTCOV,LSATAC,LADBA,LTNB,LTEB,LCOMP,&
      LFOUND,LCOD,LMENSI,LSATP,LGRERS,LGRERR,LSA,LERCOV,&
             LWAIT,NLO,      LMTEST)
!     KP,NPARM,NFILE,NPARM1,NERRM,NGRE,NGRERR,NPRED,NPRED1,&
!     LERNO,LINSOL,LREPEC,LTCOV,LSATAC,LADBA,LTNB,LTEB,LCOMP,&
!     LFOUND,LCOD,LMENSI,LSATP,LGRERS,LGRERR,LSA,LERCOV,&
!     LEROUT,LWAIT,NLO,LGRID,LMTEST)
      NI0=NI
!     write(*,*)' 3461, COPRED called,OBI ',(OBS(i),I=1,5)
      IF (LERNO) THEN
!      write(*,*)' LREPEC ',LREPEC
       IF (.NOT.LREPEC) THEN
!        write(*,*)' stored on unit 20, rec= ',npred,no,PREDCO,obs
         WRITE(20)PREDCO,OBS,NO,CLATD,SLAT,SLON,IDLAT,IDLON,MLAT,MLON
!        WRITE(20,REC=NPRED)PREDCO,OBS,NO
       ELSE
        WRITE(20)PREDCO,OBS,NO,CLATD,SLAT,SLON,IDLAT,IDLON,MLAT,MLON
!       WRITE(20,REC=(INT(NPRED/2)))PREDCO,OBS,NO
!       write(*,*)' stored on unit 20, rec= ',npred/2,no
!       write(*,7979)PREDCO,OBS,NO
 7979   format(6d13.4)
       END IF
      END IF
     ELSE
      IF (LREPEC) THEN
       OBS(IA1) = PRETAP
       IF (LADBA) OBS(IB1) = OBS(IB1)+OBS(IA1)
       IF (LTNB) OBS(IU1) = OBS(IB1)-OBS(IT1)
       IF (LTEB) OBS(IU1) = -OBS(IT1)
       IF (LCOMP) OBS(13) = OBS(12)-OBS(IU1)
      END IF
!
!     write(*,396)no,(obs(k),k=1,5)
!396  format(i6,5f12.3)
      OBS(IA) = PREDP
      IF (LADBA) OBS(IB) = OBS(IB)+OBS(IA)
      IF (LTNB) OBS(IU) = OBS(IB)-OBS(IT)
      IF (LTEB) OBS(IU) =-OBS(IT)
      IF (LCOMP) OBS(3) = OBS(2)-OBS(IU)
     END IF
!    write(*,*)' 3467 OBS3456,IU ',OBS(3),OBS(4),OBS(5),OBS(6),IU
!
     IF (LSTAT) CALL COMPA(VG,IKP,LONECO,IU,2)
!
     IF (LMAP7) THEN
! CHANGE 2005-10-10 (9 -> 8).
      IPR=MOD(NOI1,8)+1
      PRV(IPR)=OBS(IU)
!     IF (LMAP7E.AND.(.NOT.LERNO))PRVE(IPR)=OBS(K2)
!          write(*,*)' 3322 ',IPR,K2,(OBS(I),I=1,6)
      IF (IPR.EQ.8.OR.NOI1.EQ.NLO)WRITE(17,253)(PRV(I),I=1,IPR)
!     IF (IPR.EQ.8.OR.NOI1.EQ.NLO)WRITE(*,253)(PRV(I),I=1,IPR)
! WRITESTATEMENT NOT USED 2013-01-02.
  253      FORMAT(8F10.4)
! ADDED 2005-11-01.
      IF (LEROUT) THEN
!      NPRED=0
       DO I=0,NLA
        DO IPR=0,NLO
!        NPRED=NPRED+1
         IF (N1.GT.NIPCAT) THEN
! CR IS HERE ONLY USED AS A TRANSFER ITEM. CHANGE 2005-11-01.
```

```
          WRITE(*,*)' N1 EXCEEDS ALLOCATED MEM. ',N1,NIPCAT
          STOP
         END IF
        END DO
       END DO
      END IF
     ELSE
      IF (IKP.EQ.1.OR.IKP.EQ.11) LSMAL=LF
      IF (.NOT.LERNO) THEN
       CALL COUT(NO,LONECO,LSMAL,LF,0)
       IF ((LPUNCH.OR.LWRSOL).AND.LSATP.AND.(.NOT.LZETA).AND.(ISATP.EQ.2.OR.ISAT
P.GT.3))&
       WRITE(17,282)AZP,BETP,TAUP
      END If
! CORRECTION 1995.03.06 BY CCT.
      IF (LMEGR.AND.(.NOT.LMDD)) THEN
       IF (LDEFVP) THEN
        IF (LONECO) THEN
! CORRECTION 2002-04-14.
! DUDY AND DUDX ARE DERIVATIVES OF NORMAL POTENTIAL COMPUTED BY
! TRANS.
         IF (LKSIP) THEN
! CONVERSION FROM ARCSEC TO M/S**2 AND CHANGE OF SIGN.
          DUDY=DUDY-OBS(IA)*CCR(10)/RADSEC
         ELSE
          DUDY=DUDX-OBS(IA)*CCR(10)/RADSEC
         END IF
         IF (.NOT.LNUOUT) WRITE(6,281)DUDY
  281         FORMAT(2F14.10)
        ELSE
         DUDX=DUDX-OBS(IA1)*CCR(10)/RADSEC
         DUDY=DUDY-OBS(IA)*CCR(10)/RADSEC
         IF (.NOT.LNUOUT) WRITE(6,1281)DUDX,DUDY
 1281         FORMAT(/,2F14.10)
        END IF
       ELSE
        IF (LZETA) THEN
! CONVERSION TO POTENTIAL M**2/S**2.
         IF (.NOT.LNUOUT) WRITE(6,358)POT+OBS(IB)*CCR(10)
  358         FORMAT(3E19.11)
! DEACTIVATED 2005-03-30.
!             IF (LPUNCH) WRITE(17,358)POT+OBS(IB)*CCR(10)
        ELSE
! CONVERSION TO M/S**2.
         IF (.NOT.LNUOUT) WRITE(6,281)OBS(IB)*1.0D-5+GP
        END IF
       END IF
      END IF
!
      IF (LPUNCH.AND.LSATP.AND.(.NOT.LZETA).AND.ISATP.EQ.2) WRITE(17,282) AZP,BE
TP,TAUP
      IF (LMAP) IMAP(NO) = OBS(IU)*1000
     END IF
    END DO
   END DO
! ==========================================================
!
   IF (LMAP7) THEN
    WRITE(6,*)' GRID OUTPUT TO UNIT 17 WITH LABEL: '
    WRITE(6,392)RLAMIN,SLAC,SLOC,RLOMAX,GLA,GLO
    WRITE(*,395)HP
 395 FORMAT(' HEIGHT = ',F10.2)
    IF (LMAP7E) WRITE(6,*) ' ERROR-GRID OUTPUT TO UNIT 11 WITH SAME LABEL '
   END IF
   IF (LSTAT) CALL COMPA(VG,IKP,LONECO,IU,3)
   IF (LGRERR.AND.NGRERR.GT.0) WRITE(6,*)' GROSERRORS DETECTED ',&
   NGRERR
! CHANGE 2002-10-08.
   IF ((LGRERS.OR.LGRERR).AND.LCOMP) THEN
```



```
! CHANGE 2009-01-29 TO AVOID DIVISION BY ZERO.
    DO NGRR=1,8
     IF (NGRERR.GT.0) THEN
      SGRE(NGRR)=(D1*NGRE(NGRR))/NGRERR
     ELSE
      SGRE(NGRR)=D0
     END IF
    END DO
    LGRERS=LF
   END IF
!
   IF (LMAP) THEN
    K = NLA+1
    NLAST = 0
    WRITE(6,392)RLAMIN,SLAC,SLOC,RLOMAX,GLA,GLO
  392    FORMAT(6(F11.6,1X))
    DO J = 1, K
     NFIRST = NLAST + 1
     NLAST = NFIRST + NLO
     DO I=NFIRST,NLAST
      IPR=I-NFIRST
      IPR=MOD(IPR,9)+1
      PRV(IPR)=IMAP(I)/1000.0
!     IF (IPR.EQ.9.OR.I.EQ.NLAST) WRITE(6,253)(PRV(JJ),JJ=1,IPR)
! WRITE NOT USED 2013-01-02.
     END DO
    END DO
   END IF
!
   IF (LDEN.AND.LPOT) THEN
! INPUT OF COEFFICIENTS, WHICH HAVE BEEN OVERWRITTEN WHEN LDEN IS TRUE.
    REWIND 3
    READ(3)COFF
    CM3=GMP
    CMM2=AX
    CM1=OMEGA2
   END IF
  ELSE
!
! POINT OR MEAN VALUE CALCULATION.
! *************** INPUT (9) ****************************************
   LMEGR=LF
   CALL DEFDAT(LCOLLO,LPRED,LPARAM,LFOR77,LE,LINTER,LSMAL,LFORM,LP,&
   ICHAR,NMAX,ITRAC0,RRE,NGRERR)
   LSATPP=LSATP
!
   CALL INHEAD(LCOLLO,LREPEC,PW,LNFORM,LADDBP,LINVDE,LADBA,LADDBC)
!
! *************** INPUT (10) ********************************
! RETURN POINT FOR EACH NEW OBSERVATION RECORD,
 2027   CALL INP10(LINTER,LFOR77,LGRID,LFULLO,LTEST,LNOUSE,NWAR,NO2,ITRAC0,OBI,&
   COSB,SINB,COST,SINT)
! CORRECTION 2013-08-06.
   IF (LNOUSE.AND.((.NOT.LPARAM).OR.(LPARAM.AND.MP.EQ.0))) GOTO 2027
   IF (NO.GE.0) THEN
    COSLAP =  COS(RLATP)
    SINLAP =  SIN(RLATP)
    COSLOP = COS(RLONGP)
    SINLOP = SIN(RLONGP)
    IF (LINSOL.AND.LNEWSO) THEN
    PW2=VAR(SM,IS,KP,S,AAI,HP,IMAX1,LMENSI,COSLAP,SINLAP,COSLOP,SINLOP,&
    LSATP,SATROT)
    END IF
!
    IF (LMENSI.AND.(.NOT.LMEAN1)) RLATP=RLATP+STEPN*D2
! SPHERICAL APPROXIMATION 2001-09-21.
    COSLAP =  COS(RLATP)
    SINLAP =  SIN(RLATP)
    IF (LMENSI) THEN
```

```
     IF (LMEAN1) THEN
      CALL PAZIM(RLATP,RLONGP,COSLAP,SINLAP,COSLOP,SINLOP,-CAZP,-SAZP,COST2P,SIN
T2P,LTEST)
     ELSE
      IF (.NOT.LEQANG) CALL ICMEAN(BSIZEN,STEPE,NSTEPE,COSSTE,SINSTE,COSLAP,SINL
AP,LF,LF)
      RLONGP=RLONGP-STEPE*D2
     END IF
    END IF
    COSLOP = COS(RLONGP)
    SINLOP = SIN(RLONGP)
!
    IF (LPARAM.AND.(.NOT.LEQP)) THEN
!
! --------------- INPUT (10A) -------------------------------
! INPUT OF PARAMETER TYPES, IF PARAMETERS ARE GIVEN INDIVIDUALLY
! FOR EACH OBSERVATION. FOR DATA RELATED TO SATELLITE-ALTIMETRY
! WE USE THE REVOLUTION NUMBER(S) WHICH IS IMPLICITLY GIVEN BY THE
! OBSERVATION NUMBER, (CODES IKP=9 FOR CROSS-OVER DIFFERENCES AND
! IKP=11 FOR SEA-SURFACE HEIGHTS TREATED LIKE GEOID UNDULATIONS.)
     CALL INPAR(IKP,NO,ITRACK,IC,N,NOX,NPOBS,NPAOLD,LNOUSE,LINTER,LIN4,LPRED,LDP
R,LWRSOL,LTEST,LONEQ,OBI,IOBS1,NPOINT)
! RETURN TO INPUT (10) IF CROSS-OVER DIFFERENCE COULD NOT BE USED.
     IF (LNOUSE) GO TO 2027
    END IF
!
    IF (LOE1.OR.LOE2) OBS(K2) = OBI(IIE)
    IF (LOUTC) THEN
     IF (LOE1.OR.LOE2) OBS(K2) = OBI(IIE)
     IF(LOE2) OBS(K21) = OBI(IIE1)
     IF (K1.EQ.5) OBS(5)=D0
!
     IF (LREPEC.AND.IOBS2.GT.0) OBS(12) = OBI(IOBS2)
     IF (IOBS1.GT.0) OBS(2) = OBI(IOBS1)
    END IF
!
    IF (IH.EQ.0) THEN
     OBS(1) = HP
     H=HP
     IF (LMEAN.AND.LSIMH.AND.(.NOT.LWRSOL)) OBS(1) = D0
    ELSE
     H=OBI(1)
     OBS(1)=H
! CORRECTION 2003-04-08.
     IF (LMEAN.AND.LSIMH) THEN
      OBS(1) = HP
      H=HP
     END IF
    END IF
!
! CORRECTING THE OBSERVATION BY AN ADDITIVE AND MULTIPLICATIVE
! CONSTANT.
    IF (LADMU) OBS(2)=OBS(2)*DM+DA
! CORRECTION 2010-11-22.
!   IF(.TRUE.) WRITE(*,*)' 3663 O3,OIIE,IIE ',OBS(3),OBI(IIE),IIE
    IF (LADMU.AND.(.NOT.LSA).AND.IIE.GT.0) THEN
     OBI(IIE)=OBI(IIE)*ABS(DM)
!    OBI(IIE)=OBI(IIE)*DM+DA
     IF (.FALSE.) WRITE(*,*)' 3561 OBI ',OBI(IIE)
     IF (LCOMP) OBS(4)=OBI(IIE)
! THIS IS STILL NOT COMPLETELY VERIFIED.
    END IF
    IF (LKM) H = H*1.0D3
! CONVERSION FROM KM TO M.
! CORRECTION 2004-01-26.
    IF (IKP.GT.10.AND.IH.NE.0) HP = H
    IF (LMEAN.AND.LSIMH) H = D0
    IF (LDEN) HP=RRE**2/(RE-HP) - RE
! CONVERSION OF DEPTH TO ARTIFICIAL HEIGHT FOR DENSITY ANOMALIES.
```



```
    IF (LDEN) RP=RE+HP
!
    IF (.NOT.(LNGR.AND.(IORDER.NE.2.OR.LNKSIP).AND.(.NOT.LSATP).AND.(.NOT.LMDD))
) THEN
!
     RLATS=RLATP
     RLONGS=RLONGP
     COSLA=COSLAP
     SINLA=SINLAP
     COSLO=COSLOP
     SINLO=SINLOP
     REF=D0
     DO I=1,NSTEP
      CALL EUCLID(COSLA,SINLA,COSLOP,SINLOP,H,E21,AX1)
      REFI = RGRAV(IPC,IKP,REF1,REF2,REF3,SINLA,H,RG,CU,SU1,&
      LSATP)
      VREF(1)=REF1
      VREF(2)=REF2
      VREF(3)=REF3
!
! CHANGE 1990.10.19 BY CCT CALL OF AXV ADDED .
      IF (LSATP) THEN
       IF (.NOT.LGRADI) THEN
        CALL AXV(SATROT,VREF)
        IF (LGRP) REFI=VREF(3)
        IF (.NOT.LNKSIP) REFI=VREF(2)
        IF (.NOT.LNETAP) REFI=VREF(1)
       ELSE
        CALL ATBA(SATROT,RG,RG)
        IF (LALLCO) THEN
         DO I61=1,3
          DO I62=1,3
           ALLREF(I61,I62)=RG(I61,I62)*1.0D9
           IF (.NOT.LPOT) ALLCOL(I61,I62)=ALLREF(I61,I62)
          END DO
         END DO
        END IF
       END IF
      END IF
      IF (LMENSI) THEN
       IF (LMEAN1) THEN
! FILTER FACTORS INTRODUCED 1992.11.26 BY CCT.
        REF = REF+REFI*FILTER(I)
        CALL PAZIM(RLATS,RLONGS,COSLA,SINLA,COSLO,SINLO,CAZP,SAZP,&
        COSSTN,SINSTN,LTEST)
       ELSE
        REF = REF+REFI
        COSLA1=COSLA
        COSLA=COSLA*COSSTN+SINLA*SINSTN
        SINLA=SINLA*COSSTN-COSLA1*SINSTN
       END IF
      ELSE
       REF = REF+REFI
      END IF
     END DO
     REF=REF/NSTEP
! COMPUTING THE REFERENCE VALUES.
     IF (LMEGR.AND.(LGRP.OR.(.NOT.LNKSIP).OR.(.NOT.LNETAP)).AND.(.NOT.LMDD)) THE
N
! WE SUPPOSE THE FIRST DERIVATIVES ARE IN MGAL WHEN LMEGR IS TRUE.
      OBS(2) = OBS(2)-REF*1.0D5
     END IF
     IF (LMDD.AND.(.NOT.LSATP)) THEN
      OBS(2) = OBS(2)-REF*1.0D9
      IF (.FALSE.) WRITE(*,*)' OB2,REF ',OBS(2),REF
     END IF
     REF0=REF
! ADDED 2008-09-25 and changed 2012-04-19.
     LWAIT=LERNO.AND.(.NOT.LNCOL)
```

```
! ADDED 2006-08-20.
     IF (LALLCO.AND.LCOMP) THEN
      IF (IORDER.EQ.1) THEN
       ALLIN(1,1)=OBI(IOBS1)
      ELSE
       IF (IORDER.EQ.1.AND.IOBS1.GT.5) THEN
        ALLIN(1,1)=OBI(2)
        ALLIN(1,2)=OBI(3)
        ALLIN(1,3)=OBI(4)
       ELSE
! WE HERE SUPPOSE THAT GRAVITY GRADIENTS ARE IN THE ORDER XX,YY,ZZ,
! XY,XZ,YZ.
        ALLIN(1,1)=OBI(2)
        ALLIN(2,2)=OBI(3)
        ALLIN(3,3)=OBI(4)
        ALLIN(1,2)=OBI(5)
        ALLIN(1,3)=OBI(6)
        ALLIN(2,3)=OBI(7)
        ALLIN(3,2)=ALLIN(2,3)
        ALLIN(2,1)=ALLIN(1,2)
        ALLIN(3,1)=ALLIN(1,3)
       END IF
      END IF
     END IF
    END IF
!
    IF (LWLONG.AND.LDEFVP.AND.LREPEC) OBS(12) = - OBS(12)
    IF (LWLONG.AND.LONECO.AND.(.NOT.LNETAP)) OBS(2) = -OBS(2)
!
    OBS(IB) = D0
    POT=D0
    GP=D0
    DUDX=D0
    DUDY=D0
    IF (LREPEC) OBS(IB1) = D0
    IF (LTERRC) THEN
!
     OBS(ITE)=OBI(IITE)
     IF (LADBTE) OBS(IB)=OBS(ITE)
     IF (LREPEC) THEN
      OBS(ITE1)=OBI(IITE1)
      IF (LADBTE) OBS(IB1)=OBS(ITE1)
     END IF
    END IF
!
    IF (LTRAN.OR.LPOT) THEN
     CALL TRAPOT(LNPOT,LREPEC,LSPHER,LADDBP,LFULLO,POT00,RB,REF,REF0,UREF0,OBI,H
,HPP,RRE,SU,SU8,VREF)
    ELSE
! CHANGE 2004-07-09.
     IF (.NOT.LSPHER) THEN
      CALL EUCLID(COSLAP,SINLAP,COSLOP,SINLOP,HP,E22,AX2)
! NO SPHERICAL APPROXIMATION, 2001-09-21.
! CHANGE 2004-08-11.
      IF (.NOT.LSATP.AND.(.NOT.LZETA)) ISATP=0
!     IF (.NOT.LSATP.AND.(.NOT.LZETA)) ISATP=3
! THIS ASSIGNMENT INDICATES FULL ROTATION, USING THE UNIT MATRIX.
      IF (DISTO.LT.RB) THEN
       WRITE(*,*)' POINT INSIDE BJERHAMMAR SPHERE, H::= 10000 M '
       WRITE(*,*)HP,DISTO,RB
       HPP=0.0D0
      ELSE
       HPP=DISTO-RE
! CHANGE 2003-06-02.
       IF (IH.NE.0) HP=HPP
      END IF
!
      COSLAP=XY/DISTO
      SINLAP=Z/DISTO
```



```
      RLATP1=ATAN2(Z,XY)
! DLATP IS GEOCENTRIC LATITUDE MINUS GEODETIC LATITUDE. MORE CORRECT
! IS THE ANGLE BETWEEN THE NORMAL GRAVITY FIELD VECTOR, SEE RGRAV.
      DLATP=RLATP1-RLATP
      IF (ABS(DLATP).GT.0.1) THEN
       WRITE(*,*)' ERROR, RLATP,P1 = ',RLATP,RLATP1
      ELSE
! CORRECTION 2003-04-06.
       RLATP=RLATP1
       IF (IANG.EQ.6) SLAT=RLATP*180.0D0/PI
      END IF
     ELSE
      HPP=HP
     END IF
    END IF
!
    IF (LCREF) THEN
!     write(*,*)' 3740 SRAAR,IOBSR,NIR,IMAX1R ',SR,AAR,IOBSR,NIR,IMAX1R
     CALL PRED(SR,AAR,0 ,    0  ,2 ,IOBSR,NIR,IMAX1R,LT   ,LF  ,LF  ,LTCOV,&
     LSATAC,LWAIT,NPRED)
!
     OBS(IC1) = PREDP
     IF (LADDBC) OBS(IB) = OBS(IB)+OBS(IC1)
!
     IF (LREPEC) THEN
      OBS(IC11) = PRETAP
      IF (LADDBC) OBS(IB1)= OBS(IB1)+OBS(IC11)
     END IF
!      ELSE
!       LSTOP=LT
    END IF
!
    IF (LTNB) OBS(IU) = OBS(IB)-OBS(IT)
    IF (LTEB) OBS(IU) =-OBS(IT)
    IF (LK30) OBS(3) = OBS(2)-OBS(IU)
    IF (LK30) OB1 = OBS(3)
    IF (.NOT.LK30) OB1 = OBS(2)
!
    IF (LCOLLO) THEN
     NPRED=NPRED+1
     IF (NPRED.GE.1) THEN
! NI IS THE SUBSCRIPT OF THE FIRST ELEMENT IN A COLUMN IN C.
      IF (MOD(NPRED,IBSS).EQ.1) THEN
       NI=1
      ELSE
       NI=N1*((NPRED-1)-INT((NPRED-1)/IBSS)*IBSS)+1
      END IF
     END IF
     LWAIT=LERNO.AND.(.NOT.LNCOL)
     IF (LMTEST) write(*,*)' 3620 NPRED,NI ',NPRED,NI
! added 2012-04-19.
     CALL COPRED(PREDCO,PW2,OBI,WM,SM,&
     KP,NPARM,                                NPRED,NPRED1,&
     LERNO,       LREPEC,LTCOV,LSATAC,LADBA,LTNB,LTEB,LCOMP,&
     LFOUND,LCOD,LMENSI,LSATP,LGRERS,LGRERR,LSA,LERCOV,   &
     LWAIT,0,       LMTEST)
!    KP,NPARM,NFILE,NPARM1,NERRM,NGRE,NGRERR,NPRED,NPRED1,&
!    LERNO,LINSOL,LREPEC,LTCOV,LSATAC,LADBA,LTNB,LTEB,LCOMP,&
!    LFOUND,LCOD,LMENSI,LSATP,LGRERS,LGRERR,LSA,LERCOV,LF,&
!    LWAIT,0,LGRID,LMTEST)
     NI0=NI
    ELSE
     IF (LREPEC) THEN
      OBS(IA1) = PRETAP
      IF (LADBA) OBS(IB1) = OBS(IB1)+OBS(IA1)
      IF (LTNB) OBS(IU1) = OBS(IB1)-OBS(IT1)
      IF (LTEB) OBS(IU1) = -OBS(IT1)
      IF (LCOMP) OBS(13) = OBS(12)-OBS(IU1)
     END IF
```

```
!
     OBS(IA) = PREDP
     IF (LADBA) OBS(IB) = OBS(IB)+OBS(IA)
     IF (LTNB) OBS(IU) = OBS(IB)-OBS(IT)
     IF (LTEB) OBS(IU) =-OBS(IT)
     IF (LCOMP) OBS(3) = OBS(2)-OBS(IU)
    END IF
!   write(*,*)' 3884 PR,OBS2345,IU,IA,IB ',PREDP,OBS(2),OBS(3),OBS(4),OBS(5),IU,
IA,IB
!
! CHANGE 2012-05-17.
    IF (LERNO.AND.(.NOT.LNCOL)) THEN
     IF (LGRID) NO=NPRED
!    WRITE(*,*)' LREPEC ',LREPEC
     IF (.NOT.LREPEC) THEN
      WRITE(20)PREDCO,OBS,NO,CLATD,SLAT,SLON,IDLAT,IDLON,MLAT,MLON
!     write(*,*)' out to UNIT 20 ',npred,PREDCO
!     WRITE(20,REC=NPRED)PREDCO,OBS,NO
     ELSE
       WRITE(20)PREDCO,OBS,NO,CLATD,SLAT,SLON,IDLAT,IDLON,MLAT,MLON
!     WRITE(20)PREDCO,OBS,NO
!     WRITE(20,REC=INT(NPRED/2))PREDCO,OBS,NO
!     write(*,*)' stored on unit 20, rec= ',npred/2,no,npred
!     write(*,7979)PREDCO,OBS,NO
     END IF
    ELSE
      IF (LSTAT) CALL COMPA(VG,IKP,LONECO,IU,2)
      CALL COUT(NO,LONECO,LSMAL,LFULLO.AND.LNCOL.AND.(.NOT.LCOMP),&
      IORDER)
      IF ((LPUNCH.OR.LWRSOL).AND.LSATP.AND.(.NOT.LZETA).AND.(ISATP.EQ.2.OR.ISATP
.GT.3))&
      WRITE(17,282)AZP,BETP,TAUP
    END IF
!
     IF (LMEGR.AND.(.NOT.LMDD)) THEN
      IF (LDEFVP) THEN
       IF (LONECO) THEN
! CORRECTION 2002-04-14.
        IF (LKSIP) THEN
! CONVERSION FROM ARCSEC TO M/S**2 AND CHANGE OF SIGN.
         DUDY=DUDY-OBS(IA)*CCR(10)/RADSEC
        ELSE
         DUDY=DUDX-OBS(IA)*CCR(10)/RADSEC
        END IF
        IF (.NOT.LNUOUT) WRITE(6,281)DUDY
       ELSE
        DUDX=DUDX-OBS(IA1)*CCR(10)/RADSEC
        DUDY=DUDY-OBS(IA)*CCR(10)/RADSEC
       IF (.NOT.LNUOUT) WRITE(6,1281)DUDX,DUDY
      END IF
     ELSE
      IF (LZETA) THEN
! CONVERSION TO POTENTIAL M**2/S**2.
       IF (.NOT.LNUOUT) WRITE(6,358)POT+OBS(IB)*CCR(10)
! DEACTIVATED 2005-03-30.
!           IF (LPUNCH) WRITE(17,358)POT+OBS(IB)*CCR(10)
      ELSE
! CONVERSION TO M/S**2.
       IF (.NOT.LNUOUT) WRITE(6,281)OBS(IB)*1.0D-5+GP
      END IF
     END IF
    END IF
! ADDED 2002-11-25 AND REMOVED 2014-01-12.
!   IF (LPUNCH.AND.LSATP.AND.(.NOT.LZETA).AND.ISATP.EQ.2) WRITE(17,282)AZP,BETP,
TAUP,3988
!
   ELSE
    LSTOP=LT
   END IF
```



```
   IF (.NOT.LSTOP) GO TO 2027
!
!  END IF
  END IF
 END IF
!
! *************** INPUT (18) ********************************
!
 IF (LINTER) WRITE(6,*)' STOP ?'
 READ(5,*)LSTOP
 IF (LPRED) LRESOL = LF
 IF (LWAIT.AND.LPRED.AND.LERNO.AND.(.NOT.LSPHAR)) then
  IF (NPRED.EQ.IBSS.OR.MOD(NPRED,IBSS).NE.0) THEN
   if (LMTEST) write(*,*)' RESTORE_CH: ',INT(NPRED/IBSS)*IBSS+1,N1-1
   CALL RESTORE_CH(INT(NPRED/IBSS)*IBSS+1,N1-1,LF,LT,LERCOV,LMTEST)
  END IF
  write(*,*)' nes 23',NPRED,LSTOP,N1-1,NPRED,ISO,NPARM1
!bso cholsol2
  call cholsol(N1-1,0,NPARM1-1,NPRED,lf,lf,lt,lf,lf)
   print*,'BSO CHOLSOL2'
! call cholsol(N1-1,0,NPARM1-1,NPRED,lf,lf,lt,lf,lf)
! write(*,*)' ERROR ESTIMATES ,IIE,IIE1,WM ',IIE,IIE1,WM
  if (LMTEST) write(*,*)' LREPEC ',LREPEC,NPRED
  IF (.NOT.LOPEN20) THEN
  INQUIRE(99,OPENED=LOPEN,EXIST=LEXIST,POS=IPOS)
  WRITE(*,*)' UNIT 99 AT POS ',IPOS
  WRITE(*,*)' 3886 UNIT 99 OPENED ',LOPEN,LEXIST
  IF (.NOT.LOPEN) THEN
   OPEN(99,FILE=DNANE(1,99),FORM='UNFORMATTED',&
   STATUS='OLD',POSITION='REWIND')
   INQUIRE(99,OPENED=LOPEN,EXIST=LEXIST,POS=IPOS)
   WRITE(*,*)' UNIT 99 AT POS ',IPOS
   WRITE(*,*)' 3889 UNIT 99 OPENED ',LOPEN,LEXIST
  ELSE
   REWIND(99)
   WRITE(*,*)' UNIT 99 REWIND '
  END IF
! REWIND(99)
! EXPERIMENTAL CHANGE 2012-08-09.
! close(99)
  INQUIRE(20,OPENED=LOPEN,EXIST=LEXIST)
  IF (.NOT.(LEXIST.and.LOPEN)) THEN
   OPEN(20,FILE=DNANE(1,20),FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
   WRITE(*,*)' 3851 UNIT 20 OPENED ',LOPEN,LEXIST
  ELSE
   IF (.NOT.LOPEN.AND.LEXIST) OPEN(20,FILE=DNANE(1,20),FORM='UNFORMATTED',&
   STATUS='OLD')
   rewind(20)
   write(*,*)' UNIT 20 exist and is open '
  END IF
  ELSE
   rewind(99)
   rewind(20)
   write(*,*)' 3815  rewind of unit 20 and 99 '
  END IF
  do i=1,NPRED
   READ(99)OERR
   IF (OERR.GT.0.0d0) OERR=SQRT(OERR)
   CCCIJ(I)=OERR
! THE RECORD IS CREATED IN SUBROUTINE COPRED 2012-05-12.
   IF (.NOT.LREPEC) READ(20,IOSTAT=K)PREDCO,OBS,NO,CLATD,SLAT,SLON,IDLAT,IDLON,M
LAT,MLON
!  write(*,*)' after read(20),i,predco ',i,predco
!  IF (.NOT.LREPEC) READ(20,REC=I,IOSTAT=K)PREDCO,OBS,NO
   IF (LREPEC.AND.(MOD(I,2).EQ.1))THEN
!   write(*,*)' 3742 reading unit 20, rec = ',(i+1)/2,i
     READ(20,IOSTAT=K)PREDCO,OBS,NO,CLATD,SLAT,SLON,IDLAT,IDLON,MLAT,MLON
!    READ(20,REC=INT((I+1)/2))PREDCO,OBS,NO
!    READ(20,REC=INT((I+1)/2),IOSTAT=K)PREDCO,OBS,NO
```

```
!    write(*,*)' after READ(20), PREDCO,OBS ',PREDCO,OBS
     if (k.ne.0) write(*,*)' iostat= ',k
!    write(*,*)' 4065 NO,i,k2,oerr ',NO,i,k2,oerr
   end if
!  if (k.ne.0) write(*,*)' iostat= ',k
   IF (LREPEC.AND.(MOD(I,2).EQ.0)) THEN
    OBS(K2)=OERR1
    OBS(K2+10)=OERR
!   write(*,*)' 4072 i,K2,OERR,OERR1 ',i,k2,OBS(K2),OBS(K2+10)
   ELSE
    OBS(K2)=OERR
    OERR1=OERR
   END IF
!  if (LTCOV) write(*,*)' 3975 LGRERR,LGRERS ',LGRERR,LGRERS
   IF (LGRERR.AND.LGRERS) THEN
!  IF (LGRERR.OR.LGRERS) THEN
    IF (LREPEC.AND.(MOD(I,2).EQ.0)) THEN
     IF (LSA) OBI(IIE)=WM
!    WRITE(*,*)' WARNING:  NOT YET FULLY IMPLEMENTED '
     IF (LSA) OBI(IIE1)=ABS(WM)
     OBS(14)= SQRT(OBS(14)**2+OBI(IIE1)**2)
     IF (LGRERR) THEN
      LFOUND= ABS(OBS(13)).GT.REJLEV*OBS(14)
     END IF
     INDG=ABS(OBS(13)*2)/OBS(14)+1
     IF (INDG.LT.1.OR.INDG.GT.8) INDG=8
     NGRE(INDG)=NGRE(INDG)+1
     NGRERR=NGRERR+1
    ELSE
     IF (LSA) OBI(IIE)=WM
!    if (LTCOV)  write(*,*)' 3990 O4,OIIE,IIE,LSA,LGRERR ',&
!    OBS(4),OBI(IIE),IIE,LSA,LGRERR
     OBS(4)= SQRT(OBS(4)**2+OBI(IIE)**2)
     IF (LGRERR) THEN
      LFOUND= ABS(OBS(3)).GT.REJLEV*OBS(4)
     ELSE
      NGRERR=NGRERR+1
     END IF
     INDG=ABS(OBS(3)*2)/OBS(4)+1
    END IF
    IF (INDG.GT.8.OR.INDG.LT.1) INDG=8
    NGRE(INDG)=NGRE(INDG)+1
    IF (LFOUND) THEN
! OUTPUT OF DETECTED GROSS-ERRORS TO UNIT 24.
     RLATP=PREDCO(1)
     RLONGP=PREDCO(4)
     RLATP=RLATP*180.0/PI
     RLONGP=RLONGP*180.0/PI
     IF (LONECO) WRITE(12,712)NO,RLATP,RLONGP,OBS(1),OBS(2),&
     OBI(IIE),OBS(3),OBS(4)
     IF (.NOT.LONECO)WRITE(12,713)NO,RLATP,RLONGP,OBS(1),OBS(2),&
     OBS(12),OBI(IIE),OBI(IIE1),OBS(3),OBS(4),OBS(13),OBS(14)
 712   FORMAT(I11,F10.5,F11.5,F8.1,4F10.4)
 713   FORMAT(I11,F10.5,F11.5,F8.1,4F10.4,/,4F10.4)
     NGRERR=NGRERR+1
    END IF
   END IF
   IF (LSTAT) CALL COMPA(VG,IKP,LONECO,IU,2)
! THIS SEEMS NOT TO WORK 2012-05-12. check 2012-10-07.
!  IF (.NOT.LGRID.or.(.true.))  THEN
   IF (.NOT.LMAP)  THEN
! CHANGE 2013-09-28.
!  IF (.NOT.LGRID)  THEN
! ONLY WRITE FOR LGRID FALSE 2013-01-03.
    IF (LREPEC.AND.MOD(I,2).EQ.0) THEN
     CALL COUT(NO,LONECO,LSMAL,LFULLO.AND.LNCOL.AND.(.NOT.LCOMP),&
     IORDER)
    END IF
    IF (.NOT.LREPEC) CALL COUT(NO,LONECO,LSMAL,LFULLO.AND.LNCOL.AND.(.NOT.LCOMP)
```



```
,&
    IORDER)
   END IF
  end do
! CLOSE(20)
  IF (LOPEN20) REWIND(UNIT=20)
  IF (LGRID.AND.LMAP7E) then
   write(*,*)' ERROR ESTIMATES '
   INQUIRE(11,OPENED=LOPEN)
   WRITE(*,*)' UNIT 11 OPEN= ',LOPEN
!  IF (.NOT.LOPEN) OPEN(11,FILE=DNAME(1),FORM='FORMATTED',POSITION='APPEND')
! change due to new IFORT compiler.
   IF (.NOT.LOPEN) OPEN(11,FILE=DNAME(1),FORM='FORMATTED')
!  WRITE(UNIT=11,3073)(CCCIJ(i),i=1,NPRED)

   write(11,'(5D14.5)')(CCCIJ(i),i=1,NPRED)
!  write(*,'(5D14.5)')(CCCIJ(i),i=1,NPRED)
! WRITE NOT USED 2013-01-02.
 3073 format(5d14.5)
   write(*,*)' unit 11 closed '
  end if
! IF (LSTAT) CALL COMPA(VG,IKP,LONECO,IU,3)
! STATEMENT MOVED OUTSIDE IF-CLAUSE 2012-07-25.
  IF (LGRERR.AND.NGRERR.GT.0) WRITE(6,*)' GROSERRORS DETECTED ',&
   NGRERR
! CHANGE 2002-10-08.
  IF ((LGRERS.OR.LGRERR).AND.LCOMP) THEN
   DO NGRR=1,8
    IF (NGRERR.GT.0) THEN
     SGRE(NGRR)=(D1*NGRE(NGRR))/NGRERR
    ELSE
     SGRE(NGRR)=D0
    END IF
   END DO
  END IF
 end if
 IF (LSTAT.AND.(.NOT.LSPHAR)) CALL COMPA(VG,IKP,LONECO,IU,3)
   IF ((LGRERS.OR.LGRERR).AND.LCOMP.AND.LSTAT) THEN
    WRITE(*,3072)(NGRE(IGRR),IGRR=1,8),(SGRE(IGRR),IGRR=1,8)
 3072    FORMAT(' HISTOGRAM RATIO ABS(ERROR)/ERROR ESTIMATE IN 0.5 INTERVALS .',
/,8I8,/8F8.4/)
    LGRERS=LF
   END IF
!
!CLOSE(20)
 IF (LERCOV) THEN
  CLOSE(7)
  CLOSE(19)
 END IF
 IF (LEROUT) THEN
  CLOSE(19)
  LEROUT=LF
 END IF
 IF (.NOT.LSTOP) GO TO 2000
! IF LSTOP IS TRUE, THE EXECUTION WILL FINISH.
!
!IF (LMAP7E) CLOSE(11)
 CLOSE(3)
 CLOSE(8)
 CLOSE(13)
 IF (LGRERR) CLOSE(12)
 IF (LOPEN4) CLOSE(INZ)
 IF (LOPEN7.OR.(LWRSOL.AND. (.NOT.LCLU7))) THEN
  CLOSE(17)
 END IF
!
 9997 CONTINUE
 IF (LTIME) THEN
  CPU5=SYTIME(RCBASE,TIMEARRAY)
```

```
! WRITE(6,7470)TIMEARRAY(1),CPU5
! WRITE(*,7471)RCBASE
 7471  FORMAT(' TOTAL CPU TIME USED= ',F15.5,' SEC ')
 END IF
 IF (NWAR.GT.0) WRITE(*,1391)NWAR
 1391 FORMAT(' NUMBER OF WARNINGS ',I8)
 IF (NERRM.GT.0) WRITE(*,1392)NWAR
 1392 FORMAT(' NUMBER OF ERROR MESSAGES FROM NES ',I8)
 IF (LPARAM.AND.LTEST) WRITE(*,*)' IPAMAX = ',IPAMAX
 WRITE(6,*)' GEOCOL TERMINATED AT:'
 CALL FDATE(UDATE)
 WRITE(6,*)UDATE
 call timer('Total',2)
 call print_times
 close(91)
 call system('rm runfile ')
! 2012-12-17 this is to be used to check the correct termination of the
! program.
 9999 CONTINUE

!bso beg 22.11.2012
#ifdef _MPI
  call MPI_finish
#endif
!bso end 22.11.2012

END PROGRAM

! %%%%%%%%%%%%%%%%%%%% SUBROUTINE SECTION %%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE DEFDAT(LCOLLO,LPRED,LPARAM,LFOR77,LE,LINTER,LSMAL,LFORM,LP,ICHAR,NMAX
,ITRAC0,RRE,NGRERR)
! THIS MODULE DEFINES POINT OR MEAN VALUE DATA RECORDS.
! EXTRACTED FROM GEOCOL16 VER. 11, 2004-11-18. LAST CHANGE:
! 2008-10-02.
! FOR VARIABLE DESCRIPTIONS SEE MAIN PROGRAM.

 USE m_params,       ONLY : MAXO,NSAT,MXPAR,MAXCX,NMAP,NIPT,NIPCAT,INBLP,NSPHAR,M
AXSA
 USE m_params,       ONLY : NDIMC,NISIZE,NCRW,NNBL
 USE m_params,       ONLY : NCOEFF,NROOT
 USE m_geocol_data, ONLY : SATROT
 USE m_geocol_data, ONLY : FILTER,NFILTE
 USE m_geocol_data, ONLY : NFOUR,FOUCOF,LFOUR
 USE m_geocol_data, ONLY : RDD

 USE m_geocol_data, ONLY : OLDN,ITMODE,ITOLD,LCTIME,ITM0,ITMOD,ITRGAP,JR0
 USE m_geocol_data, ONLY : ALLREF,ALLGG,ALLCOL,ALLPRE,ALLTRA,ALLPR1,ALLERR,&
                           ALLVAR,ALLIN,LALLCO
 USE m_data,         ONLY : LWLONG,LIN4,NO,LCOMP,LERNO,LOPEN4,LOPEN7,RLAMIN,&
                           RLAMAX,RLOMIN,RLOMAX,LBIPOT,LINSOL,LDENOL,INZOLD,&
                           LCOERR,LLCOER,LNEWD,LGRID,LRESOL

 USE m_data,         ONLY : BSIZE,ANDEX,PRETAP,PREDP,HCZERO,NCZERO

 USE m_data,         ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN

 USE m_geocol_data, ONLY : B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,  &
                           WOBS,ISAT,SINLOP,COSLOP,BSIZEN,BSIZEE,COSLAP, &
                           SINLAP,RLONGP,RP,CAZP,SAZP,HP,RLATP,ICZERO,   &
                           NI,NR,IKP,ISATP,NOBLK,LONECO,LNKSIP,LNETAP,   &
                           LDEFVP,LOBSST,LSTART
 USE m_geocol_data, ONLY : LSATPP
 USE m_data,         ONLY : IPTYPE,ITIME,ITIME0,NPARM,NPARM1
 USE m_geocol_data, ONLY : SFACT,FPERIO,IPACAT,ITROLD,ICODE,MAXPAR,MP,&
                           IPA,NCXLAS,DXX,NUM,VARI,SCALE,SCALE2,INN,INV
 USE m_geocol_data, ONLY : INUMR,NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,&
                           LSTNO=>LTERM,LTERMA,LTERMO,LOUTC,LTRAN,LNERNO,&
                           LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT
```



```
 USE m_data,         ONLY : 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
 USE m_geocol_data, ONLY : VARNO,PPA,PPS,VG,HPK,DM,DA,WM,REJLEV,SGRE,ROTFIL,&
                            ERNAME,DNAME,FMT,NSTEP,NSTEPE,IDSAT,ILA,ILO,IKPREF,&
                            INZ,IO2,NPOBS0,NOUSE,ILAST,IPAMAX,NGR,NGRE,ICSYS, &
                            LMEAN,LSA,LADMU,LSTAT,LAREA,LZETA,LGRADI,LMENSI,&
                            LSIMH,LMEAN1,LINTRA,LGIGRS,LMEGR,LUSNGS,LSATAC,&
                            LSATP,LGRERR,LCOD,LEQP,LALLP,LGRP,LDEN,LPOTSD,&
                            LEQANG,LFILTE,LBIN,LSKIPL,LGRERS,LTILT,LSCALE,LINERT
 USE m_geocol_data, ONLY : LFOUR,SCFACT,SCFRDD,RDD


IMPLICIT NONE

 INTEGER  ::      ICHAR,IJ,NGRERR,NMAX,NPNO,  &
                  I,ITRAC0, JR1,JR2,JR3,MKP

 REAL(KIND=8) ::  COZERO,RRE

 LOGICAL ::       LFORM,LFOR77, LE,  LP(2), &
                  LPARAM,LNCOL,LCOLLO, &
                  LPRED,LSMAL,LINTER,LOK

 LNCOL=.NOT.LCOLLO
 LSTOP = LF
 NSTEP=1
 NSTEPE=1
 NO1=0
 BSIZEN=D0
 LSIMH=LT
 LMENSI=LF
 LMEAN1=LF
 LTILT=LF
 LSCALE=LF
!
! *************** INPUT (9) **********************************
!
! INPUT OF ONE DATA-SET OF OBSERVATIONS OR COORDINATES OF PREDICTION
! POINTS. ALL RECORDS MUST BE PUNCHED IN THE SAME WAY. THERE ARE THE
! FOLLOWING RESTRICTIONS AND OPTIONS: A STATION NUMBER MAY BE USED, BUT
! IT MUST OCCUPY THE FIRST DATAFIELD ON THE RECORD. THE TWO NEXT DATA-
! FIELDS MUST CONTAIN THE GEODETIC LATITUDE AND LONGITUDE ( IN AN ARBI-
! TRARY ORDER). IN CASE THE HEIGHT IS GIVEN, IT MUST BE PUNCHED IN THE
! NEXT DATAFIELD. THE FOLLOWING UP TO EIGHT DATAFIELDS WILL HAVE TO
! CONTAIN THE OBSERVED QUANTITY (OR QUANTITIES WHEN A PAIR OF DEFLECTI-
! ONS ARE OBSERVED) AND CONTINGENTLY THE STANDARD DEVIATIONS (WHEN
! LSA IS FALSE). ALSO PRECOMPUTED POTENTIAL COEFFICIENT AND TERRAIN
! POTENTIAL CONTRIBUTIONS MAY BE INPUT. THE LAST DATAFIELD MAY
! HOLD THE VALUE OF A LOGICAL VARIABLE LSTOP, TRUE FOR THE LAST
! RECORD IN THE FILE AND FALSE (I.E. BLANK ON MOST COMPUTERS) OTHERWISE.
! THIS IS ONLY USED IF NO STATION NUMBER IS INPUT. OR IF FREE-FORMAT
! INPUT IS USED FROM UNIT 5 (LIN4=.FALSE., SEE BELOW).
! IF A STATION NUMBER IS USED, THE END OF THE DATASET IS
! SUPPOSED TO HAVE BEEN REACHED, IF A NEGATIVE STATION NUMBER IS READ.
!
! ----------------------- INPUT (9) ------------------------------
!
! INPUT OF VARIABLES SPECIFYING THE CONTENT OF THE RECORDS. INO = 1,
! WHEN THE STATION NUMBER IS PUNCHED, 0 OTHERWISE, ILA, ILO THE NUMBER
! OF THE DATAFIELDS OCCUPIED BY THE LATITUDE AND THE LONGITUDE RESPEC-
! TIVELY, IANG  SPECIFYING UNITS OF ANGLES (1 FOR DEGREES, MINUTES, ARC-
! SECONDS, 2 FOR DEGREES, MINUTES, 3 FOR DEGREES AND 4 FOR 400-GRADES,
! 5 CARTESIAN COORD.
! IH = THE NUMBER OF THE DATAFIELD HOLDING THE HEIGHT (ZERO WHEN NO
! HEIGHT IS CONTAINED), IOBS1, IOBS2 = THE DATAFIELD NUMBER OF THE FIRST
! AND THE SECOND OBSERVATION, RESPECTIVELY (ZERO WHEN NO FIRST OR SECOND
! OBSERVATION), PLUS 10 TIMES THE RELATIVE POSITION OF A PRECOMPUTED
! POTENTIAL COEFFICIENT CONTRIBUTION, PLUS 100 TIMES THE RELATIVE
```

```
! POSITION OF A CONTRIBUTION FROM A TERRAIN POTENTIAL. EXAMPLE:
! SUPPOSE A RECORD CONSIST OF STATION NUMBER, LATITUDE, LONGITUDE,
! HEIGHT, KSI, STDV(KSI), PRECOMPUTED POTENTIAL COEFFICIENT CONTRI-
! BUTION AND PRECOMPUTED TERRAIN CONTRIBUTION, FOLLOWED BY THE
! SAME QUANTITIES FOR ETA. THEN INO=1,ILA=2,ILO=2,IH=4,IOBS1=435,
! IOBS2=438.
! NOTE, THAT IF STANDARD DEVIATIONS ARE PRESENT IN RECORD, THEY MUST
! FOLLOW JUST AFTER THE OBSERVATIONS. IF ONE OBSERVATION COMES RIGHT
! AFTER THE OTHER (IOBS2-IOBS1=1) THEN TWO STANDARD DEVIATIONS
! MUST FOLLOW THE SECOND OBSERVATION.
! IF IOBS1=0 AND PRECOMPUTED POTENTIAL COEFFICIENT OR TERRAIN
! CONTRIBUTIONS MUST BE INPUT, THEN THE POSITIONS OF THESE VALUES
! MUST BE GIVEN RELATIVE TO THE HEIGHT (IF PRESNENT) OR THE
! LONGITUDE/LATITUDE, DEPENDING ON WHICH ONE COMES LAST.
! THEN IKP, SPECIFYING THE KIND OF OBSERVATION, (1 FOR ZETA, 2
! FOR MEASURED GRAVITY, POINT OR MEAN GRAVITY ANOMALIES, 3 FOR KSI, 4
! FOR ETA AND 5 FOR PAIR OF DEFLECTIONS (KSI,ETA) OR (ETA,KSI),(IN THE
! SAME ORDER AS THE LATITUDE AND THE LONGITUDE)). ALSO THE CODES USED
! IN COVAX CAN BE USED, WITH 10 ADDED. IF CODES IKP EQUAL TO
! 26, 28, 30 AND 35 ARE USED, PAIRS OF QUANTITIES (16,17), (18,19),
! (20,21), (25,23) ARE COMPUTED.
!
! IN CASE LPARAM IS TRUE, OBSERVATIONS OF THE DIFFERENCE BETWEEN THE
! LOCAL GEODETIC AND THE GEOCENTRIC ELLIPSOIDAL HEIGHT (IKP = 6),
! OBSERVATIONS OF PAIRS OF DIFFERENCES BETWEEN THE GEOCENTRIC AND
! THE LOCAL GEODETIC LATITUDE AND LONGITUDE*COS(LATITUDE) (IKP = 7)
! CF. REF.(E), EQ.(12) - (14), AND SATELLITE ALTIMETER CROSS-OVER
! DIFFERENCES (IKP = 9) ARE ACCEPTED. IF A NEGATIVE VALUE OF IKP IS
! IS USED, THEN THE INPUT VALUE IS EQUAL TO THE OBSERVATION MINUS
! CONTINGENT CONTRIBUITIONS FROM POTENTIAL COEFFICIENTS, DATUM
! SHIFT OR COLLOCATION STEP 1. THIS IS USED, FOR EXAMPLE WHEN
! A RESTART-FILE IS INPUT.
!
! THEN ICSYS, AN INTEGER DEFINING THE COORDINATE SYSTEM. IF IT IS
! .LT. 0 THEN IT IS THE GEOCENTRIC (BEST) SYSTEM.
!
! THEN HP, THE MEAN HEIGHT OF THE POINTS (USED WHEN NO INDIVIDUAL
! HEIGHTS ARE GIVEN ), THE VALUES OF 10 LOGICAL VARIABLES:
!  LPUNCH  = PRINT OBS. OR PREDICTED VALUE AND CONTINGENTLY THEIR
!     DIFFERENCE, (OUTPUT TO UNIT 17).
!  LWLONG = LONGITUDE (AND ETA) ARE POSITIVE TOWARDS WEST.
!  LMEAN  = THE PREDICTED OR OBSERVED QUANTITY IS A MEAN GRAVITY VALUE.
!  LSA    = ALL OBSERVED QUANTITIES HAVE THE SAME STANDARD DEVIATION.
!     IN THIS CASE MUST THE COMMON VALUE (WM) BE INPUT SUBSEQUENTLY.
!  LKM    = THE HEIGHT IS IN UNITS OF KILOMETERS.
!  LADMU  = THE OBSERVATION HAS TO BE CORRECTED USING AN ADDITIVE
!           AND A MULTIPLICATIVE CONSTANT, AND IS CONTINGENTLY
!           AN ABSOLUTE VALUE (ONLY USABLE FOR GRAVITY AND
!           AND GRAVITY GRADIENTS PRESENTLY).
!  LSTAT  = STATISTICS OF DIFFERENCES BETWEEN OBSERVED AND PREDICTED
!     QUANTITIES ARE TO BE OUTPUT. THIS INCLUDES A PRIMITIVE HISTOGRAM
!    WITH 21 BINS. THE SIZE OF THE SAMPLING INTERVAL (VG) MUST BE INPUT
!    SUBSEQUENTLY.
!  LAREA  = ONLY DATA WITHIN GIVEN AREA ARE TO BE USED. LATITUDE AND
!    LONGITUDE BOUNDARIES MUST BE INPUT SUBSEQUENTLY.
!  LFORM  = VARIABLE FORMAT IS USED FOR DATA. FORMAT MUST BE INPUT
!    SUBSEQUENTLY, E.G. AS (I6,2F8.3,3F7.2,L2).
!  LIN4   = DATA  BE INPUT USING UNIT INZ. THE NAME OF THE FILE MUST
!    BE INPUT SUBSEQUENTLY, WITH THE VALUE OF INZ IF LFOR77 IS TRUE.
!
! SUPPLEMENTAL INFORMATION MUST BE GIVEN IN THE FOLLOWING SEQUENCE:
! (A)  NAME OF FILE CONNECTED TO UNIT INZ (LIN4 TRUE),
! (B)  IF LPUNCH IS TRUE, NAME OF FILE CONNECTED TO UNIT 17.
! (C)  FORMAT OF DATA RECORD (LFORM TRUE)
! (D)  VG, SAMPLING INTERVAL MAGNITUDE (LSTAT TRUE)
! (E)  WHEN LPARM IS TRUE: LEQP, TRUE WHEN ALL OBSERVATIONS DEPEND
!      ON THE SAME MP PARAMETERS. THEN MP AND IF LEQP IS TRUE THE PARA-
!      METER CODES. THEY MUST BE INPUT WITH THE DATA, IF LEQP IS FALSE.
! (F)  WHEN LADMU IS TRUE, AND IKP=2 OR 13 (GRAVITY), OR IKP=1 (HEIGHT
```



```
!      ANOMALY) OR IKP=10 (DENSITY ANOMALY), THEN INPUT OF A
!      MULTIPLICATIVE AND AN ADDITIVE CONSTANT AND LMEGR, TRUE IF
!      THE GRAVITY OR GRAVITY GRADIENT QUANTITY IS A MEASURED VALUE.
!      (IN THE CASE IKP=1 ONLY THE ADDITIVE CONSTANT HAS MEANING)
!      USED TO GET THE VALUES IN MGAL OR TO REMOVE BIAS (IKP=1) IF
!      NEEDED.
! (G)  LATITUDE AND LONGITUDE BOUNDARIES IF LAREA IS TRUE.
! (H)  IF LMEAN IS TRUE THEN AS (16C), LSIMH, BSIZEN, BSIZEE.
! (I)  IF LSA IS TRUE, THE  COMMON DATA ERROR, WM. (IT MUST BE FALSE
!      WHEN LERNO IS FALSE AND LNCOL OR LPRED ARE TRUE).
! (J)  IF IKP=10 (DENSITY CONTRAST), INPUT OF EXPONENT OF WEIGHT
!      FACTOR ON HARMONIC DENSITY AND RADIUS OF SPHERE INCLUDING MASSES.
! (K)  WHEN LSTAT, LERNO AND LCOMP ARE TRUE, INFORMATION ON SUSPECTED
!      GROSS-ERRORS MAY BE OUTPUT TO FORTRAN UNIT 24. THIS IS INDI-
!      CATED BY GIVING THE ERROR LEVEL (TYPICALLY 3.0). IF THIS IS
!      LARGER THAN 0.0, THEN THE FILE NAME MUST BE INPUT AS WELL.
! (L)  WHEN A ROTATED REFERENCE FRAME IS USED, IT MUST BE INDICATED
!      WHETHER THE ROTATION ONLY IS IN THE HORIZONTAL PLANE OR A FULL
!      ROTATION.
!
! IF OBSERVATION CODES LARGER THAN 10 ARE USED, HP WILL BE USED ONLY
! WHEN NO HEIGHT IS GIVEN EXPLICITLY FOR EACH POINT (IH = 0). HP IS
! ALSO USED TO COMPUTE THE STANDARD DEVIATION OF THE SIGNAL QUANTITIES
! OCCURRING IN THE HEADING OF THE OUTPUT TABLES.
!
! ================== RETURN POINT IF INPUT PARAMETERS NOT OK ========
 1119 IF (LINTER) WRITE(6,1120)
 1120 FORMAT(' INPUT DATA LINE AND OUTPUT SPECIFICATIONS',/&
 ' POSITION OF STATION NUMBER (0: NO NUMBER, -1: NO OUTPUT U6)',/&
 ' POSITION OF LATITUDE AND LONGITUDE (E.G. 2 , 3)',/&
 ' TYPE OF ANGULAR UNITS USED (1: DD MM SS.S, 2: DD MM.M 3: DD.D)'/&
 ' 4: GRADES, 5: X,Y,Z (CTRS) '/&
 ' POSITION OF HEIGHT (0: NO HEIGHT)',/&
 ' POSITION OF OBSERVATION 1 AND 2 (0 IF NO OBS. 1 OR 2)',/&
 ' DATA OR COMPUTATION QUANTITY TYPE CODE (11: GEOID,',/&
 '   13: GRAVITY, 15: TZZ, 26: (KSI,ETA), NEGATIVE: REF.SUBTR.)'/&
 ' COORD.SYST. CODE, -1 INDICATE GLOBAL SYSTEM, +100 REVERSE TR.'/&
 ' HEIGHT (IN M OR KM), ONLY USED IF NO INPUT HEIGHT',/&
 ' (USED AS HEIGHT ABOVE MEAN EARTH SPHERE IF LSPHER IS TRUE C)',/&
 ' LPUNCH - TRUE IF OUTPUT OF RESULT TO FILE',/&
 ' LWLONG - TRUE IF LONGITUDE POSITIVE EAST',/&
 ' LMEAN  - OBS. OR COMPUTED QUANTITY IS A MEAN VALUE',/&
 ' LSA    - TRUE IF ALL ERROR ESTIMATES ARE IDENTICAL',/&
 ' LKM    - TRUE IF HEIGHT IN KM',/&
 ' LADMU  - TRUE IF UNREDUCED OR CONSTANTS * OR +',/&
 ' STAT   - TRUE IF STATISTICS OF RESULT WANTED',/&
 ' LAREA  - TRUE IF DATA ONLY INSIDE SPECIFIC AREA ARE USED',/&
 ' LFORM  - TRUE IF FORMAT OF DATA IS INPUT',/&
 ' LIN4   - TRUE IF DATA NOT IN INPUT STREAM (FROM FILE)')
 READ(5,*)INO,ILA,ILO,IANG,IH,IOBS1,IOBS2,IKP,&
 ICSYS,HP,LPUNCH,LWLONG,LMEAN,LSA,LKM,LADMU,LSTAT,LAREA,LFORM,LIN4
 LNUOUT=INO.LT.0
 NO=0
 IF (LNUOUT)INO=-INO
 LRESOL=IKP.LT.0
 IF (LRESOL) IKP = - IKP
 LINTRA=ICSYS.GT.100
 IF (LINTRA) ICSYS=ICSYS-100
! THIS INDICATES, THAT THE INVERSE DATUM-SHIFT TRANSFORMATION SHOULD
! BE USED.
 LALLCO=IKP.GT.140
! ADDED 2005-09-21.
 IF (LALLCO) THEN
  IKP=IKP-100
!      IF ((.NOT.LPUNCH).OR.(.NOT.(LPRED.AND.LNCOL))) LALLCO=LF
  IF ((LPRED.OR.LNCOL).AND.LPUNCH) THEN
   WRITE(*,*)' ALL COMPONENTS WILL BE OUTPUT TO FILE ',LALLCO
  ELSE
! THE OUTPUT OF ALL COMPONENTS IS ONLY POSSIBLE IF WE ARE PREDICTING.
```

```
   WRITE(*,*)' ALL COMPONENTS WILL NOT BE OUTPUT, LPRED=LF '
   LALLCO=LF
  END IF
 END IF
!
 IKPREF=IKP
! LGIGRS IS TRUE IF SPECIAL FORMATS ARE USED.
! LUSNGS IS TRUE IF NGS FORMATS ARE USED. LSTNO IS TRUE IF
! KMS STATION NUMBERS ARE USED.
 LGIGRS=IKP.GT.40.AND.IKP.LT.90
 LUSNGS=IKP.EQ.96
 LSTNO=LGIGRS.AND.INO.EQ.11
 IF (LSTNO) LNUOUT=LF
 LNEWD=ICSYS.LT.0
 LTRAN=.NOT.LNEWD
 LMEGR=LF
 IF (LGIGRS) IKP=IKP-40
 IF (LGIGRS.AND.IKP.EQ.2)IKP=13
 IF (LUSNGS) IKP=5
! ADDED 2004-08-11.
 LZETA=IKP.EQ.1.OR.IKP.EQ.11
 LGRADI=IKP.EQ.15.OR.IKP.EQ.30.OR.IKP.EQ.35.OR.(IKP.GE.20.AND.IKP.LE.25)
 LSATP=((IKP.EQ.12.OR.IKP.EQ.16.OR.IKP.EQ.17.OR.IKP.EQ.13.OR.LGRADI.OR.IKP.EQ.11
).AND.LGIGRS)
 LSATPP=LSATP
 IF (.NOT.LPRED) LSATAC=(LSATAC.OR.LSATP)
! LSATAC REGISTRES THE STATE THAT DATA IN A SATELLITE FRAME HAS
! BEEN USED AS OBSERVATIONS.
!
 IF (LSATP) THEN
  ISATP=1
  WRITE(6,*)' DATA IN LOCAL FRAME.'
! INITIALIZING SATROT ADDED 2005-03-02.
  SATROT(1,1)=D1
  SATROT(2,2)=D1
  SATROT(3,3)=D1
  SATROT(1,2)=D0
  SATROT(2,1)=D0
  SATROT(1,3)=D0
  SATROT(3,1)=D0
  SATROT(2,3)=D0
  SATROT(3,2)=D0
 END IF
!
! FOR LSAT=TRUE, WE EXPECT 1. & 2. ORDER DERIVATIVES TO BE GIVEN IN A
! SATELLITE ORIENTED COORDINATE SYSTEM.
! THE AZIMUTH, THE ROLL AND THE PITCH ANGLES MUST BE INPUT AFTER
! EACH OBSERVATION ON A SEPARATE RECORD IN DECIMAL DEGREES OR
! IN THE FORM OF A FULL 3*3 ROTATION MATRIX OR EQUIVALENT QUARTERNION.
! MODIFICATION INTRODUCED JULY 1989 BY CCT,SEPT. 2004 AND MAR 2013..
!
 LGRERR=LSTAT.AND.LCOMP.AND.LERNO.AND.LPRED.AND.(.NOT.LNCOL)
 IF (.NOT.LOPEN4) INZ=5
 IF (LIN4) THEN
!
! --------------- INPUT (9A) --------------------------------
! FIRST INPUT OF DOCUMENT (FILE) NAME. IN FORTRAN 77, UNIT NUMBER
! MUST BE INPUT AS WELL.
  IF (LINTER)WRITE(6,*)' INPUT NAME OF FILE HOLDING DATA'
  READ(5,2103)DNAME(1)
  INZ=4
  IF (LINTER)WRITE(6,*)' INPUT FORTRAN UNIT NUMBER'
  IF (LFOR77) THEN
   READ(5,*)INZ
   IF (INZ.LT.22.OR.INZ.GT.28) WRITE(*,*) ' WARNING6: UNIT NUMBER MAY BE IN CONF
LICT WITH OTHER '
  END IF
!
  IF (.NOT.(LOPEN4.AND.OLDN(3).EQ.DNAME(1).AND.OLDN(4).EQ.DNAME(2).AND.INZOLD.EQ
```



```
.INZ)) THEN
   IF (LOPEN4) CLOSE(INZ)
   OPEN(UNIT=INZ,FILE=DNAME(1),STATUS='OLD',FORM='FORMATTED')
   WRITE(6,169)INZ,(DNAME(I),I=1,ICHAR)
  169   FORMAT(/' DATA INPUT FROM UNIT',I3,', FILE=',2A128)
   OLDN(3)=DNAME(1)
   OLDN(4)=DNAME(2)
   LOPEN4=LT
   INZOLD=INZ
  END IF
 END IF
!
! --------------- INPUT (9B) --------------------------------
!
 IF (LPUNCH) THEN
  IF (LINTER) WRITE(6,*)' INPUT NAME OF FILE TO HOLD RESULT'
  READ(5,2103)DNAME(1)
  IF (.NOT.(LOPEN7.AND.OLDN(1).EQ.DNAME(1).AND.OLDN(2).EQ.DNAME(2))) THEN
   IF (LOPEN7) THEN
    END FILE 17
   END IF
   OPEN(17,FILE=DNAME(1),STATUS='UNKNOWN',FORM='FORMATTED')
   WRITE(6,290)(DNAME(I),I=1,ICHAR)
  290   FORMAT(/' SIMULTANEOUSLY OUTPUT TO FILE ',2A128)
   LOPEN7=LT
   OLDN(1)=DNAME(1)
   OLDN(2)=DNAME(2)
  END IF
 END IF
!
! THE FOLLOWING IS TO ASSURE THAT OUTPUT ON UNIT 17 IS NOT MIXED.
 IF (LWRSOL.AND.(.NOT.LPRED).AND.LPUNCH) LPUNCH = LF
!
! --------------- INPUT (9C) --------------------------------
 IF (LINTER.AND.LFORM)WRITE(6,*) ' INPUT DATA FORMAT (EG. (I3,5F7.2) )'
 IF (LFORM) THEN
  READ(5,103)FMT(1)
  WRITE(*,*)FMT(1)
 END IF
  103 FORMAT(A128)
 2103 FORMAT(A128)
! INPUT OF FORMAT OF INPUT RECORD IF VARIABLE FORMAT CAN BE USED.
!
 IF ((LKM.OR.HP.GT.1.0D5).AND.IKP.NE.51) THEN
! CHANGE 2008-08-12.
  IF (LKM) HP = HP*1.0D3
! ADDITION 1999.12.13 AND 2005-04-11 BY CCT.
! IF UNITS ARE KM, OR HEIGHT ABOVE 100 KM WE EXPECT
! OBS TO BE SMALL AND USE A 4 DIGIT LAYOUT.
  LSMAL=LT
 ELSE
  LSMAL=LF
 END IF
!
 RP = RE+HP
 IF (LWRSOL) MKP = -IKP
 IO2=MOD(IOBS2,10)
 IF (IO2.NE.0) IO2=6
! CHANGE 2000-10-06.
 IF (LKM) THEN
  HPK=HP*1.0D-3
 ELSE
  HPK=HP
 END IF
 IF (LWRSOL) WRITE(17,902)IANG,IO2,MKP,ICSYS,HPK,LF,LF,LMEAN,LF,LKM,LF,LF,LF,LF,
LF
  902 FORMAT(' -1  2  3',I3,'  4  5',3I4,F10.2,10L2)
 LCOD = IKP.GT.5 .AND. IKP .LT. 10
!
```

```
! --------------- INPUT (9D) --------------------------------
! INPUT OF SAMPLING INTERVAL MAGNITUDE.
 IF (LSTAT) THEN
  IF (LINTER)WRITE(6,*)' INPUT SAMPLING INTERVAL SIZE'
  READ(5,*)VG
! CHANGE 2013-09-16.
  IF (VG.LT.0) WRITE(*,*) ' STATISTICS WILL BE OUTPUT AS ASCII AND HTML '
  WRITE(*,903)VG
 903 FORMAT(' SAMPLING INTERVAL= ',F8.4)
 END IF
!
 IF (LPARAM) THEN
!
  NCXLAS = 0
! --------------- INPUT (9E) --------------------------------
! INPUT OF LOGICAL VARIABLE LEQP, TRUE IF ALL OBSERVATIONS DEPEND
! ON THE SAME PARAMETERS AND NUMBER OF PARAMETERS, MP. IF LEQP IS
! TRUE, THEN INPUT OF PARAMETER IDENTIFICATION CODES. OTHERWISE
! THEY MUST BE EXPLICITLY OR IMPLICITLY GIVEN FOR EACH OBSERVATION
! E.G. BY THE REVOLUTION OR TRACK NUMBER.
  IF (LPRED) IPA = 0
  NPOBS0=0
  NOUSE=0
! CHANGE 2005-03-29.
  IF (.NOT.LCOD) ILAST=IPA+1
  ITROLD=-9999
  IF (LINTER) WRITE(6,1118)
 1118  FORMAT(                                                 &
  ' INPUT LEQP, TRUE IF ALL OBS. OR COMPUTED VALUES DEPEND ON',/&
  ' THE SAME PARAMETER (T/F) AND NUMBER OF PARAMETERS '       ,/&
  ' (IF DATA DOES NOT DEPEND ON ANY PARAMETERS INPUT T  0) '     )
  216  FORMAT(' LEQP,MP ',L2,I7)
  READ(5,*)LEQP,MP
  IF (LWRSOL) WRITE(17,216)LEQP,MP
  WRITE(*,216)LEQP,MP
! ADDED 2002-02-18.
  IF ((.NOT.LEQP).AND.(IKP.EQ.11.OR.LGRADI)) THEN
! THIS IS A PRELIMINARY SOLUTION IN ORDER TO IDENTIFY PARAMETER GROUPS.
! CHANGED 2003-04-02.
! IF ALTIMETRY (IKP=11) AND TRACK NUMBER IS USED, INPUT 1.0 0.5.
! IF OBSERVATION TIME IS USED, INPUT START TIME AND TIME OF ONE
! REVOLUTION.
   IF (LINTER) WRITE(*,*)' INPUT TIME INTERVAL AND START TIME '
   READ(5,*)PPA,PPS
   WRITE(*,*)' TIME INTERVAL PER PARAM. AND START TIME ',PPA,PPS
   IF (LWRSOL) WRITE(17,*)PPA,PPS
  END IF
!
  IF ((.NOT.LCOD).OR.LPRED) THEN
   IPA = IPA+2
  END IF
  IF (IPA.GT.IPAMAX) IPAMAX=IPA
  IF (LEQP.AND.(.NOT.LCOD).OR.LPRED) IPACAT(IPA) = MP
  IF (.NOT.(LEQP.OR.(LCOD.AND.(.NOT.LPRED)))) IPACAT(IPA) = -MP
  IF( MP.NE.0) THEN
   IF (LINTER) THEN
    WRITE(6,*)' INPUT PARAMETER CODES'
   END IF
   IF (LEQP .AND. MP.NE.0) THEN
    READ(5,*)(IPACAT(I+IPA),I=1,MP)
    CALL PARCAT(LALLP,NPNO)
    WRITE(6,170)MP,(IPACAT(I+IPA),I=1,MP)
    IF (LWRSOL) WRITE(17,150)(IPACAT(I+IPA),I=1,MP)
  150    FORMAT(12I6)
  170    FORMAT(/' OBSERVATIONS CONTRIBUTE TO/DEPEND ON ',I3,&
    ' PARAMETERS',3I6)
    IF (MP.GT.3) WRITE(6,171)MP,(IPACAT(I),I=1,MP)
  171    FORMAT(/' OBSERVATIONS CONTRIBUTE TO/DEPEND ON ',I3,&
    ' PARAMETERS',3I6,(/,12I6))
```



```
   ELSE
    IF (MP.GT.10) THEN
     WRITE(*,*)' MP > 10 '
     STOP
    END IF
    IF (.NOT.LCOD)THEN
! ICODE VALUES ARE: BIAS, 1, TILT, 2, SCALE FACTOR, 3.
! VALUES FROM 4 - 10 ARE FOR FOURIER COEFFICIENTS. HERE
! PERIOD AND PHASE MUST BE INPUT. CHANGE 2006-03-20.
     READ(5,*)(ICODE(I),I=1,MP)
     IF (MP.LT.3)  THEN
      WRITE(6,170)MP,(ICODE(I),I=1,MP)
     ELSE
      WRITE(6,171)MP,(ICODE(I),I=1,MP)
     END IF
     DO I=1,MP
      LTILT=LTILT.OR.ICODE(I).EQ.2
      LSCALE=LSCALE.OR.ICODE(I).EQ.3
! CHANGE 2006-03-20. INPUT OF PERIOD AND PHASE.
      IF (ICODE(I).GT.3) THEN
       IF (LINTER) WRITE(*,*)' INPUT PERIOD AND PHASE IN DEGREES '
       READ(5,*)FPERIO(ICODE(I),1),FPERIO(ICODE(I),2)
       WRITE(*,172)FPERIO(ICODE(I),1),FPERIO(ICODE(I),2)
  172       FORMAT(' PERIOD= ',F10.3,' PHASE (DEG.) = ',F12.5)
! CONVERSION TO RADIANS.
       FPERIO(ICODE(I),1)=D2*PI/FPERIO(ICODE(I),1)
       FPERIO(ICODE(I),2)=PI/180.0D0*FPERIO(ICODE(I),2)
      END IF
     END DO
!         IF (LTILT) WRITE(*,*)' LTILT ',LTILT
    END IF
   END IF
  END IF
 END IF
!
 DM = D1
 DA = D0
 LGRP = IKP.EQ.2.OR.IKP.EQ.13.OR.IKP.EQ.12
 LZETA = IKP.EQ.1.OR.IKP.EQ.11
 LDEN=IKP.EQ.10
 LPOTSD=LGRP.AND.(.NOT.LP(1))
!
! --------------- INPUT (9F) --------------------------------
 IF (LADMU) THEN
  IF (LINTER) WRITE(6,1121)
 1121  FORMAT(' INPUT MULTIPLICATIVE AND ADDITIVE CONSTANT AND',/&
  ' LMEGR, TRUE IF VALUE INPUT OR COMPUTED IS UNREDUCED')
  READ(5,*)DM,DA,LMEGR
  WRITE(*,1123)DM,DA,LMEGR
 1123 FORMAT(' DM= ',D16.5,', DA= ',D16.5,', LMEGR= ',L2)
 END IF
! LMEGR IS TRUE, WHEN THE MEASURED GRAVITY VALUE IS INPUT. DM AND DA ARE
! AN ADDITIVE AND A MULTIPLICATIVE CONSTANT, RESPECTIVELY, WHICH CAN BE
! USED TO CONVERT INPUT VALUES TO MGAL OR CORRECT FOR A SYSTEMATICAL
! ERROR. IF LZETA IS TRUE, THEN ONLY BIAS CORRECTION IS POSSIBLE.
!
! ADDED 2006-04-17 TO SECURE THAT IF A SCALEFACTOR IS TO BE DETERMINED
! THEN THE INPUT QUANTITY NOT IS AN ANOMALY BUT AN UNREDUCED QUANTITY.
 IF (LSCALE) THEN
  IF (.NOT.LMEGR) THEN
   WRITE(*,*)' LMEGR MUST BE TRUE , STOP '
   STOP
  END IF
 END IF
!
! --------------- INPUT (9G) --------------------------------
 IF (LAREA) THEN
  IF (LINTER) WRITE(6,*) ' INPUT MINIMUM AND MAXIMUM LATITUDE AND LONGITUDE (DEG
.)'
```

```
  READ(5,*)RLAMIN,RLAMAX,RLOMIN,RLOMAX
  WRITE(6,208)RLAMIN,RLAMAX,RLOMIN,RLOMAX
  208  FORMAT(' DATA SELECTED IN AREA BOUNDED BY:'/&
  ' LATITUDE ',2F9.2,' DEG ',/,' LONGITUDE',2F9.2,' DEG'/)
  IF (RLOMIN.GT.RLOMAX) THEN
   WRITE(*,*)' WARNING6 ',RLOMIN,RLOMAX
   RLOMIN=RLOMIN-360.0D0
  END IF
  CALL RAD(0,0,RLAMIN,RLAMIN,3)
  CALL RAD(0,0,RLOMIN,RLOMIN,3)
  CALL RAD(0,0,RLAMAX,RLAMAX,3)
  CALL RAD(0,0,RLOMAX,RLOMAX,3)
  IF (RLOMIN.GT.RLOMAX) THEN
   WRITE(*,*)' WARNING7 ',RLOMIN,RLOMAX
   RLOMIN=RLOMIN-D2*PI
  END IF
 END IF
!
! --------------- INPUT (9H) ----------------------------------
 IF (LMEAN) THEN
  LFILTE=LF
  IF (LINTER)WRITE(6,1117)
  READ(5,*)LSIMH,LEQANG,BSIZEN,BSIZEE,RLATP
  IF (LWRSOL) WRITE(17,231) LSIMH,LEQANG,BSIZEN,BSIZEE,&
  RLATP
 1117   FORMAT(' INPUT PARAMETERS DEFINING TYPE AND SIZE OF MEAN VALUE',/&
   ' LSIMH - TRUE IF WE USE EQUIVALENT HEIGHT REPR. OF MEAN',/&
   ' LEQANG- TRUE IF EQUAL ANGULAR OR 1-D BLOCK',/&
   ' BLOCK SIDE LENGTH IN LAT. & LONG. (MIN) OR (LENGTH .0 IF 1D)',/&
   ' LATITUDE OF TOTAL AREA MEAN')
  231   FORMAT(2L2,3F10.2)
! LMEAN1 IS TRUE IF MEAN VALUES ARE 1D, ALONG A SATELLITE OR
! AIRCRAFT TRACK, FOR EXAMPLE
  LMEAN1=.NOT.LSIMH.AND.(.NOT.LEQANG).AND.ABS(BSIZEE).LT.1.0D-8
! write(*,*)' 4748 ',LMEAN1,LFILTE
  IF (LMEAN1.AND.(.NOT.LFILTE)) CALL MEAN1(FILTER,NFILTE,SAZP,CAZP,LFILTE,LGRID,
LINTER)
 ENDIF
!
 IF ((LE.AND.LSA).OR.(LSA.AND.LGRERR)) THEN
!
! ---------------- INPUT (9I) -------------------------------
  IF (LINTER) WRITE(6,*) ' INPUT COMMON STANDARD DEVIATION OF OBSERVATIONS'
  READ(5,*)WM
  WRITE(*,212)WM
  212  FORMAT(' COMMON ST.DEV. OF OBS = ',F8.4)
 ELSE
  WM=D0
 END IF
!
 IF (LDEN) THEN
!
! ---------------- INPUT (9J) -------------------------------
  CALL DENDEF(NMAX,LINTER,LWRSOL,LPARAM,&
  LPOT,LBIPOT,LBIN,LINSOL,LDENOL,LSKIPL,RRE)
! INPUT OF EXPONENT OF WEIGHT FACTOR ON HARMONIC DENSITY,
! RADIUS OF SPHERE WITHIN WICH MASSES ARE LOCATED IN M. CF. REF(F),
! SECTION 3,SALE FACTOR AND VALUE OF LOGICAL VARIABLE LNPOT
! TRUE, IF NEW SET OF COEFFICIENTS ARE TO BE USED FOR THE DENSITY
! COMPUTATIONS, (DIFFERENCES BETWEEN THE ORIGINAL COEFFICIENTS AND
! COEFFICIENTS OF A TOPOGRAPHIC-ISOSTATIC REDUCTION POTENTIAL).
! IF POTENTIAL COEFFICIENTS NOT ALREADY STORED ON BINARY FORM
! ALSO THE NAME OF THE FILE TO BE CONNECTED TO UNIT 3.
! THIS ONLY APPLIES IF LPOT IS TRUE.
 END IF
!
 IF (LGRERR) THEN
!
! ----------------------- INPUT (9K) ---------------------------
```



```
! INPUT OF REJECTION LEVEL FOR GROSS-ERRORS. IF LARGER THAN 0.0,
! NAME OF FILE, WHERE SUSPECTED GROSS-ERRORS ARE OUTPUT MUST BE
! GIVEN AS WELL.
  IF (LINTER) WRITE(6,*)' INPUT REJECTION LEVEL (<0 NO REJ.)'
  READ(5,*)REJLEV
  LGRERS=LGRERR
  LGRERR=LGRERR.AND.REJLEV.GT.D0
  IF (LGRERR) THEN
   IF (LINTER)WRITE(6,*)' INPUT NAME OF FILE TO HOLD REJECTED OBS.'
   READ(5,'(A)')ERNAME
   OPEN(12,FILE=ERNAME,FORM='FORMATTED')
   WRITE(6,*)' OUTLIERS WILL BE OUTPUT TO FILE ',ERNAME
  END IF
! INITIALIZING ARRAY TO HOLD STATISTICS OF ERRORS.
  DO NGR=1,8
   NGRE(NGR)=0
   SGRE(NGR)=D0
  END DO
  NGRERR=0
 ELSE
  LGRERS=LF
! ADDED 2013-04-09.
 END IF
!
 ISATP=0
 IF (LSATP) THEN
!
! ---------------------- INPUT (9L) ---------------------------
! INPUT OF INTEGER, ISATP=1 WHEN ROTATION IS ONLY IN HORIZONTAL
! PLANE, =2 WHEN IT IS A FULL ROTATION AND 3 WHEN NO ROTATION IS
! MADE. WHEN ISATP=4 THE FULL ROTATION MATRIX IS FOUND ON A SEPARATE FILE.
! ISATP=5: QUARTERNION TRF-> GRF, ISATP=6 QUARTERNION XYZ -> GRF.
  IF (LINTER) THEN
! CHANGE 2004-08-31.
   WRITE(6,*) ' INPUT 1 FOR HORIZ. ROT, 2 FOR FULL 3D ROTATION (EULER) '
   WRITE(*,*) ' 3 FOR NO ROT., 4 FOR ROT MAT FILE, 5 FOR QUART. '
  END IF
  READ(5,*)ISATP
  IF (IKP.EQ.11)ISATP=3
  IF (ISATP.EQ.4) THEN
! ROTATION MATRIX FILE MUST HAVE FORMAT: TIME AND 6 MATRIX ELEMENTS
! GIVING ROTATION FROM ENU SYSTEM TO LOCAL SYSTEM.
   WRITE(*,*)' INPUT FILE NAME '
   READ(5,'(A)')ROTFIL
   WRITE(*,*)' ROTATION MATRIX FILE ',ROTFIL
   OPEN(13,FILE=ROTFIL)
  END IF
  IF (ISATP.GE.5) THEN
! INPUT OF QUARTERNION-FILE NAME. 2005-03-12.
   WRITE(*,*)' WARNING8: NOT FULLY IMPLEMENTED **** '
   WRITE(*,*)' INPUT FILE NAME '
   READ(5,'(A)')ROTFIL
   WRITE(*,*)' ROTATION MATRIX FILE ',ROTFIL
   OPEN(13,FILE=ROTFIL)
  END IF
 END IF
!
 ITMODE=0
 IF (LCOERR.AND.(.NOT.LPRED).AND.LSA) THEN
! CHANGE 2013-04-04.
! LCOERR IS TRUE IF CORRELATED ERRORS MAY EXIST, SEE INPUT (1D).
! ---------------------- INPUT (9M) ---------------------------
! INPUT OF LOGICAL VARIABLE SIGNIFYING WHETHER THIS DATA-SET HAS CORRELATED
! ERRORS. (CHANGE 1998-03-20).
  IF (LINTER) WRITE(*,*) ' INPUT T IF DATASET HAS CORRELATED ERRORS, OTHERWISE F
.'
  READ(5,*)LLCOER
  IF (LLCOER) THEN
   ITOLD=-999999
```

```
   WRITE(*,*) ' DATASET HAS CORRELATED ERRORS '
! THE NOISE COVARIANCE FUNCTION MAY BE SPECIFIED AS A FOURIER SERIES
! OR AS A FINITE COV. FCT.
   IF (LINTER) THEN
    WRITE(*,8701)
 8701    FORMAT(' INPUT T IF NOISE COVARIANCE FUNCTION FOURIER SERIES,',&
    ' OR F IF FINITE FUNCTION, ',/,&
    ' FOLLOWED BY TRACK-MODE, = 1 IF TRACK NUMBER INPUT AS ',/,&
    ' SEPARATE RECORD AFTER EACH OBSERVATION ',/,&
    ' AND = 2,  3 OR 4 IF IT DEPENDS ON THE "STATION NUMBER"',/,&
    ' WITH NEGATIVE VALUES IF FUNCTION DEPENDS ON TIME AND ',/&
    ' NOT SPHERICAL DISTANCE. ')
   END IF
   READ(5,*)LFOUR,ITMODE
   WRITE(*,*)' TRACK INPUT MODE = ',ITMODE
! CHANGE 2005-04-04.
   LCTIME = ITMODE.LT.0
   IF (LCTIME) ITMODE=-ITMODE
   IF (LFOUR) THEN
    WRITE(*,*)' NOT FULLY IMPLEMENTED *** WARNING9 *** '
    IF (LINTER) THEN
     WRITE(*,*)' INPUT MAX DEG. OF COEFFICIENTS (<21) '
     WRITE(*,*)' AND COEFFICIENTS (0-MAX) (TWO LINES) '
    END IF
    READ(5,*)NFOUR
    IF (NFOUR.GT.20) THEN
     WRITE(*,*)' LIMIT EXCEEDED FOUCOF, STOP '
     STOP
    END IF
    write(*,*)' DATA-SET ANDEX: JR= ',JR0
    JR1=(JR0)/2
    JR2=(JR0+2)/2
    write(*,*)' ERROR CORRELATION COEFFICIENTS BETWEEN ',JR1,JR2
    DO JR3=(JR1*(JR1+1))/2+1,(JR2*(JR2+1))/2
     READ(5,*)(FOUCOF(JR3,IJ),IJ=0,NFOUR)
     write(*,*)' ERROR CORRELATION COEFFICIENTS BETWEEN ',JR1,JR2,JR3
     WRITE(*,*)(FOUCOF(JR3,IJ),IJ=0,NFOUR)
    END DO
    IF (ABS(FOUCOF(JR1*(JR1+1)/2+1,0)-WM**2).GT.0.001*WM**2) THEN
     WRITE(*,*)' INCONSISTENCY BETWEEN COMMON ERROR STDV. AND FOUCOF ',&
     WM
    END IF
   ELSE
    IF (LINTER) WRITE(*,*) ' INPUT NOISE VARIANCE AND HALF DISTANCE (DEG.)'
    IF (LINTER) WRITE(*,*) ' TO ZERO FOR ERROR COV.FCT.'
    READ(5,*)VARNO,RDD
 8702    FORMAT(' NOISE VAR. AND RDD (DEGREES) = ',2D12.5)
    WRITE(*,8702) VARNO, RDD
   END IF
   IF (.NOT.LCTIME) RDD=RDD*PI/180.0D0
   SCFACT=VARNO/COZERO(D0,RDD,1)
   WRITE(*,*)' SCFACT ',SCFACT
  END IF
! ADDED 2002-10-09.
  IF (ITMODE.EQ.2) THEN
! THIS IS TO BE USED WHEN STATION NUMBERS (OR TIME) MAY BE
! SEGMENTED IN SLICES EQUALLY LONG (ITMOD).
   IF (LINTER) WRITE(*,*) ' INPUT START NUMBER, AND NUMBER OF POINTS IN TRACK '
   READ(5,*)ITM0,ITMOD
   WRITE(*,*)' START NO. AND NUMBER OF POINTS ',ITM0,ITMOD
   ITRAC0=0
  END IF
! NEW MODE ADDED 2003-03-19.
  IF (ITMODE.EQ.3) THEN
! THIS IS TO BE USED WHEN A NEW TRACK IS IDENTIFIED BECAUSE THE
! "STATION NUMBERS" CHANGE MORE THAN ITRGAP.
   IF (LINTER) WRITE(*,*) ' INPUT MINIMUM GAP BETWEEN TRACKS '
   READ(5,*)ITRGAP
   ITOLD=-10
```



```
   WRITE(*,*)' MIN: GAP BETWEEN TRACKS ',ITRGAP
  END IF
 ELSE
  LLCOER=LF
 END IF
!
 IF (LINTER) THEN
  WRITE(6,*)' ALL SPECIFICATIONS OK ?'
  READ(5,*)LOK
  IF (.NOT.LOK) GO TO 1119
 END IF
!
 RETURN
END SUBROUTINE DEFDAT

!-----------------------------------------------------------------
SUBROUTINE INHEAD(LCOLLO,LREPEC,PW,LNFORM,LADDBP,LINVDE,LADBA,LADDBC)
! THE SUBROUTINE COLLECTS ALL INFORMATION FROM INPUT (9).
! MOVED FROM MAIN-PROGRAM 2004-11-27. LAST CHANGE 2013-09-14.

 USE m_params,      ONLY : MAXO,NSAT
 USE m_geocol_data, ONLY : STEPN,COSSTN,SINSTN,STEPE,COSSTE,SINSTE,&
                           COST2P,SINT2P,NFILTE,SATROT,LFORM
 USE m_geocol_data, ONLY : SHIFTS,PW2,BSIZEA,E21,AX1,F1,GM1,GREF,UREF,&
                           SM,KP,KPP1,IPC,LSMAL,LADBPR,LADBTE,LNGR,&
                           LKSIP,LNCOL,LTNB,LTEB,LOE1,LOE2,LE
 USE m_data,        ONLY : LCOMP,LMDD,ICSYSL,LINSOL,LERNO,OBS,LRESOL,LGRID

 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN

 USE m_data,        ONLY : BSIZE,ANDEX,PRETAP,PREDP,HCZERO,NCZERO,LNEWD

 USE m_geocol_data, ONLY : B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,  &
                           WOBS,ISAT,SINLOP,COSLOP,BSIZEN,BSIZEE,COSLAP, &
                           SINLAP,RLONGP,RP,CAZP,SAZP,HP,RLATP,ICZERO,   &
                           NI,NR,IKP,ISATP,NOBLK,LONECO,LNKSIP,LNETAP,   &
                           LDEFVP,LOBSST,LSTART
 USE m_geocol_data, ONLY : FG,FJ,LP,OMEGA2,IORDER
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,DC,HCMAX,KCI
 USE m_geocol_data, ONLY : CCR,SIGMAX,CCV,NC1,NC2,LOCAL,LSUM
 USE m_geocol_data, ONLY : INUMR,NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,&
                           LSTNO=>LTERM,LTERMA,LTERMO,LOUTC,LTRAN,LNERNO,&
                           LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT
 USE m_geocol_data, ONLY : IMAX1,IMAX1R
 USE m_data,        ONLY : OLDCOV,S,SR,AAI,AAR,IS,IPX,LCO1,NBOLD
 USE m_geocol_data, ONLY : IMAX1,IMAX1R
 USE m_data,        ONLY : OLDCOV,S,SR,AAI,AAR,IS,IPX,LCO1,NBOLD,LDIF
 USE m_data,        ONLY : OLDB,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
 USE m_data,        ONLY : 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
 USE m_geocol_data, ONLY : VARNO,PPA,PPS,VG,HPK,DM,DA,WM,REJLEV,SGRE,ROTFIL,&
                           ERNAME,DNAME,FMT,NSTEP,NSTEPE,IDSAT,ILA,ILO,IKPREF,&
                           INZ,IO2,NPOBS0,NOUSE,ILAST,IPAMAX,NGR,NGRE,ICSYS, &
                           LMEAN,LSA,LADMU,LSTAT,LAREA,LZETA,LGRADI,LMENSI,&
                           LSIMH,LMEAN1,LINTRA,LGIGRS,LMEGR,LUSNGS,LSATAC,&
                           lsatp,lgrerr,lcOD,LEQP,LALLP,LGRP,LDEN,LPOTSD,&
                           LEQANG,LFILTE,LBIN,LSKIPL,LGRERS,LTILT,LSCALE,LINERT
 USE m_geocol_data, ONLY : SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS3,EPS2,EPS1,DL,AX2,E
22
 USE m_geocol_data, ONLY : HQD,RLATD,SINLATD,COSLATD,RLONGD,SINLOND,COSLOND
 USE m_geocol_data, ONLY : HQE,RLATE,SINLATE,COSLATE,RLONGE,SINLONE,COSLONE

IMPLICIT NONE

 LOGICAL ::       LREPEC,LINVDE,LCOLLO,LADBA,LNFORM,&
!                 LMEAN,LSA,LADMU,LSTAT,LAREA,LZETA,LGRADI,&
```

```
!                 LMENSI,LSIMH,LMEAN1,LINTRA,LGIGRS,LMEGR,LUSNGS,
                  LADDBC,&
!                 LSATAC,LSATP,LGRERR,LCOD,LEQP,LALLP,LGRP,LDEN,LPOTSD,&
!                 LEQANG,LFILTE,LBIN,LSKIPL,LGRERS,     &
                  LADDBP

 integeR ::       IKC

 REAL(KIND=8)  :: VAR,PW

!COMMON /DAT/LNEWD,LRESOL,LGRID


 LREPEC = IKP.EQ.5.OR.IKP.EQ.7.OR.(IKP.GT.25.AND.IKP.LT.36)
! INITIALIZATION OF VARIABLES IN COMMON BLOCK /PR/.
 LONECO = .NOT.LREPEC
! CORRECTIONS 1992.08.27 TO ADD KODES 22 AND 24 AND LSATP.
 LNKSIP = LONECO.AND.IKP.NE.3.AND.IKP.NE.16.AND.IKP.NE.18
! .AND.IKP.NE.20.AND.IKP.NE.25.AND.IKP.NE.22
 LNETAP = LONECO.AND.IKP.NE.4.AND.IKP.NE.17.AND.IKP.NE.19
! .AND.IKP.NE.21.AND.IKP.NE.23.AND.IKP.NE.24
 LDEFVP = .NOT.LNKSIP.OR.(.NOT.LNETAP)
!LDEFVP = .NOT.LNKSIP.OR.(.NOT.LNETAP).OR.LSATP
!LDEFVP=.FALSE.
! CHANGE 2013-03-15.
 LNGR   = .NOT.LGRP
! LREPEC IS TRUE, WHEN TWO COLUMNS CAN BE COMPUTED AT THE SAME TIME.
! LNSKIP, LNETAP IS TRUE, WHEN THE OBSERVATION OR REQUESTED PREDIC-
! TION IN P IS NOT LIKE KSI RESP. ETA.
 LKSIP  = .NOT.LNKSIP
 LNFORM = .NOT.LFORM
! OUTPUT ADDED MAY 1994 BY CCT.
 IF (LADMU.AND.LPRED.AND.(.NOT.LCOMP)) THEN
  WRITE(*,*)' ABSOLUTE VALUES OUTPUT LAST IN SI UNITS.'
  IF (LZETA) WRITE(*,*)' OUTPUT IS POTENTIAL IN M**2/S**2 '
  IF (LDEFVP) THEN
! CHANGE 2008-09-25.
   WRITE(*,*)' OUTPUT IS D/DX, D/DY IN M/S**2 OR DD/DXX ETC. '
  END IF
 END IF
!
 PW = D0
 KP = IKC(IKP)
 KCI(6) = KP
 KPP1 = KP+1
 IF (.NOT.(LNCOL.OR.LCOD)) THEN
  ANDEX(JR+1)=IKP
! WRITE(*,*)' FROM INHEAD: JR,IKP,LPRED ',JR,IKP,LPRED
 END IF
! IORDER IS ORDER OF DIFFERENTIATION ASSOCIATED WITH VARIABLES
! OF TYPE IKP.
 IORDER = 2
! CHANGE 2013-04-22.
 IF (KP.LE.8) THEN
   IF (KPP1.EQ.1.OR.KPP1.EQ.2) THEN
    IORDER = 0
   END IF
   IF (KPP1.EQ.3.OR.KPP1.EQ.4.OR.KPP1.EQ.7.OR.KPP1.EQ.8) THEN
    IORDER = 1
   END IF
 END IF
!
 LMDD = LF
 IF (IORDER.EQ.2.AND.LMEGR) LMDD=LT
! IF SECOND ORDER DERIVATIVES, LMDD IS TRUE IF THE VALUES ARE OBSERV.
!IF (LMDD) WRITE(*,*)' LMDD= ',LMDD
!
 LINVDE = LONECO.OR.(IOBS2.EQ.0).OR.(IOBS1.LT.IOBS2)
 LOUTC = LNEQ.OR.LCOMP
```



```
 IF ((.NOT.LOUTC).AND.LSTAT)WRITE(6,242)
  242 FORMAT(' *** WARNING10 *** LOUTC IS FALSE, LSTAT IS TRUE')
! BECOMES WRONG IF LOUTC IS NOT TRUE.
!
! OUTPUT OF HEADINGS AND INITIALIZATION OF VARIABLES.
!
 IF (ICSYSL.NE.ICSYS) THEN
  IF (.NOT.LPARAM) WRITE(6,173)
  173  FORMAT(2X)
  ICSYSL=ICSYS
 END IF
!
 IF (ICSYSL.NE.ICSYS.OR.(.NOT.LNEWD)) THEN
! LNEWDA IS FALSE IF BASIC SYSTEM IS NOT USED.
  WRITE(6,240)
  240  FORMAT(' SYSTEM USED:')
  CALL ICOSYS(ICSYS,0,GM1,AX1,E21,F1,UREF,GREF)
  IPC = 0
  LPOTSD = .NOT.LP(1)
 ELSE
!
  WRITE(6,210)
  210  FORMAT(/' SELECTED GEOCENTRIC SYSTEM USED.')
  AX1 = AX2
  E21 = E22
  IPC = 15
  LPOTSD = LF
 END IF
!
 LMENSI=LF
 IF (LMEAN) THEN
  IF (LSIMH) WRITE(6,205)
  205  FORMAT(/' THE FOLLOWING QUANTITIES ARE MEAN-VALUES, AND ARE',&
  ' REPRESENTED',/,' AS POINT VALUES IN THE HEIGHT H.')
  IF ((.NOT.LSIMH).AND.LEQANG) WRITE(6,232)BSIZEN,BSIZEE
  232  FORMAT(/' THE FOLLOWING QUANTITIES ARE MEAN VALUES, WITH',/,&
  ' BLOCKSIZE=',F10.2,' * ',F10.2,' MINUTES')
  IF ((.NOT.LSIMH).AND.(.NOT.LEQANG).AND.(.NOT.LMEAN1)) WRITE(6,233)BSIZEN
  233  FORMAT(/' THE FOLLOWING QUANTITIES ARE EQUAL-AREA MEAN',&
  ' VALUES, WITH BLOCK-SIZE=',F10.2,' MINUTES')
  IF (LMEAN1) THEN
   IF (BSIZEN.GT.0) THEN
    WRITE(6,236)BSIZEN
  236  FORMAT(' THE QUANTITIES ARE 1-D MEANS OVER A ',F9.4,&
     ' ARCMIN TRACK SEGMENT ')
   ELSE
    WRITE(6,*)' DATA ARE DIFFERENCES BETWEEN 2 POINT VALUES (GRACE). '
   END IF
  END IF
  LMENSI=.NOT.LSIMH
  IF (.NOT.LSIMH) THEN
   RLATP=RLATP*3600/RADSEC
! SPHERICAL APPROXIMATION.
   COSLAP=COS(RLATP)
   SINLAP=SIN(RLATP)
   BSIZEN=BSIZEN*60/RADSEC
   IF (.NOT.LEQANG)BSIZEE=D0
   BSIZEE=BSIZEE*60/RADSEC
   IF (LMEAN1) THEN
    NSTEP=NFILTE
   ELSE
    NSTEP=5
   END IF
   NSTEPE=1
   STEPE=D0
   COSSTE=D1
   SINSTE=D0
! THE CALL TO ICMEAN GIVES STEPSIZE AND COS,SIN.
   CALL ICMEAN(BSIZEN,STEPN,NSTEP,COSSTN,SINSTN,D1,D0,LT,LMEAN1)
```

```
   SHIFTS=STEPN*(NSTEP-1)/2
   COST2P=COS(SHIFTS)
   SINT2P=SIN(SHIFTS)
   IF (LMEAN1) THEN
! LMEAN1 INTRODUCED 1992.10.07 BY CCT.
!   BSIZEN=-BSIZEN
! seem wrong
! WE USE THE SWITCH OF SIGN AS AN INDICATOR OF 1D MEAN.
   ELSE
    NSTEPE=5
    IF (LEQANG) THEN
     CALL ICMEAN(BSIZEN,STEPE,NSTEPE,COSSTE,SINSTE,D1,D0,LT,LF)
    ELSE
     BSIZEA= COS(RLATP)*BSIZEE
     CALL ICMEAN(BSIZEA,STEPE,NSTEPE,COSSTE,SINSTE,D1,D0,LT,LF)
    END IF
   END IF
  END IF
 ELSE
! THIS IS USED TO INDICATE IN COMEAN, THAT MEAN-VALUES ARE
! NOT COMPUTED IN P. CORRECTION NOV. 96 BY CCT.
  STEPE=D1
  LEQANG=LF
 END IF
!
 IF ((.NOT.LCOD).AND.LCOLLO) THEN
!
! COMPUTATION OF THE ROOT MEAN SQUARE VARIATION OF THE OBSERVATIONS
! IN THE HEIGHT GIVEN BY HP AND FOR MEAN VALUES OF EQUAL AREA TYPE
! AT LATITUDE RLATP. HP  IS THE HEIGHT ABOVE THE MEAN EARTH SPHERE.
  NI = MAXC1
  IF ((.NOT.LINSOL).OR.LCO1) THEN
! CHANGE 2013-01-04.
   IF (LDIF) THEN
    RLATD=ATAN2(SINLAP,COSLAP)+2.0D0*3.1415926535D0/180.0D0
    SINLATD=SIN(RLATD)
    COSLATD=COS(RLATD)
    SINLONE=0.0D0
    COSLONE=1.0D0
   END IF
   IF (.NOT.LKM) THEN
    PW2=VAR(SM,IS,KP,S,AAI,HP,IMAX1,LMENSI,COSLAP,SINLAP,COSLAP,SINLAP,&
    LSATP,SATROT)
   ELSE
    PW2=VAR(SM,IS,KP,S,AAI,HP/1000.0,IMAX1,LMENSI,COSLAP,SINLAP,&
    COSLOP,SINLOP,LSATP,SATROT)
   END IF
   IF (PW2.GT.D0.AND.(.NOT.LINSOL.OR.LINSOL.AND.LCO1)) THEN
    PW = SQRT(PW2)
!   write(*,*)' PW2 ',pw2
   ELSE
    IF (PW2.LT.D0) WRITE(*,*)' WARNING11 ** ',PW2, ' CALL: '
    WRITE(*,*)IS,KP,S,AAI,HP,IMAX1,LMENSI,COSLAP,SINLAP,LSATP
   END IF
  END IF
!
  LSMAL=(PW.LT.0.5D0.OR.LSMAL).AND.IKP.NE.11
! CHANGE 2004-01-27. And 2008-06-12.
  IF (LSMAL) THEN
   WRITE(*,2455)PW
 2455   FORMAT(' 4 DIGIT LAYOUT IN USE ',/,' PW= ',F12.5)
  END IF
!
  IF (LEQANG) CALL ICMEAN(BSIZEE,STEPE,NSTEPE,COSSTE,SINSTE,D1,D0,LT,LF)
 END IF
!
 IF (HP.GT.3.0D5.AND.(LKSIP.OR.LGRP).AND.(LTRAN.OR.LPOT).AND.LPOTSD) WRITE(6,204
)
  204 FORMAT(                                                          &
```



```
 /' ** WARNING12: THE HEIGHT MAY BE TOO BIG FOR THE COMPUTA'     ,&
 'TION OF',/,' THE REFERENCE GRAVITY OR THE CHANGE IN LATITUDE **' )
!
 CALL HEAD(IKP,LONECO,PW,ISATP.GE.2)
! INITIALIZATION OF LOGICAL VARIABLES USED TO DETERMINE WHICH QUANTITIES
! WE WILL HAVE TO ADD TOGETHER TO FORM THE FINAL OUTPUT OR TO DETERMINE
! WHICH QUANTITIES WILL BE INPUT.
 LADBA = IB.NE.IA
 LADDBC = IB.NE.IC1
 LADDBP = IB.NE.IP
 LADBTE = IB.NE.ITE
 LADBPR = LADDBP.AND.LREPEC
 LTNB = LTRAN.AND.(IT.NE.IB)
 LTEB = LTRAN.AND.(IT.EQ.IB)
 LOE1 = (LE.OR.(LNCOL.AND.LERNO).OR.LGRERR.OR.LGRERS).AND.(.NOT.LSA).AND.LONECO
 LOE2 = (LE.OR.(LNCOL.AND.LERNO).OR.LGRERR.OR.LGRERS).AND.(.NOT.LSA).AND.LREPEC
! ERROR 2000-06-19  ????.
!    *.AND.(.NOT.LSA).AND.LREPEC
! K1 HS BEEN INITIALIZED BY THE CALL OF 'HEAD'. IT WILL BE EQUAL
! TO THE NUMBER OF QUANTITIES READ IN TO THE ARRAY OBI.
 IF (LOE1) K1 = K1+1
 IF (LOE2) K1 = K1+2
 IF ((.NOT.(LOE1.OR.LOE2)).AND.(.NOT.LSA))IIE=0
!     IF (.NOT.LOE2)IIE1=0
! ERROR 2000-06-19  ????.
 IF ((.NOT.LOE2).AND.(.NOT.LSA))IIE1=0
!     WRITE(*,*)LOE2,IIE,IIE1
 IF (LSA) THEN
  K1=MAX0(K1,IIP,IIP1,IITE,IITE1,IOBS1,IOBS2)
 ELSE
  K1=MAX0(K1,IIP,IIP1,IITE,IITE1,IIE,IIE1,IOBS1,IOBS2)
 END IF
!
 OBS(IT) = D0
 IF (LREPEC) OBS(IT1)= D0
 IF (LSTAT) CALL COMPA(VG,IKP,LONECO,IU,1)
 IF (LSA.AND.K2.NE.1) OBS(K2)=WM
 IF (LSA.AND.K2.NE.1) OBS(K21)=WM
 IF (IP.GT.0)OBS(IP)=D0
 IF (IP1.GT.0.AND.LREPEC)OBS(IP1)=D0
END SUBROUTINE INHEAD

!%%%
```

```
 USE m_geocol_data, ONLY : INUMR,NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,&
                           LSTNO=>LTERM,LTERMA,LTERMO,LOUTC,LTRAN,LNERNO,&
                           LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT
! RLATC added 2013-02-27.
 USE m_data,        ONLY : OLDB,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
 USE m_data,        ONLY : 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,LDIF
 USE m_geocol_data, ONLY : HQD,RLATD,SINLATD,COSLATD,RLONGD,SINLOND,COSLOND
 USE m_geocol_data, ONLY : VARNO,PPA,PPS,VG,HPK,DM,DA,WM,REJLEV,SGRE,ROTFIL,&
                           ERNAME,DNAME,FMT,NSTEP,NSTEPE,IDSAT,ILA,ILO,IKPREF,&
                           INZ,IO2,NPOBS0,NOUSE,ILAST,IPAMAX,NGR,NGRE,ICSYS, &
                           LMEAN,LSA,LADMU,LSTAT,LAREA,LZETA,LGRADI,LMENSI,&
                           LSIMH,LMEAN1,LINTRA,LGIGRS,LMEGR,LUSNGS,LSATAC,&
                           lsatp,lgrerr,lcOD,LEQP,LALLP,LGRP,LDEN,LPOTSD,&
                           LEQANG,LFILTE,LBIN,LSKIPL,LGRERS,LTILT,LSCALE,LINERT

IMPLICIT NONE

 LOGICAL ::       LFOR77,LINTER,LGRID,LNOUSE,&
                  LTEST,LFULLO,LORTH

 INTEGER ::       K,IJ,I,NWAR,NO2,ITRAC0,J

 REAL(KIND=8)  :: OBI(22),A0,&
                  DEGRAD,              CROT(3,3),CROT0(3,3),&
! CROT AND CROT0 ADDED 2013-02-27.
                  SINB,COSB,SINT,COST,AZCH,TAUCH,BETCH,TRTIME,QUAT(4)
!
! *************** INPUT (10) *********************************

! INPUT OF COORDINATES OF OBSERVATION OR PREDICTION POINTS AND CONTIN-
! GENTLY THE OBSERVED QUANTITIES AND THEIR STANDARD DEVIATIONS.
! THIS IS FOLLOWED BY INPUT OF PARAMETER IDENTIFICATION CODES IF
! LPARM IS TRUE AND LEQP IS FALSE, AND THE OBSERVATIONS ARE NOT
! SATELLITE ALTIMETRY OR CROSS-OVER DIFFERENCES (IKP = 11 OR 9).
!
  DEGRAD=PI/180.0D0
  LNOUSE=LF
  IJ = IANG*2+INO-1
!      IF (LGIGRS) IJ=8+IANG
! CHANGE 2003-03-05 AND 2004-02-06.
  IF (IANG.GT.4) IJ=13
! THIS PREPARES FOR X,Y,Z INPUT.
  IF (LUSNGS) IJ=14
!!!!! MAYBE  NOT CORRECT TO  RETURN TO HERE !!!!!!
!
  IF (INO.EQ.0) NO=NO+1
  IF (LFOR77.AND.(.NOT.LFORM).AND.IANG.LE.4) THEN
   IF (LINTER.AND.(.NOT.LIN4)) WRITE(6,*)' INPUT DATA RECORD, LSTOP'
! CHANGE 2013-04-22.
   IF (IANG.EQ.1) THEN
    IF (LIN4) READ(INZ,*,END=2039)NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,(OBI(I),I=1
,K1)
    IF (.NOT.LIN4) READ(5,*)NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,(OBI(I),I=1,K1),L
STOP
   ELSE
    IF (IANG.EQ.2) THEN
     IF (LIN4) READ(INZ,*,END=2039)NO,IDLAT,SLAT,IDLON,SLON,(OBI(I),I=1,K1)
     IF (.NOT.LIN4) READ(5,*)NO,IDLAT,SLAT,IDLON,SLON,(OBI(I),I=1,K1),LSTOP
    ELSE
! INPUT IN DECIMAL DEGREES, GRADES.
      IF (LIN4) THEN
        READ(INZ,*,END=2039)NO,SLAT,SLON,(OBI(I),I=1,K1)
      ELSE
        READ(5,*)NO,SLAT,SLON,(OBI(I),I=1,K1),LSTOP
      END IF
```



```
    END IF
  END IF
  GO TO 2030
 2039    NO=-1
 2030   CONTINUE
  ELSE
!
! IFORMAT WILL INPUT DATA IN VARIOUS FORMATS. (1991.08.30).
   IF ((.NOT.LOUTC.AND.LNEQ).AND.NWAR.LT.5) THEN
    WRITE(*,*)' WARNING13  LOUTC ',LOUTC
    NWAR=NWAR+1
   END IF
   CALL IFORMAT(NO,IJ,IANG,IKP,IKPREF,INZ,OBI,FMT,LMEGR,LSTOP,LOUTC)
!
  END IF
!
  IF (LNUOUT.AND.INO.GT.0.AND.NO.LT.0) THEN
! THIS IS TO ASSURE THAT STATION NUMBERS ARE OUTPUT IF END OF FILE
! IS MET IN THE INPUT (LABEL 2039).
   NO2=NO1-1
   NO2=MOD(NO2,6)+1
   WRITE(6,278)(INUMR(I),I=1,NO2)
  278   FORMAT(' ',6I11)
  END IF
!
  IF(ILA .GE. ILO)THEN
! CORRECTING INVERTED ORDER OF LAT. AND LONG.
   I = IDLAT
   IDLAT = IDLON
   IDLON = I
   I = MLAT
   MLAT = MLON
   MLON = I
   A0 = SLAT
   SLAT = SLON
   SLON = A0
  END IF
 IF (NO.LT.0) RETURN
!
! RAD CONVERTS FROM ANGULAR UNITS TO RADIANS.
 CALL RAD(IDLAT,MLAT,SLAT,RLATP,IANG)
! CONVERSION TO EAST LONGITUDE.
 IF (LWLONG.AND.IANG.LE.2) IDLON = -IDLON
 IF (LWLONG.AND.IANG.GT.2) SLON = -SLON
 CALL RAD(IDLON,MLON,SLON,RLONGP,IANG)
!
! INITIALISATION ADDED 2005-09-23.
 IF (LALLCO) THEN
  ALLREF = D0
  ALLGG  = D0
  ALLPRE = D0
  ALLCOL = D0
  ALLPR1 = D0
  ALLERR = D0
  ALLTRA = D0
 END IF
!
 IF (LSATP.AND.(.NOT.(LSATP.AND.LGRID.AND.NO.GT.1))) THEN
! ATTITUDE ANGLES FOR LGRID=TRUE ARE DEFINED EARLIER.
! CHECK THIS CCCCC
  IF (ISATP.EQ.3) THEN
   AZP  = 90.0D0
   BETP = D0
   TAUP = D0
   IF (LFULLO.AND.(.NOT.LZETA)) WRITE(*,9282)AZP,BETP,TAUP
  ELSE
   IF (ISATP.EQ.2) THEN
!
! ----------------- INPUT (10AA) -------------------------
```

```
!
! INPUT OF TILT, ROLL AND PITCH ANGLES IN DEC. DEGREES.
! AZP IS TRACK AZIMUTH FROM DUE NORTH COUNTED EAST (CONVENTION), RELATIVE
! TO ROTATING SOLID EARTH; BETP IS PITCH ANGLE POSITIVE
! FOR CLIMBING SATELLITE, TAUP IS POSITIVE FOR RIGHT ROLL.
    IF (LINTER.AND.(.NOT.LIN4)) WRITE(6,*)' INPUT TILT,PITCH,ROLL (DEG.)'
    READ(INZ,*,END=2040) AZP,BETP,TAUP
  282    FORMAT(3D17.9)
    IF (LFULLO.AND.(.NOT.LZETA)) WRITE(*,9282)AZP,BETP,TAUP
 9282    FORMAT(' ATTITUDE ANGLES (DEG.) ',3F11.5,/,3F11.5)
   END IF
  END IF
  IF (ISATP.LT.4) THEN
! AT THIS POINT THE ROTATION MATRIX SATROT(3,3)
! MUST BE ESTABLISHED.
   SAZP = SIN(AZP*DEGRAD)
   CAZP = COS(AZP*DEGRAD)
   SINB = SIN(BETP*DEGRAD)
   COSB = COS(BETP*DEGRAD)
   SINT = SIN(TAUP*DEGRAD)
   COST = COS(TAUP*DEGRAD)
! WE TREAT SATROT AS A MATHEMATICAL MATRIX. I.E. FIRST INDEX
! IS ROW NUMBER, SECOND INDEX IS COLUMN NUMBER.
! Cor 2002-09-27
   SATROT(1,1) =  SAZP*COSB
   SATROT(1,2) =  CAZP*COST+SINT*SINB*SAZP
   SATROT(1,3) = -CAZP*SINT+COST*SAZP*SINB
   SATROT(2,1) = -CAZP*COSB
   SATROT(2,2) =  SAZP*COST-SINT*SINB*CAZP
   SATROT(2,3) = -SAZP*SINT-COST*SINB*CAZP
   SATROT(3,1) = -SINB
   SATROT(3,2) =  SINT*COSB
   SATROT(3,3) =  COSB*COST
! CHECK OF CONSISTENCY.
   AZCH  = ATAN2(SATROT(1,1),-SATROT(2,1))
   BETCH = ATAN2(-SATROT(3,1),SATROT(1,1)/SIN(AZCH))/DEGRAD
   TAUCH = ATAN2(SATROT(3,2),SATROT(3,3))/DEGRAD
   AZCH  = AZCH/DEGRAD
!  IF (LFULLO) WRITE(*,9282)AZCH,AZP,BETP,BETCH,TAUP,TAUCH
  ELSE
! CHANGE 2004-08-31 AND 2004-10-20.
   IF (ISATP.EQ.4) THEN
!   READ(13,*)TRTIME,((SATROT(J,I),I=1,3),J=1,3)
    READ(13,*)TRTIME,((SATROT(I,J),I=1,3),J=1,3)
   ELSE
!   WRITE(*,*)' NOT FULLY IMPLEMENTED *** WARNING14 *** '
    READ(13,*)TRTIME,QUAT
! THE SUBROUTINE CONVERTS QUARTERNIONS TO A ROTATION MATRIX.
! ADDED 2005-02-10. CHANGE 2013-02-26. TRTIME ADDED.
    IF (ISATP.EQ.6) THEN
! HERE CONVERSION FROM GEOCENTRIC FRAME TO LOCAL EAST,NORTH,UP (RADIAL) FRAME.
! CHANGE 2013-03-04.
     COSLAP=COS(RLATC)
     SINLAP=SIN(RLATC)
     COSLOP=COS(RLONGP)
     SINLOP=SIN(RLONGP)
     CROT(1,1)=-SINLOP
     CROT(1,2)=-SINLAP*COSLOP
     CROT(1,3)= COSLAP*COSLOP
     CROT(2,1)=COSLOP
     CROT(2,2)=-SINLAP*SINLOP
     CROT(2,3)=COSLAP*SINLOP
     CROT(3,1)=D0
     CROT(3,2)=COSLAP
     CROT(3,3)=SINLAP
     IF (LFULLO) THEN
      write(*,*)' CROT ',CROT
! CHECK FOR ORTHOGONALITY. ADDED 2013-02-28.
      CROT0=D0
```



```
      DO i=1,3
       DO j=1,3
        DO k=1,3
         CROT0(I,J)=CROT0(I,J)+CROT(I,K)*CROT(J,K)
        end do
       end do
      END DO
! check of orthogonality.
      LORTH=LF
      do i=1,3
       do j=1,3
        if (j.eq.i) then
         LORTH=abs(CROT0(i,i)-D1).gt.1.0d-8.or.LORTH
        else
         LORTH=abs(CROT0(i,j)).gt.1.0d-8.or.LORTH
        end if
       end do
      end do
      if (LORTH) write(*,*)' WARNING: MISSING ORTHOGONALITY ',CROT0
     END IF
     CROT0=D0
    ELSE
     CROT=D0
    END IF
    CALL QUATMAT(QUAT)
    IF (LFULLO) THEN
     write(*,*)satrot
     call QUA_to_MAT(QUAT,SATROT)
     write(*,*)satrot
    END IF
! QUATMAT RETURNS THE EULER MATRIX ELEMENTS IN THE COMMON BLOCK
! 'ROT', IN THE MATRIX SATROT.
    IF (ISATP.EQ.6) THEN
!    SATROT=SATROT*CROT
     CROT0=SATROT
     if (lf) then
     SATROT=D0
     DO I=1,3
      DO J=1,3
       DO K=1,3
        SATROT(I,J)=SATROT(I,J)+CROT0(I,K)*CROT(K,J)
       END DO
      END DO
     END DO
     else
! CHANGE 2013-03-05.
      SATROT=MATMUL(CROT0,CROT)
     end if
     IF (LFULLO) write(*,*)' SATROT AFTER CROT ',satrot
! CHECK OF CONSISTENCY.
     SATROT1=TRANSPOSE(SATROT)
     AZCH  = ATAN2(SATROT1(1,1),-SATROT1(2,1))
     BETCH = ATAN2(-SATROT1(3,1),SATROT1(1,1)/SIN(AZCH))/DEGRAD
     TAUCH = ATAN2(SATROT1(3,2),SATROT1(3,3))/DEGRAD
     AZCH  = AZCH/DEGRAD
     AZP=AZCH
     TAUP=TAUCH
     BETP=BETCH
     IF (LFULLO) WRITE(*,9283)AZCH,BETCH,TAUCH
 9283 format(' AZ,BE,TAU',3f12.7)
    END IF
   END IF
   IF (ABS(NO-TRTIME).GT.2) THEN
    WRITE(*,*)' INCONSISTENCY TRAROT ',NO,TRTIME
    STOP
   END IF
   AZCH  = ATAN2(SATROT(1,1),-SATROT(2,1))
!  AZCH  = ATAN2(SATROT(1,1),-SATROT(2,1))/DEGRAD
   BETCH = ATAN2(-SATROT(3,1),SATROT(1,1)/SIN(AZCH))/DEGRAD
```

```
   TAUCH = ATAN2(SATROT(3,2),SATROT(3,3))/DEGRAD
   azch  = AZCH/DEGRAD
   IF (LFULLO) WRITE(*,*)' EULER ANG: ',AZCH,BETCH,TAUCH
! ADDed 2005-03-09 in order to be used in collocation.
   SAZP = SIN(AZP*DEGRAD)
   CAZP = COS(AZP*DEGRAD)
   SINB = SIN(BETP*DEGRAD)
   COSB = COS(BETP*DEGRAD)
   SINT = SIN(TAUP*DEGRAD)
   COST = COS(TAUP*DEGRAD)
   SATROT1(1,1) =  SAZP*COSB
   SATROT1(1,2) =  CAZP*COST+SINT*SINB*SAZP
   SATROT1(1,3) = -CAZP*SINT+COST*SAZP*SINB
   SATROT1(2,1) = -CAZP*COSB
   SATROT1(2,2) =  SAZP*COST-SINT*SINB*CAZP
   SATROT1(2,3) = -SAZP*SINT-COST*SINB*CAZP
   SATROT1(3,1) = -SINB
   SATROT1(3,2) =  SINT*COSB
   SATROT1(3,3) =  COSB*COST
   if (LFULLO) WRITE(*,*)' recalculation of SATROT from AZ,BE,TA ',SATROT1
  END IF
  IF (LFULLO.AND.(.NOT.LZETA)) WRITE(*,8282)SATROT
 8282  FORMAT(' SATROT ',3D16.9,/,'        ',3D16.9,/,&
  '        ',3D16.9)
! IF BETP OR TAUP ARE LARGER THAN 1 DEG., THEN FULL ROTATION IS NEEDED.
  IF ((ABS(BETP).GT.D1.OR.ABS(TAUP).GT.D1).AND.ISATP.EQ.1) WRITE(*,*)' *** WARNI
NG15 *** 3-D ROTATION NEEDED'
 END IF
!
 IF (LMEAN1) THEN
! MODIFIED 2002.10.08
  IF (LINTER.AND.(.NOT.LIN4)) THEN
   WRITE(6,*)' INPUT AZIMUTH IN DEGREES '
   IF (LLCOER.AND.ITMODE.EQ.1) WRITE(*,*)' FOLLOWED BY TRACK NUMBER '
  END IF
  IF (LLCOER.AND.ITMODE.EQ.1) THEN
   READ(INZ,*,END=2040) AZP, ITRACK
  ELSE
   IF (LDIF) THEN
! HERE WE READ THE COORDINATES OF THE SECOND POINT FORMING THE DIFFERENCE
! (GRACE).
    READ(INZ,*,END=2040)j,RLATD,RLONGD,HQD
    SINLATD = SIN(RLATD*DEGRAD)
    COSLATD = COS(RLATD*DEGRAD)
    SINLOND = SIN(RLONGD*DEGRAD)
    COSLOND = COS(RLONGD*DEGRAD)
   ELSE
    READ(INZ,*,END=2040) AZP
   END IF
   ITRACK = 0
  END IF
! CHANGE 1997-07-15: ITRACK(NO1+1) < 0 IS USED TO INDICATE ERROR-CORRELATION
! WITH DATA HAVING THE SAME ITIME-VALUE.
! THE USE OF NO1 IS INCORRECT.***********************************
  IF (LCOERR) ITRACE(NO1+1)=-ITRACK
  DEGRAD=PI/180.0D0
  SAZP = SIN(AZP*DEGRAD)
  CAZP = COS(AZP*DEGRAD)
  COSSTE = CAZP
  SINSTE = SAZP
 END IF
!
!     IF (LINSOL)
!    *CALL INSOL(NFILE,NBLO,NBLP,SNAME,BOUNDS,LOUTS,LERNO)
!     IF (LINSOL.AND.LOUTS) GO TO 2324
 IF (LAREA.AND.(RLATP.GT.RLAMAX.OR.RLATP.LT.RLAMIN).AND.(.NOT.LGRID)) THEN
  LNOUSE = LT
  RETURN
 END IF
```



```
 IF (LAREA) THEN
  IF (RLOMIN.GT.D0.AND.RLOMAX.GT.D0.AND.RLONGP.LT.D0) RLONGP = RLONGP+D2*PI
  IF (RLOMIN.LE.D0.AND.RLOMAX.LE.D0.AND.RLONGP.GT.D0) RLONGP = RLONGP-D2*PI
  IF (RLOMIN.LE.D0.AND.RLOMAX.GE.D0.AND.RLONGP.GT.180.0D0) RLONGP = RLONGP-D2*PI

  IF (RLOMIN.GT.RLONGP.OR.RLOMAX.LT.RLONGP) THEN
   LNOUSE = LT
   RETURN
  END IF
 END IF
 IF (LLCOER) THEN
  IF (ITMODE.EQ.1)THEN
   IF (LINTER.AND.(.NOT.LIN4)) WRITE(*,*) ' INPUT TRACK NUMBER '
    READ(5,*)ITRACK
  ELSE
   IF (ITMODE.EQ.2) THEN
    ITRACK=(NO-ITM0)/ITMOD+1
    IF (ITRACK.NE.ITRAC0) THEN
     WRITE(*,*)' NEW TRACK ',ITRACK
     ITRAC0=ITRACK
    END IF
   ELSE
    IF (ITMODE.EQ.3) THEN
! CHANGE 2003-03-20.
     IF (ABS(NO+IKP*1000000-ITOLD).GT.ITRGAP) THEN
      ITRACK=NO+IKP*1000000
!     WRITE(*,*)' NEW TRACK ',ITRACK
! CHANGE 2014-02-14.
      ITOLD=NO+IKP*1000000
!         IF ((.NOT.LPRED).AND.
!    *    LAREA.AND.(RLATP.LT.RLAMAX.AND.RLATP.GT.RLAMIN.AND.
!    *    RLONGP.LT.RLOMAX.AND.RLONGP.GT.RLOMIN))
!    *    WRITE(*,*)' NEW TRACK ',ITRACK
     END IF
    END IF
   END IF
  END IF
!      ITOLD=NO+IKP*1000000
! VARIABLE NAME  ITIME CHANGED TO ITRACE 2005-03-12 IN ORDER TO
! AVOID INFERFACE PROBLEM WITH PARAMETER ESTIATION.
!      ITRACE(NO1+1)=-ITRACK
 END IF

! NO1 COUNTS POINTS (WITH ONE OR TWO OBSERVATIONS).
 NO1 = NO1  + 1
 RETURN
 2040 NO=-1

END SUBROUTINE INP10

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE INPAR(IKP,NO,ITRACK,IC,N,NOX,NPOBS,NPAOLD,LNOUSE,LINTER,LIN4,LPRED,LD
PR,LWRSOL,LTEST,LONEQ,OBI,IOBS1,NPOINT)
!
! --------------- INPUT (10A) --------------------------------

! INPUT OF PARAMETER TYPES, IF PARAMETERS ARE GIVEN INDIVIDUALLY
! FOR EACH OBSERVATION. FOR DATA RELATED TO SATELLITE-ALTIMETRY
! WE USE THE REVOLUTION NUMBER(S) WHICH IS IMPLICITLY GIVEN BY THE
! OBSERVATION NUMBER, (CODES IKP=9 FOR CROSS-OVER DIFFERENCES AND
! IKP=11 FOR SEA-SURFACE HEIGHTS TREATED LIKE GEOID UNDULATIONS.)
! LAST CHANGE 2005-03-02.

