The covhsmp and covhs2p software

User's guide

G. Balmino

14 January 2009

GOCE User Toolbox (GUT)

GUT IMPLEMENTATION AND SUPPORTING SCIENTIFIC STUDIES

The covhsmp and covhs2p software - User's guide

Table of contents

The algorithms of covhsmp and covhs2p:
"*Efficient propagation of error covariance matrices of gravitational models. Application to GRACE and GOCE.*"

1. Introduction

The *covhsmp* and *covhs2p* software have been developed several years ago, in preparation of advanced space gravity missions (e.g. Aristoteles, studied between 1986 and 1993) having capabilities of providing high resolution Earth's gravity field models with good error structure. These software were written without documentation at that time, though with lots of comments in the code (in French), and left in a dormant state for many years. Variances only were needed in applications using "old" gravity models, and a simple software, based on a direct algorithm, was used (the size of the problems did not require much sophistication). The advent of the three new satellite gravity mapping missions: CHAMP, GRACE and GOCE, with a much larger number of gravity model parameters and more sophisticated applications (especially in oceanography) asked for reviving those advanced tools. Therefore they were revisited (Balmino, 2008 – paper in annex) to produce the actual software.

In a least squares approach to determine gravity field models from observations (often of various types), the error information lies in the variance-covariance matrix (abbreviated VC matrix in the following) which is the inverse of the normal matrix N, weighted by a variance factor s_0 . It provides what we call *formal* error estimates on the model parameters. The goal is to map the VC matrix information onto various geodetic functions of the gravity field, to derive: (i) errors on these functions (square root of the variance) at some points; (ii) cross-covariances of errors between pairs of points. These points are located on a reference ellipdoid approximating the Earth's surface (actually approximating the geoid) or at constant altitude in some cases. In the problem which we solve with the two software, all points are distributed regularly in latitude and longitude, i.e. lie on a grid; the errors (and covariances) at (between) any point(s) can then be efficiently computed by interpolation of the gridded values.

Our approach is fairly general. The gravity field is a peculiar case of functions approximated by truncated spherical harmonic series of the form:

$$q = \sum_{\ell \le L} \sum_{m \le \ell} f_{\ell m} \left[C_{\ell m} P_{\ell m}(\sin\varphi) \cos m\lambda + S_{\ell m} P_{\ell m}(\sin\varphi) \sin m\lambda \right] \qquad (1)$$

or

$$q = Y^t X \qquad (2)$$

In current applications q can be: geoid height, gravity anomaly or disturbance, or their vertical gradient, equivalent water height (accounting for the loading effect), topography, or any other function subject to this type of representation.

This is written on a surface (e.g. the Earth's surface, a reference ellipsoid) with φ, λ being the geo/planeto-centric latitude and longitude, respectively, and where the coefficients $C_{\ell m}$ and $S_{\ell m}$ have been predetermined from observations of q (or one or several functions of q). In gravitational potential problems, the $C_{\ell 0}$'s may be residual harmonics (when the normal potential of a reference ellipsoid is subtracted). The $C_{\ell m}$ and $S_{\ell m}$ coefficients are usually normalized (Ferrer – geodetic normalization is our choice) and ordered according to a certain numbering scheme, that is $X = \{C_{\ell m} ; S_{\ell m}\}_{\ell,m}$.

In (2) we have $Y = \{ f_{\ell m} P_{\ell m}(\sin\varphi) \cos m\lambda \, ; f_{\ell m} P_{\ell m}(\sin\varphi) \sin m\lambda \}_{\ell,m}$: vector of the corresponding spherical harmonic functions, ordered like the X components; the $P_{\ell m}$'s are the Legendre polynomials (m=0) and functions (m>0) of the geocentric latitude φ, normalized like the $C_{\ell m}$, $S_{\ell m}$; they are replaced by their definite integrals over the latitude limits of a given area when the mean value of q is considered over this area. The $f_{\ell m}$'s are constants or functions of latitude (φ) only; they may incorporate filtering coefficients, or tapering coefficients limiting the harmonic series to some window in the (ℓ,m) domain for specific problems. This will be explained in the next section.

A zone on the working surface (e.g. the Earth) and an equiangular grid covering this zone being defined, our goal is to compute:

(a) the variance of the error on q at each grid point, or its mean value over a grid cell : this is performed by *covhsmp*;

(b) the error-covariances between a given point and all other points in the grid (up to a certain distance); this is done by *covhs2p*.