 USE m_params, ONLY : NIPT,NIPCAT
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_data,        ONLY : IPTYPE,ITIME,ITIME0,NPARM,NPARM1
 USE m_geocol_data, ONLY : SFACT,FPERIO,IPACAT,ITROLD,ICODE,MAXPAR,MP,&
                           IPA,NCXLAS
```

```
 USE m_geocol_data, ONLY : VARNO,PPA,PPS,VG,HPK,DM,DA,WM,REJLEV,SGRE,ROTFIL,&
                           ERNAME,DNAME,FMT,NSTEP,NSTEPE,IDSAT,ILA,ILO,IKPREF,&
                           INZ,IO2,NPOBS0,NOUSE,ILAST,IPAMAX,NGR,NGRE,ICSYS, &
                           LMEAN,LSA,LADMU,LSTAT,LAREA,LZETA,LGRADI,LMENSI,&
                           LSIMH,LMEAN1,LINTRA,LGIGRS,LMEGR,LUSNGS,LSATAC,&
                           lsatp,lgrerr,lcOD,LEQP,LALLP,LGRP,LDEN,LPOTSD,&
                           LEQANG,LFILTE,LBIN,LSKIPL,LGRERS,LTILT,LSCALE,LINERT

IMPLICIT NONE

 LOGICAL ::      LNOUSE,LIN4,LPRED,LDPR,LWRSOL,LTEST,LONEQ,LINTER

 INTEGER ::      IKP,NO,NPAOLD,ITRACK,IC,N,&
                 NOX,NPNO,I,KK,NPOBS,&
                 IOBS1,NPOINT

 REAL(KIND=8) :: OBI(10)

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT
!
!COMMON /CPARM/SFACT(NIPCAT),FPERIO(10,2),IPTYPE(NIPT),IPACAT(3*NIPT),ITIME(NIPC
AT),ITIME0(NIPT),ITROLD,ICODE(10),NPARM,NPARM1,MAXPAR,MP,IPA,NCXLAS
!COMMON /CDEFCA/VARNO,PPA,PPS,VG,HPK,DM,DA,WM,REJLEV,SGRE(10),ROTFIL,ERNAME,DNAM
E(2),FMT(9),NSTEP,&
!               NSTEPE,IDSAT,ILA,ILO,IKPREF,INZ,IO2,NPOBS0,NOUSE,ILAST,IPAMAX,NG
R,NGRE(10),ICSYS,&
!               LMEAN,LSA,LADMU,LSTAT,LAREA,LZETA,LGRADI,LMENSI,LSIMH,LMEAN1,LIN
TRA,LGIGRS,LMEGR,&
!               LUSNGS,LSATAC,LSATP,LGRERR,LCOD,LEQP,LALLP,LGRP,LDEN,LPOTSD,LEQA
NG,LFILTE,LBIN,&
!               LSKIPL,LGRERS,LTILT,LSCALE,LINERT
!
! TRANSFERS VARIABLES FROM DEFDAT.
!
 IF (IKP.EQ.9 .OR. IKP.EQ.11.OR. IKP.EQ.13.OR.LGRADI) THEN
  IF (IKP.EQ.11.OR.LGRADI) THEN
   IF (PPA.GT.0.1D-10) THEN
    ITRACK=(NO-PPS)/PPA+IKP*100000
! CHANGE 2002-02-04 AND 2003-03-05.
   ELSE
    ITRACK=NO/10000+IKP*100000
   END IF
  END IF
! BIAS DET. FOR GRAVITY, ADDED 1997-07-17 BY CCT.
!      IF (IKP.EQ.13.AND.(.NOT.LLCOER)) THEN
  IF (IKP.EQ.13) THEN
   IF (LINTER.AND.(.NOT.LIN4)) WRITE(*,*) ' INPUT TRACK NO '
!       READ(INZ,*)ITRACK
! CHANGE 2005-09-23.
   ITRACK=IKP*100000+NO/10000000
  END IF
! CHANGE 2002-10-09 AND 2003-03-05.
  IF ((IKP.EQ.13.OR.IKP.EQ.15.).AND.PPA.GE.0.0001) ITRACK=ITRACK+10
  IF (IKP.NE.9) THEN
   IF (ITRACK.EQ.ITROLD) THEN
! CHANGE 2004-07-02.
    IF (.NOT.LPRED) IPA=IPA-MP
   ELSE
    IF (ITROLD.GT.0) THEN
     IPACAT(ILAST) = IC
     IPACAT(ILAST+1)=ABS(MP)
     ILAST=IPA+1
     IPA=IPA+2
    END IF
    ITROLD=ITRACK
    IF (LTILT) THEN
     ITIME0(NPARM+1)=NO
     IF (LF) WRITE(*,*)' ITIME0 ',ITIME0(NPARM+1),NPARM+1
    END IF
```



```
   END IF
!
   DO I=1,MP
    IF (ICODE(I).EQ.1) THEN
! HERE IS DEFINED A BIAS PARAMETER FOR EACH TRACK.
     IPACAT(IPA+I)=ITRACK
    ELSE
     IF (ICODE(I).EQ.2) THEN
! HERE IS DEFINED A TILT OF DRIFT PARAMETER FOR EACH TRACK.
      IF (N.GE.NIPCAT) THEN
       WRITE(*,*)' DIMENSION OF ARRAY ITIME EXCEEDED: STOP '
       STOP
      END IF
! ITIME HOLDS THE 'TIME' DIFFERENCE WITH ITIME0 AS THE ZERO POINT. THIS
! IS USED FOR TILT-DETERMINATION.
      ITIME(N+1)=NO-ITIME0(NPARM+1)
      IPACAT(IPA+I)=-ITRACK
     ELSE
      IF (ICODE(I).EQ.3) THEN
! HERE IS DEFINED A SCALE FACTOR PARAMETER FOR EACH TRACK.
! THE PARAMETER IS IDENTIFIED WITH THE TRACK NUMBER PLUS A CONSTANT
! TO BRING IT INTO THE INTERVAL BETWEEN 10 AND 2500. CHANGE 2006-03-20.
       IPACAT(IPA+I)= (NO-PPS)/PPA+IKP*100
! HERE WE USE THAT IKP WILL NOT BE .LT. 10. SO THERE IS ROOM FOR TRACK
! NUMBERS UP TO 1000.
       IF ((NO-PPS)/PPA.GT.1000) WRITE(*,*) ' WARNING16, TRACK NUMBER TOO LARGE
',(NO-PPS)/PPA
! still to be changed
       SFACT(N+1)=OBI(IOBS1)
! here we should not use the input value, but the reference value
! either from the normal potential or the spherical harmonic expansion
! used. remark 2006-04-17.
      ELSE
! HERE IS DEFINED A FOURIER COEFFICIENT PARAMETER FOR EACH TRACK.
! WE USE THAT ICODE(I) .GT. 3 AND IKP .GE. 10.
       IPACAT(IPA+I)= (NO-PPS)/PPA+IKP*400*ICODE(I)
       write(*,*)ipacat(IPA+I),IPA,I,NO,PPS,ICODE(I)
       ITIME(N+1)=NO
      END IF
! here something must be done for icode > 3 C
     END IF
    END IF
   END DO
!
  END IF
  IF (LF) WRITE(*,*)' IPA ',IPA,MP,ITRACK,ITROLD,NPARM
  NPOBS=NPOBS+1
!
  IF (IKP.EQ.9) THEN
! DIFFERENCES WHEN LDPR=TRUE.
   IF (LDPR) THEN
! WE ADD 1100000, IN ORDER TO AVOID CONFLICT WITH RESERVED PARAMETER
! IDENTIFICATION CODES.
    IPACAT(IPA+1)=NOX+1100000
    IPACAT(IPA+2)=NO+1100000
   ELSE
    IPACAT(IPA+1)=NO/100000+1100000
    IPACAT(IPA+2)=NO-(IPACAT(IPA+1)-10)*100000+1100000
   END IF
   CALL PARCAT(LT,NPNO)
! THE CALL PARAMETER IN PARCAT IS TRUE, BECAUSE WE DO NOT ACCEPT NEW
! PARAMETERS DEFINED ONLY FROM CROSS-OVER DIFFERENCES. CONSEQUENTLY
! CROSS-OVER DIFFERENCES MUST BE INPUT AFTER POINT OBSERVATIONS, IF
! LALLP IS TRUE.
   IF (IPACAT(IPA+1).EQ.0) IPACAT(IPA+2)=0
   IF (IPACAT(IPA+2).EQ.0) IPACAT(IPA+1)=0
   IF (IPACAT(IPA+1).NE.0) RETURN
! NOUSE COUNTS CROSS-OVER DIFFERENCES NOT USED.
   NOUSE=NOUSE+1
```

```
   LNOUSE=LT
   RETURN
!       ELSE
!        CALL PARCAT(LALLP,NPNO)
!        IF (NPAOLD.NE.NPARM.AND.LTEST.AND.IPA.GT.3)
!     *  WRITE(6,276)NPARM,IPA,(IPACAT(KK),KK=(IPA-3),IPA+4),&
!     *  IPTYPE(NPARM)
!        NPAOLD=NPARM
!        ITIME0(NPARM+1)=NO
!        NPOBS=0
!        NPOBS0=NPOBS0+1
!        RETURN
  END IF
 ELSE
  IF (LINTER) WRITE(6,*)' INPUT PARAMETER CODES'
  READ(INZ,*)(IPACAT(IPA+1),I=1,MP)
  IF (LWRSOL) WRITE(17,150)(IPACAT(IPA+I),I=1,MP)
  150  FORMAT(12I6)
  IF (MP.LT.4) WRITE(6,170)MP,(IPACAT(I+IPA),I=1,MP)
  IF (MP.GT.3) WRITE(6,171)MP,(IPACAT(I+IPA),I=1,MP)
  170  FORMAT(/' OBSERVATIONS CONTRIBUTE TO/DEPEND ON ',I3,&
  ' PARAMETERS',3I6)
  171  FORMAT(/' OBSERVATIONS CONTRIBUTE TO/DEPEND ON ',I3,&
  ' PARAMETERS',3I6,(/,12I6))
 END IF
!
 CALL PARCAT(LALLP,NPNO)
! this must be changed 2004-12-29.
 IF (LPRED.AND.MP.EQ.2) THEN
  ITIME(N+1)=NO-ITIME0(NPNO+1)
  IF (LF.AND.LONEQ) WRITE(6,9611) NO,ITIME0(NPNO+1),NPNO
 9611  FORMAT(' NO,ITIME,NPNO',3I10)
 END IF
 IF (.NOT.(NPARM.EQ.NPAOLD.OR.(.NOT.(LGRADI.OR.IKP.EQ.11)).OR.MP.GT.2.OR.LPRED))
 THEN
! CHANGE N -> NPOINT 2005-03-02.
  IF (NPOINT.NE.0.AND.MP.EQ.1.AND.NPOBS.EQ.1) WRITE(6,244) NPARM,NO
! NPOINT COUNTS NUMBER OF OBSERVATIONS CONTRIBUTING TO PARAMETERS.
  NPOINT=NPOINT+1
  NPAOLD=NPARM
  IF (LF) WRITE(*,*)' MP,IPA= ',MP,IPA
!
  IF (MP.NE.1) THEN
   ITIME0(NPARM+1)=NO
   ITIME(N+1)=0
   IF (N.NE.0.AND.NPOBS.EQ.MP) WRITE(6,244)NPARM,ITIME0(NPARM-1)
  244   FORMAT(' PARAMETER NO',I5,', LAST OBSNO ',I10, ' NOT WELL',&
   ' DETERMINED. ******')
! ERROR IF MORE THAN ONE DATASET, DETECTED 1995.10.28 BY CCT.
!         IF (N.EQ.0.OR.NPOBS.GE.MP) GO TO 2045
   IF (NPOBS0.NE.0.AND.NPOBS.LT.MP) THEN
    WRITE(6,245)NPARM,ITIME0(NPARM-1)
  245    FORMAT(' TOO FEW OBSERVATIONS FOR PARAMETER NO ',I4,&
    ', OBSNO=',I9)
    IPACAT(IPA+3)=IPACAT(IPA+1)
    IPACAT(IPA+4)=IPACAT(IPA+2)
    IPACAT(IPA-1)=N+1
    IPACAT(IPA)=0
    IPACAT(ILAST)=N
    IPACAT(IPA+2)=IPACAT(ILAST+1)
    ILAST=IPA+1
    IPA=IPA+2
    NPARM=NPARM-4
    CALL PARCAT(LALLP,NPNO)
    IF (NPAOLD.NE.NPARM.AND.LTEST.AND.IPA.GT.3) WRITE(6,276)NPARM,IPA,(IPACAT(KK
),KK=(IPA-3),IPA+4),&
    IPTYPE(NPARM)
! CHANGE 2003-03-18, AND 2004-06-21.
!     *    (IPTYPE(KK),KK=(NPARM-4),NPARM)
```



```
  276    FORMAT(8I9)
    NPAOLD=NPARM
    ITIME0(NPARM+1)=NO
   END IF
   NPOBS=0
   NPOBS0=NPOBS0+1
  END IF
 END IF

END SUBROUTINE INPAR

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE TRAPOT(LNPOT,LREPEC,LSPHER,LADDBP,LFULLO,POT00,RB,REF,REF0,UREF0,OBI,
H,HPP,RRE,SU,SU8,VREF)
! THE SUBROUTINE CALCULATES EFFECT OF DATUM-SHIFT AND THE
! CONTRIBUTION FROM A SHE. MOVED FROM MAIN PROGRAM 2004-11-27.
! REVISED 2005-12-06 AND 2009-01-15.

 USE m_params,      ONLY : MAXO,NSAT,NIPT,NIPCAT,NROOT,NNSU
 USE m_geocol_data, ONLY : COSSTE,SINSTE,COSSTN,SINSTN,FILTER,SATROT
 USE m_geocol_data, ONLY : E21,AX1,LNGR,GREF,LNCOL,LADBPR
 USE m_geocol_data, ONLY : ALLREF,ALLGG,ALLCOL,ALLPRE,ALLTRA,ALLPR1,ALLERR,&
                           ALLVAR,ALLIN,LALLCO
 USE m_data,        ONLY : NO,LCOMP,LMDD,OBS,LRESOL,LGRID,LNEWD
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN

 USE m_data,        ONLY : BSIZE,ANDEX,PRETAP,PREDP,HCZERO,NCZERO

 USE m_geocol_data, ONLY : B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,  &
                           WOBS,ISAT,SINLOP,COSLOP,BSIZEN,BSIZEE,COSLAP, &
                           SINLAP,RLONGP,RP,CAZP,SAZP,HP,RLATP,ICZERO,   &
                           NI,NR,IKP,ISATP,NOBLK,LONECO,LNKSIP,LNETAP,   &
                           LDEFVP,LOBSST,LSTART
 USE m_geocol_data, ONLY : HQD,RLATD,SINLATD,COSLATD,RLONGD,SINLOND,COSLOND
 USE m_data,        ONLY : LDIF

 USE m_geocol_data, ONLY : FG,FJ,LP,OMEGA2,IORDER
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,DC,HCMAX,KCI
 USE m_geocol_data, ONLY : CCR,SIGMAX,CCV,NC1,NC2,LOCAL,LSUM
 USE m_data,        ONLY : IPTYPE,ITIME,ITIME0,NPARM,NPARM1
 USE m_geocol_data, ONLY : SFACT,FPERIO,IPACAT,ITROLD,ICODE,MAXPAR,MP,&
                           IPA,NCXLAS
 USE m_geocol_data, ONLY : INUMR,NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,&
                           LSTNO=>LTERM,LTERMA,LTERMO,LOUTC,LTRAN,LNERNO,LK30,LK
31,&
                           LWRSOL,LSTOP,LK2EQ4,LNUOUT
 USE m_data,        ONLY : OLDB,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
 USE m_data,        ONLY : 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
 USE m_geocol_data, ONLY : VARNO,PPA,PPS,VG,HPK,DM,DA,WM,REJLEV,SGRE,ROTFIL,&
                           ERNAME,DNAME,FMT,NSTEP,NSTEPE,IDSAT,ILA,ILO,IKPREF,&
                           INZ,IO2,NPOBS0,NOUSE,ILAST,IPAMAX,NGR,NGRE,ICSYS, &
                           LMEAN,LSA,LADMU,LSTAT,LAREA,LZETA,LGRADI,LMENSI,&
                           LSIMH,LMEAN1,LINTRA,LGIGRS,LMEGR,LUSNGS,LSATAC,&
                           lsatp,lgrerr,lcOD,LEQP,LALLP,LGRP,LDEN,LPOTSD,&
                           LEQANG,LFILTE,LBIN,LSKIPL,LGRERS,LTILT,LSCALE,LINERT
 USE m_geocol_data, ONLY : X,Y,Z,XY,XY2,DISTO,DIST2
 USE m_geocol_data, ONLY : DZERO,ROOT
 USE m_geocol_data, ONLY : C20IN,G1,G2,CM3,CMM2,CM1
 USE m_geocol_data, ONLY : SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS3,EPS2,EPS1,DL,AX2,E
22

IMPLICIT NONE

 LOGICAL ::      LREPEC,LNPOT,LSPHER,LTEST,LFULLO,LADDBP
```

```
!
 INTEGER ::      I,J,IKA,MB,MA

 REAL(KIND=8) :: POT00,OBI(22),GREFI,H, REF,REF1,REF2,&
                 REF3,RGRAV,RG(3,3),CU,POT,GPOTDR,&
                 REF0,CY,SY,TANLAP,TAGLAP,UREF0,DUDY,DUDX,SINLO,COSLO,RLATS,&
                 RLONGS,RJ,DGM,COSLA,SINLA,REFI,COSLO1,COSLA1,RB,&
                 REFM,RLATP1,VREF(3),SU1,RRE,G2R(3,3),G2S(3,3),GLAP,GP,DGI,&
                 POTDIF,HPP,DLATP

 REAL*16,      DIMENSION(NNSU) :: SU
 REAL(KIND=8), DIMENSION(NNSU) :: SU8

!COMMON /CMCOV/CCI(24),CCR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),&
! HCMAX,CCV(2,2),DC(36),KCI(37),NC1,NC2,LOCAL,LSUM
! CHANGE TO 2200 2010-11-24.

!COMMON /SQROOT/DZERO,ROOT(NROOT)
! SQUARE-ROOT TABLE USED IN GPOTDR.

!COMMON /GPOTC0/C20IN,G1(3),G2(3,3),CM3,CMM2,CM1
! C20IN HOLDS C20, G1 THE FIRST DERIVATIVES, G2 THE SECOND DERIVATIVES.

! COMMON VARIABLES USED IN GPOTDR, SETCM ,LOADCM, PRED AND CXPARM.
!COMMON /GRAVCC/FG(30),FJ(30),LP(2),OMEGA2,IORDER

!COMMON /ITRANC/SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS3,EPS2,EPS1,DL,AX2,E22
! DATUM SHIFT PARAMETERS.

!COMMON /CPARM/SFACT(NIPCAT),FPERIO(10,2),IPTYPE(NIPT),IPACAT(3*NIPT),ITIME(NIPC
AT),ITIME0(NIPT),ITROLD,ICODE(10),NPARM,NPARM1,MAXPAR,MP,IPA,NCXLAS

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT
! COMMON CONSTANTS D0=0.0D0  ETC.

!COMMON /BIPAR/OLDB(4),CNR,GMP,AX,NMAX,II,IOBS,IOBSR,N1,NIR,MAXC,MAXC1,MAXC2,N,I
C,NT,IDIMC,IDIMCN,MAXBLT,JR,ISO,LPARAM,LPRED,LNEQ,LNEQ8,LNEWSO

!COMMON /CDEFCA/VARNO,PPA,PPS,VG,HPK,DM,DA,WM,REJLEV,SGRE(10),ROTFIL,ERNAME,DNAM
E(2),FMT(9),NSTEP,&
!                NSTEPE,IDSAT,ILA,ILO,IKPREF,INZ,IO2,NPOBS0,NOUSE,ILAST,IPAMAX,NG
R,NGRE(10),ICSYS,&
!               LMEAN,LSA,LADMU,LSTAT,LAREA,LZETA,LGRADI,LMENSI,LSIMH,LMEAN1,LINT
RA,LGIGRS,LMEGR,&
!                LUSNGS,LSATAC,LSATP,LGRERR,LCOD,LEQP,LALLP,LGRP,LDEN,LPOTSD,LEQA
NG,LFILTE,LBIN,LSKIPL,&
!               LGRERS,LTILT,LSCALE,LINERT
! TRANSFERS VARIABLES FROM DEFDAT.

!COMMON /CHEAD/IA,IB,IH,IP,IT,IA1,IB1,IP1,IT1,IC1,IC11,K1,IOBS1,IOBS2,ITE,ITE1,I
ITE,IITE1,IIP,IIP1,IIE,IIE1,INO,LPOT,LKM,LTERRC,LPOTIN

!COMMON /OUTC/INUMR(12),NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,LTERMA,LTERMO,L
STNO,LOUTC,LTRAN,LNERNO,LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT

!COMMON /DAT/LNEWD,LRESOL,LGRID
! /DAT/ TRANSFERS LOGICAL VARIABLES TO THE SUBROUTINE ITRAN.

!COMMON /EUCL/X,Y,Z,XY,XY2,DISTO,DIST2

!COMMON /CALLCO/ALLREF(3,3),ALLGG(3,3),ALLCOL(3,3),ALLPRE(3,3),ALLTRA(3,3),ALLPR
1(3,3),ALLERR(3,3),ALLVAR(3,3),ALLIN(3,3),LALLCO

 REF1=VREF(1)
 REF2=VREF(2)
 REF3=VREF(2)
 IF (.NOT.LRESOL) THEN
! LNEWD IS TRUE IF A NEW COORD.SYST. IS USED.
! write(*,*)' LNEWD ',LNEWD
```



```
  IF (.NOT.LNEWD) THEN
   CALL EUCLID(COSLAP,SINLAP,COSLOP,SINLOP,D0,E21,AX1)
!
   IKA=IKP
   CALL TRANS(SINLAP,COSLAP,RLATP,SINLOP,COSLOP,RLONGP,H,IKA,IT)
   IF (LINTRA)OBS(IT)=-OBS(IT)
   IF (LINTRA.AND.LREPEC)OBS(IT1)=-OBS(IT1)
!
   IF (.NOT.LMENSI) THEN
! WE ARE JUMPING FROM OUTSIDE A BLOCK TO INSIDE. WARNING.
! MUST BE CORRECTED.
    REF = D0
!    write(*,*)'IORDER,LNGR,LNKSIP,LMDD ',IORDER,LNGR,LNKSIP,LMDD
    IF (.NOT.(LNGR.AND.(IORDER.NE.2.OR.LNKSIP).AND.(.NOT.LMDD))) THEN
! LMDD IS TRUE IF SECOND ORDER MEASURED DERIVATIVES.
     CALL EUCLID(COSLAP,SINLAP,COSLOP,SINLOP,H,E22,AX2)
     REF= RGRAV(15,IKP,REF1,REF2,REF3,SINLAP,H,RG,CU,SU1,LSATP)
! CHANGE 1990.10.19 BY CCT CALL OF AXV ADDED .
     VREF(1)=REF1
     VREF(2)=REF2
     VREF(3)=REF3
     IF (LSATP.AND.(.NOT.LGRADI)) THEN
      IF (LZETA) THEN
       IF (LFULLO) WRITE(*,*)' REF ',REF
      ELSE
       IF (LFULLO) WRITE(*,1151)VREF
       CALL AXV(SATROT,VREF)
       IF (LFULLO) WRITE(*,1151)VREF
 1151       FORMAT(' VREF ',3D15.6)
      END IF
     END IF
!
     IF (LALLCO) THEN
      ALLREF(1,1)=VREF(1)
      ALLREF(1,2)=VREF(2)
      ALLREF(1,3)=VREF(3)
     END IF
!
     IF (LGRP) THEN
      OBS(IT) = (REF0-REF)*1.0D5
      IF (H.GE.1.0D4) THEN
! IF H.GE. 10 KM, THEN IT MAY BE MEANINGLESS TO CHANGE FROM ONE
! GRAVITY FORMULA TO ANOTHER. ADDED 2000-07-05 BY CCT.
       WRITE(*,*) ' ** WARNING17 *** DATUM SHIFT MAY BE ERRONEOUS '
       WRITE(*,*) ' IKA,IT,REF0,REF,OBS(IT) ',IKA,IT,REF0,REF,OBS(IT)
      END IF
     END IF
     IF (IORDER.EQ.2) OBS(IT)=(REF0-REF)*1.0D9
     IF (LPOTSD.AND.LGRP) OBS(IT) = OBS(IT)-13.7E0
    END IF
   END IF
  END IF
!
  IF (.NOT.(LNPOT.OR.LCOD)) THEN
   IF (.NOT.LPOTIN.OR.NO1.EQ.1) THEN
    IF (.NOT.LMENSI) THEN
!
     IF (IKP.EQ.2) THEN
      CALL EUCLID(COSLAP,SINLAP,COSLOP,SINLOP,D0,E22,AX2)
      POT=GPOTDR(-NMAX,0,SU,SU8)
      H=H+(POT-UREF0)/REF
     END IF
!
     IF ((IKP.NE.13.OR.LNEWD).AND.(.NOT.LDEN)) CALL EUCLID(COSLAP,SINLAP,COSLOP,
SINLOP,H,E22,AX2)
     IF (LDEN) CALL EUCLID(COSLAP,SINLAP,COSLOP,SINLOP,HP,D0,RRE)
! CORRECTION 1987.10.10. EARLIER RGRAV WAS NOT CALLED WHEN
! ONLY ETA (IKP=17 OR 4) WAS EVALUATED.
!    write(*,*)'IORDER,LNGR,LNKSIP,LMDD ',IORDER,LNGR,LNKSIP,LMDD
```

```
     IF (LZETA.OR.LDEFVP.OR.IKP.EQ.14.OR.IKP.EQ.15.OR.LSATP) THEN
! CHANGE 2013-03-17.
      REF=RGRAV(15,IKP,REF1,REF2,REF3,SINLAP,H,RG,CU,SU1,LSATP)
! CHANGE 2002-06-25.
      VREF(1)=REF1
      VREF(2)=REF2
      VREF(3)=REF3
      IF (LSATP.AND.(.NOT.LGRADI)) THEN
       IF (LZETA) THEN
        IF (LFULLO) WRITE(*,*)' REF ',REF
       ELSE
        IF (LFULLO) WRITE(*,1151)VREF
        CALL AXV(SATROT,VREF)
        IF (LFULLO) WRITE(*,1151)VREF
       END IF
      END IF
     END IF
     POT= GPOTDR(-NMAX,IORDER,SU,SU8)
     IF (LSATP.AND.LPOT) THEN
      IF (LZETA) THEN
       IF (LFULLO) WRITE(*,*)POT
      ELSE
       IF (LFULLO) WRITE(*,153)G1
  153       FORMAT(' G1 ',3D16.5)
       CALL AXV(SATROT,G1)
       IF (LFULLO) WRITE(*,153)G1
      END IF
     END IF
!
     IF (IORDER.EQ.2) THEN
      GREF= SQRT(REF3**2+REF2**2)
      CY = -REF3/GREF
      SY = -REF2/GREF
      IF (LFULLO.AND.LNCOL) THEN
       WRITE(*,*)' DDU '
       WRITE(6,358)RG
      END IF
      IF (LSATP.AND.LGRADI)THEN
       CALL ATBA(TRANSPOSE(SATROT),RG,RG)
!      RG=MATMUL(MATMUL(SATROT,RG),TRANSPOSE(SATROT))
       GO TO (7615,7626,7626,7626,7626,&
       7620,7621,7622,7623,7624,7625),(IKP-14)
 7615       REF=RG(3,3)
       GO TO 7626
 7620       REF=RG(3,2)
       GO TO 7626
 7621       REF=RG(1,3)
       GO TO 7626
 7622       REF=RG(2,2)
       GO TO 7626
 7623       REF=RG(2,1)
! CHANGED 2013-03-05.
!7623       REF=RG(2,1)*D2
       GO TO 7626
 7624       REF=RG(1,1)
       GO TO 7626
! DATA CODE=25, DIFFERENCE DDU/DXX-DDU/DYY.
 7625       REF=RG(1,1)-RG(2,2)
 7626       CONTINUE
       IF (LFULLO) WRITE(*,*)' REF ',REF
! CONVERSION TO EU.
!           REF=REF*1.0D9
!           IF (LMEGR) OBS(2)=OBS(2)-REF
!           IF (LMEGR) OBS(2)=(OBS(2)-REF)*1.0d9
! CORRECTION 2009-01-15. ONLY CONVERSION OF REF TO EOTVOES UNITS.
       IF (LMEGR) OBS(2)=OBS(2)-REF*1.0d9
      END IF
      IF (LFULLO.AND.LNCOL.AND.LSATP.AND.(.NOT.LZETA)) WRITE(6,358) RG
```



```
  358      FORMAT(3E19.11)
      IF (LSATP.AND.LGRADI.AND.LPOT) THEN
       DO I=1,3
        DO J=1,3
         G2R(I,J)=G2(I,J)
        END DO
       END DO
       IF (LFULLO.AND.LNCOL.AND.LSATP.AND.(.NOT.LZETA)) THEN
        WRITE(*,261)OBS(2),REF*1.0D9
  261   format(' obs, ref*1.0d9 ',2f11.5)
        WRITE(6,359)POT00,(G1(I),I=1,3),((G2(I,J),I=1,3),J=1,3),&
        SQRT(G1(1)**2+G1(2)**2+G1(3)**2)
       END IF
! CHANGE 2013-03-05.
       G2S=MATMUL(MATMUL(SATROT,G2R),TRANSPOSE(SATROT))
       CALL ATBA(TRANSPOSE(SATROT),G2R,G2)
       IF (LFULLO.AND.LNCOL.AND.LSATP.AND.(.NOT.LZETA)) THEN
        WRITE(*,367)G2S
 367    format(' G2S ',3D19.11,/,5X,3D19.11,/,5X,3D19.11)
        WRITE(*,*)' ROTATED '
        WRITE(6,359)POT00,(G1(I),I=1,3),((G2(I,J),I=1,3),J=1,3),&
        SQRT(G1(1)**2+G1(2)**2+G1(3)**2)
  359        FORMAT(' V=',E19.11,' DV, DDV',/4(3E19.11/),' G ',F10.7)
       END IF
! LAPLACE CHECK, 2013-02-28 AND 2013-03-15.
       if (abs(G2(1,1)+G2(2,2)+G2(3,3)).GT.1.0D-7) then
        write(*,*)' WARNING LAPLACE ',G2(1,1)+G2(2,2)+G2(3,3)
       end if
      END IF
! ADDED 2005-09-21 TO PERMIT OUTPUT OF ALL COMPONENTS.
      IF (LALLCO) THEN
       DO I=1,3
        DO J=1,3
         ALLREF(I,J)=RG(I,J)
         ALLGG(I,J)=G2(I,J)
! ADDED 2006-07-17. CHANGED 2008-09-24.
!             IF (LNCOL) THEN
          IF (LMEGR) THEN
           ALLCOL(I,J)=G2(I,J)*1.0D9
! CONVERSION TO EOTVOS.
          ELSE
           ALLCOL(I,J)=(G2(I,J)-ALLREF(I,J))*1.0D9
          END IF
! ADDED 2006-08-20.
          IF (LCOMP) ALLCOL(I,J)=ALLIN(I,J)-ALLCOL(I,J)
!             END IF
        END DO
       END DO
      END IF
     ELSE
      IF (ABS(REF2).LT.0.1D-16) THEN
       CY=D1
       SY=D0
      ELSE
       CY = - REF1/REF2
       SY = - REF/REF2
      END IF
!
! ADDED 2005-09-21 TO PERMIT OUTPUT OF ALL COMPONENTS.
      IF (LALLCO) THEN
       DO I=1,3
        ALLREF(I,1)=VREF(I)
        ALLGG(I,1)=G1(I)
! ADDED 2006-07-17.
        IF (LNCOL) THEN
         IF (LMEGR) THEN
          ALLCOL(I,1)=G1(I)*1.0D-5
! CONVERTING TO MGAL.
         ELSE
```

```
          ALLCOL(I,1)=(ALLGG(I,1)-ALLREF(I,1))*1.0D-5
         END IF
! ADDED 2006-08-20.
         IF (LCOMP) ALLCOL(I,1)=ALLIN(I,1)-ALLCOL(I,1)
        END IF
       END DO
      END IF
     END IF
!
     I=IKP
     IF (LSATP) THEN
      CY=D1
      SY=D0
     END IF
! CHANGE 1996.05.08 BY CCT.
     IF (LDEFVP.AND.LMEGR.AND.(.NOT.LGRADI)) THEN
! GLAP IS GEOCENTRIC LATITUDE AT H=0. CY,SY ARE NOW
! COS AND SIN OF DIFFERENCE GEOCENTRIC AND GEODETIC LATITUDE:
      TANLAP=SINLAP/COSLAP
      TAGLAP=(D1-E22)*TANLAP
      GLAP=ATAN(TAGLAP)
      CY=COS(GLAP-RLATP)
      SY=SIN(GLAP-RLATP)
     END IF
! WHEN A SATELLITE ORIENTED FRAME IS USED, THE SPHERICAL COORDINATE
! SYSTEM IS USED. OTHERWISE THE COORDINATE SYSTEM IS ORIENTED WITH
! RESPECT TO THE NORMAL GRAVITY VECTOR.
     IF (IKP.GT.25.AND.IKP.LT.36)I=IKP-10
     GO TO (7006,7007,7008,7009,7008,7010,7010,7010,7010,&
     7016,7006,7007,7007,7011,7011,7008,7009,7012,7013,&
     7012,7013,7023,7014,7024,7015),I
 7006     IF (.NOT.LSATP) THEN
      OBS(IP) = (POT-REF)/REF2
!       HEIGHT ANOMALY, ZETA.
     ELSE
      OBS(IP) = POT-REF
! ANOMALOUS POTENTIAL IN M**2/S**2. (ADDED 2002-09-23).
     END IF
     G1(1)=POT
     GO TO 7010
!
 7007     GP =  SQRT(G1(1)**2+G1(2)**2+G1(3)**2)
     IF (.NOT.LSATP) OBS(IP) = (GP-REF)*1.0D5
     IF (LSATP) THEN
      OBS(IP) = (G1(3)-VREF(3))*1.0D5
     ELSE
      OBS(IP) = (GP-REF)*1.0D5
     END IF
!     GRAVITY DISTURBANCE OR ANOMALY (IN ELLIPSOIDAL APPROXIMATION).
!     WHEN LSATP IS TRUE, WE USE THE GRAVITY DISTURBANCE IN THE
!     DIRECTION OF THE 3. AXIS.(1990.11.27).
     IF (IKP.EQ.13) OBS(IP)=OBS(IP)-2*(POT-REF1)/DISTO*1.0D5
!     GRAVITY ANOMALY IN SPHERICAL APPROXIMATION.
     GO TO 7010
!
! CORRECTION JULY 1989 SUBSCRIPTS 1 AND 2 IN G1 AND G2 INTERCHANGED.
! CORRECTION AUG. 92, UNITS FOR LSAT ARE MGAL.
 7008     IF (.NOT.LSATP)  THEN
      OBS(IP) = ((REF-G1(2))*CY-(REF1-G1(3))*SY)*RADSEC/REF2
      DUDY=G1(2)*CY-G1(3)*SY
     ELSE
! PREPARATION FOR LSAT, REF2 IS NOT THE CORRECT QUANTITY. 1990.11.30.
!     OBS(IP)= -(G1(2)-VREF(2))*1.0D5   ERROR 2000.03.31
      OBS(IP) = (G1(2)-VREF(2))*1.0D5
!     KSI, SEE REF(D), EQ. (72) AND (75).
     END IF
     IF (IKP.EQ.3 .OR. IKP.EQ.16) GO TO 7010
 7009     IF (.NOT.LSATP) OBS(IP1) = -G1(1)*RADSEC/REF2
     DUDX=G1(1)
```



```
!         IF (LSATP) OBS(IP1) = -(G1(1)-VREF(1))*1.0D5  ERROR 2000-03-31
     IF (LSATP) OBS(IP1) = (G1(1)-VREF(1))*1.0D5
!     ETA.
     GO TO 7010
!
! SECOND ORDER DERIVATIVES MUST BE TRANSFORMED, CF. REF(D), EQ.
! (73) - (75). CY AND SY ARE COS AND SIN OF THE ANGLE BETWEEN THE
! NORMAL GRAVITY VECTOR AND THE RADIUS-VECTOR.
 7011     IF (LSATP) THEN
      OBS(IP) = (G2(3,3)-RG(3,3))*1.0D9
     ELSE
      OBS(IP) = (G2(3,3)*CY*CY+2*CY*SY*G2(2,3)-REF)*1.0D9
!     D2T/DZ2, VERTICAL GRAVITY GRADIENT.
      IF (IKP.EQ.4) OBS(IP) = OBS(IP)-D2*((POT-REF1)/DIST2-(G1(3)-REF3)/DISTO)*1
.0D9
!     VERTICAL GRAVITY ANOMALY GRADIENT.
     END IF
     GO TO 7010
!
 7012     IF (.NOT.LSATP) THEN
      OBS(IP)= (-(G2(2,3)*CY*CY+(G2(2,2)-G2(3,3))*CY*SY)-REF)*1.0D9
     ELSE
      OBS(IP)= (G2(2,3)-RG(2,3))*1.0D9
     END iF
!     D2T/DXDZ, GRAVITY GRADIENT IN NORTHERN DIRECTION.
     IF (IKP.EQ.18 .OR. IKP.EQ.28) OBS(IP) = OBS(IP)+D3*(REF2-G1(2))*1.0D9/DISTO
!       GRAVITY ANOMALY GRADIENT IN NORTHERN DIRECTION.
     IF (IKP.EQ.18 .OR. IKP.EQ.20) GO TO 7010
!
 7013     IF (LSATP) THEN
      OBS(IP1) = (G2(1,3)-RG(1,3))*1.0D9
     ELSE
      OBS(IP1) = -(G2(1,3)*CY+G2(1,2)*SY)*1.0D9
!     D2T/DYDZ, GRAVITY GRADIENT IN EASTERN DIRECTION.
      IF (IKP.EQ.19 .OR. IKP.EQ.28) OBS(IP1) = OBS(IP1)-D3*G1(1)*1.0D9/DISTO
!     GRAVITY ANOMALY GRADIENT IN EASTERN DIRECTION.
     END IF
     GO TO 7010
!
 7023     IF (.NOT.LSATP) THEN
      OBS(IP) = (G2(2,2)*CY*CY-D2*G2(2,3)*CY*SY+G2(3,3)*SY*SY-REF)*1.0D9
     ELSE
      OBS(IP) = (G2(2,2)-RG(2,2))*1.0D9
!     D2T/DXDX
     END IF
     GO TO 7010
!
 7015     IF (LSATP) THEN
      OBS(IP) = (G2(1,1)-G2(2,2))*1.0D9-REF
     ELSE
      OBS(IP) = (G2(1,1)-G2(2,2)*CY*CY+2*G2(2,3)*CY*SY-REF)*1.0D9
!     D2T/DY2-D2T/DX2.
     END IF
     IF (IKP.EQ.25) GO TO 7010
!
 7014     IF (LSATP) THEN
      OBS(IP1)=   (G2(1,2)-RG(1,2))*1.0D9
!     OBS(IP1)= 2*(G2(1,2)-RG(1,2))*1.0D9
     ELSE
      OBS(IP1)=   (G2(1,2)*CY-G2(1,3)*SY)*1.0D9
!     OBS(IP1)= 2*(G2(1,2)*CY-G2(1,3)*SY)*1.0D9
!     2*D2T/DXDY. CHANGED 2013-03-05.
     END IF
     GO TO 7010
!
 7024     IF (.NOT.LSATP) THEN
      OBS(IP) = (G2(1,1)-REF)*1.0D9
     ELSE
      OBS(IP) = (G2(1,1)-RG(1,1))*1.0D9
```

```
     END IF
!     D2T/DYDY.
     GO TO 7010
!
 7016     OBS(IP)=POT*RP**(-KCI(32))
! DENSITY CONTRAST.
 7010     CONTINUE
    ELSE
!
! MEAN VALUE COMPUTATION.
 7062     COSLA=COSLAP
     SINLA=SINLAP
     RLATS=RLATP
     RLONGS=RLONGP
     REFM=D0
     DGM=D0
     RJ = D0
!    write(*,*)' LDIF ',Ldif
     IF (LDIF) Then
      NSTEP=2
      NSTEPE=1
     END IF
     DO MA=1,NSTEP
! CORRECTION 1996.12.19 BY CCT.
      IF ((.NOT.LMEAN1).OR.MA.EQ.1) THEN
       COSLO=COSLOP
       SINLO=SINLOP
      END IF
      IF (LDIF.AND.MA.EQ.2) THEN
       H=HQD
       COSLO=COSLOND
       SINLO=SINLOND
       COSLA=COSLATD
       SINLA=SINLATD
      END IF
      CALL EUCLID(COSLA,SINLA,COSLO,SINLO,H,E22,AX2)
      REFI=RGRAV(15,IKP,REF1,REF2,REF3,SINLA,H,RG,CU,SU1,&
             LSATP)
      VREF(1)=REF1
      VREF(2)=REF2
      VREF(3)=REF3
      IF (LDIF) THEN
       REFM=REFM+REFI*(-1**MA)
      ELSE
       REFM=REFM+REFI
      END IF
      DO MB=1,NSTEPE
       IF (MB.GT.1) CALL EUCLID(COSLA,SINLA,COSLO,SINLO,H,E22,AX2)
       POT=GPOTDR(-NMAX,1,SU,SU8)
       GREFI= SQRT(G1(1)**2+G1(2)**2+G1(3)**2)
       IF (LDIF) THEN
        DGI=pot-ref1
        IF (MA.EQ.1) THEN
!        write(*,*)MA,MB,DGI
         DGM=DGI
        end if
       ELSE
        DGI=(GREFI-REFI-D2*(POT-REF1)/DISTO)*1.0D5
       END IF
       IF (.NOT.LMEAN1) THEN
! CORRECTION DEC. 1996 BY CCT.
        IF (LEQANG) THEN
         DGM=DGM+DGI*COSLA
         RJ=RJ+COSLA
        elSE
         DGM=DGM+DGI
        END IF
        COSLO1=COSLO
        coslo=cOSLO*COSSTE-SINLO*SINSTE
```



```
        SINLO=SINLO*COSSTE+COSLO1*SINSTE
       ELSE
        IF (LDIF) THEN
         IF (MA.EQ.2) THEN
          DGM=DGM-DGI
!         write(*,*)ma,Dgi,dgm
! THIS COMPUTES THE DIFFERENCE.
         END IF
        ELSE
         DGM=DGM+DGI*FILTER(MA)
        END IF
       END IF
      END DO
!
       IF (LMEAN1) THEN
        IF (.NOT.LDIF)&
        CALL PAZIM(RLATS,RLONGS,COSLA,SINLA,COSLO,SINLO,CAZP,SAZP,&
        COSSTN,SINSTN,LTEST)
       ELSE
        COSLA1=COSLA
        COSLA=COSLA*COSSTN+SINLA*SINSTN
        SINLA=SINLA*COSSTN-COSLA1*SINSTN
       END IF
!      write(*,*)' 6405 ',ma,dgm
      END DO
! END MA LOOP.
!
      OBS(IT)=(REF0-REFM/5)*1.0D5
      IF (LPOTSD) OBS(IT)=OBS(IT)-13.7
! CORRECTION DEC. 1996 BY CCT.
      IF (LEQANG.AND.(.NOT.LMEAN1)) THEN
       OBS(IP)=DGM/RJ
      ELSE
       IF (LDIF) THEN
        OBS(IP)=DGM
!       write(*,*)' 6416 ',IP,DGM
       ELSE
        OBS(IP)=DGM/(NSTEP*NSTEPE)
       END IF
      END IF
     END IF
    END IF
!
    IF (LPOTIN.AND.NO1.NE.1) THEN
     OBS(IP)=OBI(IIP)+OBS(IT)
     IF (LREPEC) OBS(IP1)=OBI(IIP1)+OBS(IT1)
    ELSE
     IF (LPOTIN.AND.NO1.EQ.1) THEN
      POTDIF= ABS(OBS(IP)-OBI(IIP)-OBS(IT))
      IF (LDEFVP.AND.(POTDIF.GT.0.1).OR.LGRP.AND.(POTDIF.GT.2.0).OR.LZETA.AND.(P
OTDIF.GT.0.1)) WRITE(6,273)OBS(IP),OBI(IIP)
  273      FORMAT(' *** WARNING18 *** COMPUTED=',F8.2,', INPUT=',F8.2)
     END IF
    END IF
!
    IF (LADDBP) OBS(IB) = OBS(IB)+OBS(IP)
    IF (LADBPR) OBS(IB1) = OBS(IB1)+OBS(IP1)
    IF (LDEN.AND.IH.EQ.0) HP=OBS(1)
!   write(*,371)(obs(i),i=1,5)
!371 format(' 6483 obs ',5f10.3)
   END IF
  END IF
! CHANGE 2004-07-09.
  IF (.NOT.LSPHER) THEN
   CALL EUCLID(COSLAP,SINLAP,COSLOP,SINLOP,HP,E22,AX2)
  END IF
!
! NO SPHERICAL APPROXIMATION, 2001-09-21.
 IF (.NOT.LSPHER) THEN
```

```
! CHANGE 2004-08-11.
  IF (.NOT.LSATP.AND.(.NOT.LZETA)) ISATP=0
! CHANGED 2013-05-03 TO INDICATENO ROTATION.
! IF (.NOT.LSATP.AND.(.NOT.LZETA)) ISATP=3
! THIS ASSIGNMENT INDICATES FULL ROTATION, USING THE UNIT MATRIX.
  IF (DISTO.LT.RB) THEN
   WRITE(*,*)' POINT INSIDE BJERHAMMAR SPHERE, H::= 10000 M '
   WRITE(*,*)HP,DISTO,RB
   HPP=0.0D0
  ELSE
   HPP=DISTO-RE
! CHANGE 2003-06-02.
   IF (IH.NE.0) HP=HPP
  END IF
!
  COSLAP=XY/DISTO
  SINLAP=Z/DISTO
  RLATP1=ATAN2(Z,XY)
! DLATP IS GEOCENTRIC LATITUDE MINUS GEODETIC LATITUDE. MORE CORRECT
! IS THE ANGLE BETWEEN THE NORMAL GRAVITY FIELD VECTOR, SEE RGRAV.
  DLATP=RLATP1-RLATP
  IF (ABS(DLATP).GT.0.1) THEN
   WRITE(*,*)' ERROR, RLATP,P1 = ',RLATP,RLATP1
  ELSE
! CORRECTION 2003-04-06.
   RLATP=RLATP1
  END IF
 ELSE
  HPP=HP
 END IF

END SUBROUTINE TRAPOT

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE COPRED(PREDCO,LOC_PW2,OBI,WM,SM,&
                  KP,NPARM,                               NPRED,NPRED1, &
                  LERNO,       LREPEC,LTCOV,LSATAC,LADBA,LTNB,LTEB,LCOMP,  &
                  LFOUND,LCOD,LMENSI,LSATP,LGRERS,LGRERR,LSA,LERCOV,       &
                  LWAIT,NLO,      LMTEST)
!                 KP,NPARM,NFILE,NPARM1,NERRM,NGRE,NGRERR,NPRED,NPRED1, &
!                 LERNO,LINSOL,LREPEC,LTCOV,LSATAC,LADBA,LTNB,LTEB,LCOMP,  &
!                 LFOUND,LCOD,LMENSI,LSATP,LGRERS,LGRERR,LSA,LERCOV,LEROUT,&
!                 LWAIT,NLO,LGRID,LMTEST)
! THE SUBROUTINE COMPUTES PREDICTED VALUES FROM COLLOCATION
! AND LOOKS FOR GROSS ERRORS. MOVED FROM MAIN BODY 2004-12-01.
! LAST UPDATE 2013-01-07.

 USE m_params,      ONLY : MAXO,NSAT,NEQFIM,NDIMC,NISIZE,NCRW,NNBL,NALLCO
 USE m_geocol_data, ONLY : C,NCAT,ISZE,NBL,MAXBL,ISIZE,MAXCM,MI1,MI2,MMAXB,&
                           MAXFIL,NEQFI,NEQFMA,MAXBNE,DNANE,LNBL1,SATROT
 USE m_geocol_data, ONLY : ALLREF,ALLGG,ALLCOL,ALLPRE,ALLTRA,ALLPR1,ALLERR,&
                           ALLVAR,ALLIN,LALLCO
 USE m_data,        ONLY : OBS,LDIF
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN

 USE m_data,        ONLY : BSIZE,ANDEX,PRETAP,PREDP,HCZERO,NCZERO

 USE m_geocol_data, ONLY : B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,  &
                           WOBS,ISAT,SINLOP,COSLOP,BSIZEN,BSIZEE,COSLAP, &
                           SINLAP,RLONGP,RP,CAZP,SAZP,HP,RLATP,ICZERO,   &
                           NI,NR,IKP,ISATP,NOBLK,LONECO,LNKSIP,LNETAP,   &
                           LDEFVP,LOBSST,LSTART
 USE m_geocol_data, ONLY : FG,FJ,LP,OMEGA2,IORDER
 USE m_geocol_data, ONLY : INUMR,NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,&
                           LSTNO=>LTERM,LTERMA,LTERMO,LOUTC,LTRAN,LNERNO,&
                           LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT
 USE m_geocol_data, ONLY : IMAX1,IMAX1R
 USE m_data,        ONLY : OLDCOV,S,SR,AAI,AAR,IS,IPX,LCO1,NBOLD
```



```
 USE m_data,        ONLY : OLDB,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
 USE m_data,        ONLY : 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
 USE m_geocol_data, ONLY : HQD,RLATD,SINLATD,COSLATD,RLONGD,SINLOND,COSLOND
 USE m_geocol_data, ONLY : HQE,RLATE,SINLATE,COSLATE,RLONGE,SINLONE,COSLONE

IMPLICIT NONE

 LOGICAL ::      LERNO,LFOUND,LCOD,LMENSI,LSATP,LGRERS,LGRERR,LTCOV,LERCOV,&
                 LSATAC,LREPEC,LSA,LADBA,LTNB,LTEB,LCOMP,&
                 LWAIT,LCTEST,LMTEST

 INTEGER ::      KP,NPARM,J,I,NERR,&
                 NPRED,NPRED1,I61,I62,IBSS,I63,NLO,NSTART,NREL

 REAL(KIND=8) ::                         LOC_PW2,VAR,OBI(22),WM,SM(2200),&
                ! USED IN ERROR COVARIANCE COMPUTATION. ADDED 200-08-09.
                 CPQ,COVPQ,PREDCO(16),PREDCP(16)
!                CPQ,COVPQ,PREDCO(16),PREDCP(16),OERR

 REAL(KIND=8), DIMENSION(MAXO,6) :: ALLCOV

!CHARACTER(LEN=128) :: OLDCOV

!COMMON /CHEAD/IA,IB,IH,IP,IT,IA1,IB1,IP1,IT1,IC1,IC11,K1,IOBS1,IOBS2,ITE,ITE1,I
ITE,IITE1,IIP,IIP1,IIE,IIE1,INO,LPOT,LKM,LTERRC,LPOTIN

!COMMON /OUTC/INUMR(12),NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,LTERMA,LTERMO,L
STNO,LOUTC,LTRAN,LNERNO,LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT

!COMMON /BIPARC/OLDCOV(2),S,SR,AAI,AAR,NBOLD,IS,IPX,IMAX1,IMAX1R,LCO1

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT

!COMMON /BIPAR/OLDB(4),CNR,GMP,AX,NMAX,II,IOBS,IOBSR,N1,NIR,MAXC,MAXC1,MAXC2,N,I
C,NT,IDIMC,IDIMCN,MAXBLT,JR,ISO,LPARAM,LPRED,LNEQ,LNEQ8,LNEWSO

!COMMON /GRAVCC/FG(30),FJ(30),LP(2),OMEGA2,IORDER

!COMMON /CALLCO/ALLREF(3,3),ALLGG(3,3),ALLCOL(3,3),ALLPRE(3,3),ALLTRA(3,3),ALLPR
1(3,3),ALLERR(3,3),ALLVAR(3,3),ALLIN(3,3),LALLCO

 LCTEST=.FALSE.
 NERR=0
 IBSS=NEQFI(1,2)
! write(*,*)' JR ',JR
!
  CALL PRED(S ,AAI,IS,    ISO,II,IOBS,N1,IMAX1,LT   ,LF  ,LERNO,LTCOV,LSATAC,LWA
IT,NPRED)
      !PRED(SS,AAI,IS,ISP,ISO,II,IC  ,NC,IMAX1,LPRED,LBST,LCST ,LTCOV,LSATAC,LWA
IT,NPRED)
! if (lpred) write(*,*)' 6196 NI C(NI) ',NI,C(NI)
  MAXC1=NI
  MAXC2=MAXC1+N1
  LFOUND=LF
!
  IF (LERNO) THEN
   LOC_PW2 = D0
   IF (.NOT.LCOD) THEN
    IF (LDIF) THEN
     HQE=HQD
     RLATE=RLATD
     SINLATE=SINLATD
     COSLATE=COSLATE
     RLONGE=RLONGD
     SINLONE=SINLOND
```

```
     COSLONE=COSLOND
    END IF
    LOC_PW2=VAR(SM,IS,KP,S,AAI,HP,IMAX1,LMENSI,COSLAP,SINLAP,&
    COSLOP,SINLOP,LSATP,SATROT)
   END IF
!  IF (LERCOV) THEN
! STORAGE OF COORDINATES OF PREDICTION POINTS IN PREDCO, 2005-08-03.
    PREDCO(1)=RLATP
    PREDCO(2)=COSLAP
    PREDCO(3)=SINLAP
    PREDCO(4)=RLONGP
    PREDCO(5)=COSLOP
    PREDCO(6)=SINLOP
    PREDCO(7)=HP
    IF (LSATP) THEN
     DO I61=1,3
      DO I62=1,3
       PREDCO(7+(I61-1)*3+I62)=SATROT(I61,I62)
      END DO
     END DO
    END IF
   IF (LERCOV) WRITE(20,REC=NPRED+1)PREDCO,OBS
!  END IF
   IF (.NOT.LALLCO) THEN
    IF (LERCOV) THEN
     READ(20,REC=I63)PREDCP
     CPQ=COVPQ(SM,IS,KP,S,AAI,IMAX1,LMENSI,LSATP,PREDCO,&
     PREDCP)
     IF (NPARM.GT.0) THEN
      C(NI)=-CPQ
     ELSE
      C(NI)=CPQ
     END IF
    ELSE
     I63=(NPRED-1.0d0)/IBSS
!    write(*,*) ' 6040 NPRED,I63 ',NPRED,I63,(NPRED-I63*IBSS)*N1,LOC_PW2
     C((NPRED-I63*IBSS)*N1)=LOC_PW2
    END IF
!   LCTEST=.TRUE.
    IF (LCTEST.OR.LTCOV) WRITE(*,*)' copred v: NPRED, NI ',NPRED,NI,C(NI)
!   NI=NI+1
    IF (NPARM.GT.0) THEN
!    C(NI)=-LOC_PW2
    ELSE
! CHANGE 2012-04-17.
     C(NI)=LOC_PW2
    END IF
     if (LTCOV) write(*,*)' 6055 NI,LOC_PW2 ', NI,LOC_PW2
    NI=NI+1
   END IF
   IF (LCTEST) WRITE(*,*)' copred w: NPRED, NI ',NPRED,NI,LOC_PW2
! STORAGE OF COVARIANCES.
   IF (LALLCO) THEN
    IF (LERNO) THEN
! ADDED 2012-05-12.
     WRITE(*,*)' LALLCO AND LERNO NOT IMPLEMENTED '
     STOP
    END IF
    NLO=0
    NSTART=MAXC2
! MAXC2 IS THE SUBSCRIPT OF THE LAST ROW OF THE RIGHT-HAND SIDE.
    NPRED1=(IORDER+2)*(IORDER+1)/2
    I63=-N-1
    J=0
    NPRED=NPRED1
    DO I62=1,IORDER+1
     DO I=1,I62
      J=J+1
      NREL=0
```



```
      NOBLK=1
       DO I61=1,N
        NREL=NREL+1
        IF (NREL.GE.MAXO) THEN
         NREL=1
         READ(15,REC=NOBLK)ALLCOV
         NOBLK=NOBLK+1
        END IF
        I63=I63+1
        C(NSTART+I63)=ALLCOV(NREL,J)
       END DO
       IF (J.GT.1) THEN
        DO I61=1,J-1
         I63=I63+1
         C(NSTART+I63)=D0
        END DO
       END IF
       I63=I63+1
       C(NSTART+I63)=ALLVAR(I,I62)
       IF (LCTEST) WRITE(*,5511)I63,(C(NSTART+I63-I61),I61=0,J)
 5511       format(i6,6d12.5)
       IF (MOD((N+J),IBSS).EQ.0) THEN
        NBL(MI1)=N+J
        IF (LCTEST) WRITE(*,*)' restore called copred, j,N ',J,N
        CALL RESTORE_CH(N+J,0,lf,lt,LERCOV,LMTEST)
        NI=1
        NSTART=-I63
       END IF
     END DO
    END DO
   ELSE
    NPRED1=NPRED1+1
    IF (MOD(NPRED+1,IBSS).EQ.1.AND.NPRED.NE.1) THEN
     if (lf) write(*,5512)(C(I61),I61=1,10)
 5512 format(6f14.11)
!    IF (LCTEST)WRITE(*,*)' 6135 restore called copred, N1, NPRED ',&
     IF (LF)WRITE(*,*)' 6135 restore called copred, N1, NPRED ',&
     N1,NPRED
!    CALL RESTORE_CH(NPRED-IBSS+1,   0,LF,LT,LERCOV,LMTEST)
! CHANGE 2013-01-17.
     CALL RESTORE_CH(NPRED-IBSS+1,N1-1,LF,LT,LERCOV,LMTEST)
    END IF
    I=1
   END IF
   LCTEST=.FALSE.
!
  END IF
!
 IF (LREPEC) THEN
  IF (LERNO) THEN
! THIS IS OBSOLETE 2012-10-05.
! FIRST WE MOVE THE SECOND COLUMN OF THE RIGHT HAND SIDE INTO THE POSITION
! OF THE FIRST COLUMN.
!  DO J = 1, N
!   C(MAXC+J) = C(MAXC2+J)
!  END DO
!  C(MAXC2) = LOC_PW2
! changed  2012-10-05.
   C(MAXC2) = LOC_PW2
!  IF (NPARM.GT.0) C(MAXC2) = -C(MAXC2)
   NPRED1=NPRED1+1
   NPRED=NPRED+1
!  write(*,*)' 6224 PW2 ,MAXC2: NPRED ',C(MAXC2),MAXC2,NPRED
   IF (MOD(NPRED+1,IBSS).EQ.1.AND.NPRED.NE.1) THEN
    IF (LF)WRITE(*,*)' 6161 restore called copred, N1, NPRED ',&
!   IF (LCTEST)WRITE(*,*)' 6161 restore called copred, N1, NPRED ',&
    N1,NPRED
    CALL RESTORE_CH(NPRED-IBSS+1,N1-1,LF,LT,LERCOV,LMTEST)
   END IF
```

```
   NI=1
  END IF
!
  OBS(IA1) = PRETAP
  IF (LADBA) OBS(IB1) = OBS(IB1)+OBS(IA1)
  IF (LTNB) OBS(IU1) = OBS(IB1)-OBS(IT1)
  IF (LTEB) OBS(IU1) = -OBS(IT1)
  IF (LCOMP) OBS(13) = OBS(12)-OBS(IU1)
!
  IF (LGRERS.OR.LGRERR) THEN
   IF (LSA) OBI(IIE1)=ABS(WM)
  END IF
 END IF
!
 OBS(IA) = PREDP
!write(*,*)' 6779 COPRED IA, PREDP ',IA,PREDP
 IF (LADBA) OBS(IB) = OBS(IB)+OBS(IA)
 IF (LTNB) OBS(IU) = OBS(IB)-OBS(IT)
 IF (LTEB) OBS(IU) =-OBS(IT)
 IF (LCOMP) OBS(3) = OBS(2)-OBS(IU)
!write(*,*)' 6562 PR,OBS2345,IU,IA,IB ',PREDP,OBS(2),OBS(3),OBS(4),&
! OBS(5),IU,IA,IB
!write(*,*)' from copred, npred,lrepec= ',npred,lrepec
END SUBROUTINE COPRED

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE INCOV(LINTER,RB)
! PROGRAMMED BY C.C.TSCHERNING, GEOPHYSICAL INSTITUTE, UNIVERSITY
! OF COPENHAGEN, DENMARK.
! LAST UPDATE: 2001-09-21 BY CCT.
! THIS MODULE READS COVARIANCE FUNCTION PARAMETERS, CREATES NECESSARY
! TABELS FOR THE EVALUATION OF THE COVARIANCE FUNCTION.

 USE m_params, ONLY : MAXO,NSAT
 USE m_data,   ONLY : LC1,LC2,LCREF
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN

 USE m_data,        ONLY : BSIZE,ANDEX,PRETAP,PREDP,HCZERO,NCZERO

 USE m_geocol_data, ONLY : B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,  &
                           WOBS,ISAT,SINLOP,COSLOP,BSIZEN,BSIZEE,COSLAP, &
                           SINLAP,RLONGP,RP,CAZP,SAZP,HP,RLATP,ICZERO,   &
                           NI,NR,IKP,ISATP,NOBLK,LONECO,LNKSIP,LNETAP,   &
                           LDEFVP,LOBSST,LSTART
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,DC,HCMAX,KCI
 USE m_geocol_data, ONLY : CCR,SIGMAX,CCV,NC1,NC2,LOCAL,LSUM
 USE m_geocol_data, ONLY : STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE, COST2Q,SINT2
Q
 USE m_geocol_data, ONLY : DXX,NUM,VARI,SCALE,SCALE2,INN,INV
 USE m_geocol_data, ONLY : INUMR,NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,&
                           LSTNO=>LTERM,LTERMA,LTERMO,LOUTC,LTRAN,LNERNO,&
                           LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT
 USE m_geocol_data, ONLY : IMAX1,IMAX1R
 USE m_data,        ONLY : OLDCOV,S,SR,AAI,AAR,IS,IPX,LCO1,NBOLD
 USE m_data,        ONLY : OLDB,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
 USE m_data,        ONLY : 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

IMPLICIT NONE

 INTEGER ::                                KTYPE,IK,IK1,I,&
                 IMAX,IMIN,MODEL1,MODEL

 REAL(KIND=8) ::                       VG,     VAR,    &
                 SUMSIG,R,VARDG2,DR,RB,RB2,CVV,VZERO,A0
```



```
 LOGICAL ::      LINTER,LZERO,LMODEL,LOK,LMULTF

 REAL(KIND=8), DIMENSION(3,3)  :: LOC_SATROT
 REAL(KIND=8), DIMENSION(3,3,3,3)  :: COVX
 REAL(KIND=8), DIMENSION(2200) :: SM

 CHARACTER(LEN=128) :: PNAME

!COMMON /CMCOV/CCI(24),CCR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HCMAX,CCV(2,2),DC(36),KCI(37),NC1,NC2,LOCAL,LSUM
! CHANGE TO 2200 2010-11-24.
! COMMON VARIABLES USED IN COVAX.

!COMMON /OUTC/INUMR(12),NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,LTERMA,LTERMO,L
STNO,LOUTC,LTRAN,LNERNO,LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT
! IN OUTC ARE STORED SUBSCRIPTS OF VARIABLES TO BE OUTPUT AND LIMITS
! FOR DO-LOOPS IN OUTPUT. NOTE THAT OUTC OCCURS IN SUBROUTINES
! HEAD, COUT, CXPARM AND THE BLOCK DATA MODULE.

!COMMON /CHEAD/IA,IB,IH,IP,IT,IA1,IB1,IP1,IT1,IC1,IC11,K1,IOBS1,IOBS2,ITE,ITE1,I
ITE,IITE1,IIP,IIP1,IIE,IIE1,INO,LPOT,LKM,LTERRC,LPOTIN

!COMMON /COM2/DXX,NUM(70),VARI(32),SCALE,SCALE2,INN,INV
! USED BY COMPA, COMPARING OBSERVED AND PREDICTED QUANTITIES.

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT

!COMMON /CMEAQ/STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE,COST2Q,SINT2Q
! STEPSIZES USED WHEN CALCULATING MEAN VALUES.

!COMMON /BIPAR/OLDB(4),CNR,GMP,AX,NMAX,II,IOBS,IOBSR,N1,NIR,MAXC,MAXC1,MAXC2,N,I
C,NT,IDIMC,IDIMCN,MAXBLT,JR,ISO,LPARAM,LPRED,LNEQ,LNEQ8,LNEWSO

!COMMON /BIPARC/OLDCOV(2),S,SR,AAI,AAR,NBOLD,IS,IPX,IMAX1,IMAX1R,LCO1
! DATA USED WHEN STORING SOLUTIONS OR COVARIANCE FUNCTION ON
! BINARY FORM. (CHANGE MADE NOV 1986).

   IF (LCREF) GO TO 1000
!
! *************** INPUT (6) ***********************************
!
! INPUT OF THE INTEGER KTYPE DETERMINING TYPE OF DEGREE-VARIANCE
! MODEL USED FOR DEGREE-VARIANCES OF DEGREE GREATHER THAN IMAX
! (SEE BELOW). KTYPE MAY BE EQUAL TO 1, 2, OR 3, CORRESPONDING
! 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
! CHANGE 2006-02-20. INPUT OF LSUM.
 IF (KTYPE.LT.0) THEN
  LSUM=LT
  HCMAX=1.0D5
  KTYPE=-KTYPE
 ELSE
  LSUM=LF
  HCMAX=1.0D6
 END IF
!
 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
!
 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,')).')
!
!
! THIS IS THE RETURN POINT AFTER THE FIRST COLLOCATION STEP IF
! A SECOND STEP IS WANTED. NOTE THAT THE SAME DEGREE-VARIANCE
! MODEL MUST BE USED, BUT R,VARDG2 AND IMAX MAY BE CHANGED.
!
 1000 CNR = D0
 DO 1035 I = 1, 300
 1035 SIGMA(I) = D0
!
 SUMSIG = D0
 MAXC1 = 1
!
! *************** INPUT (7) **********************************
!
! INPUT OF CONSTANTS USED FOR THE FINAL SPECIFICATION OF THE DEGREE-VAR-
! IANCE MODEL:
! R      - RATIO BETWEEN THE BJERHAMMAR-SPHERE RADIUS AND THE
!          MEAN RADIUS OF THE EARTH (RE), IF POSITIVE. IF NEGATIVE IT
!          IS THE DEPTH TO THE BJERHAMMAR SPHERE IN KM.
! VARDG2 - VARIANCE OF GRAVITY ANOMALIES AT ZERO ALTITUDE.
! IMAX   - MAXIMAL DEGREE FOR EMPIRICAL DEGREE-VARIANCES.
! LZERO  - TRUE IF ALL EMPIRICAL DEGREE-VARIANCES ARE ZERO.
! LMODEL - TRUE IF THE DEGREE-VARIANCES ARE (SCALED) ERROR DEGREE
!          VARIANCES OBTAINED FROM A GEOPOTENTIAL MODEL. THE VALUES
!          ARE FOR TWO MODELS FOUND IN THE BLOCK DATA MODULE.
! THIS IS THEN FOLLOWED BY FURTHER DETAILS:
! (A) IF LMODEL TRUE, SPECIFICATION OF MODEL FOR THE VARIANCES.
! (B) IF LMODEL OR LZERO FALSE, THE EMPERICAL DEGREE-VARIANCES.
 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'            ,/&
             ' LMODEL - TRUE IF DEGREE-VAR. FROM PREDEFINED MODEL')
 READ(5,*,ERR=1111) R,VARDG2,IMAX,LZERO,LMODEL
  101 FORMAT(F8.5,1x,F7.2,I4,2L2)
 IF (LWRSOL) WRITE(17,101) R,VARDG2,IMAX,LZERO,LMODEL
 IF (R.GT.D1.OR.VARDG2.LT.D0) STOP
 IMAX1=IMAX+1
!
 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
! CHANGE 2006-02-20.
```



```
!     LSUM=LF
! CHANGE 2002.10.01
!     HCMAX = 1.0D6
!     HCMAX = 1.0D5
 WRITE(*,1911)HCMAX
 1911 FORMAT(' HCMAX = ',F10.1,' M. ')
! THIS IMPLIES, THAT THE POSSIBILITY FOR USING THE SUMMATION OF
! THE LEGENDRE SERIES IN COVAX CAN NOT BE USED. IF THIS IS NEEDED
! CHANGE LSUM,HCMAX AND THE DIMENSION OF SM (TO E.G. 2200).
 NC1=IMAX1
 NC2=3
 CALL COVAX(SM,IS)
!     CVV=VAR(SM,IS,3,S,AAI,D0,IMAX1,LF)
! ERROR DETECTED 1994.12.20 BY TK.
 CVV=VAR(SM,IS,3,S,AAI,D0,IMAX1,LF,COSLAP,SINLAP,COSLOP,SINLOP,&
     LF,LOC_SATROT)
 write(*,*)' cvv ',cvv
!
 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
!
! --------------- INPUT (7A) ---------------------------------
! 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
! MODEL .LE. 0 INDICATES THAT THE DEGREE-VARIANCES ARE INPUT FROM A
!         FILE (PNAME), AND FILE NAME MUST BE INPUT SUBSEQUENTLY.
! MODEL 1 IS A MODEL FOR THE ERROR IN RAPP'S 1978 SET
! MODEL 2 IS THE ERROR DEGREE-VARIANCES FOR RAPP'S 1981 SET,
! MODEL 3 IS THE ERROR DEGREE-VARIANCES FOR WENZELS GPM2 SET.
! MODEL 4 IS A LINEAR MODEL IN THE DEGREE, SO THAT FOR VG=1.0 THE
!    THE ERROR DEGREE VARIANCE IS EQUAL TO 1.0 FOR DEGREE 100.
! MODEL 5 IS A SIMILAR, BUT QUADRATIC MODEL.
! FOR MODEL 2 AND 3 THE INITIALIZATION TAKES PLACE IN THE
! BLOCK DATA MODULE. CONSEQUENTLY THESE MODES CAN ONLY BE USED
! WHEN THE VARIABLES, WITH WHICH THEY ARE EQUIVALENCED (RLAT),
! HAVE NOT BEEN USED FOR SOMETHING ELSE ALREADY.
 IF (MODEL.EQ.1.OR.IC.LT.1218) GO TO 1050
!
 WRITE(6,117)
  117 FORMAT(' **** ERROR DEGREE-VARIANCES DESTROYED IN',&
 ' FIRST COLLOCATION STEP **** ')
 STOP
!
 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)
! ADDITION 1999-05-17 BY CCT.
 LMULTF=(MODEL.LT.0)
 IF (LMULTF) THEN
  MODEL=0
 END IF
!
 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  continue
!SIGMA(I+1) = VG*DRAPP(I+1)
  GO TO 1043
 1053  continue
```

```
!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
! MODES 4 AND 5 ADDED 1988.11.30 BY CCT. MODEL 0, JAN. 1990.
 1043 CONTINUE
!
 IF (MODEL.NE.0) GO TO 1042
 IF (LINTER) WRITE(6,*)' INPUT NAME OF FILE WITH DEGR.VAR.'
 READ(5,'(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(A128)
 CLOSE(9)
! 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(5,*)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
!
 GO TO 1042
!
! --------------- INPUT (7B) ---------------------------------
! INPUT OF EMPIRICAL DEGREE-VARIANCES. NOTE, THAT PROBLEM MAY OCCUR
! IF FREE FORMAT IS USED, AND INPUT-DATA IS LINE NUMBERED. IN THIS
! 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)
! NOTE THAT THE DEGREE-VARIANCE OF ORDER I IS STORED IN SIGMA(I+1).
!
 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)
!
 1042 CONTINUE
 DO I = 3, IMAX1
  SIGMA0(IS+I)=SIGMA(I)
  SUMSIG = SUMSIG + SIGMA(I)
 END DO
 1040 IF (IMAX1+IS.LT.2200) GO TO 1002
 WRITE(6,108)
  108 FORMAT(' SUBSCRIPTS OF ARRAY SIGMA EXCEEDS ARRAY LIMIT, STOP.')
 STOP
!
 1002 AAI=(VARDG2-SUMSIG)*RB2*1.0D-8/CVV
 IF (AAI.LT.0.0D0) THEN
! ADDED 2006-01-20.
  WRITE(*,*)' VARDG2,SUMSIG,AAI,RB2,CVV ',VARDG2,SUMSIG,AAI,RB2,CVV
  WRITE(*,*)' WARNING19 AAI NEGATIVE '
  STOP
 END IF
```



```
 CCI(8)=AAI
 CALL COVAX(SM,IS)
 CALL COVBX(SM,LF,IS)
 CALL COVCX(SM,CVV,COVX,IS,LF)
 IF ( ABS(CVV-VARDG2).GT.0.1) WRITE(6,7464)CVV,VARDG2
 7464 FORMAT(' ** WARNING20 ** CVV,VARGD2= ',2E15.8)
!
! THE DEG.VAR. OF THE COVARIANCE FUNCTION OF THE ANOMALOUS POTENTIAL
! ARE STORED IN THE FIRST PART OF SIGMA (SUBSCRIPT 1 TO IMAX1R) FOR
! COLLOCATION I AND IN THE LAST PART (SUBSCRIPT IS=IMAX1R+3 TO
! IS+IMAX1) FOR COLLOCATION II.
!
  110 FORMAT(/' RATIO RB/RE                             =      ',F9.6,/&
 ' DEPTH TO BJERHAMMAR SPHERE (RB-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

END SUBROUTINE INCOV

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE IFORMAT(NO,IJ,IANG,IKP,IKPREF,INZ,OBI,FMT,LMEGR,LSTOP,LOUTC)
! PROGRAMMED BY CCT, LAST CHANGE 2005-09-06.

 USE m_geocol_data, ONLY : CLATD,SLON,RLATC,RLATCC
 USE m_geocol_data, ONLY : LFORM,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,NOX,RDI
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_data,        ONLY : 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
 USE m_geocol_data, ONLY : X,Y,Z,XY,XY2,DISTO,DIST2
 USE m_geocol_data, ONLY : SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS3,EPS2,EPS1,DL,AX2,E
22

IMPLICIT NONE

 REAL(KIND=8) :: GRME,GRCR,S,DH,RLAT1,COSLA,RLAT,TIMEP,XC,YC,ZC,XYC,&
                 DN,DC,SLATC,SINLAP
                !CLATD=GEOCENTRIC LATITUDE, RDI=DISTANCE FROM ORIGIN.

 INTEGER ::      IJ,I,INZ,NO,IKP,IHC,NSO,IKPREF,IANG,NC

 LOGICAL ::      LSTOP,LNFORM,LOUTC,LMEGR

 REAL(KIND=8), DIMENSION(22) :: OBI

 CHARACTER(LEN=128), DIMENSION(9) :: FMT

!-------------------------------------------------------------------------------
!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT

!COMMON /ITRANC/SINLA0,COSLA0,RLONG0,DSHIFT(7),AX2,E22

!COMMON /CHEAD/IA,IB,IH,IP,IT,IA1,IB1,IP1,IT1,IC1,IC11,K1,IOBS1,IOBS2,ITE,ITE1,I
ITE,IITE1,IIP,IIP1,IIE,IIE1,INO,LPOT,LKM,LTERRC,LPOTIN

 LNFORM=.NOT.LFORM
 GO TO(2024,2025,2026,2027,2028,2029,2028,2029,2330,2322,2223,&
 2029,2339,2340,2399),IJ
 2024 IF (LNFORM.AND.K1.EQ.2) READ(INZ,97)IDLAT,MLAT,SLAT,IDLON,MLON,SLON,OBI(1)
,OBI(2),LSTOP
   97 FORMAT(2(I4,I3,F5.2),2F8.2,L2)
```

```
 IF (LFORM) READ(INZ,FMT,END=2039) IDLAT,MLAT,SLAT,IDLON,MLON,SLON,(OBI(I),I=1,K
1),LSTOP
 GOTO 2030
 2025 IF (LNFORM) READ(INZ,61) NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,(OBI(I),I=1,K1
)
   61 FORMAT(I5,2(I4,I3,F6.2),5F8.2)
 IF (LFORM) READ(INZ,FMT,END=2039) NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,(OBI(I),I=
1,K1)
 GO TO 2030
 2026 IF(LNFORM) READ(INZ,95) IDLAT,SLAT,IDLON,SLON,(OBI(I),I=1,K1),LSTOP
   95 FORMAT(2(I4,F5.2),2F8.2,L2)
 IF (LFORM)READ(INZ,FMT,END=2039) IDLAT,SLAT,IDLON,SLON,(OBI(I),I=1,K1),LSTOP
 GOTO  2030
 2027 IF (LNFORM) READ(INZ,71)NO,IDLAT,SLAT,IDLON,SLON,(OBI(I),I=1,K1)
   71 FORMAT(I10,2(I4,F6.2),8F8.2)
 IF (LFORM) READ(INZ,FMT,END=2039) NO,IDLAT,SLAT,IDLON,SLON,(OBI(I),I=1,K1)
 GOTO  2030
 2028 IF (LNFORM) THEN
  READ(INZ,80)SLAT,SLON,(OBI(I),I=1,K1),LSTOP
 ELSE
  READ(INZ,FMT,END=2039)SLAT,SLON,(OBI(I),I=1,K1),LSTOP
 END IF
   80 FORMAT(2F10.5,2F8.2,L2)
 GO TO 2030
 2029 IF (LNFORM) READ(INZ,81)NO,SLAT,SLON,(OBI(I),I=1,K1)
   81 FORMAT(I10,2(F12.6,1X),7F8.2)
 IF (LFORM.AND.(IKP.NE.9)) THEN
  READ(INZ,FMT,END=2039)NO,SLAT,SLON,(OBI(I),I=1,K1)
 END IF
 IF (LFORM.AND.IKP.EQ.9) THEN
  READ(INZ,FMT,END=2039)NOX,NO,SLAT,SLON,(OBI(I),I=1,K1)
 END IF
 GO TO 2030
!
 2330 IF (IKP.EQ.26.AND.(.NOT.LTERRC)) READ(INZ,92) NO,IDLAT,MLAT,SLAT,IDLON,MLO
N,SLON,(OBI(I),I=1,5)
   92 FORMAT(I7,20X,2(2I3,F6.2,4X),F7.2,/,26X,2F7.2,2F6.2)
! GI STANDARD FOR DEFLECTIONS OF THE VERTICAL.
 IF (IKP.EQ.26.AND.LTERRC) READ(INZ,29) NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,(OBI(
I),I=1,7)
   29 FORMAT(I6,I5,I3,F6.2,3X,I4,I3,F6.2,5X,F11.1,2X,/,8F8.2)
!
 IF (IKP.EQ.15.OR.IKP.EQ.16) READ(INZ,94)NO,IDLAT,MLAT,SLAT,IDLON,&
 MLON,SLON,(OBI(I),I=1,3)
   94 FORMAT(I7,20X,2(2I3,F6.2,4X),F7.2,/,26X,F7.2,F6.2)
! GI STANDARD FOR KSI OR ETA SEPARATLY.
!
 IF (IKP.EQ.5.AND.IH.NE.0)READ(INZ,66)NO,IDLAT,MLAT,SLAT,IDLON,&
 MLON,SLON,(OBI(I),I=1,3)
   66 FORMAT(I5,2I3,F6.2,3X,I6,I3,F6.2,3X,F7.0,3X,2F8.3)
! SSG 3.70 FORMAT FOR DEFLECTIONS OF THE VERTICAL, NEW MEXICO.
 IF (IKP.EQ.5.AND.IH.EQ.0)READ(INZ,69)NO,IDLAT,MLAT,SLAT,IDLON,&
 MLON,SLON,OBI(1),OBI(2)
   69 FORMAT(I2,2(I4,I3,F6.2,3X),2F7.2)
! SSG 3.90 FORMAT, DEFLECTIONS, OHIO.
 IF (IKP.EQ.30)READ(INZ,67)NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,&
 (OBI(I),I=1,2)
   67 FORMAT(I4,1X,2(I4,I3,F5.1,3X),2F7.2,14X)
 IF (IKP.EQ.35)READ(INZ,68)NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,&
 (OBI(I),I=1,2)
   68 FORMAT(I4,1X,2(I4,I3,F5.1,3X),14X,2F7.2)
! SSG 3.90 STANDARD FORMAT FOR TORSION BALANCE COMPONENTS.
!
 GO TO 2030
 2399 READ(INZ,7979)NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON
 7979 FORMAT(8X,I2,I5,I3,F9.5,1X,I5,I3,F9.5,1X)
 GO TO 2030
!
 2322 IF (LTERRC.OR.LPOTIN) GO TO 2398
```



```
 READ(INZ,96)IDLAT,SLAT,IDLON,SLON,IHC,OBI(1),GRME,OBI(2),NSO,&
 NO,OBI(3),GRCR,LSTOP
 IF (IHC.EQ.3) OBI(1)=0.0D0
! CHANGE 1989.02.15 BY CCT IN ORDER TO AVOID INTEGER OVERFLOW FOR
! 32 BIT INTEGERS.
 IF (NSO.GT.99) NSO=MOD(NSO,100)
 NO=NSO*10000000+NO
 IF (LMEGR) OBI(2)=GRME+976000.0-GRCR
! INPUT OF GRAVITY DATA IN GI STANDARD FORMAT.
   96 FORMAT(1X,I2,F5.2,1X,I4,F5.2,1X,I1,F7.2,8X,F7.2,1X,F6.1,2X,&
 I3,5X,I8,F5.1,F6.2,1X,L1)
 GO TO 2030
!
 2398 IF (LOUTC) READ(INZ,8999)NO,IDLAT,SLAT,IDLON,SLON,(OBI(I),I=1,K1)
 8999 FORMAT(I10,2(I4,F5.1,3X),F9.1,2X,/,2X,5F8.2)
 IF (.NOT.LOUTC) READ(INZ,8997)NO,IDLAT,SLAT,IDLON,SLON,(OBI(I),I=1,K1)
 8997 FORMAT(I5,2(I4,F5.1,3X),F9.1,3X,6F8.2)
 GO TO 2030
!
 2223 IF(IKPREF.EQ.53)READ(INZ,87)NO,SLAT,SLON,OBI(1),OBI(2)
   87 FORMAT(I5,F10.6,1X,F10.6,1X,F6.1,1X,F5.1)
! GRAVITY DATA, NEW MEXICO FORMAT.
 IF (IKPREF.EQ.42)READ(INZ,88)NO,SLAT,SLON,OBI(1),OBI(2)
   88 FORMAT(I5,2(F9.4,1X),F7.1,1X,F6.1)
! SSG 3.90 FORMAT, GRAVITY DATA, OHIO.
 IF (IKPREF.EQ.45.AND.LTERRC)READ(INZ,28)NO,SLAT,SLON,(OBI(I),&
 I=1,K1)
   28 FORMAT(I6,1X,2(F11.6,3X),5X,F12.2,2X,/,8F8.2)
! FINNISH DEFLECTIONS OF THE VERTICAL.
 GO TO 2030
!
 2339 CONTINUE
! CHANGE 2004-05-12.
 IF (IANG.EQ.5) THEN
! INPUT OF CARTESIAN COORDINATES, INPUT MODE=5.
! CHANGE 2001-11-18 AND CORRECTION 2004-05-12 BY CCT.
! READ(INZ,*,END=2039)TIMEP,X,Y,(OBI(I),I=1,K1)
  IF (LFORM) THEN
   READ(INZ,FMT,END=2039)NO,X,Y,(OBI(I),I=1,K1)
  ELSE
   READ(INZ,*,END=2039)NO,X,Y,(OBI(I),I=1,K1)
! CHANGE 2013-02-26.
  END IF
  Z=OBI(1)
  XY2= X*X+Y*Y
  XY =  SQRT(XY2)
  DIST2 = XY2+Z*Z
  DISTO =  SQRT(DIST2)
  SLON =  ATAN2(Y,X)*180.0D0/PI
  RLATC=ATAN2(Z,XY)
! write(*,*)' RLATC ',RLATC
 ELSE
! INPUT OF  SPHERICAL GEOCENTRIC COORDINATES. ADDED 2004-01-06.
! CHANGED 2005-09-06 AND 2010-11-22.
  IF (LFORM) THEN
   READ(INZ,FMT,END=2039)NO,CLATD,SLON,(OBI(I),I=1,K1)
  ELSE
   READ(INZ,*,END=2039)TIMEP,CLATD,SLON,(OBI(I),I=1,K1)
   NO=TIMEP
  END IF
  DC=OBI(1)
  RLATC=CLATD*PI/180.0D0
  RDI=OBI(1)
  Z=OBI(1)*SIN(RLATC)
  XY=OBI(1)*COS(RLATC)
! CORRECTION 2009-09-28.
  DISTO=RDI
 END IF
!
```

```
!  COMPUTATION OF THE NEW GEODETIC LATITUDE, CF REF(C) PAGE 183.
 SINLAP=SIN(RLATC)
 S = AX2/ SQRT(D1-E22*SINLAP**2)
!      S=AX2
 DH = DISTO-AX2
 RLAT1 = RLATC
 COSLA= COS(RLATC)
 NC=0
   70 RLAT = RLAT1
!
 NC=NC+1
 RLAT1 =  ATAN2(Z,XY-E22*S*COSLA)
 COSLA =  COS(RLAT1)
 S = AX2/ SQRT(D1-E22*(D1-COSLA**2))
 DH = XY/COSLA-S
 OBI(1)=DH
!
 IF (ABS(RLAT1-RLAT).GT.1.0D-15.AND.NC.LT.30) GO TO 70
 SLATC=SLAT
 SLAT=RLAT1*180.0D0/PI
!      DC=OBI(1)
 OBI(1)=DH
 RDI=DH
!
 DN=AX2/SQRT(1.0D0-E22*SIN(RLAT1)**2)
 ZC=((1.0D0-E22)*DN+DH)*SIN(RLAT1)
 XYC=(DN+DH)*COS(RLAT1)
 XC=XYC*COS(SLON*PI/180.0D0)
 YC=XYC*SIN(SLON*PI/180.0D0)
 IF (IANG.EQ.5) THEN
  IF (ABS(X-XC).GT.1.0D0.OR.ABS(Y-YC).GT.1.0D0.OR.ABS(ZC-Z).GT.1.0D0) WRITE(*,*)
' WARNING22 ',X,XC,Y,YC,Z,ZC
 ELSE
  DISTO=SQRT(XC**2+YC**2+ZC**2)
  RLATCC=ATAN2(ZC,XYC)*180.0D0/PI
  IF (ABS(RLATCC-RLATC*180.0D0/PI).GT.1.0D-6.OR.ABS(DISTO-DC).GT.1.0) WRITE(*,*)
' WARNING23 ',RLATCC,RLATC,DISTO,DC,NC
 END IF
!
 GO TO 2030
!
 2340 READ(INZ,93)NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,&
 (OBI(I),I=1,K1),LSTOP
   93 FORMAT(1X,I5,8X,2(I3,I2,F6.2,1X),F6.2,1X,4(F5.2,1X),11X)
 GO TO 2030
!
 2039 NO=-1
 2030 CONTINUE
 RETURN
END SUBROUTINE IFORMAT

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE GEOCOLH(LINTER,TIMEARRAY,RCBASE,LNDAT,SSOBS,LSATAC,IBSS,LMTEST,&
 LUNIX)
! GEOCOL SUBMODULE, PROGRAMMED 1991.08.30.  C.C.TSCHERNING,&
! GEOPHYSICAL INSTITUTE, UNIVERSITY OF COPENHAGEN. HERE THE
! COLLOCATION NORMAL EQUATIONS ARE ESTABLISHED AND SOLVED.
! LAST UPDATE 2013-01-07.
!

 USE m_params,      ONLY : MAXO,NSAT,NPMAX,NIPT,NIPCAT,NDIMC,NCRW,NNBL  ! MAXO I
S USED IN THE COMMON BLOCKS PR AND CPARM AND IN THE DIMENSION STATEMENT.
 USE m_params,      ONLY : NCOEFF,NROOT,NNSU,NEQFIM,MAXCY                     !
 PARAMETERS GIVING THE SIZE OF THE ARRAYS HOLDING POTENTIAL COEFFICIENTS.
 USE m_geocol_data, ONLY : C,NCAT,ISZE,NBL,MAXBL,ISIZE,MI1,MI2,MMAXB,MAXFIL,&
                           NEQFI,NEQFMA,MAXBNE,DNANE,LNBL1
 USE m_cholsol,     ONLY : NBB,NN,copy_files
 USE m_geocol_data, ONLY : SR11,SR12,SR13,SR22,COSAZ,SINAZ,SATROT
```



```
 USE m_geocol_data, ONLY : STEPN,COSSTN,SINSTN,&
                           STEPE,COSSTE,SINSTE,NFILTE,COST2P,SINT2P
 USE m_data,        ONLY : LTCOV,LONEQ,LTIME,LDEFF
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN

 USE m_data,        ONLY : BSIZE,ANDEX,PRETAP,PREDP,HCZERO,NCZERO

 USE m_geocol_data, ONLY : B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,  &
                           WOBS,ISAT,SINLOP,COSLOP,BSIZEN,BSIZEE,COSLAP, &
                           SINLAP,RLONGP,RP,CAZP,SAZP,HP,RLATP,ICZERO,   &
                           NI,NR,IKP,ISATP,NOBLK,LONECO,LNKSIP,LNETAP,   &
                           LDEFVP,LOBSST,LSTART
 USE m_geocol_data, ONLY : FG,FJ,LP,OMEGA2,IORDER
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,DC,HCMAX,KCI
 USE m_geocol_data, ONLY : CCR,SIGMAX,CCV,NC1,NC2,LOCAL,LSUM
 USE m_geocol_data, ONLY : STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE, COST2Q,SINT2
Q
 USE m_data,        ONLY : IPTYPE,ITIME,ITIME0,NPARM,NPARM1,LNEWD,LGRID,LRESOL
 USE m_geocol_data, ONLY : SFACT,FPERIO,IPACAT,ITROLD,ICODE,MAXPAR,MP,&
                           IPA,NCXLAS,COFf
 USE m_geocol_data, ONLY : DXX,NUM,VARI,SCALE,SCALE2,INN,INV,CFA
 USE m_data,        ONLY : OLDT,OLDR,LFIRST
 USE m_geocol_data, ONLY : INUMR,NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,&
                           LSTNO=>LTERM,LTERMA,LTERMO,LOUTC,LTRAN,LNERNO,&
                           LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT
 USE m_geocol_data, ONLY : IMAX1,IMAX1R
 USE m_data,        ONLY : OLDCOV,S,SR,AAI,AAR,IS,IPX,LCO1,NBOLD
 USE m_data,        ONLY : OLDB,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,LDIF
 USE m_data,        ONLY : 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
 USE m_geocol_data, ONLY : C20IN,G1,G2,CM3,CMM2,CM1
 USE m_geocol_data, ONLY : SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS3,EPS2,EPS1,DL,AX2,E
22
 USE m_geocol_data, ONLY : HQD,RLATD,SINLATD,COSLATD,RLONGD,SINLOND,COSLOND

IMPLICIT NONE

 INTEGER ::       JI,ICREL,NREL,NBC,NSTEPE,NE,IKC,       N19    ,&
                  IFC,NOBLK_LOC,IRECL,J,NB,K,NC,JC,I1,&
                  NBT,ICNEXT,ICC,NCREL,NSTEPN,ND,&
                  KY,KYR,MAXBL1,I,JJR,JREL,NCC,KYREL,IOBSC,N1C,N11,NJ4,&
                  N14,NNEQ,JRR,JRR1,JRR2,      NERR,&
                  IBSS,NB1,IBT,JJ,MAXOBS,MAXCM,NRE,NJ

 LOGICAL ::              LY,LSANEQ,       LBST, &
                  LINTER,            LUNIX,LSATP,LREPEC,LGRP,LNGR,LSATAC,&
                  LMTEST,LDIFP!=.FALSE.

 REAL(KIND=8) :: RCBASE,SSOBS,PW,SHIFTS,RADEG,&
                 SINB,SINT,COST,COSB,CPU2,SYTIME,CPU3,CNRC,PW0

!REAL(KIND=4) :: COFF
 REAL(KIND=8) :: TIMEARRAY(2)

 LOGICAL ::              LEQANG,LMEAP1,LMAX1,LRESTA,LNCOL,LMENSI,&
                  LNDAT,LSTNEQ,LPARER,LOPEN

 CHARACTER(LEN=128) ::       UDATE

 REAL(KIND=8), DIMENSION(MAXO)   ::    B1,HQ1,RLAT1,SINLAT1,COSLAT1,RLONG1,SINLO
N1,COSLON1,WOBS1
!REAL(KIND=8), DIMENSION(MAXO)   :: BB,B1,HQ1,RLAT1,SINLAT1,COSLAT1,RLONG1,SINLO
N1,COSLON1,WOBS1
 REAL(KIND=8), DIMENSION(NPMAX)  :: CX
 REAL(KIND=8), DIMENSION(4)      :: B4
!REAL(KIND=8), DIMENSION(NSAT*6) :: ROTSAT
```

```
 REAL(KIND=8), DIMENSION(MAXCY)  :: CY                                  !
 CHANGE 2011-10-31 MAXCY INTRODUCED.
 REAL(KIND=8), DIMENSION(NN)    :: BB1

!COMMON /CMCOV/CCI(24),CCR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HCMAX,CCV(2,2),DC(36),KCI(37),NC1,NC2,LOCAL,LSUM
! CHANGE TO 2200 2010-11-24.
! COMMON VARIABLES USED IN COVAX.

!COMMON /DAT/LNEWD,LRESOL,LGRID
! /DAT/ TRANSFERS LOGICAL VARIABLES TO THE SUBROUTINE ITRAN.

!COMMON /OUTC/INUMR(12),NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,LTERMA,LTERMO,L
STNO,LOUTC,LTRAN,LNERNO,LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT
! IN OUTC ARE STORED SUBSCRIPTS OF VARIABLES TO BE OUTPUT AND LIMITS
! FOR DO-LOOPS IN OUTPUT. NOTE THAT OUTC OCCURS IN SUBROUTINES
! HEAD, COUT, CXPARM AND THE BLOCK DATA MODULE.

!COMMON /CHEAD/IA,IB,IH,IP,IT,IA1,IB1,IP1,IT1,IC1,IC11,K1,IOBS1,IOBS2,ITE,ITE1,I
ITE,IITE1,IIP,IIP1,IIE,IIE1,INO,LPOT,LKM,LTERRC,LPOTIN

!COMMON /COM2/DXX,NUM(70),VARI(32),SCALE,SCALE2,INN,INV
! USED BY COMPA, COMPARING OBSERVED AND PREDICTED QUANTITIES.

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT

!COMMON /GPOTC0/C20IN,G1(3),G2(3,3),CM3,CMM2,CM1

!COMMON /GPOTC3/COFF(NCOEFF)

!COMMON /GPOTC1/OLDT,OLDR,CFA,IGQ(12),LFIRST,HP9000
! COMMON VARIABLES USED IN GPOTDR, SETCM ,LOADCM, PRED AND CXPARM.

!COMMON /GRAVCC/FG(30),FJ(30),LP(2),OMEGA2,IORDER
! COMMON VARIABLES USED IN GRAVC AND RGRAV, HOLDING I.E. COEFFICI-
! ENTS OF LEGENDRE SERIES OF NORMAL POTENTIAL AND NORMAL GRAVITY
! FORMULA.

!COMMON /ITRANC/SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS3,EPS2,EPS1,DL,AX2,E22

!COMMON /CPARM/SFACT(NIPCAT),FPERIO(10,2),IPTYPE(NIPT),IPACAT(3*NIPT),ITIME(NIPC
AT),ITIME0(NIPT),ITROLD,ICODE(10),NPARM,NPARM1,MAXPAR,MP,IPA,NCXLAS

!COMMON /CMEAQ/STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE,COST2Q,SINT2Q
! STEPSIZES USED WHEN CALCULATING MEAN VALUES.

!COMMON /BIPAR/OLDB(4),CNR,GMP,AX,NMAX,II,IOBS,IOBSR,N1,NIR,MAXC,MAXC1,MAXC2,N,I
C,NT,IDIMC,IDIMCN,MAXBLT,JR,ISO,LPARAM,LPRED,LNEQ,LNEQ8,LNEWSO

!COMMON /BIPARC/OLDCOV(2),S,SR,AAI,AAR,NBOLD,IS,IPX,IMAX1,IMAX1R,LCO1

 LSANEQ = LF
 LRESTA = LF
 IFC    = 0
 NOBLK_LOC  = 0
 MI2    = 0
 NERR   = 0
 RADEG=180.0D0/3.1415926535D0

!
! *************** INPUT (12) ********************************
!
 IF (LINTER.AND.LRESOL) WRITE(6,*) ' INPUT LSANEQ, TRUE IF NORMAL EQ. SAVED AND
FIRST RED. COLUMN'
! THIS GIVES A POSSIBILITY FOR RESTART FROM ANY OBSERVATION EARLIER
! IN THE SEQUENCE OR FROM THE LAST OBSERVATION TO BUILD A SOLUTION
! BASED ON MORE DATA. IF IFC=0 AND LSANEQ=TRUE, THE UNREDUCED COEFFICIENTS
! WILL BE SAVED.
 IF (LRESOL) READ(5,*)LSANEQ,IFC
```



```
  216 FORMAT(L2,I7)
 LSTNEQ=IFC.EQ.0.AND.LRESOL
 IF (LSANEQ)WRITE(*,*)' SAVED REDUCED COLUMNS = ',IFC
 IF (LRESOL.AND.(.NOT.LSANEQ).AND.N1.EQ.IFC) GO TO 3228
 IF (.NOT.LSANEQ.AND.LRESOL.AND.N1.NE.IFC) IFC=0
 IF (MOD(IFC,NBB*IBSS).NE.0) THEN
  WRITE(*,*)' IFC MUST BE MULTIPLUM OF CHUNK SIZE ',NB*IBSS
  STOP
 END IF
!
! ADDED 2000-07-25 BY CCT.
 IF (IFC.LT.0) THEN
  LRESTA=LT
  IFC=-IFC
  WRITE(*,*)' REDUCTION OF NEQ WILL START FROM COLUMN ',IFC
 ELSE
! IFC=0
 END IF
!
! CORRECTION 2000.07.01
 LDEFF = LF
 IF (LDEFF) GO TO 2037
 LDEFF = LT
 IDIMC=NDIMC
! IN THIS PROGRAM-VERSION, THE ARRAY C HAS DIMENSION IDIMC.
  IRECL=NEQFI(1,2)**2*8
! WRITE(*,*)' IRECL= ',IRECL
  MAXOBS=(2*NCRW+IBSS**2+IBSS)/(2*IBSS+1)
!
 2037 CONTINUE
 CALL SETCAT(NB)
! NEW VARIABLE LPARER, 2004-06-24. TRUE IF ERROR ESTIMATES
! OF PARAMETERS WILL BE OUTPUT.
 LPARER=LT
 IF (LRESOL.AND.IFC.EQ.N1) GO TO 3228
!
 LPARER=.NOT.(LRESOL.AND.IFC.EQ.(N1-1))
 IF (.NOT.LPARER.AND.LPARAM) WRITE(*,*)' ERROR ESTIMATES OF PARAMETERS NOT OUTPU
T.'
 IF (MAXBLT.GT.0) WRITE(6,335)MAXBLT
  335 FORMAT(/'  ',I5,' RECORDS USED FOR NORMAL EQUATIONS.'/)
!
 IF (LRESTA) GO TO 7475
!
! COMPUTATION OF ELEMENTS OF NORMAL EQUATIONS (EQUAL TO THE COVARIANCE
! BETWEEN THE OBSERVATIONS). THE COEFFICIENTS ARE STORED IN THE ONE-DI-
! MENSIONAL ARRAY C, COLUMN AFTER COLUMN,THE DIAGONAL ELEMENT
! HAVING THE HIGHEST SUBSCRIPT.
!
! INITIALIZING VARIABLES:
! NI IS HERE THE SUBSCRIPT OF THE FIRST ELEMENT OF COLUMN NC IN ARRAY
! C. IN THE SUBROUTINE PRED, NI IS THE SUBSCRIPT OF THE ELEMENTS OF THE
! COLUMN. NB  IS THE NUMBER OF THE BLOCK IN WHICH THE COVARIANCES ARE
! STORED AND I1 IS THE NUMBER OF THE LAST COLUMN STORED IN THE BLOCK.
! ICNEXT IS THE NUMBER OF THE FIRST COLUMN WITHIN A GROUP OF DATA WITH
! THE SAME CHARACTERISTICS. (THE CHARACTERISTICS ARE GIVEN BY THE ARRAY
! ANDEX (SUBSCRIPTS JC AND JC+1)).
 NI = 1
 K = 1
 NC = 1
 JC = II
 NB=1
 I1=NBL(2)
!     WRITE(*,338)(NCAT(IIK),IIK=1,IC+1),NCAT(ISIZE),ISIZE
 NBT = 1
 ICNEXT = ISO+1
 ICREL=ISO
 B1=B
 HQ1=HQ
```

```
 RLAT1=RLAT
 SINLAT1=SINLAT
 COSLAT1=COSLAT
 RLONG1=RLONG
 SINLON1=SINLON
 COSLON1=COSLON
 WOBS1=WOBS
!
 DO 3100 IC = 1, IOBS
 IF (MOD(IC,10000).EQ.0) WRITE(*,*)' COLUMN ',IC,' PROCESSING '
 LNCOL = NC.LE.IFC
 ICC = IC+ISO
 NCC = NC+ISO-1
 NCREL=MOD(NCC,MAXO)+1
 IF (LOBSST) THEN
  NOBLK_LOC=NCC/MAXO+1
  IF (NCREL.EQ.1.OR.IC.EQ.1) THEN
! CHANGE OF TEST-OUTPUT 2012-05-18.
!  WRITE(6,*)'UNIT 16, BLK ',NOBLK_LOC,' 17 READ FOR TRANSFER B TO C.',IC
   READ(16,REC=NOBLK_LOC)B1,HQ1,RLAT1,SINLAT1,COSLAT1,RLONG1,SINLON1,COSLON1,WOB
S1
   IF (LSATAC) THEN
    READ(14,REC=NOBLK_LOC)SR11,SR12,SR13,SR22,COSAZ,SINAZ
!   IF (NOBLK_LOC.LT.0) WRITE(6,*)' UNIT 14 BLK ',NOBLK_LOC,' 18 READ FOR TRANSF
ER B TO C.'
   END IF
   IF (IC.EQ.1.AND.ISO.NE.0) THEN
    ICREL=MOD(ISO,MAXO)
   ELSE
    ICREL=0
   END IF
  END IF
 END IF
 ICREL=ICREL+1
!
 IF(ICC .NE. ICNEXT)GO TO 3003
 IKP = ANDEX(JC+1)
! INITIALIZATION OF VARIABLES IN COMMON BLOCK /PR/.
 IF (IKP.LT.100) KCI(6)=IKC(IKP)
 ICNEXT = ANDEX(JC)
 ISATP=ISAT(JC)
 LSATP=ISATP.GT.0
 BSIZEN=BSIZE(JC)
 LMENSI=ABS(BSIZEN).GT.1.0D-6
 LMEAP1=BSIZEN.LT.D0.AND.LMENSI
!write(*,*)' geocolh: ',LMENSI,LMEAP1,NFILTE,LSATP
!
 IF (LMENSI) THEN
  IF (LMEAP1) THEN
   NSTEPN=NFILTE
   LDIFP=NFILTE.LT.0
   IF (.NOT.LDIFP) THEN
    SHIFTS=ABS(BSIZEN)*(NSTEPN-1)/2
    COST2P=COS(SHIFTS)
    SINT2P=SIN(SHIFTS)
   END IF
  ELSE
   NSTEPN=5
  END IF
  NSTEPE=5
  BSIZEE=BSIZE(JC+1)
  LEQANG=BSIZEE.GT.1.0D-6
  STEPE=D0
  CALL ICMEAN(ABS(BSIZEN),STEPN,NSTEPN,COSSTN,SINSTN,D1,D0,LT,LF)
  IF (LEQANG) CALL ICMEAN(BSIZEE,STEPE,NSTEPE,COSSTE,SINSTE,D1,D0,LT,LF)
 ELSE
  STEPE=D1
 END IF
!
```



```
 JC = JC+2
 LREPEC = IKP .EQ. 5 .OR.(IKP.GT.25.AND.IKP.LT.36)
 LONECO = .NOT.LREPEC
 LGRP = IKP.EQ.2.OR.IKP.EQ.13
 LNGR = .NOT.LGRP
 LNKSIP = LONECO.AND.IKP.NE.3.AND.IKP.NE.16.AND.IKP.NE.18
! .AND.IKP.NE.20.AND.IKP.NE.25.AND.IKP.NE.22
! CHANGE 2013-03-15.
 LNETAP = LONECO.AND.IKP.NE.4.AND.IKP.NE.17.AND.IKP.NE.19
! .AND.IKP.NE.23.AND.IKP.NE.24
 LDEFVP = .NOT.LNKSIP.OR.(.NOT.LNETAP)
! CHANGE 2013-03-15.
!
 3003 LBST=LREPEC.AND.(NC.EQ.I1)
!
! IF LNCOL IS FALSE WILL PRED IN MOST CASES NOT BE CALLED. HENCE,
! WE MUST 'EXTERNALLY' UPDATE THE VALUE OF NI, WHICH OTHERWISE
! POINTS AT THE ELEMENT IN C, WHERE THE COVARIANCE IS STORED.
!
 IF (LNCOL) NI = NI+NC
 IF (IKP.GE.100) GO TO 3031
 IF (LNCOL.AND.LONECO) GO TO 3020
 IF (.NOT.LNCOL.OR.NC.NE.IFC) GO TO 3032
 LNCOL=LF
 NI=NI-NC
 IFC=IFC-1
 WRITE(6,337)IFC
  337 FORMAT(' **** WARNING24 ****  IFC DECREASED TO',I5)
 3032 IF (LNCOL) GO TO 3019
!
 COSLAP = COSLAT1(ICREL)
 SINLAP = SINLAT1(ICREL)
 COSLOP = COSLON1(ICREL)
 SINLOP = SINLON1(ICREL)
 RLONGP = RLONG1(ICREL)
 RLATP  = RLAT1(ICREL)
 HP = HQ1(ICREL)
 IF (.NOT.LSATP) GO TO 3033
!print*,'ldifp:',ldifp !bsotest
 IF (LDIFP) THEN
    RLATD=SR11(ICREL)
    RLONGD=SR12(ICREL)
    HQD=SR13(ICREL)
!   write(*,*)' 7610 ',NCREL,RLATD,RLONGD,HQD
    SINLATD=SIN(RLATD/RADEG)
    COSLATD=COS(RLATD/RADEG)
    SINLOND=SIN(RLONGD/RADEG)
    COSLOND=COS(RLONGD/RADEG)
 ELSE
 IF (ISATP.EQ.1) THEN
  CAZP=COSAZ(ICREL)
  SAZP=SINAZ(ICREL)
 ELSE
! MODIF. 1991.06.08 TO ENABLE FULL ROTATION.
  COSB=SR11(ICREL)
  SINB=SR12(ICREL)
  COST=SR13(ICREL)
  SINT=SR22(ICREL)
  CAZP=COSAZ(ICREL)
  SAZP=SINAZ(ICREL)
! ADDITION 2002-09-27.
  SATROT(1,1) =  SAZP*COSB
  SATROT(1,2) =  CAZP*COST+SAZP*SINB*SINT
  SATROT(1,3) = -CAZP*SINT+COST*SAZP*SINB
  SATROT(2,1) = -CAZP*COSB
  SATROT(2,2) =  SAZP*COST-SINT*CAZP*SINB
  SATROT(2,3) = -SAZP*SINT-COST*CAZP*SINB
  SATROT(3,1) = -SINB
  SATROT(3,2) =  COSB*SINT
```

```
  SATROT(3,3) =  COST*COSB
 END IF
 END IF
 3033 IF (LMENSI.AND.(.NOT.LEQANG).AND.(.NOT.LMEAP1)) CALL ICMEAN(BSIZEN,STEPE,N
STEPE,COSSTE,SINSTE,COSLAP,SINLAP,LF,LF)
 IF (LMEAP1) THEN
  STEPE=-D1
  COSSTE=COSAZ(ICC)
  SINSTE=SINAZ(ICC)
 END IF
 GO TO 3001
 3031 IKP = 100+K
 K = K+1
 IF (LNCOL) GO TO 3020
! AS WE FOR LREPEC=TRUE ARE COMPUTING TWO COLUMNS AT THE SAME TIME, WE
! MUST, IN CASE THE SECOND COLUMN IS THE FIRST ONE IN THE NEXT RECORD
! STORE THIS ONE TEMPORARY IN ARRAY B. THE PROBLEM WILL ONLY OCCUR WHEN
! WE ARE SETTING UP THE NORMALEQUATIONS. LBST = B-STORE.
!
 3001 CONTINUE
 CALL PRED(S ,AAI,IS,    ISO,II,IC,NC,IMAX1,LF   ,LBST,LT  ,LTCOV,LSATAC,LF   ,0
)
          !PRED(SS,AAI,IS,ISP,ISO,II,IC,NC,IMAX1,LPRED,LBST,LCST,LTCOV,LSATAC,LW
AIT,NPRED  )
!
 ND = NI-1
 IF (LONECO) GO TO 3020
! THE PRECEDING STATEMENT ASSURES, THAT THE DIAGONAL ELEMENT CORRESPON-
! DING TO ETAP BECOMES EQUAL TO THAT OF KSIP. CH 1992.07.19.
 3019 continue
 IF (.NOT.LNCOL) THEN
   IF (MOD(NC,NEQFI(1,2)).EQ.0) THEN
     MI1 = MI1+1
     NBL(MI1) = NC
     I1 = NC
!    write(*,*)' call restore -eta ',mi1,nc
     NJ=NBL(MI1-1)+1
     CALL  RESTORE_CH(NJ,0,LF,LT,LF,LMTEST)
   END IF
 END IF
 NC = NC+1
 NI = NI+NC
!
 3020 continue
 IF (NC.GE.MAXOBS) THEN
   WRITE(*,*)' TOO MANY OBSERVATIONS ',NC
   STOP
 END IF
! THE SUBROUTINE RE-PACKS THE ARRAY C, WHICH HOLDS SEVERAL
! COLUMNS, AND ONLY TRANSFER THE SUB-BLOCK WHEN IT IS FULL.
 IF (MOD(NC,NEQFI(1,2)).EQ.0)  THEN
   MI1=MI1+1
   NBL(MI1)=NC
   I1=NC
   NJ=NBL(MI1-1)+1
   IF ((.NOT.LNCOL).AND.IFC.GT.0.AND.(IFC-NBL(MI1-1)).LE.IBSS.AND.(IFC-NBL(MI1-1
)).GT.0) THEN
    NBL(MI1-1)=IFC
    write(*,*)' 7434 call restore_ch ',nj,ifc,lncol,NBL(MI1-1),MI1,NC
! CORRECTION 2012-08-01.
    CALL  RESTORE_CH(NJ,0,LT,LF,LF,LMTEST)
    MI2=MI1
   ELSE
    IF (.NOT.LNCOL) THEN
     J=0
     CALL  RESTORE_CH(NJ,0,LT,LF,LF,LMTEST)
! CHANGE 2012-08-02.
    ELSE
     CALL  RESTORE_CH(NJ,IFC,LT,LF,LF,LMTEST)
```



```
    END IF
    IF (LF) write(*,*)' call restore x,mi1,nc,ifc,lncol,nj ',mi1,nc,ifc,lncol,nj
   END IF
!   write(*,*)' call restore ',mi1,nc,ifc,lncol
 END IF
!write(*,*)' 7448 NC,I1,N,LONEQ,LNCOL ',NC,I1,N,LONEQ,LNCOL
 IF (NC.LT.I1.AND.NC.LT.N) GO TO 3100
! IN VERSIONS EARLIER THAN MAY 1, 1986, AN ERROR COULD OCCUR HERE,
! BECAUSE THE LAST COLUMN WAS NOT ASSIGNED TO C(KY) WHEN LBST WAS
! TRUE SIMULTANEOUSLY.
!
 3261 continue
 NBT = NBT+NT
 IF ((.NOT.LONEQ).OR.LNCOL) GO TO 3200

! OUTPUT OF COEFFICIENTS OF NORMAL-EQUATIONS.
 WRITE(6,380)NB
  380 FORMAT(/' COEFFICIENTS OF NORMAL-EQUATIONS, BLOCK ',I4,/&
 ' (FIRST 200 ELEMENTS AND LAST FULL BLOCK)')
 I1 = NI-1
 IF (LBST) I1 = ND
!IF (I1.GT.200) I1=200
 IF (NB.EQ.MAXBL) I1 = MAXC2
!
 LMAX1=LF
 DO KY=1,I1
   LMAX1=LMAX1.OR.(ABS(C(KY)).GT.1.0D0)
 END DO
 IF (LMAX1) THEN
  WRITE(6,381)(C(KY), KY = 1, I1)
  381  FORMAT(5F15.8)
 ELSE
! NEW OUTPUT-FEATURE ADDED 1995.11.21  CHANGED 2013-03-10 BY CCT.
  WRITE(6,1382)(C(KY), KY = 1, I1)
 1382  FORMAT(6F13.9)
 END IF

 3200 continue
 NB = NB+1
 NI = 1
 IF (NC.NE.N) I1 = NBL(NB+1)
 IF (NB.GT.MAXBL) GO TO 3201
!     WRITE(*,338)(NCAT(IIK),IIK=1,IC+1),NCAT(ISIZE),ISIZE
! WE HAVE TO READ THE WHOLE CONTENT OF BLOCK NB INTO ARRAY C, BECAUSE
! WE MUST BE SURE THAT THE RIGHT-HAND SIDE (WHICH ALREADY IS STORED)
! IS PLACED CORRECTLY.
!
 3201 IF (.NOT.LBST) GO TO 3100
!
 IF (LNCOL) GO TO 3203
 DO 3202 KY=1,NC
 KYR=KY+ISO-1
 KYREL=MOD(KYR,MAXO)+1
 IF (LOBSST) THEN
  NOBLK_LOC=KYR/MAXO+1
!     WRITE(*,*)' KY,KYR,ISO,MAXO ',KY,KYR,ISO,MAXO
  IF (KYREL.EQ.1.OR.KY.EQ.1) THEN
!  WRITE(6,*)'BLK ',NOBLK_LOC,' 1 READ FOR TRANSFER B TO C.'
   READ(16,REC=NOBLK_LOC)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
   IF (LSATAC) READ(14,REC=NOBLK_LOC)SR11,SR12,SR13,SR22,COSAZ,SINAZ
!     WRITE(6,*)'BLK ',NOBLK_LOC,' 2 READ FOR TRANSFER B TO C.'
  END IF
 END IF
 3202 C(KY) = B(KYREL)
!
 3203 NI = NC+1
! ERROR CORRECTED 1987.10.03. NI WAS EARLIER NOT UPDATED FOR LNCOL=LT.
 IF (NC.NE.N) GO TO 3100
 LBST=LF
```

```
 WRITE(6,383) NB
  383 FORMAT(' LAST COLUMN IS FIRST LOGICAL COLUMN IN BLOCK ',I4)
 GO TO 3261
!
 3100 NC = NC+1
!write(*,*)' 7517 NB,NI,NC,N ',NB,NI,NC,N
! END OF LOOP FORMING NORMAL-EQUATIONS.
!
 7475 IF (LTIME) THEN
  CPU3=SYTIME(RCBASE,TIMEARRAY)
  WRITE(6,7470)TIMEARRAY(1),CPU3
 END IF
 7470 FORMAT(' TIME USED=',F12.5,' SEC, ELAPSED TIME=',F12.5,' SEC')
!
 LNBL1=LF
  MI1 = MI1+1
  NBL(MI1) = N1
  KY = NBL(MI1-1)
  IF (IFC.GT.0.AND.IFC.GT.NBL(MI1-1)) then
   NBL(MI1-1)=IFC
   write(*,*)' NBL(MI1-1) Changed to ',ifc
! experiment 2007-12-12.
   MI2=MI1
  END IF
! CORRECTION 2007-112-11.
  MAXC=N1*(N1-1)/2-KY*(KY+1)/2
!      write(*,*)' call of restore ',mi1,n1,NBL(MI1-1),MAXC
! STORING THE RIGHT-HAND SIDE.
  DO J=1,N1
   JJR=J+ISO
   JREL=MOD(JJR,MAXO)
   IF (LOBSST) THEN
    NOBLK_LOC=JJR/MAXO+1
!        WRITE(*,*)' J, JJR,JREL,ISO ',J,JJR,JREL,ISO,MAXO
    IF (J.EQ.1.OR.JREL.EQ.1) THEN
!         WRITE(6,*)'BLK ',NOBLK_LOC,' 17 READ FOR TRANSFER B TO C.'
   READ(16,REC=NOBLK_LOC)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
!         WRITE(6,*)'BLK ',NOBLK_LOC,' 18 READ FOR TRANSFER B TO C.'
    END IF
   END IF
! CORRECTION 1999-11-24 AND CHANGE 2012-02-06 BY CCT.
   IF (JREL.EQ.0) JREL=MAXO
   C(MAXC+J)=B(JREL)
!  C(MAXC+J)=CR(J)
!   write(*,*)' maxc+j,jrel,b ',maxc+j,jrel,b(jrel)
  END DO
  IF (LONEQ) THEN
   write(*,*)' 7557 right-hand side: ',maxc+1,maxc+n1,n1
   write(*,5911)(C(J+MAXC),J=1,N1)
! FORMAT CHANGED 2013-03-10.
 5911   format(6F12.5)
   write(*,*)' cmaxcpn1 ',C(MAXC+N1)
   C(MAXC+N1)=D0
  END IF
  MAXCM=MAXC
! MAXCM IS USED IN COMMON /CRESTO/ TO TRANSFER MAXC.!JTF: CRESTO no longer exist
s, but variables are still used -- see common_blocks_all.f90 for detailed notes.
!
!  NJ=INT(N1/IBSS)*IBSS+1
   NJ=NBL(MI1-1)+1
  IF (LF) WRITE(*,*)' 7566 RESTORE MI1,MI2,N1,IFC,J,NJ ',MI1,MI2,N1,IFC,J,NJ
  CALL  RESTORE_CH(NJ,0,LT,LF,LF,LMTEST)
!
  IF (LUNIX) THEN
   CALL FDATE(UDATE)
   WRITE(6,*)UDATE
  END IF
   write(*,*)' nes 4.0 IFC= ',IFC
   if (N1.LE.IBSS) THEN
```



```
! ADDED 2013-04-09.
    WRITE(*,*)' BLOCK-SIZE SHOULD BE LE THAN NUMBER OF DATA, STOP ',&
    N1,NBB
    STOP
   END IF
   if (LSTNEQ) call copy_files(lt)
!bso cholsol3
   call cholsol(N1,IFC+1,NPARM1-1,0,lt,lt,lf,lf,lf)
   print*,'BSO CHOLSOL3'
!
 IF (LWRSOL) THEN
! PUNCHING: NUMBER OF OBSERVATION POINTS, NUMBER OF OBSERVATIONS, DIF-
! FERENCE BETWEEN SQUARESUM OF OBSERVATIONS AND NORM OF APPROXIMATION,
! A CHECK-NUMBER (KEE) CNR, AND FINALLY THE SOLUTIONS AND THE SQUARE-SUM
! OF THE OBSERVATIONS. CHANGE 2000-10-03.
  WRITE(17,361)IOBS,N1,MAXC,MAXBL,MAXBLT,CNR
  361  FORMAT(5I6,2D15.7)
  WRITE(17,364)(C(J+MAXC),J=1,N1)
  MAXBL1=MAXBL+1
  WRITE(17,363)(NBL(I),I=1,MAXBL1)
  363  FORMAT(10I7)
 END IF
!
 IBSS=NEQFI(1,2)
! MAX. NUMBER OF COLUMNS IN BLOCK.
 NB1=INT(N1/IBSS)
! NUMBER OF BLOCKS IN LAST COLUMN OF BLOCKS.
 inquire (100,OPENED=lopen)
 if (lopen) write(*,*)' unit 100 open '
!if (.NOT.lopen) open(100,file=filename(0,1),access='direct',recl=8*NN)
!
 NBC=1
 DO J = 1, N1
  JJR=J+ISO-1
  JREL=MOD(JJR,MAXO)+1
  IF (LOBSST) THEN
   NOBLK_LOC=JJR/MAXO+1
   IF (JREL.EQ.1.OR.J.EQ.1) THEN
!       WRITE(6,*)'BLK ',NOBLK_LOC,' 3 READ FOR TRANSFER B TO C.'
   READ(16,REC=NOBLK_LOC)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
    IF (LSATAC) READ(14,REC=NOBLK_LOC)SR11,SR12,SR13,SR22,COSAZ,SINAZ
!       WRITE(6,*)'BLK ',NOBLK_LOC,' 4 READ FOR TRANSFER B TO C.'
   END IF
  END IF
  IF (ABS(MAXC+J).GT.NDIMC) THEN
   WRITE(*,*)' 5: MAXC,J ',MAXC,J
  END IF
  IF (MOD(J,IBSS).EQ.1) THEN
   IBT=INT(J/IBSS)+1
   JJ=1
  ELSE
   JJ=JJ+1
  END IF
! THE BLOCK-SIZE (COLUMNS) NN MUST BE USED FOR UNIT 100. 2013-02-13.
  IF (MOD(J,NN).EQ.1) THEN
   READ(100,REC=NBC)BB1
   NBC=NBC+1
  END IF
  B(JREL)=BB1(MOD(J-1,NN)+1)
  PW=B(JREL)
! write(*,*)' transferring solutions to B from unit 100, NBC,B,JREL  ',NBC,B(JRE
L),JREL
! B(JREL)=C(MAXC+J)
! HERE MUST BE ERROR IN TRANSFER OF THE SOLUTIONS FROM C TO B. 2012-03-03.
  IF (LOBSST.AND.(JREL.EQ.MAXO.OR.J.EQ.N1)) THEN
   WRITE(16,REC=NOBLK_LOC)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
   IF (LSATAC) WRITE(14,REC=NOBLK_LOC)SR11,SR12,SR13,SR22,COSAZ,SINAZ
   WRITE(*,*)' 1 BLOCK ',NOBLK_LOC,' WRITTEN '
  END IF
```

```
! TRANSFERRING THE SOLUTIONS TO THE ARRAY B.
 END DO
 close(100)
 WRITE(*,*)' B( ',JREL,')',B(JREL)
 PW=B(JREL)
!
 LRESOL=LF
 LNCOL=LF
!
 GO TO 3229
!
! *************** INPUT (13) ********************************
!
! INPUT OF SOLUTIONS.
 3228 MAXC = 0
 IF (LINTER)WRITE(6,*)' INPUT SOLUTION'
  READ(5,361)IOBSC,N1C,MAXC,MAXBL,MAXBLT,PW,CNRC
  N11=N1-1
!  NJ4 IS NUMBER OF ELEMENTS ON ONE LINE (RECORD), AND N14 IS
! NUMBER OF LINES IN ASCII_FILE USED TO HOLD SOLUTIONS (RESTART FILE).
  NJ4=4
  N14=N11/NJ4+1
  NREL=MOD(ISO,MAXO)
!     WRITE(*,*)' N1,N14,NJ4 ',N1,N14,NJ4
  IF (LOBSST) NOBLK_LOC=ISO/MAXO+1
  DO 3364, J=1,N14
   IF (J.EQ.N14)NJ4=MOD(N11,NJ4)+1
   READ(5,364)(B4(I),I=1,NJ4)
  364   FORMAT(4D16.9)
   DO 3365, I=1,NJ4
    NREL=NREL+1
! CORRECTION 2000-10-06.
    IF (LOBSST.AND.NREL.EQ.1) THEN
          WRITE(6,*)'BLK ',NOBLK_LOC,' 8 READ, J,NREL= ',J,NREL
   READ(16,REC=NOBLK_LOC)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
     IF (LSATAC) READ(14,REC=NOBLK_LOC)SR11,SR12,SR13,SR22,COSAZ,SINAZ
!         WRITE(6,*)'BLK ',NOBLK_LOC,' 9 READ.'
!         NREL=1
    END IF
!
    B(NREL)=B4(I)
    IF (LOBSST.AND.(NREL.EQ.MAXO.OR.(J.EQ.N14.AND.NJ4.EQ.I))) THEN
   WRITE(16,REC=NOBLK_LOC)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
     IF (LSATAC) WRITE(14,REC=NOBLK_LOC)SR11,SR12,SR13,SR22,COSAZ,SINAZ
!         WRITE(*,*)' 2 BLOCK ',NOBLK_LOC,' WRITTEN, J,NREL= ',J,NREL
     NREL=0
     NOBLK_LOC=NOBLK_LOC+1
    END IF
 3365   CONTINUE
 3364  CONTINUE
!
 MAXBL1=MAXBL+1
 READ(5,363)(NBL(J),J=1,MAXBL1)
 IF (.NOT.LSANEQ) GO TO 3227
! IYX=NREAD(CC,MAXBLT,NT,IDIMCN)
!     WRITE(*,338)(NCAT(IIK),IIK=1,IC+1),NCAT(ISIZE),ISIZE
!
! CHECK OF SOLUTIONS CORRESPOND TO OBSERVATIONS( CHANGED 1993.06.16 CCT).
 3227 IF (IOBS.EQ.IOBSC.AND.N1.EQ.N1C.AND.ABS((CNRC-CNR)/CNRC).LT.0.1D-4) GO TO
3229
 WRITE(6,354)IOBS,I,N1,N1C,CNRC,CNR
  354 FORMAT(' SOLUTIONS DO NOT CORRESPOND TO INPUT DATA, STOP.',&
 /,4I4,2E15.7)
 STOP
 3229 WRITE(6,300)
  300 FORMAT(/' SOLUTIONS TO NORMAL EQUATIONS:'/)
 NNEQ=N
!     IF (LRESOL.AND.N.GT.20) THEN    - CHANGED 2000.01.14.
 IF (N.GT.NNEQ.AND.NPARM.EQ.0) THEN
```



```
  NOBLK=MAXBL1
 END IF
 JRR=ISO
 IF (LOBSST) THEN
  NOBLK_LOC=JRR/MAXO+1
 ELSE
! CHANGE 2003-12-29.
  NOBLK_LOC=1
 END IF
 JRR1=MOD(JRR,MAXO)+1
 JRR2=MAXO
! CHANGE (+1) 2004-06-24.
 3239 IF (NOBLK_LOC.EQ.NOBLK+1) JRR2=MOD(NNEQ+ISO,MAXO)
      IF (NNEQ+ISO.LE.JRR2) JRR2=NNEQ+ISO
!     WRITE(*,*)' JRR2 ',JRR2,NNEQ,ISO,MAXO,NOBLK_LOC,NOBLK
 IF (LOBSST) THEN
! LOBSST IS TRUE WHEN OBSERVATIONS HAVE BEEN STORED IN A FILE.
! WRITE(6,*)'UNIT 16 BLK ',NOBLK_LOC,' 10 READ FOR TRANSFER B TO C.'
  READ(16,REC=NOBLK_LOC)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
  IF (LSATAC) THEN
   READ(14,REC=NOBLK_LOC)SR11,SR12,SR13,SR22,COSAZ,SINAZ
!  IF (NOBLK_LOC.LT.0) WRITE(6,*) ' UNIT 14 BLK ',NOBLK_LOC,' 11 READ FOR TRANSF
ER B TO C.'
  END IF
 END IF
 IF (LONEQ) THEN
!      WRITE(*,*)JRR1,JRR2
  WRITE(6,301)(B(J), J = JRR1, JRR2)
 ELSE
! CHANGE 2002-07-17.
  IF (NOBLK_LOC.EQ.1) THEN
   WRITE(*,305)20
  305   FORMAT(' ONLY FIRST',I5,' SOLUTIONS OUTPUT.')
   NREL=MOD(ISO,MAXO)
   WRITE(6,301)(B(J+NREL), J = 1,20)
  END IF
 END IF
! CHANGE 2003-12-29.
 NOBLK_LOC=NOBLK_LOC+1
!
 JRR1=1
  301 FORMAT(1X,5D12.5)
! change 2012-09-17.
! 301 FORMAT(1X,4E16.9)
 IF (NOBLK_LOC.LE.NOBLK.AND.NPARM.NE.0) GO TO 3239
!
 IF (NPARM.GT.0.AND.LONEQ) WRITE(6,382)NPARM
  382 FORMAT(' LAST ',I3,' ELEMENTS OF SOLUTION VECTOR',/,&
 ' ARE THE VALUES OF THE ESTIMATED PARAMETERS'/)
 IF (LRESOL) WRITE(6,362)
  362 FORMAT(/' THE SOLUTIONS HAVE BEEN COMPUTED IN A PREVIOUS RUN.')
!
 MAXC2 = MAXC+N1
 WRITE(6,353)N,SSOBS,PW
  353 FORMAT(/' NUMBER OF EQUATIONS =',I7,/&
 ' NORMALIZED SQUARE-SUM OF OBSERVATIONS        =',E13.6,/,&
 ' NORMALIZED DIFFERENCE BETWEEN SQUARE-SUM OF'/&
 ' OBSERVATIONS AND NORM OF APPROXIMATION       =',E13.6,/)
!
 IF (LTIME) THEN
  CPU2=SYTIME(RCBASE,TIMEARRAY)
! WRITE(6,7470)TIMEARRAY(1),CPU2
 END IF
!
 IF (LNDAT.OR.LRESOL.OR.NPARM.EQ.0) GO TO 5230
! OUTPUT OF EXX = (AT*C**-1*A)**-1, OR PARTS, IF NPARM .GT. 6.
 LY = (NPARM.LT.7 .OR. LONEQ)
! LY SET FALSE UNTIL ERROR-CORRELATION CAN BE COMPUTED 2012-05-18.
 LY = LF
```

```
! error 2010-12-18.
 IF (LY) WRITE(6,371)
 JI=0
  371 FORMAT(' ELEMENTS OF (AT*C**-1*A)**-1.')
 IF (LPARER) THEN
! OUT-COMMENTED 2012-08-18.
! IF (LUNIX) THEN
!  N19=(N+1)*8
! ELSE
!  N19=(N+1)*2
! END IF
! IF (.NOT.LY) OPEN(19,ACCESS='DIRECT',FORM='UNFORMATTED',&
! FILE='DCOVA.BIN',RECL=N19)
! MAJOR CHANGE 2012-09-01.
  MAXC = 0
! WE START FROM A NEW COLUMN IN A CHUNK.
  N19 = MOD(N1,IBSS)
  IF (N19.EQ.0) THEN
   N19 = N1
  ELSE
   N19 = N1+IBSS-N19+1
  END IF
! write(*,*)' N19 ',N19
! CHANGE 2012-11-05.
  MAXC=0
  DO I = 1, NPARM
   DO J = 1, N1
    C(MAXC+J) = D0
   END DO
   MAXC = N1+MAXC
   C(MAXC-NPARM1+I) = D1
!  write(*,*)' i,maxc,nparm1,n1,x ',i,maxc,nparm1,n1,maxc-nparm1+i
   IF (MOD(I,IBSS).EQ.0.AND.I.NE.1) THEN
!    write(*,*)' call restore a n1-1,i ',N1-1,i,i-ibss+1
     CALL  RESTORE_CH(I-IBSS+1,N1-1,LF,LT,LY,LMTEST)
     MAXC=0
   END IF
  END DO
!
! write(*,*)' call restore a n1-1,NPARM ',N1-1,NPARM
! change 2012-08-18.
  CALL  RESTORE_CH(NPARM,N19,LF,LT,LY,LMTEST)
! CALL  RESTORE_CH(NPARM,N1-1,LF,LT,LY,LMTEST)
  write(*,*)' nes 5 '
! THE CALL OF cholsol is deferred until the part for all parameters
! has been established.
!bso cholsol4
   call cholsol(N1-1,0,NPARM,NPARM,lf,lf,lt,lf,lf)
   print*,'BSO CHOLSOL4'
! change 2012-08-10.
!  call cholsol(N1-1,0,NPARM,NPARM,lf,lf,lt,lf,lt)
! change back 2012-08-10.
!  call cholsol(N1-1,0,NPARM,NPARM,lf,lf,lf,lf,lt)
! HERE ERROR-ESTIMATES MUST BE TRANSFERRED FROM UNIT 99. 2012-05-18.
  inquire (99,OPENED=lopen,POS=I)
  if (lopen) then
    write(*,*)' unit 99 open , pos= ',I
  else
    write(*,*)' unit 99 not open '
  end if
  rewind(99)
  DO I = 1, NPARM
   READ(99)PW0
   IF (LMTEST) WRITE(*,*)I,PW0
   CX(I)=ABS(PW0)
  END DO
!   DO J=1,N20
! WHEN LY IS FALSE, ALL ERROR ESTIMATES ARE COMPUTED IN ONE CALL OF NES:
!    IF (LY) THEN
```



```
!     CX(I)=-PW0
!     LSMAL=LF
!     DO K=MMIN,MMAX
!      LSMAL=LSMAL.OR.ABS(C(K)).LT.1.0D-3
!      JI=JI+1
! CHANGE 2011-10-31.
!      IF (JI.GT.MAXCY) THEN
!       WRITE(*,*)' ANDEX JI EXCEEDS ',MAXCY
!       STOP
!      END IF
!      CY(JI)=C(K)
!     END DO
!     IF (LSMAL) THEN
!      WRITE(6,370)(C(K),K=MMIN,MMAX)
! 370       FORMAT(6D12.5)
!     ELSE
!      WRITE(6,372)(C(K),K=MMIN,MMAX)
  372       FORMAT(6F10.4)
!     END IF
!    END IF
!   END DO
!
!  END IF
! END DO
! IF (.NOT.LY) CLOSE(19)
 END IF
! END CALCULATION OF ERROR ESTIMATES OF PARAMETERS.
! OUTPUT OF CORRELATIONS.
 IF (LY) THEN
  WRITE(*,*)' CORRELATION MATRIX: '
  DO I=1,NPARM
! STANDARD DEVIATIONS.
   CY(100+I)=SQRT(CY(NPARM*(I-1)+I))
  END DO
  JI=0
  DO I=1,NPARM
   DO J=1,NPARM
! CORRELATIONS.
    JI=JI+1
    CY(JI)=CY(JI)/(CY(100+I)*CY(100+J))
   END DO
   WRITE(*,372)(CY(J),J=JI-NPARM+1,JI)
  END DO
 END IF

!
 IF (.NOT.LPARER) THEN
  WRITE(*,373)
  373  FORMAT(' PARAMETER    TYPE     ESTIMATE ')
 ELSE
  WRITE(6,374)
  374  FORMAT(' PARAMETER    TYPE    ',&
  ' ESTIMATE       ERROR ESTIMATE (FOR TILT: ZERO POINT ')
 END IF
! CHANGES - LPARER - 2004-06-24.
 DO 5234 I=1,NPARM
 IF (CX(I).GT.D0.AND.LPARER) CX(I)= SQRT(CX(I))
! CORRECTION 1992.07.21.
 NRE=I+ISO+NNEQ-NPARM-1
 NREL=MOD(NRE,MAXO)+1
 IF (LOBSST) THEN
  NOBLK_LOC=NRE/MAXO+1
  IF (I.EQ.1.OR.NREL.EQ.1) THEN
!     WRITE(6,*)'BLK ',NOBLK_LOC,' 12 READ FOR TRANSFER B TO C.'
  READ(16,REC=NOBLK_LOC)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
   IF (LSATAC) READ(14,REC=NOBLK_LOC)SR11,SR12,SR13,SR22,COSAZ,SINAZ
!     WRITE(6,*)'BLK ',NOBLK_LOC,' 13a READ FOR TRANSFER B TO C.'
  END IF
 END IF
```

```
 IF (LPARER) THEN
  IF (IPTYPE(I).GT.0) THEN
   WRITE(6,375)I,IPTYPE(I),B(NREL),CX(I)
  ELSE
! CHANGE 2005-11-09. ITIME0(I+1) -> ITIME0(I-1).
   WRITE(6,375)I,IPTYPE(I),B(NREL),CX(I),ITIME0(I-1)
  END IF
 ELSE
  WRITE(6,375)I,IPTYPE(I),B(NREL)
 END IF
 5234 CONTINUE
! CHANGE 2004-09-16.
  375 FORMAT(2I12,2F15.9,I12)
  CLOSE(99)
!
 5230 WRITE(*,*)' NUMBER OF CHUNKS USED = ',N1/(NN*NBB)+1
 LNBL1=MAXBL.EQ.1.AND.(.NOT.LRESOL)

END SUBROUTINE GEOCOLH

!--------------------------------------------------------------
!BLOCK DATA
! PROGRAMMED BY C.C.TSCHERNING, GEODETIC INSTITUTE, 1974.
! UPDATED: 2006-11-20 BY CCT.
! THE SUBROUTINE INITIALIZES A NUMBER OF VARIABLES. IT MAY BE
! SUBSTITUTED BY A "BLOCK DATA" CALL ON OTHER COMPUTERS.
! ON ICL-COMPUTERS, IT MUST HAVE A NAME, AND BE DECLARED AS
! AN EXTERNAL.
!

!IMPLICIT NONE

!INTEGER ::      IOBS2,IT,IA1,IC11,ITE1,IP1,IITE1,IITE,IIP1,IIP,IIE,&
!                IIE1,IT1,ITE,IP,IC1,IA,IB,IOBS1,IH,INO,IB1,K1

!LOGICAL ::      LKM,LPOT,LTERRC,LPOTIN


!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

!DATA LTERRC/.TRUE./,&
!     ITE,ITE1,IT,IP,IA,IA1/4*0,2*9/

!END BLOCK DATA

!--------------------------------------------------------------
DOUBLE PRECISION FUNCTION SYTIME(BASE,RTIME)

IMPLICIT NONE

 REAL(KIND=8) :: BASE
!     REAL*8 BASE,SYTIME
 REAL(KIND=8) :: RTIME(2),DTIME
!     REAL RTIME(2),DTIME
! SYTIME RETURNS THE USED PROCESSING TIME, AND RTIME RETURNS
! CPU TIME AND SYSTEM TIME.
! IN A NON-UNIX ENVIRONMENT RE-ACTIVATE SYTIME=1.0 AND DELETE REST.
!     SYTIME=1.0
! APP SYTIME=ETIME(A)
! APP CALL STIME(TM)
! DTIME IS SUN UNIX EXTENSION, 3F.
 SYTIME=DTIME(RTIME)
 BASE=BASE+RTIME(1)
!     WRITE(*,*)' CPUS,T1,T2 ',SYTIME,RTIME(1),RTIME(2) ,' TOTAL ',BASE
 RETURN
END FUNCTION SYTIME

!---------------------------------------------------------------
```



```
SUBROUTINE RAD(IDEG,MIN,SEC,RA,IANG)
! LAST CHANGE 2004-02-07.
! THE SUBROUTINE CONVERTS FOR IANG = 1,2,3,4 ANGLES IN (1) DEGREES, MI-
! NUTES, SECONDS, (2) DEGREES, MINUTES, (3) DEGREES AND (4) 400-DEGREES
! TO RADIANS.
 IMPLICIT NONE
 REAL*8 PHI,SEC,RA,SE
 INTEGER I,MIN,IDEG,J,IANG
!
 PHI = 3.1415926536D0
 I = 1
 IF (IDEG .LT. 0 .AND. IANG .LT. 3) I = -1
 GO TO (1,2,3,4,3,3),IANG
    1 J = 1
 IF (MIN.LT.0) J = -1
 SE =I*IDEG*3600+J*MIN*60+SEC
 I = J*I
 GO TO 5
    2 SE=I*IDEG*3600+SEC*60
 GO TO 5
    3 SE = SEC*3600
 GO TO 5
    4 SE = SEC*3240
    5 RA= I*SE/206264.806D0
 IF (RA.GT.PHI) RA = RA-PHI*2.0D0
 IF (RA.LT.-PHI) RA = RA+PHI*2.0D0
 RETURN
END SUBROUTINE RAD

! CDC FUNCTION FE(E2)
DOUBLE PRECISION FUNCTION FE(E2)
IMPLICIT NONE
 DOUBLE PRECISION :: E2
 FE=E2*(0.5+E2*(0.125+E2*(1.0/16+E2*5.0/128)))
RETURN
END

SUBROUTINE QCOMP(E2,XM,Q0,QDASH,DE2)
! PROGRAMMED MAY 1976 BY C.C.TSCHERNING, GID.LAST CH. FEB 1989.
! THE SUBROUTINE COMPUTES Q0/(EM**3*2) AND QDASH/(EM**2*6),
! USING PG EQ.(2-101).
IMPLICIT NONE

 REAL(KIND=8) :: E1,E2,XM,Q0,QDASH,DE2,EMP,EM2,DQ,D2
 INTEGER :: I

 E1=1.0D0-E2
 EM2=E2/E1
!     EM= SQRT(EM2)
 E1=E1* SQRT(E1)
 EMP=1.0D0
 QDASH=1.0D0/15.0D0
 Q0=QDASH
 DQ=Q0
 I=1
   10 I=I+1
 EMP=-EMP*EM2
 D2=2.0D0*I
 DQ=EMP/((D2+1.0D0)*(D2+3.0D0))
 QDASH=QDASH+DQ
 Q0=Q0+DQ*I
 IF (( ABS(DQ)*I).GT.(1.0D-11*Q0)) GO TO 10
 DE2=XM*E1/Q0
END SUBROUTINE QCOMP

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE GRAVC(EE,MODE,I,UREF,GAMMA)
! PROGRAMMED NOV 1973 BY C.C.TSCHERNING, GEODETIC INSTITUTE OF
```

```
! DENMARK. LAST UPDATE FEB 1989 BY CCT.
!
! THE SUBROUTINE COMPUTES BY THE CONSTANTS TO BE USED IN
! THE FORMULA FOR THE NORMAL GRAVITY, THE FORMULA FOR THE NORMAL
! POTENTIAL AND THE CHANGE IN LATITUDE WITH HEIGHT. CONSTANTS RELATED TO
! TWO DIFFERENT REFERENCE FIELDS MAY BE USED. THEY ARE STORED IN THE
! ARRAY FG IN THE VARIABLES SUBSCRIPTED FROM 1 TO 15 FOR THE FIRST
! FIELD AND FROM 16 TO 30 FOR THE SECOND ONE. THE ARRAY FJ CONTAINS THE
! ZONAL HARMONICS, WITH SIGN OPPOSITE TO THE USUAL CONVENTIONS, CF
! REF(C), EQ. (2-92).
!
! THE SUBROUTINE MAY CALCULATE THE CONSTANTS IN 5 DIFFERENT WAYS
! BASED ON 3 VALUES STORED AT CALL IN EE(1), EE(2), EE(3):
! MODE=1: FROM GM, AX, J2 AND OMEGA,
! MODE=2: FROM GM, AX, E2 AND OMEGA,
! MODE=3: FROM GM, AX,1/F AND OMEGA,
! MODE=4: FROM GAMMA, 1/F, AX AND OMEGA, WHERE GAMMA IS THE NORMAL
!         GRAVITY AT EQUATOR.
! MODE=5: AS MODE=4, BUT WITH 1928 GRAVITY FORMULA.
! AX IS THE SEMI-MAJOR AXIS, F THE FLATTENING, E2 THE SQUARE OF THE
! EXCENTRICITY, OMEGA THE SPEED OF ROTATION, GAMMA THE EQUATORIAL
! GRAVITY AND GM THE PRODUCT OF THE GRAVITY CONSTANT AND THE MASS
! OF THE EARTH.
!

 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN

 USE m_geocol_data, ONLY : FG,FJ,LP,OMEGA2,IORDER

 IMPLICIT NONE

 REAL(KIND=8) :: GM,AX,XM,GAMMA,DE2,EX,XJ2,E2,F,BX,QDASH,&
                 Q0,E1,EM2,EM,Q00,YM,AX2,E,F2,FM,TA,GAMMAP,XK,ZM,UREF,FE

 INTEGER ::      MODE,I,J,K,K2

 LOGICAL ::      LPOTSD

 REAL(KIND=8), DIMENSION(6) :: EE

!-----------------------------------------------------------------------------
!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT

!COMMON /GRAVCC/FG(30),FJ(30),LP(2),OMEGA2,IORDER

 LPOTSD=MODE.EQ.5
 IF (MODE.LT.4) GM=EE(1)
 IF (MODE.LT.5) AX=EE(2)
 OMEGA2 = EE(6)
 FG(I+9)=OMEGA2
 IF (MODE.LT.4) XM=OMEGA2*AX**3/(GM*15)
 LP(1+I/15) = .NOT.LPOTSD
 IF (MODE.EQ.4)GAMMA=EE(1)
 IF (MODE.LT.5)GO TO 1549
 AX=6378388.0
 EE(3)=297.0D0
 EE(2)=AX
 GAMMA=9.78049
!
 1549 GO TO (1541,1542,1543,1544,1544),MODE
! MODE=1:
 1541 DE2=D0
 EX=D0
 XJ2=EE(3)
 E2=D3*XJ2
 1530 EX=E2
 CALL QCOMP(E2,XM,Q0,QDASH,DE2)
 E2=D3*XJ2+DE2
 IF ( ABS(E2-EX).GT.(1.0D-10*EX)) GO TO 1530
```



```
 EE(3)=E2
 F=FE(E2)
 GO TO 1545
!
! MODE=2:
 1542 E2=EE(3)
 F=FE(E2)
 CALL QCOMP(E2,XM,Q0,QDASH,DE2)
 GO TO 1545
!
! MODE=3:
 1543 F=D1/EE(3)
 E2=F*(D2-F)
 EE(3)=E2
 CALL QCOMP(E2,XM,Q0,QDASH,DE2)
 GO TO 1545
! MODE 4 AND 5:
 1544 F=D1/EE(3)
 E2=F*(D2-F)
 EE(3)=E2
 CALL QCOMP(E2,0.0D0,Q0,QDASH,DE2)
 BX=AX*(D1-F)
! CF. PG, EQ.(2-73) USED TO COMPUTE GM.
 GM=(GAMMA+OMEGA2*AX*(D1+QDASH/(Q0*D2)))*(AX*BX)
 EE(1)=GM
 XM=OMEGA2*AX**3/(GM*15)
!
 1545 E1= SQRT(E2)
 EM2=E2/(D1-E2)
 EE(4)=EM2
 EE(5)=F
 EM= SQRT(EM2)
 Q00=Q0
 Q0=Q0*EM2*EM*D2
 YM=XM*D5*E1/Q0
 AX2=AX*AX
 IF (MODE.LT.4) BX=AX*(D1-F)
!
 FG(I+14) = AX
 FG(I+15) = GM
!
 IF (LPOTSD) GO TO 1501
 DO 1550 J = 1, 15
 1550 FJ(J+I) = D0
 FJ(I+1) = D1
 E =  E1*AX
 XM = OMEGA2*AX2*BX/GM
 F2 = F*F
 FM = F*XM
 TA =   ATAN(E/BX)
!     E3 = E2*AX2
!
 DO 1551 K = 1, 5
 K2 = 2*K
 1551 FJ(I+K2+1)=(-E2)**K*(D1-K2*YM/(K2+3))/(K2+1)
 IF (MODE.EQ.1) FJ(I+3)=-XJ2
!
! GAMMA IS THE NORMAL GRAVITY AT EQUATOR, CF. REF(C), EQ. (2-105A) AND
! (2-70). THE FIVE FOLLOWING COEFFICIENTS ARE FOUND IN REF(C) EQ.
! (2-115) AND (2-124).
! CORRECTION FEB 1989: THE FORMULAS FOR NORMAL GRAVITY AT THE ELLIPSOID
! HAVE BEEN TAKEN FROM MORITZ PAPER ON GRS80 IN THE GEODESISTS HANDBOOK.
! THE FORMULAS FOR THE TERMS DEPENDENT ON THE HEIGHT ARE FROM:
! HIRVONEN,R.A.: NEW THEORY OF GRAVIMETRIC GEODESY, ANN AC. SC. FENN,
! SER. A, III, NO. 56, 1960, EQ. (92).
 IF (MODE.LT.4) GAMMA =(GM/(AX*BX)-(D1+QDASH/(D2*Q00))*OMEGA2*AX)
 FG(I+1) = GAMMA
 GAMMAP= (GM/AX2+QDASH/Q00*OMEGA2*BX)
!     FG(I+2) = -F+D5*XM/D2+F2/D2-26*FM/7+15*XM*XM/D4
```

```
 XK = BX*GAMMAP/(AX*GAMMA)-D1
 FG(I+2) = E2/D2+XK
!     FG(I+4) = (-F2+D5*FM)/D2
 FG(I+4) = (D3*E2/D4+XK)*E2/D2
!     FG(I+3) = -D2*GAMMA*(D1+F+XM)/AX
 ZM = GM/(AX2*BX)
 FG(I+3) = -ZM*(D2-XM+(D1-27*XM/14+E2)*E2)
!     FG(I+5) = D2*GAMMA*(D3*F-D5*XM/D2)/AX
 FG(I+5) = ZM*(-5*XM+(D3-23*XM/7+D2*E2)*E2)
!     FG(I+6) = D3*GAMMA/AX2
 FG(I+6) = ZM*(D3-5*XM+D2*E2)/BX
! CF. REF(C), EQ. (2-118),(2-119).
 FG(I+11) = (D2*F-XM-F2)/D3+D2*FM/21.0D0
 FG(I+12) = -D4*F2/D5+D4*FM/7.0D0
! CF. REF(C), EQ. (2.61).
 UREF = GM*TA/E+OMEGA2*AX2/D3
 GO TO 1502
!
 1501 UREF = 62639787.0D0
 FG(1) = GAMMA
!
! CONSTANTS USED IN INTERNATIONAL GRAVITY FORMULA, CF.REF(C),(2-126),
! (2-131) AND (2-128).
 FG(4) = 4*0.0000059E0
 FG(2) = 0.0052884-FG(4)
 FG(3) = -0.30877724E-5
 FG(5) = 0.00045206E-5
 FG(6) = 7.265D-13
!
 1502 FG(I+13) = (FG(I+2)+FG(I+4))*6.47512D-2
 FG(I+7) = UREF
! FG(I+8) CONTAINS THE THIRD DERIVATIVE OF THE NORMAL GRAVITY.
 FG(I+8) = D4*FG(I+6)/(AX*D3)

END SUBROUTINE GRAVC

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

DOUBLE PRECISION FUNCTION RGRAV(I,IKP,REF1,REF2,REF3,SINLAP,H,RG,CU,SU,LSAT)
! PROGRAMMED BY C.C.TSCHERNING, GEODETIC INSTITUTE OF DENMARK,
! NOW GEOPHYSICAL INSTITUTE, UNIVERSITY OF COPENHAGEN.
! IN ALGOL MAY 1976 AND IN FORTRAN APRIL 1985, LAST CHANGE
! MAR 18, 2003 BY CCT.
! THE FUNCTION COMPUTES NORMAL GRAVITY FIELD REFERENCE VALUES.
! ALL UNITS S.I. IF LSAT IS TRUE, THE VALUES OF THE DERIVATIVES
! WILL BE GIVEN IN A SPHERICAL COORDINATE SYSTEM. OTHERWISE IT
! IS REFERENCED TO THE NORMAL GRAVITY VECTOR. (CH: JULY 1989).
!
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_geocol_data, ONLY : FG,FJ,LP,IORDER
 USE m_geocol_data, ONLY : X,Y,Z,XY,XY2,DISTO,DIST2

 IMPLICIT NONE

 REAL(KIND=8) :: OM2,H,SIN2,SINLAP,GREF,REF1,REF2,T,U,AX,GM,S,S2,C0,C1,GI,GJ,B,&
                 UREF,U2,UT,U3,U1,CU,SU,U33,URT,UTT,U22,U11,U13,RLAPLA,REF3

 INTEGER ::      M,I,KPP,N,N1,M1,M2,II,K,K1,K2,K0,IJ,J,&
                 M22,IJ4,M2II,IKC,IKP

 LOGICAL ::      LSAT

 REAL(KIND=8) ,DIMENSION(4)   :: C
 REAL(KIND=8) ,DIMENSION(10)  :: D
 REAL(KIND=8) ,DIMENSION(6)   :: SUM
 REAL(KIND=8) ,DIMENSION(3,3) :: RG

!-------------------------------------------------------------------------------
!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT
```



```

!COMMON /EUCL/X,Y,Z,XY,XY2,DISTO,DIST2

!COMMON /GRAVCC/FG(30),FJ(30),LP(2),OMX2,IORDER

 OM2=FG(I+9)
! CHANGE DEC 88: OMEGA**2 IS NOW TRANSFERRED THROUGH FG(I+9).
!IF (H.GT.1.0D4) OM2=D0
 IF (DISTO.GT.6400000.0) OM2=D0
! CHANGE 2000-02-14 BY CCT.
 M = IORDER
 IF (M.EQ.0) M = 1
! WE INCREASE THE ORDER OF DERIVATIVES TO BE COMPUTED TO 1, IF
! IORDER=0, BECAUSE WE IN THIS CASE ALSO NEED NORMAL GRAVITY, GREF.
!
 KPP= IKC(IKP)
!     KP = (KPP+2)/2
!     LZETA=LF
 SIN2 = SINLAP*SINLAP
 IF (LP(1+I/15).OR.KPP.NE.3) GO TO 1503
!  COMPUTATION OF THE REFERENCE GRAVITY IN UNITS OF MGAL, CF. REF.(C),
!  PAGE 77 AND 79. H MUST BE IN UNITS OF METERS.
 GREF = FG(I+1)*(D1+FG(I+2)*SIN2+FG(I+4)*SIN2*SIN2)+(FG(I+3)+FG(I+5)*SIN2+(FG(I+
6)-FG(I+8)*H)*H)*H
!     LZETA=LT
 REF1 = FG(I+7)
 REF2=GREF
 RGRAV=GREF
 RETURN
 1503 T = Z/DISTO
 U = XY/DISTO
 AX = FG(I+14)
 GM = FG(I+15)
 S = AX/DISTO
 S2 = S*S
!
! SUMMATION OF LEGENDRE-SERIES REPRESENTING THE NORMAL POTENTIAL,
! CF. REF(D).
 N = 10
 N1 = N+1
 M1 = M+1
 M2 = M+2
!     MM = (M1*M2)/2-1
! MM IS EQUAL TO THE TOTAL NUMBER OF DERIVATIVES COMPUTED.
 DO 11 II = 1, 10
   11 D(II) = D0
 DO 12 II = 1, 6
   12 SUM(II) = D0
 K = N
 K1 = N1
 K2 = N+2
 C1 = 12.0D0
 C0 = 11.0D0
 DO 13 K0 = 1, N1
 GI = (D2-D1/C0)*S
 GJ = -C0*S2/C1
 K2 = K1
 K1 = K
 C1 = C0
 C0 = C0-D1
 C(1) = FJ(K2+I)
 IJ = 1
 DO 14 J = 1, M
   14 C(J+1) = -C(J)*(K+J)
!
 DO 15 II = 1, M1
 M22 = M2-II
 DO 16 J = 1, M22
 IJ4 = IJ + 4
```

```
 B = D(IJ4)
 D(IJ4) = SUM(IJ)
 SUM(IJ) = GI*(D(IJ4)*T+(II-1)*D(IJ-M+II+1))+GJ*B+C(J)
 IJ = IJ+1
 C(J) = D0
   16 CONTINUE
   15 CONTINUE
 K = K-1
   13 CONTINUE
 IJ = 1
 DO 17 II = 1, M1
 B = GM
 M2II = M2-II
 DO 18 J = 1, M2II
 B = B/DISTO
 SUM(IJ) = SUM(IJ)*B
 IJ = IJ+1
   18 CONTINUE
   17 CONTINUE
!
! UREF IS THE NORMAL POTENTIAL, U1, U3 THE DERIVATIVES IN THE DIRECT-
! ION OF THE 1. AND 3. AXIS (NORTH AND UP, RESPECTIVELY), U11, U22,
! U33 AND U13 ARE THE CORRESPONDING SECOND ORDER DERIVATIVES. UT IS
! THE DERIVATIVES WITH RESPECT TO T = COS(GEOCENTRIC LATITUDE).
 UREF = SUM(1)+OM2*XY2/D2
 IF (M.EQ.0) GO TO 20
 U2 = U*U
 UT = SUM(M+2)/DISTO-Z*OM2
 U3 = SUM(2)+OM2*U*XY
 U1 = UT*U
 GREF =  SQRT(U3**2+U1**2)
! CU AND SU ARE COS AND SIN OF THE ANGLE BETWEEN THE NORMAL GRAVITY
! VECTOR AND THE RADIUS VECTOR IN THE MERIDIAN PLANE.
 IF (LSAT) THEN
  CU = D1
  SU = D0
 ELSE
  CU = -U3/GREF
  SU = -U1/GREF
 END IF
!
 REF2=GREF
 REF1=UREF
 IF (M.EQ.1) GO TO 20
 U33 = SUM(3)+OM2*U2
 URT = SUM(5)-2*OM2*Z
 UTT = SUM(6)-OM2*DIST2
 U22 = (U3-T*UT)/DISTO
 U11 = U22+U2*UTT/DIST2
 U13 = U*(URT-UT)/DISTO
 RLAPLA=U33+U22+U11-D2*OM2
 IF ( ABS(RLAPLA).GT.1.0D-12) WRITE(6,100)RLAPLA
  100 FORMAT(' *** WARNING25 *** LAPLACE OPERATOR =',E16.8)
 REF2=U1
 REF3=U3
 RG(1,1)=U22
 RG(2,2)=U11
 RG(3,3)=U33
 RG(2,3)=U13
 RG(3,2)=RG(2,3)
 RG(1,3)=D0
 RG(3,1)=D0
 RG(1,2)=D0
 RG(2,1)=D0
!
! CHANGE 2002-06-28.
   20 GO TO (1521,1522,1522,1523,1523,1524,1524,1525,1525,1525,1527,&
 1537,1527,1538,1528),KPP
 1521 RGRAV = UREF
```



```
 RETURN
 1522 RGRAV = GREF
! CHANGE 1990.11.02 TO TRANSFER GRAVITY VECTOR IN SPHERICAL FRAME.
 IF (LSAT) REF1=D0
 REF2=U1
 REF3=U3
 RETURN
 1523 RGRAV = U33*CU*CU+2*CU*SU*U13+SU*SU*U11
 RETURN
 1524 IF (LSAT) GO TO 1522
 RGRAV = U1
 REF1=U3
 REF2=GREF
 RETURN
 1525 IF (LSAT) THEN
  RGRAV=U13
 ELSE
  RGRAV = -(U13*CU*CU+(U11-U33)*CU*SU)
 END IF
 RETURN
 1537 RGRAV=U11*CU*CU-U13*D2*CU*SU+U33*SU*SU
 RETURN
 1527 RGRAV = D0
 RETURN
 1538 RGRAV= U22
 RETURN
 1528 RGRAV=(U22-U11*CU*CU+D2*U13*CU*SU-U33*SU*SU)
!     DDU/DDX-DDU/DDY.
!
END FUNCTION RGRAV

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE EUCLID(COSLAP,SINLAP,COSLOP,SINLOP,H,E2,AX)
! PROGRAMMED BY C.C.TSCHERNING, GEODETIC INSTITUTE OF DENMARK, 1974.
! UPDATES: NONE.
! COMPUTATION OF EUCLIDIAN COORDINATES X,Y,Z , DISTANCE AND SQUARE OF
! DISTANCE FROM Z-AXIS XY, XY2 AND DISTANCE AND SQUARE OF DISTANCE FROM
! THE ORIGIN DISTO AND DIST2 FROM GEODETIC COORDINATES REFERING TO AN
! ELLIPSOID HAVING SEMI-MAJOR AXIS EQUAL TO AX AND SECOND EXCENTRICITY
! E2.

 USE m_geocol_data, ONLY : X,Y,Z,XY,XY2,DISTO,DIST2

 IMPLICIT NONE

 REAL(KIND=8) :: DN,COSLOP,SINLOP,COSLAP,SINLAP,H,E2,AX
!AND USE DSQRT, DCOS AND DSIN IN THE FOLLOWING.

!COMMON /EUCL/X,Y,Z,XY,XY2,DISTO,DIST2

 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

END SUBROUTINE EUCLID

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE ICOSYS(I,IP,GM,AX,E2,F,UREF,GAMMA)
! PROGRAMMED BY C.C.TSCHERNING, GEODETIC INSTITUTE OF DENMARK, 1986.
! LAST MODIFICATION: 2013-12-02  BY CCT.
 USE m_geocol_data, ONLY : EE0,DSHIF0,MODEC0
 USE m_geocol_data, ONLY : SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS3,EPS2,EPS1,DL,AX2,E
```

```
22

 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN

IMPLICIT NONE

 REAL(KIND=8) :: UREF,GAMMA,GM,AX,E2,F

 INTEGER ::      MODEC,I1,I,J,IP

 REAL(KIND=8), DIMENSION(6) :: EE
 REAL(KIND=8), DIMENSION(7) :: DSHIFT

! THE PROGRAM INITIALIZES COORDINATE SYSTEM PARAMETERS ACCORDING TO
! THE VALUE OF THE PARAMETER I WHICH SPECIFIES HOW
! THE NORMAL POTENTIAL IS DEFINED, AND THEN THE 3 PARAMETERS
! DEFINING THE SYSTEM:
! MODEC      EE (1)     EE (2)    EE (3)
!   1         GM         AX        J2 (=-C(2,0))
!   2         GM         AX        E2
!   3         GM         AX        1/F
!   4,5     GAMMA        AX        1/F.
! WHERE GM IS THE PRODUCT OF THE MASS AND THE GRAVITY CONSTANT IN
! M**3/S**2, GAMMA THE EQUATORIAL NORMAL GRAVITY IN M/S**2, AX
! THE SEMI-MAJOR AXIS OF THE REFERENCE ELLIPSOID IN M, J2 THE
! THE (NOT NORMALIZED) 2.ORDER ZONAL HARMONIC, E2 THE SECOND
! EXCENTRICITY (AX**2-BX**2)/AX**2), AND F THE FLATTENING (AX-BX)/AX,
! WHERE BX IS THE SEMI-MINOR AXIS. MODEC0=5 GIVES THE INTERNATIONAL
! ELLIPSOID (1928) AND THE CASSINIS GRAVITY FORMULA, AND SUPPOSES A
! POTSDAM CORRECTION OF 13.7 MGAL MUST BE APPLIED TO MEASURED
! GRAVITY VALUES (LPOTSD=.TRUE.).
!
! I = 0: PARAMETERS IN CCOSYS, I=1: ED1950 WITH NORTH-SEA DATUM
! SHIFT AND ADDITIONAL DELTA(LAMBDA)=-0.5 ARCSEC,
! I = 2: COMMON ED1950 DATUM SHIFT FROM EDOC2 WITH DZ CORRECTION,
! I = 3: NAD1927 WITH NEW MEXICO DATUM-SHIFT, I = 4: GRS1967,
! I = 5: GRS1980, I = 6: NWL9D, I = 7: EGM08, TIDE FREE.
! I = 8: BEST CURRENT FOR FAROE ISLAND REGION,
! I = 9: ED1950 ADOPTED FOR FINLAND (LONGITUDE CHANGED).
! I =10: IAG-75, I = 11: KRASSOWSKI S. 42/57 (DDR),
! I =12: GERMAN DHDN SYSTEM ON BESSEL ELLIPSOID.
! I =13: ENGLAND/WALES SHIFT
! I= 14: REP. IRELAND SHIFT OF GPS/LEV  (2002-03-20).
! I =15: GRIM, I =16: TOPEX, I =17: WGS84, I =18: WGS84 rev. 1. (2011-05-16).
!
!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT
!COMMON /ITRANC/SINLA0,COSLA0,RLONG0,DSHIFT(7),AX2,E22

 EE(6)=(0.729211515D-4)**2
 I1=I+1
 DO 10 J=1,7
   10 DSHIFT(J)=D0
 IF (I1.GT.19.OR.I1.LT.0) THEN
! CHANGED 2014-02-15.
  WRITE(*,*)' REFERENCE MODEL UNDEFINED, MODEL NO= ',I
  STOP
 END IF
!
 GO TO (20,21,22,23,24,25,26,27,28,30,31,32,33,34,35,36,37,&
 38,39),I1
!
  100 FORMAT('+',18X,' USER DEFINED SYSTEM.')
   20 DO 11 J=1,7
 IF (J.LT.4) EE(J)=EE0(J)
   11 DSHIFT(J)=DSHIF0(J)
 MODEC=MODEC0
 WRITE(6,100)
 GO TO 29
!
```



```
  101 FORMAT('+',18X,' ED1950, NORTH-SEA.')
   21 MODEC=5
 DSHIFT(1)=-89.5
 DSHIFT(2)=-93.8
 DSHIFT(3)=-124.6
 DSHIFT(6)=0.17
 DSHIFT(7)=1.4D-6
 WRITE(6,101)
 GO TO 29
!
  102 FORMAT('+',18X,' ED1950 WITH DATUM SHIFT FROM EDOC2.')
   22 MODEC=5
 DSHIFT(1)=-81.0
 DSHIFT(2)=-113.3
 DSHIFT(3)=-118.8+2.5
 WRITE(6,102)
 GO TO 29
!
  103 FORMAT('+',18X,' NAD1927, NEW MEXICO VALUES, DLON=-0.7".')
   23 EE(1)=3.9860094D14
 EE(2)=6378206.4
 EE(3)=294.98
 DSHIFT(1)=-22.0
 DSHIFT(2)=157.0
 DSHIFT(3)=176.0
 DSHIFT(6)=-0.7
 MODEC=3
 WRITE(6,103)
 GO TO 29
!
  104 FORMAT('+',18X,' GRS1967.')
   24 MODEC=1
 EE(1)=3.98603D14
 EE(2)=6378160.0
 EE(3)=0.0010827
 WRITE(6,104)
 GO TO 29
!
  105 FORMAT('+',18X,' GRS1980.')
   25 MODEC=1
 EE(1)=3.986005D14
 EE(2)=6378137.0D0
 EE(3)=0.00108263
 EE(6)=(7.292115D-5)**2
 WRITE(6,105)
 GO TO 29
!
  106 FORMAT('+',18X,' NWL9D=NWSC9Z2.')
   26 MODEC=3
 EE(1)=3.986008D14
 EE(2)=6378145.0
 EE(3)=298.25
 DSHIFT(3)=2.5
 DSHIFT(6)=-0.5
 DSHIFT(7)=-0.4D-6
 WRITE(6,106)
 GO TO 29
!
  107 FORMAT('+',18X,' BEST CURRENT EGM2008, TIDE FREE.')
   27 MODEC=3
! CHANGED 2013-12-02 BY CCT. VALUES FROM EGM2008 HOME PAGE.
 EE(1)=3.986004418D14
 EE(2)=6378136.58D0
 EE(3)=298.257686D0
!EE(3)=0.484165143790815D-03*SQRT(5.0D0)
 WRITE(6,107)
 GO TO 29
!
  108 FORMAT('+',18X,' BEST CURRENT FOR FAEROE ISLAND REGION.')
```

```
   28 MODEC=3
 EE(1)=3.986005D14
 EE(2)=6378135.2
 EE(3)=298.2572D0
 WRITE(6,108)
 GO TO 29
!
  109 FORMAT('+',18X,' ED1950, ADOPTED FOR FINLAND.')
   30 MODEC=5
 DSHIFT(1)=-89.5
 DSHIFT(2)=-93.8
 DSHIFT(3)=-124.6
 DSHIFT(6)=-2.23
 DSHIFT(7)=1.4D-6
 WRITE(6,109)
 GO TO 29
!
  110 FORMAT('+',18X,' IAG-75.')
   31 MODEC=3
 EE(1)=3.986005D14
 EE(2)= 6378140.
 EE(3)= 298.257
 WRITE(6,110)
 GO TO 29
!
  111 FORMAT('+',18X,' KRASSOWSKY ELL. WITH SHIFT FOR DDR.')
   32 MODEC=3
 EE(1)=3.986005D14
 EE(2)=6378245.0D0
 EE(3)=298.30D0
 DSHIFT(1)=  45.5D0
 DSHIFT(2)=-126.6D0
 DSHIFT(3)= -70.2D0
 WRITE(6,111)
 GO TO 29
!
  112 FORMAT('+',18X,' BESSEL ELL. WITH DHDN SHIFT FOR W-GERMANY.')
   33 MODEC=2
 EE(1)=3.986005D14
 EE(2)=6377397.155D0
 EE(3)=0.006674372D0
 DSHIFT(1)=-580.00D0
 DSHIFT(2)= -80.90D0
 DSHIFT(3)=-395.30D0
! ORDER MAY BE WRONG - HERE ROT X,Y,Z.
 DSHIFT(4)=-0.35D0
 DSHIFT(5)= 0.10D0
 DSHIFT(6)=-3.58D0
 DSHIFT(7)=-11.1D-6
 WRITE(6,112)
 GO TO 29
!
  113 FORMAT('+',18X,' ENGLAND/WALES GPS DATUM SHIFT. ')
   34 MODEC=1
 EE(1)=3.986005D14
 EE(2)=6378137.0D0
 EE(3)=0.00108263
 EE(6)=(7.292115D-5)**2
 DSHIFT(1)=-2.92D0
 DSHIFT(2)=-6.17D0
 DSHIFT(3)= 2.46D0
 WRITE(*,113)
 GO TO 29
!
  114 FORMAT('+',18X,' REP. IRELAND GPS DATUM SHIFT. ')
   35 MODEC=1
 EE(1)=3.986005D14
 EE(2)=6378137.0D0
 EE(3)=0.00108263
```



```
 EE(6)=(7.292115D-5)**2
 DSHIFT(1)= 1.498D0
 DSHIFT(2)=15.872D0
 DSHIFT(3)= 0.374D0
!     DSHIFT(1)= 3.642D0
!     DSHIFT(2)=17.461D0
!     DSHIFT(3)=-1.152D0
 WRITE(*,114)
 GO TO 29
!
  115 FORMAT('+',18X,' GRIM ')
   36 MODEC=3
 EE(1)=3.986004369D14
 EE(2)=6378136.46D0
 EE(3)=298.25765D0
 WRITE(*,115)
 GO TO 29
!
  116 FORMAT('+',18X,' TOPEX ')
   37 MODEC=3
 EE(1)=3.986004415D14
 EE(2)=6378136.3D0
 EE(3)=298.257D0
 WRITE(*,116)
 GO TO 29
!
  117 FORMAT('+',18X,' WGS84 ')
   38 MODEC=3
 EE(1)=3.986005D14
 EE(2)=6378137.0D0
 EE(3)= 298.257222101D0
 WRITE(*,117)
 GO TO 29
!
  118 FORMAT('+',18X,' WGS84 rev1 ')
   39 MODEC=3
 EE(1)=3.986004418D14
 EE(2)=6378137.0D0
 EE(3)=298.257223563D0
 WRITE(*,118)
!
   29 CALL GRAVC(EE,MODEC,IP,UREF,GAMMA)
 GM=EE(1)
 AX=EE(2)
 E2=EE(3)
 F=1/EE(5)
 DSHIFT(7)=DSHIFT(7)+D1
 DSHIFT(4)=DSHIFT(4)/RADSEC
 DSHIFT(5)=DSHIFT(5)/RADSEC
 DSHIFT(6)=DSHIFT(6)/RADSEC
 DX  = DSHIFT(1)
 DY  = DSHIFT(2)
 DZ  = DSHIFT(3)
 EPS3=DSHIFT(4)
 EPS2=DSHIFT(5)
 EPS1=DSHIFT(6)
 DL=  DSHIFT(7)

END SUBROUTINE ICOSYS

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE COUT(NO,LONC,LSMAL,LFULLO,IORDER)
! PROGRAMMED 1974 BY C.C.TSCHERNING, UPDATED: 2008-06-09.
! THE SUBROUTINE WRITES ON UNIT 6 (1) STATION NUMBER,(2) COORDINATES,
! (3) OBSERVED VALUE (IN ORIG.REF.SYSTEM),(4) DIFFERENCE BETWEEN OBSER-
! VED AND PREDICTED VALUE, (5) ERROR OF PREDICTION, (6) TRANSFORMATION
! VALUE, (7) SPHERICAL HARMONIC SERIES CONTRIBUTION, (8) RESULT OF COLL
! I AND (9) COLL.II, (10) SUM OF QUANTITIES (7)-(9) AND (11) SUM OF (6)-
```

```
! (9) - ALL IF MEANINGFULL. IN CASE WE ARE DEALING WITH A PAIR OF DE-
! FLECTIONS, (LONC = FALSE), THE CORRESPONDING QUANTITES FOR ETA ARE
! WRITTEN A LINE BELOW.
! WHEN LPUNCH IS TRUE, THE FOLLOWING OF THE ABOVE MENTIONED QUANTITIES
! ARE WRITTEN ON UNIT 17: (1) AND (2), AND WHEN LOUTC IS TRUE (3) - (5)
! AND ELSE (11), (10) AND (5).
! WHEN LSMAL IS TRUE, A 4-DIGIT LAYOUT IS IN USE.
! WHEN LNUOUT IN COMMON /OUTC/ IS TRUE, ONLY OUTPUT OF STATION NUMBER.
!

 USE m_geocol_data, ONLY : CLATD,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,RDI,ISATP
 USE m_geocol_data, ONLY : ALLREF,ALLGG,ALLCOL,ALLPRE,ALLTRA,ALLPR1,ALLERR,&
                           ALLVAR,ALLIN,LALLCO
 USE m_geocol_data, ONLY : INUMR,NO1,I=>K2,I1=>K3,K2P3,I2=>K4,I4=>IU,I21=>K21,&
                           I31=>IU1,IANG,LPUNCH,&
                           LSTNO=>LTERM,LTERMA,LTERMO,LOUTC,LTRAN,LNERNO,LK30,LK
31,&
                           LWRSOL,LSTOP,LK2EQ4,LNUOUT
 USE m_data,        ONLY : OBS
 USE m_geocol_data, ONLY : C20IN,G1,G2,CM3,CMM2,CM1

IMPLICIT NONE

 LOGICAL ::             LONC,            LNTRAN,                        &
                                                          LSMAL,LFULLO,&
                 LNSMAL
! WHEN LFULLO IS TRUE WILL ALL VALUES COMPUTED FROM A SPHERICAL
! HARMONIC EXPANSION BE OUTPUT, I.E. 6 FOR SECOND DERIVATIVES.
! CHANGE 2005-01-25.

 INTEGER ::                                          II,I0,M,NO2,NO,&
                   J,I9,IORDER,IU

 REAL(KIND=8) :: D9

 REAL(KIND=8), DIMENSION(10) :: OBN

!-----------------------------------------------------------------------------
!COMMON /OUTC/INUMR(12),NO1,I,I1,K2P3,I2,I4,I21,I31,IANG,LPUNCH,LTERMA,LTERMO,LS
TNO,LOUTC,LTRAN,LNERNO,LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT

!COMMON /GPOTC0/C20IN,G1(3),G2(3,3),CM3,CMM2,CM1
! THE VARIABLES ARE ONLY USED IF LFULLO IS TRUE, AND CONTAINS THE
! DERIVATIVES OF THE SPHERICAL HARMONIC EXPANSION.

!COMMON /CALLCO/ALLREF(3,3),ALLGG(3,3),ALLCOL(3,3),ALLPRE(3,3),ALLTRA(3,3),ALLPR
1(3,3),ALLERR(3,3),ALLVAR(3,3),ALLIN(3,3),LALLCO
! ADDED 2006-01-20 TO PERMIT OUTPUT OF ALL DERIVATIVES. IORDER IS ORDER
! OF DIFFERENTIATION.

 D9=1.0D9
! VARIABLE LNTRAN NOT ASSIGNED VALUE BEFORE 2005-09-07.
 LNTRAN=.NOT.LTRAN
!
!write(*,*)' 9164 I,I1,K2P3,I2,I21,I31 ',I,I1,K2P3,I2,I21,I31,LK2EQ4
!write(*,*)OBS
 IF ( ABS(SLAT) .LT. 0.1D-6) SLAT = 0.0D0
 IF ( ABS(SLON) .LT. 0.1D-6) SLON = 0.0D0
! THIS IS DONE IN ORDER TO AVOID PRINTING OF SIGN, WHEN THE ARC-SECOND
! PART IS NEAR TO ZERO,(OR ZERO IS REPRESENTED BY A SMALL NEGATIVE NUM-
! BER).
 II = 2
 IF (LK30) II = 3
 OBN(1) = OBS(1)
 IF (LWRSOL.OR.(.NOT.LPUNCH)) GO TO 8010
 IF (LOUTC) GO TO 8007
 OBN(1) = OBS(1)
 I0 = 2
 OBN(2) = OBS(I4)
```



```
 IF (LNTRAN) GO TO 8031
 OBN(3) = OBS(I4-1)
 I0 = I0+1
 8031 IF (LNERNO) GO TO 8032
 I0 = I0+1
 OBN(I0) = OBS(I)
 8032 IF (LONC) GO TO 8034
 I0 = I0+1
 OBN(I0) = OBS(I31)
 IF (LNTRAN) GO TO 8033
 I0 = I0+1
 OBN(I0) = OBS(I4+9)
 8033 IF (LNERNO) GO TO 8034
 I0 = I0+1
 OBN(I0) = OBS(I21)
 8034 I2 = I0
 GO TO 8010
!
 8007 DO 8008 M = 1, I
 8008 OBN(M) = OBS(M)
 IF (LONC) GO TO 8010
 DO 8009 M = 2, I
 8009 OBN(M+I-1) = OBS(M+10)
!
 8010 IF (.FALSE.) WRITE(*,*)' OBS ',OBS(2),OBS(3),OBS(4)
 IF (.NOT.LNUOUT) GO TO 8035
 NO2=NO1-1
 NO2=MOD(NO2,6)+1
 INUMR(NO2)=NO
 IF (NO2.EQ.6.OR.LSTOP) WRITE(6,278)(INUMR(IU),IU=1,NO2)
  278 FORMAT(' ',6I11)
!
 8035 IF (IANG.NE.3.AND.IANG.NE.6)LSMAL=.FALSE.
! CHANGE 2004-02-06. AND 2005-09-06.
 IF (ABS(OBN(1)).GE.1.0D8) OBN(1)=9999999.9D0
 DO J=2,K2P3
  IF (LSMAL) THEN
   IF (ABS(OBS(J)).GE.10.0d0) THEN
    LSMAL=.FALSE.
! CHANGE 2012.12.08
!   OBS(J)=99.9999D0
!    IF (.NOT.LNUOUT) WRITE(*,*)' WARNING26: FORMAT EXPLODED '
   END IF
  ELSE
   IF (ABS(OBS(J)).GE.1000.0d0) THEN
    OBS(J)=9999.99D0
    IF (.NOT.LNUOUT) WRITE(*,*)' WARNING27: FORMAT EXPLODED '
   END IF
  END IF
 END DO
! ADDED 2006-08-20.
 GO TO (8000,8001,8002,8002,8002,9002),IANG
 8000 IF (LK31.AND.(.NOT.LNUOUT)) WRITE(6,800) NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLO
N,(OBS(J),J=1,K2P3)
 IF (.NOT.(LK31.OR.LNUOUT)) WRITE(6,810) NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,(OBS
(J),J=1,K2P3)
  800 FORMAT(' ',I7,2(I5,I3,F6.2),F9.1,/,2F7.2,F6.2,7F7.2)
 IF (LPUNCH) WRITE(17,810)NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,(OBN(J), J = 1, I2)
  810 FORMAT(I8,2(I4,I3,F6.2),F10.2,7F9.3)
 IF (LONC.AND.LWRSOL) WRITE(17,820)NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,OBS(1),OBS
(II),OBS(II+1),LSTOP
  820 FORMAT(I5,2(I4,I3,F6.2),3F9.3,L2)
 IF ((.NOT.LONC).AND.LWRSOL) WRITE(17,821)NO,IDLAT,MLAT,SLAT,IDLON,MLON,SLON,OBS
(1),OBS(II),OBS(II+10),OBS(II+1),OBS(II+11),LSTOP
  821 FORMAT(I5,2(I4,I3,F6.2),5F9.3,L2)
 GO TO 8004
!
 8001 IF (LK31.AND.(.NOT.LNUOUT)) WRITE(6,801) NO,IDLAT,SLAT,IDLON,SLON,(OBS(J),
J=1,K2P3)
```

```
 IF (.NOT.(LK31.OR.LNUOUT)) WRITE(6,811) NO,IDLAT,SLAT,IDLON,SLON,(OBS(J),J=1,K2
P3)
  801 FORMAT(I11,I5,F6.2,I8,F6.2,F9.1,/,2F7.2,F6.2,7F7.2)
 IF (LPUNCH) WRITE(17,811)NO,IDLAT,SLAT,IDLON,SLON,(OBN(J),J=1,I2)
  811 FORMAT(I10,I5,F6.2,I8,F6.2,F10.2,7F9.3)
 IF (LONC.AND.LWRSOL) WRITE(17,822) NO,IDLAT,SLAT,IDLON,SLON,&
 OBS(1),OBS(II),OBS(II+1),LSTOP
  822 FORMAT(I10,2(I4,F6.2),3F9.3,L2)
 IF ((.NOT.LONC).AND.LWRSOL) WRITE(17,823) NO,IDLAT,SLAT,IDLON,SLON,OBS(1),OBS(I
I),OBS(II+10),OBS(II+1),OBS(II+11),LSTOP
  823 FORMAT(I6,2(I4,F6.2),5F9.3,L2)
 GO TO 8004
!
 8002 IF (LSMAL) THEN
  IF (LK31.AND.(.NOT.LNUOUT)) WRITE(6,1802) NO,SLAT,SLON,(OBS(J),J=1,K2P3)
  IF (.NOT.(LK31.OR.LNUOUT)) THEN
! CHANGE 2010-10-04.
   LNSMAL=.FALSE.
   DO J=2,K2P3
    LNSMAL=ABS(OBS(J)).GE.10.0
   END DO
   IF (LNSMAL) THEN
    WRITE(6,1807)NO,SLAT,SLON,(OBS(J),J=1,K2P3)
   ELSE
    WRITE(6,1806)NO,SLAT,SLON,(OBS(J),J=1,K2P3)
   END IF
  END IF
 1802  FORMAT(' ',I10,2(F12.6,' '),F9.1,/,10F7.4)
 1806  FORMAT(' ',I10,2(F12.6,' '),F9.1,5F7.4)
 1807  FORMAT(' ',I10,2(F12.6,' '),F9.1,5F7.2)
  IF (LPUNCH) THEN
   IF (LFULLO) THEN
    IF (IORDER.EQ.0) THEN
     WRITE(17,1815)NO,SLAT,SLON,OBN(1),G1(1)
 1815     FORMAT(I10,2(F12.6,' '),F10.2,F14.4)
    ELSE
     IF (IORDER.EQ.1) THEN
! OUTPUT OF FULL GRAVITY VECTOR.
      WRITE(17,1814)NO,SLAT,SLON,OBN(1),G1(1),G1(2),G1(3)
 1814      FORMAT(I10,2(F12.6,' '),F10.2,3F12.8)
     ELSE
      WRITE(17,1813)NO,SLAT,SLON,OBN(1),G2(1,1)*D9,G2(2,2)*D9,&
      G2(3,3)*D9,G2(1,2)*D9,G2(1,3)*D9,G2(2,3)*D9
! OUTPUT OF 6 GRAVITY GRADIENTS.
 1813      FORMAT(I10,2(F12.6,' '),F10.2,6F12.4)
     END IF
    END IF
   END IF
   IF (LALLCO) THEN
    IF (IORDER.EQ.0) THEN
     WRITE(17,1915)NO,SLAT,SLON,OBN(1),ALLCOL(1,1)
 1915     FORMAT(I10,2(F12.6,' '),F10.2,F14.4)
    ELSE
     IF (IORDER.EQ.1) THEN
! OUTPUT OF FULL GRAVITY VECTOR (IN MGAL).
      WRITE(17,1914)NO,SLAT,SLON,OBN(1),ALLCOL(1,1),&
      ALLCOL(2,1),ALLCOL(3,1)
 1914      FORMAT(I10,2(F12.6,' '),F10.2,3F10.4)
     ELSE
      WRITE(17,1913)NO,SLAT,SLON,OBN(1),ALLCOL(1,1),&
      ALLCOL(2,2),ALLCOL(3,3),ALLCOL(1,2), ALLCOL(1,3),&
      ALLCOL(2,3)
! OUTPUT OF 6 GRAVITY GRADIENTS.
 1913      FORMAT(I10,2(F12.6,' '),F10.2,6F12.4)
     END IF
    END IF
   END IF
   IF (.NOT.(LFULLO.OR.LALLCO)) THEN
    WRITE(17,1812)NO,SLAT,SLON,(OBN(J), J = 1, I2)
```



```
   END IF
  END IF
 1812  FORMAT(I10,2(F12.6,' '),F10.2,7F13.4)
  IF (LONC.AND.LWRSOL) WRITE(17,1824)NO,SLAT,SLON,OBS(1),&
  OBS(II),OBS(II+1),LSTOP
 1824  FORMAT(I10,2(F12.6,' '),F8.2,2F11.4,L2)
  IF ((.NOT.LONC).AND.LWRSOL) WRITE(17,1825)NO,SLAT,SLON,&
  OBS(1),OBS(II),OBS(II+10),OBS(II+1),OBS(II+11),LSTOP
 1825  FORMAT(I10,2(F12.6,' '),F10.2,4F10.6,L2)
 ELSE
  IF (LK31.AND.(.NOT.LNUOUT)) WRITE(6,802) NO,SLAT,SLON,(OBS(J),J=1,K2P3)
  IF (.NOT.(LK31.OR.LNUOUT)) WRITE(6,806) NO,SLAT,SLON,(OBS(J),J=1,K2P3)
  802  FORMAT(' ',I10,2(F12.6,' '),F9.1,/,2F7.2,F6.2,7F7.2)
  806  FORMAT(' ',I10,2(F12.6,' '),F9.1,5F7.2)
  IF (LPUNCH) THEN
   IF (LFULLO) THEN
    IF (IORDER.EQ.0) THEN
     WRITE(17,1815)NO,SLAT,SLON,OBN(1),G1(1)
    ELSE
     IF (IORDER.EQ.1) THEN
! OUTPUT OF FULL GRAVITY VECTOR.
      WRITE(17,1814)NO,SLAT,SLON,OBN(1),G1(1),G1(2),G1(3)
     ELSE
      WRITE(17,1813)NO,SLAT,SLON,OBN(1),G2(1,1)*D9,G2(2,2)*D9,&
      G2(3,3)*D9,G2(1,2)*D9,G2(1,3)*D9,G2(2,3)*D9
! OUTPUT OF 6 GRAVITY GRADIENTS.
     END IF
    END IF
   END IF
   IF (LALLCO) THEN
    IF (IORDER.EQ.0) THEN
     WRITE(17,2015)NO,SLAT,SLON,OBN(1),ALLCOL(1,1)
 2015     FORMAT(I10,2(F12.6,' '),F10.2,F12.2)
    ELSE
     IF (IORDER.EQ.1) THEN
! OUTPUT OF FULL GRAVITY VECTOR (IN MGAL).
      WRITE(17,2014)NO,SLAT,SLON,OBN(1),ALLCOL(1,1),&
      ALLCOL(2,1),ALLCOL(3,1)
 2014      FORMAT(I10,2(F12.6,' '),F10.2,3F12.2)
     ELSE
      WRITE(17,2013)NO,SLAT,SLON,OBN(1),ALLCOL(1,1),&
      ALLCOL(2,2),ALLCOL(3,3),ALLCOL(1,2), ALLCOL(1,3),&
      ALLCOL(2,3)
! OUTPUT OF 6 GRAVITY GRADIENTS.
 2013      FORMAT(I10,2(F12.6,' '),F10.2,6F14.2)
     END IF
    END IF
   END IF
   IF (.NOT.(LALLCO.OR.LFULLO)) THEN
    WRITE(17,812)NO,SLAT,SLON,(OBN(J), J = 1, I2)
   END IF
  END IF
! OUTPUT FORMAT CHANGED 9.3 -> 10.4 2005-07-27.
  812  FORMAT(I10,2(F12.6,' '),F10.2,7F10.4)
  IF (LONC.AND.LWRSOL) WRITE(17,824)NO,SLAT,SLON,OBS(1),&
  OBS(II),OBS(II+1),LSTOP
  824  FORMAT(I10,2(F12.6,' '),3F9.3,L2)
  IF ((.NOT.LONC).AND.LWRSOL) WRITE(17,825)NO,SLAT,SLON,&
  OBS(1),OBS(II),OBS(II+10),OBS(II+1),OBS(II+11),LSTOP
  825  FORMAT(I10,2(F12.6,' '),5F9.3,L2)
 END IF
 GO TO 8004
! GEOCENTRIC COORDINATES.
 9002 OBS(1)=RDI
 OBN(1)=RDI
 IF (LSMAL) THEN
  IF (LK31.AND.(.NOT.LNUOUT)) WRITE(6,9802) NO,CLATD,SLON,(OBS(J),J=1,K2P3)
  IF (.NOT.(LK31.OR.LNUOUT)) THEN
   WRITE(6,9806)NO,SLAT,SLON,(OBS(J),J=1,K2P3)
```

```
  END IF
 9802  FORMAT(' ',I10,2(F12.6,' '),F10.1,/,10F7.4)
 9806  FORMAT(' ',I10,2(F12.6,' '),F10.1,5F7.4)
  IF (LPUNCH) THEN
   IF (LFULLO) THEN
    IF (IORDER.EQ.0) THEN
     WRITE(17,9815)NO,CLATD,SLON,OBN(1),G1(1)
 9815      FORMAT(I10,2(F12.6,' '),F10.2,F12.2)
    ELSE
     IF (IORDER.EQ.1) THEN
! OUTPUT OF FULL GRAVITY VECTOR.
      WRITE(17,9814)NO,CLATD,SLON,OBN(1),G1(1),G1(2),G1(3)
 9814      FORMAT(I10,2(F12.6,' '),F10.2,3F12.8)
     ELSE
      WRITE(17,9813)NO,CLATD,SLON,OBN(1),G2(1,1)*D9,G2(2,2)*D9,&
      G2(3,3)*D9,G2(1,2)*D9,G2(1,3)*D9,G2(2,3)*D9
! OUTPUT OF 6 GRAVITY GRADIENTS IN EOETVOES.
 9813      FORMAT(I10,2(F12.6,' '),F11.2,6F12.4)
     END IF
    END IF
   END IF
   IF (LALLCO) THEN
    IF (IORDER.EQ.0) THEN
     WRITE(17,1915)NO,CLATD,SLON,OBN(1),ALLCOL(1,1)
    ELSE
     IF (IORDER.EQ.1) THEN
! OUTPUT OF FULL GRAVITY VECTOR (IN MGAL).
      WRITE(17,1914)NO,CLATD,SLON,OBN(1),ALLCOL(1,1),&
      ALLCOL(2,1),ALLCOL(2,1)
     ELSE
      WRITE(17,1913)NO,CLATD,SLON,OBN(1),ALLCOL(1,1),&
      ALLCOL(2,2),ALLCOL(3,3),ALLCOL(1,2), ALLCOL(1,3),&
      ALLCOL(2,3)
! OUTPUT OF 6 GRAVITY GRADIENTS.
     END IF
    END IF
   END IF
   IF (.NOT.(LALLCO.OR.LFULLO)) THEN
    WRITE(17,9812)NO,CLATD,SLON,(OBN(J), J = 1, I2)
   END IF
  END IF
 9812  FORMAT(I10,2(F12.6,' '),F10.1,7F13.4)
  IF (LONC.AND.LWRSOL) WRITE(17,9824)NO,CLATD,SLON,OBS(1),&
  OBS(II),OBS(II+1),LSTOP
 9824  FORMAT(I10,2(F12.6,' '),F10.1,2F11.4,L2)
  IF ((.NOT.LONC).AND.LWRSOL) WRITE(17,9825)NO,CLATD,SLON,&
  OBS(1),OBS(II),OBS(II+10),OBS(II+1),OBS(II+11),LSTOP
 9825  FORMAT(I10,2(F12.6,' '),F10.1,4F10.6,L2)
 ELSE
  IF (LK31.AND.(.NOT.LNUOUT)) WRITE(6,902) NO,CLATD,SLON,(OBS(J),J=1,K2P3)
  IF (.NOT.(LK31.OR.LNUOUT)) WRITE(6,906) NO,CLATD,SLON,(OBS(J),J=1,K2P3)
  902  FORMAT(' ',I10,2(F12.6,' '),F10.1,/,2F7.2,F6.2,7F7.2)
  906  FORMAT(' ',I10,2(F12.6,' '),F10.1,5F7.2)
  IF (LPUNCH) THEN
   IF (LFULLO) THEN
    IF (IORDER.EQ.0) THEN
     WRITE(17,9815)NO,CLATD,SLON,OBN(1),G1(1)
    ELSE
     IF (IORDER.EQ.1) THEN
! OUTPUT OF FULL GRAVITY VECTOR.
      WRITE(17,9814)NO,CLATD,SLON,OBN(1),G1(1),G1(2),G1(3)
     ELSE
      WRITE(17,9813)NO,CLATD,SLON,OBN(1),G2(1,1)*D9,G2(2,2)*D9,&
      G2(3,3)*D9,G2(1,2)*D9,G2(1,3)*D9,G2(2,3)*D9
! OUTPUT OF 6 GRAVITY GRADIENTS.
     END IF
    END IF
   END IF
   IF (LALLCO) THEN
```



```
    IF (IORDER.EQ.0) THEN
     WRITE(17,1915)NO,CLATD,SLON,OBN(1),ALLCOL(1,1)
    ELSE
     IF (IORDER.EQ.1) THEN
! OUTPUT OF FULL GRAVITY VECTOR (IN MGAL).
      WRITE(17,1914)NO,CLATD,SLON,OBN(1),ALLCOL(1,1),&
      ALLCOL(1,2),ALLCOL(1,3)
     ELSE
      WRITE(17,1913)NO,CLATD,SLON,OBN(1),ALLCOL(1,1),&
      ALLCOL(2,2),ALLCOL(3,3),ALLCOL(1,2), ALLCOL(1,3),&
      ALLCOL(2,3)
! OUTPUT OF 6 GRAVITY GRADIENTS.
     END IF
    END IF
   END IF
   IF (.NOT.(LALLCO.OR.LFULLO)) THEN
    WRITE(17,912)NO,CLATD,SLON,(OBN(J), J = 1, I2)
   END IF
  END IF
  912  FORMAT(I10,2(F12.6,' '),F11.2,7F10.4)
  IF (LONC.AND.LWRSOL) WRITE(17,924)NO,CLATD,SLON,OBS(1),&
  OBS(II),OBS(II+1),LSTOP
  924  FORMAT(I10,2(F12.6,' '),F11.2,2F9.3,L2)
  IF ((.NOT.LONC).AND.LWRSOL) WRITE(17,925)NO,CLATD,SLON,&
  OBS(1),OBS(II),OBS(II+10),OBS(II+1),OBS(II+11),LSTOP
  925  FORMAT(I10,2(F12.6,' '),F11.2,4F9.3,L2)
 END IF
 GO TO 8004
!
! WARNING: IT LOOKS LIKE THERE IS NO CONNECTION TO THIS LABLE.
 8011 IF (LNUOUT) GO TO 8012
 IF (.NOT.LSTNO) WRITE(6,850)NO
  850 FORMAT(' ',I10)
 IF (LSTNO)WRITE(6,856)
  856 FORMAT(' ')
 IF ( ABS(OBS(1)).GE.9.0D3) WRITE(6,858)OBS(1)
 IF ( ABS(OBS(1)).LT.9.0D3) WRITE(6,857)OBS(1)
  858 FORMAT(F9.1,'M')
  857 FORMAT(F8.2,'M')
!     I9=I6+I7+20
 I9=20
 IF (LSMAL) THEN
  IF (LK31.AND.(.NOT.LNUOUT))WRITE(6,1852)(OBS(J),J=2,K2P3)
  IF (.NOT.(LK31.OR.LNUOUT).AND.I9.LE.44) WRITE(6,1853)(OBS(J),&
  J=2,K2P3)
  IF (.NOT.(LK31.OR.LNUOUT).AND.I9.GT.44) WRITE(6,1851)(OBS(J),&
  J=2,K2P3)
 1852  FORMAT(/,2F7.4,F6.3,7F7.4)
 1853  FORMAT(6F8.4)
 1851  FORMAT(/,45X,6F8.4)
 ELSE
  IF (LK31.AND.(.NOT.LNUOUT))WRITE(6,852)(OBS(J),J=2,K2P3)
  IF (.NOT.(LK31.OR.LNUOUT).AND.I9.LE.44) WRITE(6,853)(OBS(J),&
  J=2,K2P3)
  IF (.NOT.(LK31.OR.LNUOUT).AND.I9.GT.44) WRITE(6,851)(OBS(J),&
  J=2,K2P3)
  852  FORMAT(/,2F7.2,F6.2,7F7.2)
  853  FORMAT(6F8.2)
  851  FORMAT(/,45X,6F8.2)
 END IF
!
 8012 IF (.NOT.(LPUNCH.OR.LWRSOL)) GO TO 8004
!
 IF (.NOT.LSTNO) WRITE(17,850)NO
 IF ( ABS(OBS(1)).GE.1.0D4) WRITE(17,858)OBS(1)
 IF ( ABS(OBS(1)).LT.1.0D4) WRITE(17,857)OBS(1)
!
 IF (LSMAL) THEN
  IF (LPUNCH.AND.I9.LE.44) WRITE(17,1853)(OBN(J),J=2,I2)
```

```
  IF (LPUNCH.AND.I9.GT.44) WRITE(17,1851)(OBN(J),J=2,I2)
  IF (LWRSOL.AND.LONC) WRITE(17,1854)OBS(II),OBS(II+1)
  IF (LWRSOL.AND.(.NOT.LONC)) WRITE(17,1855)OBS(II),OBS(II+1),&
  OBS(II+10),OBS(II+11)
 1854  FORMAT(2F8.4)
 1855  FORMAT(6F8.4)
 ELSE
  IF (LPUNCH.AND.I9.LE.44) WRITE(17,853)(OBN(J),J=2,I2)
  IF (LPUNCH.AND.I9.GT.44) WRITE(17,851)(OBN(J),J=2,I2)
  IF (LWRSOL.AND.LONC) WRITE(17,854)OBS(II),OBS(II+1)
  IF (LWRSOL.AND.(.NOT.LONC)) WRITE(17,855)OBS(II),OBS(II+1),&
  OBS(II+10),OBS(II+11)
  854  FORMAT(2F8.2)
  855  FORMAT(4F8.2)
 END IF
 8004 IF (LALLCO.AND.(.NOT.LNERNO)) THEN
  WRITE(17,1855)ALLERR(1,1),ALLERR(2,2),ALLERR(3,3),&
  ALLERR(1,2),ALLERR(1,3),ALLERR(2,3)
 END IF
!
 IF (LNUOUT) RETURN
 IF (LK2EQ4) GO TO 8005
 IF (LSMAL) THEN
 IF (LK31.AND.IANG.NE.5) WRITE(6,1803)(OBS(J+4), J = 1, I1)
 IF (LK31.AND.IANG.EQ.5) WRITE(6,1805)(OBS(J+4), J = 1, I1)
 IF (LK30.AND.(.NOT.LK31)) WRITE(6,1833)OBS(5)
 ELSE
 IF (LK31.AND.IANG.NE.5) WRITE(6,803)(OBS(J+4), J = 1, I1)
 IF (LK31.AND.IANG.EQ.5) WRITE(6,805)(OBS(J+4), J = 1, I1)
 IF (LK30.AND.(.NOT.LK31)) WRITE(6,833)OBS(5)
 END IF
! OUTPUT OF TRA,POT,COLL1,COLL2,PRED,OR PRED+TRA
 8005 IF (LONC) RETURN
! OUTPUT OF OBS,DIFR OR ERR FOR ETA
 IF(I.GT.1.AND.LK31) THEN
  IF (LSMAL) THEN
   WRITE(6,1804)(OBS(J+10),J=2,K2P3)
  ELSE
   WRITE(6,804)(OBS(J+10),J=2,K2P3)
  END IF
 END IF
 IF (LK2EQ4) RETURN
 IF (LSMAL) THEN
  IF (LK31.AND.I.GT.1) WRITE(6,1803)(OBS(J+14), J = 1, I1)
  IF (I.LE.1.AND.LK31) WRITE(6,1805)(OBS(J+14), J = 1, I1)
 ELSE
  IF (LK31.AND.I.GT.1) WRITE(6,803)(OBS(J+14), J = 1, I1)
  IF (I.LE.1.AND.LK31) WRITE(6,805)(OBS(J+14), J = 1, I1)
 END IF
  804 FORMAT(2F7.2,F6.2,8F7.2)
  803 FORMAT('+ ',18X,8F7.2)
  805 FORMAT('  ',18X,8F7.2)
 1804 FORMAT(2F7.4,F7.3,8F7.3)
 1803 FORMAT('+ ',18X,8F7.4)
 1805 FORMAT('  ',18X,8F7.4)
 IF (LSMAL) THEN
 IF (I.GT.1.AND.(.NOT.LK31)) WRITE(6,1834)(OBS(J+10),J=2,I)
 IF (I.LE.1.AND.I1.EQ.1) WRITE(6,1835)OBS(15)
 IF(I.GT.1.AND.I1.EQ.1) WRITE(6,1833) OBS(15)
 1834 FORMAT(' ',45X,2F7.4,F6.3)
 1833 FORMAT('+',66X,F7.4)
 1835 FORMAT(67X,F7.4)
 ELSE
 IF (I.GT.1.AND.(.NOT.LK31)) WRITE(6,834)(OBS(J+10),J=2,I)
 IF (I.LE.1.AND.I1.EQ.1) WRITE(6,835)OBS(15)
 IF(I.GT.1.AND.I1.EQ.1) WRITE(6,833) OBS(15)
  834 FORMAT(' ',45X,2F7.2,F6.2)
  833 FORMAT('+',66X,F7.2)
  835 FORMAT(67X,F7.2)
```



```
 END IF

END SUBROUTINE COUT

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE COMPA(LOCALVG,IKP,LONECO,IU,MODE)
! PROGRAMMED FEB 1974 BY C.C.TSCHERNING, LAST CHANGE SEP 2013.
! THE SUBROUTINE IS USED TO COMPARE OBSERVED AND PREDICTED QUANTI-
! TIES. MODE=1, INITIALISATION, =2: UPDATES SUM AND SQUARESUM,
! =3: OUTPUT MEAN AND VARIANCE.

 USE m_data,        ONLY : OBS
 USE m_geocol_data, ONLY : DXX,NUM,VARI,SCALE,SCALE2,INN,INV
 USE m_geocol_data, ONLY : INUMR,NO1,K2,K3,K2P3,K4,K21,IU1,IANG,LPUNCH,&
                           LTERM,LTERMA,LTERMO,LOUTC,LTRAN,LNERNO,LK30,LK31,&
                           LWRSOL,LSTOP,LK2EQ4,LNUOUT

IMPLICIT NONE

 LOGICAL ::      LONECO, LSMAL,LLARGE,LHTML


 INTEGER ::      I,J,MODF,                                               &
                 MODE,IKP,K23,IU,I0,IND,    IJ,NC,INW,IKP1,INV0,INW0

 REAL(KIND=8) ::                 OB1,OB2,OB3,      VG,LOCALVG

!COMMON /COM2/DXX,NUM(70),VARI(32),SCALE,SCALE2,INN,INV

!COMMON /OUTC/INUMR(12),NO1,K2,K3,K2P3,K4,IUX,K21,IU1,IANG,LPUNCH,LTERMA,LTERMO,
LTERM,LOUTC,LTRAN,LNERNO,LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT

 LHTML=LOCALVG.LT.0
 VG=ABS(LOCALVG)
 GO TO (4031,4032,4033), MODE
 4031 DXX=0.0D0
 DO 2041 I = 1, 70
 2041 NUM(I) = 0
 DO 2042 I = 1, 20
 2042 VARI(I) = DXX
 DO I=1,4
 VARI(I+20)=1.0D10
 VARI(I+16)=-1.0D10
 VARI(I+24)=-1.0D10
 VARI(I+28)=1.0D10
 END DO
!
 SCALE = VG
 SCALE2 = SCALE/2
!
 I = 1
 INN = 1
 INV = 1
 RETURN
!
 4032 J = 0
 I0=4
 K23=K2
 IF (LNERNO) I0=3
 3028 OB3 = OBS(J+2)-OBS(J+IU)
 DO 3035  I=1,I0
 GO TO (3040,3041,3042,3044),I
 3040 OB1 = OBS(J+2)
 GO TO 3043
 3041 OB1 = OBS(J+IU)
 GO TO 3043
 3042 OB1 = OB3
 OB2=OB1
```

```
 GO TO 3043
 3044 OB1  = OBS(K23)
! OB1 IS NOW EQUAL TO THE DIFFERENCE BETWEEN MEASURED AND PREDICTED
! QUANTITIES.
! COMPUTATION OF SUM AND SQUARESUM FOR PREDICTION STATISTICS.
 3043 VARI(INV+I-1) = VARI(INV+I-1)+OB1
! UPDATING MIN, MAX.
 IF (VARI(INV+I+15).LT.OB1) VARI(INV+I+15)=OB1
 IF (VARI(INV+I+19).GT.OB1) VARI(INV+I+19)=OB1
 3035 VARI(INV+I+3) = VARI(INV+I+3)+OB1**2
! COUNTING NUMBER OF OBS OF TYPE IKP.
 NUM(INN) = NUM(INN)+1
!
 IND = ( ABS(OB2)+SCALE2)/SCALE
! CORRECTION 2003-06-02.
 IF (OB2 .LT. DXX) IND = -IND
 IND = IND+11
 IF(IND.GT.21.OR.IND.LT.1)IND=22
 IND=IND+INN
 NUM(IND) = NUM(IND)+1
 IF (LONECO) GO TO 3029
 IF (INN .EQ. 24) GO TO 3036
 INN = 24
 INV = 9
 J = 10
 K23=K21
 GO TO 3028
 3036 INN = 1
 INV = 1
 3029 RETURN
!
! OUTPUT OF PREDICTION STATISTICS.
 4033 J = 0
 LSMAL=.FALSE.
  401 FORMAT(' COMPARISON OF PREDICTIONS AND OBSERVATIONS')
 WRITE(6,401)
 DO IJ = 1,2
  INN = J*23+1
  NC = NUM(INN)
  IF(NC.EQ.0) GO TO 4020
  INV = J*8+1
  INW = INV+3
  DO I = INV,INW
   VARI(I) = VARI(I)/NC
  END DO
 IF (IKP.GT.5) GO TO 4011
 GO TO (4011,4003,4004,4005,4004),IKP
 4003 WRITE(6,402)
  402 FORMAT(' GRAVITY ANOMALIES')
 GO TO 4006
 4004 IF (IJ.EQ.2) GO TO 4005
 WRITE(6,403)
  403 FORMAT('0LATITUDE COMPONENT OF DEFLECTION OF THE VERTICAL ',&
 '(KSI)')
 GO TO 4006
 4005 WRITE(6,404)
  404 FORMAT('0LONGITUDE COMPONENT OF DEFLECTION OF THE VERTICAL ',&
 '(ETA)')
 GO TO 4006
 4011 IF (IJ.EQ.2) GO TO 4012
  WRITE(6,411)IKP
 GO TO 4006
 4012 IKP1=IKP+1
  WRITE(6,411)IKP1
  411 FORMAT('0DATA TYPE =',I3)
!
 4006 IF(NC.EQ.1) GO TO 4007
  INV0=INV
  INW0=INW
```



```
  INV = INV+4
  INW = INW+4
  DO I = INV,INW
   VARI(I) =(VARI(I)-VARI(I-4)**2*NC)/(NC-1)
   LSMAL=LSMAL.OR.(VARI(I).LT.0.1D0)
   IF (VARI(I).GT.DXX) VARI(I) =  SQRT(VARI(I))
  END DO
! ADDED 2000-03-21 BY CCT.
  IF (LSMAL) THEN
   IF (LNERNO) THEN
    WRITE(6,9405)NC,(VARI(I),I=INV0,INW0-1)
   ELSE
    WRITE(6,9405)NC,(VARI(I),I=INV0,INW0)
   END IF
 9405  FORMAT(' NUMBER:',I8/&
  '          OBSERVATIONS     PREDICTIONS     DIFFERENCE',&
  '     ERROR ESTIMATES ',/,' MEAND     ',4F16.6)
! FORMAT CHANGED 2009-01-19.
 9406  FORMAT(' ST.DEVI.  ',4F16.6,/,' MAX       ',4F16.6,/,&
  ' MIN       ',4F16.6)
 9408  FORMAT(' ST.DEVI.  ',3F16.6,/,' MAX       ',3F16.6,/,&
  ' MIN       ',4F16.6)
   IF (LNERNO) THEN
    WRITE(6,9408)(VARI(I),I = INV,INW-1),(VARI(I),I= INV+12,INW+11),&
    (VARI(I),I= INV+16,INW+15)
   ELSE
    WRITE(6,9406)(VARI(I),I = INV,INW),(VARI(I),I= INV+12,INW+12),&
    (VARI(I),I= INV+16,INW+16)
   END IF
   WRITE(6,9407)VG
  ELSE
   IF (LNERNO) THEN
    WRITE(6,405)NC,(VARI(I),I=INV0,INW0-1)
   ELSE
    WRITE(6,4405)NC,(VARI(I),I=INV0,INW0)
   END IF
  405  FORMAT(' NUMBER:',I6/&
  '0         OBSERVATIONS PREDICTIONS DIFFERENCE'/&
  ' MEAND     ',4F12.2)
  4405  FORMAT(' NUMBER:',I6/&
  '0         OBSERVATIONS PREDICTIONS DIFFERENCE ERROR ESTIMATES'/&
  ' MEAND     ',4F12.2)
  406  FORMAT(' ST.DEVI.  ',4F12.2,/,' MAX      ',4F12.2,/,&
  ' MIN      ',4F12.2)
  409  FORMAT(' ST.DEVI.  ',3F12.2,/,' MAX      ',3F12.2,/,&
  ' MIN      ',3F12.2)
!
   IF (LNERNO) THEN
! CORRECTION 2003-12-19.
    WRITE(6,409)(VARI(I),I = INV,INW-1),(VARI(I),I= INV+12,INW+11),&
    (VARI(I),I= INV+16,INW+15)
   ELSE
    WRITE(6,406)(VARI(I),I = INV,INW),(VARI(I),I= INV+12,INW+12),&
   (VARI(I),I= INV+16,INW+16)
   END IF
  END IF
!
  IF (LHTML) THEN
   IF (LSMAL) THEN
    IF (LNERNO) THEN
     WRITE(6,8405)NC,(VARI(I),I=INV0,INW0-1)
    ELSE
     WRITE(6,8405)NC,(VARI(I),I=INV0,INW0)
    END IF
 8405  FORMat(' NUMBER:',I8,'<br>',/,&
 ' <table> ',/,'<tr><th>0 </th><th>        OBSERVATIONS </th>',&
  '<th>    PREDICTIONS </th><th>    DIFFERENCE</th>',&
  '<th>     ERROR ESTIMATES </th></tr>',/,'<tr><td> MEAND   </td>  ',&
  '<td>',F16.6,'</td>',&
```

```
  '<td>',F16.6,'</td>', '<td>',F16.6,'</td>', '<td>',F16.6,'</td><tr>')
 8406  FORMAT('<tr><td> ST.DEVI.</td>  ',&
  '<td>',F16.6,'</td>', '<td>',F16.6,'</td>', '<td>',F16.6,'</td>',&
  '<td>',F16.6,'</td></tr>',/,'<tr><td> MAX     </td>  ',&
  '<td>',F16.6,'</td>', '<td>',F16.6,'</td>', '<td>',F16.6,'</td>',&
  '<td>',F16.6,'</td></tr>',/,'<tr><td> MIN     </td>  ',&
  '<td>',F16.6,'</td>', '<td>',F16.6,'</td>', '<td>',F16.6,'</td>',&
  '<td>',F16.6,'</td></tr>')
 8408  FORMAT('<tr><td> ST.DEVI.</td>  ',&
  '<td>',F16.6,'</td>', '<td>',F16.6,'</td>',&
  '<td>',F16.6,'</td></tr>',/,'<tr><td> MAX     </td>  ',&
  '<td>',F16.6,'</td>', '<td>',F16.6,'</td>',&
  '<td>',F16.6,'</td></tr>',/,'<tr><td> MIN     </td>  ',&
  '<td>',F16.6,'</td>', '<td>',F16.6,'</td>',&
  '<td>',F16.6,'</td></tr>')
    IF (LNERNO) THEN
     WRITE(6,8408)(VARI(I),I = INV,INW-1),(VARI(I),I= INV+12,INW+11),&
     (VARI(I),I= INV+16,INW+15)
    ELSE
     WRITE(6,8406)(VARI(I),I = INV,INW),(VARI(I),I= INV+12,INW+12),&
     (VARI(I),I= INV+16,INW+16)
    END IF
   ELSE
    IF (LNERNO) THEN
     WRITE(6,505)NC,(VARI(I),I=INV0,INW0-1)
    elsE
     WRITE(6,5405)NC,(VARI(I),I=INV0,INW0)
    END IF
  505  FORMAT(' NUMBER:',I6,'<br>',/&
 ' <table>',/,'<tr><th>0 </th><th>        OBSERVATIONS </th>',&
  '<th>    PREDICTIONS </th><th>    DIFFERENCE</th></tr>',/,&
  '<tr><td> MEAND   </td>  ',&
 ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td></tr>' )
  5405  FORMAT(' NUMBER:',I6,'<br> ',/,&
 ' <table> ',/,'<tr><th>0 </th><th>        OBSERVATIONS </th>',&
 ' <th> PREDICTIONS</th><th> DIFFERENCE</th><th> ERROR ESTIMATES </th>'/&
  '<tr><td> MEAND  </td>   ',&
 ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td>',&
 ' <td> ',f12.2,'</td></td></TR>' )
  506  FORMAT('<tr><td> ST.DEVI.</td>  ',&
 ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td>',&
 ' <td> ',F12.2,'</td></tr>' ,/,&
 ' <tr><td> MAX  </td> ',&
 ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td>',&
 ' <td> ',F12.2,'</td></tr>' ,/,&
  '<tr><td> MIN     </td> ',&
 ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td>',&
 ' <td> ',F12.2,'</td></tr>' )
  509  FORMAT('<tr><td> ST.DEVI.</td>  ',&
 ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td>',&
 ' <td> ',F12.2,'</td></tr>' ,/,&
 ' <tr><td> MAX  </td> ',&
 ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td>',&
 ' <td> ',F12.2,'</td></tr>' ,/,&
 '<tr><td> MIN     </td> ',&
 ' <td> ',F12.2,'</td>', ' <td> ',F12.2,'</td>',&
 ' <td> ',F12.2,'</td></tr>' )
!
    IF (LNERNO) THEN
! CORRECTION 2003-12-19.
     WRITE(6,509)(VARI(I),I = INV,INW-1),(VARI(I),I= INV+12,INW+11),&
     (VARI(I),I= INV+16,INW+15)
    ELSE
     WRITE(6,506)(VARI(I),I = INV,INW),(VARI(I),I= INV+12,INW+12),&
     (VARI(I),I= INV+16,INW+16)
    END IF
   END IF
   WRITE(*,*)' </table> '
  END IF
```



```
  407 FORMAT('0DISTRIBUTION OF DIFFERENCES, UNITS:',F6.2)
 9407 FORMAT('0DISTRIBUTION OF DIFFERENCES, UNITS:',F10.6)
!
 4007 INN = INN+1
  NC = INN+20
  LLARGE=.FALSE.
  DO I=INN,NC
   LLARGE=LLARGE.OR.(NUM(I).GT.999)
  END DO
  IF (LLARGE) THEN
    WRITE(6,419)(NUM(I),I=INN,NC),NUM(NC+1)
  419  FORMAT(' NUMBER OF DIFF. IN CELLS  ',/,3(8I8),/)
   ELSE
    WRITE(6,410)(NUM(I),I=INN,NC),NUM(NC+1)
  410 FORMAT(' ',21I3,3X,I5/&
 ' -10 -9 -8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8  9',&
 ' 10 OUTSIDE',//)
   END IF
   J = J+1
 4020 CONTINUE
  END DO
 RETURN

END SUBROUTINE COMPA

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE TRANS(SINLAP,COSLAP,RLATP,SINLOP,COSLOP,RLONGP,HP,IKA,IT)
! ORIGINAL VERSION PROGRAMMED IN 1974 BY C.C.TSCHERNING, GEODAETISK
! INSTITUT. LATEST UPDATE 2003-12-14.
!
! THE SUBROUTINE TRANSFORMS THE COORDINATES FROM ONE DATUM TO ANOTHER
! USING THE 7-PARAMETER DATUM-SHIFT GIVEN BY DX,DY,DZ,DL,EPS1,EPS2,
! EPS3 AND COMPUTES THE CORRESPONDING CHANGE OF DEFLECTIONS OF THE
! VERTICAL AND HEIGHT-ANOMALIES (GEOID UNDULATIONS).
!
! INPUT OF COS AND SIN TO LATITUDE, LATITUDE, LONGITUDE (RADIANS),
! IKA SIGNIFYING WHICH KIND OF CHANGE IN THE OBSERVATIONS WE WANT TO
! COMPUTE AND IT EQUAL TO THE SUBSCRIPT IN THE ARRAY OBS IN WHICH THE
! RESULT IS RETURNED.
!
! CHANGE MADE 1987.10.07: IF LGRID IS TRUE, THE COORDINATES ARE
! NOT UPDATED. THIS ASSURES, THAT THE USE OF GPOTDR IS FASTER
! WHEN A GRID IS USED, SINCE THE LATITUDE IS NOT CHANGED WHEN
! THE DATUM IS NON-GEOCENTRIC.