$\Gamma = s_0^2 \, N^{-1}$ being the covariance matrix of X, equal to the inverse of the normal matrix N, scaled by the unit variance factor s_0, one has:

- for problem (a), at any point P:

$$\sigma^2(q) = Y^t \, \Gamma \, Y \tag{3}$$

- for problem (b):

$$\begin{bmatrix} \sigma^2(q_1) & \mathrm{cov}(q_1,q_2) \\ \mathrm{cov}(q_1,q_2) & \sigma^2(q_2) \end{bmatrix} = \begin{bmatrix} Y_1^t \Gamma Y_1 & Y_1^t \Gamma Y_2 \\ Y_2^t \Gamma Y_1 & Y_2^t \Gamma Y_2 \end{bmatrix} \tag{4}$$

where the subscripts 1, 2 refer to a pair of points P_1 and P_2 ; the diagonal terms can be computed as in problem (a), making (a) a particular case of problem (b). The *covhs2p* software provides variances at each grid node and covariances between each node and neighbouring ones and could suffice, but for many problems the variances only are needed which justified the development of the simpler software *covhsmp*.

An important feature of both software is that the full (square) matrix Γ is required, for sake of efficiency, and that it is stored on disc (allowing the treatment of large problems) – see section 5.1, 5.2 and 5.3.

2. Features common to both software

2.1. Constants, model degree and order, grid definition

- *Constants*:

Both software need, for geodetic functions, the following constants:

GM	:	product of the Newton gravitational constant by the body (Earth) mass
a	:	equatorial semi-major axis of reference, used in the gravity model
f	:	flattening of the reference ellipsoid (used for latitude conversion)

ω : mean angular velocity of the rotating body (or reference ellipsoid)

Units are those of the S.I. system

For other types of function, GM, a and ω are in general not needed (except *a* if one uses a Gauss filter – see below).

- *model degree and order*:

The model coefficients themselves $\{C_{\ell m}\ ;\ S_{\ell m}\}_{\ell,m}$ are never needed, but one has to know the minimum and maximum values of ℓ,m and the ordering scheme.

We first define*:* *lmin*, *lsup* : minimum and maximum degree
mmin, *msup* : minimum and maximum order
with $mmin \leq lmin$, and $msup \leq lsup$.

The harmonic coefficients are ordered as follows (according to *m*) :

- constant coefficient : C_0 (corresponds to $\ell = m = 0$)
- zonal coefficients : $C_{\ell 0}$, for $\ell = lbeg(0)$ to $lend(0)$
- tesseral coefficients : $C_{\ell m}$ and $S_{\ell m}$
  for $m = mmin$ to $msup$ and for each m, $\ell = lbeg(m)$ to $lend(m)$.

Usually $lbeg(m) = m$ and $lend(m) = lsup$, but it is possible to define different values for some *m*; for example it is frequent to have no constant term, no degree one terms, and to start the order one terms at $\ell = 2$.

- *grid definition*:

The grid is limited by:
parallels of latitudes : φ_{min} , φ_{max}
meridians of longitude : λ_{min} , λ_{max} (λ is >0 eastward).

The grid stepsizes are $\delta\varphi$, $\delta\lambda$ in latitude and longitude (respectively); $\delta\varphi$ and $\delta\lambda$ must divide $\varphi_{max} - \varphi_{min}$ and $\lambda_{max} - \lambda_{min}$ respectively.

Then we may have point values ($K_{mp} = 1$) or mean values - over each grid bin ($K_{mp} = 0$). In *covhsmp* mean values are average values (rigorously computed), whereas in *covhs2p* they are point values computed at the center of each cell (choice justified by most applications which use point values indeed). Consequently the grid has $N_\varphi = (\varphi_{max} - \varphi_{min})/ \delta\varphi + K_{pm}$ "lines" (or bands) of latitude and $N_\lambda = (\lambda_{max} - \lambda_{min})/ \delta\lambda + K_{pm}$ meridians (or bands in longitude).

The grid nodes and the values attached to them (a single number in the case of *covhsmp*, a set of numbers – the covariances, in the case of *covhs2p*) are ordered (and values are stored accordingly on disc) in matrix fashion and by decreasing latitudes. The *i*.th row corresponds to one "line" of latitude $\varphi_i = \varphi_{max} - \delta\varphi\ (1-K_{pm})/2 - (i-1)\ \delta\varphi$, and the *j*.th column corresponds to one meridian of longitude $\lambda_j = \lambda_{min} + \delta\lambda\ (1-K_{pm})/2 + (j-1)\ \delta\lambda$, the coordinates being those of the bin center in the case of mean (or pseudo-mean) values.

In the variance case (*covhsmp*), we compute the error (square root of variance) σ_{ij} at each grid node.