 USE m_data,        ONLY : OBS,LNEWD,LRESOL,LGRID
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_geocol_data, ONLY : X,Y,Z,XY,XY2,DISTO,DIST2
 USE m_geocol_data, ONLY : SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS3,EPS2,EPS1,S1 => DL
,&
                           AX2,E22

IMPLICIT NONE

 LOGICAL ::            LREPEC,LONECO

 REAL(KIND=8) :: SINLAP,COSLAP,RLATP,SINLOP,COSLOP,RLONGP,HP,DELON,&
                 X0,Y0,Z0,X1,Y1,DELAT,&
                 XY20,XY0,DIST20,DISTO0,RLONG,S,DH,RLAT1,COSLA,RLAT,DLO,DLA,&
                 DX1,DY1,DZ1

 INTEGER ::      IKA,IT,IKP,IT1

!COMMON /DAT/LNEWD,LRESOL,LGRID
!COMMON /EUCL/X,Y,Z,XY,XY2,DISTO,DIST2
!COMMON /ITRANC/SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS3,EPS2,EPS1,S1,AX2,E22
!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT
```

```
 IKP = IKA
 IT1 = IT
 LREPEC = IKP.EQ.5.OR.IKP.EQ.7.OR.IKP.GT.25.AND.(.NOT.LNEWD)
 LONECO = .NOT.LREPEC
 IF (LREPEC) IT1 = IT+10
 IF (IKP.GT.25) IKP=IKP-10
 IF (LRESOL) IKP = 8
! WHEN A RESTART FILE IS INPUT, THE COORDINATES MAY BE IN THE
! LOCAL DATUM, BUT THE OBSERVATIONS IN THE GEOCENTRIC ONE.
 DX1=DX
 DY1=DY
 DZ1=DZ
 X1 = X
 Y1 = Y
 DELAT=RLATP*180.0/PI
 DELON=RLONGP*180.0/PI
! RC  IF (DELON.LT.0) DELON=DELON+360.0
! RC  IF (DELON.LT.280.0 .AND. DELON.GT.261.0
! RC * .AND.DELAT.LT.33.0 .AND. DELAT.GT.29.0)
! RC *CALL NADTRA(RLATP,RLONGP,DX1,DY1,DZ1)
 X0 = DX1+S1*(X+EPS1*Y-EPS2*Z)
 Y0 = DY1+S1*(Y-EPS1*X1+EPS3*Z)
 Z0 = DZ1+S1*(Z+EPS2*X1-EPS3*Y1)
 XY20= X0*X0+Y0*Y0
 XY0 =  SQRT(XY20)
 DIST20 = XY20+Z0*Z0
 DISTO0 =  SQRT(DIST20)
 IF (DISTO0.lt.600.0) THEN
! ADDED 2013-04-18.
  WRITE(*,*) ' DISTANCE FROM ORIGIN TOO SMALL, INDICATING AN ERROR, STOP ',&
  DISTO0
  STOP
 END IF
 RLONG =  ATAN2(Y0,X0)
 IF ( ABS(RLONG-RLONGP).GT.PI) RLONG=RLONG-2*PI
!
!  COMPUTATION OF THE NEW GEODETIC LATITUDE, CF REF(C) PAGE 183.
 S = AX2/ SQRT(D1-E22*SINLAP**2)
 DH = D0
 RLAT1 = RLATP
 COSLA=COSLAP
   70 RLAT = RLAT1
!
 RLAT1 =  ATAN2(Z0,XY0-E22*S*COSLA)
 COSLA =  COS(RLAT1)
 S = AX2/ SQRT(D1-E22*(D1-COSLA**2))
 DH = XY0/COSLA-S
 IF ( ABS(RLAT1-RLAT).GT.1.0D-10) GO TO 70
!
 DLO = (RLONG-RLONGP)*RADSEC
 DLA = (RLAT1-RLATP)*RADSEC
 IF ( ABS(DLA).GT.30.0 .OR.  ABS(DLO).GT.30.0) WRITE(6,96) DLA,DLO
   96 FORMAT(' ** WARNING28 ** DLA=',F10.1,', DLO =',F10.1)
 IF (LGRID) GO TO 95
!
 RLONGP = RLONG
 RLATP = RLAT1
 SINLOP= SIN(RLONG)
 COSLOP= COS(RLONG)
 COSLAP=COSLA
 SINLAP= SIN(RLATP)
 X=X0
 Y=Y0
 Z=Z0
 XY=XY0
 XY2=XY20
 DISTO=DISTO0
 DIST2=DIST20
!
```



```
   95 IF (.NOT.LRESOL) GO TO 69
! IF WE READ A SOLUTION, THEN THE OBSERVATIONS HAVE ALREADY BEEN
! CORRECTED FOR CHANGES DUE TO A DATUM-SHIFT, I.E. DH=DLO=DLA=0.0.
 DH=D0
 DLO=D0
 DLA=D0
!
   69 IF (IKP.GT.17) GO TO 75
 GO TO (71,75,72,73,72,71,72,76,76,76,71,75,75,75,75,72,73),IKP
   71 OBS(IT) = DH
 IF (.FALSE.) WRITE(*,7771)HP,DH
 7771 FORMAT(' HP,DH= ',2F14.1)
 GO TO 75
   72 OBS(IT) = -DLA
 IF (LONECO) GO TO 75
   73 OBS(IT1) = -DLO*COSLA
 GO TO 75
   76 OBS(IT) = D0
 OBS(IT1) = D0
   75 RETURN
END SUBROUTINE TRANS

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

INTEGER FUNCTION IKC(IKP)
! PROGRAMMED MAY 1984 BY C.C.TSCHERNING. UPDATE: 1999-05-20 BY CCT.
! THE SUBROUTINE CONVERT THE IDENTIFICATION NUMBERS TO THESE
! USED BY COVAX.
IMPLICIT NONE

 INTEGER :: IKP

 IF (IKP.LT.10) GO  TO 6
 IKC=IKP-10
 IF (IKC.GT.17) IKC = IKC-10
 IF (IKC .GT. 17) WRITE(6,10)IKP
   10 FORMAT(' ** WARNING29 ** IKP=',I5)
 RETURN
    6 GO TO (1,2,3,4,5),IKP
    1 IKC=1
 RETURN
    2 IKC=3
 RETURN
    3 IKC=6
 RETURN
    4 IKC=7
 RETURN
    5 IKC=6
 RETURN
END FUNCTION IKC

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 SUBROUTINE PRED(SS,AAI,IS,    ISO,II,IC,NC,IMAX1,LPRED,LBST,LCST,&
  LTCOV,LSATAC,LWAIT,NPRED)
! PROGRAMMED 1974 BY C.C.TSCHERNING. UPDATED 2013-09-16 BY CCT.
! THE SUBROUTINE COMPUTES THE COVARIANCES BETWEEN A QUANTITY OF TYPE IKP
! (HAVING COORDINATES RLONGP,RLATP) AND IC OTHER QUANTITIES HAVING COOR-
! DINATES STORED IN THE ARRAYS RLAT,RLONG. INFORMATION ABOUT THE KIND OF
! QUANTITIES IS FOUND IN THE ARRAY ANDEX.
! WHEN ONLY ONE OF THE QUANTITIES DEALT WITH IS A PARAMETER (IKP OR
! IKQ GE 100) THE PARTIAL WITH RESPECT TO THE PARAMETER IS COMPUTED
! BY 'APARM'. IF BOTH ARE PARAMETERS, THEN THE FUNCTION OF THE SUB-
! ROUTINE IS TO ASSIGN THE CONTRIBUTION FROM QUANTITIES ONLY DEPEN-
! DEPENDENT ON PARAMETERS TO THE APPROPRIATE ELEMENTS OF THE NORMAL-
! EQUATION MATRIX, C. THE ELEMENTS ARE TRANSFERRED BY CX AND COM-
! PUTED BY "CXPARM".
! BECAUSE THE SUBROUTINE MAY BE CALLED SEVERAL TIMES FOR THE SAME TYPE
! OF QUANTITY SOME COMMON VARIABLES ARE TRANSFERRED THROUGH A USE STATE-
```

```
! MENT.
! THE INTEGERS II AND IS GIVES INFORMATION ABOUT FROM WHICH PLACE IN THE
! DIFFERENT ARRAYS THE COORDINATES AND DEGREE-VARIANCES ARE TO BE PICKED
! UP (ACCORDING TO COLL.I OR II). THE COMPUTED COVARIANCES ARE STORED IN
! THE ARRAY C. THUS WHEN LBST IS TRUE, THEY ARE FIRST STORED IN ARRAY B
! AND LATER TRANSFERRED TO C.
! WHEN LCST IS TRUE, THE PROCEDURE IS USED TO COMPUTE EITHER THE COEF-
! FICIENTS OF THE NORMAL EQ. OR THE VECTOR OF COVARIANCES USED IN THE
! COMPUTATION OF THE ERROR OF PREDICTION.
! WHEN LPRED IS TRUE (COMP. OF PREDICTIONS), THE PRODUCT OF THE COVARI-
! ANCES AND THE SOLUTIONS TO THE NORMAL-EQ.(FOUND IN B) ARE ACCUMULATED
! IN THE VARIABLE PREDP (RESP. PRETAP).

 USE m_params,      ONLY : MAXO,NSAT,NPMAX,NIPT,NIPCAT,NALLCO,NDIMC,NISIZE,NCRW,
NNBL
 USE m_params,      ONLY : NSPHAR,NCX
! PARAMETERS GIVING THE SIZE OF THE ARRAYS HOLDING POTENTIAL COEFFICIENTS
! FOR MAX=360 (REALS) see m_params.f90.
 USE m_geocol_data, ONLY : C,NBL,MAXBL
!                          SR11A,SR12A,SR13A,SR22A,COSAZA,SINAZA
 USE m_geocol_data, ONLY : SLOP,CLOP,SLOQ,CLOQ
 USE m_geocol_data, ONLY : SR11,SR12,SR13,SR22,COSAZ,SINAZ,SATROT
 USE m_geocol_data, ONLY : NFILTE
 USE m_geocol_data, ONLY : LFOUR,SCFACT,SCFRDD,RDD
 USE m_data,        ONLY : ITRACE,NDSET,LCZERO,LCREF,LLCOEE,LFOURI
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_geocol_data, ONLY : CTIME,FOUCOF,LFOUR,NFOURI

 USE m_data,        ONLY : BSIZE,ANDEX,PREDP,PRETAP,HCZERO,NCZERO

 USE m_geocol_data, ONLY : B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,  &
                           WOBS,ISAT,SINLOP,COSLOP,BSIZEN,BSIZEE,COSLAP, &
                           SINLAP,RLONGP,RP,CAZP,SAZP,HP,RLATP,ICZERO,   &
                           NI,IKP,IKQ,ISATP,NOBLK,LONECO,LNKSIP,LNETAP,   &
                           LDEFVP,LOBSST,LSTART
 USE m_geocol_data, ONLY : IORDER
 USE m_geocol_data, ONLY : ALLREF,ALLGG,ALLCOL,ALLPRE,ALLTRA,ALLPR1,ALLERR,&
                           ALLVAR,ALLIN,LALLCO
 USE m_geocol_data, ONLY : A,SX,RB2,TT,BZ,KT,KT1,K,IIZ,JJ,N3,KK,&
                           KXQ,KXP,ND,NRX,ND1,ND2,LSPHAR,IR
!COMMON /DDY/A,SX,RB2,TT,BZ,KT,KT1,K,IIZ,JJ,N3,KK,KXQ,KXP,ND,NRX,ND1,ND2
 use m_cholsol,     only : NN
! NN is the block-size.
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,HCMAX,KCI
 USE m_geocol_data, ONLY : CCR,SIGMAX,CCV,NC1,NC2,LOCAL,LSUM
!COMMON /CMCOV/CCI(24),CCR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),&
!HCMAX,CCV(2,2),DCX(36),KVI(37),N1,N2,LOCAL,LSUM
 USE m_data,        ONLY : KSAT,LTESTS,NWAR,LY
!USE m_geocol_data, ONLY : COVX,CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
 USE m_geocol_data, ONLY :      CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
                           NDP,NDQ,LX,LNX
!COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFA,KSAT(17,2),LSATPP,LSATQ,NDX1(5),&
!NDX2(5),NDP,NDQ,NWAR,LY,LX(7,5),LNX(7,5),LTESTS
 USE m_data,        ONLY : K7,K9,K11,K13,K15,K17,K19,K21,K23,K8,C11,&
                           J2,I3,I4,LN,L
!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)
 USE m_geocol_data, ONLY : STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE, COST2Q,SINT2
Q
 USE m_data,        ONLY : IPTYPE,ITIME,ITIME0,NPARM,NPARM1
 USE m_geocol_data, ONLY : SFACT,FPERIO,IPACAT,ITROLD,ICODE,MAXPAR,MP,&
                           IPA,NCXLAS
 USE m_geocol_data, ONLY : HQD,RLATD,SINLATD,COSLATD,RLONGD,SINLOND,COSLOND
 USE m_geocol_data, ONLY : HQE,RLATE,SINLATE,COSLATE,RLONGE,SINLONE,COSLONE

IMPLICIT NONE

 LOGICAL ::      LPRED,LBST,LCST,LCOD,LDIFP,LDIFQ,&
                 LSAME,LEQP,LREPER,LREROW,LDEFVQ,LKSIQ,LETAQ,LNBC      ,&
```



```
!                LSAME,LEQP,LREPER,LREROW,LDEFVQ,LKSIQ,LETAQ,LNBC,LTEST,&
                 LMEANP,LMEANQ,LEQANP,LEQANQ,&
                 LTCOV,LTCOVN,LNPARQ,LROT,&
                 LMEAP1,LMEAQ1,LCOERR,LLCOER,LSATAC,&
                 LSATPQ,LCHECK,CHECKC,LCTIME,&
                 LWAIT,LSATP,LPARMP,LPARMQ,LMULTI,&
                            LOLDP,LOLDQ,       LDGP,LDGQ,LSAT,LOPEN
!                LSPHP,LSPHQ,LOLDP,LOLDQ,LSUMC,LDGP,LDGQ,LSAT,LOPEN

 INTEGER ::      omp_get_thread_num
 INTEGER ::                          NPRED,       IQ,MT,MR,MY,ME,MC,MB,&
!INTEGER ::      ITRGAP,ITRACK,ITOLD,NPRED,IRSZE,IQ,MT,MR,MY,ME,MC,MB,&
                 IKB,IETA,JR,        ICREL,JRNEXT,JRSTOP,IMAX1,&
!                IKB,IETA,JR,JR1,JR2,ICREL,JRNEXT,JRSTOP,IMAX1,&
                 NSTEPP,IT,IPT,ILAST,IC,NR1,NRREL,ICBL,NSTEPQ,&
                 IFIRST,IA,I,IIT,IKA,ISO,IIR,KPP,KQQ,IAA,IBB,ICC,NC,&
!                IFIRST,IA,I,IIT,I0,IKA,ISO,IIR,KPP,KQQ,IAA,IBB,ICC,IDD,NC,&
                 INDX,NEWCX,IGG,IKC,     IS,NERCOV,NREL,MAXX,MAXO9,MAXO6,&
                            IDSET,IMSET,     &
!                NISTART,ITMODE,ITM0,ITMOD,IDSET,IMSET,ILC, &
                 M,KA,KB,KC,KD,KE,IB,MA,KS,KX,KY,KZ,NDT,I1,IP,K1,J1,NCASE,&
                 NDTOT,II,IIX,KP,KQ,KKP1,KKP2,KKQ1,KKQ2,&
                 idd,ISATQ,&
                 KV10,KV11,KV12,KV13,KV14,KV15,KV20,KV21,KV22,KV23,KV24,KV25,&
                 N1,N2,IX,I2,K2,KG,NR,JX,IIY,K6,M6,J,M1,IJ,KM,IX1,JX1,NFOUR
 INTEGER(KIND=8) :: NISTART,ILC
! CHANGE 2013-03-01.
!COMMON /DDY/A,SX,RB2,TT,BX,KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,NR,ND1,ND2
!REAL(KIND=8) :: COVX1,ROTSUM,ROTSUA,&
 REAL(KIND=8) :: ROTSUM,ROTSUA,STEPQN,STEPQE,CFA,&
                 COSB,SINB,SINT,ERRCOV,COMEAN,&
                 PSI2,PSI3,R2,R4,COZERO,APARM,SS,S,AAI,BSIZQN,BSIZQE,&
                 COV,COSLAQ,SINLAQ,SINLOQ,COSLOQ,DLAT,DLONG,CAZQ,SAZQ,&
                 COSDLO,T,T1,SINDLO,SIDLO2,SINDLA,GM,HHP,HHQ,PSI,&
                 COST,CCR10,CCR11,CI11,CI12,CI16,CI17,CI18,CI19,CI20,&
                 CS,SC,SCC,CC,CCS,CSC,CPCD,CQSD,CQCD,SSX,P2,P3,CNX,CN33,&
                 SI,D27,RL,RL1,RL2,RL3,RL4,RL6,RL5,RL7,CN23,RPP,RB2X,&
                 RP2,RP2Q,RQ2,RPQ,FAK5,RPQ2,RN,RNL,GI,GJ,S3,S4,S5,R2PQ,D37,&
                 BX,D3132,D313,C11P,C11Q,CF,COSLAPP,&
                 SP,CP,SQ,CQ,SD,CD,ST,T2,S2,CPSD,RQ,PREDP0,DIFPRE,DEGRAD
!                SP,CP,SQ,CQ,SD,CD,ST,T2,S2,CPSD,RQ,DX,PREDP0,DIFPRE

 REAL(KIND=8), DIMENSION(NDIMC)  :: CT
!REAL(KIND=8), DIMENSION(MAXO)   :: CT
 REAL(KIND=8), DIMENSION(MAXO)   :: RLONGQ
 REAL(KIND=8), DIMENSION(MAXO)   :: RLATQ
 REAL(KIND=8), DIMENSION(NSAT)   :: SR11A,SR12A,SR13A,SR22A,COSAZA,SINAZA
 REAL(KIND=8), DIMENSION(MAXO)   :: BT
 REAL(KIND=8), DIMENSION(MAXO)   :: WOBSQ
! LOCAL ARRAYS TO BE USED IN OMP.
 REAL(KIND=8), DIMENSION(MAXO,6) :: ALLCOV
 REAL(KIND=8), DIMENSION(2200)   :: SM
 REAL(KIND=8), DIMENSION(NPMAX)  :: CPX
! THIS ARRAY IS USED TO TRANSFER CONTRIBUTIONS TO PARAMETERS FROM UNIT 2.
 REAL(KIND=8), DIMENSION(6*NSAT) :: ROTSAT
 REAL(KIND=8), DIMENSION(3,3)    :: SROTQ,SRTTQ,SROTP
 REAL(KIND=8), DIMENSION(3,3,3,3) ::  COVX
 REAL(KIND=8), DIMENSION(6,6)    :: DD
 REAL(KIND=8), DIMENSION(6,8)  :: CX
 REAL(KIND=8), DIMENSION(8,8,MAXO)  :: CXI
 REAL(KIND=8), DIMENSION(8,5)  :: CN,DCN
 REAL(KIND=8), DIMENSION(5)    :: CZ
 REAL(KIND=8), DIMENSION(6)    :: RM,DC,Q,CY,V,U,G,P,R,SS1
 REAL(KIND=8), DIMENSION(17)   ::  C11X
 REAL(KIND=8), DIMENSION(36)   ::  D
 REAL(KIND=8), DIMENSION(24)   ::  CTI
!REAL(KIND=8), DIMENSION(36,MAXO)   ::  DXX

 REAL(KIND=8), DIMENSION(56)   ::  CRR
```

```
! change 2012-10-30.
 INTEGER,      DIMENSION(37)   :: KVI
 LOGICAL,      DIMENSION(7)    :: LNY
 LOGICAL,      DIMENSION(7,5)    :: LNZ

 IF (LALLCO) THEN    ! all derivatives computed simultaneously
! CHANGE 2008-09-24 AND 2012-08-04..
  ALLPRE=D0
 END IF
 STEPQN=STEQN
 STEPQE=STEQE
 DEGRAD=3.1415639d0/180.0d0
 KVI=KCI
 N1=NC1
 N2=NC2
 MAXX=MAXO
 LMULTI = LF
 LSATP  = ISATP.GE.1
 SRTTQ=0
 SRTTQ(1,1)=D1
 SRTTQ(2,2)=D1
 SRTTQ(3,3)=D1
!ADDITION 2013-04-30.
 IF (ISATP.EQ.0.OR.ISATP.EQ.1.OR.ISATP.EQ.3) THEN
  SATROT=SRTTQ
 END IF
 LSATPP = LSATP
 LTESTS = LCZERO.and.(.false.)
! change 2012-09-13.
 LNBC   = .NOT.LBST.AND.LCST
 LCOD   = IKP.GE.6 .AND. IKP .LT.10
!write(*,*)' 9694 IKP,LCOD,LPARMP ',IKP,LCOD,LPARMP
 LPARMP = IKP.GE.100
 IKB    = IKP
 IF (IKP.GE.26) IKB = IKP-10
 IF (IKP.EQ.5)  IKB = 3
 IETA   = IKB+1
 JR     = II
 IDSET=0
 NR     = ISO+1
 ICREL  = ISO
 JRNEXT = NR
 PRETAP = D0
 PREDP  = D0
 PREDP0 = D0
 N2=NC2
 NC1    = IMAX1
 N1=NC1
! N1 WAS NOT INITIALIZED CORRECTLY BEFORE 2012-10-02.
 NREL=NR
!write(*,*)' 9723 NR,NRREL ',NR,NRREL
 NOBLK=1
 HHP=HP
 IF (.NOT.(LPARMP.OR.LCOD)) THEN
  COVX=D0
  LMEANP = ABS(BSIZEN).GT.1.0D-6
  LEQANP = LMEANP.AND.BSIZEE.GT.1.0D-6
  LMEAP1 = LMEANP.AND.BSIZEN.LT.D0
  NSTEPP = 1
  IF (LMEANP) THEN
   NSTEPP = NFILTE
   LDIFP=NFILTE.LT.0
!  WRITE(*,*)' P',LDIFP,NFILTE,RLATD,RLONGD
  END IF
  CCI(10)= SS
  CCI(9) = (RE+SS)**2
  CCI(8) = AAI
  CCR(4) = SINLAP
  CCR(6) = COSLAP
```



```
  RP     = RE+HHP
  CCR(2) = HHP
  IF (LSATP) THEN
   CCR(10) = D1
   CCR10   = D1
  ELSE
   CCR(10) = GMC/RP**2
   CCR10   = GMC/RP**2
  END IF
 ELSE
  IT = IKP-100
! IT IS THE PARAMETER NUMBER (1 - NPARM).
  IF(IT.GT.0) IPT = IPTYPE(IT)
  IPA    = 0
  ILAST  = ISO
 END IF
!
 IF (LPRED) THEN
  IMSET=NDSET(INT(ISO/10)+1)
 ELSE
  DO IIR=1,NDSET(INT(ISO/10)+1)
!  WRITE(*,*)' 9637 IIR,ANDEX(IIR+ISO),IC,ISO,JRNEXT ',IIR,&
!  ANDEX((IIR+ISO)*2),IC,ISO,JRNEXT
   IF (IC+ISO.GE.JRNEXT.AND.IIR+ISO.LE.ANDEX((IIR+ISO)*2)) THEN
    IMSET=IIR
!   WRITE(*,*)' 9641 IIR+ISO,ANDEX(IIR+ISO)*2),IIR,IMSET',&
!   IIR+ISO,ANDEX((IIR+ISO)*2), IIR,IMSET
   END IF
   JRNEXT=ANDEX((IIR+ISO)*2)
  END DO
 END IF
 JRNEXT=1+ISO
 IR = 0
! change 2012-08-18.
 IF (LCST) THEN
  IF (LPRED) THEN
   IF (NPRED.GT.0) THEN
    NISTART=(NPRED-1-INT((NPRED-1)/NN)*NN)*NC+1
    if (LTCOV) write(*,*)' 9758 NPRED,NISTART ,NC',NPRED,NISTART,NC
   ELSE
    NISTART=1
   END IF
  ELSE
   ILC =INT((IC-1)/NN)*NN
   NISTART=INT(IC*(IC-D1)/D2+1-ILC*(ILC+D1)/D2)
! CHANGE 2013-03-01.
  END IF
  IF (NISTART.LT.0) THEN
   write(*,*)' 9762 NI,IC,ILC,NISTART,IMSET at start ',&
   NI,IC,ILC,NISTART,IMSET
   STOP
  END IF
 ELSE
  NISTART=1
 END IF
 DO IDSET=1,IMSET
  LSATQ=ISAT(JR).GE.1
  ISATQ=ISAT(JR)
  LSATPQ=(ISATP.GE.1.OR.ISAT(JR).GE.2).and.(.not.LSPHAR)
! change 2012-09-06.
! LSATPQ=ISATP.GE.1.OR.ISAT(JR).GE.2
! IF (LSATPQ.and.LPRED) THEN
   KPP = KVI(6)
   KQQ = KVI(7)
!  write(*,*)' 10181 KPP,KQQ ',kpp,kqq
! END IF
  LROT=ISATP.GE.2.OR.ISAT(JR).GE.2
  BSIZQN = BSIZE(JR)
  BSIZQE = BSIZE(JR+1)
```

```
! ADDITION 1998.03.20
! write(*,*)' LLCOEE,LFOURI ',LLCOEE(JR),LFOURI(JR),LFOURI(JR+1),&
! scfrdd(jr),scfrdd(jr+1)
  IF (LLCOEE(JR) .AND.(.NOT.LPRED)) THEN
! LCOERR INITIALIZED 2012-11-18.
   LCOERR=LT
   LFOUR=LFOURI(JR)
   LCTIME=LFOURI(JR+1)
   IF (LFOUR) THEN
    NFOUR=NFOURI(JR)
   ELSE
    SCFACT=SCFRDD(JR)
    RDD=SCFRDD(JR+1)
   END IF
  ELSE
   LCOERR=LF
  END IF
  LMEANQ = ABS(BSIZQN).GT.1.0D-7
  LEQANQ = LMEANQ.AND.BSIZQE.GT.1.0D-6
  LMEAQ1 = LMEANQ.AND.BSIZQN.LT.D0
  NSTEPQ=1
  IF (LMEANQ) THEN
   IF (LMEAQ1) THEN
    NSTEPQ=NFILTE
    LDIFQ=NFILTE.LT.0
!   write(*,*)' Q ',LDIFQ,NFILTE,BSIZQN
   ELSE
    NSTEPQ=5
   END IF
   CALL ICMEAN(ABS(BSIZQN),STEPQN,NSTEPQ,COSSQN,SINSQN,D1,D0,LT,LF)
   IF (LEQANQ) CALL ICMEAN(BSIZQE,STEPQE,5,COSSQE,SINSQE,D1,D0,LT,LF)
  END IF
  IKQ = ANDEX(JR+1)
  JR = JR+2
!
  LREPER = IKQ.EQ.5 .OR. (IKQ.GT.25.AND.IKQ.LT.36)
! LREPER IS TRUE IF A ROW CAN BE REPEATED.
  LDEFVQ =(IKQ.GE.3.AND.IKQ.LE.5).OR.(IKQ.GE.16.AND.IKQ.LE.19)
! CHANGE 2013-03-15.
!  .OR.(IKQ.GE.16.AND.IKQ.LT.36)
  LPARMQ = IKQ.GE.100
! LPARM IS TRUE IF P DEPENDS ON PARAMETERS.
  LNPARQ = .NOT.LPARMQ
! LNPARQ IS TRUE IF Q IS NOT ASSOCIATED WITH PARAMETERS.
! SEE REF(J).
  LMULTI=((.not.LCZERO).and.(.NOT.LSPHAR).AND.(.NOT.LREPER).AND.LNPARQ &
  .AND.(.NOT.LPARMP).AND.(.NOT.LCOD)).AND.((.NOT.LMEANQ).AND.(.NOT.LMEANP) &
  .AND.(.NOT.LMEAQ1).AND.ISO.EQ.0.and.(.NOT.LSUM).AND.LONECO.AND.(.NOT.LCREF))
! change 2012-10-01.
  IF (.NOT.(LPARMQ.OR.LCOD)) THEN
   KVI(7) = IKC(IKQ)
   KQQ=KVI(7)
   KCI=KVI
   CALL COVBX(SM,LSATPQ,IS)
   KVI=KCI
   IF (LMULTI) THEN
!   IF (LPRED) WRITE(*,*)' MULTIPROCESSING POSSIBLE,&
!   JR,NDSET(INT(ISO/10)+1),IC = ',JR,NDSET(INT(ISO/10)+1),IC
   END IF
  ELSE
   KT     = 0
   KG     = 0
   IPA    = 0
   ILAST  = IIR-1
  END IF
!
  IF (LOBSST) THEN
   nr1=jrnext-1
   ICBL=NR1/MAXO+1
```



```
   NRREL=MOD(NR1,MAXO)+1
!  write(*,*)' NR1,ICBL,NRREL ',NR1,ICBL,NRREL
! CHANGE 2004-07-09. LSTART INTRODUCED TO ASSURE CORRECT STARTING POINT
! FOR OBSERVATION RECORD.
   IF (NRREL.EQ.1.OR.JRNEXT.EQ.1.OR.LSTART) THEN
    LSTART=LF
    INQUIRE(16,OPENED=LOPEN)
    MAXO9=MAXO*9*8
    MAXO6=MAXO*6*8
!   WRITE(*,*)' LOPEN ',LOPEN,MAXO9
    IF (.NOT.LOPEN) THEN
!    WRITE(*,*)' LOPEN ',LOPEN,MAXO9
      OPEN(16,ACCESS='DIRECT',FORM='UNFORMATTED',RECL=MAXO9)
!     OPEN(16,ACCESS='DIRECT',FORM='UNFORMATTED',STATUS='SCRATCH',RECL=MAXO9)
    END IF
!   WRITE(6,*)'BLK ',ICBL,' 13 READ FOR TRANSFER B TO C.',NR1
    READ(16,REC=ICBL)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
! ERROR - CHANGE 2012-05-08.
! READ(16,REC=NOBLK)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
    BT=B
! THIS IS TO HAVE A LOCAL COPY OF THE SOLUTION VECTOR, B.
    IF (LSATAC) THEN
     INQUIRE(14,OPENED=LOPEN)
!    WRITE(*,*)' LOPEN ',LOPEN,MAXO6
     IF (.NOT.LOPEN) THEN
      OPEN(14,ACCESS='DIRECT',FORM='UNFORMATTED',RECL=MAXO6)
!     OPEN(14,ACCESS='DIRECT',FORM='UNFORMATTED',STATUS='SCRATCH',RECL=MAXO6)
     END IF
     READ(14,REC=ICBL)SR11A,SR12A,SR13A,SR22A,COSAZA,SINAZA
     IF (NI.LT.0) WRITE(*,*)' ROTSAT1, SR11A1 ',ROTSAT(1),SR11A(1)
!   WRITE(6,*)'BLK ',ICBL,' UNIT 14 READ .'
    END IF
!   WRITE(6,*)'BLK ',ICBL,' UNIT 14 READ .'
    IF (IR.EQ.1.AND.ISO.NE.0) THEN
     ICREL=MOD(ISO,MAXO)
    ELSE
     ICREL=0
    END IF
   END IF
  ELSE
   BT=B
   SR11A=SR11
   SR12A=SR12
   SR13A=SR13
   SR22A=SR22
   COSAZA=COSAZ
   SINAZA=SINAZ
  END IF
!
  JRSTOP=MIN(IC,ANDEX(JR-2)-1)
! write(*,*)' LOOP: JRNEXT,MIN(IC+ISO,ANDEX(JR-2)-1),NISTART,JRSTOP ',&
! JRNEXT,ANDEX(JR-2)-1,NISTART,JRSTOP
! write(*,*)' 10414 LMULTI ',LMULTI
  IF (LMULTI) THEN
   KP=KVI(6)
   KQ=KVI(7)
! INTRODUCING LOCAL VARIABLES TO BE USED IN OMP LOOP.
   KKP1=KSAT(KP,1)
   KKP2=KSAT(KP,2)
   KKQ1=KSAT(KQ,1)
   KKQ2=KSAT(KQ,2)
   KV10=KVI(10)
   KV11=KVI(11)
   KV12=KVI(12)
   KV13=KVI(13)
   KV14=KVI(14)
   KV15=KVI(15)
   KV20=KVI(20)
   KV21=KVI(21)
```

```
   KV22=KVI(22)
   KV23=KVI(23)
   KV24=KVI(24)
   KV25=KVI(25)
   CI11=CCI(11)
   LDGP=KP.EQ.3
   LDGQ=KQ.EQ.3
   NDTOT=NDP+NDQ+1
   NCASE=NDP+1+NDQ*3
   IKA = IKQ
   SP=SINLAP
   COSLAPP=COSLAP
   CP=COSLAPP
   SLOP = SINLOP
   SLOP = SINLOP
   CLOP = COSLOP
   RP=RE+HHP
   RP2=RP*RP
   RPP=RP
   CT=D0
   JRSTOP=MIN(IC,ANDEX(JR-2)-1)
   IF (IKQ.GE.26) IKA=IKQ-8
   LSAT=LSATPQ
   LOLDP = (KP.EQ.12) .OR. (KP.EQ.14) .OR. LSAT
   LOLDQ = (KQ.EQ.12) .OR. (KQ.EQ.14) .OR. LSAT
   GM=GMC
   LTESTS=LF
   RLATQ=RLAT
   RLONGQ=RLONG
   WOBSQ=WOBS
   if (ND1.eq.1.and.lpred.and.ltests) then
    nwar=nwar+1
    write(*,*)' new multi ',nwar
   end if
!  PREDP0=D0
   CRR=CCR
   CCR11=CCR(11)
   CTI=CCI
   IIT=IIZ
   CXI=D0
   C11X=C11
   SROTP=TRANSPOSE(SATROT)
   LNY=LN
   LNZ=LNX
   RB2X=RB2
!
!$OMP PARALLEL DEFAULT (none) SHARED (MAXX,LTCOV,KV10,KV11,KV12,KV13,KV14,KV15,K
V20,KV21,KV22,KV23,KV24,KV25,KKP1,KKP2,KKQ1,KKQ2,JJ,LSAT,SATROT,RB2X,N1,CIX,NDX2
,SIGMA,K19,LNZ,LSATPP,C11X,IIZ,IIT,SIGMAX,L,LNY,ND,ND2,A,NDP,ND1,NDTOT,NDQ,D3,D5
,D4,KP,KQ,J2,I4,I3,NCASE,LOLDP,CXI,LOLDQ,LDGP,LDGQ,PREDP,LTESTS,BT,D0,D1,D2,CP,S
P,COSLAPP,SINLAP,IS,LFOUR,LCTIME,KPP,KQQ,CTIME,RPP,HHP,WOBSQ,LDEFVQ,LT,RP2,SR11A
,SR12A,SR13A,SR22A,KSAT,NERCOV,CT,LCST,NPRED,NISTART,ITRACE,LCOERR,LPRED,COSAZA,
SINAZA,COSAZ,SINAZ,CCR10,COSLAT,LDEFVP,SAZP,CAZP,LF,SINLAT,SINLON,COSLON,RLATP,R
LATQ,RLONGP,RLONGQ,HQ,JRNEXT,JRSTOP,IC,JR,LSATQ,GM,GMC,ISAT,LROT,CCI,CI11,RE,LSA
TPQ,LSATP,SROTP,NFOURI,NFOUR,FOUCOF,PI,ISATQ,RDD,SCFACT)
!$OMP DO PRIVATE(idd,IB,MA,KA,KB,KC,KD,KE,MY,MC,MB,ME,MT,MR,IQ,J1,CFA,IJ,K2,K,I2
,J,M1,KM,IIX,IIY,IX,IX1,JX1,M6,CF,CZ,K6,C11P,C11Q,CPSD,S2,ST,T2,SI,S3,S4,S5,R2PQ
,D37,CN23,D3132,D313,I1,IP,K1,RL,RL1,RL2,RL3,RL4,RL5,RL6,RL7,RP2Q,RQ2,RPQ,FAK5,R
PQ2,RN,RNL,GI,GJ, CX,RM,DC,Q,BX,CY,V,U,G,P,R,SS1,CI12,CN,DCN,SQ,CQ,SD,CD,CS,SC,S
CC,CC,CCS,CSC,CPCD,CQSD,CQCD,SSX,P2,P3,CNX,CN33,D27,S,D,DD,M,KS,KY,KX,KZ,NR,RQ,C
I20,T1,T,HHQ,CCV,COSB,SINB,COST,SINT,COVX,LLCOER,ERRCOV,PSI,NDT,NR1,NRREL,ICREL,
COV,COSLAQ,SINLAQ,SINLOQ,CRR,COSLOQ,DLAT,DLONG,CAZQ,SAZQ,COSDLO,SIDLO2,SINDLO,CC
R11,CI16,CI17,CI18,CI19,SINDLA,SRTTQ,SROTQ,CTI,COZERO,R2,R4,PSI2,PSI3)   SCHEDUL
E (dynamic)
 DO IIR = JRNEXT,JRSTOP
!  WRITE(*,*)' IR NR ', IR,NR
! PRINT *,'   I am thread number ',omp_get_thread_num(),IIR,PREDP
    NR=IIR
    NR1=NR-1
```



```
    NRREL=MOD(NR1,MAXX)+1
    ICREL=NRREL
! change, so that ICREL is independent of the actual thread.
!   ICREL=ICREL+1
!   if (IIR.NE.NRREL) write(*,*)' IIR,NNREL ',IIR,NRREL
    COV = D0
!
! CHANGE 2013-02-16 IIR CHANGED.
    COSLAQ = COSLAT(ICREL)
    SINLAQ = SINLAT(ICREL)
    SINLOQ = SINLON(ICREL)
    COSLOQ = COSLON(ICREL)
!   COSLAQ = COSLAT(IIR)
!   COSLAQ = COS(RLATQ(IIR))
!   SINLAQ = SINLAT(IIR)
!   SINLAQ = SIN(RLATQ(IIR))
!   SINLOQ = SINLON(IIR)
!   SINLOQ = SIN(RLONGQ(IIR))
!   COSLOQ = COSLON(IIR)
!   COSLOQ = COS(RLONGQ(IIR))
    DLAT = -(RLATP-RLATQ(ICREL))
    DLONG = RLONGP-RLONGQ(ICREL)
    HHQ = HQ(ICREL)
!   DLAT = -(RLATP-RLATQ(IIR))
!   DLONG = RLONGP-RLONGQ(IIR)
!   HHQ = HQ(IIR)
    RQ = RE+HHQ
    IF (LSATQ) THEN
     CAZQ = COSAZ(ICREL)
     SAZQ = SINAZ(ICREL)
!    CCR(11) = D1
     CCR11   = D1
    ELSE
!    CCR(11) = GMC/RQ**2
     CCR11   = GMC/RQ**2
    END IF
!
    COSDLO  = COS(DLONG)
    SIDLO2  = SIN(DLONG/D2)**2
    SINDLO   = SIN(DLONG)
! CHANGE 2012-08-23.
    CTI(20) = D0
    CI20=D0
    CTI(16) = SIDLO2
    CI16=SIDLO2
    CTI(17) = SIN(DLAT/D2)
    CI17=SIN(DLAT/D2)
!   SINDLA  = CTI(17)**2
    SINDLA  = CI17**2
    CTI(18) = COS(DLAT)
    CI18=COS(DLAT)
    CTI(19) = COS(DLAT/D2)
    CI19=COS(DLAT/D2)
!   IF (ABS(DLAT).GT.1.0D-3.AND.ABS(DLONG*COSLAQ).GT.1.0D-3) THEN
    IF (ABS(DLAT).GT.1.0D-3.OR.ABS(DLONG*COSLAQ).GT.1.0D-3) THEN
     T =  SINLAQ*SINLAP+COSLAPP*COSLAQ*COSDLO
    CXI(7,3,IIR)=T
     T1 = D1-T
    ELSE
     CTI(20) = D1
     CI20=D1
! T IS COSINE TO THE SPHERICAL DISTANCE BETWEEN P AND Q, CF.REF(B),
! EQ.(57).
     T1 = D2*(SINDLA+COSLAPP*COSLAQ*SIDLO2)
     T = D1-T1
!    IF (LTCOV) WRITE(*,*)' 10137 T,IIR ',T,IIR
    CXI(7,3,IIR)=T
    END IF
!   CXI(7,3,IIR)=T
```

```
    CXI(7,4,IIR)=CI16
    CXI(7,5,IIR)=CI17
    CXI(7,6,IIR)=CI18
    CXI(7,7,IIR)=CI19
    CXI(8,1,IIR)=COSLAQ
    CXI(8,2,IIR)=SINLAQ
    CXI(8,3,IIR)=COSDLO
    CXI(8,4,IIR)=SINDLO
    CXI(8,5,IIR)=RLATQ(IIR)
    CXI(8,6,IIR)=RLONGQ(IIR)
    CXI(8,7,IIR)=COSLAPP
    CXI(8,8,IIR)=SINLAP
!
    if ((.not.lpred).and.iir.le.0)write(*,1558)IIR,T,HHP,HHQ,COSLAQ,SINLAQ,&
    COSLAPP,SINLAP,COSDLO,SINDLO,CI16,CI17,CI19
 1558 format(' IIR,T,HHP,HHQ,CQ,SQ,CP,SP,CL,SL,CI16,CI17,CI19 ',i4,f6.3,2d12.5,&
  9f6.3)
  SQ=SINLAQ
  CQ=COSLAQ
  SD=-SINDLO
  CD= COSDLO
  RQ = RE+HHQ
!
! COMPUTATION OF THE CONSTANT USED TO CONVERT THE COVARIANCE INTO
! PROPER UNITS.
 CI12 = CI11/(RPP**KV22*RQ**KV23*CCR11**KV21*CCR10**KV20)
!CTI(12) = CTI(11)/(RPP**KV22*RQ**KV23*CCR(11)**KV21*CCR(10)**KV20)
!
 S = RB2X/(RPP*RQ)
 if (iir.lt.0.and.(.not.lpred)) write(*,*)' 10152 IIR,S,P,RQ,RB2X ',IIR,S,RPP,RQ
,RB2X
!
! COMPUTATION OF THE QUANTITIES D(1)-D(36),CF.REF(A),SECTION 3.
! (MODIFIED ACCORDING TO REF.(C)).
 D=D0
 IF (ND.NE.0) THEN
! go-to statements changed to if-then-else 2012-09-17.
!IF (ND.EQ.0) GO TO 55
!
  D(1) = D1
  CS = CP*SQ
  SC = SP*CQ
  SCC = SC*CD
  CC = CP*CQ
  CCS = CC*SD
  CSC = CS*CD
  IF (CI20.LE.0.5) THEN
! CF. REF.(D), EQ. (7) AND (8).
! ERROR 2002-10-06. CHANGE OF SIGN ON CCI(17)*CCI(19).
   D(2)= D2*(CI17*CI19+SP*CQ*CI16)
!D(2)= D2*(CCI(17)*CCI(19)+SP*CQ*CCI(16))
   D(7)= D2*(-CI17*CI19+SQ*CP*CI16)
!D(7)= D2*(-CCI(17)*CCI(19)+SQ*CP*CCI(16))
   IF (ABS(D(2)-CS+SCC).GT.1.0D-6 .OR. ABS(D(7)-SC+CSC).GT.1.0D-6) THEN
    WRITE(*,*) ' WARNING37Y D(2),CS+SCC,IIR ',D(2),(CS-SCC),IIR
    WRITE(*,*) ' WARNING38Y D(7),SC-CSC,NRREL ',D(7),(SC-CSC),NRREL
    WRITE(*,*)' TCI161719,SPCPSQCQ ',T,CI16,CI17,CI19,SP,CP,SQ,CQ
   END IF
  ELSE
   D(2) = CS-SCC
   D(7) = SC-CSC
!  CPSD = CP*SD
  END IF
! CPSD MOVED 2012-11-22.
  CPSD = CP*SD
!
  CPCD = CP*CD
  CQSD = CQ*SD
  CQCD = CQ*CD
```



```
  D(3) = CQSD
  D(13)=-CPSD
!
  IF (ND.NE.1) THEN
   SSX = SP*SQ
   D(8) = CC+SSX*CD
! CF. REF.(D). EQ.(9).
   IF(CI20.LT.0.5) THEN
! D(8)=CI18-D2*SP*SQ*CI16
! D(8)=CCI(18)-D2*SP*SQ*CCI(16)
! IF (ABS(D(8)-(CC+SSX*CD)).GT.1.0D-6) THEN
!  WRITE(*,*)' D(8) ',D(8),(CC+SSX*CD)
!  D(8)=-D(8)
! END IF
   END IF
   D(9) = -SQ*SD
   D(14)= SP*SD
   D(15)= CD
   IF (.not.LOLDP) THEN
    D(4) = D(2)+D(3)
    D(6) = D(3)-D(2)
   ELSE
    D(4) = -T
    D(6) = -CQCD/CP
   end if
   IF (.not.LOLDQ) then
    D(19)= D(13)+D(7)
    D(31)= D(13)-D(7)
   else
     D(19)= -T
     D(31)= -CPCD/CQ
    END IF
!
    IF (ND.NE.2) THEN
     IF (.not.LOLDP) THEN
      D(10) = D(9)+D(8)
      D(12) = D(9)-D(8)
      D(16) = D(15)+D(14)
      D(18) = D(15)-D(14)
     ELSE
      D(10) = -D(7)
      D(12) = SQ*CD/CP
      D(16) = CPSD
      D(18) = SD/CP
     END IF
     IF (.NOT.LOLDQ) THEN
      D(20) = D(14)+D(8)
      D(32) = D(14)-D(8)
      D(21) = D(15)+D(9)
      D(33) = D(15)-D(9)
     ELSE
      D(20) = -D(2)
      D(21) = -CQSD
      D(32) = SP*CD/CQ
      D(33) = -SD/CQ
     END IF
!
     IF (ND.NE.3) THEN
      IF ((LOLDP.AND.LOLDQ)) THEN
       D(22) = T
       D(24) = CQCD/CP
       D(34) = CPCD/CQ
       D(36) = CD/CC
      ELSE
       IF (LOLDQ) THEN
        D(22) = D(21)+D(20)
        D(24) = D(21)-D(20)
        D(34) = D(33)+D(32)
        D(36) = D(33)-D(32)
```

```
       ELSE
        D(22) = D(16)+D(10)
        D(34) = D(16)-D(10)
        D(24) = D(18)+D(12)
        D(36) = D(18)-D(12)
      END IF
     END IF
    END IF
   END IF
  END IF
 END IF
!
 DO KY=1,6
! DO M=1,6
  DO MT=1,6
!  DD(KY,MT)=D(MT+(KY-1)*6)
! CHANGE 2013-03-18 and back 29.
   DD(MT,KY)=D(MT+(KY-1)*6)
  end do
 end do
!
 S2 = S*S
 ST = S*T
 T2 = T*T
 P2 = (3.0D0*T*T-D1)/D2
 P3 = (3.0d0*S*T+D1)/D2
!
! INITIALIZING ARRAY ELEMENTS.
 CX = D0
 CY = D0
 DC = D0
 DO KS = 1, 40
  CRR(KS+11) = D0
 END DO
 Q(1)=D0
 RM(1)=D0
  V=D0
  U=D0
  G=D0
  R=D0
  P=D0
  SS1=D0
  Q=D0
  CN = D0
  DCN = D0
!
 IF (.NOT.LSAT) THEN
!
! SUMMATION AND DIFFERENTIATION OF THE LEGENDRE SERIES, CF.REF(A),EQ.
! (49) AND (51).
  K1 = N1
  K2 = N1+1
  KX = N1-1
  DO MR = 1, N1
   GI = (D2*KX+D1)*S/(KX+1)
   GJ = -(KX+1)*S*S/(KX+2)
   K2 = K1
   K1 = KX
   KX = KX-1
!  write(*,*)' 10205 KX,M,K1,K2 ',KX,M,K1,K2
   SI = SIGMA(KX+2)
   DO I = 2, ND2
    BX = DC(I)
    DC(I) = CY(I)
    CY(I) = GI*(DC(I)*T+(I-2)*DC(I-1))+GJ*BX+SI
    SI = D0
   END DO
  END DO
  if (lpred.and.nd1.eq.1.and.ltests) write(*,*)' 10371 CY ',CY(2)
```



```
 ELSE
  V=D0
  U=D0
  G=D0
  R=D0
  P=D0
  SS1=D0
  Q=D0
!
! INITIALIZING ARRAY ELEMENTS.
  CX = D0
  CN = D0
  DCN = D0
!
! SUMMATION AND DIFFERENTIATION OF THE LEGENDRE SERIES, CF.REF(A),EQ.
! (49) AND (51).
  K1 = N1
  K2 = N1+1
  KX = N1-1
! if (iir.eq.0.and.(.not.lpred)) write(*,*)' N1,NDX2 ',N1,NDX2,S,T
  DO MY = 1, N1
! DO K=N1-1,1,-1
   GI = (D2*KX+D1)*S/(KX+1)
   GJ = -(KX+1)*S*S/(KX+2)
   KX = KX-1
   DO NDT=1,5
    if (k2.lt.1) write(*,*)' K2,NDT ',k2,ndt
    if ((KX+2).lt.1) write(*,*)' KX+2,NDT ',kx+2,ndt
    SI = SIGMAX(KX+2,NDT)
    DO IP = 2, NDX2(NDT)
     BX = DCN(IP,NDT)
     DCN(IP,NDT) = CN(IP,NDT)
     CN(IP,NDT) = GI*(DCN(IP,NDT)*T+(IP-2)*DCN(IP-1,NDT))+GJ*BX+SI
     SI = D0
!    if (iir.eq.0) write(*,5557)CN(I,NDT),DCN(I-1,NDT),SIGMAX(KX+2,NDT),I,NDT
! 5557     format(' 10356 CN,DCN,SIG,I,NDT ',3d16.5,2i3)
    END DO
   END DO
  END DO
  if (iir.eq.0)write(*,5559)CN
 5559 format(6d12.4)
!
 END IF
! COMPUTATION OF THE FUNCTIONS L=R(1), N=1/RN, M=RM(2), F0=P(2), CF.
! REF.(A), EQ. (31)-(33),(40) AND (77A).
  RL2 = D1-D2*S*T+S*S
  RL =  SQRT(RL2)
  R(1) = RL
  RL1 = D1/RL
  RN = D1/(D1+RL-S*T)
  RL2 = D1/RL2
  RNL = RN*RL1
  RM(2) = D1-RL-S*T
  P(2) = S*LOG(D2*RN)
  RL3 = RL2*RL1
  RL5 = RL3*RL2
  S3 = S*S*S
  R(2) = -S*RL1
  IF (ND.NE.0) THEN
! COMPUTATION OF THE DERIVATIVES WITH RESPECT TO T.
! CF. REF.(A), EQ. (77B),(69A),(57).
   R(3) = -S*S*RL3
   RM(3) = -R(2)-S
   P(3) = S*S*(RNL+RN)
   IF (ND.NE.1) THEN
!
! CF. REF.(A), EQ. (77C),(69B),(58).
    R(4) = -3.0d0*S3*RL5
    RM(4) = -R(3)
```

```
    P(4) = S3*(RL3+(D1+(D2+RL1)*RL1)*RN)*RN
    IF (ND.NE.2) THEN
!
! CF. REF.(A), EQ. (77D),(69C),(59).
     RL4 = RL2*RL2
     RL7 = RL5*RL2
     S4 = S*S*S*S
     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.NE.3) THEN
!
! 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)
     END IF
    END IF
   END IF
  END IF
!
  IF (.NOT.LNY(2)) THEN
! COMPUTATION OF THE FUNCTION F-1 AND ITS DERIVATIVES, CF. REF.(A),
! EQ. (41) AND (61) - (65).
   U(2) = S*(RM(2)+T*P(2))
   IF (ND2.GE.3) THEN
    DO KA = 3, ND2
     U(KA) = S*(RM(KA)+T*P(KA)+(KA-2)*P(KA-1))
    END DO
   END IF
  END IF
!
  IF (.NOT.LNY(1)) THEN
! COMPUTATION OF THE FUNCTION F-2 AND ITS DERIVATIVES, CF. REF.(A) EQ.
! (42), AND (65)- (68).
   DO  KB = 2, ND2
    K=KB
    IF (KB.eq.2) V(2) =&
    S*(RM(KB)*P3+S*((KB-2)*D3*RM(KB-1)/D2+P2*P(KB)+D3*T*P(KB-1)&
    *(KB-2)+S*(D1-T*T)/4.0D0))
    IF (KB.eq.3) V(3) =&
     S*(RM(KB)*P3+S*((KB-2)*D3*RM(KB-1)/D2+P2*P(KB)+D3*T*P(KB-1)&
    *(KB-2)-S*T/D2))
    IF (KB.eq.4) V(4) =&
    S*(RM(KB)*P3+S*((KB-2)*D3*RM(KB-1)/D2+P2*P(KB)+D3*T*P(KB-1)&
    *(KB-2)+D3*P(2)-S/D2))
    IF (KB.eq.5) V(5) =&
    S*(RM(KB)*P3+S*((KB-2)*D3*RM(KB-1)/D2+P2*P(KB)+D3*T*P(KB-1)*(KB-2)+9.0D0*P(3
)))
    IF (KB.eq.6) V(6) =&
    S*(RM(KB)*P3+S*((KB-2)*D3*RM(KB-1)/D2+P2*P(KB)+D3*T*P(KB-1)*(KB-2)+18.0D0*P(
4)))
   END DO
!  if (iir.lt.0)  write(*,5551)V,RM,P,S,T
!5551 format('V ',6d12.5,/,6d12.5,/,6d12.5,/,6d12.5,/,6d12.5)
!
  END IF
!
!if (iir.eq.1.and.(.not.lpred))write(*,*)' 10483 L ', L,LN,IIZ,ND,J1,ND2,RN,RNL,
RL,RL1,RL2,RL4,RL6,S
 IF (.not.LNY(3)) THEN
! COMPUTATION OF THE FUNCTION F1 AND ITS DERIVATIVES, CF. REF.(A) EQ.
! (36), REF.(B), EQ.(101) AND REF.(A), EQ.(70),(71).
  Q(2) = LOG(D1+D2*S/(D1-S+RL))
  IF (ND.NE.0) THEN
   Q(3) = S*S*RNL
```



```
   IF (ND.NE.1) THEN
    Q(4) = S3*((RL1+D1)*RN+RL2)*RNL
    IF (ND.NE.2) THEN
     Q(5) = S4*(D3*RL4+((D2+D3*RL1)*RL2+(D2 +(D4+D2*RL1)*RL1)*RN)*RN)*RNL
     IF (ND.NE.3) THEN
      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
     END IF
    END IF
   END IF
  END IF
!  if (iir.eq.1.and.(.not.lpred)) write(*,5552)Q
! COMPUTATION OF THE FUNCTION F2 AND ITS DERIVATIVES, CF. REF.(A), EQ.
! (3),(72)-(75).
  P(2) = (RL-D1+T*Q(2))/S
  IF (ND.NE.0) THEN
   DO  KC = 3, ND2
    P(KC) = (R(KC-1)+T*Q(KC)+(KC-2)*Q(KC-1))/S
   END DO
  END IF
!  if (iir.eq.1.and.(.not.lpred)) write(*,5552)P
  I1 = IIZ-1
  K1 = 1
  J1 = I1
  IF (I1.LT.2) THEN
   DO ME = 2, ND2
    IF (I1.EQ.0) G(ME) = Q(ME)
    IF (I1.EQ.1) G(ME) = P(ME)
   END DO
  END IF
  IF (L(4)) J1 = JJ-1
! write(*,*)' PREC  10446 JJ,J1,IIY,I1 ',JJ,J1,IIY,I1
  IF (J1.GT.1) THEN
!
! CF. REF.(A), EQ. (38),(76).
  DO KD = 2, J1
   DO  M = 2, ND2
    BX = Q(M)
    Q(M) = P(M)
    P(M) = (R(M-1)+(2*KD-1)*((M-2)*Q(M-1)+T*Q(M))-(KD-1)/S*BX)/(KD*S)
!   P(M) = (R(M-1)+(2*KX-1)*((M-2)*Q(M-1)+T*Q(M))-K1/S*B)/(KX*S)
   END DO
!  if (iir.eq.1.and.lpred) write(*,5553)P,JJ,J1,KX,ND2
   if (.false.) write(*,5553)P,JJ,J1,KD,ND2
 5553 format(' 10380 P,JJ,J1 ',6d11.3,4i3)
   IF (KD.EQ.I1) THEN
    DO  M = 2, ND2
     G(M) = P(M)
    END DO
   END IF
  END DO
!
  END IF
  IF (.NOT.LNY(6)) THEN
! CF. REF.(A), EQ. (34),(55).
  SS1(2) = S*S*(T-S)*RL3
  IF (ND.GT.0) SS1(3) = S*S*(RL3+D3*(T-S)*S*RL5)
!
! CF. REF.(A), EQ. (35).
  END IF
  IF (L(7)) THEN
   CX(2,8)= S*S*((T+S)*RL3+D3*S*(T*T-D1)*RL5)
  end if
!
   END IF
   DO MB=1,6
    CX(MB,1)=CY(MB)
    CX(MB,2)=V(MB)
    CX(MB,3)=U(MB)
```

```
    CX(MB,4)=G(MB)
    CX(MB,5)=P(MB)
    CX(MB,6)=R(MB)
    CX(MB,7)=SS1(MB)
    CXI(MB,1,ICREL)=CY(MB)
    CXI(MB,2,ICREL)=V(MB)
    CXI(MB,3,ICREL)=U(MB)
    CXI(MB,4,ICREL)=G(MB)
    CXI(MB,5,ICREL)=P(MB)
    CXI(MB,6,ICREL)=R(MB)
    CXI(MB,7,ICREL)=SS1(MB)
   END DO
   CXI(7,1,ICREL)=T
   CXI(7,2,ICREL)=S
   if (NRREL.lt.0) write(*,5552)CX,Q,JJ,I1,J1
 5552 format(' C ',6d11.3,/,' V  ',6d11.3,/,' U  ',6d11.3,/,&
   ' G  ',6d11.3,/,' P  ',6d11.3,/,' R  ',6d11.3,/,'SS1',6d11.3,/,&
   ' CX ',6d11.3,/,' Q  ',6d11.3,3I4)
!
   IF (.NOT.LSAT) THEN
! ADDING THE DIFFERENT TERMS, CF. REF.(A), EQ. (22),(47).
! TIPLIED BY RB**2 IN UNITS OF MGAL**2, THE INTEGERS K(2),K(3) OF EQ.
 DO MC = 2, ND2
! CF. REF.(A), EQ. (50),(52).
  CY(MC) = S*CY(MC)
! IF (LTESTS)WRITE(*,*)' CM',CY(M),M
  CRR(MC*8 -4) = CY(MC)
  DO KE = 1, 7
   IF (.NOT.LNY(KE)) THEN
! STORING THE TERMS FOR TRANSFER TO THE CALLING PROGRAM USING THE COMMON
! AREA /CMCOV/.
    CRR(MC*8+KE -4) = A*CX(MC,KE+1)*CCI(KE)
    IF (KE.EQ.5) CRR(MC*8+KE-4) = -CRR(MC*8+KE-4)
    if (KE.EQ.1) CXI(1,1,IIR)=CY(MC)
    CY(MC) = CY(MC)+CRR(MC*8+KE-4)
    CXI(KE+1,1,IIR)=CY(MC)
    if (.false.) WRITE(*,1)A,CX(MC,KE+1),CCI(KE),CY(MC),KE,MC,ND2
 1  FORMAT(' A,CX,CI,C,KE,MC,ND2 ',4E15.7,3I2)
   END IF
  END DO
  CRR(MC+50)=CY(MC)
! CXI(MC,1,IIR)=CY(MC)
 END DO
!     write(*,*)(CY(M),M=2,ND2)
!
 ELSE
!
! FOR THIS SECTION SEE REF.(I) FOR ALL EQUATIONS.
  RQ2=RQ*RQ
  RPQ=RQ*RPP
  DO NDT=1,5
   DO MC = 2, NDX2(NDT)
    CN(MC,NDT)=CN(MC,NDT)*S
!       if (M.EQ.3.AND.NDT.EQ.2) WRITE(*,*)' CM',CN(M,NDT),M,NDT,S,LSAT
    DO KX = 1, 7
     IF (.NOT.LNZ(KX,NDT)) THEN
      FAK5=D1
      IF (KX.EQ.5) FAK5=-D1
!       if (M.EQ.3.AND.NDT.EQ.2) WRITE(*,*)' CM1',CN(M,NDT)
      CN(MC,NDT)=CN(MC,NDT)+A*CX(MC,KX+1)*CIX(KX,NDT)*FAK5
!       if (M.EQ.3.AND.NDT.EQ.2) WRITE(*,*)' CM2',CN(M,NDT)
      if (.false.) WRITE(*,1)A,CX(MC,KX+1),CIX(KX,NDT),CN(MC,NDT),KX,NDT
     END IF
    END DO
    CN(MC-1,NDT)=CN(MC,NDT)*(-1)**(NDT+1)
   END DO
  END DO
  if (iir.eq.0) WRITE(*,*)' LDEFVP,LDEFVQ,NDP,NDQ,KP,KQ=',LDEFVP,LDEFVQ,&
  NDP,NDQ,KP,KQ
```



```
!
! WE NOW CALCULATE THE CROSS-COVARIANCES BETWEEN ALL QUANTI-
! TIES OF THE GIVEN ORDERS.
! NCASE=NDP+1+NDQ*3
! GO TO (801,802,803,804,805,806,807,808,809),NCASE
  IF (NCASE.EQ.1) THEN
! NO DERIVATIVES IN P OR Q.  CHANGED 2005-02-18, SO THAT
! HEIGHT ANOMALIES CAN BE USED.
   COVX(1,1,1,1)=CN(1,1)/(CCR10*CCR11)
!  COVX(1,1,1,1)=CN(1,1)/(CRR(10)*CRR(11))
  END IF
! 1 DERIVATIVE IN P, NONE IN Q. REF(I), EQ. (16) AND (17).
  IF (NCASE.EQ.2) THEN
   COVX(1,1,1,1)=D(3)*CN(2,1)/RPP
   COVX(2,1,1,1)=D(2)*CN(2,1)/RPP
   COVX(3,1,1,1)=CN(1,2)/RPP
! GRAVITY ANOMALY WITH GEOID. ADDED 1992.09.07.
   IF (LDGP) COVX(3,1,1,1)=(-CN(1,2)-D2*CN(1,1))/RPP
  END IF
! 2 DERIVATIVES IN P, NONE IN Q. REF(I), EQ. (24)-(28).
  IF (NCASE.EQ.3) THEN
   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
  END IF
! NO DERIVATIVE IN P, 1 IN Q. REF(I), EQ. (18), (19).
  IF (NCASE.EQ.4) THEN
   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
! GRAVITY ANOMALY WITH GEOID. ADDED 1992.09.07.
 IF (LDGQ) COVX(3,1,1,1)=(-CN(1,2)-D2*CN(1,1))/RQ
  END IF
! 1 DERIVATIVE IN BOTH P AND Q. REF(I), EQ. (20)-(23).
  IF (NCASE.EQ.5) THEN
   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
! 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
  END IF
! 2 DERIVATIVES IN P, ONE IN Q. REF(I), EQ. (29)-(33).
  IF (NCASE.EQ.6) THEN
   RP2Q=RP2*RQ
 5558 format(6F10.6)
  CNX=CN(2,2)-T*CN(3,1)+D(3)*D(3)*CN(4,1)-CN(2,1)
!if (.false.) write(*,5556)CN(2,2),T,CN(3,1),D(3),CN(4,1),CN(2,1)
```

```
!5556 format(3d16.5/3d16.5/3d16.5)
 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)        +D(3)*D(3)*CN(3,2)-T*CN(2,2))/RP2Q
! CHANGE 2013-03-15.
!COVX(1,1,3,1)=(CN(1,3)+CN(1,2)+D(3)*D(3)*CN(3,2)-T*CN(2,2))/RP2Q
!     COVX(2,1,1,1)=(D(2)*D(3)*D(13)*CN(4,1)+D(17)*CN(2,1)
!    *     +(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
! 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)))/RP2
Q
 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)))/RP2
Q
 COVX(3,1,3,1)=DD(3,1)*CN(2,3)/RP2Q
!     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
! check 2010-11-30.
 COVX(2,2,3,1)=(CN(1,3)+D(2)*D(2)*CN(3,2)-T*CN(2,2))/RP2Q
!     write(*,*)' 16733 ',COVX(2,2,1,3)
 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
! 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))&
! CHANGE 2013-03-15.
 COVX(1,1,3,1)=-(CN(1,3)+D2*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
!     COVX(3,1,3,1)=-DD(1,3)*(CN(2,3)+D2*CN(2,2))/RP2Q
 IF (LTESTS) WRITE(*,*)' COVX(3,1,3,1)= ',COVX(3,1,3,1)
 COVX(1,2,3,1)=COVX(2,1,3,1)
! check 2010-11-30.
 COVX(2,2,3,1)=-(CN(1,3)+D3*CN(1,2)+D(2)*D(2)*(CN(3,2)+D2*CN(3,1)) &
!     COVX(2,2,3,1)=-(D3*CN(1,2)+D(2)*D(2)*(CN(3,2)+D2*CN(3,1))
 -T*(CN(2,2)+D2*CN(2,1)))/RP2Q
!     write(*,*)' 16760 ',COVX(2,2,3,1)*1.0D14
 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
!IF (.TRUE.) WRITE(*,5555) LDGQ,COVX(1,1,1,1),COVX(2,2,1,1),COVX(3,3,1,1),rp2q,C
NX,DD(2,1),DD(3,1),D2
 5555 FORMAT(l2,4d16.5/4d16.5)
  END IF
!GOTO 810
! NO DERIVATIVE IN P, TWO IN Q. REF(I), EQ. (24)-(28).
```



```
  IF (NCASE.EQ.7) THEN
  COVX(1,1,1,1)=(CN(1,2)+D(13)*D(13)*CN(3,1)-T*CN(2,1))/RQ2
! 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
! ERROR 2002-11-26.
!     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
  END IF
! ONE DERIVATIVE IN P, TWO IN Q.  REF(I), EQ. (29)-(33).
  IF (NCASE.EQ.8) THEN
! 808 RPQ2=RP*RQ2
   RPQ2=RPP*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
! check 2010-11-30.
!     COVX(3,1,1,1)=(CN(1,3)+CN(1,2)+D(13)*D(13)*CN(3,2)
 COVX(3,1,1,1)=(CN(1,3)+D(13)*D(13)*CN(3,2)-T*CN(2,2))/RPQ2
! 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))/RP
Q2
!     COVX(2,1,2,1)=(DD(2,2)*DD(1,3)*CN(3,1)+DD(1,2)*(DD(3,2)*CN(3,1)
! 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)))/RPQ
2
 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)))/RPQ
2
 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
! check 2010-11-30.
!     COVX(3,1,2,2)=(CN(1,2)+D(7)**2*CN(3,2)
 COVX(3,1,2,2)=(CN(1,3)+D(7)**2*CN(3,2)-T*CN(2,2))/RPQ2
!     write(*,*)' 16813 ',COVX(3,1,2,2)*1.0D14,D(7)**2*
!    *CN(3,2)/RPQ2*1.0D14
 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
! 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
! GRAVITY ANOMALY ADDED 1992.09.30.
 IF (LDGP) THEN
! check 2010-11-30.
!      COVX(3,1,1,1)=-(CN(1,3)+D3*CN(1,2)+D(13)*D(13)*(CN(3,2)
  COVX(3,1,1,1)=-(CN(1,3)+D2*CN(1,2)+D(13)*D(13)*(CN(3,2)+D2*CN(3,1))-T*(CN(2,2)
+D2*CN(2,1)))/RPQ2
```

```
! 2000-04-03
  COVX(3,1,2,1)=-DD(1,2)*DD(1,3)*(CN(3,2)+D2*CN(3,1))/RPQ2
!       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
  IF (LTESTS) WRITE(*,*)' COVX(3,1,3,1) ', COVX(3,1,3,1)
  COVX(3,1,1,2)=COVX(3,1,2,1)
! check 2010-11-30.
! COVX(3,1,2,2)=-(CN(1,3)+D2*CN(1,2)+D(7)**2*(CN(3,2) &
! CHANGE 2013-05-03.
  COVX(3,1,2,2)=-(CN(1,3)+D3*CN(1,2)+D(7)**2*(CN(3,2) &
!       COVX(3,1,2,2)=-(D3*CN(1,2)+D(7)**2*(CN(3,2)
    +D2*CN(3,1))-T*(CN(2,2)+D2*CN(2,1)))/RPQ2
!       write(*,*)' 16844 ',COVX(3,1,2,2)*1.0D14
  COVX(3,1,3,2)=-D(7)*(CN(2,3)+D2*CN(2,2))/RPQ2
!       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
!IF (.TRUE.) WRITE(*,5555) LDGP,COVX(1,1,1,1),COVX(2,2,1,1),COVX(3,3,1,1),rpq2,C
NX,DD(2,1),DD(3,1),D2
  END IF
!GO TO 810
! TWO DERIVATIVES IN BOTH P AND Q. REF(I), EQ. (34)-(46).
  IF (NCASE.EQ.9) THEN
! 809 R2PQ=RPQ**2
   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)       &
! CHANGE 2013-03-16
!! 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*SD**2)+T*T)                     &
   +T*CN(2,1)+CN(3,1)*(D2*(CD**2-D3132)+T*T)                        &
   -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)+T*T)+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
! SUSPECTED ERROR 2002-10-07
!    *  +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
 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
!
 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
! 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
!     COVX(3,1,2,1)=(D(3)*D(13)*D(7)*(CN(3,2)-CN(3,1))
```



```
!    *  +(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
!
 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
!     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
!
 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)
!
 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)                                  &
   +T*T)+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)+(T*T-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
!
 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
!
 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)
!
 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)
!
 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
  END IF
  END IF
! 810 END IF
!
  IF (.NOT.LSAT) THEN
! write(*,6656)ND1,LOLDP,LOLDQ
! 6656 format(' 11008 ND1 ',i3,2l5)
! INTEGERS SPECIFYING THE KINDS OF DIFFERENTIATION WITH RESPECT TO THE
! LATITUDES AND/OR THE LONGITUDES, CF. REF.(A), SECTION 3.
 IQ = KV10
 J = KV12
 K = KV11
 M = KV13
 J1 = KV14
 M1 = KV15
 IF (.false.) WRITE(*,*) ' 11045,iq,j,k,m,j1,m1,nd1 ',i,j,k,k,j1,m1,nd1,LOLDP,LO
LDQ
 IF (LOLDP.OR.LOLDQ) THEN
!IF (.NOT.(LOLDP.OR.LOLDQ)) GO TO 110
!
 IJ = IQ+J
 IF (I.GT.3) IJ = 5
 KM = K+M
 IF (K.GT.3) KM = 5
!
! COMPUTATION OF THE DERIVATIVES OF ORDER ND WITH RESPECT TO THE LATI-
! TUDES AND THE LONGITUDES, CF. REF.(A), EQ. (43) - (46).
!GOTO (80,81,82,83,84),ND1
  IF (ND1.EQ.1) THEN
```



```
!  80 COV = CY(2)
   COV = CY(2)
   write(*,*)' 10983 CY(2),CI12,IIR ',CY(2),CI12,IIR
  END IF
!GOTO 85
  IF (ND1.EQ.2) THEN
   COV = -CY(3)*D(IQ+6*(K-1))
  END IF
  IF (ND1.EQ.3) THEN
   COV = D(IQ)*D(J1)*D(6*(K-1)+1)*D(6*(M1-1)+1)*CY(4)+D(IJ+6*(KM-1))*CY(3)
!     write(*,*)' cov82 ',cov
  END IF
  IF (ND1.EQ.4) THEN
   COV = (-D(IJ+6*(KM-1))*CY(3)+(D(IJ)*D(6*(KM-1)+1)+D(IQ+6*(K-1))        &
    *D(J1+6*(M1-1))+D(IQ+6*(M1-1))*D(J1+6*(K-1)))*CY(4)
&
    +D(IQ)*D(J1)*D(6*(K-1)+1)*D(6*(M1-1)+1)*CY(5))
  END IF
  IF (ND1.EQ.5) THEN
   COV  = D(IJ+6*(KM-1))*CY(3)+(D(IJ+6*(K-1))*D(6*(M-1)+1)               &
   +D(IQ+6*(KM-1))*D(J)+D(J+6*(KM-1))*D(IQ)+D(IJ+6*(M-1))                  &
   *D((K-1)*6+1)+D(IJ)*D(6*(KM-1)+1)+D(IQ+6*(K-1))*D(J+6*(M-1))           &
   +D(IQ+6*(M-1))*D(J+6*(K-1)))*CY(4)+(D(IJ)*D(6*(K-1)+1)*D(6*(M-1)+1)    &
   +D(IQ+6*(K-1))*D(J)*D(6*(M-1)+1)+D(IQ+6*(M-1))*D(J)*D(6*(K-1)+1)       &
   +D(J+6*(K-1))*D(IQ)*D(6*(M-1)+1)+D(J+6*(M-1))*D(IQ)*D(6*(K-1)+1)       &
   +D(6*(KM-1)+1)*D(IQ)*D(J))*CY(5)+D(IQ)*D(J)*D(6*(K-1)+1)*D(6*(M-1)+1)*CY(6)
  END IF
!
! GIVING THE COVARIANCE THE PROPER UNITS.
  COV = COV*CI12
! COV = COV*CTI(12)
!WRITE(*,*) ' 11014 ',cov
 CCV(1,1)=COV
!
 ELSE
!GO TO 199
! 110 CF=CTI(12)
  CF=CI12
! CF=CTI(12)
  IF (KP.EQ.13) CF=CF/D2
  IF (KQ.EQ.13) CF=CF/D2
  DO IX = 2, ND2
   CZ(IX-1) = CY(IX)*CF
  END DO
  CCV(1,2) = D0
  CCV(2,1) = D0
  CCV(2,2) = D0
!GO TO (112, 113, 114, 115, 115), ND1
  IF (ND1.EQ.1) THEN
   CCV(1,1) = CZ(1)
#ifdef _OPENMP
   if (iir.lt.0) write(*,*)' 11104 CZ1,IIR ',CZ(1),IIR,omp_get_thread_num()
#else
   if (iir.lt.0) write(*,*)' 11104 CZ1,IIR ',CZ(1),IIR
#endif
   CXI(7,8,IIR)=CZ(1)
!  CT(IIR)=CZ(1)
!  ================================================================
  KZ=1
  END IF
!
  IF (ND1.EQ.2) THEN
   IF (I.NE.1) THEN
!  IF (I.EQ.1) GO TO 116
    CCV(1,1) = CZ(2)*D(3)
    CCV(2,1) = CZ(2)*D(2)
!  ================================================================
    KZ=2
   ELSE
```

```
    CCV(1,1) = CZ(2)*D(13)
    CCV(1,2) = CZ(2)*D(7)
!  ================================================================
    KZ=3
   END IF
  END IF
!
  IF (ND1.EQ.3) THEN
   IF (I.LE.1) THEN
!  IF (I.GT.1) GO TO 117
    CCV(1,2) = CZ(3)*D(19)*D(31)
    CCV(1,1) = CZ(3)*D(7)*D(13)*D2
!  ================================================================
    KZ=4
   ELSE
    IF (K.LE.1) THEN
     CCV(2,1) = CZ(3)*D(4)*D(6)
     CCV(1,1) = CZ(3)*D(2)*D(3)*D2
!  ================================================================
     KZ=5
    ELSE
     CCV(1,1) = CZ(2)*D(15)+CZ(3)*D(13)*D(3)
     CCV(2,2) = CZ(2)*D(8) +CZ(3)*D(2)*D(7)
     CCV(1,2) = CZ(2)*D(9) +CZ(3)*D(3)*D(7)
     CCV(2,1) = CZ(2)*D(14)+CZ(3)*D(13)*D(2)
!  ================================================================
     KZ=6
! FIRST ORDER HORIZONTAL DERIVATIVES IN BOTH P AND Q.
    END IF
   END IF
  END IF
! 115 CONTINUE
  IF (ND1.GT.3) THEN
!
   IIX=2
   DO IX = 1, 2
!DO 119 IX = 1, 2
    IIY=2
! DO 120 JX = 1, 2
    DO JX = 1, 2
     IF (ND.NE.4) THEN
!  IF (ND.EQ.4) GO TO 121
! SECOND ORDER HORIZONTAL DERIVATIVE IN P OR Q.
!     write(*,*)' 17072 2. order deriv ',KP,KQ
      IX1=IX
      JX1=JX
      IF (KP.LT. 12) THEN
!     IF (KVI(6) .LT. 12) THEN
       CF = JX
       JX1=IIY
       I = J2(IX)
       J1 = 1
       K = I4(JX)
       M1 = I3(JX)
!GO TO 123
      ELSE
       CF = IX
! 122 CF = IX
       IX1=IIX
       I = I4(IX)
       J1 = I3(IX)
       K = J2(JX)
       M1 = 1
      END IF
! 123 K6 = 6*(K-1)
      K6 = 6*(K-1)
      M6 = 6*(M1-1)
      CCV(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
```



```
!  ================================================================
      KZ=7
!GO TO 120
     ELSE
      I = I4(IX)
! 121 I = I4(IX)
      J = I3(IX)
      K = I4(JX)
      M = I3(JX)
      K6 = 6*(K-1)
      M6 = 6*(M-1)
      CCV(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
! ================================================================
      KZ=8
     END IF
     IIY=1
    END DO
! 120 IIY=1
! 119 IIX=1
   IIX=1
   END DO
  END IF
 COV = CCV(KV24,KV25)
! 198 COV = CCV(KVI(24),KVI(25))
! ================================================================
 IF (iir.lt.0.and.(.not.LPRED))WRITE(6,7788) KZ,I,J,K,M,CCV(1,1),CCV(1,2),CCV(2,
1),&
 CCV(2,2)
 7788 FORMAT(/'  KZ,   I,   J,   K,   M,   CCV(1,1),   CCV(1,2), ',&
 ' CCV(2,1)   CCV(2,2)'/1X,5I4,4F10.4)
! 199 CONTINUE
  END IF
 ELSE
  COV=COVX(KKP1,KKP2,KKQ1,KKQ2)
! IF (KP.NE.KPP.or.KQ.ne.KQQ.or.KKP1.ne.KSAT(KP,1).or.KKQ1.ne.KSAT(KQ,1)) then
!  write(*,*)' KP,KPP,KQ,KQQ inconsistency ',KP,KpP,KQ,KQQ
!  stop
! end if
! COV=COVX(KSAT(KP,1),KSAT(KP,2),KSAT(KQ,1),KSAT(KQ,2))
! if (abs(COV).gt.1.0d8) then
! PRINT *,'   I am thread number ',omp_get_thread_num()
! write(*,5559)CN
  if (lpred.and.iir.lt.0) write(*,5560)COV,ksat(kp,1),ksat(kp,2)
 5560 format(' 11035 COV,',d12.5,4i3)
! end if
  IF (KP.EQ.15.AND.KQ.NE.15) COV=COV-COVX(2,2,KSAT(KQ,1),KSAT(KQ,2))
! 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
   IF (KP.EQ.1.AND.(.NOT.LSATPP)) THEN
!        WRITE(*,*)' C11(KP),CRR(11) ',C11(KP),CRR(10)
    C11P=C11X(KP)/(CCR10**K19(KP))
!   C11P=C11X(KP)/(CRR(10)**K19(KP))
   ELSE
! CHANGE 2003-04-01 AND 2011-07-25.
!     C11P=C11(KP)/(CCR(10)**K19(KP))
   C11P=C11X(KP)
   END IF
  END IF
  IF (KQ.EQ.6.OR.KQ.EQ.7) THEN
   C11Q=1.0D5
  ELSE
   IF (KQ.EQ.1.AND.(.NOT.LSATQ)) THEN
!        WRITE(*,*)' C11(KQ),CCR(11) ',C11(KQ),CCR(11)
    C11Q=C11X(KQ)/(CCR11**K19(KQ))
```

```
   ELSE
!        C11Q=C11(KQ)/(CRR(11)**K19(KQ))
    C11Q=C11X(KQ)
   END IF
  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
! 2000-04-04.
! IF (abs(cov).gt.1.0d4)WRITE(*,*)' KSAT ',KSAT(KP,1),KSAT(KP,2),KSAT(KQ,1),&
! KSAT(KQ,2),' COV ',COV,' CFA ',CFA
 END IF
!=====================================================
 IF (LPRED.and.NRREL.LT.0) write(*,5508)NRREL,COV,T,CFA,KZ,ND1,LSAT
 5508 format(' 11268 NRREL,COV,T,CFA,KZ,ND1 ',i3,3d15.6,2i3,l2)
    CCV(1,1)=COV
!
! IF WE USE A SATELLITE ORIENTED FRAME, THE COVARIANCE MATRIX MUST
! BE ROTATED TO THIS FRAME. CHANGE AUG 89 BY CCT. CORRECTED MAR 95.
    IF (LSATP.AND.LDEFVP) CALL SROT(CCV,SAZP,CAZP,1,LF)
    IF (LSATQ.AND.LDEFVQ) CALL SROT(CCV,SAZQ,CAZQ,1,LT)
    IF (LSATPQ) THEN
     IF (LROT) THEN
! CHANGE 2003-03-11 AND 2013-05-03..
      IF (ISAT(JR-2).GE.1.AND.(.NOT.LSATQ)) THEN
!     IF (ISAT(JR-2).NE.0) THEN

       COSB = SR11A(ICREL)
       SINB = SR12A(ICREL)
       COST = SR13A(ICREL)
       SINT = SR22A(ICREL)
       CAZQ = COSAZA(ICREL)
       SAZQ = SINAZA(ICREL)
       if (icrel.lt.0) write(*,156)icrel,cosb,sinb,cost,sint,cazq,sazq
  156  format(' 11676',i5,6f10.7)
       SROTQ(1,1) =  SAZQ*COSB
       SROTQ(1,2) =  CAZQ*COST+SAZQ*SINB*SINT
       SROTQ(1,3) = -CAZQ*SINT+COST*SAZQ*SINB
       SROTQ(2,1) = -CAZQ*COSB
       SROTQ(2,2) =  SAZQ*COST-SINT*CAZQ*SINB
       SROTQ(2,3) = -SAZQ*SINT-COST*CAZQ*SINB
       SROTQ(3,1) = -SINB
       SROTQ(3,2) =  COSB*SINT
       sROTQ(3,3) =  COST*COSB
      ELSE
! CORREC tion 2003-03-05.
       SROTQ(1,1) = D1
       SROTQ(2,2) = D1
       SROTQ(3,3) = D1
       SROTQ(1,2) = D0
       SROTQ(1,3) = D0
       SROTQ(2,1) = D0
       SROTQ(2,3) = D0
       SROTQ(3,1) = D0
       SROTQ(3,2) = D0
      END IF
!  IF (LTCOV) THEN
!     IF (mod(iir,10000).eq.-10) theN
      if (icrel.lt.0) then
!      WRITE(*,*)'COVX ',COVX
     WRITE(*,150)SATROT,SROTQ,((((COVX(IAA,IBB,ICC,IDD),IAA=1,3),IBB=1,3),ICC=1,
3),IDD=1,3)
      END IF
!
!     IF (.NOT.(KPP.EQ.3.OR.KQQ.EQ.3)) THEN
      if (IIR.lt.0.and.NCASE.EQ.9.and.lpred) then
       do ibb=1,3
        DO ICC=1,3
```



```
         if (abs(covx(1,1,ibb,icc)+covx(2,2,ibb,icc)+covx(3,3,ibb,icc)).gt.&
         abs(covx(3,3,3,3))*1.0d-5) write(*,157)1,IBB,ICC,&
         covx(1,1,ibb,icc),covx(2,2,ibb,icc),covx(3,3,ibb,icc),&
         covx(1,1,ibb,icc)+covx(2,2,iBB,ICC)+COVX(3,3,IBB,ICC)
         if (abs(covx(ibb,icc,1,1)+covx(ibb,icc,2,2)+covx(ibb,icc,3,3)).gt.&
         abs(covx(3,3,3,3))*1.0d-5) write(*,157)2,ibb,icc,&
         covx(ibb,icc,1,1),covx(ibb,icc,2,2),covx(ibb,icc,3,3),&
         covx(ibb,icc,1,1)+covx(ibb,icc,2,2)+covx(ibb,icc,3,3)
 157     format(' warn ',3i2,4d14.5)
        end do
       end do
      end if

       if (IIR.lt.0.and.NCASE.EQ.9.and.lpred) then
        WRITE(*,150)SATROT,SROTQ,&
        ((((COVX(IAA,IBB,ICC,IDD),IAA=1,3),IBB=1,3),ICC=1,3),IDD=1,3)
       end if
       CALL COVROT(COVX,SATROT,SROTQ)

      if (IIR.lt.0.and.NCASE.EQ.9.and.lpred) then
       WRITE(*,150)SATROT,SROTQ,&
        ((((COVX(IAA,IBB,ICC,IDD),IAA=1,3),IBB=1,3),ICC=1,3),IDD=1,3)
       end if
!1150 FORMAT(' SATROT,SROTQ,COVX ',/,9F8.4,/,9F8.4/21(5D14.5/))
!     END IF
!
      if (IIR.lt.0.and.NCASE.EQ.9.and.lpred) then
       do ibb=1,3
        do icc=1,3
         if (abs(covx(1,1,ibb,icc)+covx(2,2,ibb,icc)+covx(3,3,ibb,icc)).gt.&
         abs(covx(3,3,3,3))*1.0d-5) writE(*,157)3,IBB,ICC,&
         covx(1,1,ibb,icc),covx(2,2,ibb,icc),covx(3,3,ibb,icc),&
         covx(1,1,ibb,icc)+covx(2,2,iBB,ICC)+COVX(3,3,IBB,ICC)
         if (abs(covx(ibb,icc,1,1)+covx(ibb,icc,2,2)+covx(ibb,icc,3,3)).gt.&
         abs(covx(3,3,3,3))*1.0d-5) write(*,157)4,ibb,icc,&
         covx(ibb,icc,1,1),covx(ibb,icc,2,2),covx(ibb,icc,3,3),&
         covx(ibb,icc,1,1)+covx(ibb,icc,2,2)+covx(ibb,icc,3,3)
        end do
       end do
      end if
!     if (iir.lt.3.and.lsphar) write(*,*)' 9900 KPP,KQQ ',KPP,KQQ
      IF (KPP.NE.15.AND.KQQ.NE.15) THEN
       COV = COVX(KSAT(KPP,1),KSAT(KPP,2),KSAT(KQQ,1),KSAT(KQQ,2))*CFA
!      if  (IIR.EQ.1.and.lpred) then
       if (mod(IIR,10000).EQ.-10.and.lpred) then
        write(*,*)' ks1212 CFA,COV ',KSAT(KPP,1),KSAT(KPP,2),KSAT(KQQ,1),&
       KSAT(KQQ,2),CFA,COV
!       write(*,7676)COVX(3,3,3,3),t,cfa
 7676   format('  COVX ',4d14.5)
       end if
      ELSE
       IF (KPP.EQ.15.AND.KQQ.NE.15) THEN
! DDT/DXX-DDT/DYY IN P.
        COV = (COVX(KSAT(14,1),KSAT(14,2),KSAT(KQQ,1),&
         KSAT(KQQ,2))-COVX(KSAT(12,1),KSAT(12,2),KSAT(KQQ,1),KSAT(KQQ,2)))*CFA
       ELSE
        IF (KPP.NE.15.AND.KQQ.EQ.15) THEN
! DDT/DXX-DDT/DYY IN Q.
         COV = ( COVX(KSAT(KPP,1),KSAT(KPP,2),KSAT(14,1),KSAT(14,2))   &
            -COVX(KSAT(KPP,1),KSAT(KPP,2),KSAT(12,1),KSAT(12,2)))*CFA
        ELSE
! DDT/DXX-DDT/DYY IN BOTH P AND Q.
         COV = ( 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)))*CFA
! OBS IN P AND Q ARE BOTH DDT/DXX-DDT/DYY.
         END IF
        END IF
```

```
       END IF
      END IF
     ELSE
      COV=CCV(1,1)
     END IF
   IF (LCST) THEN
!   C(NI) = COV
!   C(NISTART+IR-1) = COV
    CT(NRREL) = COV
!   CT(IIR) = COV
!   if (nd1.ne.1) CT(IIR) = COV
! PRINT *,'   I am thread number ',omp_get_thread_num(),COV,IIR
    if (IIR.lt.0.and.(lpred).and.ND1.EQ.1) write(*,*)&
    ' IIR,NISTART+IR-1,COV ',IIR,NISTART+IIR-1,COV
    LLCOER = ((.NOT.LFOUR.AND.(ITRACE(IC).EQ.ITRACE(IIR).AND.ITRACE(IC).LT.0))&
    .OR.(LFOUR.AND.LCOERR)).AND.(.NOT.LPRED)
    IF (IC.GT.MAXX.OR.IIR.GT.MAXX) THEN
     LLCOER=LF
    ELSE
!       IF ((.NOT.LPRED).AND.LLCOER) THEN
     IF ((.NOT.LPRED).AND.LLCOER) THEN
      IF (LFOUR) THEN
! PSI MUST BE IN SECONDS.
       ERRCOV=D0
       PSI=ABS(CTIME(IC)-CTIME(IIR))
       IF (PSI.LE.NFOUR) THEN
        PSI=PSI/NFOUR
        DO IBB=0,NFOUR
         ERRCOV=ERRCOV+FOUCOF(1,IBB)*COS(PSI*IBB*PI/NFOUR)
        END DO
!       write(*,*)' ERRCOV,PSI= ',ERRCOV,PSI
       END IF
! NOT YET FULLY IMPLEMENTED 2012-10-10.
      ELSE
!  2005-04-04 IMPLEMENTED THAT TIME DIFFERENCES AND NOT SPHERICAL
! DISTANCE  MAY BE AN ARGUMENT IN COZERO.
       IF (LCTIME) THEN
        PSI=ABS(CTIME(IC)-CTIME(IIR))
       ELSE
        PSI=ACOS(T)
       END IF
!      COZERO = D1
       PSI2=PSI**2
       PSI3=PSI**3
       R2=RDD*RDD
       R4=R2*R2
       IF (PSI.LT.D2*RDD) 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*RDD))
       ELSE
        COZERO=D0
       end if
       ERRCOV = SCFACT*COZERO
       if (ltcov) write(*,8041)errcov,psi,rdd,scfact
 8041  format(' errcov,psi,rdd,scfact ',4d16.6)
!       ERRCOV = SCFACT*COZERO(PSI,RDD,1)
! ACTIVATED 2014-02-14.
       NERCOV=NERCOV+1
      END IF
    IF (LTCOV) WRITE(*,7049)ERRCOV,IC,IIR,ITRACE(IC),CT(NRREL),&
!   IF (LT) WRITE(*,7049)ERRCOV,IC,IIR,ITRACE(IC),CT(NRREL),&
        CTIME(IC),CTIME(IIR)
 7049    FORMAT(' ERROR-CORR.',F14.7,' FOR OBS ',2I7,&
    ' ADDED TO COV. ON ',/,' TRACK ',I11,' COV= ',F14.7, ' TIMES ',2f15.1)
! WE ADD NOISE TO THE DIAGONAL ELEMENT, CHANGE 1992.07.19. AND 1997-07-15.
      CT(NRREL) = CT(NRREL)+ERRCOV
     END IF
    END IF
```



```
    IF ((IC.EQ.IIR).AND.(.NOT.LPRED).AND.(.NOT.LLCOER)) THEN
     CT(NRREL)=COV+WOBSQ(NRREL)**2
!   if (IIR.eq.IC.and.(.not.lpred)) write(*,*)&
!   ' IIR,NISTART+IR-1,COV,WO ',IIR,NISTART+IIR-1,COV,WOBSQ(NRREL)
    END IF
   END IF
!IF (LPRED.AND.NRREL.LT.2) WRITE(*,7111)PREDP,CCV,COV,0.0D0,IIR,LSATPQ
   IF (LPRED) THEN
    PREDP = PREDP+COV*BT(NRREL)
!   PREDP = PREDP+COV*BT(IIR)
     IF (NPRED.GT.0) THEN
!    C(NI)= COV
!    C(NISTART+IIR-1) = COV
     CT(NRREL) = COV
!    if (mod(iir,10000).eq.1)  write(*,*)' 11585 NRREL,c ',NRREL,COV
     END IF
     if (iir.lt.0) write(*,*)' 9921 NISTART+IIR-1,c ',NISTART+IIR-1,COV

   END IF
!
!7018 NI = NI+1
!   NR1 = NR
!   NR = NR+1
   END DO
!$OMP END DO
!$OMP BARRIER
!$OMP END PARALLEL
   IR=MIN(IC,ANDEX(JR-2)-1)
!  if (lpred) write(*,*)' 11860 JRNEXT,IR,IKP,IKQ predp0 ',JRNEXT,IR,IKP,IKQ,pre
dp0
!  write(*,1441) (CT(IIR),IIR=JRNEXT,IR)
 1441 format(6d11.4)
   DO IIR=JRNEXT,IR
    NR=IIR
    NR1=NR-1
    NRREL=MOD(NR1,MAXX)+1
    ICREL=NRREL
    C(NISTART+NRREL-1)=CT(NRREL)
    IF (LPRED.or.(.true.)) THEN
    if (LTCOV) write(*,1438) NRREL,NISTART+NRREL-1,CT(NRREL),CXI(7,1,NRREL),&
    CXI(7,2,NRREL),cxi(7,3,NRREL),CXI(7,4,NRREL),CXI(7,5,NRREL),CXI(7,6,NRREL),&
    CXI(7,7,NRREL),CXI(7,8,NRREL),CXI(8,1,NRREL),CXI(8,2,NRREL),CXI(8,3,NRREL),C
XI(8,4,NRREL),&
   CXI(8,5,NRREL),CXI(8,6,NRREL),CXI(8,7,NRREL),CXI(8,8,NRREL),&
    ((CXI(M,K,ICREL),M=1,6),K=1,7)
 1438 format(' 11438 NRREL,NISTART,NRREL-1,CT(NRREL),CY ',2i5,2D15.7,/,&
     7D15.7,/,8F9.6,&
     ' C ',7d11.3,/,' V  ',7d11.3,/,' U  ',7d11.3,/,&
     ' G  ',7d11.3,/,' P  ',7d11.3,/,' R  ',7d11.3,/,'SS1',7d11.3)
     PREDP0=PREDP0+CT(ICREL)*BT(ICREL)
!    IF ((IIR.eq.1.OR.IIR.EQ.IR).and.lpred) WRITE(*,*)' 11622 IIR,ICREL,PREDP0,C
T,BT',&
!    IIR,ICREL,PREDP0,CT(ICREL),BT(ICREL)
!    PREDP0=PREDP0+CT(NRREL)*BT(NRREL)
    END IF
   END DO
   PREDP=PREDP0
     if (lf) write(*,7222)predp0
 7222 format(' 11886 predp0 ',d15.4)
   DIFPRE=PREDP-PREDP0
   IF (LPRED.AND.(ABS(DIFPRE).GT.1.0D-10).AND.IDSET.EQ.1) THEN
! CORRECTION 2012-11-26.
    IF (LTESTS.AND.ABS(DIFPRE/((PREDP+PREDP0)/2)).GT.1.0D-3)&
    WRITE(*,8861)PREDP,PREDP0,DIFPRE
 8861 format(' PREDP-PREDP0 ',3D16.9)
    PREDP=PREDP0
   END IF
!  WRITE(*,*)' IR,MIN(IC,ANDEX(Jr-2)-1 ',IR,MIN(IC,ANDEX(JR-2)-1)
   NI=NISTART+IR
```

```
!  if ((NISTART+IR).NE.NI.OR.NR1.NE.IR.OR.NR.NE.(IR+1)) THEN
!  write(*,*)' NISTART,NI,NR1,IC,IR,NR ',NISTART,NI,NR1,IC,IR,NR
!  end if
  ELSE
!==================================================
!  write(*,*)' 11212 JRNEXT,MIN(IC+ISO,ANDEX(JR-2)-1),IC,ISO,ANDEX(JR-2),JR ',&
!  JRNEXT,MIN(IC+ISO,ANDEX(JR-2)-1),IC,ISO,ANDEX(JR-2),JR
   DO IIR = JRNEXT,MIN(IC+ISO,ANDEX(JR-2)-1)
   IR = IIR-ISO
!  NR = IR
! change 2012-10-03.
   NR1=NR-1
   NRREL=MOD(NR1,MAXO)+1
!  IF (LCST) write(*,*)' 11248 NR,NRREL ',NR,NRREL
   LTESTS = LCZERO .AND.IR.LT.3
   ICREL=ICREL+1
   IF (LCZERO.and.ir.lt.0) WRITE(*,*)' IR,ICREL ', IR,ICREL
!  LTCOVN=LTCOV.AND.(NRREL.EQ.JRNEXT.OR.(MOD(IR,25).EQ.24).OR.IR.EQ.IC)
   LTCOVN=.FALSE.
!
   LREROW = LREPER
!
   IF ((.NOT.LPARMP).AND.LNPARQ.AND.(.NOT.LCOD)) GOTO 3152

   COV = D0
!  IF (.NOT.(LNPARQ.AND.LCOD)) THEN
   IF (LNPARQ.AND.LCOD) GO TO 3018
! PARTIALS WITH RESPECT TO PARAMETERS.
   IF (IIR.EQ.(ILAST+1)) THEN
    IFIRST = IIR
    ILAST  = IPACAT(IPA+1)
    MP   = IPACAT(IPA+2)
    IPA  = IPA+2
    LEQP = MP.GE.0
    MP   = IABS(MP)
   END IF

   IF (LPARMQ.OR.IIR.EQ.IFIRST.OR.(.NOT.LEQP).OR.LCOD) THEN
!   IF (LPARMQ) IPT = IPTYPE(KT+1)
    IF (LPARMQ) IPT = IPTYPE(KG+1)
    LSAME = LF
    if (LTCOV) write(*,*)' 11399 IPT,KT,KG ',IPT,KT,KG
    IF (MP.NE.0) THEN
     IA = 1
     I  = 0
 3003   I = I+1
   LSAME = LSAME .OR. (IPT.EQ.IPACAT(IPA+I))
!  write(*,*)' 11404 IPT,IPACAT(IPA+I),IPA,I,LSAME ',&
!  IPT,IPACAT(IPA+I),IPA,I,LSAME
   IF ((.NOT.LSAME).AND.I.LT.MP) GO TO 3003
   IF (IKP.EQ.9.AND.LSAME.AND.I.EQ.2) IA=-1
   IF (IA.EQ.-1)WRITE(6,*)IA
! WE PUT IA=-1, BECAUSE IKP=9 INDICATES A SATELLITE ALTIMETRY
! CROSS-OVER DIFFERENCE, WITH THE SECOND PART ASSOCIATED WITH THE
! MINUS PART.
  END IF
  IF (LPARMP) IPA = IPA+MP
! write(*,*)'11287 LPARMP,Q,LSAME,IKP,MP,IA ',LPARMP,LPARMQ,LSAME,IKP,MP,IA
 END IF
!
 KQ = 2
! IF WE HAVE TWO OBSERVATIONS IN ONE POINT, A JUMP BACK TO THIS
! LABEL WILL BE MADE, WITH KQ NOW EQUAL TO 1.
 3014 IKA = IKQ
 IF (IKQ.GE.26) IKA=IKQ-8
 IF (LREPER) IKA = IKA-KQ
 COV = D0
 IF (LTCOV) WRITE(*,*)' 11422 LPARMP,LNPARQ,LSAME ICREL,Kg ',&
 LPARMP,LNPARQ,LSAME,ICREL,KG
```



```
 IF (LPARMP.AND.LNPARQ.AND.LSAME) THEN
  COV = APARM(COSLAT(ICREL),SINLAT(ICREL),RLONG(ICREL),HQ(ICREL),IKA,IPT,IIR)
! IF (LTCOVN.or.(.true.)) WRITE(*,*)' 11299 IKAIPTIIRCOV ',IKA,IPT,IIR,COV
 END IF
 KT = KT+1
 KG = KG+1
 IF (LPARMQ.OR.LCOD) GO TO 3015
 GO TO 3020
!
 3152 CONTINUE
 COSLAQ = COSLAT(ICREL)
 SINLAQ = SINLAT(ICREL)
 SINLOQ = SINLON(ICREL)
 COSLOQ = COSLON(ICREL)
! CHANGE 1999-05-17 BY CCT.
 SLOP = SINLOP
 CLOP = COSLOP
 SLOQ = SINLOQ
 CLOQ = COSLOQ
 DLAT = -(RLATP-RLAT(ICREL))
 DLONG = RLONGP-RLONG(ICREL)
 IF (LSATQ) THEN
  CAZQ = COSAZ(ICREL)
  SAZQ = SINAZ(ICREL)
  CCR(11) = D1
  CCR11= D1
 END IF
 IF (LMEANQ.AND.(.NOT.LEQANQ).AND.(.NOT.LMEAQ1)) CALL ICMEAN(BSIZQN,STEPQE,5,COS
SQE,SINSQE,COSLAQ,SINLAQ,LF,LF)
 IF (LMEAQ1) THEN
  IF (LDIFQ) THEN
   RLATE=SR11(ICREL)
   RLONGE=SR12(ICREL)
   HQE=SR13(ICREL)
   if (icrel.lt.0) write(*,*)' 11830, lat,lon,h ',RLATE,RLONGE,HQE
   SINLATE=SIN(RLATE*DEGRAD)
   COSLATE=COS(RLATE*DEGRAD)
   SINLONE=SIN(RLONGE*DEGRAD)
   COSLONE=COS(RLONGE*DEGRAD)
  ELSE
   COSSQE = COSAZ(ICREL)
   SINSQE = SINAZ(ICREL)
   STEPQE = -D1
  END IF
 ELSE
! 2001-07-15.
  STEPQE = D1
 END IF
!
 CI20=D1
 CCI(20)=D1
 CI17 = SIN(DLAT/D2)
 CCI(17) = CI17
 COSDLO  = COS(DLONG)
 SIDLO2  = SIN(DLONG/D2)**2
 SINDLO   = SIN(DLONG)
 CI16=SIDLO2
 CCI(16)= CI16
 SINDLA  = CCI(17)**2
 CI18=COS(DLAT)
 CI19=COS(DLAT/D2)
 CCI(18)=CI18
 CCI(19)=CI19
 IF (.NOT.((ABS(DLAT).LT.1.0D-2).OR.(ABS(DLONG*COSLAQ).LT.1.0D-2))) THEN
!  COSDLO=COSLOP*COSLOQ+SINLOP*SINLOQ
! ERROR 2002-09-30.
!  IF (LDEFVP.OR.LDEFVQ) CCR(8)=-SINLOP*COSLOQ+COSLOP*SINLOQ
 T =  SINLAQ*SINLAP+COSLAP*COSLAQ*COSDLO
 T1 = D1-T
```

```
 IF (T.LT.-1.0D0) THEN
!   WRITE(*,*)' WARNING, T < -1 ',T
  T = -1.0D0
 END IF
 IF (T.GT.1.0D0) THEN
!     WRITE(*,*)' WARNING, T >  1 ',T
  T = 1.0D0
 END IF
!
 ELSE
  CI20=D0
  CCI(20)=D0
!
! T IS COSINE TO THE SPHERICAL DISTANCE BETWEEN P AND Q, CF.REF(B),
! EQ.(57).
!3200 T1 = D2*(SINDLA+COSLAP*COSLAQ*SIDLO2)
  T1 = D2*(SINDLA+COSLAP*COSLAQ*SIDLO2)
  T = D1-T1
  IF (.NOT.(LPRED.OR.(IKP.NE.IKQ).OR.IC.EQ.IR)) THEN
! THE VARIABLE CCI(20) IS USED TO INDICATE WHETHER THE PRECISE
! FORMULAE MAY BE USED IN COVAX.
 IF (( ABS(T1).LT.1.0D-9).AND.( ABS(HHP-HQ(ICREL)).LT.1.0))WRITE(6,300) IC,IR
  300 FORMAT(' ** WARNING30 ** CURRENT OBS ',I7,' MAY BE IDENTI',&
 'CAL TO OBS NO',I7)
!
  END IF
 END IF
 CCR(1) = T
 CCR(5) = SINLAQ
 CCR(7) = COSLAQ
 CCR(8) = -D1*SIN(DLONG)
 CCR(9) = COSDLO
!
 LKSIQ=LREPER.OR.IKQ.EQ.3.OR.IKQ.EQ.16.OR.IKQ.EQ.18.OR.IKQ.EQ.20.OR.IKQ.EQ.25.OR
.IKQ.EQ.22
 LETAQ=IKQ.EQ.4.OR.IKQ.EQ.17.OR.IKQ.EQ.19.OR.IKQ.EQ.21.OR.IKQ.EQ.23.OR.IKQ.EQ.24
!  IF (LSPHAR) WRITE(*,*)' LKSIQ,LETAQ= ',LKSIQ,LETAQ
!
 CCR(3) = HQ(ICREL)
 HHQ = HQ(ICREL)
 RQ = RE+HQ(ICREL)
 IF (LSATQ) THEN
  CCR(11) = D1
 ELSE
  CCR(11) = GMC/RQ**2
 END IF
 KQ = 1
 IF (LKSIQ) KQ = 2
 KP = 1
 IF (.NOT.LNKSIP) KP = 2
!
! FINITE COV FEATURE REMOVED 2011-12-25 BY CCT.
!LCZERO=IKP.EQ.ICZERO.AND.IKQ.EQ.ICZERO.AND.(ABS(HHP-HCZERO).LT.D1).AND.(ABS(HQ(
ICREL)-HCZERO).LT.D1)
! LCZERO IS TRUE WHEN A FINITE COVARIANCE FUNCTION MUST BE USED.
!IF (LCZERO.AND.NCZERO.EQ.-1) THEN
! WE MUST DETERMINE COVARIANCE FUNCTION PARAMETERS
! WE NOW DETERMINE THE VALUE OF THE CORRELATION DISTANCE.
! WRITE(*,*)' FINITE COVARIANCE FUNCTION IN USE C '
! CCR(1) = D1
! CALL COVCX(SM,COV,COVX,IS,LSATPQ)
! C0  = COV
! PSI = D0
! C1  = C0
! DPSI = 5.0D-4
! TEST3 = D1
!3120  PSI = PSI+DPSI
! C2  = C1
! CCR(1) = COS(PSI)
```



```
! CALL COVCX(SM,COV,COVX,IS,LSATPQ)
! C1 = COV
! TEST1 = TEST3
! TEST3 = C1/C0
! IF (TEST3.GT.1.0D0.OR.TEST3.LT.D0) WRITE(*,*) ' WARNING31 ',C0,C2,C1
! IF (TEST3.GT.0.5D0) GO TO 3120
! DPSI=DPSI/D2
! PSI=PSI-DPSI
! KPSI=0
!3121  CCR(1)=COS(PSI)
! CALL COVCX(SM,COV,COVX,IS,LSATPQ)
! C3 = COV
! KPSI = KPSI+1
! TEST2 = C3/C0
! DPSI = DPSI/D2
! IF (TEST2.GT.0.5D0.AND.TEST3.LT.0.5D0) THEN
!  TEST1 = TEST2
!  PSI = PSI+DPSI
! ELSE
!  TEST3 = TEST2
!  PSI = PSI-DPSI
! END IF
! IF (KPSI.LT.115.AND.ABS(0.5D0-TEST2).GT.1.0D-8) GO TO 3121
! PSI1 = PSI
! 1 RDD = FINDR(PSI1,1)
! WRITE(*,3122)C0,PSI1,RDD
!3122  FORMAT(' C0= ',F12.4,' CORREL.DIST.',F10.5,' R ',F10.5)
! SCFACT=C0/COZERO(D0,RDD,1)
! NCZERO = 0
!END IF
!
 IF (LMEANP.OR.LMEANQ) GO TO 3040
!
!   if (lpred.and.ir.le.2) write(*,*)' T,CQ,SQ,CP,SP,C8,C9,CI16,CI17,CI19 ',T,CO
SLAQ,SINLAQ,COSLAP,SINLAP,&
!   CCR(8),CCR(9),CI16,CI17,CI19
! WRITE(*,*)' 10169 dlat,ci17 ',dlat,ci17
  LTESTS=LCZERO.and.nrrel.lt.2.and.LPRED

!write(*,5507)NRREL,cov,t,cfa,kp,kq
 if (nrrel.lt.0) write(*,5507)NRREL,cov,t,cfa,kp,kq
  CALL COVCX(SM,COV,COVX,IS,LSATPQ)
  CFA=CFX
 if (nrrel.lt.0.and.lpred) write(*,5507)NRREL,cov,t,cfa,kp,kq
 5507 format(' 11437 NRREL,COV,T,CFA ',1i3,3d15.6,2i3)
  CCV(KP,KQ)=COV
!
! IF WE USE A SATELLITE ORIENTED FRAME, THE COVARIANCE MATRIX MUST
! BE ROTATED TO THIS FRAME. CHANGE AUG 89 BY CCT. CORRECTED MAR 95.
 IF (LSATP.AND.LDEFVP) CALL SROT(CCV,SAZP,CAZP,1,LF)
 IF (LSATQ.AND.LDEFVQ) CALL SROT(CCV,SAZQ,CAZQ,1,LT)
 IF (LSATPQ) THEN
  IF (LROT) THEN
! CHANGE 2003-03-11.
   IF (ISAT(JR-2).NE.0) THEN
    COSB = SR11A(ICREL)
    SINB = SR12A(ICREL)
    cost = sr13a(icrel)
    SINT = SR22A(ICREL)
    CAZQ = COSAZA(ICREL)
    SAZQ = SINAZA(ICREL)
    if (ir.lt.0)write(*,*)' 10524 JR,ISAT(JR-2),cscscs ',&
    JR,ISAT(JR-2),COSB,SINB,COST,SINT,CAZQ,SAZQ,ICREL
    SROTQ(1,1) =  SAZQ*COSB
    SROTQ(1,2) =  CAZQ*COST+SAZQ*SINB*SINT
    SROTQ(1,3) = -CAZQ*SINT+COST*SAZQ*SINB
    SROTQ(2,1) = -CAZQ*COSB
    SROTQ(2,2) =  SAZQ*COST-SINT*CAZQ*SINB
    SROTQ(2,3) = -SAZQ*SINT-COST*CAZQ*SINB
```

```
    SROTQ(3,1) = -SINB
    SROTQ(3,2) =  COSB*SINT
    SROTQ(3,3) =  COST*COSB
   ELSE
! CORRECTION 2003-03-05.
    SROTQ(1,1) = D1
    SROTQ(2,2) = D1
    SROTQ(3,3) = D1
    SROTQ(1,2) = D0
    SROTQ(1,3) = D0
    SROTQ(2,1) = D0
    SROTQ(2,3) = D0
    SROTQ(3,1) = D0
    SROTQ(3,2) = D0
   END IF
!  IF (LTCOV) THEN
   IF (ir.lt.0) then
     WRITE(*,*)'COVX ',COVX
!    WRITE(*,150)SATROT,SROTQ,((((COVX(IAA,IBB,ICC,IDD),IAA=1,3),IBB=1,3),ICC=1,
3),IDD=1,3)
  150 FORMAT(' SATROT,SROTQ,COVX ',/,9F8.4,/,9F8.4/21(5D14.5/))
   END IF
!
!  IF (.NOT.(KPP.EQ.3.OR.KQQ.EQ.3)) THEN
! ACTIVATED STATEMENT 2010-12-01.
    LCHECK = CHECKC(1)
    CALL COVROT(COVX,SATROT,SROTQ)
! CHANGE 2010-10-28.
    LCHECK = LCHECK.AND.CHECKC(2)
    IF ((.NOT.LCHECK).AND.NWAR.LT.25) THEN
         WRITE(*,153)IR,SATROT,SROTQ
 153     FORMAT(' CHECKC,IR ',I6,/,2(9F8.5,/))
! CHECK OF ORTHOGONALITY OF ROTATION MATRICES.
     DO IAA  = 1,3
      ROTSUM = D0
      ROTSUA = D0
      DO IBB = 1,3
       ROTSUM = ROTSUM+SATROT(IBB,IAA)*SATROT(IBB,IAA)
       ROTSUA = ROTSUA+SROTQ(IBB,IAA)*SROTQ(IBB,IAA)
      END DO
      IF (ABS(ROTSUM-D1).GT.1.0D-8) WRITE(*,*)' ROTSUM ',ROTSUM,IAA
      IF (ABS(ROTSUA-D1).GT.1.0D-8) WRITE(*,*)' ROTSUA ',ROTSUM,IAA
     END DO
    END IF
!  END IF
!  IF (LTCOV) THEN
!   WRITE(*,151)((COVX(IAA,IBB,KSAT(KQQ,1),KSAT(KQQ,2)),IAA=1,3),IBB =1,3)
 151  FORMAT(9E9.2)
!  END IF
  END IF
!
! if (ir.lt.3.and.lsphar) write(*,*)' 10272 KPP,KQQ ',KPP,KQQ
  IF ((KPP.NE.15.AND.KQQ.NE.15).and.(.not.LSPHAR)) THEN
   COV = COVX(KSAT(KPP,1),KSAT(KPP,2),KSAT(KQQ,1),KSAT(KQQ,2))*CFA
       if (IR.eq.0.and.(.not.LPRED)) then
!       write(*,*)' ks1212 CFA,COV ',KSAT(KPP,1),KSAT(KPP,2),KSAT(KQQ,1),&
!      KSAT(KQQ,2),CFA,COV
        write(*,7676)COVX(3,3,3,3),t,cfa
       end if
   IF ((.not.LSPHAR).and.LPRED.AND.NRREL.LT.0) THEN
    WRITE(*,*)' 10254',KSAT(KPP,1),KSAT(KPP,2),KSAT(KQQ,1),KSAT(KQQ,2),KPP,KQQ,C
FA,COV
    WRITE(*,151)((COVX(IAA,IBB,KSAT(KQQ,1),KSAT(KQQ,2)),IAA=1,3),IBB=1,3)
    ITCOUN = ITCOUN+1
   END IF
! CHANGE 2002-10-23.
  ELSE
   IF (KPP.EQ.15.AND.KQQ.NE.15) THEN
! DDT/DXX-DDT/DYY IN P.
```



```
    COV = (COVX(KSAT(14,1),KSAT(14,2),KSAT(KQQ,1),KSAT(KQQ,2))-COVX(KSAT(12,1),K
SAT(12,2),KSAT(KQQ,1),KSAT(KQQ,2)))*CFA
   ELSE
    IF (KPP.NE.15.AND.KQQ.EQ.15) THEN
! DDT/DXX-DDT/DYY IN Q.
     COV = ( COVX(KSAT(KPP,1),KSAT(KPP,2),KSAT(14,1),KSAT(14,2))   &
            -COVX(KSAT(KPP,1),KSAT(KPP,2),KSAT(12,1),KSAT(12,2)))*CFA
    ELSE
! DDT/DXX-DDT/DYY IN BOTH P AND Q.
     COV = ( 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)))*CFA
! OBS IN P AND Q ARE BOTH DDT/DXX-DDT/DYY.
    END IF
   END IF
! CHANGE 1992.08.26.
  END IF
   GO TO 3020
 END IF
 if (lsphar.and.nrrel.lt.0) write(*,*)'10281 cov kp,kq,cov ',kp,kq,cov
 GO TO 3011
!
 3040 CONTINUE
!write(*,*)'10285 cov kp,kq,cov ',kp,kq,cov
 COV = COMEAN(SM,IS,HHP,HHQ,COSLAP,SINLAP,COSLOP,SINLOP,COSLAQ,SINLAQ,&
       COSLOQ,SINLOQ,NSTEPP,NSTEPQ,LSATPQ,NRREL.LT.0)
 if (nrrel.lt.0.and.lpred) write(*,*)'11554 cov kp,kq,cov ',kp,kq,cov
!IF (ABS(COV).LT.1.0D-10.AND.LCZERO) NCZERO = NCZERO+1
!
 3011 if (NRREL.LT.0.and.lpred) WRITE(*,*) ' 10288 ',COV,CCV(KP,KQ),KP,KQ,LNETAP
,LNKSIP
 IF (LNETAP) GO TO 3012
!write(*,*)' 11577 CCV ',CCV
!  COVARIANCE BETWEEN ETAP AND OTHER FUNCTIONALS IN Q.
 KP = 1
 IF (LNPARQ) COV = CCV(KP,KQ)
!if (lsphar.and.nrrel.lt.4) write(*,*)'10293 cov kp,kq,cov ',kp,kq,cov
 3015 IF (LONECO) GO TO 3012
 IF (LPARMQ.AND.LSAME) THEN
  COV = APARM(COSLAP,SINLAP,&
  RLONGP,HHP,IETA,IPT,IIR)*IA
  IF (LTCOV) WRITE(*,*)' 11594 IETAIPTIIRIACOV ',IETA,IPT,IIR,IA,COV
 END IF
!
 IF (LNBC) C(NI+NC) = COV
 IF (LBST) B(NRREL) = COV
 IF(LPRED) PRETAP = PRETAP+B(NRREL)*COV
 IF (LTCOVN) WRITE(6,3961)NI,NC,KP,KQ,IKP,IKQ,T,COV,PRETAP,&
     NRREL,WOBS(NRREL)
 3961 FORMAT(' NI,NC,KPKQIKPIKQTVCOVPRETAP,NRREL,W ',6I3,F6.3,2D16.6,i3,F8.4)
!
 3012 IF(LNKSIP)GO TO 3013
!  COVARIANCE BETWEEN KSIP AND OTHER FUNCTIONALS IN Q.
 KP=2
!
 3013 IF (LNPARQ.AND.(.NOT.LCOD))THEN
  COV=CCV(KP,KQ)
  if ((lczero).and.LPRED.AND.NRREL.LT.0) write(*,*)' 10313 ',CCV(KP,KQ),KP,KQ
 END IF
!write(*,*)' 11612 LPARMQ,LPARMP ',LPARMQ,LPARMP
 IF (LPARMQ.AND.LPARMP) THEN
! BLOCKING OF CX IMPLEMENTED 1992.12.18 BY CCT.
!INDX=IT*(IT-1)/2+KT-1
 INDX=IT*(IT-1)/2+KG-1
 NEWCX = INDX/NPMAX+1
 INDX = MOD(INDX,NPMAX)+1
 IF (INDX.EQ.1) THEN
  READ(2,REC=NEWCX)CPX
```

```
  IF (LTESTS) THEN
  WRITE(*,*)' 11620 CUNIT 2 READ, BLOCK', NEWCX
  WRITE(*,*)(CPX(IGG),IGG=1,6)
  END IF
 END IF
 COV = CPX(INDX)
 IF (LTCOVN) WRITE(*,*)' INDXCOV ',INDX,COV
 END IF
! THE VALUES OF CX ARE COMPUTED IN THE SUBROUTINE TRANS.
 IF (LPARMQ.AND.(.NOT.LPARMP.OR.LCOD).AND.LSAME) THEN
  COV=APARM(COSLAP,SINLAP,RLONGP,HP,IKB,IPT,IC+1)*IA
  IF (LTCOVN) WRITE(*,*)' 11631 IKBIPTICIACOV ',IKB,IPT,IC,IA,COV
 END IF
 LTCOVN=LF
 3020 IF (LCST.AND.(.NOT.LWAIT)) THEN
  C(NI) = COV
!  IF (LTESTS) WRITE(*,*)' NI, IC,IR ',NI,IC,IR,COV
  LLCOER = ITRACE(IC).EQ.ITRACE(IR).AND.ITRACE(IC).LT.0.AND.LCOERR
  IF (IC.GT.MAXO.OR.IR.GT.MAXO) THEN
   LLCOER=LF
  ELSE
!       IF (LNPARQ.AND.(.NOT.LPRED).AND.LLCOER) THEN
   IF ((.NOT.LPRED).AND.LLCOER) THEN
    IF (LFOUR) THEN
     ERRCOV=D0
! NOT YET IMPLEMENTED 2002-10-10.
    ELSE
!  2005-04-04 IMPLEMENTED THAT TIME DIFFERENCES AND NOT SPHERICAL
! DISTANCE  MAY BE AN ARGUMENT IN COZERO.
     IF (LCTIME) THEN
      PSI=ABS(CTIME(IC)-CTIME(IR))
     ELSE
      PSI=ACOS(T)
     END IF
!    ERRCOV = SCFACT*COZERO(PSI,RDD,1)
     NERCOV=NERCOV+1
    END IF
    IF (LTCOV) WRITE(*,3049)ERRCOV,IC,IR,ITRACE(IC),C(NI)
 3049    FORMAT(' ERROR-CORR.',F12.5,' FOR OBS ',2I5,&
    ' ADDED TO COV. ON ',/,' TRACK ',I9,' COV= ',F13.6)
! WE ADD NOISE TO THE DIAGONAL ELEMENT, CHANGE 1992.07.19. AND 1997-07-15.
    C(NI) = C(NI)+ERRCOV
   END IF
  END IF
  IF (LNPARQ.AND.(IC.EQ.IR).AND.(.NOT.LPRED).AND.(.NOT.LLCOER)) THEN
   C(NI)=COV+WOBS(NRREL)**2
  END IF
 END IF
!IF (LSPHAR.AND.LPRED.AND.LTESTS) WRITE(*,*)' PREDP,COV,B,N ',&
 IF ((LSPHAR).AND.LPRED.AND.NRREL.LT.0) WRITE(*,7111)PREDP,CCV,COV,0.0d0,NRREL,L
SATPQ
 7111 format(' 10443 PREDP,CCV,COV,B,N,kp,kq,nrrel,lsatpq ',7d14.5,i4,l2)
 IF(LPRED) THEN
  PREDP = PREDP+COV*B(NRREL)
  IF (.NOT.LALLCO) THEN
! AS AN INTERMIDIATE STEP WE STORE THE COVARIANCES IN CR FOR LATER
! TRANSFER USING RESTORE. 2007-10-30.
    if (NI.GT.0.and.(.not.LSPHAR).and.LTCOV) write(*,*)&
    ' 11893 NPRED, NI,COV ',&
   NPRED,NI,COV
   C(NI)= COV
  END IF
  IF (LALLCO) THEN
! ACCUMULATING THE PREDICTION FROM COLLOCATION. 2005-09-23.
! HERE MUST BE CORRECTED FOR 1.ORDER DERIVATIVES. 2006-09-13.
   IF (IORDER.EQ.1) THEN
    DO IAA=1,3
     ALLPRE(IAA,1)=ALLPRE(IAA,1)+COVX(IAA,1,KSAT(KQQ,1),&
     KSAT(KQQ,2))*CFA*B(NRREL)
```



```
    END DO
   END IF
   IF (IORDER.EQ.2) THEN
    DO  IAA=1,3
     DO IBB=1,3
      ALLPRE(IAA,IBB)=ALLPRE(IAA,IBB)+COVX(IAA,IBB,KSAT(KQQ,1),&
      KSAT(KQQ,2))*CFA*B(NRREL)
!          write(*,*)' all ',alleol(iaa,ibb),covx(iaa,ibb,ksat(kqq,1),&
!    *     ksat(kqq,2)),nr
     END DO
    END DO
   END IF
   ICC=0
   NREL=NREL+1
   DO IAA=1,3
    DO IBB=1,IAA
! HERE WE STORE COVARIANCES TO BE USED IN ERROR-ESTIMATION IN COPRED.
     ICC=ICC+1
     ALLCOV(NREL,ICC)=COVX(IAA,IBB,KSAT(KQQ,1),KSAT(KQQ,2))*CFA
!    ALLCOV(NR,ICC)=COVX(IAA,IBB,KSAT(KQQ,1),KSAT(KQQ,2))*CFA
    END DO
   END DO
! WE MUST TAKE CARE OF THE LAST BLOCK.
   IF (NREL.EQ.MAXO) THEN
    NREL=0
    WRITE(15,REC=NOBLK)ALLCOV
    NOBLK=NOBLK+1
   END IF
  END IF
 END IF
!
!write(*,*)' 12399 predp ',predp
 IF (LTCOVN) WRITE(6,7)NI,NC,KP,KQ,IKP,IKQ,T,COV,PREDP,&
   NRREL,WOBS(NRREL)
  7   FORMAT(' KPKQIKPIKQTCOVPRECNI,NRREL,W=',6I3,F13.10,2F12.7,i3,F12.7)
!
!END IF
 3018 NI = NI+1
!NI = NI+1
 NR1 = NR
 NR = NR+1
!write(*,*)' NR1,IC,IR ',NR1,IC,IR
 IF (.NOT.LREROW) GO TO 3019
! READ OBS-BLOCK, CHANGE 1992.07.19.
 NRREL=MOD(NR1,MAXO)+1
 IF (LOBSST) THEN
  ICBL=NR1/MAXO+1
  IF (NRREL.EQ.1.OR.IR.EQ.1) THEN
!     WRITE(6,*)'BLK ',ICBL,' 15 READ FOR TRANSFER B TO C.'
  READ(16,REC=ICBL)B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS
   IF (LSATAC) READ(14,REC=ICBL)SR11,SR12,SR13,SR22,COSAZ,SINAZ
!       WRITE(6,*)'BLK ',NOBLK,' 16 READ FOR TRANSFER B TO C.'
   ICREL=0
  END IF
 END IF
! WE ADD NOISE TO THE DIAGONAL ELEMENT, CHANGE 1992.07.19.
 IF (LNPARQ.AND.(IC.EQ.IR).AND.(.NOT.LPRED)) THEN
  COV=COV+WOBS(NRREL)**2
  IF (LBST) THEN
   B(NRREL) = COV
  ELSE
   C(NI+NC)= COV
!  write(*,*)' 11921 NI,NC,COV ',NI,NC,COV
  END IF
 END IF
!
 IF (IC.EQ.IR.AND.(.NOT.LPRED)) GO TO 3019
 LKSIQ = .FALSE.
 LETAQ = .TRUE.
```

```
 KQ=1
 LREROW = .FALSE.
 IF (LPARMP) GO TO 3014
 GO TO 3011
 3019 CONTINUE
 END DO
 END IF
!IF (LPRED) WRITE(*,*)' NI,NR,NC ', NI,NR,NC
 JRNEXT=ANDEX(JR-2)
!
!WRITE(*,*)' LPRED,IMSET ',LPRED,IMSET
 END DO
! END OF LOOP COMPUTING A NR-1 VECTOR OF COVARIANCES.
 IF (LALLCO) THEN
! ADDED 2008-09-24.
  IF (IORDER.EQ.1) THEN
   DO IAA=1,3
     ALLCOL(IAA,1)=ALLPRE(IAA,1)+ALLCOL(IAA,1)
   END DO
  END IF
  IF (IORDER.EQ.2) THEN
   DO  IAA = 1,3
    DO IBB = 1,3
     ALLCOL(IAA,IBB) = ALLPRE(IAA,IBB)+ALLCOL(IAA,IBB)
    END DO
   END DO
  END IF
 END IF
!write(*,*)' 12465 predp ',predp
END SUBROUTINE PRED
      DOUBLE PRECISION FUNCTION COZERO(PSI,R,MODEL)
! THE SUBROUTINE WILL EVALUATE A COVARIANCE FUNCTION WHICH
! IS ZERO AFTER THE DISTANCE (PSI) IS LARGER THAN OR EQUAL TO 2*R.
! PROGRAMMED APRIL 1996 BY C.C.TSCHERNING, GEOPHYSICAL
! DEPARTMENT. LAST CHANGE 2002-10-24.
!     PSI -  SPHERICAL DISTANCE IN RADIANS
!     R   -  2*R DISTANCE IN RADIANS FOR WHICH FUNCTION IS ZERO.
!     MODEL - PARAMETER TO DISTINGUISH BETWEEN DIFFERENT MODELS.
!
      IMPLICIT NONE
      REAL*8 D0,D2,D3,PI,PSI2,PSI,PSI3,R2,R,R4
      INTEGER MODEL
!
      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

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE SETCAT(NB)
!SUBROUTINE SETCAT(IFC,NB,LFULL,LRESTA,LSATAC)
```



```
! PROGRAMMED 1989.07.10 BY EXTRACTION FROM GEOCOL MAIN, BY
! C.C.TSCHERNING., LAST UPDATE 2000-07-27 BY CCT.
! IT IS PARTLY ABSOLETE, 2012-05-09.