In the covariance computation case (*covhs2p*), we call the grid domain *the working zone* [Z], or also (in the comments embedded in the software) "*inner zone*"; that is [Z] = [$\varphi_{N\varphi}$, φ_1] x [λ_1 , $\lambda_{N\lambda}$]. Then we define a moving *window* $W_{HK}(N_{ij})$ around each node (*i, j*) which consists of all points (φ_h , λ_k) such that *i*-H ≤ *h* ≤ *i*+H and *j*-K ≤ *k* ≤ *j*+K; H and K are chosen by the user and characterize the maximum distance at which we compute the covariances (this distance depends on latitude since we work in spherical coordinates and with equiangular bins). The union of [Z] and all W_{HK} 's is a domain called the *envelope* {E[Z]} of the inner zone. {E[Z]} may go beyond a pole or have a λ- extension larger than 2π, of which we have taken care. Therefore, at each node (*i,j*) of the regular grid we compute the tensor components $C_{ij}^{hk} = \mathrm{cov}(q_{ij}, q_{hk})$ for all neighbouring nodes (*h,k*) in the window of (*i,j*) – see fig. 1

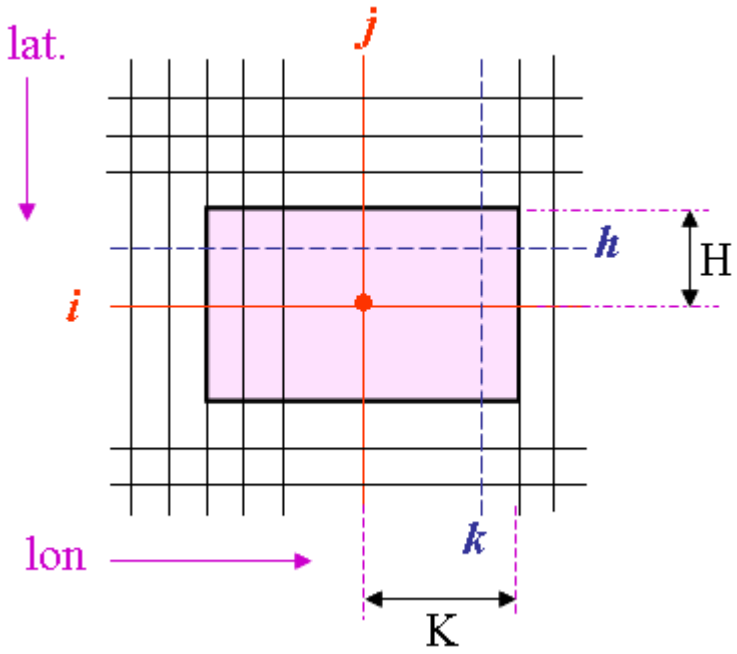

Fig. 1. Geometry of the computation of covariances. Grid nodes can be at corners of equiangular cells or at their center. C_{ij}^{hk} are computed for each (*i,j*) and (*h,k*) such that *i*-*H* ≤ *h* ≤*i*+*H* and *j*-*K* ≤ *k* ≤*j*+*K*, with $C_{ij}^{hk} = C_{hk}^{ij}$. The coloured "spherical" rectangle centered at node (*i,j*) is the window of this node.

2.2. Considered functions: geoid, gravity, etc.

We here give the expression of the $f_{\ell m}$'s in formula (1) apart from the filtering coefficients; for most of the considered functions *q* (especially the geodetic ones), $f_{\ell m}$ only depends on the degree ℓ and we write it as the product of a constant (f_0) and of a g_ℓ term, both depending on function *q*:

- geoid height (in meter): $f_0 = GM / a$; $g_\ell = (a/r)^{\ell+1} / \gamma$
- free-air gravity anomaly Δg (in milligal): $f_0 = 10^5\ GM / a^2$; $g_\ell = (\ell - 1)\ ((a/r)^{\ell+2}$

- gravity disturbance (milligal): $f_0 = 10^5\ GM / a^2$; $g_\ell = (\ell + 1)\ ((a/r)^{\ell+2}$

- normal gravity gradient = $\partial^2 T / \partial r^2$ (Eötvös), where T is the disturbing potential (true potential of body minus potential of the reference dynamic ellipsoid:

$$f_0 = 10^9 \; GM / a^3 \quad ; \quad g_\ell = (\ell + 1)\,(\ell + 2)\;\; ((a/r)^{\ell+3}$$

- normal gradient of Δg (milligal/meter) : $f_0 = -10^5 \; GM / a^3 \quad ; \quad g_\ell = (\ell - 1)\,(\ell + 2)\;\; ((a/r)^{\ell+3}$

- equivalent water height (meter): $f_0 = g /(4\pi\rho_0 G) \quad ; \quad g_\ell = (2\ell + 1) / (1 + k'_\ell)$.