 USE m_params,      ONLY : NDIMC,NISIZE,NCRW,NNBL,MAXO,NSAT,MAXO9,NEQFIM
 USE m_geocol_data, ONLY : C,NCAT,ISZE,NBL,MAXBL,ISIZE,                  &
                           NEQFI,NEQFMA,MAXBNE,DNANE,LNBL1

 USE m_data,        ONLY : BSIZE,ANDEX,PRETAP,PREDP,HCZERO,NCZERO

 USE m_geocol_data, ONLY : B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,  &
                           WOBS,ISAT,SINLOP,COSLOP,BSIZEN,BSIZEE,COSLAP, &
                           SINLAP,RLONGP,RP,CAZP,SAZP,HP,RLATP,ICZERO,   &
                           NI,NR,IKP,ISATP,NOBLK,LONECO,LNKSIP,LNETAP,   &
                           LDEFVP,LOBSST,LSTART
 USE m_data,        ONLY : OLDB,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

IMPLICIT NONE

 INTEGER ::      MAXBNO,MAXCOL,M,NBT,NB,I,I1,MBSIZE

 LOGICAL ::      LL

! LFULL IS TRUE IF A FULL MATRI IS USED. LRESTA IS TRUE IF THE
! NORMAL EQUATIONS ALREADY HAVE BEEN ESTABLISHED AND REDUCED PARTIALLY.

!COMMON /BIPAR/OLDB(4),CNR,GMP,AX,NMAX,II,IOBS,IOBSR,N1,NIR,MAXC,MAXC1,MAXC2,N,I
C,NT,IDIMC,IDIMCN,MAXBLT,JR,ISO,LPARAM,LL(4)

! SETTING UP A CATALOGUE OF THE NORMAL EQUATIONS
! NB IS THE RECORD NUMBER, IC COUNTS THE NUMBER OF COLUMNS WITHIN A
! RECORD, AND THIS NUMBER IS STORED IN NCAT(ISIZE). NCAT (I) WILL CONTAIN
! THE SUBSCRIPT OF THE DIAGONAL ELEMENT OF COLUMN I-1, AND ALL THE
! ELEMENTS OF ISZE WILL BE ZERO BECAUSE WE WORK WITH A FULL MATRIX.
! NBL(I) CONTAINS THE NUMBER OF THE LAST COLUMN IN RECORD I-1.  NOTE
! THAT MAXIMALLY MAXCOL COLUMNS MAY BE STORED IN ONE RECORD.

 MAXBNE = -1
 MAXBNO = -1
 MAXCOL = ISIZE-2
 MBSIZE = NEQFI(1,2)
! BLOCK-SIZE FOR BLOCKS USED IN MULTIPROCESSING.
!     WRITE(*,*)' MAXCOL ',MAXCOL
 2199 M = (MAXCOL*(MAXCOL+1))/2
 IF (M.LE.IDIMC) GO TO 2200
 MAXCOL = MAXCOL-1
 WRITE(6,330)M,IDIMC,MAXCOL
  330 FORMAT(I6,' SPACE NEEDED, BUT ',I10,' AVAILABLE',/,&
 ' MAXCOL SET EQUAL TO', I10)
 GO TO 2199
 2200 NBT = 1
 NBL(1) = 0
 NB = 1
 IC = 0
 I1 = 0

! ADDED 2007-08-20.
! EVERY BLOCK-COLUMN CONTAIN THE SAME NUMBER OF ORDINARY COLUMNS.
! THIS MUST BE MODIFIED IF MORE COLUMNS ARE ADDED.
 DO I = 1,N1
  I1 = I1+I
  IF (MOD(I,MBSIZE).EQ.0.OR.I.EQ.N1) THEN
   NB = NB+1
   NBL(NB) = I
  END IF
 END DO
 WRITE(*,*)NB,' COLUMNS OF BLOCKS USED N1= ',N1,' MBSIZE=',MBSIZE
```

```
 MAXC  = I1-N1-NBL(NB-1)
 MAXC2 = I1-NBL(NB-1)
 IF (.FALSE.) WRITE(*,*)' MAXC,MAXC2 ',MAXC,MAXC2

END SUBROUTINE SETCAT

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

FUNCTION APARM(CLAT,SLAT,RLON,HP,IKP,IT,IIR)
! PROGRAMMED SEPT. 1984 BY C.C.TSCHERNING, GEODAETISK INSTITUT, DANMARK.
! LATEST UPDATE JAN 07, 2013.
! THE SUBROUTINE COMPUTES THE CONTRIBUTION FROM PARAMETERS TO
! OBSERVATIONS OF TYPE IKP.
! THE INTEGER IIR IS THE OBSERVATION NUMBER, WHICH IS USED WHEN
! E.G. A TIME DEPENDENT TILT IS USED. THE TIME IS STORED IN THE
! ARRAY ITIME IN CTIME.
! SCALE FACTOR VARIABLES ARE STORE IN SFACT.
! REF(A): TSCHERNING, C.C.: DETERMINATION OF DATUM-SHIFT PARAMETERS
!         USING LEAST SQUARES COLLOCATION. BOLL.GEODESIA SC. AFF.,
!         AN. XXXV, NO.2, 1976.
!    (B): SOLER,T.: ON DIFFERENTIAL TRANSFORMATIONS BETWEEN CARTESIAN
!         AND CURVELINEAR (GEODETIC) COORDINATES. REP. OF THE DEP. OF
!         GEODETIC SCIENCE, NO. 236, THE OHIO STATE UNIVERSITY, 1976.
! *** WARNING *** NOT ALL MODES HAVE BEEN TESTED.

 USE m_params, ONLY : NIPT,MAXO,NIPCAT
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_data,        ONLY : IPTYPE,ITIME,ITIME0,NPARM,NPARM1
 USE m_geocol_data, ONLY : SFACT,FPERIO,IPACAT,ITROLD,ICODE,MAXPAR,MP,&
                           IPA,NCXLAS
 USE m_geocol_data, ONLY : SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS3,EPS2,EPS1,S1 => DL
,&
                           AX2,E22

IMPLICIT NONE

 INTEGER ::      IKP,IT,IIR,IKA,ITA,I

 REAL(KIND=8) :: CLAT,SLAT,RLON,CLON,SLON,COSDLO,SINDLO,W2,W,RN,RNH,&
                 HP,RM,RMH,APARM

!LOGICAL ::      LCOERR,LLCOER,LCTIME

!COMMON /ITRANC/SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS1,EPS2,EPS3,S1,AX2,E22
!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT
!COMMON /CPARM/SFACT(NIPCAT),FPERIO(10,2),IPTYPE(NIPT),IPACAT(3*NIPT),ITIME(NIPC
AT),ITIME0(NIPT),ITROLD,ICODE(10),NPARM,NPARM1,MAXPAR,MP,IPA,NCXLAS

 IKA = IKP
 IF (IKP.GT.25) IKA = IKA-10
 IF (IT.EQ.0) GO TO 3799
 IF (IT.GT.7.AND.IT.LT.11) GO TO 11
!     IF (IABS(IT).GT.10) GO TO 3797
 IF (IABS(IT).GT.1000) GO TO 3797
 ITA=MOD(IT,50)
 CLON =  COS(RLON)
 SLON =  SIN(RLON)
 GO TO 12
!
   11 COSDLO = COS(RLON-RLONG0)
 SINDLO = SIN(RLON-RLONG0)
!
   12 W2 = D1-E22*SLAT**2
 W =  SQRT(W2)
 RN = AX2/W
 RNH= RN+HP
 RM = AX2*(D1-E22)/(W*W2)
 RMH= RM+HP
!
```



```
 IF (IKA.EQ.2.OR.IKA.EQ.13.OR.(IKA.GT.17.AND.IKA.LT.26)) GOTO 3797
!
 GOTO (3701,3799,3703,3704,3703,3701,3703,3704,3799,3799,&
 3701,3799,3799,3799,3797,3703,3704),IKA
!
 3701 GO TO (21,22,23,24,25,26,27,28,29,30),ITA
! DERIVATIVES RELATED TO THE HEIGHT ANOMALY (ZETA) OR TO HEIGHT
! DIFFERENCES, CF. PG EQ.(5-55).
   21 APARM = -CLAT*CLON
 GO TO 3798
   22 APARM = -CLAT*SLON
 GO TO 3798
   23 APARM = -SLAT
 GO TO 3798
   24 APARM = -AX2*W+HP
 GO TO 3798
   25 APARM = -RN*E22*SLAT*CLAT*SLON
 GO TO 3798
   26 APARM = RN*E22*SLAT*CLAT*CLON
 GO TO 3798
   27 APARM = D0
 GO TO 3798
! NEXT THREE CASES CF. PG, EQ. (5-59).
   28 APARM = -(COSLA0*SLAT-SINLA0*CLAT*COSDLO)*RMH/RADSEC
 GO TO 3798
   29 APARM = -CLAT*SINDLO*RNH/RADSEC
 GO TO 3798
   30 APARM = SINLA0*SLAT+COSLA0*CLAT*COSDLO
 GO TO 3798
!
 3703 GO TO (41,42,43,44,45,46,47,48,49,50),ITA
! DERIVATIVES RELATED TO LATITUDE/KSI.
   41 APARM = -SLAT*CLON*RADSEC/RMH
 GO TO 3798
   42 APARM = -SLON*SLAT*RADSEC/RMH
 GO TO 3798
   43 APARM = CLAT*RADSEC/RMH
 GO TO 3798
   44 APARM = -RN*E22*SLAT*CLAT/RMH
 GO TO 3798
   45 APARM = -SLON*(AX2*W+HP)/RMH
 GO TO 3798
   46 APARM = CLON*(AX2*W+HP)/RMH
 GO TO 3798
   47 APARM = D0
 GO TO 3798
! CF. PG EQ.(5-59).
   48 APARM = CLAT*COSLA0+SLAT*SINLA0*COSDLO
 GO TO 3798
   49 APARM = -SLAT*SINDLO
 GO TO 3798
   50 APARM = -(SINLA0*CLAT-COSLA0*SLAT*COSDLO)*RADSEC/RMH
 GO TO 3798
!
 3704 GO TO (61,62,63,63,65,66,67,68,69,70),ITA
! DERIVATIVES RELATED TO CLAT*LONGITUDE/ETA, CF. PG EQ.(5-55).
   61 APARM = -SLON*RADSEC/RNH
 GO TO 3798
   62 APARM = CLON*RADSEC/RNH
 GO TO 3798
   63 APARM = D0
 GO TO 3798
   65 APARM = SLAT*CLON*(D1-E22*RN/RNH)
 GO TO 3798
   66 APARM = SLAT*CLON*(D1-E22*RN/RNH)
 GO TO 3798
   67 APARM = -CLAT
 GO TO 3798
   68 APARM = SINLA0*SINDLO
```

```
 GO TO 3798
   69 APARM = COSDLO
 GO TO 3798
   70 APARM = COSLA0*SINDLO*RADSEC/RNH
 GO TO 3798
!
 3797 IF (IT.GT.10000)APARM = D1
! this should probably be 100000. remark 2006-04-17.
! THIS IS THE ELEMENT IN THE A-MATRIX CORRESPONDING TO A BIAS.
!
! IT NEGATIVE IDENTIFIES THE SECOND PARAMETER ASSOCIATED WITH A
! TILT PARAMETER.
 IF (IT.LT.0)APARM=FLOAT(ITIME(IIR))
!     IF (IT.GT.10.AND.IT.LT.10000) APARM=SFACT(IIR)
 IF (IT.GT.10.AND.IT.LT.3400)  then
  APARM=SFACT(IIR)
!      write(*,*)' aparm ',it,aparm
 end if
 IF (IT.GT.16000.AND.IT.LT.97000) THEN
! PARAMETER ASSOCIATED WITH FOURIER COEFFICIENT. PERIOD IN FPERIO(I,1)
! AND PHASE IN FPERIO(I,2). 2006-03-20.
  I=IT/(IKA*400)
  IF (I.GT.10.OR.I.LT.4) THEN
   WRITE(*,*)' PARAMETER CODE WRONG ',I,IT
   STOP
  END IF
  APARM=COS(ITIME(IIR)*FPERIO(I,1)+FPERIO(I,2))
!      write(*,*)' aparm ',aparm,ika,it,fperio(I,1),fperio(I,2),i,&
!    * itime(iir)
 END IF
! SCALE FACTOR ADDED 2004-12-19.
 GO TO 3798
 3799 APARM=D0
 3798 RETURN
END FUNCTION APARM

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE PARCAT(LALLP,NPNO)
! PROGRAMMED FEB. 1985 BY C.C.TSCHERNING. LATEST UPDATE:
! AUG. 1993 BY CCT.
! THE SUBROUTINE INITIALIZES THE PARAMETER CATALOGUES IPACAT AND
! IPTYPE. WHEN THE LOGICAL VARIABLE LALLP IS FALSE, IT IS CHECKED
! WHETHER THE PARAMETERS STORED IN IPACAT(IPA+1),...,IPACAT(IPA+MP)
! ARE NEW PARAMETERS, AND THE CATALOGUE IPTYPE AND THE COUNTER
! NPARM ARE UPDATED. WHEN LALLP IS TRUE, IT IS CHECKED WHETHER
! THE PARAMETER CODES STORED IN IPACAT ARE ACCEPTABLE PARAMETERS,
! I.E. FOUND IN IPTYPE. IF NOT FOUND, THEY ARE PUT EQUAL TO ZERO.
!
! PARAMETERS IN CALL AND COMMON BLOCK CPARM:
!  IPACAT (CALL AND RETURN, INTEGER ARRAY OF DIMENSION 3*NIPT) HOLDS
!                PARAMETER CODES TO BE CHECKED IN VARIABLES WITH
!                SUBSCRIPTS IPA+1 TO IPA+MP.
!  IPTYPE (CALL AND RETURN, INTEGER ARRAY OF DIMENSION NIPT) HOLDS
!                FOR LALLP TRUE ALL ACCEPTABLE PARAMETER CODES AND
!                FOR LALLP FALSE ALL EARLIER ACCEPTED CODES
!                AND AT RETURN ALL CURRENTLY ACCEPTED CODES.
!  IPA    (CALL, INTEGER) IPA+1 POINTS AT CALL AT FIRST NEW PARAMETER
!                CODE IN IPACAT.
!  NPARM  (CALL AND RETURN) ACTUAL NUMBER OF ACCEPTED PARAMETERS.
!  MP     (CALL, INTEGER) IPA+MP POINTS AT LAST ACCEPTED PARAMETER.
!  NPNO   (RETURN,INTEGER) RETURNS THE PARAMETER NUMBER OF THE
!                PARAMETER ASSOCIATED WITH IPACAT(IPA+MP).
!

 USE m_params, ONLY : NIPT,NIPCAT
 USE m_data,        ONLY : IPTYPE,ITIME,ITIME0,NPARM,NPARM1
 USE m_geocol_data, ONLY : SFACT,FPERIO,IPACAT,ITROLD,ICODE,MAXPAR,MP,&
                           IPA,NCXLAS
```



```

IMPLICIT NONE

 INTEGER ::      I,J,NPNO

 LOGICAL ::      LALLP, LSAME

!COMMON /CPARM/SFACT(NIPCAT),FPERIO(10,2),IPTYPE(NIPT),IPACAT(3*NIPT),ITIME(NIPC
AT),ITIME0(NIPT),ITROLD,ICODE(10),NPARM,NPARM1,MAXPAR,MP,IPA,NCXLAS

 DO 10 I = 1, MP
 LSAME = .FALSE.
 DO 20 J = 1, NPARM
 IF (.NOT.LSAME)NPNO=J
   20 LSAME = LSAME .OR. (IPACAT(IPA+I).EQ.IPTYPE(J))
 IF (LSAME) GO TO 10
 IF (LALLP) GO TO 50
!
 NPARM = NPARM+1
 IF (NPARM.EQ.MAXPAR) THEN
! CORRECTION 2003-12-28.
  WRITE(6,99)MAXPAR
   99  FORMAT(' *** WARNING34 *** TOO MANY PARAMETERS, MAXPAR = ',I6)
  STOP
 END IF
 IPTYPE(NPARM) = IPACAT(IPA+I)
! THE PARAMETER TYPE CATALOGUE IS UPDATED.
 GO TO 10
!
   50 IPACAT(IPA+I) = 0
! PARAMETER CODE = 0 MEANS INDEPENDENT OF PARAMETERS.
   10 CONTINUE
 RETURN
END SUBROUTINE PARCAT

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE WRPAR                   !!JTF: NO ARGUMENTS ??
! PROGRAMMED FEB. 1985 BY C.C.TSCHERNING, LATEST UPDATE: MAR, 10, 2005 BY CCT.
! PARAMETER TYPE 11, SCALE FACTOR ADDED. FOURIER COEFFICIENTS ADDED
! 2006-03-25.
! THE SUBROUTINE LISTS THE PARAMETERS.

 USE m_params, ONLY : NIPT,NIPCAT
 USE m_data,        ONLY : IPTYPE,ITIME,ITIME0,NPARM,NPARM1
 USE m_geocol_data, ONLY : SFACT,FPERIO,IPACAT,ITROLD,ICODE,MAXPAR,MP,&
                           IPA,NCXLAS

IMPLICIT NONE

 INTEGER ::      IB,IBT,IBS,IBA,JI,I,IBF

 INTEGER, DIMENSION(NIPT) :: IPKIND

!COMMON /CPARM/SFACT(NIPCAT),FPERIO(10,2),IPTYPE(NIPT),IPACAT(3*NIPT),ITIME(NIPC
AT),ITIME0(NIPT),ITROLD,ICODE(10),NPARM,NPARM1,MAXPAR,MP,IPA,NCXLAS

 WRITE(6,149)NPARM
  149 FORMAT('0NUMBER OF PARAMETERS=',I4)
 NPARM1 = NPARM+1
! IB COUNTS BIAS PARAMETERS, IBT TILT-PARAMETERS AND IBS SCALE FACTORS.
! IBF COUNTS FOURIER COEFFICIENTS.
 IB = 0
 IBT=0
 IBS=0
 IBA=0
 IBF=0
 DO 1302 I = 1, NPARM
!     WRITE(*,*)' IPTYPE(I),I ',IPTYPE(I),I
```

```
 JI = IABS(IPTYPE(I))
 IF (JI.GT.10) GO TO 1312
! PARAMETERS OF TYPE BETWEEN 11 AND 999 ARE SCALE FACTORS.
 JI=MOD(JI,50)
 GO TO (1303,1304,1305,1306,1307,1307,1307,1309,1310,1311),JI
 1303 WRITE(6,151)
  151 FORMAT('  DELTA X')
 GO TO 1302
 1304 WRITE(6,152)
  152 FORMAT('  DELTA Y')
 GO TO 1302
 1305 WRITE(6,153)
  153 FORMAT('  DELTA Z')
 GO TO 1302
 1306 WRITE(6,154)
  154 FORMAT('  DELTA L')
 GO TO 1302
 1307 JI = 8-IPTYPE(I)
 WRITE(6,155)JI
  155 FORMAT('  EPS',I1)
 GO TO 1302
 1309 WRITE(6,156)
  156 FORMAT(' DKSI0')
 GO TO 1302
 1310 WRITE(6,157)
  157 FORMAT(' DETA0')
 GO TO 1302
 1311 WRITE(6,158)
  158 FORMAT(' DH ')
 GO TO 1302
 1312 JI = IABS(IPTYPE(I))
 IF (IPTYPE(I).LT.3400.AND.IPTYPE(I).GT.10) IBS=IBS+1
! CHANGE 2006-03-25 FOR FOURIER COEFFICIENTS.
 IF (IPTYPE(I).GT.16000.AND.IPTYPE(I).LT.97000) IBF=IBF+1
 IF (IPTYPE(I).GT.10000) IB = IB+1
 IF (IPTYPE(I).LT.0) IBT=IBT+1
 IBA=IBA+1
 IPKIND(IBA) = JI
 1302 CONTINUE
!
 IF (IB.GT.0.AND.IBT.EQ.0) WRITE(6,160) (IPKIND(I),I=1,IB)
 IF (IBS.GT.0.AND.IBT.EQ.0) WRITE(6,159) (IPKIND(I),I=1,IB)
 IF (IB.GT.0.AND.IBT.GT.0.AND.IBS.EQ.0) WRITE(6,161) (IPKIND(I),I=1,IB+IBT)
 IF (IB.GT.0.AND.IBT.GT.0.AND.IBS.GT.0) WRITE(6,162) (IPKIND(I),I=1,IB+IBT+IBS)
 IF (IBF.GT.0) WRITE(*,163)(IPKIND(I),I=1,IBF)
! HERE IS MISSING POSSIBILITY FOR OTHER PARAM.
  159 FORMAT(' SCALE FACTOR ')
  160 FORMAT(' BIAS-PARAMETERS',/,(9I8))
  161 FORMAT(' BIAS AND TILT PARAMETERS',/,(9I8))
  162 FORMAT(' BIAS,TILT AND SCALE PARAMETERS',/,5(9I8,/))
  163 FORMAT(' FOURIER COEFFICIENTS ',/,(9I8))
!
 RETURN
END SUBROUTINE WRPAR

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE CXPARM(SINLAP,COSLAP,RLONGP,HP,IKA)
! PROGRAMMED FEB. 1985 BY C.C.TSCHERNING, LATEST UPDATE: 02 APR 1998.
! THE SUBROUTINE COMPUTES THE CONTRIBUTIONS FROM OBSERVATIONS WHICH
! ONLY DEPEND ON PARAMETERS. THE CONTRIBUTIONS ARE ACCUMULATED IN CX.
!
! CURRENTLY 3 DATATYPES ARE ACCEPTED WITH DATA KIND CODES 6, 7 AND 9.
!   ELLIPSOIDAL HEIGHT DIFFERENCE (6)
!   DIFFERENCE BETWEEN LATITUDE AND LONGITUDE (*COS(LATITUDE)) (7)
!   SATELLITE ALTIMETRY CROSS-OVER DIFFERENCES (9).
! THE OCCURRENCE OF THE LAST DATA TYPE IS THE REASON THAT THE FACTOR
! II CHANGES FROM PLUS TO MINUS 1, WHEN THE SECOND PARAMETER IS FOUND.
!
```



```
 USE m_params, ONLY : NCOFF,NROOT,NNSU,NCX,NEQIV,NIPT,NIPCAT
 USE m_data,   ONLY : OBS
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_data,        ONLY : IPTYPE,ITIME,ITIME0,NPARM,NPARM1
 USE m_geocol_data, ONLY : SFACT,FPERIO,IPACAT,ITROLD,ICODE,MAXPAR,MP,&
                           IPA,NCXLAS,COFF
 USE m_geocol_data, ONLY : INUMR,NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,&
                           LTERM,LTERMA,LTERMO,LOUTC,LNTRAN=>LTRAN,LNERNO,LK30,L
K31,&
                           LWRSOL,LSTOP,LK2EQ4,LNUOUT
 USE m_geocol_data, ONLY : C20IN,G1,G2,CM3,CMM2,CM1

IMPLICIT NONE

!REAL(KIND=4) :: COFF

 REAL(KIND=8) :: PW,TU,HP,SINLAP,COSLAP,RLONGP,APARM

 INTEGER ::      IB,IER,IKP,IKA,IKK,I,ITP,II,J,IK,IGG,NEWCX,K,IK1

 LOGICAL ::      LSAME,LREPEC,LONECO,LTEST

 REAL(KIND=8), DIMENSION(320) :: U
 REAL(KIND=8), DIMENSION(NCX) :: CX

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT

!COMMON /OUTC/INUMR(12),NO1,K2,K3,K3P2,K4,IU,K21,IU1,IANG,LPUNCH,LTERMA,LTERMO,L
TERM,LOUTC,LNTRAN,LNERNO,LK30,LK31,LWRSOL,LSTOP,LK2EQ4,LNUOUT

!COMMON /CPARM/SFACT(NIPCAT),FPERIO(10,2),IPTYPE(NIPT),IPACAT(3*NIPT),ITIME(NIPC
AT),ITIME0(NIPT),ITROLD,ICODE(10),NPARM,NPARM1,MAXPAR,MP,IPA,NCXLAS
! NCXLAS POINTS AT THE LAST BLOCK OF UNIT 2 IN CORE (CX).1992.12.18.

!COMMON /GPOTC3/COFF(NCOFF)

 IB = 2
 IER = K2
 LTEST=.FALSE.
 IF (LK30) IB = 3
 NPARM1 = NPARM+1
 IKP = IKA
 LREPEC = IKP.EQ.5.OR.IKP.EQ.7.OR.IKP.GT.25
 LONECO = .NOT.LREPEC
 IF (IKP.GT.25) IKP=IKP-10
! IKK POINTS AT A FICTIVE LAST COLUMN WITH LENGTH MAXPAR. THE
! CONTRIBUTION TO THE RIGHT HAND SIDE IS STORED IN THE LAST BLOCK
! OF UNIT 2, BECAUSE WE DO NOT YET KNOW THE NUMBER OF PARAMETERS.
 IKK = MAXPAR*(MAXPAR+1)/2
!
   10 DO 21 I = 1, NPARM
 ITP = IPTYPE(I)
 II = 1
 LSAME=LF
 J = 0
   12 J = J+1
 LSAME=LSAME.OR.IPACAT(IPA+J).EQ.ITP
 IF ((.NOT.LSAME).AND.J.LT.MP) GO TO 12
! FOR CROSS-OVER DIFFERENCES THE CONTRIBITION IS NEGATIVE.
 IF (IKP.EQ.9 .AND. J.EQ.2) II = -1
 U(I)=D0
   21 IF (LSAME) U(I) = APARM(COSLAP,SINLAP,RLONGP,HP,IKP,ITP,0)*II
!
 U(NPARM1) = OBS(IB)
 PW = OBS(IER)**(-2)
!
 IK = -1
 READ(2,REC=1)CX
```

```
 IF (LTEST) THEN
  WRITE(*,*)' READH 2 ',(CX(IGG),IGG=1,6)
 END IF
 NCXLAS=1
 DO 41 I = 1, NPARM1
  IF (I.EQ.NPARM1) IK=IKK-1
  DO 41 K = 1, I
   IK = IK+1
   TU = U(I)*U(K)*PW
   IF (ABS(TU).GT.-1.0D-10) THEN
! BLOCKING OF CX IMPLEMENTED 1992.12.18 BY CCT.
    NEWCX = IK/NCX+1
    IK1 = MOD(IK,NCX)+1
    IF (NEWCX.NE.NCXLAS) THEN
     IF (NCXLAS.GT.0) THEN
      WRITE(2,REC=NCXLAS)CX
      IF (LTEST) THEN
      WRITE(6,*)' DUNIT 2, BLOCK ',NCXLAS,' WRITTEN'
      WRITE(*,*)(CX(IGG),IGG=1,6)
      END IF
     END IF
     READ(2,REC=NEWCX)CX
     IF (LTEST) THEN
     WRITE(*,*)' EUNIT 2,READ BLOCK ',NEWCX
     WRITE(*,*)(CX(IGG),IGG=1,6)
     END IF
     NCXLAS=NEWCX
    END IF
   END IF
   CX(IK1) = CX(IK1)+U(I)*U(K)*PW
   41  CONTINUE
  WRITE(2,REC=NEWCX)CX
  IF (LTEST) THEN
  WRITE(6,*)' FUNIT 2, BLOCK ',NEWCX,' WRITTEN'
  WRITE(*,*)(CX(IGG),IGG=IK1-3,IK1)
  END IF
!
 IF (LONECO) GO TO 69
!
 IB = IB+10
 IER = IER+10
 IKP = 4
 LONECO = LT
 GO TO 10
!
   69 RETURN
END SUBROUTINE CXPARM

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE MEAN1(LOC_FILTER,LOC_NFILTE,SAZP,CAZP,LFILTE,LGRID,LINTER)
! PROGRAMMED 1992.12.11 BY CCT. LAST UPDATE:  2013-09-10 BY CCT.
 USE m_geocol_data, ONLY : HQD,RLATD,SINLATD,COSLATD,RLONGD,SINLOND,COSLOND
 USE m_data,        ONLY : LDIF
IMPLICIT NONE

 REAL(KIND=8) :: SAZP,CAZP,SFILT,AZP,DEGRAD

 REAL(KIND=8), DIMENSION(11) :: LOC_FILTER

 LOGICAL ::      LFILTE,LGRID,LINTER,LTEST

 INTEGER ::      LOC_NFILTE,I
  LTEST=.FALSE.
  IF (LTEST) write(*,*)'12871',lfilte
 IF (.NOT.LFILTE) THEN
! IF 1-D MEANS ARE USED, IT IS HERE POSSIBLE TO INPUT UP TO 5 WEIGHTS
! WITH SUM EQUAL TO NUMBER OF WEIGHTS (LOC_NFILTE). ONLY ONE SET OF WEIGHTS
! MUST BE USED.
```



```
! CHANGE 1992.11.26 AND 2011-010-04 BY C.C.TSCHERNING.
 IF (LGRID)  WRITE(6,*)' 1-D MEANS NOT TO BE USED WITHOUT CAUTION'
 IF (LINTER) WRITE(*,*)' INPUT NUMBER OF FILTER FACTORS '
 READ(5,*)LOC_NFILTE
 IF (LOC_NFILTE.GT.11) THEN
  WRITE(*,*)' MAX LIMIT IS 11 '
  STOP
 END IF
 IF (LOC_NFILTE.LT.0) THEN
! LDIF IS TRUE, IF THE OBSERVATION IS THE DIFFERENCE BETWEEN VALUES IN TWO
! POINTS (GRACE). 2013-09-10.
  LDIF=.TRUE.
 ELSE
  LDIF=.FALSE.
  IF (LINTER) WRITE(*,*)' INPUT ',LOC_NFILTE,' FILTER FACTORS '
  READ(5,*)(LOC_FILTER(I),I=1,LOC_NFILTE)
  DO 2073, I=1,LOC_NFILTE
  2073 SFILT=SFILT+ABS(LOC_FILTER(I))
  IF (ABS(SFILT-LOC_NFILTE).GT.1.0D-5) WRITE(6,*) ' *** WARNING35 *** FILTER MUS
T SUM TO LOC_NFILTE.'
 END IF
 LFILTE=.TRUE.
 END IF
 IF (LDIF) THEN
  WRITE(*,*)' COORDINATES OF SECOND POINT MUST BE INPUT AS SECOND RECORD. '
 ELSE
 IF (LGRID)  THEN
 IF (LINTER)WRITE(6,*)' INPUT AZIMUTH IN DEGREES '
 READ(5,*) AZP
 DEGRAD=3.1415926535D0/180.0D0
 SAZP = SIN(AZP*DEGRAD)
 CAZP = COS(AZP*DEGRAD)
 END IF
 END IF
!
 RETURN
END SUBROUTINE MEAN1

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE HEAD(IKP,LONECO,PWO,LSATP)
! PROGRAMMED APR 1974 BY C.C.TSCHERNING, LAST UPDATE JAN 2005 BY CCT.
! OUTPUT OF HEADINGS AND INITIALIZATION OF THE FOLLOWING VARIABLES:
! IA,IB,IP,IT,I1,IA1,IB1,IP1,IT1,I21,I31,ICI,IC11 (ALL SUBSCRIPTS OF
! DIFFERENT  QUANTITIES), K2 - K4 (SUBSCRIPT BOUNDARIES FOR OUTPUT
! QUANTITIES), K1 = UPPER LIMIT FOR QUANTITIES READ INTO OBS.

 USE m_data,        ONLY : LC1,LC2,LCREF
 USE m_geocol_data, ONLY : INUMR,NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,&
                           LTERM,LTERMA,LTERMO,LOUTC,LTRAN,LNERNO,LK30,LK31,&
                           LWRSOL,LSTOP,LK2EQ4,LNUOUT
 USE m_data,        ONLY : IA,IB,IH,IP,IT,IA1,IB1,IP1,IT1,IC1,IC11,K1,&
                           IOBS1, IOBS2,ITE,ITE1,IITE,IITE1,IIP,IIP1,IIE,&
                           IIE1,INO,LPOT0=>LPOT,LKM,LTERRC,LPOTIN

IMPLICIT NONE

 LOGICAL ::      LONECO, LPOT1,                         &
                 LF,LERNO,       LSATP,LCOD,LSWI

 INTEGER ::      IKP,KADD,IREL,KM10,KMR,IC2

 REAL(KIND=8) :: PWO

!COMMON /OUTC/INUMR(12),NO1,K2,K3,K2P3,K4,IU,K21,IU1,IANG,LPUNCH,LTERMA,LTERMO,L
TERM,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,I
ITE,IITE1,IIP,IIP1,IIE,IIE1,INO,LPOT0,LKM,LTERRC,LPOTIN
```

```
 LF = .FALSE.
 LPOT1 = LPOT0
 LCOD = IKP.GT.5 .AND. IKP.LT.10
 IF (LCOD) LPOT1 = LF
 LERNO = .NOT.LNERNO
 LSWI=IOBS2.GT.0.AND.(IABS(IOBS2-IOBS1).EQ.1)
! THIS INDICATES THAT THE ORDER OF THE OBS ARE INVERTED.
 IT=11
! K1 POINTS AT THE LAST ELEMENT IN THE OBSERVATION RECORD.
 K1 = 0
 IIE=0
 IIE1=0
 IIP=0
 IIP1=0
 IITE=0
 IITE1=0
 LPOTIN=LF
 LTERRC=LF
 IF (IH.NE.0) K1=1
 KADD=1
! IREL IS THE POSITION OF THE LAST NON-DATA ELEMENT IN AN INPUT RECORD.
! THE INITIAL VALUE IS 2, BECAUSE LATITUDE AND LONGITUDE MUST BE PRESENT.
 IREL=2
 IF (INO.NE.0)IREL=IREL+1
!
! THIS IS IF THERE IS ONLY ONE DATA-ELEMENT.
 IF (IOBS2.EQ.0) GO TO 2303
 KADD=2
 KM10=MOD(IOBS2,10)
 KMR=(IOBS2-KM10)/10
 IF (KM10.EQ.0.AND.IH.EQ.0) KM10=IREL
 IF (KM10.EQ.0.AND.IH.NE.0) KM10=IH
 IOBS2=KM10-IREL
 IF (LERNO) IIE1=IOBS2+1
 IF (LSWI.AND.LERNO)IIE1=IOBS2+2
 IIP1=MOD(KMR,10)
 IITE1=(KMR-IIP1)/10
 IF (IITE1.NE.0) IITE1=IITE1+IOBS2
 IF (KM10.EQ.IH.OR.KM10.EQ.IREL) IOBS2=0
!
 2303 IF (IOBS1.EQ.0) GO TO 2304
! IF IOBS1 .GT. 10, THEN THE INPUT RECORD INCLUDES POSSIBLY
! CONTRIBUTION FROM POTENTIAL COEFFICIENTS AND/OR TERRAIN.
! CHANGE 2006-08-20.
 IF (IOBS1.GT.10) THEN
  KM10=MOD(IOBS1,10)
 ELSE
  KM10=IOBS1
 END IF
 IF (KM10.EQ.0.AND.IH.EQ.0) KM10=IREL+1
 IF (KM10.EQ.0.AND.IH.NE.0) KM10=IH
! IF THERE IS NO OBSERVATION IN THE INPUT RECORD, WE SUPPOSE THAT
! THE POSITION OF ALL OTHER DATA ELEMENTS IS COUNTED RELATIVE TO
! THE HEIGHT OR THE LONGITUDE.
 KMR=(IOBS1-KM10)/10
 IOBS1=KM10-IREL
! HERE CHANGE NEEDED -----------------------------------.
 IIP=MOD(KMR,10)
 IITE=(KMR-IIP)/10
 IF (IITE.NE.0) IITE=IITE+IOBS1
 IF (LERNO) IIE=IOBS1+1
 IF (LSWI)IIE=IOBS2+1
 IF (IIP.NE.0) IIP=IIP+IOBS1
 IF (KM10.EQ.IH.OR.KM10.EQ.IREL) IOBS1=0
!
! LTERRC AND LPOTIN ARE TRUE, IF TERRAIN CONTRIBUTION, POTENTIAL-
! COEFFICIENT CONTRIBUTION ARE INPUT AND NOT COMPUTED BY GEOCOL.
! IITE, IIP ARE THE SUBSCRIPTS IN THE INPUT ARRAY OBS.
```



```
 LTERRC=IITE.NE.0
 LPOTIN=IIP.NE.0
 IF ((IH.NE.0.AND.KM10.NE.IH).OR.(IH.EQ.0.AND.KM10.NE.IREL)) K1=K1+KADD
! CHANGE DEC 1986 TO PERMIT NO OBSERVATION IN INPUT RECORD.
 IF (LTERRC)K1=K1+KADD
 IF (LPOTIN)K1=K1+KADD
!
 2304 IF (IKP.GT.26) GO TO 2003
 GO TO (2008,2009,2010,2011,2012,2301,2302,2003,2003,2003,2008,&
 2312,2009,2314,2315,2010,2011,2318,2319,2320,2321,2322,2323,&
 2324,2325,2003),IKP
 2008 IF (LSATP) THEN
  WRITE(6,2204)
 2204 FORMAT('0   NO      LATITUDE     LONGITUDE     H  T M**2/S**2 ')
 ELSE
  WRITE(6,204)
  204 FORMAT('0   NO      LATITUDE     LONGITUDE     H    ZETA (M)')
 END IF
 GO TO 2013
 2009 WRITE(6,205)
  205 FORMAT('0   NO      LATITUDE     LONGITUDE     H   DELTA G (',&
 'MGAL)')
 GO TO 2013
 2010 IF (LSATP) WRITE(6,306)
 IF (.NOT.LSATP) WRITE(6,206)
  206 FORMAT(/,'  NO      LATITUDE     LONGITUDE     H   KSI (ARCSEC)')
  306 FORMAT(/,'  NO      LATITUDE     LONGITUDE     H    TY  MGAL  ')
 GO TO 2013
 2011 IF (LSATP) WRITE(6,307)
 IF (.NOT.LSATP) WRITE(6,207)
  207 FORMAT('0   NO      LATITUDE     LONGITUDE     H   ETA (ARCSEC)')
  307 FORMAT(/,'  NO      LATITUDE     LONGITUDE     H   TX  MGAL    ')
 GO TO 2013
 2012 WRITE(6,208)
  208 FORMAT('0   NO      LATITUDE     LONGITUDE     H    KSI/ETA (',&
 'ARCSEC)')
 GOTO 2013
 2003 WRITE(6,200)IKP
  200 FORMAT('0   NO      LATITUDE     LONGITUDE     H KIND=',I3)
 GO TO 2013
 2301 WRITE(6,201)
  201 FORMAT('0   NO      LATITUDE     LONGITUDE     H    HEIGHT DIF.')
 GO TO 2013
 2302 WRITE(6,202)
  202 FORMAT('0   NO      LATITUDE     LONGITUDE     H    LAT.,LONG*',&
 'COS(LAT) DIFF.')
 GO TO 2013
 2312 WRITE(6,312)
  312 FORMAT(//'   NO      LATITUDE     LONGITUDE     H    GRA.DIST.',&
 ' MGAL')
 GO TO 2013
 2314 WRITE(6,314)
  314 FORMAT(//'   NO      LATITUDE     LONGITUDE     H    D(DELG)/DZ',&
 ' EU')
 GO TO 2013
 2315 WRITE(6,315)
  315 FORMAT(//'   NO      LATITUDE     LONGITUDE     H    TZZ   EU  ')
 GO TO 2013
! COORDINATE SYSTEM: X - EAST, Y-NORTH, Z-UP.
 2318 WRITE(6,318)
  318 FORMAT(//'   NO      LATITUDE     LONGITUDE     H    D(DELG)/DX',&
 ' EU')
 GO TO 2013
 2319 WRITE(6,319)
  319 FORMAT(//'   NO      LATITUDE     LONGITUDE     H    D(DELG)/DY',&
 ' EU')
 GO TO 2013
 2320 WRITE(6,320)
  320 FORMAT(//'   NO      LATITUDE     LONGITUDE     H    TYZ   EU  ')
```

```
 GO TO 2013
 2321 WRITE(6,321)
  321 FORMAT(//'   NO      LATITUDE     LONGITUDE     H    TXZ   EU  ')
 GO TO 2013
 2322 WRITE(6,322)
  322 FORMAT(//'   NO      LATITUDE     LONGITUDE     H    TYY   EU  ')
 GO TO 2013
 2323 WRITE(6,323)
! 323 FORMAT(//'   NO      LATITUDE     LONGITUDE     H    TXY     ',&
  323 FORMAT(//'   NO      LATITUDE     LONGITUDE     H    TXY (*2)',&
 ' EU')
 GO TO 2013
 2324 WRITE(6,324)
  324 FORMAT(//'   NO      LATITUDE     LONGITUDE     H    TXX  EU ')
 GO TO 2013
 2325 WRITE(6,325)
  325 FORMAT(//'   NO      LATITUDE     LONGITUDE     H    TYY-TXX ',&
 ' EU')
!
 2013 GO TO (2018,2019,2020,2021,2020,2020),IANG
 2018 WRITE(6,209)
  209 FORMAT('             D  M  S       D  M  S     M')
 GO TO 2022
 2019 WRITE(6,210)
  210 FORMAT('                D  M          D  M     M')
 GO TO 2022
 2020 WRITE(6,211)
  211 FORMAT('            DEGREES       DEGREES      M')
 GO TO 2022
 2021 WRITE(6,212)
  212 FORMAT('            GRADES        GRADES       M')
!
 2022 IF (LKM) WRITE(6,213)
  213 FORMAT('+',37X,'K')
!
!     WRITE(*,*)' LCOD, PWO= ',LCOD,PWO
 IF ((.NOT.LCOD).AND. ABS(PWO).GT.1.0D-6) WRITE(6,267)PWO
  267 FORMAT('+',45X,' STAN.DEV.= ',F11.6)
!
! WE NOW COMPUTE THE SUBSCRIPT OF THE DIFFERENT QUANTITIES, WHICH WILL
! BE STORED FOR LATER OUTPUT IN THE ARRAY OBS. THE DIFFERENCE BETWEEN
! THE OBSERVATION GIVEN IN THE ORIGINAL AND THE NEW REF.SYSTEM IN
! OBS(IT), THE CONTRIBUTION TO THE REF.POT. FROM THE HARMONIC EXPANSION
! IN OBS(IP), THE CONTRIBUTION FROM COLL.I IN OBS(IC1) AND FROM COLL.II
! IN OBS(IC2).
 IC1 = 11
 IF (LTRAN) GO TO 2105
 IF (LTERRC) GO TO 2401
 IF(LPOT1) GO TO 2102
 IF (LC1) GO TO 2201
 IB = 4
 GO TO 2104
!
 2401 ITE=5
 IF (LPOT1) GO TO 2402
 IF (LC1) GO TO 2403
 IB=5
 WRITE(6,280)
  280 FORMAT(66X,' TERR')
 GO TO 2104
!
 2402 IP=6
 IF (LC1) GO TO 2404
 IB=7
 WRITE(6,281)
  281 FORMAT(22X,' TERR    POT    PRED')
 GO TO 2104
!
 2404 IC1=7
```



```
 IF (LC2) GO TO 2405
 IB=8
 WRITE(6,282)
  282 FORMAT(22X,' TERR    POT   COLL1   PRED')
 GO TO 2104
!
 2405 IC2=9
 IB=10
 WRITE(6,283)
  283 FORMAT(22X,' TERR    POT   COLL1  COLL2  PRED')
 GO TO 2104
!
 2403 IC1=6
 IF (LC2) GO TO 2406
 IB=7
 WRITE(6,284)
  284 FORMAT(22X,' TERR   COLL1  PRED')
 GO TO 2104
!
 2406 IC2=7
 IB=8
 WRITE(6,285)
  285 FORMAT(22X,' TERR   COLL1  COLL2  PRED')
 GO TO 2104
!
 2201 IC1 = 5
 IF (LC2) GO TO 2101
 IB=5
 WRITE(6,250)
  250 FORMAT(66X,' PRED')
 GO TO 2104
!
 2101 IB=7
 IC2=6
 WRITE(6,251)
  251 FORMAT(21X,' COLL1  COLL2   PRED')
 GO TO 2104
!
 2102 IP=5
 IF (LC1) GO TO 2202
 IB = 5
 WRITE(6,245)
  245 FORMAT(66X,' POT')
 GO TO 2104
!
 2202 IC1 = 6
 IF (LC2) GO TO 2103
 IB=7
 WRITE(6,252)
  252 FORMAT(22X,' POT    COLL   PRED')
 GO TO 2104
!
 2103 IC2=7
 IB=8
 WRITE(6,253)
  253 FORMAT(22X,' POT   COLL1  COLL2  PRED')
 GO TO 2104
!
 2105 IT=5
 IF(LTERRC)GO TO 2411
 IF(LPOT1) GO TO 2107
 IF (LC1) GO TO 2205
 IB = 5
 WRITE(6,246)
  246 FORMAT(66X,' TRA')
 2104 K3 = IB-4
 IU = IB
 GO TO 2110
!
```

```
 2411 ITE=6
 IF (LPOT1)GO TO 2412
 IF (LC1) GO TO 2413
 IB=6
 WRITE(6,290)
  290 FORMAT(21X,'   TRA   TERR  TERR-TRA')
 GO TO 2109
!
 2412 IP=7
 IF (LC1) GO TO 2414
 IB=8
 WRITE(6,291)
  291 FORMAT(21X,'   TRA   TERR    POT   PRED  PRED-TRA')
 GO TO 2109
!
 2414 IC1=8
 IF (LC2) GO TO 2415
 IB=9
 WRITE(6,292)
  292 FORMAT(21X,'   TRA   TERR    POT   COLL1   PRED  PRED-TRA')
 GO TO 2109
!
 2415 IC2=9
 IB=10
 WRITE(6,293)
  293 FORMAT(21X,'   TRA   TERR    POT   COLL1  COLL2  PRED',&
 '  PRED-TRA')
 GO TO 2109
!
 2413 IC1=7
 IF (LC2) GO TO 2416
 IB=8
 WRITE(6,294)
  294 FORMAT(21X,'   TRA   TERR  COLL1  PRED  PRED-TRA')
 GO TO 2109
!
 2416 IC2=8
 IB=9
 WRITE(6,295)
  295 FORMAT(21X,'   TRA   TERR  COLL1  COLL2  PRED   PRED-TRA')
 GO TO 2109
!
 2205 IC1 = 6
 IF (LC2) GO TO 2106
 IB=6
 WRITE(6,254)
  254 FORMAT(21X,'  TRA    PRED PRED-TRA')
 GO TO 2109
!
 2106 IC2=7
 IB=8
 WRITE(6,255)
  255 FORMAT(21X,'  TRA   COLL1   COLL2  PRED  PRED-TRA')
 GO TO 2109
!
 2107 IP=6
 IF (LC1) GO TO 2208
 IB = 6
 WRITE(6,247)
  247 FORMAT(21X,'  TRA   POT   POT-TRA')
 GO TO 2109
!
 2208 IC1 = 7
 IF (LC2) GO TO 2108
 IB=8
 WRITE(6,256)
  256 FORMAT(21X,'  TRA    POT    COLL   PRED  PRED-TRA')
 GO TO 2109
!
```



```
 2108 IC2=8
 IB=9
 WRITE(6,257)
  257 FORMAT(21X,'  TRA    POT   COLL1  COLL2  PRED PRED-TRA')
 2109 K3=IB-3
 IU = IB+1
!
 2110 LK30 = K3.GT.0
 LK31 = K3.GT.1
! K3 IS THE NUMBER OF QUANTITIES IN THE SECOND OUTPUT SEQUENCE. IF
! IT IS LESS THAN OR EQUAL TO 1, THEN ONLY ONE LINE IS USED FOR
! OUTPUT. THIS IS REGISTERED BY LK31.
!
 IF (LC2) IA = IC2
 IF (.NOT.LCREF) IA = IC1
!
 IF (LOUTC) GO TO 2125
 IF(LERNO) GO TO 2112
 K2=1
 GO TO 2135
!
 2112 K2=2
 IF (LK31) WRITE(6,260)
  260 FORMAT('+  ERR')
 IF (.NOT.LK31) WRITE(6,270)
  270 FORMAT('+',44X,'  ERR')
 GO TO 2135
 2125 IF (LK30.OR.LERNO) GO TO 2127
 K2=2
 WRITE(6,271)
  271 FORMAT(44X,'  OBS')
 GO TO 2135
!
 2127 IF (LERNO) GO TO 2128
 K2=3
 IF (LK31) WRITE(6,262)
  262 FORMAT('+  OBS    DIF')
 IF (.NOT.LK31) WRITE(6,272)
  272 FORMAT('+',44X,'  OBS    DIF')
 GO TO 2135
!
 2128 IF (LK30) GO TO 2111
 K2 = 3
 IF (LK30) WRITE(6,258)
  258 FORMAT(1X,'  OBS     ERR')
 IF (.NOT.LK30) WRITE(6,278)
  278 FORMAT(44X,'  OBS     ERR')
 GO TO 2135
!
 2111 K2 = 4
 IF (LK31) WRITE(6,263)
  263 FORMAT('+  OBS    DIF    ERR')
 IF (.NOT.LK31) WRITE(6,273)
  273 FORMAT('+',44X,'  OBS    DIF    ERR')
!
 2135 K4 = K2
 K2P3=K2
 LK2EQ4=K2.EQ.4
 IF (LK2EQ4) K2P3=K2+K3
! LK2EQ4 IS TRUE, IF ONE OUTPUT SEQUENCE CAN BE WRITTEN AS AN
! UNBROKEN LINE. OTHERWISE IT IS WRITTEN IN TWO PARTS, SEE THE
! SUBROUTINE OUT.
!write(*,*)' 13395 K2,k3,K2P3,LK2EQ4 ',K2,k3,K2P3,LK2EQ4
!
 IT1 = IT
 IP1 = IP
 IF (LONECO) GO TO 2313
 IB1 = IB+10
 IU1 = IU+10
```

```
 IA1 = IA+10
 IT1 = IT+10
 ITE1=ITE+10
 IP1 = IP+10
 IC11 = IC1+10
 K4 = 2*K2-1
 2313 K21 = K2+10
!
 RETURN
END SUBROUTINE HEAD

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

FUNCTION GPOTDR(NMAX,ORDER,SU,SU8)
!
!  MODIFICATION OF JULY 1984 FOR INCLUSION IN GEOCOL-PROGRAM OF
!  GI REG.NO. 81013 AUTHOR -C.C.TSCHERNING, DANISH GEODETIC INSTITUTE
!                                      JULY 1981 IN ALGOL REF.(2)
!                          -C.C.GOAD, NOAA/NOS/NATIONAL GEODETIC SURVEY
!                                      MAY 1982 TRANSLATED TO FORTRAN
! LATEST MODIFICATION DEC 2008 BY CCT.
!
! REFERENCES:
! (1) TSCHERNING, C.C.:ON THE CHAIN-RULE METHOD FOR COMPUTING
!     POTENTIAL DERIVATIVES. MANUSCRIPTA GEODAETICA, VOL.1,
!     PP. 125-141, 1976
!
! (2) TSCHERNING, C.C., AND PODER, K.: SOME APPLICATIONS OF CLENSHAW
!     SUMMATION, PRESENTED AT VIII SYMPOSIUM ON MATHEMATICAL GEODESY,
!     COMO, ITALY, SEPT 7-9, 1981
! AND REF.(J) IN MAIN PROGRAM.
!
!  THE PROCEDURE COMPUTES THE VALUE AND UP TO THE SECOND-ORDER
! DERIVATIVES OF THE POTENTIAL OF THE EARTH (W) OR OF ITS
! CORRESPONDING ANOMALOUS POTENTIAL(T).
!
!  THE POTENTIAL IS REPRESENTED BY A SERIES OF SOLID SPHERICAL
! HARMONICS, WITH UN-NORMALIZED OR QUASI-NORMALIZED COEFFICIENTS.
! THE CHAIN-RULE IS USED ALONG WITH THE CLENSHAW ALGORITHM.
! THE ARRAY C MUST HOLD THE COEFFICIENTS C(1)=C(1,0),C(2)=C(1,1),
! C(3)=S(1,1), ETC. UP TO C((N+1)**2-1=S(N,N).  C(0,0) IS STORED IN C0
! WHICH IMPLICITLY ACTS AS C(0) (SEE THE COMMON BLOCK CM).
!
!
! PARAMETERS:
!
! (A) INPUT VALUES:
!
! NMAX
!    THE ABSOLUTE VALUE OF NMAX IS EQUAL TO THE MAXIMAL DEGREE AND
!    ORDER OF THE SERIES. NEGATIVE NMAX INDICATES THAT THE COEFFICIENTS
!    ARE QUASI-NORMALIZED. IN THIS VERSION NMAX MUST NOT EXCEED 2190
!
! ORDER
!    ORDER OF DERIVATIVES
!    0 FOR POTENTIAL ONLY
!    1 FOR POTENTIAL AND FIRST DERIVATIVES
!    2 FOR POTENTIAL, FIRST DERIVATIVES, AND SECOND DERIVATIVES
!
! EUCL
!    COMMON BLOCK HOLDING EUCLIDIAN RECTANGULAR COORDINATES (X,Y,Z),
!    DISTANCE AND SQUARE OF DISTANCE TO Z-AXIS AND ORIGIN XY, XY2,
!    DISTO AND DIST2.
!
! C
!    C((ABS(NMAX)+1)**2)    ARRAY OF C'S AND S'S DESCRIBED ABOVE
!    CM3=GM
!    CM2=A THE SEMI-MAJOR AXIS OF THE REFERENCE ELLIPSOID
!    CM1=THE SQUARE OF THE ANGULAR VELOCITY (=0,WHEN DEALING WITH T)
```



```
!    C(1)=1.0D0 FOR W AND =0.0D0 FOR T
!
! C20IN
!    HOLDS C(2,0) AS A REAL.
!
!
!    ROOT(K)=SQRT(K), 0.LE.K.LE.2(ABS(N)+1)-1  WHEN NMAX.LT.0
!
!
! (B) RETURN VALUES:
!
! G1 AND G2
!    THE RESULT IS STORED IN G1 AND G2 AS FOLLOWS (CH, JULY 1989):
!
!    G1(1)=DW/DY, G1(2)=DW/DX, G1(3)=DW/DZ
!    G2(1,1)=DDW/DYY, G2(1,2)=G2(2,1)=DDW/DXDY,
!    G2(1,3)=G2(3,1)=DDW/DYDZ, G2(2,2)=DDW/DXX,
!    G2(2,3)=G2(3,2)=DDW/DXDZ AND G2(3,3)=DDW/DZZ
!    WHERE W MAY BE INTERCHANGED WITH T AND
!    VARIABLES X, Y, Z ARE THE CARTESIAN COORDINATES
!    IN A LOCAL (FIXED) FRAME WITH ORIGIN IN THE POINT
!    OF EVALUATION, X POSITIVE NORTH, Y POSITIVE EAST,
!    AND Z POSITIVE IN THE DIRECTION OF THE RADIUS
!    VECTOR, (CF. REF.(1),EQ (4) AND (5)).
!    THE VALUES OF W OR T WILL BE RETURNED IN GPOTDR.
!
! (C)  PASSED AND RETURNED VALUES:
!
! SU AND SU8
!    ARRAYS OF DIMENSION K*(N+1), WHERE K=2 FOR NO DERIVATIVES,
!    =6 FOR 0-TH AND FIRST DERIVATIVES, =10 FOR 0-TH, FIRST AND
!    SECOND DERIVATIVES.  HERE ARE STORED THE PARTIAL SUMS, CF.
!    REF.(2), EQ. (29), OF P(N,M)*(A/R)**(N+1-M)/P(M,M)*(C(N,M) OR
!    S(N,M))  FROM N=M TO N=N, AND THE DERIVATIVES OF THESE SUMS.
!    THIS MAKES IT UNNECESSARY TO RECALCULATE THESE QUANTITIES, IF
!    THE PROCEDURE IS CALLED SUBSEQUENTLY WITH THE SAME VALUE OF T
!    AND R, AND THE SAME ORDER.
!
!
! GPOTC1
!    VARIABLES IN COMMON BLOCK GPOTC1 KEEPS CERTAIN CONSTANTS
!    WHICH REPEATEDLY ARE USED, AND KEEPS TRACK OF WHETHER ALREADY
!    STORED VALUES (IN SU) CAN BE USED. IZ IS SET TO ZERO IN BLOCK
!    DATA.
!

 USE m_params, ONLY : NCOEFF,NROOT,NNSU
 USE m_geocol_data, ONLY : IZ,NMAXSV,I1,I2,I3,I4,I5,I6,I7,I8,I9,CFA
 USE m_geocol_data, ONLY : C => COFF
 USE m_data,        ONLY : OLDT,OLDR,LFIRST
 USE m_geocol_data, ONLY : X,Y,Z,XY,XY2,DISTO,DIST2
 USE m_geocol_data, ONLY : DZERO8 => DZERO,ROOT
 USE m_geocol_data, ONLY : C20IN,G1,G2,CM3,CM2 => CMM2,CM1

IMPLICIT NONE

 INTEGER :: J,NLAST,I,NMAX
!
! PARAMETERS GIVING THE SIZE OF THE ARRAYS HOLDING POTENTIAL
! COEFFICIENTS FOR MAX=2200.
!360  PARAMETER (NCOEFF=130322,NROOT=722,NNSU=3610)


 INTEGER ::      CAPN,ORDER,CAPN21,OLDORD,M2,M1,M0,M,KM,MAX2,IM,MPLUS1,&
                 IMM,NPM1

 LOGICAL ::      QUASI,DERIV1,DERIV2,POLE,FIRST,NEW,OLD,NPOLE,LINT

!REAL(KIND=4) :: C
```

```
 REAL(KIND=8) :: GPOTDR,P8,R8,S8,T8,U8,&
                 SL8,CL8,T28,S28,CL28,VC8,VS8,SQNPM18,&
                 VZC8,VZS8,VXC8,VXS8,VXXC8,VXXS8,VZZC8,VZZS8,&
                 VXZC8,VXZS8,CM8,SM8,SQNPM28,&
!                VXZC8,VXZS8,CM8,SM8,SQNM28,SQNPM28,SQ18,A18,B28,&
                 U08,AUX8,M218,M21U8,M21T8,M21U08,VZZM8,VXYM8,&
                 VXZM8,VYZM8,VXXM8,VYYM8,VXM8,VYM8,VZM8,VM8

 REAL*16 ::      DZERO,P,R,S,T,U,SQNPM1,CM,SM,SQNPM2,&
                 SL,CL,T2,S2,CL2,VC,VS,VZC,VZS,&
                 OM2,VXC,VXS,VXXC,VXXS,VZZC,VZZS,VXZC,VXZS,&
                 U0,AUX,M21,M21U,M21T,M21U0,VZZM,VXYM,VXZM,&
                 VYZM,VXXM,VYYM,VXM,VYM,VZM,VM

 REAL*16,      DIMENSION(NROOT) :: SMLP1,CMLP1
 REAL*16,      DIMENSION(NNSU)  :: SU

 REAL(KIND=8), DIMENSION(NROOT) :: SMLP18,CMLP18
 REAL(KIND=8), DIMENSION(NNSU)  :: SU8

!-----------------------------------------------------------------------------
!COMMON /EUCL/X,Y,Z,XY,XY2,DISTO,DIST2

!COMMON /SQROOT/DZERO8,ROOT(NROOT)

!COMMON /GPOTC1/OLDT,OLDR,CFA,IZ,OLDORD,I1,I2,I3,I4,I5,I6,I7,I8,I9,NMAXSV,FIRST,
HP9000

!COMMON /GPOTC0/C20IN,G1(3),G2(3,3),CM3,CM2,CM1

!COMMON /GPOTC3/C(NCOEFF)

!EQUIVALENCE(SML(1),SMLP1(2)),(CML(1),CMLP1(2))
!EQUIVALENCE(SML8(1),SMLP18(2)),(CML8(1),CMLP18(2))

 FIRST=LFIRST
 J=IABS(NMAX)
 NLAST=(J+1)**2+1
 DZERO=0.0D0
 IF(NMAXSV.NE.NMAX)FIRST=.FALSE.
!write(*,*)x,y,z
 NMAXSV=NMAX
 IF(FIRST)GO TO 100
 FIRST=.TRUE.
 OLDT=2.0D0
 I=J+1
 I1=I+1
 I2=I1+I
 I3=I2+I
 I4=I3+I
 I5=I4+I
 I6=I5+I
 I7=I6+I
 I8=I7+I
 I9=I8+I
  100 CAPN=NMAX
! DISTANCE FROM ROTATION AXIS
 P=XY
 P8=XY
! DISTANCE FROM ORIGIN
 R=DISTO
 R8=DISTO
! COSINE OF COLATITUDE
 T=Z/DISTO
 T8=Z/DISTO
! SINE OF COLATITUDE
 U=XY/DISTO
 U8=XY/DISTO
```



```
! SINE OF LONGITUDE
 SL=Y/XY
 SL8=Y/XY
! COSINE OF LONGITUDE
 CL=X/XY
 CL8=X/XY
 T2=T+T
 T28=T8+T8
! WE CHANGE FROM DOUBLE TO QUADRUPLE PRECISION IF ABS(LATITUDE) >
! 50 DEGREES AND ABS(NMAX) > 520. CHANGE 2008-07-01.
 LINT=ABS(NMAX).LT.520.OR.ABS(U).GT.0.65D0
 POLE= ABS(U).LE.1.0D-9
 NEW= ABS(OLDR-R).GT.1.0D-3 .OR.  ABS(OLDT-T).GT.1.0D-9 .OR. OLDORD.NE.ORDER .OR
. POLE
 OLD=.NOT.NEW
 NPOLE=.NOT.POLE
 IF(OLD)GO TO 200
 OLDR=R
 OLDT=T
 OLDORD=ORDER
  200 QUASI=.FALSE.
 IF(CAPN.LT.0)QUASI=.TRUE.
 IF(QUASI)CAPN=-CAPN
! COMPUTE AE/R
 S=CM2/R
 S8=CM2/R8
 S2=S**2
 S28=S8**2
 CMLP1(1)=1.0D0
 CMLP18(1)=1.0D0
!     CML(0)=1.0D0
 SMLP1(1)=0.0D0
 SMLP18(1)=0.0D0
!     SML(0)=0.0D0
 DERIV1=.FALSE.
 IF(ORDER.GT.0)DERIV1=.TRUE.
 DERIV2=.FALSE.
 IF(ORDER.GT.1) DERIV2=.TRUE.
!
!  SML(M) AND CML(M) ARE THE SINE AN COSINE OF M*LONGITUDE.
!  MODIFIED JAN 1989 AS PROPOSED BY C.GOAD.
!
!     SML(1)=SL
!     CML(1)=CL
 SMLP1(2)=SL
 CMLP1(2)=CL
 SMLP18(2)=SL8
 CMLP18(2)=CL8
 CL2=CL*2.0D0
 CL28=CL8*2.0D0
!
 M2=2
 M1=1
 M0=0
 DO M=2,CAPN
!      SML(M)=SML(M1)*CL2-SML(M0)
!      CML(M)=CML(M1)*CL2-CML(M0)
!      IF (LINT) THEN
   SMLP1(M+1)=SMLP1(M1+1)*CL2-SMLP1(M0+1)
   CMLP1(M+1)=CMLP1(M1+1)*CL2-CMLP1(M0+1)
!      ELSE
   SMLP18(M+1)=SMLP18(M1+1)*CL28-SMLP18(M0+1)
   CMLP18(M+1)=CMLP18(M1+1)*CL28-CMLP18(M0+1)
!      END IF
  M0=M1
  M1=M
 END DO
!
 CAPN21=CAPN+CAPN+1
```

```
 VM=0.0D0
 VXM=0.0D0
 VYM=0.0D0
 VZM=0.0D0
 VM8=0.0D0
 VXM8=0.0D0
 VYM8=0.0D0
 VZM8=0.0D0
 IF(DERIV2) THEN
  VXXM=0.0D0
  VYYM=0.0D0
  VZZM=0.0D0
  VXYM=0.0D0
  VXZM=0.0D0
  VYZM=0.0D0
  VXXM8=0.0D0
  VYYM8=0.0D0
  VZZM8=0.0D0
  VXYM8=0.0D0
  VXZM8=0.0D0
  VYZM8=0.0D0
 END IF
 KM=(CAPN+1)**2+1
 MAX2=CAPN21
!
! WE NOW USE THE CLENSHAW ALGORITHM, CF. REF.(2), EQ(8-13),
! MODIFIED IN AN OBVIOUS WAY FOLLOWING REF.(1).
!

!$OMP PARALLEL PRIVATE(IM)
!$OMP DO

 DO IM=IZ,CAPN
  CALL VMSUM(IM,CAPN,CAPN21,OLD,DERIV1,DERIV2,LINT,POLE,U,U8,T,T8,S,S8,S2,S28,C,
SU,SU8,ROOT,I1,I2,I3,I4,I5,I6,I7,I8,I9)
 END DO
!$OMP END DO NOWAIT
!$OMP END PARALLEL
!
 DO IMM=IZ,CAPN
  M=CAPN-IMM
  MPLUS1=M+1
  CM=CMLP1(MPLUS1)
  SM=SMLP1(MPLUS1)
  CM8=CMLP18(MPLUS1)
  SM8=SMLP18(MPLUS1)
  IF (LINT) THEN
   VC8=SU8(M+1)
   VS8=SU8(M+I1)
  ELSE
   VC=SU(M+1)
   VS=SU(M+I1)
  END IF
  IF(QUASI) THEN
   IF (LINT) THEN
    SQNPM18=ROOT(MAX2)
    SQNPM28=ROOT(MAX2+1)
   ELSE
    SQNPM1=ROOT(MAX2)
    SQNPM2=ROOT(MAX2+1)
   END IF
  END IF
  NPM1=MAX2
  MAX2=MAX2-2
  IF(DERIV1) THEN
   IF (LINT) THEN
    VXC8=SU8(M+I2)
    VXS8=SU8(M+I3)
    VZC8=SU8(M+I4)
```



```
    VZS8=SU8(M+I5)
   ELSE
    VXC=SU(M+I2)
    VXS=SU(M+I3)
    VZC=SU(M+I4)
    VZS=SU(M+I5)
   END IF
   IF(DERIV2) THEN
    IF (LINT) THEN
     VZZC8=SU8(M+I6)
     VZZS8=SU8(M+I7)
     VXZC8=SU8(M+I8)
     VXZS8=SU8(M+I9)
    ELSE
     VZZC=SU(M+I6)
     VZZS=SU(M+I7)
     VXZC=SU(M+I8)
     VXZS=SU(M+I9)
    END IF
   END IF
  END IF
!
! THE COMPUTATION OF DERIVATIVES IN DIRECTION OF POS LONGITUDE,Y,
! INVOLVES THE DIVISION BY U=COS(LATITUDE).  THIS DIVISION IS
! PERFORMED IMPLICITLY, BY STOPPING THE CLENSHAW SUMMATION AT M=1.
! THIS IS DONE BY PUTTING U0=1.0.  THIS TRICK PERMITS THE USE OF THE
! PROCEDURE AT POLES, EXCEPT FOR THE SECOND-ORDER DERIVATIVE WRT
! LONGITUDE.  BUT HERE WE USE THE VALIDITY OF THE LAPLACE EQUATION
! AND COMPUTE THE SECOND-ORDER DERIVATIVES WRT X AND Z
!
  IF (LINT) THEN
   U08=U8
   IF(M.EQ.0)U08=1.0D0
! REF.(2) EQ.(35)
   AUX8=NPM1
   IF(QUASI)AUX8=SQNPM18/SQNPM28
   M218=S8*AUX8
   M21U8=M218*U8
   IF(DERIV1) THEN
    M21T8=M218*T8
    M21U08=M218*U08
    IF (DERIV2) THEN
     M2=M*M
     VZZM8=VZZC8*CM8+VZZS8*SM8+M21U8*VZZM8
     IF(M.GT.0)VXYM8=M*(VXS8*CM8-VXC8*SM8)+M21U8*VXYM8-M21T8*VYM8
     VXZM8=VXZC8*CM8+VXZS8*SM8-M21T8*VZM8+M21U8*VXZM8
     VYZM8=(VZS8*CM8-VZC8*SM8)*M+M21U08*VYZM8
     IF (POLE) VXXM8=VXXC8*CM8+VXXS8*SM8+M218*(U8*(VXXM8-VM8)-T28*VXM8)
     IF (NPOLE) VYYM8=-(VC8*CM8+VS8*SM8)*M2+M21U08*VYYM8
    END IF
    VXM8=VXC8*CM8+VXS8*SM8-M21T8*VM8+M21U8*VXM8
    VYM8=M*(VS8*CM8-VC8*SM8)+M21U08*VYM8
    VZM8=VZC8*CM8+VZS8*SM8+M21U8*VZM8
   END IF
   VM8=VC8*CM8+VS8*SM8+M21U8*VM8
  ELSE
   U0=U
   IF(M.EQ.0)U0=1.0D0
! REF.(2) EQ.(35)
   AUX=NPM1
   IF(QUASI)AUX=SQNPM1/SQNPM2
   M21=S*AUX
   M21U=M21*U
   IF(DERIV1) THEN
    M21T=M21*T
    M21U0=M21*U0
    IF (DERIV2) THEN
     M2=M*M
     VZZM=VZZC*CM+VZZS*SM+M21U*VZZM
```

```
      IF(M.GT.0)VXYM=M*(VXS*CM-VXC*SM)+M21U*VXYM-M21T*VYM
      VXZM=VXZC*CM+VXZS*SM-M21T*VZM+M21U*VXZM
      VYZM=(VZS*CM-VZC*SM)*M+M21U0*VYZM
      IF(POLE) VXXM=VXXC*CM+VXXS*SM+M21*(U*(VXXM-VM)-T2*VXM)
      IF(NPOLE)VYYM=-(VC*CM+VS*SM)*M2+M21U0*VYYM
     END IF
     VXM=VXC*CM+VXS*SM-M21T*VM+M21U*VXM
     VYM=M*(VS*CM-VC*SM)+M21U0*VYM
     VZM=VZC*CM+VZS*SM+M21U*VZM
    END IF
    VM=VC*CM+VS*SM+M21U*VM
   END IF
  END DO
!
  IF (LINT) THEN
   M21=M218
   VM=VM8
   IF (DERIV1) THEN
    VXM=VXM8
    VYM=VYM8
    VZM=VZM8
    IF (DERIV2) THEN
     VXXM=VXXM8
     VYYM=VYYM8
     VZZM=VZZM8
     VXZM=VXZM8
! CORRECTION 2008-09-12.
     VYZM=VYZM8
     VXYM=VXYM8
    END IF
   END IF
  END IF
!
! NOW THE CONTRIBUTIONS FROM THE ROTATIONAL POTENTIAL ARE ADDED
!
! COMPUTE OMEGA**2
  IF (R.GT.6400000.0D0) THEN
!      IF (R.GT.6388000.0D0) THEN CHANGE 2008-12-18.
   OM2=0.0D0
  ELSE
   OM2=CM1
  END IF
! MODIFICATION JULY, 1984.
! COMPUTE GM/R
  S=CM3/R
  S8=CM3/R8
!
  IF (LINT) THEN
   GPOTDR=S8*VM8+OM2*P**2*0.5E0
  ELSE
   GPOTDR=S*VM+OM2*P**2*0.5E0
  END IF
  IF(.NOT.DERIV1) RETURN
! COMPUTE GM/R**2
  S=S/R
! CORRECTION JULY 1989: SUBSCRIPTS 1,2 IN G1 AND G2 INTERCHANGED.
  G1(2)=S*VXM-T*P*OM2
  G1(1)=S*VYM
  G1(3)=VZM*S+U**2*OM2*R
  IF(.NOT.DERIV2) RETURN
! COMPUTE GM/R**3
  S=S/R
! HERE THE LAPLACE EQUATION IS USED
  IF(.NOT.NPOLE) THEN
   VXXM=VXXM+VZM
   VYYM=-(VXXM+VZZM)
  ELSE
   VYYM=VZM+(VYYM-T*VXM)/U
   VXXM=-(VZZM+VYYM)
```



```
 END IF
 G2(2,2)=VXXM*S+OM2*T**2
 G2(1,2)=S*VXYM*M21
! CORRECTION 1988.08.23. BEFORE THIS THE M21 FACTOR WAS MISSING.
 G2(2,1)=G2(1,2)
 G2(2,3)=S*(VXZM-VXM)-U*T*OM2
 G2(3,2)=G2(2,3)
 G2(1,1)=VYYM*S+OM2
 G2(1,3)=S*(VYZM-VYM)
 G2(3,1)=G2(1,3)
 G2(3,3)=S*VZZM+U**2*OM2
! write(*,2111)s*VXZM,S*VXM,s*VYZM,s*VYM
!2111 format('SVXZM,SVXM,sVYZM,SVYM ',4d15.7)
 RETURN
END FUNCTION GPOTDR

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE VMSUM(IM,CAPN,CAPN21,OLD,DERIV1,DERIV2,LINT,POLE,U,U8,T,T8,S,S8,S2,S2
8,C,SU,SU8,ROOT,I1,I2,I3,I4,I5,I6,I7,I8,I9)
! THE SUBROUTINE COMPUTES PARTIAL SUMS OF SPHERICAL HARMONIC SERIES
! FOR FIXED ORDER. EXTRACTED FROM OLD VERSION OF GPOTDR 2008-07-08.
!   CALL PARAMETERS:
!   IM       = ORDER
!   CAPN     = MAXIMAL DEGREE AND ORDER. CAPN21=2*CAPN+1.
!   OLD      = TRUE, IF EARLIER CALL WAS FOR SAME LATITUDE AND R.
!   DERIV1, DERIV2, TRUE FOR 1. OR 2. DERIVATIVES.
!   POLE     = TRUE IF CLOSE TO POLE.
!   U,T      = SINE AND COS OF LATITUDE, WITH 8 IN DOUBLE PRECISION.
!   C        = SHPERICAL HARMONIC COEFFICIENTS
!   S        = A/R, S2 =S**2.
!   ROOT     = SQUARE ROOT TABLE.
!   I1-I9    = POINTERS IN ARRAY SU FOR EACH PARTIAL SUM.
!   RETURN PARAMETER: SU,SU8 PARTIAL SUMS FOR EACH ORDER.
!

 USE m_params, ONLY : NCOEFF,NROOT,NNSU

IMPLICIT NONE

 LOGICAL :: OLD,DERIV1,DERIV2,LINT,POLE

 INTEGER :: I5,I6,I7,I8,I9,CAPN,CAPN21,M2,M,KM,ITWO,IM,MPLUS1,&
            K,N21,I1,I2,I3,I4,NM1,N1,NPM1,N,IN,NM2,N2,KT

! PARAMETERS GIVING THE SIZE OF THE ARRAYS HOLDING POTENTIAL COEFFICIENTS.

 REAL(KIND=8) :: S8,T8,U8,S28,SQNM18,SQNPM18,VC8,VS8,&
                 VS18,VC18,VXS18,VXC18,VZC8,VZS8,&
                 VXXC18,VXXS18,CKZ8,CK1Z8,&
                 VZS18,VZC18,VXC8,VXS8,VXXC8,VXXS8,VZZC8,VZZS8,&
                 VZZC18,VZZS18,VXZC8,VXZS8,&
                 VXZC18,VXZS18,SQNM28,SQNPM28,SQ18,A18,B28,&
                 A1T8,A1U8,CK8,CK18,V28,D0

 REAL*16 ::      S,T,U,S2,SQNM1,SQNPM1,VC,VS,VS1,VC1,VXS1,VXC1,VZC,VZS,&
                 VXXC1,VXXS1,CKZ,CK1Z,&
                 VZS1,VZC1,VXC,VXS,VXXC,VXXS,VZZC,VZZS,VZZC1,VZZS1,VXZC,VXZS,&
                 VXZC1,VXZS1,SQNM2,SQNPM2,SQ1,A1,B2,A1T,A1U,CK,CK1,V2

 REAL(KIND=8), DIMENSION(NCOEFF) :: C
! CHANGE 2013-02-28.

 REAL(KIND=8), DIMENSION(NNSU)  :: SU8
 REAL(KIND=8), DIMENSION(NROOT) :: ROOT

 REAL*16,      DIMENSION(NNSU)  :: SU
```

```
  D0=0.0D0
  M=CAPN-IM
  KT=(CAPN+1)**2+1
  MPLUS1=M+1
  IF(M.EQ.0) THEN
   ITWO=1
   KT=KT-IM*2-1
  ELSE
   ITWO=2
   KT=KT-IM*2-2
  END IF
!      KM=KM-ITWO
  KM=KT
!      IF (KM.NE.KT) write(*,*)' kt,km ',ktikm
  K=KM
  N21=CAPN21
  VS=0.0D0
  VC=0.0D0
  VS1=0.0D0
  VC1=0.0D0
  VXS1=0.0D0
  VXC1=0.0D0
  VZS=0.0D0
  VZC=0.0D0
  VZS1=0.0D0
  VZC1=0.0D0
  VXC=0.0D0
  VXS=0.0D0
  VS8=0.0D0
  VC8=0.0D0
  VS18=0.0D0
  VC18=0.0D0
  VXS18=0.0D0
  VXC18=0.0D0
  VZS8=0.0D0
  VZC8=0.0D0
  VZS18=0.0D0
  VZC18=0.0D0
  VXC8=0.0D0
  VXS8=0.0D0
  IF(DERIV2) THEN
   VXXC=0.0D0
   VXXS=0.0D0
   VXXC1=0.0D0
   VXXS1=0.0D0
   VZZC=0.0D0
   VZZS=0.0D0
   VZZC1=0.0D0
   VZZS1=0.0D0
   VXZC=0.0D0
   VXZS=0.0D0
   VXZC1=0.0D0
   VXZS1=0.0D0
   VXXC8=0.0D0
   VXXS8=0.0D0
   VXXC18=0.0D0
   VXXS18=0.0D0
   VZZC8=0.0D0
   VZZS8=0.0D0
   VZZC18=0.0D0
   VZZS18=0.0D0
   VXZC8=0.0D0
   VXZS8=0.0D0
   VXZC18=0.0D0
   VXZS18=0.0D0
  END IF
  NM1=CAPN-M+2
  N1=CAPN+1
  NPM1=CAPN+M+2
```



```
  SQNM1=ROOT(NM1)
  SQNM18=SQNM1
  SQNPM1=ROOT(NPM1)
  SQNPM18=SQNPM1
  IF(DERIV2)M2=M*M
  IF(.NOT.OLD) THEN
  N=CAPN+1
  IF (.NOT.LINT) THEN
!
   DO IN=M,CAPN
    N=N-1
    NM2=NM1
    NM1=NM1-1
    NPM1=NPM1-1
! REF.(2) EQ.(40)
! REF.(2) EQ(30B)
    SQNM2=SQNM1
    SQNM1=ROOT(NM1)
    SQNPM2=SQNPM1
    SQNPM1=ROOT(NPM1)
    SQ1=SQNM1*SQNPM1
    A1=S*N21/SQ1
    B2=-S2*SQ1/(SQNM2*SQNPM2)
!
    A1T=A1*T
    A1U=A1*U
    N21=N21-2
    CK=C(K)
    CK1=C(K+1)
    K=K-N21
! REF.(2), EQ(33)
    V2=VC1
    VC1=VC
    VC=VC1*A1T+V2*B2+CK
    V2=VS1
    VS1=VS
    VS=VS1*A1T+V2*B2+CK1
    IF(DERIV1) THEN
     CKZ=CK*N1
     CK1Z=CK1*N1
! REF.(2),  EQ(10)
     V2=VXC1
     VXC1=VXC
     VXC=VXC1*A1T+VC1*A1U+V2*B2
     V2=VXS1
     VXS1=VXS
     VXS=VXS1*A1T+VS1*A1U+V2*B2
     V2=VZC1
     VZC1=VZC
     VZC=VZC1*A1T+V2*B2-CKZ
     V2=VZS1
     VZS1=VZS
     VZS=VZS1*A1T+V2*B2-CK1Z
     N1=N
     IF(DERIV2) THEN
      N2=N+2
! REF.(2), EQ(41)
      V2=VZZC1
      VZZC1=VZZC
      VZZC=VZZC1*A1T+V2*B2+N2*CKZ
      V2=VZZS1
      VZZS1=VZZS
      VZZS=VZZS1*A1T+V2*B2+N2*CK1Z
      IF(POLE) THEN
! REF.(2), EQ(12)      SECOND-ORDER DERIVATIVE WRT LATITUDE
       V2=VXXC1
       VXXC1=VXXC
       VXXC=A1T*(VXXC1-VC1)+(A1U+A1U)*VXC1+V2*B2
       V2=VXXS1
```

```
      VXXS1=VXXS
      VXXS=A1T*(VXXS1-VS1)+(A1U+A1U)*VXS1+V2*B2
     END IF
! REF.(2) EQ(10,40) DERIVATIVE WRT R AND LATITUDE
     V2=VXZC1
     VXZC1=VXZC
     VXZC=VXZC1*A1T+VZC1*A1U+V2*B2
     V2=VXZS1
     VXZS1=VXZS
     VXZS=VXZS1*A1T+VZS1*A1U+V2*B2
    END IF
   END IF
  END DO
!
 ELSE
  DO IN=M,CAPN
   N=N-1
   NM2=NM1
   NM1=NM1-1
   NPM1=NPM1-1
! REF.(2) EQ.(40)
! REF.(2) EQ(30B)
   SQNM28=SQNM18
   SQNM18=ROOT(NM1)
   SQNPM28=SQNPM18
   SQNPM18=ROOT(NPM1)
   SQ18=SQNM18*SQNPM18
   A18=S8*N21/SQ18
   B28=-S28*SQ18/(SQNM28*SQNPM28)
   A1T8=A18*T8
   A1U8=A18*U8
   N21=N21-2
   CK8=C(K)
   CK18=C(K+1)
   K=K-N21
! REF.(2), EQ(33)
   V28=VC18
   VC18=VC8
   VC8=VC18*A1T8+V28*B28+CK8
   V28=VS18
   VS18=VS8
   VS8=VS18*A1T8+V28*B28+CK18
   IF(DERIV1) THEN
    CKZ8=CK8*N1
    CK1Z8=CK18*N1
! REF.(2),  EQ(10)
    V28=VXC18
    VXC18=VXC8
    VXC8=VXC18*A1T8+VC18*A1U8+V28*B28
    V28=VXS18
    VXS18=VXS8
    VXS8=VXS18*A1T8+VS18*A1U8+V28*B28
    V28=VZC18
    VZC18=VZC8
    VZC8=VZC18*A1T8+V28*B28-CKZ8
    V28=VZS18
    VZS18=VZS8
    VZS8=VZS18*A1T8+V28*B28-CK1Z8
    N1=N
    IF(DERIV2) THEN
     N2=N+2
! REF.(2), EQ(41)
     V28=VZZC18
     VZZC18=VZZC8
     VZZC8=VZZC18*A1T8+V28*B28+N2*CKZ8
     V28=VZZS18
     VZZS18=VZZS8
     VZZS8=VZZS18*A1T8+V28*B28+N2*CK1Z8
     IF(POLE) THEN
```



```
! REF.(2), EQ(12)      SECOND-ORDER DERIVATIVE WRT LATITUDE
      V28=VXXC18
      VXXC18=VXXC8
      VXXC8=A1T8*(VXXC18-VC18)+(A1U8+A1U8)*VXC18+V28*B28
      V28=VXXS18
      VXXS18=VXXS8
      VXXS8=A1T8*(VXXS18-VS18)+(A1U8+A1U8)*VXS18+V28*B28
     END IF
! REF.(2) EQ(10,40) DERIVATIVE WRT R AND LATITUDE
     V28=VXZC18
     VXZC18=VXZC8
     VXZC8=VXZC18*A1T8+VZC18*A1U8+V28*B28
     V28=VXZS18
     VXZS18=VXZS8
     VXZS8=VXZS18*A1T8+VZS18*A1U8+V28*B28
    END IF
   END IF
  END DO
! END OF DO IN=M,CAPN FOR QUADRUPLE PRECISION.
!
  END IF
!
  IF (LINT) THEN
   SU8(M+1)=VC8
   SU8(M+I1)=VS8
  ELSE
   SU(M+1)=VC
   SU(M+I1)=VS
  END IF
  IF(DERIV1) THEN
   IF (LINT) THEN
    SU8(M+I2)=VXC8
    SU8(M+I3)=VXS8
    SU8(M+I4)=VZC8
    SU8(M+I5)=VZS8
   ELSE
    SU(M+I2)=VXC
    SU(M+I3)=VXS
    SU(M+I4)=VZC
    SU(M+I5)=VZS
   END IF
   IF(DERIV2) THEN
    IF (LINT) THEN
     SU8(M+I6)=VZZC8
     SU8(M+I7)=VZZS8
     SU8(M+I8)=VXZC8
     SU8(M+I9)=VXZS8
    ELSE
     SU(M+I6)=VZZC
     SU(M+I7)=VZZS
     SU(M+I8)=VXZC
     SU(M+I9)=VXZS
    END IF
   END IF
  END IF
 END IF
! END IF FOR OLD=.FALSE.
 RETURN
END SUBROUTINE VMSUM

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE LOADCS(DNAME,FMT,NMAX,LFMT,LBIN,LSKIPL)
! PROGRAMMED BY C.C.GOAD, NGS, 1981. MODIFIED JAN. 2013 BY CCT.

 USE m_params, ONLY : NCOFF,NROOT,NICC,NNSU
 USE m_geocol_data, ONLY : CFA
 USE m_geocol_data, ONLY : C => COFF
 USE m_data,        ONLY : OLDT,OLDR
```

```
 USE m_geocol_data, ONLY : DZERO,ROOT
 USE m_geocol_data, ONLY : C20IN,G1,G2,CM3,CM2 => CMM2,CM1

IMPLICIT NONE

 INTEGER ::            NLAST,NMAX,NSLINE,I,N21,NMT,NB,&
            NC,NB0,NB1,N,M,ITWO,NMC,J,K,       NBMAX

 LOGICAL :: LBIN,LFMT,      L386,LNZERO,LSKIPL,LCHARA

!REAL(KIND=4) :: C,C0,CNM,SNM
 REAL(KIND=8) ::   C0,CNM,SNM
! CHANGE 2013-02-28.

 REAL(KIND=8) :: SQ2,C00,CNMD,SNMD,S21,SQ2N1

 CHARACTER(LEN=6)   :: ALABEL
 CHARACTER(LEN=128) :: ALINE

 CHARACTER(LEN=128), DIMENSION(2) :: DNAME
 CHARACTER(LEN=128), DIMENSION(9) :: FMT

!-----------------------------------------------------------------------------
!COMMON /GPOTC0/C20IN,G1(3),G2(3,3),CM3,CM2,CM1

!COMMON /GPOTC3/C(NCOFF)

!COMMON /GPOTC1/OLDT,OLDR,CFA,IZ,IOLDOR,IPO(9),NMAXSV,FIRST,HP9000

!COMMON /SQROOT/DZERO,ROOT(NROOT)
! GPOTC1 IS ONLY USED HERE TO TRANSFER THE VALUE OF CFA TO GPOTDR.

 NLAST=(NMAX+1)**2
 L386=NLAST.GT.NCOFF
 CFA=1.0D0
 SQ2= SQRT(2.0D0)
!IF (LINT) CFA=1.0D12
 IF (LBIN.AND.(.NOT.L386)) OPEN(9,FILE=DNAME(1),STATUS='OLD',&
 FORM='UNFORMATTED')
!    *FORM='UNFORMATTED',RECL=8)
 IF (LBIN.AND.L386) OPEN(9,FILE=DNAME(1),STATUS='OLD',&
 FORM='UNFORMATTED')
 IF (LBIN.AND.L386) RETURN
!
 IF (L386) THEN
! WHEN THERE IS TOO LITTLE SPACE FOR  THE COEFFICIENTS, AND THEY
! ARE NOT YET ON BINARY FORM, THEN THEY ARE HERE REFORMATTED TO
! A BINARY FORMAT. CHANGE 1992,12.15 BY CCT.
 OPEN(39,FILE=DNAME(1),STATUS='OLD',FORM='FORMATTED')
! COEFF FILE MAY CONTAIN HEADING, WHICH MUST BE SKIPPED.
 IF (LSKIPL) THEN
 WRITE(*,*)' INPUT NUMBER OF LINES TO BE SKIPPED '
 READ(5,*)NSLINE
! ADDED 2006-09-13 TO BE ABLE TO READ GRACE COEFFICIENTS.
 LCHARA=.FALSE.
 IF (NSLINE.LT.0) THEN
  NSLINE=-NSLINE
  LCHARA=.TRUE.
  WRITE(*,*)' 6 CHARACTERS WILL BE SKIPPED '
 END IF
 IF (NSLINE.GT.0) THEN
 DO 386, I=1,NSLINE
 386  READ(39,'(A)')ALINE
 END IF
 END IF
!
 WRITE(6,*)' INPUT NAME OF OUTPUT FILE WITH BINARY COEFF. '
 READ(5,'(A)')DNAME(2)
 OPEN(9,FILE=DNAME(2),FORM='UNFORMATTED')
```



```
!
 LNZERO=.TRUE.
 DZERO=0.0D0
 N21=2*(NMAX+1)
 DO 50 I=1, N21
  50  ROOT(I)= SQRT(DFLOAT(I))

 NLAST=(NMAX+1)**2+1
!
 WRITE(6,102)
  102 FORMAT('  COEFFICIENTS UP TO N=5 ')
 NBMAX=NLAST/NCOFF
 IF (NBMAX*NCOFF.LT.NLAST) NBMAX=NBMAX+1
 NLAST=NLAST-1
 NMT=0
 NB=1
 NC=0
 NB0=0
 NB1=NCOFF
 WRITE(6,*)' BLOCKS USED FOR COEFF =',NBMAX
 C00=1.0D0
 DO 90 I=1,NCOFF
   90 C(I)=0.0E0
!
 1100 CONTINUE
 IF ((.NOT.LFMT).AND.NMAX.LT.201) READ(39,101) N,M,CNMD,SNMD
 IF ((.NOT.LFMT).AND.NMAX.GT.200) READ(39,3601) N,M,CNMD,SNMD
 IF (LFMT) THEN
  IF (LCHARA) THEN
 1101   READ(39,FMT)ALABEL,N,M,CNMD,SNMD
   IF (ALABEL.EQ.'GRDOTA') THEN
    WRITE(*,*)' DOT RECORD SKIPPED ',N,M
    GO TO 1101
   END IF
  ELSE
   READ(39,FMT)N,M,CNMD,SNMD
  END IF
 END IF
!
 IF (N .LT.5.AND.NB.EQ.1) WRITE(6,103)N,M,CNMD,SNMD
 IF (MOD(N,101).EQ.0.AND.M.EQ.N.AND.M.GT.5) WRITE(6,*)N,' DEG FINISHED READING '
 IF (N.GT.0) GOTO 1105
 LNZERO=.FALSE.
 C00 = CNMD
 C0 = C00
 IF (C00.LT.1.0D-8)THEN
 C00=1.0D0
 CNMD=C00
 END IF
 1105 CNMD=CNMD/C00
 SNMD=SNMD/C00
 IF (N.GT.NMAX .OR. M.GT.NMAX) GO TO 1100
 ITWO=2
 IF (M.EQ.0)ITWO=1
 NMC= (NMAX-M)*(NMAX-M+1)+(NMAX-N)*ITWO+1
 IF (NMC.LE.NB0.OR.NMC.GT.NB1) GO TO 300
 NMC=NMC-NB0
 S21= ROOT(2*N+1)
 IF (M.EQ.0) THEN
! STORE C20 IN DOUBLE PRECISION FOR LATER USE
 IF (N.EQ.2) THEN
 C20IN=CNMD*S21
 CNMD=0.0D0
 END IF
 CNM=CNMD*S21
 NC=NC+1
 ELSE
 CNM=CNMD*S21*SQ2
 SNM=SNMD*S21*SQ2
```

```
 NC=NC+2
 END IF
 C(NMC)=CNM
 NMT=NMT+1
 IF (M.NE.0) THEN
  NMT=NMT+1
  C(NMC+1)=SNM
!      WRITE(6,*)NMT,NMC,NC,N,M
 END IF
 300  IF ((N.EQ.NMAX.AND.M.EQ.NMAX).OR.NMT.GE.NCOFF) THEN
  IF (NMT.NE.NCOFF) WRITE(6,*)' ONLY ',NMT,' COEFF IN BLOCK ',NB
  IF (NB.EQ.NBMAX) C(NLAST-NB0)=0.0D0
  WRITE(9)(C(I),I=1,NCOFF)
  IF (NB.EQ.NBMAX) WRITE(9)C20IN
  NB0=NB1
  NB1=NB1+NCOFF
!      WRITE(6,*)' BLOCK ',NB,NB0,NB1
  REWIND(39)
  NB=NB+1
  NMT=0
  NC=0
! IF SOME COEFFICIENTS ARE MISSING, THEY ARE PUT TO ZERO HEREBY:
  DO I=1,NCOFF
   C(I)=0.0E0
  END DO
 END IF
 IF (NB.GT.NBMAX) GO TO 93
 IF (NB.EQ.NBMAX.AND.N .EQ. NMAX .AND. M .EQ. NMAX) GO TO 1200
 GO TO 1100
 1200 IF (NB.EQ.NBMAX) C(NLAST-NB0)=1.0D0
 WRITE(9)(C(I),I=1,NMT),C20IN
   93 IF (NC.LT.NLAST) WRITE(6,*)' ONLY ',NC,' COEFFICIENTS READ'
 CLOSE(39)
 LBIN=.TRUE.
 REWIND(9)
 RETURN
 END IF
!
 IF (.NOT.LBIN) OPEN(9,FILE=DNAME(1),STATUS='OLD',FORM='FORMATTED')
 IF (.NOT.LBIN) GO TO 106
! FOR A BINARY FILE WE SKIP THE FIRST 8 REALS.
 NLAST=(NMAX+1)**2
!IF (LINT) GO TO 9040
!     READ(9)(C(I),I=1,8) CHANGE 1998.07.06 CCT.
 DO I=1,NLAST
  READ(9)C(I)
 END DO
 C20IN=C(5)
 GO TO 200
!9040 READ(9)C20IN
!DO I=1,NLAST
! READ(9)IC(I)
!END DO
!GO TO 200
!
  106 WRITE(6,102)
 C00=1.0D0
!IF (LINT) IC(1)=0
  C(1)=C00
! COEFF FILE MAY CONTAIN HEADING, WHICH MUST BE SKIPPED.
 LCHARA=.FALSE.
 IF (LSKIPL) THEN
  WRITE(*,*)' INPUT NUMBER OF LINES TO BE SKIPPED '
  READ(5,*)NSLINE
! ADDED 2006-09-13 TO BE ABLE TO READ GRACE COEFFICIENTS.
  LCHARA=.FALSE.
  IF (NSLINE.LT.0) THEN
   NSLINE=-NSLINE
   LCHARA=.TRUE.
```



```
   WRITE(*,*)' 6 CHARACTERS WILL BE SKIPPED '
  END IF
  IF (NSLINE.GT.0) THEN
  DO 387, I=1,NSLINE
 387    READ(9,'(A)')ALINE
  END IF
  WRITE(*,*)NSLINE,' SKIPPED '
 END IF
!
  100 CONTINUE
 IF (LBIN) READ(9)N,M,CNMD,SNMD
! change 2005-07-26. Free format introduced.
!     IF ((.NOT.LBIN).AND.(.NOT.LFMT).AND.NMAX.LT.201) READ(9,101)N,M,&
!    *CNMD,SNMD
!     IF ((.NOT.LBIN).AND.(.NOT.LFMT).AND.NMAX.GT.200) READ(9,3601)
!    *N,M,CNMD,SNMD
 IF ((.NOT.LBIN).AND.(.NOT.LFMT)) READ(9,*)N,M,CNMD,SNMD
 IF ((.NOT.LBIN).AND.LFMT) THEN
  IF (LCHARA) THEN
 1102   READ(9,FMT)ALABEL,N,M,CNMD,SNMD
   IF (ALABEL.EQ.'GRDOTA') THEN
    WRITE(*,*)' DOT RECORD SKIPPED ',N,M
    GO TO 1102
   END IF
  ELSE
   READ(9,FMT)N,M,CNMD,SNMD
  END IF
 END IF
  101 FORMAT(2I4,2E16.9)
 CNM=CNMD
 SNM=SNMD
 3601 FORMAT(I3,1X,I3,2(1X,E19.12))
 IF (N .LT.5) WRITE(6,103)N,M,CNM,SNM
  103 FORMAT(I5,I4,2E17.9)
 IF (MOD(N,101).EQ.0.AND.M.EQ.N.AND.M.GT.5) WRITE(6,*)N,' DEG FINISHED READING '
 IF (N.GT.0) GO TO 105
 C00 = CNM
 C0 = C00
 IF (C00.EQ.0.0D0) C00=1.0D0
 C(1)=C0
!IF (LINT) IC(1)=0
 GO TO 100
  105 CNM=CNM/C00
 SNM=SNM/C00
 IF (N.GT.NMAX .OR. M.GT.NMAX) GO TO 100
 J=N*N
 SQ2N1= SQRT(2*N+1.0D0)
 IF (M.EQ.0) GO TO 104
 K =M+M
!IF (LINT) IC(J+K)=CNM*CFA*SQ2N1*SQ2
!IF (LINT) IC(J+K+1)=SNM*CFA*SQ2N1*SQ2
  C(J+K)=CNM
  C(J+K+1)=SNM
 IF (N .EQ. NMAX .AND. M .EQ. NMAX) GO TO 200
 GO TO 100
  104 C(J+1)=CNM
!IF (N.EQ.2.AND.LINT) IC(J+1)=0
! STORE C20 IN DOUBLE PRECISION FOR LATER USE
 IF (N.EQ.2) C20IN=CNMD*SQ2N1
!IF (LINT.AND.N.NE.2) IC(J+1)=CNM*CFA*SQ2N1
 GO TO 100
  200 CONTINUE
 IF (.NOT.L386) CLOSE(9)
 RETURN
END SUBROUTINE LOADCS

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE SETCM(CAPN,LBIN )
```

```
! PROGRAMMED BY C.C.GOAD, 1981. LAST MODIFIED SEP 1992 BY CCT.

 USE m_params, ONLY : NCOFF,NROOT,NNSU
 USE m_geocol_data, ONLY : C => COFF
 USE m_geocol_data, ONLY : DZERO,ROOT
 USE m_geocol_data, ONLY : C20IN,G1,G2,CM3,CM2 => CMM2,CM1

IMPLICIT NONE

 INTEGER CAPN,NLAST,N1,N21,I,MAXBL,NC20,NC,&
 NB0,NB1,NB,N,NNB0,N2,K,J,KJ2

 LOGICAL L386,LBIN
!
!REAL*4 C,C0
 REAL*8   C0
!REAL*4   C0
 REAL*8 GG,D,S21,SQ2,SMALLC,C5
!
!COMMON /SQROOT/DZERO,ROOT(NROOT)
!COMMON /GPOTC0/C20IN,G1(3),G2(3,3),CM3,CM2,CM1
!COMMON /GPOTC3/C(NCOFF)
!
! THIS ROUTINE SETS THE SQUARE ROOT TABLE IN COMMON
! SQROOT AND CREATES QUASI-NORMALIZED COEFFICIENTS FROM NORMALIZED
! COEFFICIENTS IN COMMON GPOTC0. (QUASI-NORMALIZED COEFFICIENTS MAY
! BE INPUT. THIS IS CHECKED USING C(2,0)).
!
! IF MORE SPACE IS NEEDED FOR THE COEFFICIENTS THAN AVAILABLE, THEN
! THEY MUST BE STORED ON DISK QUASI-NORMALIZED, BINARY, WITH C(0,0)
! AND C(2,0) ZERO AND C(2,0) IN DOUBLE PRECISION AS THE LAST COEFF.
 NLAST=(CAPN+1)**2+1
 L386=NLAST.GT.NCOFF
 N1=CAPN+1
 DZERO=0.0D0
 N21 = 2*(CAPN+1)
 DO 50 I=1, N21
   50 ROOT(I)= SQRT( FLOAT(I))
 G1(1)=0.0D0
 G1(2)=0.0D0
 G1(3)=0.0D0
!
!IF (LINT) RETURN
 IF (L386)THEN
 MAXBL=NLAST/NCOFF
 IF (MAXBL*NCOFF.NE.NLAST) MAXBL=MAXBL+1
 NLAST=NLAST-1
 NC20=NLAST-2-((MAXBL-1)*NCOFF)
 NC=0
 NB0=0
 NB1=NCOFF
 NB=0
 DO 151 N=1,NLAST
 NNB0=N-NB0
 IF (NNB0.EQ.NCOFF.OR.N.EQ.NLAST) THEN
 NB=NB+1
 READ(9)(C(I),I=1,NNB0)
 IF (N.EQ.NLAST) THEN
 READ(9)C20IN
 C5=C(NC20)
 IF (ABS(C5).GT.1.0D-8) WRITE(6,*)' C20, C5=',C20IN, C5
 END IF
 NB0=NB1
 NB1=NB1+NCOFF
 END IF
  151 CONTINUE
 RETURN
 END IF
 IF (ABS(C20IN+1.0827E-3).LT.1.0D-6.AND.LBIN) THEN
```



```
 WRITE(6,10)
   10 FORMAT(' QUASI-NORMALIZED COEFFICIENTS INPUT.')
 RETURN
 END IF
!
   90 SMALLC=1.0D0
 C0=C(1)
 C(1)=1.0D0
 IF(C0.NE.0.0D0)SMALLC=1.0D0/C0
 SQ2= SQRT(2.0D0)
!
 DO 200 N=1,CAPN
 N2=N+N
 S21 = ROOT(N2+1)
 K=N**2+1
! D IS THE QUASI-NORMALIZATION FACTOR FOR ZONAL TERMS
 D=SMALLC*S21
 C(K)=C(K)*D
! GG IS THE QUASI-NORMALIZATION FACTOR FOR NON-ZONAL TERMS
 GG=D*SQ2
 DO 100 J=1,N
 KJ2=J+J+K
 C(KJ2-1)=C(KJ2-1)*GG
 C(KJ2)=C(KJ2)*GG
  100 CONTINUE
  200 CONTINUE
 RETURN
END SUBROUTINE SETCM
! ------------------------------------------------------------
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!                                                                      C
!                       I F R A C                                      C
!                                                                      C
!  SUBROUTINE GIVING TRUE INTEGER PART OF REAL REAL    C
!                                                                      C
!  RF, JUNE 1983                                                       C
!                                                                      C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
INTEGER FUNCTION IFRAC(R)

IMPLICIT NONE

 REAL*8 R
 IF (R.LT.0.0D0) GO TO 1
   IFRAC = R
 RETURN
 1      IFRAC = R - 0.999999999D0
 RETURN
END FUNCTION IFRAC

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE ICMEAN(BSIZE,STEP,NSTEP,COSST,SINST,COSLAT,SINLAT,LEQANG,LMEA1)
! PROGRAMMED BY C.C.TSCHERNING, GEODETIC INSTITUTE, NOV 1985.
! THE SUBROUTINE INITIALIZES STEP VARIABLES FOR MEAN VALUE
! COMPUTATION. CHANGED 1996.10.08 BY CCT.
! LEQANG IS TRUE, WHEN WE DEAL WITH EQUAL-ANGULAR BLOCK AVERAGES.
! LMEA1 IS TRUE WHEN WE HAVE 1-D MEANS.
IMPLICIT NONE

 LOGICAL ::      LEQANG,LMEA1,LTEST

 REAL(KIND=8) :: BSIZE,STEP,COSST,SINST,COSLAT,SINLAT,BSIZEA

 INTEGER ::      NSTEP,NSTEP1

 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))
! CORRECTION 1995.11.21 BY CCT.
   10 IF (LMEA1) THEN
! FOR 1-D MEANS, THE POINTS ARE SUPPOSED TO BE DISTRIBUTED EQUIDISTANTLY
! ON THE INTERVAL OF SIZE BSIZE. FOR 2-D MEANS THEY ARE DISTRIBUTED
! 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 SUBROUTINE ICMEAN

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

FUNCTION COMEAN(SM,IS,HP,HQ,COSLAP,SINLAP,COSLOP,SINLOP,COSLAQ,SINLAQ,COSLOQ,SIN
LOQ,NSTEPP,NSTEPQ,LSAT,LTESTC)
! PROGRAMMED NOV 1985 BY C.C.TSCHERNING, GEODETIC INSTITUTE.
! THE SUBROUTINE COMPUTES MEAN VALUES OF COVARIANCES.
! CHANGED 2013-09-11.
 USE m_geocol_data, ONLY : COSSTN,COSSTE,SINSTN,SINSTE,STEPE,STEPN,FILTER
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,DC,HCMAX,KCI
 USE m_geocol_data, ONLY : CCR,SIGMAX,CCV,NC1,NC2,LOCAL,LSUM
 USE m_geocol_data, ONLY : STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE, COST2Q,SINT2
Q
 USE m_geocol_data, ONLY : HQD,RLATD,SINLATD,COSLATD,RLONGD,SINLOND,COSLOND
 USE m_geocol_data, ONLY : HQE,RLATE,SINLATE,COSLATE,RLONGE,SINLONE,COSLONE
 USE m_geocol_data, ONLY : IKP,IKQ

IMPLICIT NONE

 LOGICAL ::      LMEAP1,LMEAQ1,LTEST,LTESTC,LSAT,LDIFP,LDIFQ

 REAL(KIND=8) :: RADEG,RLAT,RJ,SINLAP,SINLAQ,COVM,COLAP,HP,HQ,HPP,HQQ,&
                 COSLAP,SILAP,COLOP,COSLOP,SILOP,RLAY,COLOQ,COSLOQ,SILOQ,RLAX,&
                 COSDLO,T,RLOX,COV,COLOQ1,COLAQ1,COLOP1,COLAP1,COMEAN,&
                 RLONG,SINLOP,SINLOQ,COLAQ,SILAQ,RLOY,COSLAQ,DEGRAD

 INTEGER ::      i,nstepe,nsteqe,NSTEPP,NSTEPQ,MLAP,&
                 MLOP,J,IS,MLAQ,MLOQ,KP,KQ

 REAL(KIND=8), DIMENSION(2200) :: SM
 REAL(KIND=8), DIMENSION(4)    :: COVME,CVC
 REAL(KIND=8), DIMENSION(3,3,3,3) :: COVX

!COMMON /CMEAQ/STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE,COST2Q,SINT2Q

!COMMON /CMCOV/CCI(24),CCR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HCMAX,CCV(4),DC(36),KVI(39),LOCAL,LSUM

!LTEST=LTCOV
 LTEST=.FALSE.
 RADEG=180.0/3.1415926535D0
 DEGRAD=1.0D0/RADEG
! CCI(20)=1 INDICATES THAT NOT-SO PRECISE EQUATIONS WILL BE USED IN
! COVCX. 2002.10-30.
 CCI(20)=1.0D0
 RLAT=0.0D0
 RJ = 0.0D0
!     STEQE=5.0D0
 LDIFP=NSTEPP.LT.0
 LDIFQ=NSTEPQ.LT.0
 IF (LDIFP.OR.LDIFQ) THEN
```



```
  KP=KCI(6)
  KQ=KCI(7)
  LTEST=LTESTC
  FILTER(1)=1.0d0
  FILTER(2)=-1.0d0
 END IF
 LMEAP1=STEPE.LT.1.0D-8
 LMEAQ1=STEQE.LT.1.0D-8
!IF (LTEST) WRITE(*,*)' STEPE,STEQE ',STEPE,STEQE
 NSTEPE=ABS(NSTEPP)
 NSTEQE=ABS(NSTEPQ)
 IF (LMEAP1) NSTEPE=1
 IF (LMEAQ1) NSTEQE=1
!write(*,*)' 14642 ',LDIFP,LDIFQ,LMEAP1,LMEAQ1,NSTEPP,NSTEPQ
!    write(*,333)rlatd,rlongd,rlate,rlonge
 333 format(4f12.5)
!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
!
 2999 COVM=0.0D0
 IF (LTEST.AND.(.NOT.LTESTC)) WRITE(*,*)' LMP,Q,SPEN,SQEN ',LMEAP1,LMEAQ1,&
 STEPE,STEPN,STEQE,STEQN
 DO  I=1,4
  COVME(I)=0.0D0
 END DO
!
 COLAP=COSLAP
 SILAP=SINLAP
 HPP=HP
!
 DO 3043 MLAP=1,ABS(NSTEPP)
  CCR(2)=HPP
  CCR(4)=SILAP
  CCR(6)=COLAP
  IF (MLAP.EQ.1.OR.(.NOT.LMEAP1)) THEN
   HPP=HP
   COLOP=COSLOP
   SILOP=SINLOP
  END IF
!     IF (MLAP.EQ.1.AND.LMEAP1) THEN
!     CALL PAZIM(RLAT,RLONG,COLAP,SILAP,COLOP,SILOP,&
!    *-COSSTE,-SINSTE,COST2P,SINT2P,LTEST)
!     END IF
  IF (LTEST) RLAY=ATAN2(SILAP,COLAP)*RADEG
   RLAY=ATAN2(SILAP,COLAP)*RADEG
!
  DO 3044 MLOP=1,NSTEPE
   COLAQ=COSLAQ
   SILAQ=SINLAQ
    RLOY=ATAN2(SILOP,COLOP)*RADEG
   IF (LTEST) THEN
    RLOY=ATAN2(SILOP,COLOP)*RADEG
!   WRITE(*,*)' LAP,LOP',RLAY,RLOY
   END IF
!
   DO 3045 MLAQ=1,ABS(NSTEPQ)
    IF (MLAQ.EQ.1.OR.(.NOT.LMEAQ1)) THEN
    HQQ=HQ
    COLOQ=COSLOQ
    SILOQ=SINLOQ
   END IF
!     IF (MLAQ.EQ.1.AND.LMEAQ1) THEN
!     CALL PAZIM(RLAT,RLONG,COLAQ,SILAQ,COLOQ,SILOQ,&
```

```
!    *-COSSQE,-SINSQE,COST2Q,SINT2Q,LTEST)
!     END IF
   CCR(3)=HQQ
   CCR(5)=SILAQ
   CCR(7)=COLAQ
   IF (LTEST) RLAX=ATAN2(SILAQ,COLAQ)*RADEG
   RLAX=ATAN2(SILAQ,COLAQ)*RADEG
!
   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
    RLOX=ATAN2(SILOQ,COLOQ)*RADEG
    IF (LTEST) THEN
     RLOX=ATAN2(SILOQ,COLOQ)*RADEG
!    WRITE(*,*)' LAQ,LOQ, T' ,RLAX,RLOX,T
    END IF
!IF (LCZERO) THEN
! FINITE COVARIANCE FUNCTIONS INTRODUCED MAY, 1996 BY CCT.
! PSI=ACOS(T)
! COV=SCFACT*COZERO(PSI,RDD,1)
! CCV(1)=COV
!ELSE
     CALL COVCX(SM,COV,COVX,IS,LSAT)
     CVC(1)=CCV(1,1)
     CVC(2)=CCV(1,2)
     CVC(3)=CCV(2,1)
     CVC(4)=CCV(2,2)
! CHANGE 2011-10-04.
!       CALL COVCX(SM,COV,IS,.FALSE.)
     IF (LTEST)&
     WRITE(*,334)COV,kp,kq,mlaq,mloq,mlap,mlop,filter(mlap),filter(mlaq),&
     rlax,rlox,rlay,rloy,hpp,hqq,ccr(10),ccr(11)
 334 format(' COV ',f8.3,2i3,4i2,2f4.1,4f8.3,2f9.1,2f5.2)
!END IF
! CORRECTION FOR LATITUDE FACTOR MADE DEC. 1996.
     IF (.NOT.LMEAP1.AND.(.NOT.LMEAQ1)) THEN
     DO I=1,4
      COVME(I)=COVME(I)+CVC(I)*COLAP*COLAQ
     END DO
     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
!
    IF (.NOT.LMEAQ1) THEN
     COLOQ1=COLOQ
     COLOQ=COLOQ*COSSQE-SILOQ*SINSQE
     SILOQ=SILOQ*COSSQE+COLOQ1*SINSQE
    END IF
 3046 CONTINUE
!
 IF (LMEAQ1) THEN
  IF (LDIFQ) THEN
```



```
   COLAQ=COSLATE
   SILAQ=SINLATE
   COLOQ=COSLONE
   SILOQ=SINLONE
   HQQ=HQE
  ELSE
   CALL PAZIM(RLAT,RLONG,COLAQ,SILAQ,COLOQ,SILOQ,&
   COSSQE,SINSQE,COSSQN,SINSQN,.FALSE.)
  END IF
 ELSE
  COLAQ1=COLAQ
  COLAQ=COLAQ*COSSQN+SILAQ*SINSQN
  SILAQ=SILAQ*COSSQN-COLAQ1*SINSQN
 END IF
 3045 CONTINUE
!
 IF (.NOT.LMEAP1) THEN
  COLOP1=COLOP
  COLOP=COLOP*COSSTE-SILOP*SINSTE
  SILOP=SILOP*COSSTE+COLOP1*SINSTE
 END IF
 3044 CONTINUE
!
 IF (LMEAP1) THEN
  IF (LDIFP) then
   COLAP=COSLATD
   SILAP=SINLATD
   COLOP=COSLOND
   SILOP=SINLOND
   HPP=HQD
  ELSE
   CALL PAZIM(RLAT,RLONG,COLAP,SILAP,COLOP,SILOP,&
   COSSTE,SINSTE,COSSTN,SINSTN,.FALSE.)
  END IF
 ELSE
  COLAP1=COLAP
  COLAP=COLAP*COSSTN+SILAP*SINSTN
  SILAP=SILAP*COSSTN-COLAP1*SINSTN
 END IF
 3043 CONTINUE
!
 J=ABS(NSTEPP*NSTEPQ*NSTEPE*NSTEQE)
 IF (LMEAP1.AND.LMEAQ1) RJ = J
 IF (LDIFP.OR.LDIFQ) RJ=1
 COMEAN=COVM/RJ
 IF (.NOT.LMEAP1.AND.(.NOT.LMEAQ1)) THEN
  DO I=1,4
   CVC(I)=COVME(I)/RJ
  END DO
 ELSE
  CVC(1)=COVM/RJ
 END IF
! THESE 4 ASSIGNMENTS ARE CAUSED BY THAT IN THE OLD COMMON BLOCK CCV
! WAS 1-DIMENSIONAL, WHILE IN THE USE STATEMENT IT IS 2-D.
  CCV(1,1)=CVC(1)
  CCV(1,2)=CVC(2)
  CCV(2,1)=CVC(3)
  CCV(2,2)=CVC(4)
 IF (LTEST) WRITE(*,*)' COMEAN, J, RJ ',COMEAN,J,RJ

END FUNCTION COMEAN

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE COVAX(SM,IS)
! ORIGINAL VERSION PROGRAMMED JULY 1975 BY C.C.TSCHERNING. LATEST
! MODIFICATION 2013-01-07.
!
! THIS SUBROUTINE PREPARES CONSTANTS USED FOR COVARIANCE FUNCTION EVALU-
```

```
! ATION, WHICH IS EXECUTED USING THE SUBROUTINES COVBX AND COVCX.
!
! THE COVARIANCE FUNCTION USED IS DEFINED ACCORDING TO A DEGREE-VARIANCE
! MODEL AND A SET OF EMPIRICAL (POTENTIAL) DEGREE-VARIANCES. THE DEGREE-
! VARIANCE MODEL IS SPECIFIED THROUGH THE VALUES OF KI(1)-KI(5),CI(8)-
! CI(10) AND THE PARAMETERS N1 AND LOCAL OCCURRING IN THE COMMON BLOCK
! /CMCOV/. EMPIRICAL ANOMALY DEGREE-VARIANCES WILL HAVE TO BE STORED IN
! SIGMA0 WHEN LOCAL IS FALSE, AND ARE USED FOR THE COMPUTATION OF RESI-
! DUAL POTENTIAL DEGREE-VARIANCES, (SEE REF(A), EQ.(16)).
!
! BY THE CALL OF COVAX, THE KIND OF COVARIANCE FUNCTION TO BE USED IS
! DETERMINED. THE VALUE OF KI(5) WILL DETERMINE THE DEGREE-VARI-
! ANCE MODEL (1,2 OR 3, CF.REF(A),EQ.(17)) THAT WILL BE USED. THE QUAN-
! TITIES K(2),K(3) MUST BE STORED IN KI(3),KI(4), AND BE EQUAL TO ZERO
! WHEN NOT USED (EG.,KI(3),KI(4) BOTH ZERO WHEN KI(5)=1). THE QUANTITY
! A(I) MUST BE STORED IN CI(8) IN UNITS OF (M/SEC)**4, AND THE SQUARE OF
! THE RATIO BETWEEN THE RADIUS OF THE BJERHAMMAR-SPHERE (RB) AND THE
! MEAN RADIUS OF THE EARTH (RE) MUST BE STORED IN CI(10).
!
! THERE ARE THEN THREE POSSIBILITIES:
! (1) ONE OF THE DEGREE-VARIANCE MODELS IS USED WITHOUT MODIFICATIONS.
!     THE SUMMATION LIMIT P OF REF.(A),EQ.(20) IS THEN FIXED TO 3.
!     BECAUSE THIS IS EQUIVALENT TO REQUIRING THE FIRST 3 DEGREE-VARIAN-
!     AREA /CMCOV/ MUST BE EQUAL TO 3 AND .TRUE., RESPECTIVELY.
!     CES TO BE ZERO, THE VARIABLES N1 AND LOCAL STORED IN THE COMMON
! (2) A NUMBER (N1) OF THE ANOMALY DEGREE-VARIANCES (DEGREE ZERO TO
!     N1-1) ARE PUT EQUAL TO EMPIRICAL DETERMINED QUANTITIES. THE ANO-
!     MALY DEGREE-VARIANCE OF DEGREE K WILL HAVE TO BE STORED IN
!     SIGMA0(IS+K+1) IN UNITS OF MGAL**2 WHEN CALLING COVAX. LOCAL MUST
!     BE EQUAL TO FALSE. COVAX WILL CONVERT THE ANOMALY DEGRE5-VARIANCES
!     INTO POTENTIAL DEGREE-VARIANCES. THE POINTER IS MUST BE POSITIVE.
! (3) THE N1 FIRST DEGREE-VARIANCES (DEGREE 0 - N1-1) ARE EQUAL TO ZERO.
!     THIS MEANS, THAT THE VALUES OF A (N1-1)-ORDER LOCAL COVARIANCE
!     FUNCTION WILL BE COMPUTED. LOCAL MUST HAVE THE VALUE .TRUE..
! IN ALL CASES N1 MUST BE LESS THAN 300.
!
! THE COVARIANCES WILL GENERALLY BE COMPUTED BY CLOSED EXPRESSIONS, BUT
! THEY MAY IN CERTAIN CASES BE USELESS IN BIG ALTITUDES OF NUMERICAL
! REASONS, CF. REF(A), SECTION 4. IN THEESE CASES MUST THE LOGICAL VARI-
! ABLE LSUM BE TRUE AND THE VARIABLE HMAX MUST HAVE ASSIGNED A VALUE
! EQUAL TO THE CRITICAL ALTITUDE. WHEN LSUM IS TRUE AND THE HEIGHT OF
! P OR Q IS GREATHER THAN HMAX, WILL THE SERIES REF(A), EQ.(16), ABBRE-
! VIATED TO DEGREE N2-1 BE USED FOR THE COMPUTATION OF THE COVARIANCES.
! THE VALUES OF LSUM, N2 AND HMAX WILL (IN THE SAME WAY AS FOR THE PARA-
! METERS SPECIFYING THE DEGREE-VARIANCE MODEL) BE TRANSFERRED TO COVAX
! THROUGH THE COMMON AREA /CMCOV/, BUT AN ARRAY SM IS TRANSFERRED AS A
! PARAMETER IN THE CALL IN ORDER TO ENABLE VARIABLE DIMENSIONING (SPECI-
! FIED BY THE VARIABLE N2 IN /CMCOV/).
!
! THE CALL OF COVAX WILL ALSO INITIALIZE CERTAIN VARIABLES USED IN
! SUBSEQUENT COMPUTATIONS.
!
! REFERENCES:
! (A) TSCHERNING,C.C.: COVARIANCE EXPRESSIONS FOR SECOND AND LOWER ORDER
!     DERIVATIVES OF THE ANOMALOUS POTENTIAL, REPORTS OF THE DEP. OF
!     GEODETIC SCIENCE NO. 225,1976.
! (B) TSCHERNING,C.C. AND R.H.RAPP: CLOSED COVARIANCE EXPRESSIONS
!     FOR GRAVITY ANOMALIES, GEOID UNDULATIONS, AND DEFLECTIONS OF
!     THE VERTICAL IMPLIED BY ANOMALY DEGREE-VARIANCE MODELS. DEP-
!     ARTMENT OF GEODETIC SCIENCE, THE OHIO STATE UNIVERSITY,
!     REPORT NO. 208, 1974.
! (C) KRARUP, T. AND C.C.TSCHERNING: EVALUATION OF ISOTROPIC COVARIANCE
!     FUNCTIONS OF TORSION BALANCE OBSERVATIONS. BULLETIN GEOD-
!     DESIQUE, VOL. 58, NO. 2, PP. 180-192, 1984.
! (D) TSCHERNING,C.C.: IMPLEMENTATION OF ALGOL-PROCEDURES FOR COV-
!     ARIANCE COMPUTATION ON THE RC 4000-COMPUTER. THE DANISH
!     GEODETIC INSTITUTE INTERNAL REPORT NO. 12, 1976.
! (H) TSCHERNING, C.C.: PREDICTION OF SPHERICAL HARMONIC
!     COEFFICIENTS USING LEAST-SQUARES COLLOCATION. SEPT. 1999.
```



```
! (I) TSCHERNING, C.C.: COMPUTATION OF COVARIANCES OF DERIVATIVES OF THE
!     ANOMALOUS GRAVITY POTENTIAL IN A ROTATED REFERENCE FRAME.
!     MANUSCRIPTA GEODAETICA, VOL. 18, NO. 3, PP. 115-123, 1993.

 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_geocol_data, ONLY : A,SX,TT,BZ,RB2,KT,KT1,K,IIZ,JJ,N3,&
                           ND,ND1,ND2,NRX,KXP,KXQ
!COMMON /DDY/A,S,RB2,T,B,KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,NR,ND1,ND2
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,KCI
 USE m_geocol_data, ONLY : SIGMAX,NC1,NC2,LOCAL,LSUM
!COMMON /CMCOV/CI(24),CR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HMAX,CV(2,2),D(36),KI(37),N1,N2,LOCAL,LSUM

IMPLICIT NONE

 INTEGER :: KP,KQ,KI(37),N1,N2,II,KK,NR,IS

 REAL(KIND=8) :: T,CI(24),S,RB,RE2

 REAL(KIND=8), DIMENSION(2200) :: SM
! THE ARRAY SM IS USED TO STORE THE DEGREE-VARIANCES WHEN THE LOGICAL
! VARIABLE LSUM IS TRUE. IN CASE THE SUBSCRIPT LIMIT IS CHANGED IS IT
! NECESSARY TO CHANGE THE VALUE OF THE VARIABLE N2 ACCORDINGLY.

!-------------------------------------------------------------------------------
!COMMON /CMCOV/CI(24),CR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HMAX,CV(2,2),D(36),KI(37),N1,N2,LOCAL,LSUM
! CHANGE TO 2200 2010-11-24.

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GM,ITCOUN,LF,LT

!COMMON /DDY/A,S,RB2,T,B,KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,NR,ND1,ND2

! THE COMMON BLOCK CONTAINS THE VALUES OF PARAMETERS USED FOR THE COM-
! PUTATIONS AND RETURN VALUES OF FUNCTIONS AND CONSTANTS, WHICH HAVE
! BEEN USED IN THE COMPUTATIONS.
! PARAMETERS USED FOR THE COMPUTATIONS:
!   CI(8) = THE CONSTANT A(I) OF REF.(A), EQ.(17) IN UNITS OF (M/SEC)**4
!   CI(10) THE SQUARE OF THE RATIO BETWEEN THE BJERHAMMAR-SPHERE RADIUS
!   (RB) AND THE MEAN RADIUS OF THE EARTH (RE), OR IF NEGATIVE RB-RE,
!   (CHANGE MADE 3 JULY 1985).
!   SIGMA0(IS+1)-SIGMA0(IS+N1) MUST CONTAIN THE EMPIRICAL ANOMALY
!   DEGREE VARIANCES IN UNITS OF MGAL**2.
!   KI(3) = K(2) OF DEG.VAR. MODEL 2 OR 3,
!   KI(4) = K(3) OF DEG.VAR. MODEL 3, CF. REF.(A), EQ.(17).
!   KI(5) = THE DEG.VAR. MODEL NUMBER, (EQUAL TO 1, 2 OR 3),
!   N1 = THE NUMBER OF EMPIRICAL DEGREE-VARIANCES USED (LOCAL =.FALSE.)
!   OR (ORDER+1) OF THE LOCAL COVARIANCE FUNCTION USED (LOCAL=.TRUE.).
!   HMAX, N2, LSUM. HMAX IS THE HEIGHT ABOVE WHICH THE LEGENDRE SERIES
!   OF MAXIMAL DEGREE N2-1 WILL BE USED FOR THE COMPUTATION OF THE CO-
!   VARIANCES WHEN LSUM IS TRUE. N2 MUST BE GREATHER THAN 2 AS WELL AS
!   GREATHER THAN N1.
! RETURN VALUES:
!   CI(10) RB-RE, A NEGATIVE VALUE (MODIFICATION 3 JULY 1985).
!   CI(9) = RB**2.
!

 CI=CCI
 KI=KCI
 N1=NC1
 N2=NC2
 KT = KI(5)
 KT1 = KT+1
 S=SX
 T=TT
 II=IIZ
 KP=KXP
 KQ=KXQ
 NR=NRX
```

```
 IF (KT.GE.3) GO TO 15
 DO 16 K = KT, 2
   16 KI(K+2) = D0
   15 KI(1) = -2
 KI(2) = -1
!
 IF ((KT.LT.3).OR.(KT.EQ.3.AND.KI(4).GT.KI(3))) GO TO 17
! ASSURING, THAT KI(4).GT.KI(3), BECAUSE THIS FACT IS USED IN SUB-
! 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
!     N3 = N1
 A = CI(8)
 S = CI(10)
 IF (S.GT.D0) GO TO 40
! S IS HERE RB-RE, A NEGATIVE VALUE. (MODIFICATION 3 JULY 1985).
 RB=RE+S
 RB2=RB*RB
 RE2=RE*RE
 S=RB2/RE2
 SX=S
 GO TO 41
   40 RB2 = S*RE2
 CI(10)=RE*( SQRT(S)-D1)
   41 CI(9) = RB2
 RB2 = RB2*1.0D-10
 T = D0
 TT=T
!
 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
! CONVERSION FROM MGAL**2 TO M**2/SEC**2.
 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
 CCI=CI
 KCI=KI
 RETURN
END SUBROUTINE COVAX

! ------------------------------------------------------------------
SUBROUTINE COVBX(SM,LSAT,IS)
! ORIGINAL VERSION PROGRAMMED JULY 1975 BY C.C.TSCHERNING AS A SUB-
! ENTRY OF COVAX. NEW VERSION CREATED SEP 1987 BY CCT.
! NEW VERSION JUNE 4, 1991. LAST UPDATE 2012-10-19 BY CCT.
!
! THE CALL OF COVBX WILL FIX CERTAIN CONSTANTS USED FOR THE COMPUTA-
! TIONS, WHICH ARE INDEPENDENT OF THE POINTS P AND Q. WHEN COVBX IS CAL-
! LED, THE KIND OF QUANTITIES BETWEEN WHICH THE COVARIANCE IS TO BE
! COMPUTED MUST BE SPECIFIED.  THIS IS DONE BY STORING IN KI(6) AND
```



```
! KI(7) INTEGERS EQUAL TO THE EQUATION NUMBERS OF REF.A, EQ.(1) - (9)
! (12) AND (14), AND 10, 11, 13, 15 CORRESPONDING TO REF.(C), EQ.
! (3) - (6). HOWEVER, THE QUANTITY OF KIND 2 IS NOW THE GRAVITY
! DISTURBANCE (CHANGED FROM THE SAME QUANTITY DIVIDED BY R).
! ADDED 1999.02.12 IS (17), FOR COEFFICIENTS OF SPHERICAL HARMONICS.
!
! REFERENCES (A) - (I): SEE COVAX.
!

 USE m_geocol_data, ONLY : SIGMAP,SLOP,SLOQ,CLOP,CLOQ,IIDEG,JJORD
 USE m_geocol_data, ONLY : A,SX,TT,RB2,BZ,KT,KT1,K,IIZ,JJ,N3,&
                           KK,KXQ,KXP,ND,NRX,ND1,ND2
!COMMON /DDY/A,S,RB2,T,B,KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,NR,ND1,ND2

 USE m_data,        ONLY : LSPOUT
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,DC,HCMAX,KCI
 USE m_geocol_data, ONLY : CCR,SIGMAX,CCV,NC1,NC2,LOCAL,LSUM
 USE m_data,        ONLY : KSAT,LTESTS,NWAR,LY
!USE m_geocol_data, ONLY : COVX,CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
 USE m_geocol_data, ONLY :      CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
                           NDP,NDQ,LX,LNX
 USE m_data,        ONLY : K7,K9,K11,K13,K15,K17,K19,K21,K23,K8,C11,&
                           J2,I3,I4,LN,L
!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)

IMPLICIT NONE

 REAL(KIND=8) :: GM,B,T,S,RE2,SNN,BB0,RKP,REM

 INTEGER ::      NR,II,KQ,KP,KI(37),N1,N2,&
                 M,MK,IS,I,NDT,NDTOT,NDY,KU

 LOGICAL ::      LSAT,LTEST,LSPHAR

 REAL(KIND=8), DIMENSION(2200) :: SM
 REAL(KIND=8), DIMENSION(24)   :: CI
 REAL(KIND=8), DIMENSION(56)   :: CR
 REAL(KIND=8), DIMENSION(2,2)  :: CV
 REAL(KIND=8), DIMENSION(36)   :: D

!-----------------------------------------------------------------------------
!COMMON /CMCOV/CI(24),CR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HMAX,CV(2,2),D(36),KI(37),N1,N2,LOCAL,LSUM

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GM,ITCOUN,LF,LT

!COMMON /DDX/K7(17),K9(17),K11(17),K13(17),K15(17),K17(17),K19(17),K21(17),K23(1
7),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 /CSAT/COVX(3,3,3,3),CIX(7,5),CFA,KSAT(17,2),LSATPP,LSATQ,NDX1(5),NDX2(5)
,NDP,NDQ,NWAR,LY,LX(7,5),LNX(7,5),LTESTS
! THE COMMON BLOCK CONTAINS THE VALUES OF PARAMETERS USED FOR THE COM-
! PUTATIONS AND RETURN VALUES OF FUNCTIONS AND CONSTANTS, WHICH HAVE
! BEEN USED IN THE COMPUTATIONS.
! PARAMETERS USED FOR THE COMPUTATIONS:
!   CI(8) = THE CONSTANT A(I) OF REF.(A), EQ.(17) IN UNITS OF (M/SEC)**4
!   CI(10) THE SQUARE OF THE RATIO BETWEEN THE BJERHAMMAR-SPHERE RADIUS
!   (RB) AND THE MEAN RADIUS OF THE EARTH (RE), OR IF NEGATIVE RB-RE,
!   (CHANGE MADE 3 JULY 1985).
!   CI(13) USER DEFINED VALUE OF CI(11). CI(14), CI(15) USER DEFINED
!   VALUES OF CI(21) - CI(24).
!   SIGMA0(IS+1)-SIGMA0(IS+N1) MUST CONTAIN THE POTENTIAL ANOMALY
!   DEGREE-VARIANCE CORRECTIONS, CF. REF.(A), EQ.16.
!   KI(3) = K(2) OF DEG.VAR. MODEL 2 OR 3,
!   KI(4) = K(3) OF DEG.VAR. MODEL 3, CF. REF.(A), EQ.(17).
```

```
!   KI(5) = THE DEG.VAR. MODEL NUMBER, (EQUAL TO 1, 2 OR 3),
!   KI(6),KI(7) THE INTEGER SPECIFYING THE KIND OF QUANTITY WHICH IS
!   ASSOCIATED WITH P, Q, RESPECTIVELY,
!   KI(26) − KI(34) USER SPECIFIED VALUES FOR KI(10) − KI(23).
!   KI(35) − KI(37) USED BY SUBROUTINE COVCG FOR STATISTICAL PURPOSES.
!   N1 = THE NUMBER OF EMPIRICAL DEGREE−VARIANCES USED (LOCAL =.FALSE.)
!   OR (ORDER+1) OF THE LOCAL COVARIANCE FUNCTION USED (LOCAL=.TRUE.).
!   HMAX, N2, LSUM. HMAX IS THE HEIGHT ABOVE WHICH THE LEGENDRE SERIES
!   OF MAXIMAL DEGREE N2−1 WILL BE USED FOR THE COMPUTATION OF THE CO−
!   GREATHER THAN N1.
! RETURN VALUES:
!   CI(1)−CI(7), THE QUANTITIES C(J,Q) OF REF.(A), EQ.(47), WITH
!   CI(1) − CI(KI(5)+1) = C(J,Q), CI(5) = C(KI(5)+2,Q),
!   CI(6) = C(KI(5)+3,Q), CI(7) = C(KI(5)+4,Q),
!   CI(11),CI(12) QUANTITIES USED TO GIVE THE COMPUTED
!   COVARIANCES THE PROPER UNITS.
!   CI(21) − CI(24) THE QUANTITIES M(1) − M(4) OF REF.(A) EQ. (26) −
!   (29). (CHANGE MADE 1986.10.20).
!   SIGMA(IS+4) − SIGMA(IS+N1), THE POTENTIAL DEGREE−VARIANCES MULTI−
!   PLIED BY THE FACTORS GIVEN IN REF.(A), TABLE 1.
!   SIGMA(IS+1) − SIGMA(IS+3), THE DEGREE−VARIANCES OF DEGREE 0,1,2
!   MINUS TERMS OF THE SAME DEGREES ACQUIRED FROM REF.(A), EQ.(34),(35),
!   (41) AND (42).
!   KI(8),KI(9) THE NUMBER OF DIFFERENTIATIONS IN RADIAL DIRECTION AND
!   WITH RESPECT TO T = COS(SPHERICAL DIST.) TO BE PERFORMED.
!   KI(10) − KI(15) THE CONSTANTS I,K,J,M,J1,M1 OF REF.(A), SECTION 2.
!   KI(16) − KI(19) THE QUANTITIES M(1) − M(4) OF REF.(A), EQ.(26)−(29).
!   KI(20),KI(21) THE EXPONENT OF THE REFERENCE GRAVITY,
!   KI(22),KI(23) THE EXPONENT OF THE RADIAL DISTANCE AND
!   KI(24),KI(25) SUBSCRIPTS OF THE RESULT STORED IN CV (COMMON CMCOV).
!
! CHANGE 2010−11−23. and 2011−03−26.
!     DIMENSION SM(2001),SIGMAX(400,5)
! THE ARRAY SM IS USED TO STORE THE DEGREE−VARIANCES WHEN THE LOGICAL
! VARIABLE LSUM IS TRUE. IN CASE THE SUBSCRIPT LIMIT IS CHANGED IS IT
! NECESSARY TO CHANGE THE VALUE OF THE VARIABLE N2 ACCORDINGLY.
!
! SIGMAX IS USED TO HOLD DEGREE−VARIANCES OF RADIAL DERIVATIVES
! UP TO ORDER 2 IN P AND Q. (CHANGE MAY 1991).
!
! THE ARRAYS K7 − K23 CONTAINS TABLES OF QUANTITIES RELATED TO THE KIND
! OF COVARIANCES (1 − 14) WHICH MAY BE COMPUTED. THEIR ACTUAL VA−
! LUES WILL AFTER CALL OF COVBX BE STORED IN THE ELEMENTS OF THE ARRAY
! KI HAVING SUBSCRIPTS 8 − 25.
! K7 CONTAINS THE ORDER OF DIFFERENTIATION WITH RESPECT TO T,K8 THE
! ORDER OF DIFFERENTIATION WITH RESPECT TO THE RADIUS, CF.REF(A),TABLE
! 1. K9,K11,K13 THE KIND OF DIFFERENTIATIONS TO BE COMPUTED WITH RESPECT
! TO THE LATITUDE (2) AND THE LONGITUDE (3), CF.REF(A),SECTION 3. K15
! AND K17 CONTAINS AN INTEGER, WHICH WILL BE ADDED TO THE DEGREE. THE
! SUM WILL THEN BE MULTIPLIED WITH THE DEGREE−VARIANCE OF THE CORRESPON−
! DING DEGREE WHEN A FIRST AND/OR SECOND DIFFERENTIATION WITH RESPECT
! TO THE RADIAL DISTANCE HAS TAKEN PLACE.
! C11 CONTAIN QUANTITIES USED TO GIVE THE COVARIANCES THE PROPER UNITS.
!
!     LTEST=LTESTS
!     LTESTS=.TRUE.
 LTEST=.FALSE.
 KI=KCI
 N1=NC1
 N2=NC2
 CI=CCI
 CV=CCV
 D=DC
 CR=CCR
 LSPHAR=.FALSE.
 ITCOUN=0
! ADDED 2011−06−16.
 GM=GMC
! ADDED DUE TO USE OF use m_data STATEMENT, WHERE GM IS CALLED GMC.
```



```
 KP=KXP
 KQ=KXQ
 II=IIZ
 B=BZ
 T=TT
 S=SX
 NR=NRX
 RB2 = CI(9)
 RE2=RE**2
 S=RB2/RE2
 SX=S
 A = CI(8)
 II=KI(3)
 IIZ=II
 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
!
 DO 20 M = 1, 2
 K = KI(M+5)
! FOR M = 1, K IS EQUAL TO THE KIND EVALUATED IN P AND FOR M = 2 EQUAL
! TO THE KIND EVALUATED IN Q.
!
 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)
!
 CI(11) = CI(11)*C11(K)
!     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
!
! USER DEFINED VALUES OF KI AND CI. MAY BE USER FOR DENSITY CONTRAST
! COVARIANCES, CF. REF.(D), SECTION 3.
   42 IF (K.NE.17) THEN
  DO MK=1,8
   KI(M+MK*2+7)=KI(MK+25)
  END DO
  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)
!      LSPHAR=.FALSE.
!      WRITE(*,*)' LSPHAR=F '
 ELSE
  LSPHAR=.TRUE.
  CI(21)=D0
  CI(22)=D0
  CI(23)=D0
  CI(24)=D0
!      WRITE(*,*)' LSPHAR=T '
!      WRITE(*,*)' N1,N3 ',N1,N3
 END IF
   20 CONTINUE
!
 KQ = K
```

```
 KP = KI(6)
   19 ND = KI(8)
 NR = KI(9)
!
 NDP=K7(KP)+K8(KP)
 NDQ=K7(KQ)+K8(KQ)
!     WRITE(*,*)' COVBX: ND,NDP,NDQ= ',ND,NDP,NDQ
! ND AND NR ARE THE NUMBER OF DIFFERENTIATIONS WITH RESPECT TO T AND
! THE RADIAL DISTANCES, RESPECTIVELY. NDP, NDQ ARE THE TOTAL NMBER OF
! DERIVATIVES IN P, Q, REPECTIVELY.
!
 IF (LSAT.AND.(.NOT.LSPHAR)) GO TO 100
! 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.2200) THEN
  WRITE(*,*)' WARNING36 N1.GT.2200 '
  STOP
 END IF
 SNN=S**3
 DO 21 M = 3, 2001
! CHANGE TO 2200 2010-11-24. CHANGE BACK TO 2001 2011-03-26.
 B = D1
 DO I = 1, 4
  IF ( ABS(CI(I+20)).GT.0.0) B = B*(M+CI(I+20)-1)
 END DO
 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 K = 1, KT1
  B = B/(M+KI(K)-1)
 END DO
! STORING THE MODIFIED DEGREE-VARIANCES OF DEGREE M-1 IN SM(M) AND AD-
! 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)
  SIGMAP(M)=SM(M)*SNN/BB0
 ELSE
  SIGMAP(M)=A*SNN*B/BB0
 END IF
!
! CF. REF(H), EQ. (4).
 SIGMAP(M)=SIGMAP(M)/(D2*M-D1)
   21 CONTINUE
 IF (N1.GT.2) THEN
  SM(3) = SIGMA(IS+3)
  SIGMAP(3)=SM(3)*(S**3)/(BB0*5.0D0)
 ELSE
  SIGMAP(3)=0.0D0
 END IF
!
 IF (LSPHAR.AND.LSPOUT.AND.LTEST) THEN
  LSPOUT=.FALSE.
  WRITE(*,*) ' GRAVITY ANOMALY AND POTENTIAL DEG.VAR. DEG 3-200 '
  WRITE(*,249) (SIGMAP(K)*(2*K-1)*(K-2)**2*1.0D10/RE2,K=3+IS,200+IS)
  WRITE(*,249) (SIGMAP(K),K=3+IS,200+IS)
  249  FORMAT(8F9.4)
 END IF
 IF (LSUM) N1 = N3
!
! EVALUATION OF THE QUANTITIES C(J,NR), CF.REF(A), TABLE 2.
 DO 23 K = 1, 7
   23 CI(K) = D0
!
```



```
 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))
! CF.,EQ.(19). WE WILL THEN COMPUTE THE QUANTITIES GIVEN IN REF(A)
! 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)
!
 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
!
 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))
!
! THE LOGICAL ARRAYS L AND LN REGISTER WHICH TERMS THAT WILL HAVE TO
! 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))
!
 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)) GOTO 32
 GOTO (34,34,35,35,34,36,37),K
   34 B = D1
 GOTO 33
   35 B = D1/(M+KI(K) -1)
 GOTO 33
   36 B = (M-1)
 GOTO 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(6E13.6,I3)
!
 ND1 = ND+1
 ND2 = ND1+1
!     write(*,*)' covbx ',nd1,nd2
 KCI=KI
 CCI=CI
 CCR=CR
 CCV=CV
 DC=D
 RETURN
!
  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
!     write(*,*)' covbx_2 ',nd1,nd2
!
 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
 NRX=NR
 NDY=NDTOT-M
 IF (LF)WRITE(6,*)NDT,CI(21),CI(22),CI(23),CI(24)
! UPDATING THE DEGREE-VARIANCES, CF. REF(A), TABLE 1.
 SIGMAX(1,NDT) = D0
 SIGMAX(2,NDT) = D0
 DO M = 3, N1
  B = D1
  DO I = 2, NDT
   B = B*(M+CI(I+19)-1)
 END DO
! NOGET GALT HER.
! SIGMAX(M,NDT) NOW CONTAINS DEGREE-VARIANCES OF RADIAL DERIVATIVES IN
! P AND Q UP TO ORDER 2.
  IF (M.LE.N3) SIGMAX(M,NDT) = SIGMA0(IS+M)*B
 END DO
!
! EVALUATION OF THE QUANTITIES C(J,NR), CF.REF(A), TABLE 2.
 DO K = 1, 7
  CI(K) = D0
 END DO
!
 DO K = 1, KT1
  CI(K) = D1
  DO KU = 1, KT1
   IF (K.NE.KU) CI(K) = CI(K)/(KI(KU)-KI(K))
  END DO
 END DO
! CF.,EQ.(19). WE WILL THEN COMPUTE THE QUANTITIES GIVEN IN REF(A)
! 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)
!
 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
```



```
!
 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)
!
! THE LOGICAL ARRAYS L AND LN REGISTER WHICH TERMS THAT WILL HAVE TO
! 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)
!
 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)) GOTO 132
 GOTO (134,134,135,135,134,136,137),K
  134 B = D1
 GOTO 133
  135 B = D1/(M+KI(K) -1)
 GOTO 133
  136 B = (M-1)
 GOTO 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
!
 NDX1(NDT) = NDY+1
 NDX2(NDT) = NDY+2
  101 CONTINUE
 KCI=KI
 CCI=CI
 CCR=CR
 CCV=CV
 DC=D
 RETURN
 END SUBROUTINE COVBX
!
SUBROUTINE COVCX(SM,COV,COVX,IS,LSAT)
! ORIGINALLY PROGRAMMED JULY 1975 BY C.C.TSCHERNING AS A SUB-
! ENTRY TO COVAX. SEPARATE SUBROUTINE CREATED SEPT 1987 BY CCT.
! LATEST MODIFICATION 2013-03-25   BY CCT.
!
! COMPUTATION OF THE COVARIANCE IN A SPECIFIC PAIR OF POINTS, OR
! BETWEEN A FUNCTIONAL ASSOCIATED WITH A POINT AND A SPHERICAL-HARMONIC
! COEFFICIENT. THE  VALUE IS RETURNED THROUGH THE PARAMETER COV.
! THE COVARIANCES COMPUTED WILL BE IN UNITS CORRESPONDING TO THE KIND
! OF QUANTITIES, I.E. FOR KIND (1) METERS, (2) EOTVOS (E), (3) MGAL,
! (4),(5) E, (6),(7) ARCSECONDS, (8) - (14) E, (17) UNITLESS.
! THE FOLLOWING QUANTITIES MUST BE STORED IN THE ELEMENTS OF THE ARRAY
! CR WHEN COVCX IS CALLED: (1) COSINE TO THE SPHERICAL DISTANCE BET-
! WEEN P AND Q, (2),(3) THE HEIGHT OF P, Q RESPECTIVELY, (4),(5) SINE
! OF THE LATITUDE THE  OF P, Q, RESPECTIVELY, (6),(7) COSINE OF THE
! LATITUDE OF P, Q, RESPECTIVELY, (8),(9) SINE AND COSINE OF THE
! LONGITUDE DIFFERENCE. THE REFERENCE GRAVITY WILL HAVE TO BE STORED
! IN CR(10),CR(11) FOR P, Q RESPECTIVELY (WHEN USED, OTHERWISE STORE
! 1.0). FOR KIND 17, COS AND SIN OF LONGITUDES MUST BE STORED IN THE
! COMMON BLOCK /PDEGV/.
!
! THE CALL OF COVCX WILL RESULT IN THE COMPUTATION OF THE COVARIANCE ,
! WHICH IS TRANSFERRED TO THE CALLING PROGRAM THROUGH THE VARIABLE COV.
! THE RESULT WILL ALSO BE TRANSFERRED IN THE COMMON CMCOV, BY THE ARRAY
! CV(2,2). IN CASE IT IS POSSIBLE TO COMPUTE MORE THAN ONE QUANTITY AT
! A TIME (I.E. WHEN DERIVATIVES WITH RESPECT TO T=COS(SPHERICAL DIST-
```

```
! TANCE) ARE COMPUTED, KINDS 6 - 11, 13 AND 15), THE COVARIANCE
! OF TYPE 6, 8, 10 AND 23 WILL BE STORED IN THE ELEMENT WITH SUBSCRIPT
! 2 AND OTHERWISE IN THE ELEMENT WITH SUBSCRIPT 1. THE KIND OF THE
! FUNCTIONALS IN P WILL DETERMINE THE VALUE OF THE FIRST SUBSCRIPT
! WHILE THE KIND OF THE FUNCTIONALS IN Q WILL DETERMINE THE SECOND
! SUBSCRIPT. EXAMPLE: KIND 6 IN P AND KIND 1 IN Q WILL DELIVER
! THE COVARIANCE BETWEEN THE PRIME-VERTICAL VERTICAL DEFLECTION AND
! AND THE HEIGHT ANOMALY IN CV(1,1), BETWEEN THE MERIDIAN VERTICAL
! DEFLECTIAN AND THE HEIGHT ANOMALY IN CV(2,1).
!
! WHEN LSAT IS TRUE, THE 4D ARRAY COVX HOLDS THE VECTORS OR MATRICES
! OF COVARIANCES BETWEEN ALL 0, 1 OR 2 DERIVATIVES.
!

 USE m_params,      ONLY : IIMAX,NSPHAR
 USE m_geocol_data, ONLY : SIGMAP,IIDEG,JJORD,SLOP,CLOP,SLOQ,CLOQ,ROOT0
 USE m_geocol_data, ONLY : SUMIJ,CCCIJ,SQ2,YS,YC,VV,V1,GS,GC,DDS,DDC,SN2,AXS,&
                           GMS,IIOLD,JOLD,IR,LSPHAR,LTSPH
 USE m_geocol_data, ONLY : A,SX,TT,RB2,BZ,KT,KT1,K,IIZ,JJ,N3,&
                           KK,KXQ,KXP,ND,NRX,ND1,ND2
!COMMON /DDY/A,S,RB2,T,B,KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,NR,ND1,ND2
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,DC,HCMAX,KCI
 USE m_geocol_data, ONLY : CCR,SIGMAX,CCV,NC1,NC2,LOCAL,LSUM
!COMMON /CMCOV/CI(24),CR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HMAX,CV(2,2),D(36),KI(37),N1,N2,LOCAL,LSUM
 USE m_data,        ONLY : KSAT,LTESTS,NWAR,LY
!USE m_geocol_data, ONLY : COVX,CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
! CHANGED 2013-03-25.
 USE m_geocol_data, ONLY :      CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
                           NDP,NDQ,LX,LNX
 USE m_data,        ONLY : K7,K9,K11,K13,K15,K17,K19,K23,K8,C11,&
                           J2,I3,I4,LN,L
!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)

IMPLICIT NONE

 INTEGER ::       II,KQ,KP,NR, &
                  KI(37),N1,N2,IMAX1,I21,I,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

 REAL(KIND=8) :: GM,CY,R2PQ,HMAX,&
                  B,HP,HQ,SP,SQ,CP,CQ,SD,CD,RP,RQ,&
                  RE2,CLAT,SLAT,CLON,SLON,&
                  RH,GAMM,COV,CJLO,SJLO,WWC,WWS,COVC,WW,&
                  SC,CS,SCC,CC,CCS,COVS,CSC,CPSD,CQSD,CPCD,CQCD,SS,&
                  T,S,S2,ST,T2,P2,P3,GI,GJ,SI,RL,RL2,RL1,RN,&
                  RNL,RL3,RL5,S3,RL4,RL7,S4,S5,RL6,SS2,RP2,&
                  RQ2,RPQ,FAK5,RP2Q,CNX,RPQ2,D3132,D313,CN23,CN33,D37,D27,&
                  CF,C11P,C11Q,CI(24),CR(56),D(36),CFA

 LOGICAL ::       LSUMC,LOLDP,LOLDQ,LTEST,&
                  LSAT,LDGP,LDGQ,LCOS,LSPHP,LSPHQ

 REAL(KIND=8), DIMENSION(3)    :: GG,GGC,GGS
!REAL(KIND=8), DIMENSION(4)    :: SS1
 REAL(KIND=8), DIMENSION(5)    :: CZ
 REAL(KIND=8), DIMENSION(6)    :: RM,Q,C,V,U,G,P,R,SS1

 REAL(KIND=8), DIMENSION(3,3,3,3)  :: COVX
 REAL(KIND=8), DIMENSION(3,3)  :: DDD,DDDC,DDDS
 REAL(KIND=8), DIMENSION(2,2)  :: CV
 REAL(KIND=8), DIMENSION(6,6)  :: DD
 REAL(KIND=8), DIMENSION(6,8)  :: CX
 REAL(KIND=8), DIMENSION(8,5)  :: CN,DCN

 REAL(KIND=8), DIMENSION(2200) :: SM
```



```
! THE ARRAY SM IS USED TO STORE THE DEGREE-VARIANCES WHEN THE LOGICAL
! VARIABLE LSUM IS TRUE. IN CASE THE SUBSCRIPT LIMIT IS CHANGED IS IT
! NECESSARY TO CHANGE THE VALUE OF THE VARIABLE N2 ACCORDINGLY.

!-----------------------------------------------------------------------------
!COMMON /CMCOV/CI(24),CR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HMAX,CV(2,2),D(36),KI(37),N1,N2,LOCAL,LSUM
! CHANGE TO 2200 2010-11-24.

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GM,ITCOUN,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)
!COMMON /DDX/K7(15),K9(15),K11(15),K13(15),K15(15),K17(15),K19(15),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
! B changed for BZ 2012-09-13. and back

!COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFA,KSAT(17,2),LSATPP,LSATQ,NDX1(5),NDX2(5)
,NDP,NDQ,NWAR,LY,LX(7,5),LNX(7,5),LTESTS

!COMMON /CON3/SUMIJ((NSPHAR+1)**2),CCCIJ((NSPHAR+1)**2),SQ2,YS,YC,VV,V1,GS(3),GC
(3),DDS(3,3),SN2(0:NSPHAR),AXS,GMS,DDC(3,3),IIOLD,JOLD,IR,LSPHAR,LTSPH
! THE COMMON BLOCK CONTAINS THE VALUES OF PARAMETERS USED FOR THE COM-
! PUTATIONS AND RETURN VALUES OF FUNCTIONS AND CONSTANTS, WHICH HAVE
! BEEN USED IN THE COMPUTATIONS.
! PARAMETERS USED FOR THE COMPUTATIONS:
!   CI(8) = THE CONSTANT A(I) OF REF.(A), EQ.(17) IN UNITS OF (M/SEC)**4
!   CI(10) THE SQUARE OF THE RATIO BETWEEN THE BJERHAMMAR-SPHERE RADIUS
!   (RB) AND THE MEAN RADIUS OF THE EARTH (RE), OR IF NEGATIVE RB-RE,
!   (CHANGE MADE 3 JULY 1985).
!   CI(13) USER DEFINED VALUE OF CI(11). CI(14), CI(15) USER DEFINED
!   VALUES OF CI(21) - CI(24).
!   NEW VARIABLES ADDED MAY 1, 1986 AND NOV 1986:
!   CI(16) - CI(24), WHERE CI(20)=0.0 IF PRECISE FORMULAE FOR DERIVATIVES
!   MAY BE USED. IN THIS CASE IS CI(16)=SIN(LONGITUDE DIFFERENCE/2)**2,
!   CI(17)=SIN(LATITUDE DIFFERENCE/2), CI(18)=COS(LATITUDE DIFFERENCE),
!   CI(19)=COS(LATITUDE DIFFERENCE/2). OTHERWISE CI(20)=1.0.
!   CR(2),CR(3) THE HEIGHT OF P, Q, RESPECTIVELY, (UNITS METERS),
!   CR(4),CR(5) SINE OF THE LATITUDE OF P, Q, RESPECTIVELY,
!   CR(6),CR(7) COSINE OF THE LATITUDE OF P, Q, RESPECTIVELY,
!   CR(8),CR(9) SINE AND COSINE OF THE LONGITUDE DIFFERENCE,
!   CR(10),CR(11) THE REFERENCE GRAVITY IN P, Q, RESPECTIVELY (WHEN
!   USED, OTHERWISE STORE 1.0E0), (UNITS M/SEC**2).
!   KI(3) = K(2) OF DEG.VAR. MODEL 2 OR 3,
!   KI(4) = K(3) OF DEG.VAR. MODEL 3, CF. REF.(A), EQ.(17).
!   KI(5) = THE DEG.VAR. MODEL NUMBER, (EQUAL TO 1, 2 OR 3),
!   KI(6),KI(7) THE INTEGER SPECIFYING THE KIND OF QUANTITY WHICH IS
!   ASSOCIATED WITH P, Q, RESPECTIVELY,
!   KI(26) - KI(34) USER SPECIFIED VALUES FOR KI(10) - KI(23).
!   KI(35) - KI(37) USED BY SUBROUTINE COVCG FOR STATISTICAL PURPOSES.
!   N1 = THE NUMBER OF EMPIRICAL DEGREE-VARIANCES USED (LOCAL =.FALSE.)
!   OR (ORDER+1) OF THE LOCAL COVARIANCE FUNCTION USED (LOCAL=.TRUE.).
!   HMAX, N2, LSUM. HMAX IS THE HEIGHT ABOVE WHICH THE LEGENDRE SERIES
!   OF MAXIMAL DEGREE N2-1 WILL BE USED FOR THE COMPUTATION OF THE CO-
!   VARIANCES WHEN LSUM IS TRUE. N2 MUST BE GREATHER THAN 2 AS WELL AS
!   GREATHER THAN N1.
! RETURN VALUES:
!   CR(ND*8+12), THE VALUES OF THE ND'TH DERIVATIVE OF THE SUM OF THE
!   FINITE LEGENDRE-SERIES, CF.REF.(A), EQ.(20),(48) AND (52).
!   CR(ND*8+13) - CR(ND*8+19), THE VALUES OF THE ND'TH DERIVATIVES OF
!   THE FUNCTIONS F(-2), F(-1), F(KI(3)), F(KI(4)), S0, S1, S2, CF. REF.
!   (A), EQ. (42), (41), (39), (39), (30), (34) AND (35).
!   SIGMA0(IS+1) - SIGMA0(IS+N1) THE POTENTIAL DEGREE-VARIANCE
!   CORRECTIONS, CF. REF.(A), EQ.(16), (AFTER THE CALL OF COVAX).
!   SIGMA(IS+4) - SIGMA(IS+N1), THE POTENTIAL DEGREE-VARIANCES MULTI-
!   PLIED BY THE FACTORS GIVEN IN REF.(A), TABLE 1.
!   SIGMA(IS+1) - SIGMA(IS+3), THE DEGREE-VARIANCES OF DEGREE 0,1,2
```

```
!   MINUS TERMS OF THE SAME DEGREES ACQUIRED FROM REF.(A), EQ.(34),(35),
!   (41) AND (42).
!   KI(8),KI(9) THE NUMBER OF DIFFERENTIATIONS IN RADIAL DIRECTION AND
!   WITH RESPECT TO T = COS(SPHERICAL DIST.) TO BE PERFORMED.
!   KI(10) - KI(15) THE CONSTANTS I,K,J,M,J1,M1 OF REF.(A), SECTION 2.
!   KI(16) - KI(19) THE QUANTITIES M(1) - M(4) OF REF.(A), EQ.(26)-(29).
!   KI(20),KI(21) THE EXPONENT OF THE REFERENCE GRAVITY,
!   KI(22),KI(23) THE EXPONENT OF THE RADIAL DISTANCE AND
!   KI(24),KI(25) SUBSCRIPTS OF THE RESULT STORED IN CV (COMMON CMCOV).
!
! ARRAYS CN, DCN, SIGMAX, DD ADDED MAY 1991.
!
! REFERENCES (A)-(I) SEE COVAX.
!
!     DIMENSION SM(2001),CX(6,8),DC(6),SIGMAX(400,5),CN(8,5),DCN(8,5),&
! CHANGE 2010-09-28.

!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(2201)),(D(1),DD(1,1))
! CHANGE 2010-09-28.
!    *,(CX(2,8),SS2),(SIGMAX(1,1),SIGMA0(401)),(D(1),DD(1,1))
! K7 CONTAINS THE ORDER OF DIFFERENTIATION WITH RESPECT TO T,K8 THE
! ORDER OF DIFFERENTIATION WITH RESPECT TO THE RADIUS, CF.REF(A),TABLE
! 1. K9,K11,K13 THE KIND OF DIFFERENTIATIONS TO BE COMPUTED WITH RESPECT
! TO THE LATITUDE (2) AND THE LONGITUDE (3), CF.REF(A),SECTION 3. K15
! AND K17 CONTAINS AN INTEGER, WHICH WILL BE ADDED TO THE DEGREE. THE
! SUM WILL THEN BE MULTIPLIED WITH THE DEGREE-VARIANCE OF THE CORRESPON-
! DING DEGREE WHEN A FIRST AND/OR SECOND DIFFERENTIATION WITH RESPECT
! TO THE RADIAL DISTANCE HAS TAKEN PLACE.
! C11 CONTAIN QUANTITIES USED TO GIVE THE COVARIANCES THE PROPER UNITS.
!
 LTEST=LSPHAR.AND.ITCOUN.LT.5.AND.LTESTS
 IF (LTESTS) THEN
 KI(35)=KI(35)+1
 ITCOUN=ITCOUN+1
 END IF

! write(*,555)A,S,II,N1,(CR(KP),KP=1,9)
!555 format(' COVCX 14990 A,S,CR ',2D14.7,2i3,/,9F10.5)

!COMMON /CMCOV/CI(24),CR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HMAX,CV(2,2),D(36),KI(37),N1,N2,LOCAL,LSUM
! THIS IS BECAUSE DIFFERENT NAMES ARE USED.
 CFA=CFX
 KPQ=KXP
 CI=CCI
 CR=CCR
 CV=CCV
 LCOS=LF
 COVC=D0
 COVS=D0
 D=DC
 HMAX=HCMAX
 KI=KCI
 N1=NC1
 N2=NC2
 GM=GMC
 II=IIZ
 NR=NRX
 B=BZ
 S=SX
 KP=KXP
 KQ=KXQ
 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)
 RP = RE+HP
 RQ = RE+HQ
 RE2= RE**2
 COVX=D0
!
 KP=KI(6)
 KQ=KI(7)
 IF (LSAT.AND.(.FALSE.)) WRITE(*,*)' kpkq ',KP,KQ,KI(1),KI(2),KI(3),KI(4),KI(5)
!
! CHANGE 2003-03-22.
 LDGP=KP.EQ.3
 LDGQ=KQ.EQ.3
 IF (KP.EQ.17.OR.KQ.EQ.17) THEN
!      WRITE(*,*)'10385 CX, KP,KQ,LSAT= ',KP,KQ,LSAT
!
  IF (KP.NE.17.AND.KQ.EQ.17) THEN
   LSPHQ=LT
   KPQ=KP
   CLAT=CP
   SLAT=SP
   SLON=SLOP
   CLON=CLOP
   RH=RE+HP
   GAMM=CR(10)
   IF (ITCOUN.LT.0.AND.LTESTS) WRITE(*,*)' LSAT ',LSAT,' KPQ=KP= ',KPQ,GAMM
   CLAT=CP
  END IF
!
  IF (KQ.NE.17.AND.KP.EQ.17) THEN
   LSPHP=LT
   KPQ=KQ
   CLAT=CQ
   SLAT=SQ
   SLON=SLOQ
   CLON=CLOQ
   RH=RE+HQ
   GAMM=CR(11)
   IF (ITCOUN.LT.0.AND.LTESTS) WRITE(*,*)' LSAT ',LSAT,' KPQ=KQ= ',KPQ,GAMM
  END IF
! LTESTS=.FALSE.
!
  SQ2=SQRT(D2)
  IMAX1=IIDEG+1
  I21=2*(IIDEG+1)
  IF (IIMAX.LT.I21) WRITE(*,*) ' IMAX TOO LARGE '
!
  DO I=1, I21
   ROOT0(I)= SQRT(DFLOAT(I-1))
  END DO
!
 IF (KP.EQ.17.AND.KQ.EQ.17) THEN
! COV IS THE VARIANCE OF THE (I,J)'TH COEFFICIENT. CHANGE 2006-11-20
! SO THAT IT REFERS TO THE SEMI-MAJOR AXIS ASSOCIATED WITH THE COEFFICIENTS.
  COV=SIGMAP(IIDEG+1)*SN2(IIDEG)
  IF (IIDEG.LT.2) COV=1.0D3
  IF (LTSPH.AND.(IIDEG.GT.25.AND.IIDEG.LT.33)) WRITE(*,*)' IIDEG+1, COV= ',IIDEG
+1,COV
 ELSE
! PREDICTION OF SPHERICAL HARMONIC COEFFICIENT OF DEGREE IIDEG AND ORDER
! JJORD.
!
! SETTING ORDER OF DIFFERENTIATION.
  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
!      IF  (KPQ.GE.6.OR.KPQ.LE.11) THEN   ** ERRONEOUS  **
! 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
!
  CFA=D1
  IIOLD=-1
  JOLD=-1
  JMAX1=ABS(JJORD)+1
  LCOS=JJORD.GE.0
!
! SEE REF(H) EQ. (6).
  I=IMAX1
  J=JMAX1
! THE CALCULATION OF THE COVARIANCES (EQUAL TO THE SPHERICAL HARMONICS
! ON WHICH IS APPLIED THE RELEVANT FUNCTIONAL) IS DONE BY RECURSION.
! THE RECURSION ELEMENTS ARE STORED ON SCRATCH UNIT 98 WITH ONE
! RECORD PER OBSERVATION. THE ELEMENTS ARE FIRST CALCULATED FOR THE
! COSINE TERM AND STORED (LCOS IS TRUE)  AND REUSED FOR THE SIN TERM
! AND THEN UPDATED AND STORED ON UNIT 98. 2011-08-16.
  CALL SPHARMA(SLAT,CLAT,SLON,CLON,SJLO,CJLO,RH,I-1,J-1,IDIF,&
  .TRUE.,.FALSE.,LCOS)
!
  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
!
  WWC=SIGMAP(I)*SN2(I-1)*YC
! UNITS OF M.
  IF (IDIF.GT.0) THEN
   GGC(1)=SIGMAP(I)*SN2(I-1)*GC(1)/AXS
   GGC(2)=SIGMAP(I)*SN2(I-1)*GC(2)/AXS
   GGC(3)=SIGMAP(I)*SN2(I-1)*GC(3)/AXS
!       IF (ITCOUN.LT.3) WRITE(*,*)' GCS ',GC,GS
   IF (.NOT.LSAT) THEN
    GGC(1)=-GGC(1)*RADSEC/GAMM
    GGC(2)=-GGC(2)*RADSEC/GAMM
! UNITS OF ARCSEC.
! GRAVITY DISTURBANCE:
    GGC(3)=-GGC(3)*1.0D5
! UNITS OF MGAL USED.
! GRAVITY ANOMALY MISSING CC
   END IF
   IF (IDIF.GT.1) THEN
    DO 990,KKC=1,3
    DO 990,KKD=1,3
  990     DDDC(KKC,KKD)=SIGMAP(I)*SN2(I-1)*DDC(KKC,KKD)*1.0D9/AXS**2
! EU USED.
   END IF
!
  END IF
!
  WWS=SIGMAP(I)*SN2(I-1)*YS
  IF (IDIF.GT.0) THEN
   GGS(1)=SIGMAP(I)*SN2(I-1)*GS(1)/AXS
   GGS(2)=SIGMAP(I)*SN2(I-1)*GS(2)/AXS
   GGS(3)=SIGMAP(I)*SN2(I-1)*GS(3)/AXS
   IF (.NOT.LSAT) THEN
```



```
! GRAVITY ANOMALY MISSING CC (SEE BELOW AT LABEL 1013).
    GGS(1)=-GGS(1)*RADSEC/GAMM
    GGS(2)=-GGS(2)*RADSEC/GAMM
! UNITS OF ARCSEC. CORRECTED 2011-06-15.
    GGS(3)=-GGS(3)*1.0D5
! UNITS OF MGAL USED.
   END IF
   IF (IDIF.GT.1) THEN
    DO KKC=1,3
     DO KKD=1,3
  991      DDDS(KKC,KKD)=SIGMAP(I)*SN2(I-1)*DDS(KKC,KKD)*1.0D9/AXS**2
!          WRITE(*,*)' DDDC/S11 ',DDDC(1,1),DDDS(1,1)
     END DO
    END DO
! EU USED.
   END IF
!
  END IF
!
  IF (.NOT.LSAT) THEN
   GOTO (1011,1012,1013,1014,1015,1016,1017,1018,1019,1020,&
         1021,1024,1023,1022,1025),KPQ
! HEIGHT ANOMALY (M).
 1011   IF (ITCOUN.LT.5.AND.J.EQ.25.AND.I.EQ.25.AND.LTEST) WRITE(*,*) ' 1011 ',W
WC,WWS,GAMM
   COVC=WWC/GAMM
   COVS=WWS/GAMM
   GO TO 1126
! GRAVITY DISTURBANCE (MGAL).
 1012   COVC= WWC*I/RH*1.0D5
   COVS= WWS*I/RH*1.0D5
   GO TO 1126
! GRAVITY ANOMALY (MGAL).
 1013   COVC= WWC*(I-2)/RH*1.0D5
   COVS= WWS*(I-2)/RH*1.0D5
!
   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
!
   GO TO 1126
! 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
! VERTICAL GRAVITY GRADIENT.
 1015   COVC=WWC*I*(I+1)/(RH*RH)*1.0D9
   COVS=WWS*I*(I+1)/(RH*RH)*1.0D9
!
   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
!
   GO TO 1126
! DEFLECTION, MERIDIAN COMP.
 1016   COVC= GGC(2)
   COVS= GGS(2)
   IF (LTEST) THEN
    WRITE(*,*)' GGC,GGS1 ', COVC,COVS
    ITCOUN=ITCOUN+1
   END IF
   GO TO 1126
! DEFLECTION, PRIME VERTICAL COMP.
 1017   COVC= GGC(1)
   COVS= GGS(1)
```

```
   IF (LTEST) THEN
    WRITE(*,*)' GGC,GGS1 ', COVC,COVS
    ITCOUN=ITCOUN+1
   END IF
   GO TO 1126
! PRIME VERTICAL DER. OF GRAVITY ANOMALY
! ERROR HERE CCCCC
 1019   COVC=-GGC(2)*I/RH
   COVS=-GGS(2)*I/RH
   GO TO 1126
! MERIDIAN DER. OF GRAVITY ANOMALY
 1018   COVC=-GGC(1)*I/RH
   COVS=-GGS(1)*I/RH
   GO TO 1126
! MERIDIAN DER. OF GRAVITY DISTURBANCE. CORR. 2000-03-27 BY CCT.
 1020   COVC=-DDDC(3,2)
   COVS=-DDDS(3,2)
   GO TO 1126
! PRIME VERTICAL DER. OF GRAVITY DISTURBANCE.
 1021   COVC=-DDDC(3,1)
   COVS=-DDDS(3,1)
   GO TO 1126
! 2. ORDER PRIME VERTICAL DER.
 1022   COVC=DDDC(1,1)
   COVS=DDDS(1,1)
   GO TO 1126
! MIXED PRIME VERTICAL & MERIDIAN DER. * 2. (TORSION BALANCE).
 1023   COVC= DDDC(1,2)*D2
   COVS= DDDS(1,2)*D2
   GO TO 1126
! 2. ORDER MERIDIAN COMP.
 1024   COVC=DDDC(2,2)
   COVS=DDDS(2,2)
   GO TO 1126
! DIFFERENCE 2. ORDER HORIZONTAL DER. (TORSION BALANCE).
 1025   COVC=(DDDC(1,1)-DDDC(2,2))
   COVS=(DDDS(1,1)-DDDS(2,2))
!
 1126   CONTINUE
   IF (LCOS) THEN
    COVX(1,1,1,1)=COVC
    COV=COVC
   ELSE
    COVX(1,1,1,1)=COVS
    COV=COVS
   END IF
!
  ELSE
   IF (LCOS) THEN
    WW=WWC
    DO IKK=1,3
! SIGN CHANGED 2011-06-17.
     GG(IKK)=-GGC(IKK)*1.0D5
     DO JKK=1,3
      DDD(IKK,JKK)=DDDC(IKK,JKK)
     END DO
    END DO
   ELSE
    WW=WWS
    DO IKK=1,3
! SIGN CHANGED 2011-06-17.
     GG(IKK)=-GGS(IKK)*1.0D5
     DO JKK=1,3
      DDD(IKK,JKK)=DDDS(IKK,JKK)
     END DO
    END DO
   END IF
!
   NCASE=NDP+1+NDQ*3
```



```
!       WRITE(*,*)NDP,NDQ,NCASE
   GO TO (1801,1802,1803,1804,1810,1810,1807,&
   1810,1810),NCASE
! NO DERIVATIVES IN P OR Q.
1801    COVX(1,1,1,1)=WW/GAMM
   GO TO 1810
! 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)
! GRAVITY ANOMALY WITH GEOID. ADDED 1992.09.07.
   IF (LDGP) COVX(3,1,1,1)=GG(3)-1.0D5*D2*WW/RP
! ERROR DETECTED 2003-07-24.
!       IF (LDGP) COVX(3,1,1,1)=GG(3)-D2*WW/RP
   GO TO 1810
! 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
! 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)
! GRAVITY ANOMALY WITH GEOID. ADDED 1999.09.07, CORR 000.04.28.
   IF (LDGQ) THEN
    COVX(1,1,3,1)=GG(3)-1.0D5*D2*WW/RQ
    IF (ITCOUN.LT.4.AND.I.LT.5.AND.(.FALSE.)) WRITE(*,*) ' LDGQ ',WW,GG,COVX(1,1
,3,1)
   END IF
! ERROR DETECTED 2003-07-24.
   GO TO 1810
! 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)
!
 1810   CONTINUE
   COV=COVX(KSAT(KP,1),KSAT(KP,2),KSAT(KQ,1),KSAT(KQ,2))
   IF (ITCOUN.LT.0) THEN
    WRITE(*,*)' KSAT ',KSAT(KP,1),KSAT(KP,2),KSAT(KQ,1),&
    KSAT(KQ,2),' COV ',COV,'NDP,NDQ ',NDP,NDQ,' IDIF ',IDIF
    write(*,1515)COVX
 1515 format(5d14.5)
   END IF
!       ITCOUN=ITCOUN+1
!
! THIS PERMITS TEST OF LSAT PREDICTION OF COEFFICIENTS FOR
! 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
!
   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
!
  END IF
!
  IF (J.EQ.1) THEN
   CCCIJ((I-1)**2+1)=COVC
  ELSE
   IF (.NOT.LCOS) CCCIJ((I-1)**2+2*(J-1)+1)=COVS
   IF (LCOS) CCCIJ((I-1)**2+2*(J-1))=COVC
  END IF
!
  if (j.lt.4.AND.IR.EQ.0) WRITE(*,511) I,J,COVC,COVS,CLAT,CJLO,SJLO,COVX(1,1,3,1
),KP,KQ,IDIF
  511  FORMAT(' I,J,COVC,LLY,CLAT,C/SJLO,KP,KQ= ',2I3,2D15.5,4F6.3,3I3)
!      ITCOUN=ITCOUN+1
!
 END IF
! END CALCULATION OF COVARIANCE.
!
! CHANGE HERE TO TAKE CARE OF KSI, ETA 2000-05-02
 GOTO (2011,2011,2011,2011,2011,2016,2016,2018,2018,2020,&
        2020,2022,2024,2022,2024),KPQ
! EV, DERIVATIVES  Z, DGZ, ZZ.
 2011 IF (LCOS) THEN
  CV(1,1)=COVC
 ELSE
  CV(1,1)=COVS
 END IF
 GOTO 2026
!
! 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
!
! 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
!
! 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
!
! 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
!
! 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
!
 2026 ITCOUN=ITCOUN+1
 CCV=CV
 CFX=CFA
```

```
 RETURN
 END IF
!
! IN HEIGH ALTITUDES AND WHEN LSUM IS TRUE WILL THE COVARIANCE BE COM-
! PUTED BY A SUMMATION OF THE LEGENDRE-SERIES ABBREVIATED TO DEGREE
! N2-1.
 LSUMC = LSUM .AND. (HP.GT.HMAX .OR. HQ.GT.HMAX)
! COMPUTATION OF THE CONSTANT USED TO CONVERT THE COVARIANCE INTO
! PROPER UNITS.
 CI(12) = CI(11)/(RP**KI(22)*RQ**KI(23)*CR(11)**KI(21)*CR(10)**KI(20))
!
 S = RB2/(RP*RQ)
!if (ltests) write(*,*)' 19246 S,RP,RQ,RB2 ',s,rp,rq,rb2
!     IF(CI(10).LT.D0) S=D1-(RE*(HP+HQ+D2*(RE-CI(10)))+HP*HQ
!    *- (RE-CI(10))**2)/(RP*RQ)
! CODE 12 IS THE SECOND ORDER MERIDIONAL DERIVATIVE.
 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 (.false.) WRITE(*,*) '16257, OLDP,Q ',LOLDP,LOLDQ,LSAT,KI(6),KI(7)
 IF (LSUMC) N1 = N3
!
! COMPUTATION OF THE QUANTITIES D(1)-D(36),CF.REF(A),SECTION 3.
! (MODIFIED ACCORDING TO REF.(C)).
!     IF (.TRUE.)WRITE(*,*)' COVCX ND=',ND
!DO I=1,36
! D(I)=D0
!END DO
 D=D0
 IF (ND.EQ.0) GO TO 55
!
 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
! CF. REF.(D), EQ. (7) AND (8).
! 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(*,*) ' WARNING37 D(2) ',D(2),(CS-SCC)
  WRITE(*,*) ' WARNING38 D(7) ',D(7),(SC-CSC)
  WRITE(*,*)' 161719spcpsqcq ',CI(16),CI(17),CI(19),SP,CP,SQ,CQ
 END IF
 GOTO 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
!
 IF (ND.EQ.1) GO TO 55
 SS = SP*SQ
 D(8) = CC+SS*CD
! 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) THEN
 D(4) = D(2)+D(3)
 D(6) = D(3)-D(2)
 ELSE
!GO TO 92
 D(4) = -T
 D(6) = -CQCD/CP
 END IF
 IF (.NOT.LOLDQ) GO TO 93
!  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
!
   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
!
   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
 DO K=1,6
  DO M=1,6
!  if (DD(M,K).NE.D(M+(K-1)*6)) THEN
!   write(*,*)' wrong indexing in COVCY '
!   stop
!  end if
   DD(K,M)=D(M+(K-1)*6)
! CHANGE 2013-03-20.
!  DD(M,K)=D(M+(K-1)*6)
  end do
 end do
```

```
!IF (.true.) WRITE(*,1555)(D1-T),CI(20),CI(17),CR(8)
!1555 FORMAT(' 15614 T1,CI20,17,CR8',4D14.5)
!
 S2 = S*S
 ST = S*T
 T2 = T*T
 P2 = (D3*T2-D1)/D2
 P3 = (D3*ST+D1)/D2
!
! INITIALIZING ARRAY ELEMENTS. NOTE THE USE OF THE EQUIVALENCING.
 CX = D0
 C = D0
 DC = D0
 DO K = 1, 40
  CR(K+11) = D0
 END DO
 Q(1)=D0
 RM(1)=D0
!
 IF (.NOT.LSAT) THEN
!
! SUMMATION AND DIFFERENTIATION OF THE LEGENDRE SERIES, CF.REF(A),EQ.
! (49) AND (51).
  IF (LSUMC) N1 = N2
  K1 = N1
  K2 = N1+1
  K = N1-1
! write(*,*)' 15623 N1,K,IS ',N1,K,IS
  DO M = 1, N1
   GI = (D2*K+D1)*S/K1
   GJ = -K1*S2/K2
   K2 = K1
   K1 = K
   K = K-1
   IF (.NOT.LSUMC) THEN
    SI = SIGMA(IS+K2)
   ELSE
    SI = SM(K2)
   END IF
   I2 = 0
   I1 = 1
   DO 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
    I1 = I
   END DO
  END DO
! write(*,*)' 15656 C  ',C
  IF (LSUMC) N1 = N3
!
!      IF (LSUMC) GO TO 75
 ELSE
  CX=D0
  V=D0
  U=D0
  G=D0
  R=D0
  P=D0
  SS1=D0
  Q=D0
  KP=KI(6)
  KQ=KI(7)
  LDGP=KP.EQ.3
  LDGQ=KQ.EQ.3
!
! INITIALIZING ARRAY ELEMENTS. NOTE THE USE OF THE EQUIVALENCING.
```



```
  DO  K = 1, 8
   DO  M = 1, 6
    CX(M,K) = D0
   END DO
  END DO
  NDTOT=NDP+NDQ+1
  DO  K = 1, 8
   DO  NDT=1,NDTOT
    CN(K,NDT) = D0
    DCN(K,NDT) = D0
   END DO
  END DO
!
! SUMMATION AND DIFFERENTIATION OF THE LEGENDRE SERIES, CF.REF(A),EQ.
! (49) AND (51).
  K1 = N1
  K2 = N1+1
  K = N1-1
  DO M = 1, N1
   GI = (D2*K+D1)*S/K1
   GJ = -K1*S2/K2
   K2 = K1
   K1 = K
   K = K-1
   DO NDT=1,5
    SI = SIGMAX(K2,NDT)
    I2 = 0
    I1 = 1
    DO 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
     if (.false.) write(*,5557)CN(I,NDT),SIGMAX(K2,NDT),I,NDT
 5557     format(' CN,SIG,I,NDT ',2d16.5,2i3)
     I1 = I
    END DO
   END DO
  END DO
!      write(*,5557)CN(3,2),3,2
!
 END IF
 IF (LSUMC) GO TO 75
! COMPUTATION OF THE FUNCTIONS L=R(1), N=1/RN, M=RM(2), F0=P(2), CF.
! REF.(A), EQ. (31)-(33),(40) AND (77A).
  CY=D0
  V=D0
  U=D0
  G=D0
  R=D0
  P=D0
  SS1=D0
  Q=D0
!

  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*LOG(D2*RN)
  RL3 = RL2*RL1
  RL5 = RL3*RL2
  S3 = S2*S
  R(2) = -S*RL1
```

```
  IF (ND.EQ.0) GO TO 56
!
! COMPUTATION OF THE DERIVATIVES WITH RESPECT TO T.
! 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
!
! 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
!
! 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
!
! 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)
!
   56  IF (LN(2)) GO TO 58
! COMPUTATION OF THE FUNCTION F-1 AND ITS DERIVATIVES, CF. REF.(A),
! EQ. (41) AND (61) - (65).
  U(2) = S*(RM(2)+T*P(2))
  IF (ND2.LT.3) GO TO 58
  DO K = 3, ND2
   57  U(K) = S*(RM(K)+T*P(K)+(K-2)*P(K-1))
  END DO
!
   58  IF (LN(1)) GO TO 60
! COMPUTATION OF THE FUNCTION F-2 AND ITS DERIVATIVES, CF. REF.(A) EQ.
! (42), AND (65)- (68).
  DO 59 K = 2, ND2
   GOTO (61,61,62,63,64,65),K
   61   CY = S*(D1-T2)/4.0E0
   GOTO 59
   62   CY = -ST/D2
   GOTO 59
   63   CY = D3*P(2)-S/D2
   GOTO 59
   64   CY = 9.0E0*P(3)
   GOTO 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))
!
   60  CONTINUE
   IF (LTESTS.and.(.false.)) write(*,5559)v,RM,P,S,T
 5559 format('V ',6d12.5,/,6d12.5,/,6d12.5,/,6d12.5,/,6d12.5)
   IF (LN(3)) GOTO 73
! COMPUTATION OF THE FUNCTION F1 AND ITS DERIVATIVES, CF. REF.(A) EQ.
! (36), REF.(B), EQ.(101) AND REF.(A), EQ.(70),(71).
  Q(2) = LOG(D1+D2*S/(D1-S+RL))
  IF (ND.EQ.0) GOTO 66
  Q(3) = S2*RNL
  IF (ND.EQ.1) GOTO 66
  Q(4) = S3*((RL1+D1)*RN+RL2)*RNL
  IF (ND.EQ.2) GOTO 66
  Q(5) = S4*(D3*RL4+((D2+D3*RL1)*RL2+(D2 +(D4+D2*RL1)*RL1)*RN)*RN)*RNL
```



```
  IF (ND.EQ.3) GOTO 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
!
! COMPUTATION OF THE FUNCTION F2 AND ITS DERIVATIVES, CF. REF.(A), EQ.
! (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
! write(*,*)' COVCX 15842 JJ,J1,II,I1 ',JJ,J1,II,I1
  IF (J1.LE.1) GO TO 73
!
! 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 (LTESTS) write(*,5553)P,JJ,J1,K,ND2
   if (.false.) write(*,5553)P,JJ,J1,K,ND2
 5553 format(' 15757 P,JJ,J1 ',6d11.3,4i3)
   IF (K.NE.I1) GO TO 71
   DO 70 M = 2, ND2
   70   G(M) = P(M)
   71  K1 = K
!
   73  IF (LN(6)) GO TO 72
! 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)
!
! CF. REF.(A), EQ. (35).
   72  IF (L(7)) THEN
        SS2= S2*((T+S)*RL3+D3*S*(T2-D1)*RL5)
        CX(2,8)=SS2
       END IF
!
   75  CONTINUE
   DO M=1,6
    CX(M,1)=C(M)
    CX(M,2)=V(M)
    CX(M,3)=U(M)
    CX(M,4)=G(M)
    CX(M,5)=P(M)
    CX(M,6)=R(M)
    CX(M,7)=SS1(M)
   END DO
   if (.false.) THEN
    write(*,5552)CX,Q,JJ,I1,J1
 5552 format(' C ',6d11.3,/,' V  ',6d11.3,/,' U  ',6d11.3,/,&
   ' G  ',6d11.3,/,' P  ',6d11.3,/,' R  ',6d11.3,/,'SS1',6d11.3,/,&
   ' CX ',6d11.3,/,' Q  ',6d11.3,3I4)
   END IF
  IF (.NOT.LSAT) THEN
! ADDING THE DIFFERENT TERMS, CF. REF.(A), EQ. (22),(47).
! TIPLIED BY RB**2 IN UNITS OF MGAL**2, THE INTEGERS K(2),K(3) OF EQ.
 DO 79 M = 2, ND2
! CF. REF.(A), EQ. (50),(52).
 C(M) = S*C(M)
```

```
 IF (LTEST)WRITE(*,*)' CM',C(M),M
 CR(M*8 -4) = C(M)
 DO 78 K = 1, 7
 IF (LN(K)) GO TO 78
! STORING THE TERMS FOR TRANSFER TO THE CALLING PROGRAM USING THE COMMON
! 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 (.false.) WRITE(*,1)A,CX(M,K+1),CI(K),C(M),K
    1 FORMAT(' A,CX,CI,C,K,NDT ',4E15.7,3I2)
   78 CONTINUE
   79 CR(M+50)=C(M)
!     write(*,*)(C(M),M=2,ND2)
!
 ELSE
!
! 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 (M.EQ.3.AND.NDT.EQ.2) WRITE(*,*)' CM',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
!       if (M.EQ.3.AND.NDT.EQ.2) WRITE(*,*)' CM1',CN(M,NDT)
    CN(M,NDT)=CN(M,NDT)+A*CX(M,K+1)*CIX(K,NDT)*FAK5
!       if (M.EQ.3.AND.NDT.EQ.2) WRITE(*,*)' CM2',CN(M,NDT)
    if (.false.) WRITE(*,1)A,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 (.false.) WRITE(*,*)' NDP,NDQ,KP,KQ=',NDP,NDQ,KP,KQ
!
! WE NOW CALCULATE THE CROSS-COVARIANCES BETWEEN ALL QUANTI-
! TIES OF THE GIVEN ORDERS.
  NCASE=NDP+1+NDQ*3
  GO TO (801,802,803,804,805,806,807,808,809),NCASE
! NO DERIVATIVES IN P OR Q.  CHANGED 2005-02-18, SO THAT
! HEIGHT ANOMALIES CAN BE USED.
 801  COVX(1,1,1,1)=CN(1,1)/(CR(10)*CR(11))
 GO TO 810
! 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
! 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
! 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
! 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
! 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
! 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
! 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
 GOTO 810
! 2 DERIVATIVES IN P, ONE IN Q. REF(I), EQ. (29)-(33).
  806 RP2Q=RP2*RQ
!     write(*,5558)D
!     write(*,5558)DD
 5558 format(6F10.6)
 CNX=CN(2,2)-T*CN(3,1)+D(3)*D(3)*CN(4,1)-CN(2,1)
 if (.false.) write(*,5556)CN(2,2),T,CN(3,1),D(3),CN(4,1),CN(2,1)
 5556 format(3d16.5/3d16.5/3d16.5)
 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
!     COVX(2,1,1,1)=(D(2)*D(3)*D(13)*CN(4,1)+D(17)*CN(2,1)
!    *     +(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
! 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)))/RP2
Q
 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)))/RP2
Q
 COVX(3,1,3,1)=DD(3,1)*CN(2,3)/RP2Q
!     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
! check 2010-11-30.
 COVX(2,2,3,1)=(CN(1,3)+D(2)*D(2)*CN(3,2)-T*CN(2,2))/RP2Q
!     write(*,*)' 16733 ',COVX(2,2,1,3)
 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
! 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
!     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)
! check 2010-11-30.
 COVX(2,2,3,1)=-(CN(1,3)+D3*CN(1,2)+D(2)*D(2)*(CN(3,2)+D2*CN(3,1)) &
!     COVX(2,2,3,1)=-(D3*CN(1,2)+D(2)*D(2)*(CN(3,2)+D2*CN(3,1))
 -T*(CN(2,2)+D2*CN(2,1)))/RP2Q
!     write(*,*)' 16760 ',COVX(2,2,3,1)*1.0D14
 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
 IF (.FALSE.) WRITE(*,5555) LDGQ,COVX(1,1,1,1),COVX(2,2,1,1),COVX(3,3,1,1),rp2q,
CNX,DD(2,1),DD(3,1),D2
 5555 FORMAT(l2,4d16.5/4d16.5)
 GOTO 810
! 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
! ERROR 2002-11-26.
!     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
! 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
! check 2010-11-30.
!     COVX(3,1,1,1)=(CN(1,3)+CN(1,2)+D(13)*D(13)*CN(3,2)
 COVX(3,1,1,1)=(CN(1,3)+D(13)*D(13)*CN(3,2)-T*CN(2,2))/RPQ2
! 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))/RP
Q2
!     COVX(2,1,2,1)=(DD(2,2)*DD(1,3)*CN(3,1)+DD(1,2)*(DD(3,2)*CN(3,1)
! 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)))/RPQ
2
 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)))/RPQ
2
 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
! check 2010-11-30.
!     COVX(3,1,2,2)=(CN(1,2)+D(7)**2*CN(3,2)
 COVX(3,1,2,2)=(CN(1,3)+D(7)**2*CN(3,2)-T*CN(2,2))/RPQ2
```



```
!     write(*,*)' 16813 ',COVX(3,1,2,2)*1.0D14,D(7)**2*
!    *CN(3,2)/RPQ2*1.0D14
 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
! 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
! GRAVITY ANOMALY ADDED 1992.09.30.
 IF (LDGP) THEN
! check 2010-11-30.
!      COVX(3,1,1,1)=-(CN(1,3)+D3*CN(1,2)+D(13)*D(13)*(CN(3,2)
  COVX(3,1,1,1)=-(CN(1,3)+D2*CN(1,2)+D(13)*D(13)*(CN(3,2)+D2*CN(3,1))-T*(CN(2,2)
+D2*CN(2,1)))/RPQ2
! 2000-04-03
  COVX(3,1,2,1)=-DD(1,2)*DD(1,3)*(CN(3,2)+D2*CN(3,1))/RPQ2
!      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
  IF (LTEST) WRITE(*,*)' COVX(3,1,3,1) ', COVX(3,1,3,1)
  COVX(3,1,1,2)=COVX(3,1,2,1)
! check 2010-11-30.
  COVX(3,1,2,2)=-(CN(1,3)+D2*CN(1,2)+D(7)**2*(CN(3,2) &
!      COVX(3,1,2,2)=-(D3*CN(1,2)+D(7)**2*(CN(3,2)
    +D2*CN(3,1))-T*(CN(2,2)+D2*CN(2,1)))/RPQ2
!      write(*,*)' 16844 ',COVX(3,1,2,2)*1.0D14
  COVX(3,1,3,2)=-D(7)*(CN(2,3)+D2*CN(2,2))/RPQ2
!      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
! 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
! SUSPECTED ERROR 2002-10-07
!    *  +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
 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
```

```
!
 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
! 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
!     COVX(3,1,2,1)=(D(3)*D(13)*D(7)*(CN(3,2)-CN(3,1))
!    *  +(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
!
 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
!     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
!
 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)
!
 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
!
 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
!
 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)
!
 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)
!
 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
  810 END IF
!
  204 IF (.NOT.LSAT) THEN
! INTEGERS SPECIFYING THE KINDS OF DIFFERENTIATION WITH RESPECT TO THE
! 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 (.FALSE.) WRITE(*,*) ' 16986,i,j,k,m,j1,m1,nd1 ',i,j,k,k,j1,m1,nd1,LOLDP,LOL
DQ
 IF (.NOT.(LOLDP.OR.LOLDQ)) GO TO 110
!
 IJ = I+J
 IF (I.GT.3) IJ = 5
```

```
 KM = K+M
 IF (K.GT.3) KM = 5
!
! COMPUTATION OF THE DERIVATIVES OF ORDER ND WITH RESPECT TO THE LATI-
! TUDES AND THE LONGITUDES, CF. REF.(A), EQ. (43) - (46).
 GOTO (80,81,82,83,84),ND1
   80 COV = C(2)
 GOTO 85
   81 COV = -C(3)*D(I+6*(K-1))
 GOTO 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)
!     write(*,*)' cov82 ',cov
 GOTO 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))
 GOTO 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)
!
! GIVING THE COVARIANCE THE PROPER UNITS.
   85 COV = COV*CI(12)
 CV(1,1)=COV
!
 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)
!  ================================================================
  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)
!  ================================================================
  KZ=2
 GO TO 198
  116 CV(1,1) = CZ(2)*D(13)
 CV(1,2) = CZ(2)*D(7)
!  ================================================================
  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
!  ================================================================
 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
!  ================================================================
 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)
!  ================================================================
 KZ=6
! FIRST ORDER HORIZONTAL DERIVATIVES IN BOTH P AND Q.
 GO TO 198
  115 CONTINUE
!
 IIX=2
 DO 119 IX = 1, 2
 IIY=2
 DO 120 JX = 1, 2
 IF (ND.EQ.4) GO TO 121
! SECOND ORDER HORIZONTAL DERIVATIVE IN P OR Q.
!     write(*,*)' 17072 2. order deriv ',KI(6),KI(7)
 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
!  ================================================================
 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
! ================================================================
 KZ=8
  120 IIY=1
  119 IIX=1
  198 CONTINUE
  COV = CV(KI(24),KI(25))
! ================================================================
 IF (.FALSE.)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,5I4,4F10.4)
  199 CONTINUE
  CCV=CV
  CFX=CFA
  RETURN
 ELSE
  COV=COVX(KSAT(KP,1),KSAT(KP,2),KSAT(KQ,1),KSAT(KQ,2))
! if (ltests) write(*,7789)COV,KSAT(KP,1),KSAT(KQ,1)
 7789 format(' 16388 COV,',d12.5,4i3)
  IF (KP.EQ.15.AND.KQ.NE.15) COV=COV-COVX(2,2,KSAT(KQ,1),KSAT(KQ,2))
! 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
   IF (KP.EQ.1.AND.(.NOT.LSATPP)) THEN
!        WRITE(*,*)' C11(KP),CR(11) ',C11(KP),CR(10)
    C11P=C11(KP)/(CR(10)**K19(KP))
   ELSE
! CHANGE 2003-04-01 AND 2011-07-25.
!     C11P=C11(KP)/(CR(10)**K19(KP))
   C11P=C11(KP)
   END IF
  END IF
  IF (KQ.EQ.6.OR.KQ.EQ.7) THEN
   C11Q=1.0D5
  ELSE
   IF (KQ.EQ.1.AND.(.NOT.LSATQ)) THEN
!        WRITE(*,*)' C11(KQ),CR(11) ',C11(KQ),CR(11)
    C11Q=C11(KQ)/(CR(11)**K19(KQ))
   ELSE
!        C11Q=C11(KQ)/(CR(11)**K19(KQ))
    C11Q=C11(KQ)
   END IF
  END IF
! if (ltests) write(*,*)' 16414 C11P,C11Q ',C11P,C11Q,C11(KP),K19(KP),KP
  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)
! write(*,*)' COV,CFA 16417 ',COV,CFA
  COV=COV*CFA
! 2000-04-04.
  IF (.false.)WRITE(*,*)' KSAT ',KSAT(KP,1),KSAT(KP,2),KSAT(KQ,1),&
  KSAT(KQ,2),' COV ',COV,' CFA ',CFA
 END IF
 CCV=CV
 CFX=CFA
 RETURN
END SUBROUTINE COVCX

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

FUNCTION VAR(SM,IS,KP,DRM,AAI,HP,IMAX1,LMEAN,CP,SP,CLOP,SLOP,LSAT,SROT)
! PROGRAMMED FEB 1985 BY C.C.TSCHERNING. UPDATE: SEP 16, 2013.
! THE FUNCTION COMPUTES THE VARIANCE OF A SIGNAL QUANTITY OF TYPE
! KP USING COVBX AND COVCX.

 USE m_geocol_data, ONLY : STEPN,COSSTN,SINSTN,STEPE,COSSTE,&
                           SINSTE,COST2P,SINT2P,NFILTE
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_data,        ONLY : LDIF
 USE m_geocol_data, ONLY : ALLREF,ALLGG,ALLCOL,ALLPRE,ALLTRA,ALLPR1,ALLERR,&
                           ALLVAR,ALLIN,LALLCO
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,DC,HCMAX,KCI
 USE m_geocol_data, ONLY : CCR,SIGMAX,CCV,NC1,NC2,LOCAL,LSUM
 USE m_data,        ONLY : KSAT,LTESTS,NWAR,LY
!USE m_geocol_data, ONLY : COVX,CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
 USE m_geocol_data, ONLY :      CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
                           NDP,NDQ,LX,LNX
 USE m_geocol_data, ONLY : STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE, COST2Q,SINT2
Q
 USE m_geocol_data, ONLY : HQD,RLATD,SINLATD,COSLATD,RLONGD,SINLOND,COSLOND
 USE m_geocol_data, ONLY : HQE,RLATE,SINLATE,COSLATE,RLONGE,SINLONE,COSLONE

IMPLICIT NONE

 LOGICAL ::      LMEAN,LSAT

 REAL(KIND=8) :: CI(24),CR(56),D(36),&
                 GM,AAI,DRM,RP,HP,CVV,STEQQN,&
                 VAR,COMEAN,CP,SP,CFC,CQ,SQ,RLATP,RLATQ,&
                 RLONP,RLONQ,CLOP,SLOP
```



```
 INTEGER ::      KI(37),N1,N2,IMAX1,KP,IS,I,J

 REAL(KIND=8), DIMENSION(2200) :: SM
 REAL(KIND=8), DIMENSION(3,3)  :: SROT
 REAL(KIND=8), DIMENSION(3,3,3,3)  :: COVX

!COMMON /CMCOV/CI(24),CR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HMAX,D(40),KI(37),N1,N2,LOCAL,LSUM
! CHANGE TO 2200 2010-11-24.

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GM,ITCOUN,LF,LT

!COMMON /CMEAQ/STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE,COST2Q,SINT2Q

!COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFA,KSAT(17,2),LSATPP,LSATQ,NDX1(5),NDX2(5)
,NDP,NDQ,NWAR,LY,LX(7,5),LNX(7,5),LTESTS

!COMMON /CALLCO/ALLREF(3,3),ALLGG(3,3),ALLCOL(3,3),ALLPRE(3,3),ALLTRA(3,3),ALLPR
1(3,3),ALLERR(3,3),ALLVAR(3,3),ALLIN(3,3),LALLCO

 CI=CCI
 CR=CCR
 D=DC
 KI=KCI
 N1=NC1
 N2=NC2
 GM=GMC
 STEQQN=STEQN
 RLONP=ATAN2(SLOP,CLOP)
 RLONQ=RLONP
 CI(8) = AAI
 CI(9) = (RE+DRM)**2
 CI(10)= DRM
 CI(20)= D1
 N1 = IMAX1
 KI(6) = KP
 KI(7) = KP
 KCI(6) = KP
 KCI(7) = KP
 RP = RE+HP
!     CALL COVBX(SM,.FALSE.,IS)
 KCI=KI
 CCI=CI
 CALL COVBX(SM,LSAT,IS)
!     write(*,*)' covb called LSAT,KP ',LSAT,KP
 CR(1) = D1
 CR(2) = HP
 CR(3) = HP
 CR(4) = SP
 CR(5) = SP
 CR(6) = CP
 CR(7) = CP
 CR(8) = D0
 CR(9) = D1
! CHANGE 2005-01-18.
 IF (LSAT) THEN
  CR(10)=D1
  CR(11)=D1
 ELSE
  CR(10)= GM/(RP*RP)
  CR(11) = CR(10)
 END IF
!     write(*,*)' stat covc, Kp= ',kp
 IF (.NOT.LMEAN) THEN
!      write(*,*)' covcx lsat ',is,lsat
 CCR=CR
  CALL COVCX(SM,CVV,COVX,IS,LSAT)
!      write(*,*)' covcx lsatx ',is,lsat
```

```
  IF (LSAT) THEN
   CALL COVROT(COVX,SROT,SROT)
   IF (LTESTS) WRITE(*,101)COVX
 101    FORMAT(6D12.4)
!       CVV=COVX(KSAT(KP,1),KSAT(KP,2),KSAT(KP,1),KSAT(KP,2))
   IF (LALLCO) THEN
! USED TO STORE ALL VARIANCES, IF SEVERAL DERIVATIVES ARE COMPUTED
! SIMULTANEOUSLY 2006-01-30.
! CONVERSION FACTORS TO MGAL**2 and E**2.
    IF (KP.EQ.6.OR.KP.EQ.7.OR.KP.EQ.2) CFC=1.0D10
    IF (KP.GT.7.OR.KP.EQ.5) CFC=1.0D18
    DO I=1,3
     DO J=1,3
      ALLVAR(I,J)=COVX(I,J,I,J)*CFC
     END DO
    END DO
   END IF
   IF (KP.NE.25) THEN
    CVV=COVX(KSAT(KP,1),KSAT(KP,2),KSAT(KP,1),KSAT(KP,2))
! CHANGE 2002-10-23.
   ELSE
! 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 (.FALSE.) WRITE(*,100) CVV,KSAT(KP,1),KSAT(KP,2),KP
 100    FORMAT(' CVV, KP ',D14.6,3I3)
  END IF
 ELSE
! CHANGE 2001-07-15.
  STEQQN=STEQN
  STEQN=STEPN
  COSSQN=COSSTN
  SINSQN=SINSTN
  STEQE=STEPE
  COSSQE=COSSTE
  SINSQE=SINSTE
  COST2Q=COST2P
  SINT2Q=SINT2P
  CQ=CP
  SQ=SP
  IF (LDIF) THEN
   RLATP=ATAN2(SP,CP)
   CQ=COS(RLATQ)
   SQ=SIN(RLATQ)
   HQD=HP
!  RLONGD=0.0D0
!  SINLOND=0.0d0
!  COSLOND=1.0D0
   HQE=HP
   RLATE=RLATD
   SINLATE=SINLATD
   COSLATE=COSLATD
   RLONGE=RLONGD
   SINLONE=SINLOND
   COSLONE=COSLOND
  END IF
  CVV=COMEAN(SM,IS,HP,HP,  CP,    SP,  CLOP,  SLOP,    CP,    SP,  CLOP,  SLOP,N
FILTE,NFILTE,LSAT,.TRUE.)
! CVV=COMEAN(SM,IS,HP,HP,  CP,    SP,    D1,    D0,    CP,    SP,    D1,    D0,N
FILTE,NFILTE,LSAT,.TRUE.)
! CHANGE 2011-12-28.
! LF,LF,LF,LSAT)
! write(*,*)' CVV ',CVV
! out-commented 2012-08-18.
 END IF
```



```
! CHANGE 2000-04-11 AND 2002-09-30 AND 2011-10-04  AND 2012-02-09 BY CCT.
      IF (LSAT) THEN
!IF (LSAT.AND.(.FALSE.)) THEN
! WRITE(*,*)' LSAT=LT '
  IF (KP.EQ.6.OR.KP.EQ.7.OR.KP.EQ.2) THEN
  CVV=CVV*1.0D10
! CONVERSION TO MGAL.
!      CVV=CVV*(CR(10)*1.0D5/RADSEC)**2
  ELSE
   IF (KP.GT.7.OR.KP.EQ.5) THEN
! SCALING FOR 2-ORDER DERIVATIVES (TO EU**2).
    CVV=CVV*1.0D18
! SCALING FOR 2*TXY. 2002-11-26. CHANGED 2013-03-05.
!   IF (KP.EQ.13)CVV=CVV*4.0D0
    IF (LTESTS) WRITE(*,*)' KP, CVV ',KP,CVV
   END IF
  END IF
 END IF
 VAR = CVV
! CHANGE 2001-07-15.
 STEQN=STEQQN
 CCI=CI
 CCR=CR
 DC=D
 KCI=KI

 END FUNCTION VAR

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 SUBROUTINE ATBA(A,B,C)
! PROGRAMMED AUG 89 BY C.C.TSCHERNING.
! THE SUBROUTINE WILL COMPUTE THE PRODUCT OF THE 3*3 MATRICES A TRANS-
! POSED, B AND A AND STORE THE RESULT IN C.
 IMPLICIT NONE

 INTEGER ::      J,K,N

 REAL(KIND=8), DIMENSION(3,3) :: A,B,C,D,E


! 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
!  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)
!
! 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
!  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)
!
 DO 50 K=1,3
 DO 50 J=1,3
   50 C(K,J)=E(K,J)
 RETURN
 END SUBROUTINE ATBA

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE SROT(CV,SA,CA,IDIM,LTP)
! PROGRAMMED AUG 89 BY C.C.TSCHERNING. CHANGED 1995.03.16 BY CCT.
! THE SUBROUTINE WILL FOR IDIM=1 COMPUTE THE PRODUCT OF THE ROTATION
! MATRIX AND THE 2 X 2 MATRIX CV AND FOR IDIM=2 THE MATRIX CV,
```

```
! ROTATED CLOCKWISE THE ANGLE A. SA=SIN(A), CA=COS(A), GIVEN.
! LTP IS TRUE IF CV MUST BE TRANSPOSED BEFORE MULTIPLICATION.
IMPLICIT NONE

 INTEGER ::      IDIM

 LOGICAL ::      LTP

 REAL(KIND=8) :: SA,CA,CV21,CV11,CV12,SA2,CA2,CS

 REAL(KIND=8), DIMENSION(2,2) :: CV

 IF (LTP) THEN
 CV21=CV(2,1)
 CV(2,1)=CV(1,2)
 CV(1,2)=CV21
 END IF
 IF (IDIM.EQ.1) THEN
 CV11=CA*CV(1,1)-SA*CV(2,1)
 CV12=-SA*CV(2,2)+CA*CV(1,2)
 CV21=CA*CV(2,1)+SA*CV(1,1)
 CV(2,2)=SA*CV(1,2)+CA*CV(2,2)
 CV(1,1)=CV11
 CV(2,1)=CV21
 CV(1,2)=CV12
 ELSE
 SA2=SA*SA
 CA2=CA*CA
 CS=CA*SA
 CV11=CA2*CV(1,1)-CS*(CV(2,1)+CV(1,2))+SA2*CV(2,2)
 CV12=(CA2-SA2)*CV(1,2)+CS*(CV(1,1)-CV(2,2))
 CV(2,2)=SA2*CV(1,1)+CS*(CV(1,2)+CV(2,1))+CA2*CV(2,2)
 CV(1,2)=CV12
 CV(2,1)=CV(1,2)
 CV(1,1)=CV11
 END IF
 IF (LTP) THEN
 CV21=CV(2,1)
 CV(2,1)=CV(1,2)
 CV(1,2)=CV21
 END IF
 END SUBROUTINE SROT

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE AXV(A,V)
! THE SUBRIUTINE WILL COMPUTE THE PRODUCT OF THE MATRIX A AND THE
! VECTOR V AND RETURN IT IN V. PROGRAMMED 1990.11.03 BY CCT.
IMPLICIT NONE

 INTEGER ::      I,J

 REAL(KIND=8), DIMENSION(3)   :: V,Y
 REAL(KIND=8), DIMENSION(3,3) :: A

 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 AXV

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 SUBROUTINE COVROT(COVX,SROTP,SROTQ)
! THE SUBROUTINE WILL COMPUTE THE ROTATED COVARIANCE MATRIV OR VECTOR
! USING THE ROTATION MATRICES SRORP, SROTQ ASSOCIATED WITH THE POINTS
```



```
! P, Q, RESPECTIVELY. SEE REF(I), SECTION 3.
! PROGRAMMED BY C.C.TSCHERNING, GEOPHYSICAL INSTITUTE, UNIVERSITY OF
! COPENHAGEN, JUNE, 1991.
! (I) TSCHERNING, C.C.: COMPUTATION OF COVARIANCES OF DERIVATIVES OF THE
!     ANOMALOUS GRAVITY POTENTIAL IN A ROTATED REFERENCE FRAME.
!     MANUSCRIPTA GEODAETICA, VOL. 18, NO. 3, PP. 115-123, 1993.
! LAST UPDATE 2013-03-25.
!
 USE m_data,        ONLY : KSAT,LTESTS,NWAR,LY
 USE m_geocol_data, ONLY :      CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
                           NDP,NDQ,LX,LNX
 IMPLICIT NONE

 INTEGER ::      NCASE,IM,JM,I,J

 REAL(KIND=8), DIMENSION(3,3,3,3) :: COVX
 REAL(KIND=8), DIMENSION(3,3) :: SROTP,SROTQ,A
 REAL(KIND=8), DIMENSION(3)   :: V

!COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFA,KSAT(17,2),LSATPP,LSATQ,NDX1(5),NDX2(5)
,NDP, NDQ,NWAR,LY,LX(7,5),LNX(7,5),LSATS

 NCASE=NDP+1+NDQ*3

 GO TO (801,802,803,804,805,806,807,808,809),NCASE
! 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
!
! 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
!
! 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
!
! 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
!
! 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
!
! 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
!
! 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
!
! 2 DERIV. IN P AND Q.
  809 CONTINUE
 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
!
  801 RETURN
 END SUBROUTINE COVROT

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE PAZIM(RLATP,RLONGP,COSLAP,SINLAP,COSLOP,SINLOP,CAZP,SAZP,COSDT,SINDT,
LTEST)
! THE SUBROUTINE WILL FROM A POINT WITH LATITUDE AND LONGITUDE
! SPECIFIED IN THE CALL PRODUCE THE CORRESPONDING VALUES IN A
! NEW POINT IN DISTANCE DT  AND AZIMUTH GIVEN BY
! COS AND SIN - CAZP, SAZP.
! PROGRAMMED BY C.C.TSCHERNING, OCT. 92. LAST CHANGE: 2002-10-24.
 USE m_geocol_data, ONLY : PW2

IMPLICIT NONE

 REAL(KIND=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.and.(.FALSE.)) WRITE(*,*)' PAZIM - LAT,LONG=',DLATP,DLONGP

 END SUBROUTINE PAZIM

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

LOGICAL FUNCTION CHECKC(NPOS)
! USING THE LAPLACE EQUATION TO CHECK TO COVARIANCES.
! PROGRAMMED 2002-10-07 BY C.C.TSCHERNING, LATEST UPDATE: 2005-04-16.
! INPUT:
!  NPOS - CALL - USED TO INDICATE FROM WHERE THE SUBROUTINE IS CALLED.
!  COVX - CSAT - HOLDS COVARIANCES. TWO FIRST SUBSCRIPTS REALTED
!                TO ONE POINT (P) AND THE LAST TWO TO A SECOND POINT (Q).
!                INDEX 1: EAST DERIVATIVE, 2: NORTH DERIVATIVE,
!                3: UP DERIVATIVE (RADIUS VECTOR).
! NDP,NDQ CSAT - NUMBER OF DERIVATIVES IN P, Q, RESPECTIVELY.
!
! OUTPUT
! NWAR - CSAT - NUMBER OF WARNINGS
! IF LOUT IS TRUE, OUTPUT OF LAPLACE EQUATION, SUM OF ABSOLUTE VALUE OF
!    THE 3 TERMS, THE 3 TERMS.
!
 USE m_geocol_data, ONLY : PW2
 USE m_data,        ONLY : KSAT,LTESTS,NWAR,LY
!USE m_geocol_data, ONLY : COVX,CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
 USE m_geocol_data, ONLY :      CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
                           NDP,NDQ,LX,LNX

IMPLICIT NONE

 INTEGER ::      I,J,NPOS,NCASE
```

```
 REAL(KIND=8) :: TEST1,TEST2,TEST4,TEST5,ATEST1,ATEST2
 REAL(KIND=8), DIMENSION(3,3,3,3) :: COVX

 LOGICAL ::      LOUT,CHECK,LF

!COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFX,KSAT(17,2),LSATPP,LSATQ,NDX1(5),NDX2(5)
,NDP,NDQ,NWAR,LY,LX(7,5),LNX(7,5),LTESTS

 LOUT=.TRUE.
!     LOUT=NWAR.LT.25
 LF=.FALSE.
 CHECKC=.TRUE.
 CHECK=.TRUE.
 J=1
 NCASE=NDP+1+NDQ*3
 GO TO (810,810,803,810,810,806,807,808,809),NCASE
! 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
  CHECK=LF
!      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
!
! 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
   CHECK=LF
   IF (LOUT.AND.NWAR.LE.50) THEN
! CHANGE 2010-03-10.
   write(*,*)' two in P 1 in Q '
!       WRITE(*,10)NPOS,I,J,TEST2,TEST5,COVX(1,1,I,1),COVX(2,2,I,1),&
!    *  COVX(3,3,I,1)
   NWAR=NWAR+1
   IF (NWAR.EQ.50) WRITE(*,*) ' WARNING WILL TERMINATE BECAUSE MORE THAN 50 '
   END IF
  END IF
 END DO
 GO TO 810
!
 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
  CHECK=LF
!      IF (LOUT.AND.NWAR.LE.50)
!    * 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
  IF (NWAR.EQ.50) WRITE(*,*) ' WARNING WILL TERMINATE BECAUSE MORE THAN 50 '
 END IF
 GO TO 810
!
! 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
   CHECK=LF
   IF (LOUT.AND.NWAR.LE.50) THEN
```



```
    NPOS=1
!        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
    IF (NWAR.EQ.50) WRITE(*,*) ' WARNING WILL TERMINATE BECAUSE MORE THAN 50 '
   END IF
  END IF
 END DO
 GO TO 810
!
! 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.PW2*1.0D-3) THEN
    CHECK=LF
    IF (LOUT.AND.NWAR.LE.-2) THEN
     NPOS=2
!         WRITE(*,10)NPOS,I,J,TEST1,TEST4,COVX(I,J,1,1),COVX(I,J,2,2),&
!     *    COVX(I,J,3,3)
   10     FORMAT(' WARNING42  ',I2,2I3,5D12.5)
     NWAR=NWAR+1
    END IF
   END IF
   IF (ATEST2.GT.TEST5*1.0D-4.AND.ATEST2.GT.PW2*1.0D-3) THEN
    CHECK=LF
    IF (LOUT.AND.NWAR.LE.-2) THEN
     NPOS=3
     WRITE(*,11)NPOS,I,J,TEST2,TEST5,COVX(1,1,I,J),COVX(2,2,I,J),&
     COVX(3,3,I,J),CHECK
   11     FORMAT(' WARNING43 ',I2,2I3,5D12.5,L2)
     NWAR=NWAR+1
    END IF
   END IF
  END DO
 END DO
  810 CHECKC=CHECK
 RETURN

END FUNCTION CHECKC

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE DENDEF(NMAX,LINTER,LWRSOL,LPARAM,LPOT,LOC_LBIPOT,LBIN,LINSOL,LDENOL,L
SKIPL,RRE)
! THE SUBROUTINE WILL INPUT PARAMETERS FOR AND DEFINE
! A DENSITY MODEL. MOVED FROM MAIN PROGRAM 2004-11-10.
!
! INPUT OF EXPONENT OF WEIGHT FACTOR ON HARMONIC DENSITY,
! RADIUS OF SPHERE WITHIN WICH MASSES ARE LOCATED IN M. CF. REF(F),
! SECTION 3,SALE FACTOR AND VALUE OF LOGICAL VARIABLE LNPOT
! TRUE, IF NEW SET OF COEFFICIENTS ARE TO BE USED FOR THE DENSITY
! COMPUTATIONS, (DIFFERENCES BETWEEN THE ORIGINAL COEFFICIENTS AND
! COEFFICIENTS OF A TOPOGRAPHIC-ISOSTATIC REDUCTION POTENTIAL).
! IF POTENTIAL COEFFICIENTS NOT ALREADY STORED ON BINARY FORM
! ALSO THE NAME OF THE FILE TO BE CONNECTED TO UNIT 3.
! THIS ONLY APPLIES IF LPOT IS TRUE.

 USE m_params, ONLY : NCOEFF
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_geocol_data, ONLY : FG,FJ,LP,OMEGA2,IORDER
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,DC,HCMAX,KCI
 USE m_geocol_data, ONLY : CCR,SIGMAX,CCV,NC1,NC2,LOCAL,LSUM,CFA
 USE m_data,        ONLY : OLDT,OLDR,LFIRST
```

```
 USE m_geocol_data, ONLY : COFF
! QUASI NORMALIZED SPHERICAL HARMONIC COEFFICIENTS, UNITLESS.
 USE m_geocol_data, ONLY : C20IN,G1,G2,CM3,CMM2,CM1

IMPLICIT NONE

 LOGICAL ::      LINTER,LWRSOL,LNCOF,       LBIN,LFORM,&
                        LPARAM,LPOT,LOC_LBIPOT,LINSOL,&
                 LDENOL,LSKIPL

 INTEGER ::      I,IEHD,ICHAR,IJ1,IJ,MIJ,NMAX

 REAL(KIND=8) :: RRE,DSCALE,FACT

 CHARACTER(LEN=128), DIMENSION(9) :: FMT
 CHARACTER(LEN=128), DIMENSION(2) :: PNAME

!COMMON /CMCOV/CCI(24),CCR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HCMAX,CCV(2,2),DC(36),KCI(37),NC1,NC2,LOCAL,LSUM
! CHANGE TO 2200 2010-11-24.

!COMMON /GPOTC0/C20IN,G1(3),G2(3,3),CM3,CMM2,CM1
! C20IN HOLDS C20, G1 THE FIRST DERIVATIVES, G2 THE SECOND DERIVATIVES.
! COMMON VARIABLES USED IN COVAX. SEE THIS SUBROUTINE FOR VARIABLES.

!COMMON /GPOTC1/OLDT,OLDR,CFA,IGQ(12),LFIRST,HP9000
! COMMON VARIABLES USED IN GPOTDR, SETCM ,LOADCM, PRED AND CXPARM.

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT
! COMMON CONSTANTS D0=0.0D0  ETC.

!COMMON /GRAVCC/FG(30),FJ(30),LP(2),OMEGA2,IORDER
! COMMON VARIABLES USED IN GRAVC AND RGRAV, HOLDING I.E. COEFFICI-
! ENTS OF LEGENDRE SERIES OF NORMAL POTENTIAL AND NORMAL GRAVITY
! FORMULA.

 ICHAR=1
 IF (LINTER) WRITE(6,*)' INPUT DENSITY SPECIF.'
  239 FORMAT(I3,2F10.1,L2)
 READ(5,*)IEHD,RRE,DSCALE,LNCOF
 IF (LWRSOL) WRITE(17,239)IEHD,RRE,DSCALE,LNCOF
 KCI(32)=IEHD-1
! THE USED POWER OF R IS ONE SMALLER, BECAUSE R GENERALLY IS RAISED
! TO THE POWER OF I+1.
 CCI(15)=1.5+IEHD/2.0
 CCI(13)=DSCALE/(6.67E-8*PI*RRE**(3-IEHD))
 IF (LPOT) THEN
  IF (.NOT.(LBIN.OR.LOC_LBIPOT.OR.LINSOL.OR.LPARAM.OR.LDENOL)) THEN
!
   IF (LINTER) WRITE(6,*)' INPUT NAME OF FILE WITH COEFF.'
   READ(5,2103)PNAME(1)
   IF (LWRSOL) WRITE(17,2103)(PNAME(I),I=1,ICHAR)
  END IF
  IF (.NOT.LDENOL) THEN
   IF ((.NOT.LPARAM).AND.(.NOT.LOC_LBIPOT)) OPEN(3,FILE=PNAME(1),&
   STATUS='UNKNOWN',FORM='UNFORMATTED')
   IF (.NOT.(LBIN.OR.LOC_LBIPOT.OR.LINSOL.OR.LPARAM)) THEN
    WRITE(3)COFF
   END IF
  END IF
  LDENOL=LT
!
  IF (LNCOF)  THEN
! INPUT OF LBIN, TRUE IF THE COEFFICIENTS ARE ON BINARY FORM, LFORM,
! TRUE IF THE FORMAT OF THE COEFFICIENTS ARE INPUT, THE NAME OF THE
! OF THE FILE HOLDING THE COEFFICIENTS AND IF LFORM IS TRUE THE FORMAT.
!
   IF (LINTER) WRITE(6,*)' INPUT BIN, FORM'
   READ(5,*)LBIN,LFORM
```



```
   IF (LWRSOL) WRITE(17,105)LBIN,LFORM
  105   FORMAT(8L2)
   IF (LINTER) WRITE(6,*)' INPUT NAME OF FILE'
   READ(5,2103)PNAME(1)
   WRITE(6,241)(PNAME(I),I=1,ICHAR)
  241   FORMAT(' NEW COEFFICIENTS INPUT FROM FILE ',2A128)
   IF (LWRSOL) WRITE(17,2103)(PNAME(I),I=1,ICHAR)
 2103   FORMAT(A128)
   IF (LINTER.AND.LFORM) WRITE(6,*)' INPUT FORMAT'
   IF (LFORM) READ(5,103)FMT(1)
  103   FORMAT(I5,I4,2E17.9)
   IF (LWRSOL.AND.LFORM) WRITE(17,103)FMT(1)
   CALL LOADCS(PNAME,FMT,NMAX,LFORM,LBIN,LSKIPL)
   IF (.NOT.LBIN) CALL SETCM(NMAX,LBIN)
  END IF
!
! WE NOW MODIFY THE GM, AX AND OMEGA,  C(0,0), C(2,0), C(4,0) AND C(6,0)
! SO THAT WE WORK IN SPHERICAL APPROXIMATION WITH A DENSITY CONTRAST
! FUNCTION. SEE ALSO REF.(F) SECTION 3. NOTE THET WE USE THE MEAN EARTH
! RADIUS AND NOT THE RADIUS RRE. THIS IS BECAUSE THE ORIGINAL COEF-
! FICIENTS ARE MULTIPLIED WITH THIS.
  CM3=CM3*CCI(13)
  CMM2=RE
  CM1=D0
  COFF(1)=D0
  COFF(5)=COFF(5)-FJ(18)
  COFF(17)=COFF(17)-FJ(20)
  COFF(37)=COFF(37)-FJ(22)
  IJ1=4
  DO IJ=2,NMAX
   IJ1=IJ1+1
   FACT=(IJ+0.5E0)*(IJ+CCI(15))
   COFF(IJ1)=COFF(IJ1)*FACT
   DO MIJ=1,IJ
    COFF(IJ1+1)=COFF(IJ1+1)*FACT
    COFF(IJ1+2)=COFF(IJ1+2)*FACT
    IJ1=IJ1+2
   END DO
  END DO
!
 END IF
 WRITE(6,9776)IEHD,DSCALE,RRE
 9776 FORMAT(' WEIGHT FACTOR ON HARMONIC DENSITIES IS R**(',I3,&
 ')*',F10.1,/,' DENSITY DISTRIBUTION IS WITHIN SPHERE WITH ',&
 'RADIUS ',F10.2,' M',/)
 IF (LPOT) WRITE(6,9777)(COFF(IJ),IJ=1,17)
 9777 FORMAT(' HARMONIC COEFFICIENTS FROM C(0,0) TO C(4,0)',/,&
 5(4E15.7,/))
!
 RETURN
END SUBROUTINE DENDEF

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE QUATMAT(quat)

!     Programmed sept. 2004 by M.L.Veicherts
!      Creates the DCM called ROTMAT from quaternion array input:
!     q = (xi,yj,zk,scalar)

 USE m_params,      ONLY : NSAT
 USE m_geocol_data, ONLY : SATROT

IMPLICIT NONE

 REAL(KIND=8) :: quat(4),d2,q11,q12,q13,q14,q22,q23,q24,q33,q34,q44

 d2 = 2.0d0
```

```
 q11 = quat(1) * quat(1)
 q12 = quat(1) * quat(2)
 q13 = quat(1) * quat(3)
 q14 = quat(1) * quat(4)

 q22 = quat(2) * quat(2)
 q23 = quat(2) * quat(3)
 q24 = quat(2) * quat(4)

 q33 = quat(3) * quat(3)
 q34 = quat(3) * quat(4)

 q44 = quat(4) * quat(4)

 SATROT(1,1) =  q11 - q22 - q33 + q44
 SATROT(1,2) =  d2 * (q12 + q34)
 SATROT(1,3) =  d2 * (q13 - q24)
 SATROT(2,1) =  d2 * (q12 - q34)
 SATROT(2,2) =  - q11 + q22 - q33 + q44
 SATROT(2,3) =  d2 * (q23 + q14)
 SATROT(3,1) =  d2 * (q13 + q24)
 SATROT(3,2) =  d2 * (q23 - q14)
 SATROT(3,3) =  - q11 - q22 + q33 + q44

END SUBROUTINE QUATMAT

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

FUNCTION COVPQ(SM,IS,KP,DRM,AAI,IMAX1,LMEAN,LSAT,PREDCO,PREDCP)
! PROGRAMMED  AUG 2005 BY C.C.TSCHERNING. UPDATE: JAN 07, 2013.
! THE FUNCTION COMPUTES THE COVARIANCE OF  TWO SIGNAL QUANTITES OF TYPE
! KP USING COVBX AND COVCX.

 USE m_geocol_data, ONLY : STEPN,COSSTN,SINSTN,STEPE,COSSTE,&
                           SINSTE,COST2P,SINT2P,NFILTE
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN
 USE m_data,        ONLY : CCI,SIGMA,SIGMA0,DC,HCMAX,KCI
 USE m_geocol_data, ONLY : CCR,SIGMAX,CCV,NC1,NC2,LOCAL,LSUM
 USE m_data,        ONLY : KSAT,LTESTS,NWAR,LY
!USE m_geocol_data, ONLY : COVX,CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
 USE m_geocol_data, ONLY :      CIX,CFX,LSATPP,LSATQ,NDX1,NDX2,&
                           NDP,NDQ,LX,LNX
 USE m_geocol_data, ONLY : STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE, COST2Q,SINT2
Q

IMPLICIT NONE

 LOGICAL      :: LMEAN,LSAT

 REAL(KIND=8) :: CI(24),CR(56),D(36),&
                 GM,AAI,DRM,RP,HP,CVV,STEQQN,&
                 COMEAN,CP,SP,HMAX

 REAL(KIND=8) :: COVPQ,RQ,HQ,RLATP,RLATQ,SINLOP,SINLOQ,COSLOP,&
                 COSLOQ,RLONGQ,RLONGP,CQ,SQ,SD,CD,T

 INTEGER ::      KI(37),N1,N2,IMAX1,KP,IS,I61,I62

 REAL(KIND=8), DIMENSION(16)   :: PREDCO,PREDCP
 REAL(KIND=8), DIMENSION(3,3,3,3) :: COVX
 REAL(KIND=8), DIMENSION(2200) :: SM
 REAL(KIND=8), DIMENSION(3,3)  :: SROTP,SROTQ

!COMMON /CMCOV/CI(24),CR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HMAX,D(40),KI(37),N1,N2,LOCAL,LSUM
! CHANGE TO 2200 2010-11-24.

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GM,ITCOUN,LF,LT
```



```
!COMMON /CMEAQ/STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE,COST2Q,SINT2Q

!COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFA,KSAT(17,2),LSATPP,LSATQ,NDX1(5),NDX2(5)
,NDP,NDQ,NWAR,LY,LX(7,5),LNX(7,5),LTESTS

 HMAX=HCMAX
 CI=CCI
 CR=CCR
 N1=NC1
 N2=NC2
 KI=KCI
 D=DC
 GM=GMC
 CI(8) = AAI
 CI(9) = (RE+DRM)**2
 CI(10)= DRM
 CI(20)= D1
 N1 = IMAX1
 KI(6) = KP
 KI(7) = KP
 HP=PREDCP(7)
 HQ=PREDCO(7)
 RP = RE+HP
 RQ = RE+HQ
 RLATP=PREDCP(1)
 CP=PREDCP(2)
 SP=PREDCP(3)
 RLONGP=PREDCP(4)
 COSLOP=PREDCO(5)
 SINLOP=PREDCP(6)
 RLATQ=PREDCO(1)
 CQ=PREDCO(2)
 SQ=PREDCO(3)
 RLONGQ=PREDCO(4)
 COSLOQ=PREDCO(5)
 SINLOQ=PREDCO(6)
 CD=COS(RLONGP-RLONGQ)
 SD=SIN(RLONGP-RLONGQ)
 T=SP*SQ+CP*CQ*CD
 KCI=KI
 CCI=CI
 CALL COVBX(SM,LSAT,IS)
 CR(1) = T
 CR(2) = HP
 CR(3) = HP
 CR(4) = SP
 CR(5) = SQ
 CR(6) = CP
 CR(7) = CQ
 CR(8) = SD
 CR(9) = CD
 IF (LSAT) THEN
  CR(10)=D1
  CR(11)=D1
 ELSE
  CR(10)= GM/(RP*RQ)
  CR(11) = CR(10)
 END IF
 CCR=CR
 IF (.NOT.LMEAN) THEN
  CCI=CI
! write(*,*)' 17512 call covcx '
  CALL COVCX(SM,CVV,COVX,IS,LSAT)
  IF (LSAT) THEN
   DO I61=1,3
    DO I62=1,3
     SROTQ(I61,I62)=PREDCO(7+(I61-1)*3+I62)
     SROTP(I61,I62)=PREDCP(7+(I61-1)*3+I62)
!    SROTQ(I61,I62)=PREDCO(7+I61*3+I62)
```

```
!    SROTP(I61,I62)=PREDCP(7+I61*3+I62)
    END DO
   END DO
   CALL COVROT(COVX,SROTP,SROTQ)
   IF (LTESTS) WRITE(*,101)COVX
 101    FORMAT(6D12.4)
   IF (KP.NE.25) THEN
    CVV=COVX(KSAT(KP,1),KSAT(KP,2),KSAT(KP,1),KSAT(KP,2))
   ELSE
! 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
! NOT FULLY IMPLEMENTED.
  WRITE(*,*)' WARNING44: MEAN VALUES NOT IMPLEMENTED '
  STEQQN=STEQN
  STEQN=STEPN
  COSSQN=COSSTN
  SINSQN=SINSTN
  STEQE=STEPE
  COSSQE=COSSTE
  SINSQE=SINSTE
  COST2Q=COST2P
  SINT2Q=SINT2P
  CVV=COMEAN(SM,IS,        CP,    SP,COSLOP,SINLOP,    CQ,    SQ,COSLOQ,SINLOQ,N
FILTE,NFILTE,LSAT,.FALSE.)
 END IF
! 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
! CONVERSION TO MGAL.
!      CVV=CVV*(CR(10)*1.0D5/RADSEC)**2
  ELSE
   IF (KP.GT.7.OR.KP.EQ.5) THEN
!
    CVV=CVV*1.0D18
! SCALING FOR 2*TXY. 2002-11-26. CHANGED 2013-03-05
!   IF (KP.EQ.13)CVV=CVV*4.0D0
    IF (LTESTS) WRITE(*,*)' KP, CVV ',KP,CVV
   END IF
  END IF
 END IF
 COVPQ = CVV
! CHANGE 2001-07-15.
 STEQN=STEQQN
END FUNCTION COVPQ

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 SUBROUTINE RESTORE_CH(NJ,ILR,lOBS,lERR_OBS,lERR_COV,LMTEST)
! NEW SUBROUTINE ADDED 2012-02-29 WRITTEN BY MV&CCT. LAST CHANGE 2013-01-07.
! FOR IMPLEMENTATION BACKGROUND SEE REF(J).

! NJ    absolute column number of 1. col in C!
! ILR   last reduced column of observation related covariances.
! remaining required information is placed in m_cholsol:
! N,NN,NC,NBB,NB

! This subroutine reformats the blocks-wise matrix reporesentation into a chunk-
wise representation
! In order to utÃlise a new set of routines to cholesky factorise and solve the
 system of normal eq.
```



```
 use m_geocol_data, only : C
 use m_cholsol,     only : alloc,dealloc,chunk_write,chunk_read,CH,   &
                           add_files_col,generate_block_givens,       &
                           chunk,lTEST,chunk_file_close,              &
                           lPARAM,N,NN,NC,NBB,NB,block_write,lt,lf,   &
                           get_col_num,single_row_read,chunk_reset,   &
                           single_col_readx

 IMPLICIT NONE

 INTEGER           :: NX,i,j,ic,ib,k,NCX,NBX,NJ,imin,imax,ish,    &
                      jb_abs,ILR,istart,iNN,NXX
 LOGICAL           :: lERR_OBS,lERR_COV,lOBS,LMTEST,lFirst_CHUNK,lstore
! lstore will be true when coefficients have to be stored in a chunk. This
! is not needed, if the coefficients already have been reduced.
!real(KIND=8)      :: a_test

 CHARACTER(LEN=16) :: tag

 REAL(KIND=8), DIMENSION(:,:,:), ALLOCATABLE :: A_BLOCS

 LTEST=(.NOT.lOBS).AND.LMTEST
 lstore=(NJ.GT.ILR.and.lobs).or.(.not.lobs)
 if (lf) write(*,*)' 17712 lstore,NJ,ILR= ',lstore,nj,ilr
 if (lERR_COV) then
  write(*,*)' error-covariances not yet implemented, stop '
  stop
 end if

 imin = 1
 lFIRST_CHUNK = lf
! NN is the number of columns within a block. NBB is the number of blocks within

! a chunk.
! NCX is the chunk column number. NBX is the block column number within the chun
k.
 if (ILR.EQ.0.or.lobs) then
   call get_col_num(Nj,NCX,NBX,NX)   ! find corresponding numbers of chunk,relat
ive block and relative single data num
   if (.not.lobs)write(*,*)' from ilr=0: NX= ',NX
 else
   call get_col_num(Nj+ILR,NCX,NBX,NXX)   ! find corresponding numbers of chunk,
relative block and relative single data num
!  write(*,*)' get_col_n1: Nj,ILR,NCX,NBX,NXX',Nj,ILR,&
!  NCX,NBX,NXX
! change 2012-08-17.
   NX=NXX
   call get_col_num(Nj+CEILING(1.0*(ILR+1)/(NN*NBB))*NN*NBB,NCX,NBX,NX)   ! find
 corresponding numbers of chunk,relative block and relative single data num
!  write(*,*)' get_col_n: Nj,CE,NCX,NBX,NX',Nj,CEILING(1.0*ILR/(NN*NBB)),&
!  NCX,NBX,NX
 end if
! jb_abs is the total number of blocks in a column.
 IF (lOBS) THEN
   jb_abs = (NCX-1)*NBB + NBX
 ELSE
!    jb_abs = (ILR-MOD(ILR,NN))/NN+1
   jb_abs = (ILR+1)/NN + MERGE(1,0,MOD(ILR+1,NN).GT.0)   ! evt. ILR + 1 ?
 END IF
! statement out-commented since it was placed erroneously 2012-08-10.
! jb_abs = (NCX-1)*NBB + NBX
 ish = CEILING(1.0*ILR/(NN*NBB))
 if (LTEST) write(*,*) ' Nj,ILR,NCX,NBX,NX,jb_abs,ish ' , Nj,ILR,NCX,NBX,NX,jb_a
bs,ish
!if (LMTEST) write(*,*) ' Nj,ILR,NCX,NBX,NX,jb_abs,ish ' , Nj,ILR,NCX,NBX,NX,jb_
abs,ish
! ILR/(NN*NBB)   !  OBS set this right = number of chunks holding neq!
```

```
! initiating blocks in new chunks and creating appropiate files:
 if (MOD(NJ,NN*NBB).EQ.1) then
   lFIRST_CHUNK = lt
!   imin = 1
   imax = NCX

! ------ making file for solution file:
   if (NJ.EQ.1) then
    call add_files_col(imin,imax,NCX,NCX,lf,lf,lf,lt)  ! making solution file
!  if (NJ+ILR.EQ.1) call add_files_col(imin,imax,NCX,NCX,lf,lf,lf,lt)  ! making
solution file
    if (LMTEST) write(*,*)' solution file created '
   end if

! ------ making chunk files for normal eq.
   if (lOBS) then
     tag = 'neq'
     call add_files_col(imin,imax,NCX,NCX,lt,lf,lf,lf)
     ish = 1
   end if
! ------ making chunk files for error estimates
   if (lERR_OBS) then    ! Error of OBS/PARAMS
     tag = 'err'
     call add_files_col(imin,imax,NCX,ish,lf,lT,lf,lf)
   end if
! ------ making chunk files for error covariances   ! NOT TESTED
   if (lERR_COV) then        ! error covariances - triangular again
    write(*,*)' lERR_COV ',LERR_COV
     imin = ish
     imax = ish + 1
     tag  = 'errcov'
     call add_files_col(imin,imax,NCX,ish,lf,lt,lt,lf)
     imin = ish+1
   end if
 end if

! create container (A_BLOCS) to reorder C array:
 allocate(A_BLOCS(jb_abs,NN,NN))  ! NN equals IBSS in geocol!
 A_BLOCS = 0.d0
 k = 0
 if (lERR_OBS) then
   do j = 1,NN
     do ib = imin,jb_abs     ! imin may !=1 when lERR_COV is true
       if (ib.EQ.jb_abs) then
         iNN = NXX
       else
         iNN = NN
       end if
       do i = 1,iNN
!        if (ib.LT.jb_abs) then
!          k = k + 1
!          A_BLOCS(ib,i,j) = C(k)
!        else
!          if (i.LE.NX-1) then
         k = k + 1
         A_BLOCS(ib,i,j) = C(k)
!        if (LMTEST.and.ib.eq.jb_abs.and.i.EQ.iNN) write(*,*)&
!       ' ib,i,j,k,C(k) ',ib,i,j,k,C(k)
!          end if
!        end if
! change 2012-05-01.
!          if (k.lt.37) write(*,112) ib,j,k,C(k)
           if (k.lt.0) write(*,112) ib,j,k,C(k)
 112        format(' C-ERROR: ib,j,k,C(k) ',3I5,D14.7)
!         if (Nj.EQ.1.AND.jb_abs.EQ.1) call block_write(NN,A_BLOCS(ib,:,:))
!         if (k.EQ.1) write(*,*) ' A_TEST ib,i,j,C(k) ',ib,i,j,C(k)
       end do
     end do
   end do
```



```
 else     ! for ordinary OBS:  ERR_COV is not considered for now! :(
   if (lstore) then
   do j = 1,NN
     do ib = imin,jb_abs     ! imin may !=1 when lERR_COV is true
       do i = 1,NN
         if (ib.LT.jb_abs.OR.i.LE.j) then
           k = k + 1
           A_BLOCS(ib,i,j) = C(k)
!          if (k.lt.36) write(*,111) ib,j,k,C(k)
           if (k.eq.0) write(*,111) ib,j,k,C(k)
 111        format(' C-OBS  : ib,j,k,C(k) ',3I5,D14.7)
         end if
       end do
     end do
   end do
   else
    if (lf) write(*,*)' block ',ib,' not stored '
   end if
 end if

! put A_BLOCS in chunks:
 if (.not.allocated(CH)) then
   allocate(CH(NCX,NCX))
   if (LF) write(*,*)' CH allocated, NCX ',NCX
 else
   write(*,*)' WARNING - CHUNK already allocated! ',NCX,NCX,size(CH)
   deallocate(CH)
   allocate(CH(NCX,NCX))
   write(*,*)' CHUNK forced allocated ? ',size(CH)
 end if

 k = 0
 if (ILR.EQ.0.or.(.not.lstore)) then
   istart = NCX
 else
   istart = ish
 end if
!write(*,*)' istart ',istart
 do ic = imin,istart
   if (LF) write(*,*)' chunk indices for ch alloc - ic,NCX:  ',ic,NCX
   call alloc(ic,NCX)
   if (.NOT.lFIRST_CHUNK) then
!    if (LMTEST) write(*,*)' calling read ch ',ic,NCX
     call chunk_read(ic,NCX)
   else
     if (LMTEST) write(*,*)' reset chunk ',ic,NCX
     call chunk_reset(ic,NCX)
   end if
   if (lstore) then
   do ib = 1,NBB
     k = k + 1
     if (k.LE.jb_abs) then
       CH(ic,NCX)%BL(ib,NBX)%A(:,:) = A_BLOCS(k,:,:)
        if (lERR_OBS) then
!         write(*,*) ' corresponding chunk indices: ',ic,NCX,ib,NBX
!         if (ic.eq.istart) call block_write(NN,CH(ic,NCX)%BL(ib,NBX)%A)
        end if
!       if (lERR_OBS.AND.k.EQ.1) write(*,*) ' A_TEST ic,NCX,ib,NBX,A_BLOCS(k,:,:
): ', ic,NCX,ib,NBX,A_BLOCS(k,1,1)
!       if (ic.EQ.1.AND.NCX.EQ.5.AND.ib.EQ.10.AND.NBX.EQ.1) then
!         write(*,*)' Abloc overfoersel check: ',k,CH(ic,NCX)%BL(ib,NBX)%A(1,:)

!       end if
!     else
!       CH(ic,NCX)%BL(ib,NBX)%A = 0.d0
     end if
   end do
   call chunk_write(ic,NCX)
   call dealloc(ic,NCX)
```

```
   end if
 end do
! if (lERR_OBS) then
! NX = 15
! if (lOBS) then
!   NCX = 4
! else
!   NCX = 8
! end if
! write(*,*)' from RESTORE: col 15: ',NX,NCX
! call single_col_readx(NX,NCX)

 DEALLOCATE(CH)
 DEALLOCATE(A_BLOCS)

 END SUBROUTINE RESTORE_CH

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SUBROUTINE SPHARMA(SLAT,CLAT,SLON,CLON,SJLO,CJLO,R,I0,J0,IDIF,LFULL,LSPHAP,LCOS)
! CALCULATION OF THE VALUES AND THE UP TO 2. ORDER DERIVATIVES
! OF SOLID SPHERICAL HARMONIC FUNCTIONS  PIM0=Y(I0,J0)(LAT,LON,R) USING
! RECURSION BASED ON PIM1=Y(I0-1,J0), PIM2=Y(I0-2,J0)  WHEN J0 .NE. J0.
! OTHERWISE THE RECURSION IS BASED ON PII=Y(I0-1,J0-1) AND
! Y(I0-2,J0-1). THE CALCULATION OF FIRST ORDER DERIVATIVES AT THE
! POLES IS DONE USING A RECURSION FORMULAE, WHERE THE COS(LAT) THEN IS
! ELIMINATED. THE SECOND ORDER DERIVATIVE WITH RESPECT TO X IS AT
! THE POLES CALCULATED USING THE LAPLACE EQUATION.
! PROGRAMMED FEBRUARY 1999  BY C.C.TSCHERNING. LAST CHANGE 2011-08-05.
! REFERENCES:
! TSCHERNING, C.C.: ON THE CHAIN-RULE METHOD FOR COMPUTING POTENTIAL
! DERIVATIVES. MANUSCRIPTA GEODAETICA, VOL. 1, PP. 125-141, 1976.
! TSCHERNING, C.C. AND K.PODER: SOME GEODETIC APPLICATIONS OF CLENSHAW
! SUMMATION. BOLLETINO DI GEODESIA E SCIENZE AFFINI, VOL. XLI, NO. 4,
! PP. 349-375, 1982.
!
! VARIABLES AT CALL: SLAT, CLAT: SINE AND COSINE OF LATITUDE, R THE
! SIZE OF THE RADIUS VECTOR, IDIF THE MAXIMAL ORDER OF DIFFERENTIATION
! (UP TO 2),  CJLO, CJLO: COS AND SIN OF J*LONGITUDE,
! LFULL A LOGICAL VARIABLE TRUE IF FULLY NORMALIZED FUNCTIONS
! ARE USED.
! LCOS, TRUE IF THE ORDER IS POSITIVE OR ZERO, IN WHICH CASE
! THE RECURSION ELEMENTS NOT ARE OUTPUT TO UNIT 98. THEY HAVE TO
! BE USED AGAIN WHEN LCOS IS FALSE.
! LAST CHANGE 2011-08-11 BY CCT.

 USE m_params,      ONLY : IIMAX,NSPHAR
 USE m_geocol_data, ONLY : PII,PIM0,PIM1,PIM2,DLP,DLP0,DLP1,DLP2,DAP,DAP0,DAP1,&
                           DAP2,DDAP,DDAP0,DDAP1,DDAP2,DDAL0,DDAL1,VI,ROOT0
 USE m_geocol_data, ONLY : SUMIJ,CCCIJ,SQ2,YS,YC,VV,V1,GS,GC,DDS,DDC,SN2,AXS,&
                           GMS,IIOLD,JOLD,IR,LSPHAR,LTSPH
 USE m_data,        ONLY : D0,D1,D2,D3,D4,D5,RE,GMC,RADSEC,PI,LT,LF,ITCOUN

IMPLICIT NONE

 REAL(KIND=8) :: R,Q,DDAL,SLON,CLON,SJL1,&
                 RQ,CLAT,SLAT,A,DDAL2,B,PM,Q2,Q3,V,CJLO,SJLO,DDC0,FACT

 INTEGER ::      I,I0,J,J0,J1,IDIF,K,N,IR1

 LOGICAL ::      LFULL,LSPHAP,LCOS

 REAL(KIND=8), DIMENSION(20) :: CR

!COMMON /DCON/D0,D1,D2,D3,D4,D5,RE,RADSEC,PI,GMC,ITCOUN,LF,LT
!COMMON /CON3/SUMIJ((NSPHAR+1)**2),CCCIJ((NSPHAR+1)**2),SQ2,YS,YC,VV,V1,GS(3),GC
(3),DDS(3,3),SN2(0:NSPHAR),AXS,GMS,DDC(3,3),IIOLD,JOLD,IR,LSPHAR,LTSPH
!
 IF (I0.NE.0.) THEN
```



```
! GET VALUES OF SPHERICAL HARMONIC RECURSION ELEMENTS FOR OBSERVATION IR.
  READ(98,REC=IR)CR
  PII=CR(1)
  PIM0=CR(2)
  PIM1=CR(3)
  PIM2=CR(4)
  DLP=CR(5)
  DLP0=CR(6)
  DLP1=CR(7)
  DLP2=CR(8)
  DAP=CR(9)
  DAP0=CR(10)
  DAP1=CR(11)
  DAP2=CR(12)
  DDAP=CR(13)
  DDAP0=CR(14)
  DDAP1=CR(15)
  DDAP2=CR(16)
  DDAL0=CR(17)
  DDAL1=CR(18)
  CJLO=CR(19)
  SJLO=CR(20)
 ELSE
  CJLO=D1
  SJLO=D0
  IIOLD=-1
  JOLD=-1
  PIM0=D1
 END IF
 IF (LSPHAP) THEN
  Q=RE/R
 ELSE
  Q=AXS/R
 END IF
 J=J0
 I=I0
 J1=J+1
 IF (I.EQ.J) THEN
!      IF (J.NE.(JOLD+1)) WRITE(*,*)' WARNING45 J '
  PIM2=D0
  PIM1=D0
!      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 (J.NE.0) THEN
! CALCULATING COS(J*LON) and SIN(J*LON) FROM COS((J-1)*LON) AND SIN.
   SJL1=SJLO
   SJLO=SJLO*CLON+CJLO*SLON
   CJLO=CJLO*CLON-SJL1*SLON
  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 (IR.EQ.0) WRITE(*,110)I0,J0,CJLO,SJLO
 110   FORMAT(' spharma, i0,j0,CJLO,SJLO ',2i4,3f7.4)
!       IF (J.NE.JOLD.OR.I.NE.(IIOLD+1)) WRITE(*,*)' WARNING46 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
! CORRECTION 1999-02-28 BY CCT - FORGOTTEM UNTIL 2000-04-25.
!      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
 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
! 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
! 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
!
 END IF
 IF (.NOT.LCOS.OR.J0.EQ.0) THEN
 IF (IR.EQ.0) WRITE(*,*)' WRITE 98 ',I0,J0
 IR1=1
 CR(IR1)=PII
 CR(IR1+1)=PIM0
 CR(IR1+2)=PIM1
 CR(IR1+3)=PIM2
 CR(IR1+4)=DLP
 CR(IR1+5)=DLP0
 CR(IR1+6)=DLP1
 CR(IR1+7)=DLP2
 CR(IR1+8)=DAP
```

```
 CR(IR1+9)=DAP0
 CR(IR1+10)=DAP1
 CR(IR1+11)=DAP2
 CR(IR1+12)=DDAP
 CR(IR1+13)=DDAP0
 CR(IR1+14)=DDAP1
 CR(IR1+15)=DDAP2
 CR(IR1+16)=DDAL0
 CR(IR1+17)=DDAL1
 CR(19)=CJLO
 CR(20)=SJLO
 WRITE(98,REC=IR)CR
 END IF
 RETURN
END SUBROUTINE SPHARMA

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! INSERTING NEW CHOLESKY FACTORISATION ROUTINES

! Choleski redution of large equation systems:
!  A*x    = b
!  C*CT*x = b
!  C * y  = b
!  CT * x = y

! N is the rank of A (or C) matrix

      subroutine cholsol(NXX,Nstartx,Nparamx,Nerrx,lOBSx,lSOLx,lERR_OBSx,lERR_CO
Vx,lERR_PARAMx)
! Programmed by M.Veicherts, 2011, last modification by cct 2012-03-16.
! For details about the implementation see REF(J).
! Nstartx   starting column for semi-reduced normal equations
! Nparamx   number of parameters.
! Nerrx     number of error-estimates to be computed.
! NXX       number of observations +1 (?).
      use m_cholsol, only                    : alloc,dealloc,           &
                                               alloc_CHB,dealloc_CHB,   &
                                               reset_CHB,ID_stamp,      &
                                               chunk_write,chunk_read,  &
                                               block_write,get_row_num,&
                                               generate_block_givens,   &
                                               reassemble,reneforsberg,&
                                               get_chunk_subsumA,       &
                                               get_chunk_subsumB,       &
                                               get_chunk_subsumB_err,       &
                                               factorizeA,factorizeB,   &
                                               chol_sol,get_col_num,    &
                                               finalize_cholsol,        &
                                               CH,SOL,single_row_read,  &
                                               formats,chunk,           &
                                               time_stamp,test_sol,     &
                                               ltest,ltest2,ltest3,     &
                                               chunk_file_close,init,   &
                                               alloc_A_chunks,          &
                                               dealloc_A_chunks,        &
                                               read_A_chunks,           &
                                               Nstart,lexist,d0,d1,     &
                                               lMPI,lPARAM,lOBS,lt,lf,  &
                                               lSOL,NC_SOL,Nparam,      &
                                               N,NN,NC,NBB,NB,NC_ERR,   &
                                               Nerr,Nerr_start,         &
                                               lERR_OBS,lERR_COV,       &
                                               lERR_PARAM,              &
                                               factorizeB_err,          &
                                               factorizeA_err,          &
                                               NCX_err,NBX_err,NR,      &
                                               sumup_err,               &
                                               single_col_read,         &
                                               single_col_readx,        &
```



```
                                               single_row_readx,       &
                                               factorizeB_par,         &
                                               factorizeA_par,         &
                                               get_chunk_subsumA_par,  &
                                               get_chunk_subsumB_par,  &
                                               dealloc_sol,            &
                                               diagonal_read
      use m_timing, only                     : timer

      IMPLICIT NONE

! pgf90 -w -O -mp cholsol_mem_7.f90 -o cholsol_mem_7f
! ifort -openmp cholsol_mem_7.f90 -o cholsol_mem_7i

      integer                               :: OMP_GET_THREAD_NUM

      integer                               :: i,ic,jc,Nerr1,  &
                                               ib,jb,Nstartx,Nerrx, &
                                               NCX,NBX,NX,NXX,       &
                                               Nparamx,NC_clean

      integer, dimension(8)                 :: T1,T2,T0
      real*8                                :: err
      real*8, dimension(:,:),   allocatable :: C
!     real*8, dimension(:,:),   allocatable :: C,A,BL
      real*8, dimension(:),     allocatable :: RF
      character*8, dimension(9)             :: ID
      logical                               :: lrf,ltest_sol,lOBSx,&
      lSOLx,lERR_OBSx,lERR_COVx,lERR_PARAMx
!     character*16                          :: tag
      type(chunk)                           :: CHB

      call timer('Cholsol',1)
      if (ltest) write(*,*) ' SUBROUTINE cholsol --- '
      if (ltest) write(*,*)' NBB,NN,NXX:  ', NBB,NN,NXX

!     NC   = NXX/(NBB*NN) + MERGE(0,1,mod(NXX,NBB*NN).EQ.0)
! change 2013-01-21.
      NC   = (NXX+1)/(NBB*NN) + MERGE(0,1,MOD(NXX+1,NBB*NN).EQ.0)
      N    = NXX
      lOBS = lOBSx
      lERR_OBS = lERR_OBSx
      lSOL = lSOLx                      ! include parameter estimation
      lERR_COV=lERR_COVx
      lERR_PARAM=lERR_PARAMx
      ltest = lt
      Nerr = Nerrx
      if (Nparamx.GT.0) then
        lPARAM   = lt
        NC_clean = (NXX-Nparamx)/(NBB*NN) !  + MERGE(0,1,mod((NXX-Nparam),NBB*NN
).EQ.0)
        Nparam   = NXX - Nparamx
      else
        lPARAM   = lf
        NC_clean = NC
      end if
      if (ltest2) write(*,*)' Nparam ',Nparam
      Nstart = Nstartx
      if (ltest2) write(*,*)' NXX,Nstart,Nparam,Nerr,lOBS,lSOL,lERR_OBS,lERR_COV
,lERR_PARAM,NC_clean',NXX,Nstart,Nparam,Nerr,lOBS,lSOL,lERR_OBS,lERR_COV,lERR_PA
RAM,NC_clean
      if (lt) write(*,*)' NXX,Nstart,Nparam,Nerr,lOBS,lSOL,lERR_OBS,lERR_COV,lER
R_PARAM,NC_clean',NXX,Nstart,Nparam,Nerr,lOBS,lSOL,lERR_OBS,lERR_COV,lERR_PARAM,
NC_clean

      ltest_sol = lf
      lexist    = Nstart.gt.1

      if (.NOT.lexist) Nstart = 1   ! column number of start fact.
```

```
      lSOL   = lOBS
!      Nc_sol = 1

! N is the rank of the design matrix A
! NN is the rank of the block submatrix
! NB number of blocks in a row of matrix A
! NBB is number of blocks in a row of a chunk
! NC rank of chunks

! adjusting the chunk and block configuration:
!      N  = N + Nparam
!      N  = N + 1                   ! to include the result column  ! maybe onl
y when testing!

!      call init

      Ncx = 1
      if (lexist) then                          ! start from already saved semi-r
educed solution
        call get_col_num(Nstart,NCX,NBX,NX)    ! Nchunk, Nblock and Nsingle data
        if (NBX.EQ.NB.AND.NX.EQ.NN) NCX = NCX + 1
!       if (lt) write(*,*) ' L1 INFO - Nstart,NCX,NBX,NX ',Nstart,NCX,NBX,NX
      end if

      if (ltest2) write(*,*) ' L1 INFO - NCX,NC,1,NC ',NCX,NC,1,NC

! rene forsberg check if N less than 31:2
!     if (N.LE.1) then
        lrf = lf
        ltest_sol=lf
!     else
!       lrf = lf
!     end if

!     if (lrf) allocate(RF(N-1))

!---- Allocation with array of chunks ----

! Here it is a full allocation. This will be changed to
! allocation (and deallocation) on demand and for individual machines:
! Chunks should e.g. be allocated rowwise.
! NC value set correct?

      call DATE_AND_TIME(VALUES = T1)

      if (ltest_sol) then
        allocate(CH(Nc,Nc))
        allocate(C(NN,NN))
        write(*,*) ' INFO - generating block Givens matrix '
        do ic = NCX,NC
          do jc = ic,NC
            if (ltest) write(*,*) ' INFO L3 - Givens chunk progress',NC,ic,jc
            call alloc(ic,jc)   ! creates CH

! -------------------------------------------------------
! for back solution testing a vector of ones are created:
! The following procedure for creating an SPD matrix
! can be used when setting up the observation equations:
! -------------------------------------------------------

            do ib = (ic-1)*NBB+1,ic*NBB
!              k = (ib+1)*ib/2                          ! block number - not use
d anymore!
              do  jb = (jc-1)*NBB+1,jc*NBB
                call generate_block_givens(NN,C,ib,jb)

! inserting a soluble test solution of ones and twies!:

                if (ic.EQ.NC.OR.jc.EQ.NC) then
```



```
                  if (ib.EQ.NB) C(N-(NB-1)*NN,:) = 1.d0
                  if (jb.EQ.NB) C(:,N-(NB-1)*NN) = 1.d0
                  if (ib.EQ.NB.AND.jb.EQ.NB) C(N-(NB-1)*NN,N-(NB-1)*NN) = N*2.d0
!                  if (ib.EQ.NB.AND.jb.EQ.NB) C(N-(NB-1)*NN,N-(NB-1)*NN) = 16.d0
                end if
                CH(ic,jc)%BL(mod(ib-1,NBB)+1,mod(jb-1,NBB)+1)%A = C
              end do
            end do
!            if (lexist) then  ! read saved solution
! If data are to be read from file: NOT finished - should probably be moved to m
ain (OMP) loop!
!              if (.NOT.lexist.AND..NOT.ltest) then   ! read fresh observations
!              end if
            call chunk_write(ic,jc)
            call dealloc(ic,jc)
          end do
        end do
        deallocate(C)
        deallocate(CH)
      else
        write(*,*) ' INFO - read or create neq! ',NCX,NC
      end if

!     if (ltest) write(*,*) ' INFO L3 - after givens! '

! ---------------------------------------------

      call time_stamp(T1,T2,ic)

! Reassembling the start matrix if N < 30, - for tests):

!      if (N.LE.29) call reassemble

! Note that rank of matrix is N-1 because the N'th col is equation right side

      if (lrf) call reneforsberg(N-1,formats(1),formats(2),RF)

! ------ FACTORISATION MAIN LOOP BEGINS -------

!     if (ltest) write(*,*) ' INFO - Start factorising '
!      tt1 = cputime()
      call DATE_AND_TIME(VALUES = T1)
      T0 = T1

      ID(1) = ' read A '
      ID(2) = ' read B '
      ID(3) = ' write A'
      ID(4) = ' write B'
      ID(5) = ' fact A '
      ID(6) = ' fact B '
      ID(7) = ' subsumA'
      ID(8) = ' subsumB'
      ID(9) = ' fact AB'

! -----------------------------------------------------
! ------------ MAIN PARALLEL OMP LOOP: ----------------
! -----------------------------------------------------

      if (lOBS) then
        call timer('Obs',1,'Cholsol')
        allocate(CH(Nc,Nc))

        do ic = 1,NC_clean
          call alloc_A_chunks(ic)   ! allocating A_CH for get_chunk_subsum
          call alloc_CHB(CHB)       ! allocating CHB for subsum result
          if (ic.GT.1) then
            call read_A_chunks(ic)  ! loading all chunks above ic - used in get_
chunk_subsum
            if (ic.GE.NCX) then
```

```
!            write(*,*)' get_chunk_subsumA called,ic,NCX ',ic,NCX
             call get_chunk_subsumA(ic,CHB) ! this routine is separated in A and
 B part to enable local omp
            end if
          end if
#ifdef _OPENMP
          if (ltest2) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(7))
#endif
!bso this should be MPI
          call alloc(ic,ic)        ! allocating CH(ic,ic)
          call chunk_read(ic,ic)   ! fill chunk with data from either HD or rout
ine
!             if (ltest2) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(1))

          if (ic.GE.NCX) then       ! if solution is restarting the first A_chun
ks should not be reprocessed!
!           write(*,*)' processing chunk row,ic,NCX ',ic,NCX
            do ib = 1,NBB
!             if (ltest3) call block_write(NINT(SQRT(1.d0*size(CHB%BL(ib,ib)%A))
),CH(ic,ic)%BL(ib,ib)%A)
              call factorizeA(ib,ic,CHB%BL(ib,ib)%A)  ! if (ic.GT.1) CHB is used
 here.

!$OMP PARALLEL DEFAULT (none) SHARED(ib,ic,NBB,CHB)
!$OMP DO PRIVATE (jb) SCHEDULE (static)

              do jb = ib+1,NBB
                call factorizeB(ib,jb,ic,ic,CHB%BL(ib,jb)%A)
              end do

!$OMP END DO
!$OMP END PARALLEL

            end do
          end if
          call dealloc_CHB(CHB)                ! deallocating CHB

!$OMP PARALLEL SHARED (ic,NBB,NC,NC_clean,ltest3) PRIVATE(T1,T2)
!$OMP SINGLE

          if (ic.GE.NCX) call chunk_write(ic,ic)    ! Full chunk A is finished a
nd now written by single thread
! change GT to GE 2012-05-16.

!$OMP END SINGLE NOWAIT

!$OMP DO PRIVATE (jc,ib,jb,CHB) SCHEDULE (static)

          do jc = MAX(ic+1,NCX),NC                          ! for the remaining
chunks!
!           write(*,*)' now reducing chunk (ic,jc) ',ic,jc

            call alloc_CHB(CHB)                             ! allocating CHB
            if (ic.GT.1) THEN
              call get_chunk_subsumB(ic,jc,CHB)  ! creates CHB
!             write(*,*)' chunk_subsumB called, ic,jc ',ic,jc
            end if

            call alloc(ic,jc)
            call chunk_read(ic,jc)

#ifdef _OPENMP
            if (ltest3) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(2))
#endif
            do ib = 1,NBB
              do jb = 1,NBB
!                if (ltest) CALL id_STAMP(t1,t2,omp_get_THREAD_NUM(),ID(6))
                call factorizeB(ib,jb,ic,jc,CHB%BL(ib,jb)%A)
              end do
```



```
            end do
#ifdef _OPENMP
            if (ltest3) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(6))
#endif
            call chunk_write(ic,jc)   ! chunk B (ic,jc) is finished and now writ
ten + deallocated:
            call dealloc(ic,jc)
            call dealloc_CHB(CHB)   ! all finished - buffer chunk is closed
#ifdef _OPENMP
            if (ltest3) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(4))
#endif
          end do

!$OMP END DO
!$OMP END PARALLEL

          call dealloc(ic,ic)  ! all chunks in row finished and chunk A closed
          call dealloc_A_chunks(ic)  ! deallocating A_CH

          if (ltest) then
            write(*,*) ' L1 - INFO: CHUNK ROW: ',ic
            call time_stamp(T1,T2,ic)
            T1 = T2
          end if
        end do

! ////////////////////  PARAMETER SECTION ////////////////////////
! // copy of routine above but with negative accumulation ////

        if (lPARAM) then  ! call routines with (test for) negative accumulation:
          write(*,*) ' ...continue factorising watching out for parameters! ',NC
_clean,NC

          do ic = NC_clean+1,NC
            call alloc_A_chunks(ic)   ! allocating A_CH for get_chunk_subsum
            call alloc_CHB(CHB)       ! allocating CHB for subsum result
            if (ic.GT.1) then
              call read_A_chunks(ic)  ! loading all chunks above ic - used in ge
t_chunk_subsum
              if (ic.GT.NC_clean) call get_chunk_subsumA_par(ic,CHB) ! this rout
ine is separated in A and B part to enable local omp
            end if
#ifdef _OPENMP
            if (ltest2) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(7))
#endif
            call alloc(ic,ic)        ! allocating CH(ic,ic)
            call chunk_read(ic,ic)   ! fill chunk with data from either HD or ro
utine
!               if (ltest2) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(1))
            if (ic.GT.NC_clean) then    ! if solution is restarting the first A_
chunks should not be reprocessed!
              do ib = 1,NBB
!               if (ltest3) call block_write(NINT(SQRT(1.d0*size(CHB%BL(ib,ib)%A
))),CH(ic,ic)%BL(ib,ib)%A)
                call factorizeA_par(ib,ic,CHB%BL(ib,ib)%A)  ! if (ic.GT.1) CHB i
s used here.

!$OMP PARALLEL DEFAULT (none) SHARED(ib,ic,NBB,CHB)
!$OMP DO PRIVATE (jb) SCHEDULE (static)

                do jb = ib+1,NBB
                  call factorizeB_par(ib,jb,ic,ic,CHB%BL(ib,jb)%A)
                end do

!$OMP END DO
!$OMP END PARALLEL

              end do
            end if
```

```
           call dealloc_CHB(CHB)               ! deallocating CHB

!$OMP PARALLEL SHARED (ic,NBB,NC,NC_clean,ltest3) PRIVATE(T1,T2)
!$OMP SINGLE

           if (ic.GT.NC_clean) call chunk_write(ic,ic)    ! Full chunk A is fin
ished and now written by single thread
! change GT to GE 2012-05-16.

!$OMP END SINGLE NOWAIT

!$OMP DO PRIVATE (jc,ib,jb,CHB) SCHEDULE (static)

           do jc = MAX(ic+1,NCX),NC                          ! for the remainin
g chunks!

             call alloc_CHB(CHB)                             ! allocating CHB
             write(*,*)' IN PARAM_SECT ic,jc,NCX : ',ic,jc,NCX
             if (ic.GT.1) call get_chunk_subsumB_par(ic,jc,CHB)  ! creates CHB

             call alloc(ic,jc)
             call chunk_read(ic,jc)
#ifdef _OPENMP
             if (ltest3) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(2))
#endif
             do ib = 1,NBB
               do jb = 1,NBB
!               if (ltest) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(6))
                call factorizeB_par(ib,jb,ic,jc,CHB%BL(ib,jb)%A)
               end do
             end do
#ifdef _OPENMP
             if (ltest3) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(6))
#endif
!            do ib = 1,NBB
!              do jb = 1,NBB
!               if (ltest) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(6))
!                call factorizeB_par(ib,jb,ic,jc,CHB%BL(ib,jb)%A)
!              end do
!            end do
#ifdef _OPENMP
             if (ltest3) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(6))
#endif
             call chunk_write(ic,jc)   ! chunk B (ic,jc) is finished and now wr
itten + deallocated:
             call dealloc(ic,jc)
             call dealloc_CHB(CHB)   ! all finished - buffer chunk is closed
#ifdef _OPENMP
             if (ltest3) call ID_stamp(T1,T2,OMP_GET_THREAD_NUM(),ID(4))
#endif
           end do

!$OMP END DO
!$OMP END PARALLEL

           call dealloc(ic,ic)        ! all chunks in row finished and chunk A
closed
           call dealloc_A_chunks(ic)  ! deallocating A_CH

           if (ltest) then
             write(*,*) ' L1 - INFO: CHUNK PARAM ROW: ',ic
             call time_stamp(T1,T2,ic)
             T1 = T2
           end if
         end do

       end if ! END OF PARAMETER ESTIMATION !

! -------------------------------------------------------
```



```
! ----------- FACTORISATION MAIN LOOP ENDS --------------
! -------------------------------------------------------

        call time_stamp(T0,T2,ic)

        if (ltest) write(*,*) ' INFO - Cholesky factorisation finished '
!      if (N.LE.29)  call reassemble

        if (lsol) then
          call chol_sol(err)
!         if (.NOT.lrf.AND.N.LT.16.AND.LF) then
!           write(*,*) ' INFO - solution up to N = 16 :'
!           k = 0
!           do i = 1,NC
!             do j = 1,NBB
!               k = k + 1
!               write(*,formats(2)) SOL(i)%VEC(j)%b
!             end do
!           end do
!         end if
! for testing solution against reneforsberg routine:
!       if (lrf) call test_sol(RF)
!          call dealloc_sol(NC)

        end if

        call time_stamp(T1,T2,ic)
!     if (lrf) deallocate(RF)

! end of ordinary Cholesky solution.

        deallocate(CH)
        call timer('Obs',2,'Cholsol')
      end if !  END OF MAIN LOOP
      IF (lOBS) return    ! and not lERR ??

! change 2012-03-12 and back 2012-03-17.
! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      if (ltest) write(*,*) ' and error_estimates... '

      Nerr_start = N+NBB-MOD(N,NBB)+1

! ------ find required number of Error-chunks NC_ERR:  and allocate:
!     Nerr    = 0
!     Nparam  = 0    ! temporary .
!     if (lERR_PARAM) Nerr  = Nerr + Nparam
! change 2012-07-31.
      if (lERR_OBS.or.lERR_PARAM)   Nerr1 = Nerr + NN*NBB-mod(N,NN*NBB)+N
      NC_err = Nerr1/NN/NBB
      if (mod(Nerr1,NN*NBB).NE.0) NC_err=NC_err+1
! changed 2012-03-16.
!     if ((d1*Nerr/NN/NBB-d1*NC_err*NN*NBB).GT.1.d-3) NC_err = NC_err+1
      if (ltest2) write(*,*)' L1 - INFO  Nerr,Nerr1,NC_err, Nc ',Nerr,Nerr1,NC_e
rr,Nc

!      write(*,*)' single row original '
!      call single_row_read(NXX+1,NC_err)

!      deallocate(CH)
      allocate(CH(Nc,NC_err))      ! making rectangular chunk matrix

! create all columns for error estimates:  (not finished)
!      allocate(ERR_EST(Nerr))
!      ERR_EST  = d0
!      allocate(ERR_COL_VEC(N))                ! kan laves meget smartere, men i
kke nu!

! making error observation chunks:  (this is only for parameters!)
```

```
!      Nerr_x = N - 1
!      ERR_COL_VEC         = d0
!      ERR_COL_VEC(Nerr_x) = 1.d0

! ------------ making error and error_cov entries in main matrix
      if (ltest_sol) then
        call timer('test_sol',1,'Cholsol')
        allocate(C(NN,NN))
        if (lERR_OBS.OR.lERR_PARAM) then
          if (ltest) write(*,*)' INFO - generating error obs.'
          do jc = NC+1,NC+NC_err
            do ic = 1,NC
              call alloc(ic,jc)   ! creates CH
              do ib = (ic-1)*NBB+1,ic*NBB
                do jb = (jc-1)*NBB+1,jc*NBB
                  C = d0   ! should be replaced by real covariances!
!               call test_zero(C)
                  CH(ic,jc)%BL(mod(ib-1,NBB)+1,mod(jb-1,NBB)+1)%A = C
                end do
              end do
              call chunk_write(ic,jc)
              call dealloc(ic,jc)
            end do
          end do
        end if
        if (lERR_COV) then
          if (ltest) write(*,*)' INFO - generating error cov.'
          do jc = NC+1,NC+NC_err
            do ic = NC+1,jc
              call alloc(ic,jc)   ! creates CH
              do ib = (ic-1)*NBB+1,ic*NBB
                do jb = (jc-1)*NBB+1,jc*NBB
                  C = d0   ! should be replaced by real covariances!
!                 call test_zero(C)
                  CH(ic,jc)%BL(mod(ib-1,NBB)+1,mod(jb-1,NBB)+1)%A = C
                end do
              end do
              call chunk_write(ic,jc)
              call dealloc(ic,jc)
            end do
          end do
        end if
        deallocate(C)
        call timer('test_sol',2,'Cholsol')
      end if

      call DATE_AND_TIME(VALUES = T1)
      T0 = T1

! --------  start calculating errors:

      if (lERR_OBS.OR.lERR_PARAM) then
        call timer('Err_obs/param',1,'Cholsol')
        call get_col_num(NXX,NCX_err,NBX_err,NR)
        write(*,*)' INFO - estimating error obs. rank:',NXX,NCX_err,NBX_err,NR
        do ic = 1,NC-1         ! NCX may differ from 1 if solution is restarting
          call alloc_A_chunks(ic)    ! allocating A_CH for get_chunk_subsum
          call alloc_CHB(CHB)        ! allocating CHB for subsum result
          if (ic.GT.1) then
            call read_A_chunks(ic)   ! loading all chunks above ic for subsum
            call get_chunk_subsumA(ic,CHB)
          end if
          call alloc(ic,ic)          ! allocating CH(ic,ic)
          call chunk_read(ic,ic)     ! fill chunk with data from HD or routine

! --- NOTE - chunk A has already been  factorized!
 if (lf) write(*,*)'NC+1,NC_err, first part ',NC+1,NC_err
!$OMP PARALLEL
!$OMP DO PRIVATE (jc,ib,jb,CHB)  SCHEDULE (static)
```



```
          do jc = NC+1,NC_err        ! for the remaining chunks!
! changed 2012-03-16.
!         do jc = NC+1,NC+NC_err        ! for the remaining chunks!

            call alloc_CHB(CHB)                              ! allocating CHB
            if (ic.GT.1) call get_chunk_subsumB(ic,jc,CHB)  ! creates CHB

            call alloc(ic,jc)
            call chunk_read(ic,jc)

            do ib = 1,NBB
              do jb = 1,NBB
                call factorizeB(ib,jb,ic,jc,CHB%BL(ib,jb)%A)
              end do
            end do

            call chunk_write(ic,jc)

            call dealloc(ic,jc)
            call dealloc_CHB(CHB)
          end do

!$OMP END DO
!$OMP END PARALLEL

          call dealloc(ic,ic)   ! all chunks in row finished. Chunk A closed
          call dealloc_A_chunks(ic)                          ! deallocating A_CH

          if (ltest.and.(.false.)) then
            write(*,*) ' L1 - INFO: CHUNK ROW err: ',ic
            call time_stamp(T1,T2,ic)
            T1 = T2
          end if
        end do

! and for the last row of chunks with caution! : (Actually only difference is th
at
! the last chunk is only factorised up to the number of equations! )

!       write(*,*)' INFO - before last row of chunks '
        ic = NC
        call alloc_A_chunks(ic)    ! allocating A_CH for get_chunk_subsum
        call alloc_CHB(CHB)        ! allocating CHB for subsum result
        if (ic.GT.1) then
          call read_A_chunks(ic)   ! loading all chunks above ic for subsum
          call get_chunk_subsumA(ic,CHB)
        end if
        call alloc(ic,ic)          ! allocating CH(ic,ic)
        call chunk_read(ic,ic)     ! fill chunk with data from HD or routine

! --- NOTE - chunk A has already been factorized!

!       write(*,*)' NC+1,NC_err ',NC+1,NC_err
 if (lf) write(*,*)'NC+1,NC_err, second part ',NC+1,NC_err
!$OMP PARALLEL
!$OMP DO PRIVATE (jc,ib,jb,CHB)  SCHEDULE (static)
        do jc = NC+1,NC_err        ! for the remaining chunks!
! changed 2012-03-16.

          call alloc_CHB(CHB)                              ! allocating CHB
          if (ic.GT.1) then
           call get_chunk_subsumB_err(ic,jc,CHB)  ! creates CHB
!          write(*,*)'chunk_subsumB_err called, ic,jc ',ic,jc
!         if (ic.GT.1) call get_chunk_subsumB(ic,jc,CHB)  ! creates CHB
          end if

          call alloc(ic,jc)
          call chunk_read(ic,jc)
```

```
          do ib = 1,NBB
            do jb = 1,NBB
              call factorizeB_err(ib,jb,ic,jc,CHB%BL(ib,jb)%A)
            end do
          end do

          call chunk_write(ic,jc)
          call dealloc(ic,jc)
          call dealloc_CHB(CHB)
        end do

!$OMP END DO
!$OMP END PARALLEL

        call dealloc(ic,ic)   ! all chunks in row finished. Chunk A closed
        call dealloc_A_chunks(ic)              ! deallocating A_CH

!$OMP PARALLEL
!$OMP DO  PRIVATE (jc)
        do jc = NC+1,NC_err
          call sumup_err(NXX+1,jc)
        end do
!$OMP END DO
!$OMP END PARALLEL

        if (ltest.and.lf) then
          write(*,*) ' L1 - INFO: CHUNK ROW err: ',ic
          call time_stamp(T1,T2,ic)
          T1 = T2
        end if
        call timer('Err_obs/param',2,'Cholsol')
      end if

!     if (lERR_OBS) then
!     call single_row_read(Nerr+1,NC_err)
!     write(*,*)' single_row_read: ',Nerr+1,NC_err
!     else
      call single_row_read(NXX+1,NC_err)
      write(*,*)' single_row_read: ',NXX+1,NC_err
!     end if

! ------- AND FOR ERROR COVARIANCES: -------

      if (lERR_COV) then
      call timer('Err_cov',1,'Cholsol')

        if (ltest) write(*,*)' INFO - estimating error cov.'
        do ic = NC+1,NC+NC_err ! NCX may differ from 1 if solution is restarting
!          lfirst_write = lt
          call alloc_A_chunks(ic)    ! allocating A_CH for get_chunk_subsum
          call alloc_CHB(CHB)        ! allocating CHB for subsum result

          if (ic.GT.1) then
            call read_A_chunks(ic)   ! loading all chunks above ic for subsum
            call get_chunk_subsumA(ic,CHB)
          end if

          call alloc(ic,ic)          ! allocating CH(ic,ic)
          call chunk_read(ic,ic)     ! fill chunk with data from HD or routine

          do ib = 1,NBB
            call factorizeA_err(ib,ic,CHB%BL(ib,ib)%A) ! CHB is used here.
            jc = ic

! $OMP PARALLEL SHARED(ib,ic,jc,NBB,CHB)
! $OMP DO PRIVATE (jb)  SCHEDULE (static)

            do jb = ib+1,NBB
              call factorizeB_err(ib,jb,ic,jc,CHB%BL(ib,jb)%A)
```



```
            end do


! $OMP END DO
! $OMP END PARALLEL

          end do

          call dealloc_CHB(CHB)      ! deallocating CHB

!$OMP PARALLEL SHARED(ic,NC,NC_err,NBB)
!$OMP SINGLE

          call chunk_write(ic,ic)   ! Full chunk A is finished and now written b
y single thread

!$OMP END SINGLE NOWAIT

!$OMP DO PRIVATE (jc,ib,jb,CHB)  SCHEDULE (static)
          do jc = ic+1,NC+NC_err            ! for the remaining chunks!

            call alloc_CHB(CHB)                              ! allocating CHB
            if (ic.GT.1) call get_chunk_subsumB(ic,jc,CHB)  ! creates CHB

            call alloc(ic,jc)
            call chunk_read(ic,jc)
            do ib = 1,NBB
              do jb = 1,NBB
                call factorizeB_err(ib,jb,ic,jc,CHB%BL(ib,jb)%A)
              end do
            end do

            call chunk_write(ic,jc)
            call dealloc(ic,jc)
            call dealloc_CHB(CHB)
          end do

!$OMP END DO
!$OMP END PARALLEL

          call dealloc(ic,ic)   ! all chunks in row finished. Chunk A closed
          call dealloc_A_chunks(ic)                          ! deallocating A_CH

          if (ltest) then
            write(*,*) ' L1 - INFO: CHUNK ROW: ',ic
            call time_stamp(T1,T2,ic)
            T1 = T2
          end if
        end do
      call timer('Err_cov',2,'Cholsol')
      end if

      if (allocated(CH)) deallocate(CH)

!     write(*,*) ' INFO - error estimates for columns ',Nerr_start,' to' ,Nerr_s
tart+Nerr-1
!     write(*,*) '  column no.        error estimate '
!      call single_row_read(NXX+1,NC_err)
      if (lt) then
        do i = 1,Nerr
!          write(*,30) Nerr_start + i - 1 ,err_est(i)
!          write(*,30) Nerr_start + i - 1 ,err_est(i)

        end do
      end if
30    format(I7,d12.3)
      call finalize_cholsol

      call timer('Cholsol',2)
```

```
      end subroutine cholsol

! %%%%%%%%%%%%%%%%% END OF PROGRAM %%%%%%%%%%%%%%%%%%%%%%%
      SUBROUTINE FILENAMEGENERATOR(IUNIT)
! THE SUBROUTINE WILL GENERATE NAMES TO BE USED FOR THE FILES USED
! TO HOLD DATA.
! WRITTEN JULY 2007 BY M.VEICHERTS. CHANGE 2012-09-20 BY CCT.
! FMIOLD  = LAST USED FILE-NAME
! DNANE   = ARRAY HOLDING NAMES
! FMINEW  = NEXT FILE NUMBER.

      USE  m_geocol_data, ONLY :  DNANE

      IMPLICIT NONE

      INTEGER IH,i,j,IUNIT
      CHARACTER(10) t
      CHARACTER(8) d

!
      CALL DATE_AND_TIME(TIME=t,DATE=d)

       I =  IUNIT
       j = MOD(I,10)
       IH = INT(I/10)
       DNANE(1,I) = &
!     'GEOCOL19_'//d//'_'//t(1:6)//'_'//CHAR(IH+48)//CHAR(j+48)//'.BIN'
       trim(DNANE(1,1))//d//'_'//t(1:6)//'_'//CHAR(IH+48)//CHAR(j+48)//'.BIN'
       WRITE(*,*) ' FILENAME  ',DNANE(1,I),' AND NO. ',IUNIT

      RETURN
      END SUBROUTINE FILENAMEGENERATOR
      subroutine Qua_to_Mat(qua, matr)
!
! Compute rotation matrix from quaternions
!
  implicit none
  !
  ! Formal parameters
  !
  REAL*8 :: qua(4)     ! Input: quaternions
  REAL*8 :: matr(3,3)  ! Output: rotation matrix
  !
  ! Local declarations
  !
  REAL*8 :: q(4) ! Normalized quaternions
  !
  q = qua/SQRT(DOT_PRODUCT(qua,qua))
  !
  matr(1,1) = -q(2)*q(2) - q(3)*q(3)
  matr(1,2) =  q(1)*q(2) + q(3)*q(4)
  matr(1,3) =  q(1)*q(3) - q(2)*q(4)
  matr(2,1) =  q(1)*q(2) - q(3)*q(4)
  matr(2,2) = -q(1)*q(1) - q(3)*q(3)
  matr(2,3) =  q(2)*q(3) + q(1)*q(4)
  matr(3,1) =  q(1)*q(3) + q(2)*q(4)
  matr(3,2) =  q(2)*q(3) - q(1)*q(4)
  matr(3,3) = -q(1)*q(1) - q(2)*q(2)
  !
  matr = matr + matr
  matr(1,1) = matr(1,1) + 1.0d0
  matr(2,2) = matr(2,2) + 1.0d0
  matr(3,3) = matr(3,3) + 1.0d0
  !
end subroutine Qua_to_Mat
! ======================================================================
module m_input
! ======================================================================
```



```
    implicit none
! ----------------------------------------------------------------------
    type command_argument
      character(len=128)   :: val  ! value of argument
      integer              :: len  ! length of argument
      integer              :: stat ! status of argument
    end type command_argument
! ======================================================================
  contains
! ======================================================================
  subroutine open_job(MPI_pid)
! ======================================================================
    implicit none
! ----------------------------------------------------------------------
    integer, intent(in)    :: MPI_pid
    type(command_argument) :: com_arg
    integer                :: ios  ! I/O status
! ======================================================================
! Retrive job from command line argument (first argument)
! -------------------------------------------------------

    call get_command_argument(1,com_arg%val,com_arg%len,com_arg%stat)

! ----------------------------------------------------------------------
    select case ( com_arg%stat )
! ----------------------------------------------------------------------
      case ( 0 )

! -------------
! Open job-file
! -------------

        open(unit=5, file=com_arg%val(1:com_arg%len), iostat=ios, &
             status='old', action='read')

! -----------------
! Check for success
! -----------------

        if ( ios /= 0 ) then
          write(*,'(a)') 'Error opening job-file:'
          write(*,'(a)') 'job-file: '//com_arg%val(1:com_arg%len)
          write(*,'(a,i3)') 'iostat: ',ios
          stop 'Stopping'
        end if

        if ( MPI_pid == 0 ) then
          write(*,'(a)') '=============================================='
          write(*,'(a)') 'job-file: '//com_arg%val(1:com_arg%len)
          write(*,'(a)') '=============================================='
        end if

! ---------------
! Advance 3 lines
!----------------

        read(5,*)
        read(5,*)
        read(5,*)

      case (-1 )

        print '(a)', 'ERROR: filename for job-file too long (>128)'
        stop

      case default

        print '(a)', 'ERROR: Could not retrieve filename for job-file!'
        stop
```

```
! ----------------------------------------------------------------------
   end select
! ----------------------------------------------------------------------

! ======================================================================
  end subroutine open_job
! ======================================================================
end module m_input
! ======================================================================
!DESCRIPTION: This module is a transcript of the BLOCK DATA section of geocol18.

! Change 2013-09-11.
!             Thus, it initializes the variables w their associated data, as did

!             BLOCK DATA.
MODULE M_DATA

  USE m_params, ONLY : NIPT,NIPCAT
! only a few parameters used for array lengths below

IMPLICIT NONE

  INTEGER :: III

!-------------------------------------------------------------------------------
!
  REAL(KIND=8) :: RLAMAX =  0.0D0
  REAL(KIND=8) :: RLOMAX =  0.0D0
  REAL(KIND=8) :: RLAMIN =  0.0D0
  REAL(KIND=8) :: RLOMIN =  0.0D0
  REAL(KIND=8) :: HCZERO = -3.0D8
  REAL(KIND=8) :: RE     =   6371.0D3
! MEAN EARTH RADIUS.
  REAL(KIND=8) :: GMC    =      3.986005D14
! PRODUCT OF GRAVIT. CONSTANT AND EARTH MASS.
  REAL(KIND=8) :: D1     =      1.0D0
  REAL(KIND=8) :: D2     =      2.0D0
  REAL(KIND=8) :: D3     =      3.0D0
  REAL(KIND=8) :: D4     =      4.0D0
  REAL(KIND=8) :: D5     =      5.0D0
  REAL(KIND=8) :: D0     =      0.0D0
  REAL(KIND=8) :: OLDT   =      0.0D0
  REAL(KIND=8) :: OLDR   =      0.0D0
  REAL(KIND=8) :: PREDP  =      0.0D0
  REAL(KIND=8) :: PRETAP =      0.0D0
  REAL(KIND=8) :: RADSEC = 206264.806D0
  REAL(KIND=8) :: PI     =      3.1415926535D0
  REAL(KIND=8) :: HPOLD  =     -1.0D5
  REAL(KIND=8) :: HQOLD  =     -1.0D5
  REAL(KIND=8) :: HCMAX  =      1.0D9
  REAL(KIND=8) :: S      =      0.0D0
  REAL(KIND=8) :: SR     =      0.0D0
  REAL(KIND=8) :: AAI    =      0.0D0
  REAL(KIND=8) :: AAR    =      0.0D0
  REAL(KIND=8) :: CNR    =      0.0D0
  REAL(KIND=8) :: GMP    =      0.0D0
  REAL(KIND=8) :: AX     =      0.0D0

  REAL(KIND=8), DIMENSION(   4) :: OLDB    = 0.0D0
  REAL(KIND=8), DIMENSION(   4) :: BIPC    = 0.0D0
  REAL(KIND=8), DIMENSION(   7) :: BIP     = 0.0D0
  REAL(KIND=8), DIMENSION(  22) :: OBS     = 0.0D0
  REAL(KIND=8), DIMENSION(  36) :: D       = 0.0D0
  REAL(KIND=8), DIMENSION(  36) :: DC      = 0.0D0
  REAL(KIND=8), DIMENSION(  42) :: BSIZE   = 0.0D0
  REAL(KIND=8), DIMENSION(2200) :: SIGMA   = (/(0.0D0,III=1,2200)/)
  REAL(KIND=8), DIMENSION(2200) :: SIGMA0  = (/(0.0D0,III=1,2200)/)
  REAL(KIND=8), DIMENSION(  17) :: C11     = (/1.0D0,1.0D5,1.0D5,1.0D9,1.0D9,-20
```



```
6264.806D0,-206264.806D0,1.0D9,1.0D9,1.0D9,1.0D9,1.0D9,1.0D9,1.0D9,1.0D0,1
.0D0/)
! REAL(KIND=8), DIMENSION(  17) :: C11     = (/1.0D0,1.0D5,1.0D5,1.0D9,1.0D9,-20
6264.806D0,-206264.806D0,1.0D9,1.0D9,1.0D9,1.0D9,1.0D9,2.0D9,1.0D9,1.0D9,1.0D0,1
.0D0/)
  REAL(KIND=8), DIMENSION(  24) :: CCI     = (/0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0
D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,&
                                                0.0D0,0.5D0,0.0D0,0.0D0,0.0D0,0.0
D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0,0.0D0/)

!-------------------------------------------------------------------------------
!
  INTEGER :: KFP    = -1
  INTEGER :: ICSYSL = -2
  INTEGER :: NCZERO = -1
  INTEGER :: IPX    = -1
  INTEGER :: NWAR   =  0
  INTEGER :: ITCOUN =  0
  INTEGER :: NBOLD  =  0
  INTEGER :: ITE    =  0
  INTEGER :: ITE1   =  0
  INTEGER :: INZOLD =  0
  INTEGER :: NPARM  =  0
  INTEGER :: NO     =  0
  INTEGER :: NAI    =  0
  INTEGER :: NLA    =  0
  INTEGER :: IS     =  0
  INTEGER :: ISO    =  0
  INTEGER :: IT     =  0
  INTEGER :: IP     =  0
  INTEGER :: IOBS   =  0
  INTEGER :: IOBSR  =  0
  INTEGER :: N1     =  0
  INTEGER :: NIR    =  0
  INTEGER :: MAXC   =  0
  INTEGER :: NMAX   =  0
  INTEGER :: MAXC1  =  0
  INTEGER :: MAXC2  =  0
  INTEGER :: N      =  0
  INTEGER :: IC     =  0
  INTEGER :: NT     =  0
  INTEGER :: IH     =  0
  INTEGER :: IB1    =  0
  INTEGER :: IP1    =  0
  INTEGER :: IT1    =  0
  INTEGER :: IC1    =  0
  INTEGER :: IC11   =  0
  INTEGER :: K1     =  0
  INTEGER :: IOBS1  =  0
  INTEGER :: IOBS2  =  0
  INTEGER :: IITE   =  0
  INTEGER :: IITE1  =  0
  INTEGER :: IIP    =  0
  INTEGER :: IIP1   =  0
  INTEGER :: IIE    =  0
  INTEGER :: IIE1   =  0
  INTEGER :: INO    =  0
  INTEGER :: IDIMC  =  0
  INTEGER :: IDIMCN =  0
  INTEGER :: MAXBLT =  0
  INTEGER :: IA     =  9
  INTEGER :: IB     =  0
  INTEGER :: IA1    =  9
  INTEGER :: INL    = 10
  INTEGER :: IEM    = 25
  INTEGER :: II     =  2
  INTEGER :: JR     =  2
  INTEGER :: NPARM1 =  1
```

```
  INTEGER, DIMENSION(     2) :: NDSET    = (/0,0/)
! COUNTS NUMBER OF DIFFERENT  INPUT DATASETS.
  INTEGER, DIMENSION(     2) ::   J2     = (/3,2/)
  INTEGER, DIMENSION(     2) ::   I3     = (/6,3/)
  INTEGER, DIMENSION(     2) ::   I4     = (/4,2/)
  INTEGER, DIMENSION(    12) ::   IGP    = (/0,0,0,0,0,0,0,0,0,0,0,0/)
  INTEGER, DIMENSION(    13) ::   IPAR   = (/0,0,0,0,0,0,0,0,0,0,0,0,0/)
  INTEGER, DIMENSION(    17) ::   K7     = (/0,0,0,0,0,1,1,1,1,1,1,2,2,2,2,0,0/)
  INTEGER, DIMENSION(    17) ::   K9     = (/1,1,1,1,1,2,3,2,3,2,3,2,2,3,4,0,0/)
                          ! this was missing...
  INTEGER, DIMENSION(    17) ::   K11    = (/0,0,0,0,0,0,0,0,0,0,0,2,3,3,6,0,0/)
  INTEGER, DIMENSION(    17) ::   K13    = (/1,1,1,1,1,1,1,1,1,1,1,2,3,3,6,0,0/)
  INTEGER, DIMENSION(    17) ::   K15    = (/0,1,-1,-1,1,0,0,-1,-1,2,2,0,0,0,0,0
,0/)
  INTEGER, DIMENSION(    17) ::   K17    = (/0,0,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0/)
  INTEGER, DIMENSION(    17) ::   K19    = (/1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0/)
  INTEGER, DIMENSION(    17) ::   K21X   = (/0,1,1,2,2,1,1,2,2,2,2,2,2,2,2,2,2/)
  INTEGER, DIMENSION(    17) ::   K21    = (/0,1,1,2,2,1,1,2,2,2,2,2,2,2,2,2,2/)
  INTEGER, DIMENSION(    17) ::   K23    = (/1,1,1,1,1,2,1,2,1,2,1,1,1,2,2,0,0/)
  INTEGER, DIMENSION(    17) ::   K8     = (/0,1,1,2,2,0,0,1,1,1,1,0,0,0,0,0,0/)

  INTEGER, DIMENSION(    37) ::   KCI    = (/0, 0, 0, 0, 0, 0, 0, 0, 0, 0, &
                                            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, &
                                            0, 0, 0, 0, 0, 1, 0, 1, 0, 2, &
                                            0,-1, 1, 0, 0, 0, 0 /)
  INTEGER, DIMENSION(    37) ::   KVI    = (/0, 0, 0, 0, 0, 0, 0, 0, 0, 0, &
                                            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, &
                                            0, 0, 0, 0, 0, 1, 0, 1, 0, 2, &
                                            0,-1, 1, 0, 0, 0, 0 /)
  INTEGER, DIMENSION(    42) ::   ANDEX  = (/(0,III=1,42)/)
!!JTF: INDEX is not the best name... and it is 42 elements long. really only ini
t first two elements??

 INTEGER, DIMENSION(  17,2) ::   KSAT   = RESHAPE((/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/),SHAPE(KSAT))
!BSOREMOVE  INTEGER, DIMENSION(  17,2) ::   KSAT   =   (/1,3,3,3,3,2,1,2,1,2,1,2
,&
!BSOREMOVE                                                  2,1,1,1,1/)
!BSOREMOVE                                                 1,1,1,1/),(/17,2/))
! KSAT HOLDS THE MAPPING BETWEEN THE DATA CODES AND THE POSITIONS IN THE
!  ARRAY COVCX HOLDING THE COVARIANCES. SEE SUBROUTINE COVCX.

  INTEGER, DIMENSION(NIPCAT) ::   ITRACE = (/(1,III=1,NIPCAT)/)
  INTEGER, DIMENSION(NIPCAT) ::   ITIME  = (/(0,III=1,NIPCAT)/)
  INTEGER, DIMENSION(  NIPT) ::   IPTYPE = (/(0,III=1,NIPT)/)
  INTEGER, DIMENSION(  NIPT) ::   ITIME0 = (/(0,III=1,NIPT)/)

!-----------------------------------------------------------------------------
!
  LOGICAL ::  LT     = .TRUE.
  LOGICAL ::  LNEQ   = .TRUE.
  LOGICAL ::  LSPOUT = .TRUE.
  LOGICAL ::  LCO1   = .TRUE.
  LOGICAL ::  LNERNO = .TRUE.
  LOGICAL ::  LWRSOL = .FALSE.
  LOGICAL ::  LBIPOT = .FALSE.
  LOGICAL ::  LBICOV = .FALSE.
  LOGICAL ::  LBISOL = .FALSE.
  LOGICAL ::  LINSOL = .FALSE.
  LOGICAL ::  LTABH  = .FALSE.
  LOGICAL ::  LDENOL = .FALSE.
  LOGICAL ::  LPOSDA = .FALSE.
  LOGICAL ::  LFIRST = .FALSE.
  LOGICAL ::  LCREF  = .FALSE.
  LOGICAL ::  LC1    = .FALSE.
  LOGICAL ::  LC2    = .FALSE.
  LOGICAL ::  LDEFF  = .FALSE.
```



```
  LOGICAL ::  LMDD   = .FALSE.
  LOGICAL ::  LIN4   = .FALSE.
  LOGICAL ::  LOPCOF = .FALSE.
  LOGICAL ::  LF     = .FALSE.
  LOGICAL ::  LPOT   = .FALSE.
  LOGICAL ::  LPOTIN = .FALSE.
  LOGICAL ::  LGRID  = .FALSE.
  LOGICAL ::  LERNO  = .FALSE.
  LOGICAL ::  LCOMP  = .FALSE.
  LOGICAL ::  LCOM   = .FALSE.
  LOGICAL ::  LWLONG = .FALSE.
  LOGICAL ::  LPRED  = .FALSE.
  LOGICAL ::  LCLU7  = .FALSE.
  LOGICAL ::  LOPEN7 = .FALSE.
  LOGICAL ::  LRESOL = .FALSE.
  LOGICAL ::  LTIME  = .FALSE.
  LOGICAL ::  LTCOV  = .FALSE.
  LOGICAL ::  LONEQ  = .FALSE.
  LOGICAL ::  LTERRC = .FALSE.
  LOGICAL ::  LTABLE = .FALSE.
  LOGICAL ::  LTABLR = .FALSE.
  LOGICAL ::  LNEQ8  = .FALSE.
  LOGICAL ::  LOPEN4 = .FALSE.
  LOGICAL ::  LCOERR = .FALSE.
  LOGICAL ::  LLCOER = .FALSE.
  LOGICAL ::  LY     = .FALSE.
  LOGICAL ::  LKM    = .FALSE.
  LOGICAL ::  LPARAM = .FALSE.
  LOGICAL ::  LNEWSO = .FALSE.
  LOGICAL ::  LCZERO = .FALSE.
! VARIABLE USED TO AVOID MULTIPROCESSING IN PRED. 2012-09-13.
  LOGICAL ::  LTESTS = .FALSE.
  LOGICAL ::  LNEWD  = .FALSE.
  LOGICAL ::  LDIF   = .FALSE.
  LOGICAL, DIMENSION(7) :: L   = (/(.FALSE.,III=1,7)/)
  LOGICAL, DIMENSION(7) :: LN   = (/(.FALSE.,III=1,7)/)
  LOGICAL, DIMENSION(42):: LLCOEE = (/(.FALSE.,III=1,42)/)
  LOGICAL, DIMENSION(42):: LFOURI   = (/(.FALSE.,III=1,42)/)


!-----------------------------------------------------------------------------
!
  CHARACTER(LEN=128), DIMENSION(2) :: OLDCOV
  CHARACTER(LEN=128), DIMENSION(4) :: OLDN

!-----------------------------------------------------------------------------
!
! INTEGER :: 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,   ITIME,INUMR,IP1,K21,K2,IU,IITE1,IITE,IIP1,IIP,IIE,&
!           !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    ,K13,K19,NCZERO,NLA,INO,IB1,ISO,IPX,IS,&
!           !KFQ,JR,NOBLK,IXX,K13,K19,NCZERO,NLA,INO,IB1,ISO,IPX,IS,&
!            JJORD,IIDEG,K1,IPTYPE,K23,I3,IPA,KFP,NPARM,NPARM1,MAXPAR,&
!            II,NMAX,MAXB,   NCXLAS,ICODE,ITRACE,ITMODE,ITM0,ITMOD,ITROLD,&
!           !II,NMAX,MAXB,IX,NCXLAS,ICODE,ITRACE,ITMODE,ITM0,ITMOD,ITROLD,&
!            ITRGAP,ITRACK,ITOLD,NERCOV
! REAL*8 ::  GM,RLOMAX,RLAMAX,RLOMIN,RLAMIN,B,HQ,RLAT,&
!            SINLAT,COSLAT,RLONG,SINLON,COSLON,WOBS,SINLOP,SFACT,FPERIO,&
!            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      ,SCALE,SCALE2      ,OLDT,RADSEC,CFA,SIGMAP,HPOLD,&
!           !VARI,DGPM2,SCALE,SCALE2,DRAPP,OLDT,RADSEC,CFA,SIGMAP,HPOLD,&
```

```
!            D5,D0,D1,D3,D4,PRETAP,CTIME,SIGMAX
!
! LOGICAL :: L,LN,LOPEN7,LONECO,LNKSIP,LNETAP,LDEFVP,LSTOP,LRESOL,&
!            LC1,LC2,LCREF,LKM,LNEQ,LT,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,LY,LNX,LTESTS,LOBSST,&
!            LCOERR,LSPOUT,LTRAN,LLCOER,LCTIME,LSTART,LSATPP,LSATQ
!
! CHARACTER*128 :: OLDN,OLDCOV




!=====================================================================!
 !DATA ITRACE/NIPCAT*1/

 !---------------------------------------------------------------------!
 !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/
  !KSAT HOLDS THE MAPPING BETWEEN THE DATA CODES AND THE POSITIONS IN THE ARRAY
COVCX HOLDING THE COVARIANCES. SEE SUBROUTINE COVCX.

 !---------------------------------------------------------------------!
 !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/

 !---------------------------------------------------------------------!
 ! 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/

 !---------------------------------------------------------------------!
 !DATA RE,GMC,D1,D2,D3,D4,D5,D0,BSIZE,SIGMA,SIGMA0,OLDT,OLDR,PREDP,PRETAP,OBS,RA
DSEC,PI/6371.0D3,3.986005D14,1.0D0,2.0D0,3.0D0,4.0D0,5.0D0,4469*0.0D0,206264.806
D0,3.1415926535D0/,LT,LNEQ,LSPOUT,&
 !LCO1,LNERNO,LWRSOL,LBIPOT,LBICOV,LBISOL,LINSOL,LTABH,LDENOL,LPOSDA,LFIRST,LCRE
F,LC1,LC2,LDEFF,LMDD,LIN4,LOPCOF,LF,LGRID,LERNO,LCOMP,LCOM,LWLONG,LPRED,LCLU7,LO
PEN7,LRESOL,LTIME,LTCOV,LONEQ,&
 !LTERRC,LTABLE,LTABLR,LNEQ8,LOPEN4/5*.TRUE.,34*.FALSE./,RLAMAX,RLOMAX,RLAMIN,RL
OMIN,HCZERO,ICSYSL,NCZERO/4*0.0D0,-3.0D8,-2,-1/,ITCOUN,IPAR,NBOLD,ITE,ITE1,INZOL
D,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/I
XX,IPX,NWAR/0,0,1,0,1,0,1,2,3,0,1,2,3,4,5,6,0,-2,-1,0/

 !---------------------------------------------------------------------!
 !DATA IPTYPE,ITIME0/NIPT*0,NIPT*0/ITIME/NIPCAT*0/LCOERR,LLCOER,LY/3*.FALSE./


END MODULE M_DATA
MODULE m_geocol_data
! last change 2013-09-16.

 USE m_params, ONLY : NISIZE, NNBL, NDIMC, NCRW, NEQFIM, IIMAX, NSAT, SMPAR, NIP
CAT, MAXO,NSPHAR, NIPT, NIPCAT, NCOEF, NROOT

IMPLICIT NONE

 !-------------------------PR-----------------------------!
 REAL(KIND=8) :: SINLOP,COSLOP,BSIZEN,BSIZEE,COSLAP,SINLAP,&
                 RLONGP,RP,CAZP,SAZP,HP,RLATP,OMEGA2

 INTEGER      :: ICZERO,NI,NR,IKP,IKQ,ISATP,NOBLK
```



```
 LOGICAL      :: LONECO,LNKSIP,LNETAP,LDEFVP,LOBSST,LSTART
 !-------------------------DDY----------------------------!
!COMMON /DDY/A,S,RB2,T,B,KT,KT1,K,II,JJ,N3,KK,KQ,KP,ND,NR,ND1,ND2
 REAL(KIND=8) :: A,SX,TT,BZ,RB2
 INTEGER ::      KT,KT1,K,IIZ,JJ,N3,KK,KXQ,KXP,ND,ND1,ND2,NRX

 !-------------------------PR-----------------------------!

 INTEGER ::      ISIZE,MAXBL,MAXCM,MI1,MI2,MMAXB,MAXFIL,IIDEG,JJORD,NEQFMA,&
                 MAXBNE,IORDER,IIOLD,JOLD,IR,FMIOLD,FMINEW,INN,INV,&
                 IDLAT,IDLON,MLAT,MLON,NOX,NFILTE,NFOUR,KP,KPP1,IPC,NAI,&
                 NLA,INL,IEM,MODEC0,ITMODE,ITM0,ITMOD,ITRGAP,ITRACK,ITOLD,&
                 NERCOV,IZ,OLDORD,I1,I2,I3,I4,I5,I6,I7,I8,I9,NMAXSV,JR0,&
                 NO1,K2,K3,K2P3,K4,IU,K21,IANG,IU1,IMAX1,IMAX1R,&
                 NSTEP,NSTEPE,IDSAT,ILA,ILO,IKPREF,INZ,IO2,NPOBS0,NOUSE,&
                 ILAST,IPAMAX,NGR,ICSYS


 LOGICAL ::      LNBL1,LFORM,LFOUR,LSMAL,LADBPR,LADBTE,LNGR,LKSIP,LNCOL,LTNB,LTE
B,&
                 LOE1,LOE2,LE,LCTIME,LSPHAR,LTSPH,LALLCO,HP9000,LFIRST,&
                LPUNCH,LTERMO,LTERM,LOUTC,LTRAN,LNERNO,LK30,LK31,LNUOUT,LK2EQ4,&
                 LSTOP,LWRSOL,LTERMA,LMEAN,LSA,LADMU,LSTAT,LAREA,LZETA,LGRADI,&
                 LMENSI,LSIMH,LMEAN1,LINTRA,LGIGRS,LMEGR,LUSNGS,LSATAC,&
                 LSATP,LGRERR,LCOD,LEQP,LALLP,LGRP,LDEN,LPOTSD,LEQANG,LFILTE,&
                 LBIN,LSKIPL,LGRERS,LTILT,LSCALE,LINERT

 REAL(KIND=8) :: PII,PIM0,PIM1,PIM2,DLP,DLP0,DLP1,DLP2,DAP,DAP0,DAP1, &
                 DAP2,DDAP,DDAP0,DDAP1,DDAP2,DDAL0,DDAL1,VI,SLOP,SLOQ,&
                 CLOP,CLOQ,CLATD,RDI,SLAT,SLON,STEPN,COSSTN,SINSTN,   &
                 STEPE,COSSTE,SINSTE,COST2P,SINT2P,SCFACT,RDD,SHIFTS, &
                 PW2,BSIZEA,E21,AX1,F1,GM1,GREF,UREF,GM,AXS,GMS,SQ2,YS,&
                 YC,V1,VV,DXX,SCALE,SCALE2,CFA,RLATC,rlatcC,AZP,BETP,TAUP,&
                 SINLA0,COSLA0,RLONG0,DX,DY,DZ,EPS1,EPS2,EPS3,DL,S1,AX2,E22,&
                 VARNO,PPA,PPS,VG,HPK,DM,DA,WM,REJLEV,&
                 X,Y,Z,XY,XY2,DISTO,DIST2,DZERO,C20IN,CM3,CMM2,CM1,&
                 HQD,RLATD,SINLATD,COSLATD,RLONGD,SINLOND,COSLOND,&
                 HQE,RLATE,SINLATE,COSLATE,RLONGE,SINLONE,COSLONE


 INTEGER,      DIMENSION(NEQFIM,2)  :: NEQFI
 INTEGER,      DIMENSION(NISIZE)    :: NCAT,ISZE
 INTEGER,      DIMENSION(0:NNBL)    :: NBL
 INTEGER,      DIMENSION(12)        :: INUMR
 INTEGER,      DIMENSION(42)        :: NFOURI
 INTEGER,      DIMENSION(42)        :: ISAT
 INTEGER,      DIMENSION(70)        :: NUM
 INTEGER,      DIMENSION(10)        :: NGRE


!LOGICAL,      DIMENSION(42)    :: LFOURI,LLCOEE
! MOVED TO m_data.f90 2012-11-19.
 LOGICAL,      DIMENSION(2)     :: LP

 REAL(KIND=8), DIMENSION(MAXO)   :: B,HQ,RLAT,SINLAT,COSLAT,RLONG,SINLON,COSLON,
WOBS
 REAL(KIND=8), DIMENSION(NCRW)   :: C
 REAL(KIND=8), DIMENSION(IIMAX)  :: ROOT0                              ! ROOT IS
 A PRECOMPUTED SQUARE ROOT-TABLE (ROOT0(1)=0 !).
 REAL(KIND=8), DIMENSION(NROOT)  :: ROOT                              ! ROOT IS
A PRECOMPUTED SQUARE ROOT-TABLE (ROOT0(1)=0 !).
 REAL(KIND=8), DIMENSION(SMPAR)  :: SIGMAP
 REAL(KIND=8), DIMENSION(NIPCAT) :: CTIME

 REAL(KIND=8), DIMENSION(NSAT)  :: SR11, SR12, SR13, SR22, COSAZ, SINAZ
 REAL(KIND=8), DIMENSION(NSAT)  :: SR11A,SR12A,SR13A,SR22A,COSAZA,SINAZA
 REAL(KIND=8), DIMENSION(11)    :: FILTER
 REAL(KIND=8), DIMENSION(10)    :: SGRE
```

```
 REAL(KIND=8), DIMENSION(3)     :: EE0,G1
 REAL(KIND=8), DIMENSION(7)     :: DSHIF0
 REAL(KIND=8), DIMENSION(32)    :: VARI

 REAL(KIND=8), DIMENSION(42)    :: SCFRDD
 REAL(KIND=8), DIMENSION(2200)  :: SM
 REAL(KIND=8), DIMENSION(55,0:21)  :: FOUCOF
 REAL(KIND=8), DIMENSION(3,3)   :: SATROT1,SATROT,ALLREF,ALLGG,ALLCOL,ALLPRE,&
                                   ALLTRA,ALLPR1,ALLERR,ALLVAR,ALLIN,G2

 REAL(KIND=8), DIMENSION(30)    :: FJ,FG
 REAL(KIND=8), DIMENSION((NSPHAR+1)**2) :: SUMIJ,CCCIJ
 REAL(KIND=8), DIMENSION(0:NSPHAR) :: SN2
 REAL(KIND=8), DIMENSION(3)     :: GS,GC
 REAL(KIND=8), DIMENSION(3,3)   :: DDS,DDC
!REAL(KIND=4), DIMENSION(NCOEF) :: COFF
! CHANGE 2013-02-28.
 REAL(KIND=8), DIMENSION(NCOEF) :: COFF

 REAL(KIND=8), DIMENSION(:,:,:), ALLOCATABLE :: A_BLOCS

 CHARACTER(LEN=128)                      :: ROTFIL,ERNAME
 CHARACTER(LEN=128), DIMENSION(9)        :: FMT
 CHARACTER(LEN=128), DIMENSION(4)        :: OLDN
 CHARACTER(LEN=128), DIMENSION(2)        :: DNAME

 CHARACTER(LEN=128), DIMENSION(2,100) :: DNANE
!COMMON /CMCOV/CCI(24),CCR(56),SIGMA0(2200),SIGMA(2200),SIGMAX(2200,5),
! HCMAX,CCV(2,2),DC(36),KCI(37),NC1,NC2,LOCAL,LSUM
! COMMON VARIABLES USED IN COVAX. SEE THIS SUBROUTINE FOR VARIABLES.
 REAL(KIND=8), DIMENSION(24)     :: CCI
 REAL(KIND=8), DIMENSION(56)     :: CCR
 REAL(KIND=8), DIMENSION(36)     :: DC
 REAL(KIND=8), DIMENSION(2200)   :: SIGMA0,SIGMA
 REAL(KIND=8), DIMENSION(2200,5) :: SIGMAX
 REAL(KIND=8), DIMENSION(2,2)    :: CCV
 REAL(KIND=8), DIMENSION(4)      :: CVC
 LOGICAL                         :: LOCAL,LSUM
 INTEGER                         :: NC1,NC2
 INTEGER,      DIMENSION(37)     :: KCI
 REAL(KIND=8)                    :: HCMAX
!COMMON /CSAT/COVX(3,3,3,3),CIX(7,5),CFA,KSAT(17,2),LSATPP,LSATQ,NDX1(5),
! NDX2(5),NDP,NDQ,NWAR,LY,LX(7,5),LNX(7,5),LTESTS
! KSAT,LY,NWAR,LTESTS ARE INITIALIZED in m_data.f90.
 REAL(KIND=8), DIMENSION(3,3,3,3) :: COVX
 REAL(KIND=8), DIMENSION(7,5)     :: CIX
 REAL(KIND=8)                     :: CFX
 LOGICAL                          :: LSATPP,LSATQ
 LOGICAL,      DIMENSION(7,5)     :: LX,LNX
 INTEGER                          :: NDP,NDQ
 INTEGER,      DIMENSION(5)       :: NDX1,NDX2
!COMMON /CMEAQ/STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE, COST2Q,SINT2Q
! STEPSIZES USED WHEN CALCULATING MEAN VALUES.
 REAL(KIND=8) :: STEQN,COSSQN,SINSQN,STEQE,COSSQE,SINSQE,COST2Q,SINT2Q
!COMMON /CPARM/SFACT(NIPCAT),FPERIO(10,2),IPTYPE(NIPT),IPACAT(3*NIPT),&
!ITIME(NIPCAT),ITIME0(NIPT),ITROLD,ICODE(10),NPARM,NPARM1,MAXPAR,MP,IPA,NCXL
 REAL(KIND=8), DIMENSION(NIPCAT)  :: SFACT
 REAL(KIND=8), DIMENSION(10,2)    :: FPERIO
 INTEGER,      DIMENSION(3*NIPT)  :: IPACAT
 INTEGER,      DIMENSION(10)      :: ICODE
 INTEGER                          :: MAXPAR,MP,IPA,NCXLAS,ITROLD



CONTAINS

!-----------------------------------------------------------------------!
 SUBROUTINE init_data
 !Subroutine for any data initialization needed...
```



```
 END SUBROUTINE init_data

!-----------------------------------------------------------------------!
END MODULE m_geocol_data
MODULE m_params
! VERSION 2013-04-14.

IMPLICIT NONE

  INTEGER, PARAMETER :: IIMAX  = 20000
  INTEGER, PARAMETER :: INBLP  = 150
  INTEGER, PARAMETER :: MAXCX  = 28920
  INTEGER, PARAMETER :: MAXCY  = 200
  INTEGER, PARAMETER :: MAXO   = 16200*11                              ! MAXO IS
USED IN THE COMMON BLOCKS PR AND CPARM AND IN THE DIMENSION STATEMENT
! INTEGER, PARAMETER :: MAXO   = 16200*3                               ! MAXO IS
USED IN THE COMMON BLOCKS PR AND CPARM AND IN THE DIMENSION STATEMENT
! INTEGER, PARAMETER :: MAXO   = 100                                   ! MAXO IS
USED IN THE COMMON BLOCKS PR AND CPARM AND IN THE DIMENSION STATEMENT
! changed 2012-07-21.
  INTEGER, PARAMETER :: MAXO9  = 9*MAXO
  INTEGER, PARAMETER :: MAXOD  = 9*MAXO                                 ! MAXOD M
UST BE EQUAL TO 9*MAXO (= 145800)
  INTEGER, PARAMETER :: MAXSA  = 6*MAXO                                 ! (= 9720
0)
  INTEGER, PARAMETER :: MXPAR  = 2500
  INTEGER, PARAMETER :: NALLCO = 60000
  INTEGER, PARAMETER :: NCTA   = 1600
  INTEGER, PARAMETER :: NCRW   = 100000000
  INTEGER, PARAMETER :: NCX    = 28920
  INTEGER, PARAMETER :: NDIMC  = 99986000
  INTEGER, PARAMETER :: NEQFIM = 60
  INTEGER, PARAMETER :: NEQIV  = 130001
  INTEGER, PARAMETER :: NICC   = 2422201
  INTEGER, PARAMETER :: NIDIMC = 100000000
  INTEGER, PARAMETER :: NIICC  = 2422201
  INTEGER, PARAMETER :: NIPCAT = 100002
! NIPCAT MUST BE EQUAL TO MAXIMAL NUMBER OF DATA USED WHEN LPARAM IS TRUE
  INTEGER, PARAMETER :: NIPT   = 1500
  INTEGER, PARAMETER :: NISIZE = 14000
  INTEGER, PARAMETER :: NMAP   = 400
  INTEGER, PARAMETER :: NNBL   = 20000
  INTEGER, PARAMETER :: NNSU   = 22010
  INTEGER, PARAMETER :: NPMAX  = 28920
  INTEGER, PARAMETER :: NROOT  = 4402
  INTEGER, PARAMETER :: NSAT   = MAXO
! INTEGER, PARAMETER :: NSAT   = 16200
  INTEGER, PARAMETER :: NSPHAR = 360
  INTEGER, PARAMETER :: SMPAR  = 2001

  LOGICAL, PARAMETER :: LT = .TRUE.
  LOGICAL, PARAMETER :: LF = .FALSE.

!NB: similar names same value, error - or redundant definitions?
  INTEGER, PARAMETER :: NCOEF  = 4844402
  INTEGER, PARAMETER :: NCOEFF = 4844402
  INTEGER, PARAMETER :: NCOFF  = 4844402

CONTAINS

!-----------------------------------------------------------------------!
 SUBROUTINE init_params
 !Subroutine for any parameter initialization needed...
 END SUBROUTINE init_params

!-----------------------------------------------------------------------!
END MODULE m_params
! iiort -c -check bounds -traceback -openmp -assume byterecl m_cholsol.f90_p:wq!
! version 2013-02-11 - unused variables isolated, close added in chunk_read..
```

```
! change in SUMUP.

      module m_cholsol
      implicit none

! programmed by M.Veicherts, 2011, last modification 2012-03-16 by cct.
! N      = 500.000 ('typical' value) -- rank of equation system (N x N)
! NN     =     100 ('typical' value) -- rank of block matrix (NN x NN)
! NB     =     500 ('typical' value) -- Number of Blocks on a full row (~ N/NN)
! NBB    =      50 ('typical' value) -- Number of blocks on a chunk
! NC     =      10 ('typical' value) -- Number of Chunks on a full row ( ~ NB/NB
B). A chunk corresponds to a file.
! NF     = ? ('typical' value) -- is the current number of files created
! NPC    = ? ('typical' value) -- is the number of PCs in the cluster
! Nstart = ? ('typical' value) -- is the eq. number where to restart chol factor
isation

      integer                                    :: N,Nn,Nb,Nbb,Nc,Nf, &
                                                    Nstart,Nc_sol,     &
                                                    Nc_err,Nproc,Nerr, &
                                                    Nparam,Nerr_start, &
                                                    NCX_err,NBX_err,NR,&
                                                    NCX_err2

! Nnodes Number of nodes in cluster network (for MPI implementation)

      integer                                    :: Nnodes,Nfil

! Blocks are defined as :

      type block
        real*8 , dimension(:,:), allocatable     :: A
      end type block

! Chunks are defined as:

      TYPE chunk
        type(block) , dimension(:,:),allocatable :: BL
      END TYPE chunk

      type(chunk), dimension(:,:), allocatable   :: CH       ! chunks in A matri
x
      type(chunk), dimension(:), allocatable     :: A_CH     ! column of chunks
above the diagonal element (used for get_subsum)

! A single column (e.g. the solution) is defined as :
      type vect
        real*8 , dimension(:), allocatable       :: b
      end type vect

      type sol_chunk
        type(vect), dimension(:), allocatable    :: VEC
      end type sol_chunk

      type(sol_chunk), dimension(:), allocatable :: SOL

      logical, dimension(:,:,:,:), allocatable   :: ZERO_BL      ! flag array to
 show if blocks contain only zeros

      integer                                    :: ifmax,jfmax
      integer     , dimension(:,:), allocatable  :: filno
      character*72, dimension(:,:), allocatable  :: filename

      character*128                              :: file_root_name,    &
                                                    file_root_name_old
! formats for test output:
      character*16, dimension(4)                 :: formats
      real*8                                     :: d0,d1
      parameter  (d0 = 0.0d0,d1 = 1.0d0)
```



```
      logical                                    :: ltest,ltest2,lt,lf,&
                                                    lMPI,lparam,lSOL,  &
                                                    lERR_COV,lzero_bl, &
                                                    lerr_param,lobs,   &
                                                    lfirst_chol,ltest3,&
                                                    lerr_obs,lexist

      parameter (lt =.true.,lf =.false., Nfil = 100)

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      contains
! ------------------------------------------------------

      subroutine chunk_reset(ci,cj)
      implicit none

      integer      :: bi,bj,ci,cj,istart

      if (ltest2) write(*,*) ' INFO - chunk_reset : ci,cj ',ci,cj

      if (.NOT.allocated(CH)) allocate(CH(ci,cj))
      if (ltest2) write(*,*) ' INFO - efter alloc CH ',NN,NBB,ci,cj,allocated(CH
(ci,cj)%BL)

      if (.NOT.allocated(CH(ci,cj)%BL)) allocate(CH(ci,cj)%BL(NBB,NBB))
      if (ltest2) write(*,*) ' INFO - efter alloc BL ',ci,cj

      do bi = 1,NBB
        if (ci.EQ.cj) then
          istart = bi
        else
          istart = 1
        end if
        do bj = istart,NBB
          if (.NOT.allocated(CH(ci,cj)%BL(bi,bj)%A)) allocate(CH(ci,cj)%BL(bi,bj
)%A(NN,NN))
          CH(ci,cj)%BL(bi,bj)%A = d0
        end do
      end do

      end subroutine chunk_reset

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! routine to allocate a chunk of data. Either in CH array with indices (ci,cj) O
R
! as allocating CHX in A or B type!  B : (ci,cj) = (-1,-2), A: (ci,cj) = (-1,-1)
!
      subroutine alloc(ci,cj)
      implicit none

      integer      :: bi,bj,ci,cj,istart
!     logical      :: lnalloc

!      lnalloc = lF
!     if (lERR_OBS) write(*,*) ' INFO 120 - alloc : ci,cj ',ci,cj
      if (ltest2) write(*,*) ' INFO - alloc : ci,cj ',ci,cj

      if (.NOT.allocated(CH)) then
        write(*,*)' do not come here! '
!        lnalloc = lT
        allocate(CH(ci,cj))
      end if
!     if (lERR_OBS) write(*,*) ' INFO 128 - efter alloc CH ',NN,NBB,ci,cj,alloca
ted(CH(ci,cj)%BL)
      if (ltest2) write(*,*) ' INFO - efter alloc CH ',NN,NBB,ci,cj,allocated(CH
(ci,cj)%BL)
```

```
      if (allocated(CH(ci,cj)%BL)) deallocate(CH(ci,cj)%BL)
!      if (.NOT.allocated(CH(ci,cj)%BL)) then
      allocate(CH(ci,cj)%BL(NBB,NBB))

!     if (lERR_OBS) write(*,*) ' INFO 135  - efter alloc BL ',ci,cj
      if (ltest2) write(*,*) ' INFO - efter alloc BL ',ci,cj

      do bi = 1,NBB
        if (ci.EQ.cj) then
          istart = bi
        else
          istart = 1
        end if
        do bj = istart,NBB
          if (.NOT.allocated(CH(ci,cj)%BL(bi,bj)%A)) allocate(CH(ci,cj)%BL(bi,bj
)%A(NN,NN))
!          CH(ci,cj)%BL(bi,bj)%A = d0
        end do
      end do
!      if (lnalloc) deallocate(CH)

      end subroutine alloc

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! routine to deallocate chunks of data. Either in CH array with indices (ci,cj)
OR
! as deallocating CHX in A or B type!  B : (ci,cj) = (-1,-2), A: (ci,cj) = (-1,-
1)

      subroutine dealloc(ci,cj)
      implicit none

      integer  :: bi,bj,ci,cj,istart

      do bi = 1,NBB
        if (ci.EQ.cj) then
          istart = bi
        else
          istart = 1
        end if
        do bj = istart,NBB
          if (allocated(CH(ci,cj)%BL(bi,bj)%A)) deallocate(CH(ci,cj)%BL(bi,bj)%A
)
        end do
      end do
!      if (allocated(CH(ci,cj)) deallocate(CH(ci,cj))
      if (allocated(CH(ci,cj)%BL)) deallocate(CH(ci,cj)%BL)

      end subroutine dealloc

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! routine to allocate a chunk of data used for subsums

      subroutine alloc_CHB(CHB)

      integer      :: bi,bj
      type(chunk)  :: CHB

      if (.NOT.allocated(CHB%BL)) allocate(CHB%BL(NBB,NBB))

      do bi = 1,NBB
        do bj = 1,NBB
          if (.NOT.allocated(CHB%BL(bi,bj)%A)) allocate(CHB%BL(bi,bj)%A(NN,NN))
          CHB%BL(bi,bj)%A = d0
        end do
      end do

      end subroutine alloc_CHB
```



```
! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! routine to deallocate CHB chunk

      subroutine dealloc_CHB(CHB)

      integer      :: bi,bj
      type(chunk)  :: CHB

      do bi = 1,NBB
        do bj = 1,NBB
          if (allocated(CHB%BL(bi,bj)%A)) deallocate(CHB%BL(bi,bj)%A)
        end do
      end do
      if (allocated(CHB%BL)) deallocate(CHB%BL)

      end subroutine dealloc_CHB

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine alloc_SOL(NCX)

      integer      :: ci,NCX,bi

      if (.NOT.allocated(SOL))then
        allocate(SOL(NCX))
!       write(*,*)' SOL allocated NCX= ',NCX
      else
        write(*,*)' ERROR - SOL already allocated! '
      end if
      do ci = 1,NCX
        allocate(SOL(ci)%VEC(NBB))
        do bi = 1,NBB
          allocate(SOL(ci)%VEC(bi)%b(NN))
          SOL(ci)%VEC(bi)%b = d0
        end do
      end do

      end subroutine alloc_SOL

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine dealloc_SOL(NCX)

      integer      :: bi,ci,NCX

      do ci = 1,NCX
        do bi = 1,NBB
          if (allocated(SOL(ci)%VEC(bi)%b)) deallocate(SOL(ci)%VEC(bi)%b)
        end do
        if (allocated(SOL(ci)%VEC)) deallocate(SOL(ci)%VEC)
      end do
      if (allocated(SOL)) deallocate(SOL)

      end subroutine dealloc_SOL

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! routine to allocate an array of chunks used for get_chunk_subsums

      subroutine alloc_A_chunks(ci)

      integer   :: bi,bj,ci,h

      if (.NOT.allocated(A_CH)) allocate(A_CH(ci-1))

      do h = 1,ci-1
        if (.NOT.allocated(A_CH(h)%BL)) allocate(A_CH(h)%BL(NBB,NBB))
        do bi = 1,NBB
          do bj = 1,NBB
```

```
          if (.NOT.allocated(A_CH(h)%BL(bi,bj)%A)) allocate(A_CH(h)%BL(bi,bj)%
A(NN,NN))
          A_CH(h)%BL(bi,bj)%A = d0
        end do
      end do
    end do

    end subroutine alloc_A_chunks

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! routine to deallocate CHB chunk

      subroutine dealloc_A_chunks(ci)

      integer   :: bi,bj,ci,h

      do h = 1,ci-1
        do bi = 1,NBB
          do bj = 1,NBB
            if (allocated(A_CH(h)%BL(bi,bj)%A)) deallocate(A_CH(h)%BL(bi,bj)%A)
          end do
        end do
        if (allocated(A_CH(h)%BL)) deallocate(A_CH(h)%BL)
      end do
      if (allocated(A_CH)) deallocate(A_CH)

      end subroutine dealloc_A_chunks

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! routine to close chunk files.

      subroutine chunk_file_close(ci,cj)

      integer  :: ci,cj

      close(filno(ci,cj))

      end subroutine chunk_file_close

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! routine to set chunk of data to zero

      subroutine reset_CHB(CHB)
      implicit none

      integer      :: bi,bj
      type(chunk)  :: CHB

      if (.NOT.allocated(CHB%BL)) allocate(CHB%BL(NBB,NBB))

      do bi = 1,NBB
        do bj = 1,NBB
          if (.NOT.allocated(CHB%BL(bi,bj)%A)) allocate(CHB%BL(bi,bj)%A(NN,NN))
          CHB%BL(bi,bj)%A = d0
        end do
      end do

      end subroutine reset_CHB

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine chunk_write(ci,cj)
!bso
      use m_timing
!bso
      implicit none
```



```
! block numbers in the chunk files are renamed from the global numbers
! to local chunk numbers. The column wise numbering direction is kept
! ci,cj is chunk indices and ic is CHB(ic) number

!   [1:NBB*(NBB+1)/2] for diagonal chunks and
!   [1:NBB*NBB] for rectangular chunks

      integer :: bi,bj,k,ci,cj,istart,ierr
      logical :: lopen

     call timer('chunk_write',1)
! k corresponds to block number
! filenumber is closely associated to chunk number

      k      = 0
      istart = 1

      inquire (filno(ci,cj),OPENED=lopen)
      if (ltest3) write(*,*)' L3 - INFO ci,cj,filno,filename: ',ci,cj,filno(ci,c
j),trim(filename(ci,cj))
      if (.NOT.lopen) open(filno(ci,cj),file=filename(ci,cj),access='direct',rec
l=8*NN*NN)
!'
      do bi = 1, NBB
        if (ci.EQ.cj) istart = bi
        do bj = istart, NBB
          k = k + 1
!          write(*,*) 'filno,k,',filno,k
          write(filno(ci,cj),rec=k,iostat=ierr,err=100) CH(ci,cj)%BL(bi,bj)%A
!          if (ci.EQ.4.AND.cj.EQ.8)then
!            write(*,*)' chunk 4,8 blocks: ',bi,bj
!            call block_write(NN,CH(ci,cj)%BL(bi,bj)%A)
!          end if
        end do
      end do

      close(filno(ci,cj))
      call timer('chunk_write',2)
      return
100   write(*,*)' ERROR WRITE encountered iostat= ',ierr,filno(ci,cj),filename(c
i,cj),k
      stop 'FATAL ERROR'
      end subroutine chunk_write

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine chunk_read(ci,cj)
!bso
      use m_timing
!bso
      implicit none

! block numbers in the chunk files are renamed from the global numbers
! to local chunk numbers. The column wise numbering direction is kept

!   [1:NBB*(NBB+1)/2] for diagonal chunks and
!   [1:NBB*NBB] for rectangular chunks

      integer :: bi,bj,k,ci,cj,istart,ierr
      logical :: lopen

      call timer('chunk_read',1)
      k      = 0
      istart = 1

      if (lTEST2) write(*,*)' fil no. og navn :',ci,cj
! ,filno(ci,cj),filename(ci,cj),NN,NBB
! determine whether it is a diagonal or a rectangular datachunk:
```

```
      inquire (filno(ci,cj),OPENED=lopen) ! should not be needed, bso
      if (.NOT.lopen) open(filno(ci,cj),file=filename(ci,cj),access='direct',rec
l=8*NN*NN)
!'
      do bi = 1, NBB
        if (ci.EQ.cj) istart = bi
        do bj = istart, NBB
          k = k + 1
          if (lTEST2.and.k.lt.3) write(*,*)' reading block ',ci,cj,bi,bj,filno(c
i,cj),filename(ci,cj),size(CH(ci,cj)%BL(bi,bj)%A)
          read(filno(ci,cj),rec=k,IOSTAT=ierr,ERR=100) CH(ci,cj)%BL(bi,bj)%A
!         if (lTEST2) write(*,*)' reading block ',bi,bj

        end do
      end do
      if (lTEST2) write(*,*)' finished chunk_read '
      close(filno(ci,cj))
      call timer('chunk_read',2)
      return
100   write(*,*)' ERROR READ encountered iostat= ',ierr,filno(ci,cj),filename(ci
,cj),k
      stop 'FATAL ERROR'
      end subroutine chunk_read

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine chunk_read_CHB(ci,cj,CHB)
      implicit none

! block numbers in the chunk files are renamed from the global numbers
! to local chunk numbers. The column wise numbering direction is kept

!   [1:NBB*(NBB+1)/2] for diagonal chunks and
!   [1:NBB*NBB] for rectangular chunks

      integer      :: bi,bj,k,ci,cj,istart
      type(chunk)  :: CHB
      logical      :: lopen

      k      = 0
      istart = 1

! determine whether it is a diagonal or a rectangular datachunk:

      INQUIRE (filno(ci,cj),OPENED=lopen)
      if (.NOT.lopen) open(filno(ci,cj),file=filename(ci,cj),access='direct',rec
l=8*NN*NN)
!'
      do bi = 1, NBB
        if (ci.EQ.cj) istart = bi
        do bj = istart, NBB
          k = k + 1
          read(filno(ci,cj),rec=k) CHB%BL(bi,bj)%A
        end do
      end do

      end subroutine chunk_read_CHB

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine read_A_chunks(ci)
      implicit none

! block numbers in the chunk files are renamed from the global numbers
! to local chunk numbers. The column wise numbering direction is kept

!   [1:NBB*(NBB+1)/2] for diagonal chunks and
!   [1:NBB*NBB] for rectangular chunks
```



```
      integer :: bi,bj,h,k,ci
      logical :: lopen

      do h = 1,ci-1
!       filno = ci*(ci-1)/2 + h
        inquire (filno(h,ci),OPENED=lopen)
        if (.NOT.lopen) open(filno(h,ci),file=filename(h,ci),access='direct',rec
l=8*NN*NN)
!'
        k = 0
        do bi = 1, NBB
          do bj = 1, NBB
            k = k + 1
            read(filno(h,ci),rec=k) A_CH(h)%BL(bi,bj)%A
!           if (k.eq.1)write(*,775)A_CH(h)%BL(bi,bj)%A(1,1)
!775        format(' read_A_ch: A(1,1) ',d14.5)
          end do
        end do
        close(filno(h,ci))
      end do

      end subroutine read_A_chunks

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine single_col_read(Ncol)
      implicit none

      integer    :: bi,ci,bj,Ncol,NCX,NBX,NX
!     logical    :: l

! Find chunk number (NCX), block number (NBX) and relative col number bj:

!      write(*,*)' from single_col_read: Ncol =  ' ,Ncol
      call get_col_num(Ncol,NCX,NBX,NX)
!      write(*,*)' resultat fra get_col_num: ', NCX,NBX,NX
!      NCX = 4
!      call alloc_SOL(NCX)

! The column is extracted as:
! 1. extract up to chunk before chunk with block containing last bj element
! 2. extract from chunk containing bj last element up to block befre last block
! 3. extract from last block
!      write(*,*)' NCX:', NCX
      do ci = 1,NCX-1
        call alloc(ci,NCX)
        call chunk_read(ci,NCX)
        do bj = 1,NBB
          SOL(ci)%VEC(bj)%b(:) = CH(ci,NCX)%BL(bj,NBX)%A(1:NN,NX)
          if (ltest2.AND.N.LE.40) write(*,'(20D14.7)') SOL(ci)%VEC(bj)%b(:)
        end do
        call dealloc(ci,NCX)
!        call chunk_file_close(ci,NCX)
        close(filno(ci,NCX))
      end do

!      write(*,*)' inden ny alloc:', NCX

      call alloc(NCX,NCX)
      call chunk_read(NCX,NCX)
      do bi = 1,NBX-1
        SOL(NCX)%VEC(bi)%b(:) = CH(NCX,NCX)%BL(bi,NBX)%A(1:NN,NX)
        if (ltest2.OR.N.LE.40) write(*,'(20D14.7)') SOL(NCX)%VEC(bi)%b(:)
      end do

!      write(*,*)' inden SOL:',NBX,NX

      SOL(NCX)%VEC(NBX)%b(1:NX) = CH(NCX,NCX)%BL(NBX,NBX)%A(1:NX,NX)
```

```
      call dealloc(NCX,NCX)

!      call chunk_file_close(NCX,NCX)
      close(filno(NCX,NCX))

      if (ltest2.OR.N.LE.40) write(*,'(20D14.7)') SOL(NCX)%VEC(NBX)%b(1:NX)
!      call dealloc_SOL(NCX)

      end subroutine single_col_read

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine single_row_read(Nrow,NC_max)
! programmed by M.V. Changed 2012-05-12 by cct.
      implicit none

      integer         :: jc,jb,Nrow,NC_max,NCX,NBX,NX,ibstart,i99,ii,&
                         ipos,k99
      logical         :: lopen,lexist

! Find chunk number (NCX), block number (NBX) and relative col number bj:

! change 2012-08-19.
      if (mod(Nrow,NN*NBB).ne.1) then
!      write(*,*)' nrow not fist roe in a block '
!      Nrow=Nrow-mod(Nrow,NN*NBB)+NN*NBB+1
      end if
      call get_col_num(Nrow,NCX,NBX,NX)
      call alloc_SOL(NC_max)

!     ibstart = NBX
      ibstart = 1
      ii      = NX
       if (lf) write(*,*)' from single row Nrow,NCX,NBX,NX,NN,NC_max ',Nrow,NCX,
NBX,NX,NN,NC_max

      inquire(99,OPENED=lopen)
      if (.not.lopen) write(*,*)' lopen ',lopen

! The row is extracted as:
! 1. extract up to chunk before chunk with block containing last bj element
! 2. extract from chunk containing bj last element up to block befre last block
! 3. extract from last block

      if (.NOT.allocated(CH)) allocate(CH(NCX,NC_max))
      i99=1
      do jc = NCX+1,NC_max
        call alloc(NCX,jc)
        call chunk_read(NCX,jc)
        do jb = ibstart,NBB
          if (lf) write(*,99) jb,jc,ibstart,NC_max,ii,NN
99        format(' jb,jc,ibstart,NC_max,ii,NN: ',6I4)
! experimental change 2011-03-21.
          SOL(jc)%VEC(jb)%b(1:NN) = CH(NCX,jc)%BL(NBX,jb)%A(NX,1:NN)
!         SOL(jc)%VEC(jb)%b(ii:NN) = CH(NCX,jc)%BL(NBX,jb)%A(NX,ii:NN)
          if (lf) write(*,'(20d14.5)') SOL(jc)%VEC(jb)%b(1:NN)
!         if (ibstart.eq.1) then
          if (ii.eq.1.or.ibstart.eq.1) then
! change 2012-08-11 and change back 2012-08-18.
           do k99=1,NN
            write(99) SOL(jc)%VEC(jb)%b(k99)
!           if (lf) write(*,*) SOL(jc)%VEC(jb)%b(k99)
            i99=i99+1
           end do
!         write(*,'(20d14.5)') sqrt(SOL(jc)%VEC(jb)%b(ii:1))
!         call block_write(NN,CH(NCX,jc)%BL(NBX,jb)%A)
          ii = 1
         end if
        end do
```



```
        call dealloc(NCX,jc)
        close(filno(NCX,jc))
        ibstart = 1
      end do
      deallocate(CH)
      call dealloc_SOL(NC_max)
      write(*,*)'error-estimates output to unit 99 as ',i99-1,' records'
      INQUIRE(99,OPENED=LOPEN,EXIST=LEXIST,POS=IPOS)
      IF (.NOT.LOPEN) WRITE(*,*)' from m_cholsol: OPENED,EXIST,POS ',&
      LOPEN,LEXIST,IPOS

      end subroutine single_row_read

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine single_row_readx(Nrow,NC_max)
      implicit none

      integer                                  :: jc,jb,Nrow,NC_max,   &
                                                  NCX,NBX,NX,jmax,&
!                                                 NCX,NBX,NX,ifil,jmax,&
                                                  ibstart,ii
      type(sol_chunk),allocatable,dimension(:) :: CC
      logical                                  :: lDUMP
!     logical                                  :: lDUMP,lOPEN

      lDUMP = .true.
! Find chunk number (NCX), block number (NBX) and relative col number NX:

      call get_col_num(Nrow,NCX,NBX,NX)
!      write(*,*)' from single rowx Nrow,NCX,NBX,NX,NN,NC_max ',&
!      Nrow,NCX,NBX,NX,NN,NC_max

      ibstart = NBX
      ii      = NX

! -- Allocate solution chunk/vector: --
      jmax = NC_max-NCX+1
      allocate(CC(jmax))
      do jc = 1,jmax
        allocate(CC(jc)%VEC(NBB))
        do jb = 1,NBB
          allocate(CC(jc)%VEC(jb)%b(NN))
        end do
      end do

! The row is extracted as:

      do jc = NCX,NC_max
        call alloc(NCX,jc)
        call chunk_read(NCX,jc)
        do jb = ibstart,NBB
          if (lF) write(*,99) jb,jc,ibstart,NC_max
99        format(' jb,jc,ibstart,NC_max: ',4I4)
! experimental change 2011-03-21.
          CC(jc)%VEC(jb)%b(ii:NN) = CH(NCX,jc)%BL(NBX,jb)%A(NX,ii:NN)
          write(*,'(20d14.7)') CC(jc)%VEC(jb)%b(ii:NN)
!         write(*,'(20d14.5)') sqrt(SOL(jc)%VEC(jb)%b(ii:1))
!         call block_write(NN,CH(NCX,jc)%BL(NBX,jb)%A)
          ii = 1
        end do

!        do jb = 1,NBB
!          if (jc.GT.NCX.OR.jb.GE.NBX) then
!            CC(jc-NCX+1)%VEC(jb)%b(:) = CH(NCX,jc)%BL(NBX,jb)%A(1:NN,NX)
!            if (ltest2) write(*,'(20D14.7)') CC(jc-NCX+1)%VEC(jb)%b(:)
!          end if
!        end do
        call dealloc(NCX,jc)
```

```
      end do

      if (lDUMP) then
! ... inquire open write close
        open(filno(0,1)-1,file='SINGLE_ROW.DAT')
        do jc = 1,jmax
          do jb = 1,NBB
            if (jc.GT.NCX.OR.jb.GE.NBX) then
              write(filno(0,1)-1,102) CC(jc)%VEC(jb)%b(:)
102           format(20D14.7)
            end if
          end do
        end do
        close(filno(0,1)-1)
      end if

! deallocate:
      do jc = 1,jmax
        do jb = 1,NBB
          deallocate(CC(jc)%VEC(jb)%b)
        end do
        deallocate(CC(jc)%VEC)
      end do
      deallocate(CC)

      end subroutine single_row_readx

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine single_col_readx(Ncol,NC_max)
      implicit none

      integer                                  :: ic,ib,Ncol,NC_max,   &
                                                  NCX,NBX,NX
!                                                 NCX,NBX,NX,ifil
      type(sol_chunk),allocatable,dimension(:) :: CC
      logical                                  :: lDUMP
!     logical                                  :: lDUMP,lOPEN

      lDUMP = .true.
! Find chunk number (NCX), block number (NBX) and relative col number NX:

      call get_col_num(Ncol,NCX,NBX,NX)
       write(*,*)' from single colx Ncol,NCX,NBX,NX,NN,NC_max ',Ncol,NCX,NBX,NX,
NN,NC_max

! -- Allocate solution chunk/vector: --
!     imax = NC_max
      allocate(CC(NC_max))
      do ic = 1,NC_max
        allocate(CC(ic)%VEC(NBB))
        do ib = 1,NBB
          allocate(CC(ic)%VEC(ib)%b(NN))
        end do
      end do

! The row is extracted as:

      do ic = 1,NC_max
        call alloc(ic,NCX)
        call chunk_read(ic,NCX)
        do ib = 1,NBB
          if (ic.LT.NCX.OR.ib.LE.NBX) then
            CC(ic)%VEC(ib)%b(:) = CH(ic,NCX)%BL(ib,NBX)%A(1:NN,NX)
            if (ltest2) write(*,'(20D14.7)') CC(ic)%VEC(ib)%b(:)
          end if
        end do
        call dealloc(ic,NCX)
      end do
```



```
      if (lDUMP) then
! ... inquire open write close
        open(filno(0,1)-1,file='SINGLE_ROW.DAT')
        do ic = 1,NC_max
          do ib = 1,NBB
            if (ic.GT.NCX.OR.ib.GE.NBX) then
              write(filno(0,1)-1,102) CC(ic)%VEC(ib)%b(:)
102           format(20D14.7)
            end if
          end do
        end do
        close(filno(0,1)-1)
      end if

! deallocate:
      do ic = 1,NC_max
        do ib = 1,NBB
          deallocate(CC(ic)%VEC(ib)%b)
        end do
        deallocate(CC(ic)%VEC)
      end do
      deallocate(CC)

      end subroutine single_col_readx

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! input N1,N2 are 1. and last column numbers of requested read.

      subroutine diagonal_read(N1,N2)
      implicit none

      integer                                  :: ic,ib,N1,N2,NC1,NC2, &
                                                  NB1,NB2,NX1,NX2,&
!                                                 NB1,NB2,NX1,NX2,ifil,&
                                                  Nbstart,Nbstop,      &
                                                  Nstart,Nstop,i
      type(sol_chunk),allocatable,dimension(:) :: CC
      logical                                  :: lDUMP

      lDUMP = .true.
! Find chunk number (NCX), block number (NBX) and relative col number NX:

      call get_col_num(N1,NC1,NB1,NX1)
      call get_col_num(N2,NC2,NB2,NX2)

      write(*,*)' from diagonal_read N1,NC1-2,NB1-2,NX1-2,NN,N2 ',N1,NC1,NC2,NB1
,NB2,NX1,NX2,NN,N2

! -- Allocate solution chunk/vector: --
      allocate(CC(NC1:NC2))
      do ic = NC1,NC2
        allocate(CC(ic)%VEC(NBB))
        do ib = 1,NBB
          allocate(CC(ic)%VEC(ib)%b(NN))
        end do
      end do

! The diagonal is extracted as:

      do ic = NC1,NC2
        call alloc(ic,ic)
        call chunk_read(ic,ic)
        do ib = 1,NBB
          do i = 1,NN
            CC(ic)%VEC(ib)%b(i) = CH(ic,ic)%BL(ib,ib)%A(i,i)
            if (ltest2) write(*,'(20D14.7)') CC(ic)%VEC(ib)%b(:)
          end do
        end do
```

```
        call dealloc(ic,ic)
      end do

      if (lDUMP) then        ! ... inquire open write close ??
        open(filno(0,1)-1,file='DIAGONAL.DAT')
        do ic = NC1,NC2
          if (ic.EQ.NC1) then
            Nbstart = NB1
          else
            Nbstart = 1
          end if
          if (ic.EQ.NC2) then
            Nbstop = NB2
          else
            Nbstop = NB2
          end if
          do ib = Nbstart,Nbstop
            if (ic.EQ.NC1.AND.ib.EQ.NB1) then
              Nstart = NX1
            else
              Nstart = 1
            end if
            if (ic.EQ.NC2.AND.ib.EQ.NB2) then
              Nstop = NX2
            else
              Nstop = NN
            end if
            write(filno(0,1)-1,102) CC(ic)%VEC(ib)%b(Nstart:Nstop)
          end do
        end do
102     format(20D14.7)
        close(filno(0,1)-1)
      end if

! deallocate CC:
      do ic = NC1,NC2
        do ib = 1,NBB
          deallocate(CC(ic)%VEC(ib)%b)
        end do
        deallocate(CC(ic)%VEC)
      end do
      deallocate(CC)

      end subroutine diagonal_read

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine get_col_num(NNN,NCX,NBX,NX)
      implicit none

! delivering the numbers of chunk, block and single data for
! column number NNN

      integer, INTENT(out)  :: NCX,NBX,NX
      integer, INTENT(in)   :: NNN                            ! column number NN
N

      NBX = INT(NNN/NN)  + MERGE(1,0,(MOD(NNN,NN).GT.0))      ! col block number

      NX  = NNN - (NBX-1)*NN                                  ! col number in bl
ock
      NCX = INT(NBX/NBB) + MERGE(1,0,(MOD(NBX,NBB).GT.0))     ! col chunk number

!  Correcting NBX to be the block number within last chunk:

      NBX = mod(NBX-1,NBB)+1                                 ! col block number
in chunk
      if (ltest2) write(*,*)' INFO - get_col_num():', NNN,NCX,NBX,NX
```



```
      end subroutine get_col_num

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine get_row_num(N1,N2,NCX1,NCX2,NBX,NX)
      implicit none

! delivering the numbers of start and end chunk, block and single and relatve ro
w number for

      integer, INTENT(out)  :: NCX1,NCX2,NBX,NX
      integer, INTENT(in)   :: N1,N2                             ! column number NNN
      NBX = INT(N1/NN)  + MERGE(1,0,(MOD(N1,NN).GT.0))      ! col block number
      NX  = N1 - (NBX-1)*NN                                ! col number in block
      NCX1 = INT(NBX/NBB) + MERGE(1,0,(MOD(NBX,NBB).GT.0))   ! col chunk number
!  Correcting NBX to be the block number within last chunk:

      NBX = mod(NBX-1,NBB)+1                                   ! col block numbe
r in chunk
      NCX2 = NCX1
      if (ltest2) write(*,*)' INFO - get_col_num():', N1,NCX1,NBX,NX

      end subroutine get_row_num

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! not operational yet ! should maybe be implemented alternatively!? with option
for adding multiple cols?
! Ncol : place for column to be inserted
! col  : the column data
! missing : opening and closing chunk files!!
      subroutine add_col(Ncol,col)
      implicit none
!
      integer                                      :: i,j,NCX,NBX,NX
      integer ,  INTENT(in)                        :: Ncol
      type (sol_chunk), dimension(NC),INTENT(in)  :: col

! Find chunk number (NCX), block number (NBX) and relative col number bj:

      call get_col_num(Ncol,NCX,NBX,NX)

      if (ltest) write(*,*) ' INFO - add_col(): Ncol,NCX,NBX,NX',Ncol,NCX,NBX,NX

! The column is added as: (NOT FINISHED!)

      do i = 1,NCX-1
        do j = 1,NBB
          CH(i,NCX)%BL(j,NBX)%A(1:NN,NX) = col(i)%VEC(j)%b(:)
        end do
      end do

      do i = 1,NBX-1
        CH(NCX,NCX)%BL(i,NBX)%A(1:NN,NX) = col(NCX)%VEC(i)%b(:)
      end do

      CH(NCX,NCX)%BL(NBX,NBX)%A(1:NX,NX) = col(NCX)%VEC(NBX)%b(1:NX)

      end subroutine add_col

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine vect2sol(Ncol,col,col_out)

      integer, intent(in)                          :: Ncol
      real*8, dimension(Ncol), intent(in)          :: col
      integer                                      :: i,j,h
      type (sol_chunk), dimension(NC),intent(out) :: col_out
```

```
      if (ltest) write(*,*) ' INFO - vect2col: Ncol ',Ncol

      do i = 1,NC
        do j = 1,NBB
          h = (i-1)*NBB*NN+(j-1)*NN
          col_out(i)%vec(j)%b(1:NN) = col(h+1:h+NN)
        end do
      end do

      end subroutine vect2sol

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine block_write(NNN,A)
      integer                     :: NNN,i,j
      real*8,dimension(NNN,NNN)  :: A

      do i = 1,NNN
        write(*,10) (A(i,j),j=1,NNN)
      end do
10    format(21D14.7)

      end subroutine block_write

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! NOT FINISHED !
      subroutine sol_write(cj)
      implicit none

! block numbers in the sol files are renamed from the global numbers
! to local chunk numbers. The column wise numbering direction is kept
! cj is sol_chunk indices and ic is CHB(ic) number

!   [1:NBB*(NBB+1)/2] for diagonal chunks and
!   [1:NBB*NBB] for rectangular chunks

      integer :: bi,bj,k,ci,cj,istart
      logical :: lopen
!     logical :: lCHA,lopen

! k corresponds to block number
! filenumber is closely associated to chunk number
      stop "routine sol_write does not work" ! added by Brian 6/2/2014
      k      = 0
      istart = 1
      write(*,*)' sol_write, cj= ',cj

!      filno  = ifmax*(ifmax+1)/2 + 1
!      if (ci.EQ.cj) then
!        filno = ci*(ci+1)/2
!        lCHA  = lt
!      else
!        filno = ci + cj*(cj-1)/2
!        lCHA  = lf
!      end if

      inquire (filno(0,1),OPENED=lopen)
      if (.NOT.lopen) open(filno(0,1),file=filename(0,1),access='direct',recl=8*
NN)
      do bi = 1, NBB
!        if (lCHA) istart = bi
        do bj = istart, NBB
          k = k + 1
           write(*,*) 'filno,k,',filno(0,1),k
           write(filno(0,1),rec=k) CH(ci,cj)%BL(bi,bj)%A
!          if (ltest2) call block_write(NN,CH(ci,cj)%BL(bi,bj)%A)
        end do
      end do
```



```
      close(filno(0,1))

      end subroutine sol_write

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! factorizeA factorises diagonal blocks.!
! bi : Relative block number
! ci  : Chunk number
! C   : result from get_chunk_subsum
! A   : result from get_block_subsum
! written by mv, changed 2012-05-11 by cct.

      subroutine factorizeA(bi,ci,C)

      implicit none

      integer                                 :: i,j,k,bi_abs
      integer, INTENT(in)                     :: bi,ci
      real*8                                  :: elem,eps,bii,bij
      real*8, dimension(NN,NN)                :: B,A
      real*8, dimension(NN,NN), intent(IN) :: C

      if (ltest3) write(*,*)' ------factA,bi,ci  ',bi,ci

      elem = d0
      eps  = 1.d-12
      bi_abs = (ci-1)*NBB*NN+(bi-1)*NN
      B    = CH(ci,ci)%BL(bi,bi)%A

      if (bi.GT.1) then
        call get_block_subsumA(bi,ci,A)
!       write(*,731)bi,ci,A
!731    format(' b-subsumA: bi,ci,A ',2i3,4f10.5)
!        if (ltest3) call block_write(NN,A)
      else
        A = d0
      end if

      do i = 1,NN

! Diagonal element:

        do j = 1, i-1
          elem = elem + B(j,i)**2
          if (lf) write(*,*)' diag.,i,j,elem,B(j,i),(i,j) ',i,j,elem,B(j,i),B(i,
j)
        end do

        bii=B(i,i)
        if (B(i,i)-elem-C(i,i)-A(i,i).LT.0.d0) then
! change 2012-05-09 by cct.
         if ((bi_abs+i).lt.N) then
          write(*,*)' ERROR - sqrt,i,j,bi,ci,B(i,i),elem,CH,BL',i,j,bi,ci,B(i,i)
,elem,C(i,i),A(i,i),N,bi_abs+i
          B(i,i) = 1.d0
         else
          B(i,i) = B(i,i)-elem-C(i,i)-A(i,i)
         end if
        else
         if ((bi_abs+i).lt.N) then
          B(i,i) = sqrt(B(i,i)-elem-C(i,i)-A(i,i))
!200       format(' factA: ci,bi,i,B: ',3I5,D14.7)
!          write(*,200) ci,bi,i,B(i,i)
         else
          B(i,i) = B(i,i)-elem-C(i,i)-A(i,i)
         end if
        end if
!       write(*,202)i,bii,b(i,i),elem,c(i,i),a(i,i)
```

```
 202    format(' factA: i,bii,Bii,elem,cii,aii ',i3,5d15.7)
        elem   = d0

! Side elements:

        do j = i+1, NN

          do k = 1, i-1
            elem = elem + B(k,i)*B(k,j)
          end do

          if (ABS(B(i,i)).LT.eps) then
           if (bi_abs+i.le.N)  write(*,*)' ERROR - divide 0 !!, i,j,k,ci,bi,bi_a
bs,N,B ',i,j,k,ci,bi,bi_abs,N,B(i,i)
           B(i,j)=d1
          else
           bij=B(i,j)
           B(i,j) = (B(i,j)-elem-C(i,j)-A(i,j))/B(i,i)
!          write(*,213)i,j,bij,B(i,j),C(i,j),A(i,j),B(i,i)
!213       format(' factA-side: i,j,bij,Bij,Cij,Aij,Bii ',2i3,6f10.5)
          end if
          elem   = d0
        end do
      end do

      CH(ci,ci)%BL(bi,bi)%A = B
!     if (ltest) call block_write(NN,B)

      end subroutine factorizeA

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! bi,bj : block indices
! ci,cj   : chunk indices
! C       : result from get_chunk_subsum
! A       : result from get_block_subsum

      subroutine factorizeB(bi,bj,ci,cj,C)

      implicit none
      integer                               :: i,j,k
      integer                               :: bi,bj,ci,cj
      real*8                                :: elem,bij
      real*8, dimension(NN,NN)              :: B,A,D
      real*8, dimension(NN,NN), intent(IN) :: C

      if (ltest3) write(*,*)' ------factB,bi,bj,ci,cj  ',bi,bj,ci,cj

      elem = d0
      A    = CH(ci,ci)%BL(bi,bi)%A
      B    = CH(ci,cj)%BL(bi,bj)%A

      if (bi.GT.1) then
        call get_block_subsumB(bi,bj,ci,cj,D)
      else
        D = d0
      end if

! Only side elements:

      do i = 1,NN
        do j = 1,NN
          do k = 1, i-1
            elem = elem + B(k,j)*A(k,i)
          end do

          bij=B(i,j)
          B(i,j) = (B(i,j)-elem-C(i,j)-D(i,j))/A(i,i)
!         write(*,200) i,j,bij,B(i,j),elem,C(i,j),D(i,j),A(i,i)
```



```
!200       format(' factB: ',2I4,6F10.5)
          elem   = d0
        end do
      end do

      CH(ci,cj)%BL(bi,bj)%A = B
!     write(*,201)bi,bj,B
!201  format(' factB: bi,bj,B ',2i3,/,4(4F10.5,/))

      end subroutine factorizeB

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! get subsum from chunks above the A chunk which is to be factorised.

! in this routine OMP should be applied!

! bi relative block number
! ic is index counter of chunk up to ci
! ci,cj chunk indices
! CHX result chunk

      subroutine get_chunk_subsumA(ci,CHX)
!bso
      use m_timing
!bso
      implicit none

      integer                  :: i,j,k,hj
!     integer                  :: i,j,k,hj,Nstart
!     integer                  :: bi,bj,ci,cj,ic
      integer                  :: bi,bj,ci,ic
      real*8                   :: elem
      real*8,dimension(NN,NN) :: A,B,C
      type(chunk)              :: CHX

! Read previous reduced blocks 'above' A and B, and making the column product su
ms:

      call timer('get_chunk_subsumA',1)
      C    = d0
      elem = d0

      if (ltest2) write(*,*) ' subt: get_chunk_subsumA ,ci ' ,ci

      do ic = 1,ci-1            ! go through chunks above current (ci)

!BSO!$OMP PARALLEL default (none) &
!BSO!$OMP SHARED(CHX,A_CH,ic,NBB,NN) PRIVATE(bj,hj,bi,A,B,j,k,i) FIRSTPRIVATE(el
em,C)
!BSO!$OMP DO
!BSO!   reduction (+:CHX)

!$OMP PARALLEL DEFAULT(SHARED)
!$OMP DO COLLAPSE(2) PRIVATE(bj,hj,bi,A,B,C,j,k,i,elem) SCHEDULE(DYNAMIC,2)
        do bj = 1,NBB            ! go through blocks col-wise
          do hj = 1,NBB          ! go through remaining columns of blocks
            C = d0
            do bi = 1,NBB        ! step through number of rows in block
              A = A_CH(ic)%BL(bi,bj)%A
              B = A_CH(ic)%BL(bi,hj)%A
              do j = 1,NN
                do k = 1,NN
!BSO                  elem = d0
!BSO                  do i = 1,NN
!BSO                    elem = elem + A(i,j)*B(i,k)
!BSO                  end do
!BSO                  C(j,k) = C(j,k) + elem
!                 elem = d0
```

```
                C(j,k) = C(j,k) + dot_product(A(:,j),B(:,k))
              end do
            end do
          end do
!BSO $OMP CRITICAL
          CHX%BL(bj,hj)%A = CHX%BL(bj,hj)%A + C
!BSO $OMP END CRITICAL
!          write(*,701)ic,bj,hj,C(1,1),C(2,2),C(3,3)
!701       format(' ic,bj,hj,C ',3i3,4f10.5)
!          write(*,702)CHX%BL(bj,hj)%A(1,1),A(2,2),A(3,3)
!702       format(5D14.4)
!           C = d0
        end do
      end do
!$OMP END DO
!$OMP END PARALLEL

      end do

!      if (ltest2) call block_write(NN,CHX%BL(1,1)%A)

! because of the symmetry: (as in get_block_subsumA ... )
!      do bi = 1,NBB           ! go through blocks row-wise
!        do bj = bi+1,NBB        ! st
!          CHX%BL(bj,bi)%A = CHX%BL(bi,bj)%A
!        end do
!      end do
      call timer('get_chunk_subsumA',2)

      end subroutine get_chunk_subsumA

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! get subsum from chunks above the B chunk which is to be factorised.

! in this routine OMP may NOT be applied! (interferes with other parallelisation
!)

! bi relative block number
! ic is index of chunk up to ci
! ci,cj chunk indices
! CHX result chunk

      subroutine get_chunk_subsumB(ci,cj,CHX)
!bso
      use m_timing
!bso
      implicit none

      integer, intent(in)        :: ci,cj
      integer                    :: i,j,k,hj,bi,bj,ic
      real*8                     :: elem
      real*8, dimension(NN,NN)   :: A,B,C
      type(chunk), intent(inout) :: CHX
      type(chunk)                :: B_CH

! Read previous reduced blocks 'above' A and B, and making the column product su
ms:

      C    = d0
      elem = d0

      if (ltest2) write(*,*) ' get_chunk_subsumB ,ci,cj ' ,ci,cj

      call alloc_CHB(B_CH) !move outside loop
      do ic = 1,ci-1            ! go through chunks above current (ci)

        call timer('chunk_read_CHB',1)
        call chunk_read_CHB(ic,cj,B_CH)
```



```
        call timer('chunk_read_CHB',2)
        close(filno(ic,cj)) !should be moved to chunk_read_CHB
!        call chunk_file_close(ic,cj)

!        write(*,*) ' get_chunk_subsumB efter ',ic,NBB,NN

        call timer('elem',1)
!$OMP PARALLEL DEFAULT(SHARED)
!$OMP DO COLLAPSE(2) PRIVATE(bj,hj,bi,A,B,C,j,k,i,elem) SCHEDULE(DYNAMIC,2)
        do bj = 1,NBB           ! Number of blocks pr. chunk col-wise
          do hj = 1,NBB         ! Number of blocks pr. chunk col ...
            C = d0
            do bi = 1,NBB       ! step up through the rows of blocks
              A = A_CH(ic)%BL(bi,bj)%A
              B = B_CH%BL(bi,hj)%A
              do j = 1, NN
                do k = 1, NN
!                  elem = d0
                  C(j,k) = C(j,k) + dot_product(A(:,j),B(:,k))
!                  do i = 1, NN
!                    elem = elem + A(i,j)*B(i,k)
!                  end do
!                  C(j,k) = C(j,k) + elem
                end do
              end do
            end do
            CHX%BL(bj,hj)%A = CHX%BL(bj,hj)%A + C
            C = d0
          end do
        end do
!$OMP END DO
!$OMP END PARALLEL
        call timer('elem',2)
      end do
      call dealloc_CHB(B_CH)! move outside loop

      end subroutine get_chunk_subsumB

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine get_block_subsumA(bi,ci,C)
      implicit none

      integer                              :: i,j,k,h
      integer, INTENT (IN)                 :: ci,bi
      real*8                               :: elem
      real*8,dimension(NN,NN),INTENT (out) :: C
      real*8,dimension(NN,NN)              :: B,A

      if (ltest2) write(*,*) ' get_block_subsumA ,bi,ci ' ,bi,ci

! Read previous reduced blocks 'above' A, and making the column products:
! when calling this routine the block are presumed already opened and read!

      C    = d0
      elem = d0

!$OMP PARALLEL default (none) &
!$OMP SHARED (bi,ci,NN,CH,C) PRIVATE(h,A,B,j,k,i) FIRSTPRIVATE(elem)
!$OMP DO reduction (+:C)

      do h = 1,bi-1
        A = CH(ci,ci)%BL(h,bi)%A
!        B = CH(ci,ci)%BL(h,bi)%A
        B = A
        do j = 1, NN       ! go through all cols in B
          do k = 1, NN     ! go through all cols in A (because of symmetry only
'Â½ the cols')
            do i = 1, NN   ! go through all elements in col
```

```
            elem = elem + A(i,j)*B(i,k)
          end do

! $OMP CRITICAL
          C(j,k) = C(j,k) + elem
! $OMP CRITICAL
          elem = d0

        end do
      end do
    end do

!$OMP END DO
!  $OMP FLUSH
!$OMP END PARALLEL

!      do j = 1, NN       ! and because of the symmetry:
!        do i = j+1, NN
!          C(j,i) = C(i,j)
!        end do
!      end do

      end subroutine get_block_subsumA

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine get_block_subsumB(bi,bj,ci,cj,C)
      implicit none

      integer                                :: i,j,k,h
      integer, INTENT (IN)                   :: ci,cj,bi,bj
      real*8                                 :: elem
      real*8,dimension(NN,NN),INTENT (out)   :: C
      real*8,dimension(NN,NN)                :: B,A

      if (ltest3) write(*,*) ' get_block_subsumB ,bi,ci ' ,bi,ci

! Read previous reduced blocks 'above' A and B, and making the column products:
! when calling this routine the chunks are presumably already opened and read!

      C    = d0
      elem = d0

      do h = 1,bi-1
        A = CH(ci,ci)%BL(h,bi)%A
        B = CH(ci,cj)%BL(h,bj)%A
        do j = 1, NN
          do k = 1, NN

! $OMP PARALLEL  ! should only be activated when there are idle procs!
! $OMP DO
          do i = 1, NN
            elem = elem + A(i,j)*B(i,k)
          end do

! $OMP END DO
! $OMP END PARALLEL

          C(j,k) = C(j,k) + elem
          elem = d0
        end do
      end do
    end do

      end subroutine get_block_subsumB

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine time_stamp(T1,T2,ID)
```



```
      IMPLICIT NONE

      integer, dimension(8), intent(in)  :: T1
      integer, dimension(8), intent(out) :: T2
      integer                            :: ID
      real                               :: T

      call date_and_time(VALUES = T2)
      T = T2(5)*3600 + T2(6)*60 + T2(7) + T2(8)*0.001 &
        - T1(5)*3600 - T1(6)*60 - T1(7) - T1(8)*0.001
      write(*,*) T," seconds"

      end subroutine time_stamp

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine ID_stamp(T1,T2,PID,ID)
      IMPLICIT NONE

      integer, dimension(8)  :: T1,T2
      integer                :: PID
      character*8            :: ID
      real                   :: T

      call date_and_time(VALUES = T2)
      T = T2(5)*3600 + T2(6)*60 + T2(7) + T2(8)*0.001 &
        - T1(5)*3600 - T1(6)*60 - T1(7) - T1(8)*0.001
      T1 = T2
      write(*,*) ID,PID, ' ',T," seconds"

      END subroutine ID_stamp

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine reassemble
      implicit none

      integer                                 :: ci,cj,bi,bj
      real*8, dimension(NC*NBB*NN,NC*NBB*NN) :: D,B

! testing to reassemble the start matrix:
! Note that reassemble tranposes the main matrix before
! screen output because this will correspond to traditional
! indices notation with i : rows and j : columns

      D = d0
      B = d0

      do ci = 1,NC
        call alloc(ci,ci)
        call chunk_read(ci,ci)
        do bi = 1,NBB
          do bj = bi,NBB
            D(NN*NBB*(ci-1)+(bi-1)*NN+1:NN*NBB*(ci-1)+bi*NN,NN*NBB*(ci-1)+(bj-1)
*NN+1:NN*NBB*(ci-1)+bj*NN) = CH(ci,ci)%BL(bi,bj)%A
          end do
        end do
        call dealloc(ci,ci)

        do cj = ci+1,NC
          call alloc(ci,cj)
          call chunk_read(ci,cj)
          do bi = 1,NBB
            do bj = 1,NBB
              D(NN*NBB*(ci-1)+(bi-1)*NN+1:NN*NBB*(ci-1)+bi*NN,NN*NBB*(cj-1)+(bj-
1)*NN+1:NN*NBB*(cj-1)+bj*NN) = CH(ci,cj)%BL(bi,bj)%A
            end do
          end do
          call dealloc(ci,cj)
```

```
        end do
      end do

! transposing ... :
      do bi = 1,NC*NBB*NN
        do bj = 1,NC*NBB*NN
          B(bi,bj) = D(bj,bi)
        end do
      end do

      write(*,98) NC,NBB,NB,NN,N
98    format(' Reassembled NC,NBB,NB,NN,N :',5I6)
      write(*,formats(1)) B(1:N,1:N)

      end subroutine reassemble

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine chol_sol(err)
      use m_timing
      implicit none

!     integer                                  :: NX,NCX,NBX,i,j,bi,bj,&
      integer                                  :: NX,NCX,NBX,i,bi,bj,&
                                                  ci,cj
      real*8, dimension(NN)                    :: C
      real*8                                   :: err
      type(sol_chunk),allocatable,dimension(:) :: CC
      logical                                  :: lopen

! Solving the block factorised Cholesky (C) reduced equation:
!    L*LT * x = b
! in 2 steps : first as forward solution of:
!   L*y = b  ! this is already done in the choleski decomp!
!followed by backward solution of:
!   LT*x = y

! -------------------------------------
! -- Allocate solution chunk/vector: --
!     write(*,*)' chol_sol ',NC

      call alloc_SOL(NC)

      allocate(CC(NC))
      do ci = 1,NC
        allocate(CC(ci)%VEC(NBB))
        do bi = 1,NBB
          allocate(CC(ci)%VEC(bi)%b(NN))
          CC(ci)%VEC(bi)%b  = d0
        end do
      end do

! --------------------------------------
! ---- collecting solution vector ------
!--- which is already forward solved ---

!     write(*,*)' 1481 '
      call single_col_read(N)

! ---------------------------------
!   first solving last diagonal block in last chunk  (and with pred. error in N,
N):
! ---------------------------------

      call get_col_num(N,NCX,NBX,NX)
      err = SOL(NCX)%VEC(NBX)%b(NX)
! This is the reduced cellar-window, and the square-root should not be calculate
d. 2012-05-10.
!     err = sqrt(err)
```



```
      C = d0

      call alloc(NCX,NCX)
!     write(*,*)' 1496 '
      call chunk_read(NCX,NCX)
! the NX-1 enters because the last equation is NOT included in backsolution - on
ly for error.
! also implemented below (10(?) lines below)
      call back_sol(NX-1,CH(NCX,NCX)%BL(NBX,NBX)%A,SOL(NCX)%VEC(NBX)%b,C)
!     write(*,*)' after back_sol, 1501 '

! -------------------------------------------
! - then solving remaining blocks in last chunk (no. NCX)
! -------------------------------------------

      do bi = NBX-1,1,-1
        C  = d0
        do bj = bi+1,NBX-1
          call AxB_plusC(NN,CH(NCX,NCX)%BL(bi,bj)%A,SOL(NCX)%VEC(bj)%b,C)
        end do
        call AxB_plusC(NX-1,CH(NCX,NCX)%BL(bi,NBX)%A,SOL(NCX)%VEC(NBX)%b,C)
        call back_sol(NN,CH(NCX,NCX)%BL(bi,bi)%A,SOL(NCX)%VEC(bi)%b,C)
      end do
!     write(*,*)' after back_sol, 1515 '
      call dealloc(NCX,NCX)

! ------------------------------------------
! ------ solving remaining chunks: ---------
! ------------------------------------------

      do ci = NCX-1,1,-1     ! and now solve for chunks 1 - NCX-1
        do cj = ci+1,NCX
          call alloc(ci,cj)
          call chunk_read(ci,cj)
          do bi = NBB,1,-1
            if (cj.NE.NCX) then
              do bj = 1,NBB
                call AxB_plusC(NN,CH(ci,cj)%BL(bi,bj)%A,SOL(cj)%VEC(bj)%b,CC(ci)
%VEC(bi)%b)
              end do
            else
              do bj = 1,NBX-1
                call AxB_plusC(NN,CH(ci,NCX)%BL(bi,bj)%A,SOL(NCX)%VEC(bj)%b,CC(c
i)%VEC(bi)%b)
              end do
              call AxB_plusC(NX-1,CH(ci,NCX)%BL(bi,NBX)%A,SOL(NCX)%VEC(NBX)%b,CC
(ci)%VEC(bi)%b)
            end if
          end do
          call dealloc(ci,cj)
        end do

        call alloc(ci,ci)
        call chunk_read(ci,ci)

        do bi = NBB,1,-1      ! and finally for the 1. chunk in the line!
          do bj = bi+1,NBB
            call AxB_plusC(NN,CH(ci,ci)%BL(bi,bj)%A,SOL(ci)%VEC(bj)%b,CC(ci)%VEC
(bi)%b)
          end do
          call back_sol(NN,CH(ci,ci)%BL(bi,bi)%A,SOL(ci)%VEC(bi)%b,CC(ci)%VEC(bi
)%b)
        end do
        call dealloc(ci,ci)
      end do
      write(*,*)' after back_sol, 1550 '

      do ci = 1,NC
```

```
        do bi = 1,NBB
          deallocate(CC(ci)%VEC(bi)%b)
        end do
        deallocate(CC(ci)%VEC)
      end do
      deallocate(CC)

      i = 0
!     write(*,*)' fileno, SOL size ',filno(0,1),size(SOL)
      inquire (filno(0,1),OPENED=lopen)
      if (.NOT.lopen) then
       open(filno(0,1),file=filename(0,1),access='direct',recl=8*NN)
      else
       write(*,*)' file open '
      end if
      do ci = 1,NC
        do bi = 1,NBB
          i = i + 1
           if (ltest3) write(*,*)' insol',i,ci,bi
          write(filno(0,1),rec=i) SOL(ci)%VEC(bi)%b
!         write(*,formats(2)) SOL(ci)%VEC(bi)%b
!         if (ltest.AND.N.LE.30) write(*,formats(2)) SOL(ci)%VEC(bi)%b
          if (ltest3) write(*,801) SOL(ci)%VEC(bi)%b
 801      format(10d9.2)
!          if (ltest.AND.N.LE.30) write(*,formats(2)) CC(ci)%VEC(bi)%b
!          if (ltest.AND.N.LE.30) write(*,formats(2)) SOL(ci)%VEC(bi)%b
        end do
      end do
      call dealloc_SOL(NC)

!      close(filno(0,1))
! WILL BE CLOSED IN MAIN PROGRAM AFTER TRANSFER TO ARRAY B. 2012-03-13.

      end subroutine chol_sol

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine back_sol(NX,U,b,C)
      implicit none
      integer                  :: i,j
      integer, INTENT(in)      :: NX
      real*8 , INTENT(in)      :: U(NN,NN),C(NN)
      real*8 , INTENT(inout)   :: b(NN)
      real*8                   :: s

! Back solution for upper triangular matrix equation system:
!    U * x = b
! the solution x is returned in b
! C contains U*b sums from blocks (and chunks) before the last block

      do i = NX,1,-1
        s = d0
        do j = i+1,NX
          s = s + U(i,j)*b(j)
        end do
        b(i) = (b(i) - s - C(i))/U(i,i)
      end do

      end subroutine back_sol

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine AxB_plusC(NX,A,B,C)
!  PROGRAMMED AUG 2004 BY M. Veicherts
!  THE SUBROUTINE WILL COMPUTE THE PRODUCT OF THE 3*3 MATRICES
!  A AND B AND accumulate sum_loc in C.

      IMPLICIT NONE
```



```
      INTEGER                 :: i,k,NX
      REAL*8, INTENT(in)      :: A(NN,NN),B(NN)
      REAL*8, INTENT(inout)   :: C(NN)

!     if (ltest3) write(*,*)' ------AxB_plusC  ',NX

      do i = 1,NN
        do k = 1,NX
          C(i) = C(i) + A(i,k) * B(k)
        end do
      end do

      END subroutine AxB_plusC

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! Testing RF chol subroutine:

      subroutine reneforsberg(N,fmat,fsol,F)
      IMPLICIT NONE

      INTEGER                               :: i,j,N,h,Nsing
!     INTEGER, dimension(8)                 :: T1,T2
      CHARACTER*16                          :: fmat,fsol
      real*8, INTENT(out)                   :: F(N)
      real*8, dimension(:,:), allocatable  :: A
      real*8, dimension(:),   allocatable  :: B

      if (ltest)write(*,*)' ---------reneforsberg: ',N

      allocate(A(N,N))
      call generate_givens(N,A)
!      allocate(B((N+1)*(N+2)/2))
      allocate(B(n+n*(n+1)/2))

      h = 0

      do i = 1,N
        do j = 1,i
          h    = h + 1
          B(h) = A(i,j)
        end do
      end do
! result vector = 1's:

      do j = 1,N
        B(h+j) = 1.0d0
      end do
!      write(*,fmat) (B(i),i=1,(N+1)*(N+2)/2)

      call chol(B,N,nsing)

!      call time_stamp(T1,T2,0)
      if (ltest.AND.N.LE.29) then
        write(*,fmat) (B(i),i=1,n+N*(N+1)/2)
        write(*,*) ' INFO - Solution: (Nsing:)', Nsing
        write(*,fsol) (B(i),i=N*(N+1)/2+1,N+N*(N+1)/2)
      end if
      F(1:N) = B(1+N*(N+1)/2:N+N*(N+1)/2)

      deallocate(A)
      deallocate(B)
      write(*,*) ' End of RF routine '

      end subroutine reneforsberg

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'

      subroutine init
      implicit none
```

```
! N      Number of equations in system (including result column)
! NN     Number of rows/cols in a block
! NBB    Number of blocks in a row of a chunk
! NB     Number of NNxNN Blocks in a row/col of the full system
! NBB    Number of blocks in a chunk
! NC     Number of chunks in system
! Nnodes number of PCs in the system  - not activated yet

!     integer       :: i,j
      character*16  :: fnum

      if (ltest) write(*,*)' ---- init ',lfirst_chol

      if (lfirst_chol) then
        lfirst_chol = lf
        NF    = 0
        ifmax = 0
        jfmax = 0
        file_root_name_old = file_root_name
      end if

! make formats for small test solutions:
      if (N.LE.30) then
        write(fnum,'(i6)') N-1
        formats(2)   = '('//trim(adjustl(fnum))//'F7.3,/'//')'
        write(fnum,'(i6)') N
        formats(1)   =  '('//trim(adjustl(fnum))//'('//trim(adjustl(fnum))//'F7.
3,/'//')'//')'
        write(fnum,'(i6)') NN
        formats(3) = '('//trim(adjustl(fnum))//'('//trim(adjustl(fnum))//'F7.3,/
'//')'//')'
        write(fnum,'(i6)') NB*NN
        formats(4)  = '('//trim(adjustl(fnum))//'('//trim(adjustl(fnum))//'F7.3,
/'//')'//')'
!        write(*,*)  ' formats for fmat, fsol, fblock, ffull ', (formats(i),i=1,
4)
      end if

      end subroutine init

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! Nnodes - number of machines (PCs) in cluster

      subroutine weight_config(Nnodes)
      implicit none

      integer                    :: i,Nnodes,Nall,NWS,NP
      real*8, dimension(Nnodes) :: w
!     real*8, dimension(Nnodes) :: w,NW
!     real*8                    :: ws,wa,A
      real*8                    :: wa

      if (ltest) write(*,*)' ---- weight_config ',Nnodes

! Matrix 'area':

!       A   = 0.0d0
!      do i = 1,Nnodes
!        A   = A + 1.0d0*(2*i-1)
!      end do

! Matrix area (or rather blocks) pr. (MPI-) PROC:

      wa = NB*(NB+1)/2.0d0/(1.0d0*Nnodes)

!      N(1) = INT(sqrt(2.d0*wa))

      do i = 1,Nnodes
```



```
!!!     w(i) = i/wa
      end do

      NWS = d0
      do i = 1,Nnodes
        w(i) = 0
        NWS  = NWS + INT(NB/Nnodes)
        Nall = Nall
      end do

      end subroutine weight_config

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! compares solution from SOL and RF (delived in F) and writes out differences

      subroutine test_sol(F)
      implicit none

      integer   :: i,k,NX,NBX,NCX
      integer   :: bi,ci
      real*8    :: F(N-1),diff
      logical   :: ldiff

      if (ltest) write(*,*)' ---- test_solution '

      diff  =  1.0d-15
      ldiff = .false.
      if (.not.ldiff) return

      write(*,*) ' INFO - Solution test: ',N
!      write(*,99) F(1:10)
!      write(*,99) SOL(1)%VEC(1)%b(:)

      write(*,*) ' N     RF     CHOL '

      call get_col_num(N-1,NCX,NBX,NX)
      k = 0
      do ci = 1,NCX-1
        do bi = 1,NBB
          do i = 1,NN
            k = k+1
            if (ABS(F(k)-SOL(ci)%VEC(bi)%b(i)).GT.diff) then
              ldiff = .true.
              write(*,100) k,F(k),SOL(ci)%VEC(bi)%b(i)
            else
              if (k.LT.20) write(*,100) k,F(k),SOL(ci)%VEC(bi)%b(i)
            end if
          end do
        end do
      end do

99    format(D25.15)
100   format(I7,2D25.15)

! and for the last chunk:

      do bi = 1,NBX-1
        do i = 1,NN
          k = k+1
          if (ABS(F(k)-SOL(NCX)%VEC(bi)%b(i)).GT.diff) then
            ldiff = .true.
            write(*,100) k,F(k),SOL(NCX)%VEC(bi)%b(i)
          end if
        end do
      end do

! and for the last block:

      do i = 1,NX
```

```
        k = k+1
        if (ABS(F(k)-SOL(NCX)%VEC(NBX)%b(i)).GT.diff) then
          ldiff = .true.
          write(*,100) k,F(k),SOL(NCX)%VEC(NBX)%b(i)
        end if
      end do

      if (ldiff) then
        write(*,*) ' ERROR - Solution does not match RF solution'
      else
        write(*,*) ' INFO - Solution matches RF solution '
      end if

      end subroutine test_sol

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine copy_files(lTEST)

      implicit none

      integer                                    :: i,j,k,kmax,irec,nblocks
      real*8, dimension(NN,NN)                   :: C_COPY
      logical                                    :: lTEST,lEXIST,lOPEN

      kmax=ifmax*(ifmax+1)/2+10
      if (lTEST) write(*,*)' ---- copy_files: ',ifmax,jfmax,kmax

      k=1
      do i = 1,ifmax
        do j = i,jfmax     ! always true or j = 1,jfmax ?
! change 2012-07-26.
          if (i.eq.j) then
           nblocks=NBB*(NBB+1)/2
          else
           nblocks=NBB*NBB
          end if
          write(*,*)' i,j,NBB,nblocks ',i,j,NBB,nblocks
          inquire(file=filename(i,j),EXIST=lEXIST,OPENED=lOPEN)
          if (lEXIST) then
            open(kmax,file=trim(filename(i,j))//'.ORI',access='direct',recl=8*NN
*NN)
            if (.NOT.lOPEN) then
              open(filno(i,j),file=filename(i,j),access='direct',recl=8*NN*NN)
            else
              rewind(filno(i,j))
            end if
            do irec=1,nblocks
              read(filno(i,j),rec=irec) C_COPY
! something wrong here.
              write(kmax,rec=irec) C_COPY
            end do
!200         continue
            close(kmax)
            if (.NOT.lOPEN) close(filno(i,j))
          else
            write(*,*) ' WARNING - file does not exist for copy:  ',i,j,filename
(i,j)
          end if
        end do
      end do

      end subroutine copy_files

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! ltri = true if triangular matrix,else rectangular
! tag - file type : neq,err,errcov,...
! imin start chunk, imax end chunk
```



```
      subroutine add_files_col(imin,imax,jc,ish,lOBSx,lERR_OBSx,lERR_COVx,lSOLx)

      implicit none

      integer                                    :: i,j,filnox,          &
                                                    imin,imax,jc,ish
      character*72,dimension(:,:),allocatable :: filename_buf
      integer,     dimension(:,:),allocatable :: filno_buf
      character*10                               :: fni,fnj
!     logical                                    :: lexist,lOBSx,lSOLx,   &
      logical                                    :: lOBSx,lSOLx,   &
                                                    lERR_OBSx,lERR_COVx

      if (ltest2) then
        write(*,*)' ---- add_files_col ',imin,imax,jc,ish,lOBSx,lERR_OBSx,lERR_C
OVx,lSOLx
!         if (allocated(filno).AND.allocated(filename)) then
!           write(*,*) ' filno,filename: ',size(filno),size(filename)
!           write(*,*) ' filno,filename: ',filno,filename
!         end if
      end if

      if (file_root_name.NE.file_root_name_old) then
!       write(*,*)' WARNING - new filename entered - files created '
        file_root_name_old = file_root_name
        if (allocated(filename)) deallocate(filename)
        if (allocated(filno)) deallocate(filno)
        NF    = 0
        ifmax = 0
        jfmax = 0
      end if
!      if (lERR_OBSx.and.jc.eq.1) then
!        NF    = 0
!        ifmax = 0
!        jfmax = 0
!      end if
! added 2012-01-13.
!     write(*,*)' ifmax,jfmax ',ifmax,jfmax

      allocate(filno_buf(0:MAX(imax,ifmax),MAX(jc,jfmax)))
      allocate(filename_buf(0:MAX(imax,ifmax),MAX(jc,jfmax)))

      if (allocated(filno)) then
        filno_buf(0:ifmax,1:jfmax) = filno(0:ifmax,1:jfmax)
        deallocate(filno)
      end if
      if (allocated(filename)) then
        filename_buf(0:ifmax,1:jfmax) = filename(0:ifmax,1:jfmax)
        deallocate(filename)
      end if

      if (lOBSx) then
!        do j = jmin,jmax      ! for single colum adding : jmin = jmax
         do i = imin, imax   ! create a column of chunk-files:
           NF = NF + 1
           if (i.EQ.jc) then
             filnox = i*(i+1)/2
           else
             filnox = i + jc*(jc-1)/2
           end if
           write(fni,'(i6)') i
           write(fnj,'(i6)') jc
           filno_buf(i,jc)    = filnox + Nfil
           filename_buf(i,jc) = trim(file_root_name)//'_'//'ch_'//trim(adjustl(
fni))//'.'//trim(adjustl(fnj))//'.neq'
         end do
!        end do
      end if
```

```
      if (lERR_OBSx) then
!       write(*,*) ' files for errors: ',imin,ish,jc
        do i = imin,ish
          NF = NF + 1
          if (i.EQ.jc) then
! changed 2012-11-05.
!         if (i.EQ.j) then
            filnox = i*(i+1)/2
          else
            filnox = i + jc*(jc-1)/2
          end if
          write(fni,'(i6)') i
          write(fnj,'(i6)') jc
! CHANGER 2012-03-13.
          filno_buf(i,jc) = filnox + Nfil
          filename_buf(i,jc) = trim(file_root_name)//'_'//'ch_'//trim(adjustl(fn
i))//'.'//trim(adjustl(fnj))//'.err'
        end do
      end if

      if (lERR_COVx) then
        do i = ish+1, imax
!         do j = jmin, jmax   ! rectangularity !
            NF = NF + 1
            if (i.EQ.jc) then
              filnox = i*(i+1)/2
            else
              filnox = i + jc*(jc-1)/2
            end if
            write(fni,'(i6)') i
            write(fnj,'(i6)') jc
            filno_buf(i,jc) = filnox + Nfil
            filename_buf(i,jc) = trim(file_root_name)//'_'//'ch_'//trim(adjustl(
fni))//'.'//trim(adjustl(fnj))//'.cov'
          end do
!        end do
      end if

      if (lSOLx) then  ! sol chunk-files - only 1 expected, place for more : (0,
:)
        NF                 = NF + 1
        filno_buf(0,1)     = Nfil
        filename_buf(0,1) = trim(file_root_name)//'_'//'ch'//'.sol'
      end if

      ifmax = MAX(imax,ifmax)
      jfmax = MAX(jc,jfmax)

      allocate(filno(0:ifmax,jfmax))
      allocate(filename(0:ifmax,jfmax))

      filno    = filno_buf
      filename = filename_buf

      deallocate(filno_buf)
      deallocate(filename_buf)

      if (ltest3) then
        do i = 1,ifmax
          do j = i,jfmax
            write(*,*) i,j,filno(i,j),trim(filename(i,j))
          end do
        end do
      end if

      end subroutine add_files_col

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```



```
! ltri = true if triangular matrix,else rectangular
! tag - file type : neq,err,errcov,...
! ifmin start chunk, ifax end chunk

      subroutine add_files(tag,imin,imax,jmin,jmax)

      implicit none

      integer                                 :: i,j,                  &
                                                 imin,jmin,imax,jmax,  &
                                                 nchoice,filnox
      character*72,dimension(:,:),allocatable :: filename_buf
      integer,     dimension(:,:),allocatable :: filno_buf
      character*10                            :: fni,fnj
      character*16,dimension(7)               :: order
      character*16                            :: tag
!     logical                                 :: lexist

      if (ltest2) write(*,*)' ---- add_files ',TRIM(tag),imax,jmax

      order(1)  = 'neq'
      order(2)  = 'err'
      order(3)  = 'sol'
      order(4)  = 'errcov'
      order(5)  = 'remove'
      order(6)  = 'remove_sol'
      order(7)  = 'move_sol'

      do i = 1,7
        if (trim(order(i)).EQ.trim(tag)) nchoice = i
      end do

      if (file_root_name.NE.file_root_name_old) then
!       write(*,*)' WARNING - new filename entered. Neq files created'
        file_root_name_old = file_root_name
        deallocate(filename)
        deallocate(filno)
        NF  = 0
        ifmax = 0
        jfmax = 0
      end if

      allocate(filno_buf(0:MAX(imax,ifmax),MAX(jmax,jfmax)))
      allocate(filename_buf(0:MAX(imax,ifmax),MAX(jmax,jfmax)))

      if (allocated(filno)) then
        filno_buf(0:ifmax,1:jfmax) = filno(0:ifmax,1:jfmax)
        deallocate(filno)
      end if
      if (allocated(filename)) then
        filename_buf(0:ifmax,1:jfmax) = filename(0:ifmax,1:jfmax)
        deallocate(filename)
      end if

      select case (nchoice)

      case (1,4)    ! neq or errcov chunk-files - both triangular
        do j = jmin, jmax
          do i = imin, j   ! from the triangularity :)
            NF = NF + 1
            if (i.EQ.j) then
              filnox = i*(i+1)/2
            else
              filnox = i + j*(j-1)/2
            end if
            write(fni,'(i6)') i
            write(fnj,'(i6)') j
            filno_buf(i,j)    = filnox + Nfil
```

```
          filename_buf(i,j) = trim(file_root_name)//'_'//'ch_'//&
          trim(adjustl(fni))//'.'//trim(adjustl(fnj))//'.'//&
          trim(adjustl(order(nchoice)))
!            write(*,*)' buf',i,j, filno_buf(i,j),trim(filename_buf(i,j))
        end do
      end do

    case (2)    !  err chunk-files - rectangular
      do i = imin, imax
        do j = jmin, jmax   ! rectangularity !
          NF = NF + 1
          if (i.EQ.j) then
            filnox = i*(i+1)/2
          else
            filnox = i + j*(j-1)/2
          end if
          write(fni,'(i6)') i
          write(fnj,'(i6)') j
          filno_buf(i,j) = filnox + Nfil
          filename_buf(i,j) = trim(file_root_name)//'_'//'ch_'//&
          trim(adjustl(fni))//'.'//trim(adjustl(fnj))//'.'//&
          trim(adjustl(order(nchoice)))
        end do
      end do

    case (3)  ! sol chunk-files - only 1 expected, place for more : (0,:)
      NF                = NF + 1
!       filnox            = ifmax*(ifmax+1)/2 + 1   ! 1 above max No.
      filno_buf(0,1)    = Nfil
      filename_buf(0,1) = trim(file_root_name)//'_'//'ch'//'.'//trim(adjustl(o
rder(nchoice)))

    case default
      write(*, *) ' File-generate command not recognized '
    end select

    ifmax = MAX(imax,ifmax)
    jfmax = MAX(jmax,jfmax)

    allocate(filno(0:ifmax,jfmax))
    allocate(filename(0:ifmax,jfmax))

    filno    = filno_buf
    filename = filename_buf

    deallocate(filno_buf)
    deallocate(filename_buf)

    if (ltest3) then
      do i = 1,ifmax
        do j = i,MAX(ifmax,jfmax)
          write(*,*) i,j,filno(i,j),trim(filename(i,j))
        end do
      end do
    end if

    end subroutine add_files

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    subroutine int2string(N,S,k)
!  N integer, S string, k 'length' of integer
    IMPLICIT NONE
    CHARACTER*10, INTENT(OUT)  :: S
    INTEGER , iNTENT(IN)       :: N
    INTEGER , iNTENT(OUT)      :: k
    INTEGER                    :: x,N1

    N1 = N
```



```
     k  = 0
     S  = '                        '

     do while (N1.GT.0)
       x = N1 - 10*INT(N1/10)
       S(10-k:10-k) = char(x+48)
       k = k+1
       N1 = (N1-x)/10
     end do

     end subroutine int2string

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

     subroutine generate_givens(NNN,A)
     IMPLICIT NONE
     integer                :: i,j
     INTEGER, INTENT (IN) :: NNN
     real*8                 :: d1
     REAL*8, INTENT(out)  :: A(NNN,NNN)

     if (ltest2) write(*,*)' ---- generate_givens ',NNN

     d1 = 1.d0

! Generating a Givens' Matrix: J. Westlake, app. C, page 138
! Only used for testing with reneforsberg routine

     do i = 1, NNN
       do j = 1, NNN
         if (i.LE.j) A(i,j) = i*d1/j*d1
         if (i.GT.j) A(i,j) = j*d1/i*d1
       end do
     end do

     end subroutine generate_givens

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

     subroutine generate_block_givens(NN,A,bi,bj)

     IMPLICIT NONE

     INTEGER                :: i,j,bix,bjx,bi,bj
     INTEGER, INTENT (IN) :: NN
     REAL*8                 :: d1
     REAL*8, INTENT(out)  :: A(NN,NN)

     if (ltest2) write(*,*)' ---- generate_block_givens ',NN,bi,bj
     d1 = 1.0d0

! Generating a Givens' Matrix: J. Westlake, app. C, page 138
!      write(*,*) ' Generating Givens SPD matrix '

     bix = (bi-1) * NN
     bjx = (bj-1) * NN

     if (bi.EQ.bj) then
       do j = 1, NN
         do i = 1, NN
           if (i.LE.j) A(i,j) = d1*(i+bix)/(j+bjx)
           if (i.GT.j) A(i,j) = d1*(j+bjx)/(i+bix)
         end do
       end do
     end if

     if (bi.LT.bj) then
       do j = 1, NN
         do i = 1, NN
```

```
          A(i,j) = d1*(i+bix)/(j+bjx)
        end do
      end do
    end if

    if (bi.GT.bj) then
      do j = 1, NN
        do i = 1, NN
          A(i,j) = d1*(j+bjx)/(i+bix)
        end do
      end do
    end if

    end subroutine generate_block_givens

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine chol(c,n,nsing)
      implicit none
!
!                      c h o l
!
!  subroutine solves positive definite symmetric linear equations
!  using cholesky decomposition. coefficients stored columnwise
!  in c, followed by righthand side. n is number of unknowns.
!  solution is returned as last column in c.
!  'nsing' is number of singularities. it should be zero for a
!  succesful solution.
!
!  rf, nov 85

!  small change making matrix c allocatable and change to implicit none
!  by M. Veicherts, 2006,11-23.(and 12.05)
!
! cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
!     real(KIND=8) :: dimension(:), allocatable :: c
!     integer      :: n,nsing,i,ir,nc,nr,T(8),ic,nc1,np
      integer      :: n,nsing,i,ir,nc,nr,ic,nc1,np
      real(KIND=8) :: c(n+n*(n+1)/2),sum_loc,cc,ci

      if (ltest) write(*,*)' ------chol  ',n

      nsing = 0

      do 50 nr = 1,n+1
        i=nr*(nr-1)/2
        ir=i
        do 40 nc = 1,nr
          if (nc.gt.n) goto 50
          sum_loc = 0.
          ic=nc*(nc-1)/2
          i=i+1
          nc1=nc-1
          do 30 np = 1,nc1
30        sum_loc = sum_loc - c(ir+np)*c(ic+np)
          ci = c(i) + sum_loc
          if (nr.eq.nc) goto 31
          cc = c(ic+nc)
          if (cc.eq.0) then
            write(*,*) '*** Cholesky reduction zero-division'
            cc = 1.0e9
          endif
          c(i) = ci/cc
          goto 40
31        if (nr.gt.n) goto 40
          if (ci.gt.0) goto 32
            nsing = nsing+1
            c(i) = 1.0e9
            goto 40
```



```
32          c(i) = sqrt(ci)
40        continue
50      continue

!c  back substitution

        do 80 nc = n,1,-1
          ir=i
          ic=nc*(nc+1)/2
          cc = c(ic)
          if (cc.eq.0) then
            write(*,*) '*** Cholesky subsitution zero-division'
            cc = 1.0e9
          endif
          c(i) = c(i)/cc
          do 70 np = nc-1,1,-1
            ir=ir-1
            ic=ic-1
            c(ir) = c(ir) - c(i)*c(ic)
70        continue
          i = i-1
80      continue

        end subroutine chol

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        subroutine finalize_cholsol
        IMPLICIT NONE
        integer :: i

        if (ltest3) write(*,*)' ------finalize_cholsol  '

        if (allocated(CH))  deallocate(CH)
        if (allocated(SOL)) deallocate(SOL)

!        do i = 1, NF
!          if (i/=5) close(i) !bso this is not optimal!
!        end do

        end subroutine finalize_cholsol

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! routine to calculate the error estimates

! bi : Relative block number
! ci  : Chunk number
!
        subroutine sumup_err(Nrow,jc)
        implicit none

        integer                               :: i,j,ic,jc,jb,ib,Nrow,NCX, &
                                                 NBX,NX,iabs,jabs
        real*8, allocatable, dimension(:,:) :: SUMUP
        real*8 aij,sij

! Find chunk number (NCX), block number (NBX) and relative col number bj:

        call get_col_num(Nrow,NCX,NBX,NX)
        allocate(SUMUP(NBB,NN))
        SUMUP = 0.d0

         if (ltest3) write(*,*)' from sumup_err,Nrow,jc,NCX,NBX,NX,NBB,NN ',&
         Nrow,jc,NCX,NBX,NX,NBB,NN

        do ic = 1,NCX-1
          call alloc(ic,jc)
          call chunk_read(ic,jc)
```

```
        do ib = 1,NBB
          do jb = 1,NBB
            do i = 1,NN
              do j = 1,NN
                SUMUP(jb,j) = SUMUP(jb,j) + CH(ic,jc)%BL(ib,jb)%A(i,j)**2
                if (ltest2) then
                 jabs=(jc-1)*NBB*NN+(jb-1)*NN+j
                 iabs=(ic-1)*NBB*NN+(ib-1)*NN+j
                 write(*,5551) iabs,jabs,CH(ic,jc)%BL(ib,jb)%A(i,j),&
                 SUMUP(jb,j)
 5551            format(2i5,2d16.7)
                end if
              end do
            end do
          end do
        end do
        call dealloc(ic,jc)
        close(filno(ic,jc))
      end do

! and from the last chunk:

      ic = NCX
      call alloc(ic,jc)
      call chunk_read(ic,jc)

      do ib = 1,NBB
        do jb = 1,NBB
          do i = 1,NN
            do j = 1,NN
              if (ltest3) then
                jabs=(jc-1)*NBB*NN+(jb-1)*NN+j
                iabs=(ic-1)*NBB*NN+(ib-1)*NN+j
              end if
! change 2012-02-11.
              if ((ib.LE.NBX.AND.i.LE.NX).OR.ib.LT.NBX) then
                if (ib.EQ.NBX.AND.i.EQ.NX) then
                  if (ltest3) aij= CH(ic,jc)%BL(ib,jb)%A(i,j)
                  CH(ic,jc)%BL(ib,jb)%A(i,j) = &
                  CH(ic,jc)%BL(ib,jb)%A(i,j) - SUMUP(jb,j)
                  if (ltest3) then
                   sij=SUMUP(jb,j)
                   write(*,5566)NBX,iabs,jabs,aij,sij,&
                   CH(ic,jc)%BL(ib,jb)%A(i,j)
 5566              format(' NBX,ia,ja,a,s,c,',3i4,3d16.7)
                  end if
                else
                  SUMUP(jb,j) = SUMUP(jb,j) + CH(ic,jc)%BL(ib,jb)%A(i,j)**2
                  if (ltest3) write(*,5551)iabs,jabs,&
                  CH(ic,jc)%BL(ib,jb)%A(i,j),SUMUP(jb,j)
                end if
              end if
            end do
          end do
        end do
      end do

      call chunk_write(ic,jc)
      call dealloc(ic,jc)
      close(filno(ic,jc))

      deallocate(SUMUP)

      end subroutine sumup_err

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine factorizeA_err(bi,ci,C)
```



```
      implicit none

      integer                              :: i,j,k
      integer, INTENT(in)                  :: bi,ci
      real*8                               :: elem,eps
      real*8, dimension(NN,NN)             :: B,A
      real*8, dimension(NN,NN), intent(IN) :: C

      if (ltest3) write(*,*)' ------factA_err  ',bi,ci

      elem = d0
      eps  = 1.d-8
      B    = CH(ci,ci)%BL(bi,bi)%A

      if (bi.GT.1) then
        call get_block_subsumA(bi,ci,A)
!       if (ltest3) call block_write(NN,A)
      else
        A = d0
      end if

      do i = 1,NN

! Diagonal element:

        do j = 1, i-1
          elem = elem + B(j,i)**2
        end do
!        if (B(i,i)-elem-C(i,i)-A(i,i).LT.0.d0) then
!          write(*,*)' ERROR - sqrt,i,j,bi,ci,B(i,i),elem,CH,BL',i,j,bi,ci,B(i,i
),elem,C(i,i),A(i,i)
!        end if
        B(i,i) = B(i,i)-elem-C(i,i)-A(i,i)
        elem   = d0

! Side elements:

        do j = i+1, NN

          do k = 1, i-1
            elem = elem + B(k,i)*B(k,j)
          end do
           if (ABS(B(i,i)).LT.eps) write(*,*)' ERROR - divide 0 !!',i,j,k,B(i,i)
          B(i,j) = (B(i,j)-elem-C(i,j)-A(i,j))/B(i,i)
!         B(i,j) = B(i,j)-elem-C(i,j)-A(i,j)
          elem   = d0
        end do
      end do

      CH(ci,ci)%BL(bi,bi)%A = B

      end subroutine factorizeA_err

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! New set of subroutines to handle negative accumulation.
! FactorizeA, FactorizeB, get_subsum, and get_chunk_subsum

! --------------------------------------------------------

! bi,bj : block indices
! ci,cj   : chunk indices
! C       : result from get_chunk_subsum
! A       : result from get_block_subsum

      subroutine factorizeB_err(bi,bj,ci,cj,C)
      implicit none

      integer                              :: i,j,k,iNN,jNN
```

```
      integer, intent(IN)                   :: bi,bj,ci,cj
      real*8                                :: elem
!     real*8                                :: elem,Adiag
      real*8, dimension(NN,NN)              :: B,A,D
      real*8, dimension(NN,NN), intent(IN) :: C

      if (ltest3) write(*,*)' ------factB_err  ',bi,bj,ci,cj
! added 2012-08-10.
       if (ltest3) write(*,98) bi,bj,ci,cj,NCX_err,NBX_err,NR
98    format(' factB_err bi,bj,ci,cj,NCX_err,NBX_err,NR:  ',7I5)

      elem = d0
      A = CH(ci,ci)%BL(bi,bi)%A
      B = CH(ci,cj)%BL(bi,bj)%A
!      if (ci.EQ.4.AND.cj.EQ.5.AND.bi.EQ.10.AND.bj.EQ.1) then
!        write(*,*) ' B_err indices: 4,5,10,1 '
!        call block_write(NN,CH(ci,cj)%BL(bi,bj)%A)
!      end if

      if (bi.GT.1) then
!       write(*,*)' get_block_subsumB_err called, bi,bj,ci,cj ',bi,bj,ci,cj
!       call get_block_subsumB_err(bi,bj,ci,cj,D)
        call get_block_subsumB(bi,bj,ci,cj,D)
      else
        D = d0
      end if

! Only side elements:

! changed 2012.04.11 to ensure that the last element is not divided by diagonal
element

      if (bi.LT.NBX_err) then
        iNN = NN
        jNN = NN
      else
        iNN = NR
        jNN = NN
      end if
!      write(*,*)'factB : iNN,jNN ',iNN,jNN

      do i = 1,iNN
        do j = 1,jNN
          do k = 1, i-1
            elem = elem + B(k,j)*A(k,i)
!            if (elem.GT.10.d0) write(*,*) '.Berr > 10:',k,i,j,elem
          end do
          if (bi.LE.NBX_err.AND.i.LE.iNN) then
200         format(' B_err: ',2I4,5D14.7)
!            if (bj.EQ.NBX_err.AND.j.EQ.NR)
!            write(*,200) i,j,B(i,j),elem,C(i,j),D(i,j),A(i,i)
            B(i,j) = (B(i,j)-elem-C(i,j)-D(i,j))/A(i,i)
!            write(*,*)' B_err0: ',i,elem,B(i,j)
          end if
          elem = d0
        end do
      end do

      CH(ci,cj)%BL(bi,bj)%A = B

!     if (ci.EQ.4.AND.cj.EQ.5.AND.bi.EQ.10.AND.bj.EQ.1) then
!       write(*,*) ' B_err reduced: 4,5,10,1 '
!       call block_write(NN,CH(ci,cj)%BL(bi,bj)%A)
!     end if

      end subroutine factorizeB_err

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```



```
      subroutine get_block_subsumB_err(bi,bj,ci,cj,C)
      implicit none

      integer                                  :: i,j,k,h
      integer, INTENT (IN)                     :: ci,cj,bi,bj
      real*8                                   :: elem
      real*8,dimension(NN,NN),INTENT (out) :: C
      real*8,dimension(NN,NN)                  :: B,A

      if (ltest) write(*,*) ' get_block_subsumB_err ,bi,ci ' ,bi,ci

! Read previous reduced blocks 'above' A and B, and making the column products:
! when calling this routine the chunks are presumably already opened and read!

      C    = d0
      elem = d0

      do h = 1,bi-1
        A = CH(ci,ci)%BL(h,bi)%A
        B = CH(ci,cj)%BL(h,bj)%A
        do j = 1, NN
          do k = 1, NN

! $OMP PARALLEL  ! should only be activated when there are idle procs!
! $OMP DO

            do i = 1, NN
              if (cj.EQ.NC_err.AND.bi.EQ.NBX_err.AND.i.EQ.NR+1) then
                elem = elem + B(i,j)*B(i,k)
              else
                elem = elem + A(i,j)*B(i,k)
              end if
            end do

! $OMP END DO
! $OMP END PARALLEL

            C(j,k) = C(j,k) + elem
            elem = d0
          end do
        end do
      end do

      end subroutine get_block_subsumB_err

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! get subsum from chunks above the B chunk which is to be factorised.

! in this routine OMP may NOT be applied! (interferes with other parallelisation
!)

! bi relative block number
! ic is index of chunk up to ci
! ci,cj chunk indices
! CHX result chunk

      subroutine get_chunk_subsumB_err(ci,cj,CHX)
      implicit none

      integer, intent(in)          :: ci,cj
      integer                      :: i,j,k,hj,bi,bj,ic
      real*8                       :: elem
      real*8, dimension(NN,NN)     :: A,B,C
      type(chunk), intent(inout) :: CHX
      type(chunk)                  :: B_CH

! Read previous reduced blocks 'above' A and B, and making the column product su
ms:
```

```
      C    = d0
      elem = d0

      if (ltest2) write(*,*) ' get_chunk_subsumB ,ci,cj ' ,ci,cj

      do ic = 1,ci-1            ! go through chunks above current (ci)

        call alloc_CHB(B_CH)
        call chunk_read_CHB(ic,cj,B_CH)
!        call chunk_file_close(ic,cj)
        close(filno(ic,cj))

!        write(*,*) ' get_chunk_subsumB efter '

!       NCX_err,NBX_err,NR
!       if (cj.LT.NCX_err) then

        do bj = 1,NBB            ! Number of blocks pr. chunk col-wise
          do hj = 1,NBB          ! Number of blocks pr. chunk col ...
            do bi = 1,NBB        ! step up through the rows of blocks
              A = A_CH(ic)%BL(bi,bj)%A
              B = B_CH%BL(bi,hj)%A
              do j = 1, NN
                do k = 1,NN

! $OMP PARALLEL PRIVATE(i)   !  should only be enabled when there are idle procs
.
! $OMP DO
                  do i = 1, NN
                    elem = elem + A(i,j)*B(i,k)
                  end do
! $OMP END DO
! $OMP FLUSH
! $OMP END PARALLEL

                  C(j,k) = C(j,k) + elem
                  elem = d0
                end do
              end do
            end do
            CHX%BL(bj,hj)%A = CHX%BL(bj,hj)%A + C
            C = d0
          end do
        end do
        call dealloc_CHB(B_CH)
      end do

      end subroutine get_chunk_subsumB_err

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! %%%%%%%%%%%%%  PARAMETER ESTIMATION   %%%%%%%%%%%%%%%%%
! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! add a new set of routines to manage parameter estimation:
! requirements:
! factorizeA_par, factorizeB_par, get_chunk_subsumB_par, get_block_subsumB_par,
get_chunk_subsumA_par, get_block_subsumA_par

      subroutine factorizeA_par(bi,ci,C)
      implicit none

      integer                              :: i,j,k,Nparam_row,bi_abs,ia,ja,ka,N
K
      integer, INTENT(in)                  :: bi,ci
      real*8                               :: elem,eps,bii
      real*8, dimension(NN,NN)             :: B,A
      real*8, dimension(NN,NN), intent(IN) :: C

      if (ltest2) write(*,*)' ------factA_par:bi,ci,Nparam  ',bi,ci,Nparam
```



```
      elem = d0
      eps  = 1.d-8
      bi_abs= (ci-1)*NBB*NN+(bi-1)*NN
      B    = CH(ci,ci)%BL(bi,bi)%A

      if (bi.GT.1) then
        call get_block_subsumA_par(bi,ci,A)
        if (ltest2) call block_write(NN,A)
      else
        A = d0
      end if

! finding the relative number of the start of the parameter section :

      Nparam_row = Nparam - (ci-1)*NBB*NN - (bi-1)*NN
      NK = Nparam-1
      if (ltest2) write(*,*)' ----factA_par:Nparam_row  ',Nparam_row

      do i = 1,NN
        ia=bi_abs+i

! Diagonal element:

        bii=B(i,i)
        do j = 1, i-1
          ja=bi_abs+j
          if (ltest2) write(*,*)' factA_par: j,ja,i-1,Nparam_row ',j,ja,i-1,Npar
am_row
          if (ja.le.NK.and.ia.gt.NK) then
! change 2012-07-25.
!         if (ja.lt.NK.and.ia.gt.NK) then
!         if (ia.LT.NK.and.ja.ge.NK) then
!         if (i.LT.Nparam_row.or.j.LT.Nparam_row) then
!         if (j.LT.Nparam_row) then
! .OR.j.LT.Nparam_rel) then
            elem = elem - B(j,i)**2
            if (ltest2) write(*,301) j,i,ja,ia,NK,B(j,i),elem
 301        format(' factA_par neg: j,i,ja,ia,NK,B(J,i)', 5i3,2f9.4)
          else
            elem = elem + B(j,i)**2
            if (ltest2) write(*,302) j,i,ja,ia,NK,B(j,i),elem
 302        format(' factA_par pos: j,i,ja,ia,NK,B(J,i)', 5i3,2f9.4)
          end if
          if (ltest2) write(*,*)' factA_par: j,i,B(J,i)', j,i,B(j,i),elem
        end do
!       write(*,*)' i,elem ', i,elem
!       if (i.LT.Nparam_row) then
         if (B(i,i)-elem-C(i,i)-A(i,i).LT.0.d0) then
! ADDED 2012-05-24.
          if ((bi_abs+i).lt.N) then
           if (ltest2) write(*,305)i,j,bi,ci,N,B(i,i),elem,C(i,i),A(i,i)
 305       format(' ERROR SQRT factA_par: i,j,bi,ci,N,B(i,i),elem,C,A'&
          ,5i3,4D15.7)
          end if
          B(i,i) = 1.0d0
         else
           if (ia.le.N) then
            if (B(i,i)-elem-C(i,i)-A(i,i).gt.0.0d0) then
             B(i,i) = sqrt(B(i,i)-elem-C(i,i)-A(i,i))
            else
             write(*,*)' WARNING, neg. ',N,ia
             write(*,303)i,ia,bi,bi_abs,bii,b(i,i),elem,c(i,i),a(i,i)
             B(i,i)=1.0d0
            end if
          end if
         end if
!       if (ltest2) write(*,303)i,ia,bi,bi_abs,bii,b(i,i),elem,c(i,i),a(i,i)
 303    format(' factA_par: i,bii,Bii,elem,cii,aii ',i3,5d15.6)
```

```
        elem   = d0

! Side elements:

        do j = i+1, NN
          ja=bi_abs+j

          do k = 1, i-1
           ka=bi_abs+k
!           if (k.LT.Nparam_row) then
!           if (j.LT.Nparam_rel.OR.k.LT.Nparam_rel) then
!           if (ia.ge.NK.and.ka.lt.NK) then
! change 2012-07-28.
!           if (ia.ge.NK.and.ja.ge.NK.and.ka.lt.NK) then
! change again 2012-11-11.
            if (ia.ge.(NK+1).and.ka.lt.(NK+1)) then
              elem = elem - B(k,i)*B(k,j)
              if (ltest2) write(*,*)' A,bi,i,Nparam_row,elem:',bi,k,Nparam_row,e
lem
            else
              if (ltest2) write(*,*)' A,bi,i,Nparam_row,ELEM:',bi,k,Nparam_row,e
lem
              elem = elem + B(k,i)*B(k,j)
            end if
            if (ltest2) write(*,312)i,j,ia,ja,ka,NK,elem,B(k,i),B(k,j)
 312       format(' i,j,ia,ja,ka,NK,elem;Bki,kl ',6i3,3f10.5)
          end do
          if (ABS(B(i,i)).LT.eps.and.bi_abs+i.LE.N) write(*,*)&
           ' ERROR - divide 0 !!',i,j,k,B(i,i),bi_abs,N
          bii=B(i,j)
          if (abs(B(i,i)).lt.eps) B(i,i)=1.0d0
          B(i,j) = (B(i,j)-elem-C(i,j)-A(i,j))/B(i,i)
          if (ltest2) write(*,309)bii,B(i,j),elem,C(i,j),A(i,j),B(i,i)
 309      format(' factA_par/side, i,j,ia,ja,bij,B(i,j),e,C,A,Bii ',4i4,6f10.5)
          elem   = d0
        end do
      end do

      CH(ci,ci)%BL(bi,bi)%A = B

      end subroutine factorizeA_par

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine get_block_subsumA_par(bi,ci,C)
      implicit none

      integer                              :: i,j,k,h,Nparam_row,      &
                                              Nparam_col,NK,ia,ja,ka,habs,babs
      integer, INTENT (IN)                 :: ci,bi
      real*8                               :: elem
      real*8,dimension(NN,NN),INTENT (out) :: C
      real*8,dimension(NN,NN)              :: B,A

      if (ltest2) write(*,*) ' get_block_subsumA_par ,bi,ci ' ,bi,ci

! Read previous reduced blocks 'above' A, and making the column products:
! when calling this routine the block are presumed already opened and read!

      C          = d0
      elem       = d0
      Nparam_col = Nparam - (ci-1)*NBB*NN - (bi-1)*NN
!     Nparam_col = Nparam - (ci-1)*NBB*NN - (h-1)*NN
      NK=Nparam-1
      babs= (ci-1)*NBB*NN+(bi-1)*NN
      if (ltest2) write(*,*)' subsumA_par,bi',bi,babs

!$OMP PARALLEL default (none) &
!$OMP SHARED (bi,babs,ci,NN,NK,CH,C,Nparam_row,Nparam_col,Nparam,NBB) PRIVATE(h,
```



```
habs,A,B,j,k,i,ia,ja,ka) FIRSTPRIVATE(elem)
!$OMP DO reduction (+:C)

      do h = 1,bi-1
        habs= (ci-1)*NBB*NN+(h-1)*NN
        Nparam_row = Nparam - (ci-1)*NBB*NN - (h-1)*NN
        A = CH(ci,ci)%BL(h,bi)%A
        B = A
        do j = 1, NN       ! go through all cols in B
          ja=babs+j
          do k = 1, NN     ! go through all cols in A (because of symmetry only
'Â½ the cols')
            ka=babs+k
            do i = 1, NN   ! go through all elements in row
              ia=habs+i
              if (ja.gt.NK.and.ka.gt.NK.and.ia.le.NK) then
!             if (ja.le.NK.or.ka.le.NK) then
!             if (i.LT.Nparam_row) then
! .Nparam_col.OR.i.LT.Nparam_row) then  OBS j !!
                elem = elem - A(i,j)*B(i,k)
!               write(*,118)bi,i,j,k,ia,ja,ka,NK,Nparam_row,elem
!118            format(' -block_A,bi,i,j,k,ia,ja,ka,NK,Nparam_row,elem:',9i3,f9.
4)
              else
                elem = elem + A(i,j)*B(i,k)
!               write(*,119)bi,i,j,k,ia,ja,ka,NK,Nparam_row,elem
!119            format(' +block_A,bi,i,j,k,ia,ja,ka,NK,Nparam_row,elem:',9i3,f9.
4)
              end if
            end do

! $OMP CRITICAL
            C(j,k) = C(j,k) + elem
! $OMP CRITICAL
            elem = d0

          end do
        end do
      end do

!$OMP END DO
!  $OMP FLUSH
!$OMP END PARALLEL

!      do j = 1, NN       ! and because of the symmetry:
!        do i = j+1, NN
!          C(j,i) = C(i,j)
!        end do
!      end do

      end subroutine get_block_subsumA_par

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! get subsum from chunks above the A chunk which is to be factorised.

! bi relative block number
! ic is index counter of chunk up to ci
! ci,cj chunk indices
! CHX result chunk

      subroutine get_chunk_subsumA_par(ci,CHX)
      implicit none

      integer                  :: i,j,k,hj,Nparam_rel,Nparam_row,&
!     integer                  :: i,j,k,hj,Nstart,Nparam_rel,Nparam_row,&
                                     Nparam_col,hjabs
!     integer                  :: bi,bj,ci,cj,ic,ia,ja,ka,NK,bi_abs,bj_abs
      integer                  :: bi,bj,ci,ic,ia,ja,ka,NK,bi_abs,bj_abs
```

```
      real*8                  :: elem
      real*8,dimension(NN,NN) :: A,B,C
      type(chunk)             :: CHX

! Read previous reduced blocks 'above' A and B, and making the column product su
ms:

      C    = d0
      elem = d0
      NK = Nparam-1
! last row before parameters. Nparam is first parameter row.

      if (ltest2) write(*,*) ' subt: get_chunk_subsumA_par ,ci ' ,ci

      do ic = 1,ci-1            ! go through chunks above current (ci)
        Nparam_rel = Nparam - (ic-1)*NN*NBB
        if (ltest2) write(*,*)' Nparam_rel,NN,NBB ',Nparam_rel,NN,NBB
        if (Nparam_rel.GT.NN*NBB.and.lf) then
!$OMP PARALLEL default (none) &
!$OMP SHARED(CHX,A_CH,ic,ci,NBB,NN,NK) PRIVATE(bj,hj,bi,A,B,j,k,i,ia,ja,ka) FIRS
TPRIVATE(elem,C)
!$OMP DO
!  reduction (+:CHX)

          do bj = 1,NBB            ! go through blocks col-wise
            do hj = 1,NBB          ! go through remaining columns of blocks
              do bi = 1,NBB        ! step through number of rows in block
!              write(*,130)C
!130           format(' C= ',4i3,4f11.6,/,12X,4f11.6)
!              C = d0
               A = A_CH(ic)%BL(bi,bj)%A
               B = A_CH(ic)%BL(bi,hj)%A
               do j = 1,NN
                 ja=(ci-1)*NN*NBB+(bj-1)*NN+j
!                ja=(ic-1)*NN*NBB+(bj-1)*NN+j
                 do k = 1,NN
                    ka=(ci-1)*NN*NBB+(hj-1)*NN+k
!                   ka=(ic-1)*NN*NBB+(hj-1)*NN+k
                  elem = d0
                  do i = 1,NN
                   ia=(ic-1)*NN*NBB+(bi-1)*NN+i
                    if (ja.gt.NK.and.ka.gt.nk.and.ia.le.NK) then
                     elem = elem - A(i,j)*B(i,k)
                     write(*,111)ia,ja,ka,elem
 111                 format(' ChunkP-: ia,ja,ka,elem ',3i3,f9.4)
                    else
                     elem = elem + A(i,j)*B(i,k)
                     write(*,112)ia,ja,ka,i,j,k,ic,bi,bj,hj,elem
 112                 format(' ChunkP+: ia,ja,ka,i,j,k,ic,bi,bj,hj,elem ',10i3,
f9.4)
                    end if
                  end do
                  C(j,k) = C(j,k) + elem
!                elem = d0
                 end do
               end do
              end do

! $OMP CRITICAL
              CHX%BL(bj,hj)%A = CHX%BL(bj,hj)%A + C
! $OMP END CRITICAL

              call block_write(NN,CHX%BL(bj,hj)%A)
              C = d0
            end do
          end do

!$OMP END DO
!$OMP END PARALLEL
```



```
        else   ! here we will encounter negative accumulation
!       write(*,*)' for Nparam_rel.le.NN*NBB ',Nparam_rel
!       write(*,130)ci,ic,0,0,C

!$OMP PARALLEL default (none) &
!$OMP SHARED(CHX,A_CH,ci,ic,NBB,NN,NK,Nparam,Nparam_col,Nparam_row,ltest2) PRIVA
TE(bj,hj,bi,A,B,j,k,i,ia,ja,ka,bi_abs,bj_abs,hjabs) FIRSTPRIVATE(elem,C)
!$OMP DO
!  reduction (+:CHX)

          do bj = 1,NBB            ! go through blocks col-wise
            Nparam_col = Nparam - (bj-1)*NN - (ci-1)*NN*NBB
            bj_abs=(bj-1)*NN+(ic-1)*NN*NBB
            do hj = 1,NBB          ! go through remaining columns of blocks
              hjabs=(bj-1)*NN+(ic-1)*NN*NBB
              do bi = 1,NBB        ! step through number of rows in block
                bi_abs=(bi-1)*NN+(ic-1)*NN*NBB
                Nparam_row = Nparam - (bi-1)*NN - (ic-1)*NN*NBB
                A = A_CH(ic)%BL(bi,bj)%A
                B = A_CH(ic)%BL(bi,hj)%A
                do j = 1,NN
                  ja=(ci-1)*NN*NBB+(bj-1)*NN+j
!                 ja=(ic-1)*NN*NBB+(bj-1)*NN+j
                  do k = 1,NN
                    ka=(ci-1)*NN*NBB+(hj-1)*NN+k
!                   ka=(ic-1)*NN*NBB+(hj-1)*NN+k
                    elem = d0
                    do i = 1,NN
                      ia=(ic-1)*NN*NBB+(bi-1)*NN+i
!                     ia=bi_abs+i
!                     if (i.LT.Nparam_row) then
! .OR.j.LT.Nparam_col.OR.k.LT.Nparam_col) then
!                     if (i.LT.Nparam_row.OR.j.LT.Nparam_col.OR.k.LT.Nparam_col)
 then
                      if (ja.gt.NK.and.ka.gt.nk.and.ia.le.NK) then
 114                    format(' -chunk_A,ic,bi,i,j,k,ia,ja,ka,NK:',9i3,3f10.5)
                        elem = elem - A(i,j)*B(i,k)
                        if (ltest2) write(*,114)ic,bi,i,j,k,ia,ja,ka,NK,elem,A(i
,j),B(i,k)
                      else
 115                    format(' +chunk_A,ic,bi,i,j,k,ia,ja,ka,NK:',9i3,3d14.5)
                        elem = elem + A(i,j)*B(i,k)
                        if (ltest2) write(*,115)ic,bi,i,j,k,ia,ja,ka,NK,elem,A(i
,j),B(i,k)
                      end if
                    end do
                    C(j,k) = C(j,k) + elem
!                   elem = d0
                  end do
                end do
              end do

! $OMP CRITICAL
              CHX%BL(bj,hj)%A = CHX%BL(bj,hj)%A + C
! $OMP END CRITICAL
!             write(*,130)ci,ic,bj,hj,C,CHX%BL(bj,hj)%A

              C = d0
            end do
          end do

!$OMP END DO
!$OMP END PARALLEL

        end if
      end do

       if (ltest2) write(*,*)' block-write 1,1 _ 1,2 _, 2,2 '
```

```
      if (ltest2) call block_write(NN,CHX%BL(1,1)%A)
      if (ltest2) call block_write(NN,CHX%BL(1,2)%A)
      if (ltest2) call block_write(NN,CHX%BL(2,2)%A)

! because of the symmetry: (as in get_block_subsumA ... )
!      do bi = 1,NBB          ! go through blocks row-wise
!        do bj = bi+1,NBB        ! st
!          CHX%BL(bj,bi)%A = CHX%BL(bi,bj)%A
!        end do
!      end do

      end subroutine get_chunk_subsumA_par

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! bi,bj : block indices
! ci,cj   : chunk indices
! C       : result from get_chunk_subsum
! A       : result from get_block_subsum

      subroutine factorizeB_par(bi,bj,ci,cj,C)
      implicit none

      integer                             :: i,j,k,ia,ja,ka,Nparam_row,Nparam_co
l
      integer                             :: bi,bj,ci,cj,bi_abs,bj_abs,NK
      real*8                              :: elem,bij
      real*8, dimension(NN,NN)            :: B,A,D
      real*8, dimension(NN,NN),intent(IN) :: C

      if (ltest2) write(*,*)' ------factB_par  ',bi,bj,ci,cj,allocated(CH(ci,ci)
%BL),allocated(CH(ci,cj)%BL)

      elem = d0
!      write(*,*)' sizes: ' ,size(CH(ci,ci)%BL(bi,bi)%A),size(CH(ci,cj)%BL(bi,bj
)%A)
      A    = CH(ci,ci)%BL(bi,bi)%A
      B    = CH(ci,cj)%BL(bi,bj)%A

      Nparam_row = Nparam - (ci-1)*NBB*NN - (bi-1)*NN
      Nparam_col = Nparam - (cj-1)*NBB*NN - (bj-1)*NN
      bi_abs = (bi-1)*NN+(ci-1)*NBB*NN
      bj_abs = (bj-1)*NN+(cj-1)*NBB*NN
      NK=Nparam-1

      if (bi.GT.1) then
        call get_block_subsumB_par(bi,bj,ci,cj,D)
      else
        D = d0
      end if

! Only side elements:

      do i = 1,NN         !  rows
        ia=i+bi_abs
        do j = 1,NN       !  cols
          ja=j+bj_abs
          do k = 1, i-1   !
             ka=k+bi_abs
!            if (ja.gt.NK.and.ka.gt.nk) then
!           write(*,*)' i,j,k,ia,ja,ka ',i,j,k,ia,ja,ka
            if (ja.gt.NK.and.ia.gt.NK.and.ka.le.NK) then
! change 2012-06-09.
!           if (ja.ge.NK.and.ia.ge.NK.and.ka.lt.NK) then
! 2012-05-30 change.
              elem = elem - B(k,j)*A(k,i)
             if (ltest2) write(*,198)i,j,k,ia,ja,ka,Nparam-1,elem
 198          format(' neg.acc: i,j,k,ia,ja,ka,Nparam-1,ee ',7i3,f9.4)
            else
```



```
 199          format(' pos.acc: ci,cj,i,j,k,ia,ja,ka,Nparam,Nparam_row,Nparam_co
l,elem ',11i3,f9.4)
              elem = elem + B(k,j)*A(k,i)
              if (ltest2) write(*,199)ci,cj,i,j,k,ia,ja,ka,Nparam,Nparam_row,Npa
ram_col,elem
            end if
          end do

!          write(*,200) i,j,B(i,j),elem,C(i,j),D(i,j),A(i,i)
!200       format(' factB: ',2I4,5D14.7)
!          write(*,200) i,j,B(i,j),elem,C(i,j),D(i,j),A(i,i)
!         if (i.LT.Nparam_row.OR.j.LT.Nparam_col) then
           bij=B(i,j)
           B(i,j) = (B(i,j)-elem-C(i,j)-D(i,j))/A(i,i)
           if (ltest2) write(*,210)i,j,ia,ja,bij,B(i,j),elem,C(i,j),D(i,j),A(i,i
)
 210       format(' facB_par: i,j,ia,ja,bij,Bij,e,Cij,Dij,Aii ',4i3,6f9.4)
!         else
!          B(i,j) = (B(i,j)-elem-C(i,j)-D(i,j))
!          write(*,*)' no div. by A, B(i,j) ',B(i,j)
!         end if
          elem   = d0
        end do
      end do

      CH(ci,cj)%BL(bi,bj)%A = B

      end subroutine factorizeB_par

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      subroutine get_block_subsumB_par(bi,bj,ci,cj,C)
      implicit none

      integer                             :: i,j,k,h,Nparam_row,      &
                                             Nparam_col,NK,ia,ja,ka,bjabs,biabs
,bhabs
      integer, INTENT (IN)                :: ci,cj,bi,bj
      real*8                              :: elem
      real*8,dimension(NN,NN),INTENT (out) :: C
      real*8,dimension(NN,NN)             :: B,A

      if (ltest2) write(*,*) ' get_block_subsumB_par ,bi,ci ' ,bi,ci

! Read previous reduced blocks 'above' A and B, and making the column products:
! when calling this routine the chunks are presumably already opened and read!

      C    = d0
      elem = d0

      Nparam_col = Nparam - (cj-1)*NBB*NN - (bj-1)*NN
      NK=Nparam-1
      bjabs=(cj-1)*NBB*NN+(bj-1)*NN
      biabs=(ci-1)*NBB*NN+(bi-1)*NN

      do h = 1,bi-1
        Nparam_row = Nparam - (ci-1)*NBB*NN - (h-1)*NN
        bhabs=(ci-1)*NBB*NN+(h-1)*NN

        A = CH(ci,ci)%BL(h,bi)%A
        B = CH(ci,cj)%BL(h,bj)%A
        do j = 1, NN
          ja=biabs+j
          do k = 1, NN
            ka=bjabs+k

! $OMP PARALLEL  ! should only be activated when there are idle procs!
! $OMP DO
```

```
           do i = 1, NN
             ia=bhabs+i
             if (ka.gt.NK.and.ja.gt.NK.and.ia.le.NK) then
!            if (i.LT.Nparam_row) then
! .OR.j.LT.Nparam_col.OR.k.LT.Nparam_col) then
               elem = elem - A(i,j)*B(i,k)
               if (ltest2) write(*,116)bi,i,ia,ja,ka,Nparam_row,elem
 116           format(' -block_B,bi,i,ia,ja,ka,Nparam_row:',6i3,f9.4)
             else
 117           format(' +block_B,bi,i,ia,ja,ka,Nparam_row:',6i3,f9.4)
               elem = elem + A(i,j)*B(i,k)
               if (ltest2) write(*,117)bi,i,ia,ja,ka,Nparam_row,elem
             end if
           end do

! $OMP END DO
! $OMP END PARALLEL

           C(j,k) = C(j,k) + elem
           elem = d0
         end do
       end do
     end do
     if (ltest2) write(*,118)C
 118  format(' block_B: C ',4f10.5)

     end subroutine get_block_subsumB_par

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

! in this routine OMP may NOT be applied! (interferes with other parallelisation
!)

! bi relative block number
! ic is index of chunk up to ci
! ci,cj chunk indices
! CHX result chunk

     subroutine get_chunk_subsumB_par(ci,cj,CHX)
     implicit none

     integer, intent(in)         :: ci,cj
     integer                     :: i,j,k,ia,ja,ka,hj,bi,bj,ic,Nparam_row,NK,
   &
                                    Nparam_col,Nparam_rel
     real*8                      :: elem
     real*8, dimension(NN,NN)    :: A,B,C
     type(chunk), intent(inout) :: CHX
     type(chunk)                 :: B_CH

! Read previous reduced blocks 'above' A and B, and making the column product su
ms:

     C    = d0
     elem = d0

     if (ltest2) write(*,*) ' get_chunk_subsumB_par,ci,cj ' ,ci,cj

     do ic = 1,ci-1            ! go through chunks above current (ci)
       Nparam_rel = Nparam - (ic-1)*NN*NBB
       NK=Nparam-1
       if (Nparam_rel.GT.NN*NBB.and.lf) then

         call alloc_CHB(B_CH)
         call chunk_read_CHB(ic,cj,B_CH)
         close(filno(ic,cj))
          write(*,*)' in if section 1,ic,Nparam_rel',ic,Nparam_rel

         do bj = 1,NBB           ! Number of blocks pr. chunk col-wise
```



```
           do hj = 1,NBB         ! Number of blocks pr. chunk col ...
             do bi = 1,NBB       ! step up through the rows of blocks
               A = A_CH(ic)%BL(bi,bj)%A
               B = B_CH%BL(bi,hj)%A
               do j = 1, NN
                 ja=(ic-1)*NN*NBB+(bj-1)*NN+j
! change 2012-07-28.
!                ja=(ic-1)*NN*NBB+(bi-1)*NN+j
                 do k = 1, NN
                   ka=(cj-1)*NBB*NN+(hj-1)*NN+k

! $OMP PARALLEL PRIVATE(i)   !  should only be enabled when there are idle procs
.
! $OMP DO
                   do i = 1, NN
                     ia=(ic-1)*NN*NBB+(bi-1)*NN+i
                     write(*,130)ic,bi,i,j,k,ia,ja,ka,Nparam_rel
 130                 format(' chunk1_B-,ic,bi,i,j,k,ia,ja,ka,Nparam_row:',9i3)
                     elem = elem - A(i,j)*B(i,k)
                   end do
! $OMP END DO
! $OMP FLUSH
! $OMP END PARALLEL
                   C(j,k) = C(j,k) + elem
                   elem = d0
                 end do
               end do
             end do
             CHX%BL(bj,hj)%A = CHX%BL(bj,hj)%A + C
             write(*,*)' result A1:,bj,hj ', bj,hj,CHX%BL(bj,hj)%A
             C = d0
           end do
         end do
         call dealloc_CHB(B_CH)
         write(*,*)' in if section22'

       else

         if (ltest2) write(*,*)' in else section, NK ',NK
         call alloc_CHB(B_CH)
         call chunk_read_CHB(ic,cj,B_CH)
         close(filno(ic,cj))

         do bj = 1,NBB           ! Number of blocks pr. chunk col-wise
           Nparam_col = Nparam - (cj-1)*NN*NBB - (bj-1)*NN
           do hj = 1,NBB         ! Number of blocks pr. chunk col ...
             do bi = 1,NBB       ! step up through the rows of blocks
               Nparam_row = Nparam - (ic-1)*NN*NBB - (bi-1)*NN
               A = A_CH(ic)%BL(bi,bj)%A
               B = B_CH%BL(bi,hj)%A
               do j = 1, NN
                 ja=(ci-1)*NN*NBB+(bj-1)*NN+j
                 do k = 1, NN
                   ka=(cj-1)*NBB*NN+(hj-1)*NN+k

! $OMP PARALLEL PRIVATE(i)   !  should only be enabled when there are idle procs
.
! $OMP DO
                   do i = 1, NN
                     ia=(ic-1)*NN*NBB+(bi-1)*NN+i
!                    if (ia.le.NK) then
! change 2012-07-25 and back 07-28.
                     if (ia.le.NK.and.ja.gt.NK.and.ka.gt.NK) then
!                    if (i.LT.Nparam_row) then
!.AND.j.LT.Nparam_col.AND.k.LT.Nparam_col) then
                       if (ltest2) write(*,131)ic,bi,i,j,k,ia,ja,ka,Nparam_row
 131                   format(' chunk_X-,ic,bi,i,j,k,ia,ja,ka,Nparam_row:',9i3)
                       elem = elem - A(i,j)*B(i,k)
                     else
```

```
                        if (ltest2) write(*,132)ic,bi,i,j,k,ia,ja,ka,Nparam_row
 132                    format(' chunk_X+,ic,bi,i,j,k,ia,ja,ka,Nparam_row:',9i3)
                        elem = elem + A(i,j)*B(i,k)
                      end if
                    end do
! $OMP END DO
! $OMP FLUSH
! $OMP END PARALLEL
                    C(j,k) = C(j,k) + elem
                    elem = d0
                  end do
                end do
              end do
              CHX%BL(bj,hj)%A = CHX%BL(bj,hj)%A + C
              C = d0
              if (ltest2) write(*,*)' result A:,bj,hj ', bj,hj,CHX%BL(bj,hj)%A
            end do
          end do
          call dealloc_CHB(B_CH)

        end if
      end do

      end subroutine get_chunk_subsumB_par

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

      end module m_cholsol

! %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! ========================================================================
module m_timing
! ========================================================================
  implicit none
  integer             :: t_size
  integer             :: t_cnt   ! total number of calls to 'timer'
  double precision    :: t_timer ! overhead spent in 'timer' routine
  type times
    character(len=32) :: t_name
    character(len=32) :: t_sub
    double precision  :: t_total
    double precision  :: t_tmp
    integer           :: t_cnt
  end type times
  type (times), dimension(:), allocatable :: program_time

  contains

  subroutine timer(t_name,n_call,sub)
    implicit none
    character(len=*), intent(in)           :: t_name
    character(len=*), intent(in), optional :: sub
    integer,          intent(in)           :: n_call
    integer                               :: tmp_size
    integer                               :: i
    character(len=32)                     :: t_sub
    double precision                      :: t

    t = wtime()

    if ( present(sub) ) then
      t_sub = sub
    else
      t_sub = ''
    end if

    if ( .not.allocated(program_time) ) then
      t_cnt   = 0
      t_timer = 0.0
```



```
      t_size  = 1
      allocate(program_time(t_size))
      program_time(t_size)%t_name  = t_name
      program_time(t_size)%t_sub   = t_sub
      program_time(t_size)%t_total = 0.0
      program_time(t_size)%t_cnt   = 0
    end if

    tmp_size = t_size

    t_find : do i = 1, tmp_size
      if ( t_name == program_time(i)%t_name .and. t_sub == program_time(i)%t_sub
 ) then
        if ( n_call == 0 ) then
          program_time(i)%t_total = program_time(i)%t_total + t - program_time(i
)%t_tmp
          program_time(i)%t_tmp   = t
        else if ( n_call == 1 ) then
          program_time(i)%t_tmp = t
        else if ( n_call == 2 ) then
          program_time(i)%t_total = program_time(i)%t_total + t - program_time(i
)%t_tmp
          program_time(i)%t_cnt = program_time(i)%t_cnt + 1
        else
          write(6,'(a,1i4)') 'call to timer with n_call = ',n_call
          write(6,'(a)')    'allowed values are 0 (elapsed), 1 (begin), and 2 (e
nd)'
        end if
        exit t_find
      end if
      if ( i == tmp_size ) then
        call add_time
        if ( n_call == 0 .or. n_call == 1 ) then
        program_time(t_size)%t_name  = t_name
        program_time(t_size)%t_sub   = t_sub
        program_time(t_size)%t_tmp   = t
        program_time(t_size)%t_total = 0.0
        program_time(t_size)%t_cnt   = 0
        else
          write(6,'(a,1i4)') 'cannot initialize timer with n_call = ',n_call
        end if
      end if
    end do t_find

! -------------------------
! Update timer routine info
! -------------------------

    t_cnt   = t_cnt + 1
    t_timer = t_timer + wtime() - t

! ========================================================================
  end subroutine timer
! ========================================================================

! ========================================================================
  subroutine add_time
! ========================================================================
    implicit none
    type (times), dimension(:), allocatable :: tmp_time
! ========================================================================
! Allocate and set temporary array
! ------------------------------

    allocate(tmp_time(t_size))
    tmp_time = program_time

! --------------------------------
! Extend main array be reallocation
```

```
! ----------------------------------

    deallocate(program_time)
    t_size = t_size + 1
    allocate(program_time(t_size))

! ------------------
! Restore main array
! ------------------

    program_time(1:t_size-1) = tmp_time(:)

! --------------------------
! Deallocate temporary array
! --------------------------

    deallocate(tmp_time)

! ========================================================================
  end subroutine add_time
! ========================================================================

  function etime(t_name) result(t)
    implicit none
    character(len=*), intent(in) :: t_name
    double precision             :: t
    integer                      :: i

    t = -1.0

    t_find : do i = 1, t_size
      if ( t_name == program_time(i)%t_name ) then
        t = wtime() - program_time(i)%t_tmp
        exit t_find
      end if
    end do t_find

  end function etime

  function ttime(t_name) result(t)
    implicit none
    character(len=*), intent(in) :: t_name
    double precision             :: t
    integer                      :: i

    t = -1.0

    t_find : do i = 1, t_size
      if ( t_name == program_time(i)%t_name ) then
        t = wtime() - program_time(i)%t_tmp + program_time(i)%t_total
        exit t_find
      end if
    end do t_find

  end function ttime

! ========================================================================
  subroutine print_times(verbose)
    implicit none
    logical, optional :: verbose
    logical           :: l_verbose
    integer           :: i, j
! ========================================================================
    if ( allocated(program_time) ) then
      write(6,'(a)') '=========================================='
      write(6,'(a)') 'Geocol timing data'
      write(6,'(a)') '=========================================='

      if ( present(verbose) ) then
```



```
        l_verbose = verbose  ! input value
      else
        l_verbose = .false.  ! default value
      endif

      do i = 1, t_size
        if ( program_time(i)%t_name /= 'Total' .and. program_time(i)%t_cnt > 0 )
 then

        if ( program_time(i)%t_sub == '' ) then
          write(6,'(a32,1f10.3)') print_name(program_time(i)%t_name,program_time
(i)%t_cnt,l_verbose),program_time(i)%t_total
          do j = 1, t_size
            if ( program_time(j)%t_sub == program_time(i)%t_name) then
              write(6,'(a32,1f10.3)') ' '//print_name(program_time(j)%t_name,pro
gram_time(j)%t_cnt,l_verbose),program_time(j)%t_total
            end if
          end do
        end if
        end if
      end do
      if (l_verbose) then
        write(6,'(a)') '------------------------------------------'
        write(6,'(a32,1f10.3)') print_name('Timing overhead',t_cnt,l_verbose),t_
timer
      end if
      write(6,'(a)') '=========================================='
      do i = 1, t_size
        if ( program_time(i)%t_name == 'Total' .and. program_time(i)%t_cnt > 0 )
 then
          write(6,'(a,1f10.3)') program_time(i)%t_name,program_time(i)%t_total
          write(6,'(a)') '=========================================='
        end if
      end do
    else
      write(6,'(a)') 'No timers active'
    endif
! ========================================================================
  end subroutine print_times
! ========================================================================

! ========================================================================
! NOTE: this function is not optimal but should be modified to take
!       leap years into account. However, if only used to compute
!       delta times, it will give the correct result.
function wtime() result (time)
! ========================================================================
  implicit none
  double precision      :: time
  integer, dimension(8) :: t
  integer, dimension(12), parameter :: dinm = [31,28,31,30,31,30,31,31,30,31,30,
31] ! days_in_month
! ========================================================================
! Get current date and time
! -------------------------

  call date_and_time(values=t)

! ------------------
! Convert to seconds
! ------------------
  time =(t(1) * 365.0d0   & ! years
       + dinm(t(2))       & ! months
       + t(3))* 86400.0d0 & ! days
       + t(5) * 3600.0d0  & ! hours
       + t(6) * 60.0d0    & ! minutes
       + t(7) * 1.0d0     & ! seconds
       + t(8) * 0.001d0     ! 1/1000 seconds
```

```
! ======================================================================
end function wtime
! ======================================================================

! ======================================================================
function wtime2() result (time)
! ======================================================================
  implicit none
  double precision :: time
  integer :: cnt
  integer :: cnt_rate
  integer :: cnt_max
! ======================================================================
! Get current date and time
! -------------------------

  call system_clock(cnt, cnt_rate, cnt_max)

! ------------------
! Convert to seconds
! ------------------

  time = dble(cnt)/dble(cnt_rate) ! seconds

! ======================================================================
end function wtime2
! ======================================================================

function print_name(t_name,t_cnt,l_cnt) result(p_name)
  implicit none
  character(len=*),  intent(in) :: t_name
  integer,           intent(in) :: t_cnt
  logical,           intent(in) :: l_cnt
  character(len=32)             :: p_name
  character(len=32)             :: c_cnt

  if ( l_cnt ) then
    write(c_cnt,'(1i32)') t_cnt
    c_cnt = ' ('//trim(adjustl(c_cnt))//')'
  else
    c_cnt = ''
  end if

  p_name = trim(t_name)//c_cnt

end function print_name

function get_num_open_files(names) result (num_open_files)
  implicit none
  logical, optional :: names
  integer :: num_open_files
  integer :: i
  integer, parameter :: inum = 10000
  character(len=128) :: filename
  logical :: file_open, print_name, file_named

  num_open_files = 0

  print*,'a'
  if (present(names)) then
    print_name = names
  else
    print_name = .false.
  end if

  do i = 1, inum
    print*,'b',i
    inquire(unit=i,opened=file_open)
    if (file_open) then
```



```
      num_open_files = num_open_files + 1
      if (print_name) then
        inquire(unit=i,named=file_named)
        if (file_named) then
          inquire(unit=i,name=filename)
        else
          filename = 'un-named file'
        end if
        write(6,'(i6,a)'), i,' '//trim(filename)
      end if
    end if
  end do

end function get_num_open_files

! ======================================================================
end module m_timing
! ======================================================================
! ======================================================================
! This module contains routines, functions, and variables related to
! MPI multiprocessing.
! ======================================================================
module m_MPI
! ======================================================================
#ifdef _MPI
! ----------------------------------------------------------------------
  implicit none
! ----------------------------------------------------------------------
  integer          :: MPI_pid
  integer          :: MPI_ierr
  integer          :: MPI_numprocs
  character(len=4) :: MPI_fid
! ----------------------------------------------------------------------
  include 'mpif.h'
! ======================================================================
contains
! ======================================================================


! ======================================================================
! Initialize MPI variables: MPI_pid and MPI_numprocs
! ======================================================================
subroutine MPI_initialize
  implicit none
  logical :: l_ini ! flag for testing status
! ======================================================================
  call MPI_initialized(l_ini, MPI_ierr)
  if ( .not. l_ini ) then
    call MPI_init(MPI_ierr)
    call MPI_comm_rank(MPI_comm_world, MPI_pid, MPI_ierr)
    call MPI_comm_size(MPI_comm_world, MPI_numprocs, MPI_ierr)
    write(mpi_fid,'(1i4.4)') MPI_pid
  end if
! ======================================================================
end subroutine MPI_initialize
! ======================================================================


! ======================================================================
subroutine MPI_stop
! ======================================================================
  call MPI_abort(MPI_comm_world, 0, MPI_ierr)
! ======================================================================
end subroutine MPI_stop
! ======================================================================

! ======================================================================
subroutine MPI_finish
  implicit none
```

```
  logical :: l_fin ! flag for testing status
! ======================================================================
  call MPI_finalized(l_fin, MPI_ierr)
  if ( .not. l_fin ) then
    call MPI_finalize(MPI_ierr)
  end if
! ======================================================================
end subroutine MPI_finish
! ======================================================================

#endif
! ======================================================================
end module m_MPI
! ======================================================================
##HOWTO-USE: do a "make clean", followed by a "make" if you want to be sure of a
 fresh compilation of all modules.
##          "make" (without the "clean") does not save much time in this sparse
 list of modules.
##################
# Default values #
##################

# Target
default: geocol19

# Compiler #
#FC = g95
#FC = gfortran
#FC = pgf90
FC = ifort
#FC = mpif90

##Debug options, enables tracing through error output: default, debug ON
#DEBUG = -O0 -g -check all -debug all -debug-parameters all -traceback -ftrapuv
-fpe-all=0
#DEBUG =
 DEBUG = -g -traceback
#DEBUG = -warn all -O0 -g -check all -debug all -debug-parameters all -traceback
 -ftrapuv -fpe-all=0
#DEBUG = -O0 -g -check all -debug all -debug-parameters all -traceback -ftrapuv
-fpe-all=0
#DEBUG = -g -ftrapuv -fpe-all=0

##Floating point models: default, no floating point options..
FLT   =
#FLT   = -fp-model precise

##Optimization options: "-fast" is autoparallelization, "-O1" & "-O2" are optimi
zation options: default, NO optimization ("-O0")
#FOPT =
#FOPT = -O0
#FOPT = -O1
#FOPT = -O2
#FOPT = -fast
 FOPT = -O3

##various options: default, byterecl and arrays in heap storage
## only ifort ##
OPTS =
#OPTS = -heap-arrays
#OPTS = -assume byterecl -heap-arrays -check bounds -warn unused
 OPTS = -assume byterecl -heap-arrays -check bounds
#OPTS = -assume byterecl -check bounds
#OPTS = -assume byterecl
#OPTS = -assume byterecl -heap-arrays  -warn unused
 OPTS = -assume byterecl -heap-arrays

##OpenMP: default, process OMP directives
#OMP = -fpp
```



```
## ifort settings ##
#OMP = -fpp
#OMP = -fpp -D_MPI
#OMP = -fpp -D_MPI -openmp
#OMP = -openmp-report2
#OMP = -parallel
 OMP = -fpp -openmp
#OMP = -fpp -D_MPI -openmp

## gfortran settings ##
#OMP = -cpp -ffree-line-length-none -fmax-array-constructor=262144
#OMP = -cpp -fopenmp -ffree-line-length-none -fmax-array-constructor=262144
#OMP = -cpp -D_MPI -ffree-line-length-none -fmax-array-constructor=262144
#OMP = -cpp -D_MPI -fopenmp -ffree-line-length-none -fmax-array-constructor=2621
44

##Stack all values of above in variable flags
FFLAGS = $(FOPT) $(OMP) $(OPTS) $(FLT) $(DEBUG)

##This one -- I don't know whether needed ... but probably ok.
.KEEP_STATE:

.SUFFIXES: .f90 .o
.f90.o:
        $(FC) -c $(FFLAGS) $<

GEOCOL19 = m_MPI.o m_timing.o m_input.o m_cholsol.o m_geocol_data.o m_data.o m_p
arams.o geocol19.o
#GEOCOL19 = m_timing.o m_input.o m_cholsol.o m_geocol_data.o m_data.o m_params.o
 geocol19.o

GEOCOL18 = geocol.o

geocol19: $(GEOCOL19)
        $(FC) $(FFLAGS) $(GEOCOL19) -o geocol19

geocol18: $(GEOCOL18)
        $(FC) $(OPTS) $(OMP) $(DEBUG) $(GEOCOL18) -o geocol18

clean:
        rm -f *.o src.tgz *.mod *.lst

src.zip:
        zip -v src.zip Make* *.f90 *.log */

info:
        @echo
        @echo "Selected Makeflags:"
        @echo "  Experiment = $(Experiment)"
        @echo "  HOST     = $(HST)"
        @echo "  FC       = $(FC)"
        @echo "  FOPT     = $(FOPT)"
        @echo "  OPTS     = $(OPTS)"
        @echo "  HW       = $(HW)"
        @echo "  FLANG    = $(FLANG)"
        @echo "  PAR      = $(PAR)"
        @echo "  DBG      = $(DBG)"
        @echo "  FFLAGS   = $(FFLAGS)"
        @echo "  PFLAGS   = $(PFLAGS)"
        @echo

#mpi:

################
# Dependencies #
################

geocol19.o:          m_params.o m_geocol_data.o m_cholsol.o m_data.o m_MPI.o m_t
```

```
iming.o m_input.o
geocol19.o:          m_params.o m_geocol_data.o m_cholsol.o m_data.o m_timing.o
m_input.o
m_geocol_data.o:     m_params.o
im_data.o:           m_params.o
m_input.o:           m_MPI.o
echo /home/cct/dgravsoft/geocol/geocol19
rm runfile
ulimit -s unlimited
./geocol19<<!
t
f f f t f f f f
24               ! here number of processors is input
5  5
neqs10j
f t f f f f f t
t
5 - GRS80
2
4
-2.3 1543.0 36 f  t
-1 2 1.000
/home/cct/cct1/dgravsoft/egm96.edg
t
 -1 2 3 3 4 6 0 55 -1 265000.0  f f f t f f t f f t
/home/cct/dgravsoft/GO_TZZ_36_10a.dat
25
0.015
0.0015
3
t
t f
f f f
f t t f
f
  1 2 3 3 4 6 0 55 -1 265000.0  f f f t f f t f f t
/home/cct/dgravsoft/GO_TZZ_36a.dat
29
0.005
0.0015
-1.0
3
t
f
t t f f
f
0.0 36.0 0.0 36.0 4.0 4.0   11 -1 0.0 t t f
gr.dat
gre.dat
f
f f t f
  1 2 3 3 4 6 0 55 -1 265000.0  f f f t f f t f f t
/home/cct/dgravsoft/GO_TZZ_36a.dat
27
0.005
3
t
f
t t t t
f
6378136.6 3.986004418D14
t
coef_s10
2 2
name
40 40
/home/cct/cct1/dgravsoft/sph_36.co
(2i5,2d19.12)
t
```



```
t
```