- other functions: f_0 is defined by the user, and $g_\ell = 1$ for all ℓ.

In the above, G, M, a are as previously defined, r is the radius vector and γ the normal gravity at the computation point; k'_ℓ is the load Love number of degree ℓ .

2.3. Computation of Legendre functions

Normalized Legendre functions of degree ℓ and order m (polynomials when $m = 0$) are defined as:

$$P_{lm}(u) = \left[\frac{(2 - \delta_{0m})(2l+1)(l-m)!}{(l+m)!} \right]^{1/2} \frac{(1-u^2)^{m/2}}{2^l l!} \frac{d^{l+m}}{du^{l+m}} \left[(u^2 - 1)^l \right] \qquad (5)$$

where $u = \sin\varphi$. Since the factor $(1\text{-}u^2)^{m/2}$ is $\cos^m\varphi$ and may yield underflows close to the poles, we use recursive formulas on the polynomials (of degree l-m) which are obtained in dropping this factor (applied afterwards). These formulas can be found in Balmino et al. (1990).

The definite integrals $I_{lm} = \int_{\varphi_1}^{\varphi_2} P_{lm}(\sin\varphi)\cos\varphi \; d\varphi$ needed in the case of mean values are computed by an algoritm adapted from Gerstl (1980), which is described in Balmino (1994).

2.4. Filters

Filtering coefficients in the spectral domain can be introduced as in Jekeli (1981). They are isotropic filters which result in multiplying the Legendre functions and polynomials (and their integrals) for each degree ℓ by spectral factors. We have four types of filters, defined in the spatial domain; ψ being the angular distance on the unit sphere, they are:

- Meissl-Pellinen (w_P): it is spatially defined by:
  $w_P(\psi) = 1$ if $\psi \le \psi_0$; $w_P(\psi) = 0$ otherwise; ψ_0 is the size of the averaging cap.

- Hanning filter (w_H): it is such that:
  $w_H(\psi) = [1+\cos(\pi\,\psi/\psi_0)]/2$ if $\psi \le \psi_0$; $w_H(\psi) = 0$ otherwise; ψ_0 is as above.

- Gauss global filter (w_G), defined by:

$$w_G(\psi) = \frac{1}{2\pi} \frac{\alpha}{1 - e^{-2\alpha}} e^{-\alpha(1-\cos\psi)}$$

The parameter α is computed as $\alpha = -\mathrm{Log}(\Omega) / [\, 1 - \cos(\Delta/a)\,]$, where Ω is the value (between 0 and 1) of kernel w_G at distance Δ (in meter) at the surface of the sphere of radius a.

- Gauss cap-limited filter (w_G^*), defined by:

$w_G^*(\psi) = w_G(\psi)$ if $\psi \le \psi_0$; $w_G^*(\psi) = 0$ otherwise.

The corresponding spectral factors are computed by recursive formulas as in Jekeli (ibid.) with some refinement in the case of w_H and w_G^*.

2.5. Input files

The software use so-called "directing files"; they have many identical records but some differences in *covhsmp* and *covhs2p* so they are described in sections 3.1 and 4.1. Both software share exactly two files, one containing the VC matrix, the other one the load Love numbers.

2.5.1. Variance-covariance matrix

We first define a case identifier XX...X (15 characters max., input from keyboard in the PC version). Then the VC matrix, which is on unit *nucov* must have the following name:

'matcov_XX...X'

The matrix must be "full" - square (and in binary, sequential form on nucov). The file has no header preceeding the matrix rows.The ordering (rows-columns) must correspond exactly to the ordering of the unknown vector components X_k .

2.5.2. Love numbers

This file is only required when one computes the errors (or covariances) on the equivalent water height, which corresponds to the perturbation of a mean gravity field model (this is the type of analysis made with the GRACE mission derived models and time variations).

The file name is 'fic_love_load' ; it is on unit nu7, which is formatted ("free" format). It contains the load Love numbers up to degree 250 (sufficient for the current applications). The first and last values are given below.

```
  1  0.00E+00
  2 -3.05E-01
  3 -1.96E-01
  4 -1.34E-01
  5 -1.05E-01
  6 -9.03E-02
  7 -8.21E-02
  8 -7.67E-02
  9 -7.26E-02
 10 -6.92E-02
 ...
 ...
245 -6.64E-03
246 -6.62E-03
247 -6.60E-03
248 -6.58E-03
249 -6.56E-03
250 -6.54E-03
```

3. Using covhsmp

The first thing to do is to set the maximum dimensions which the problem requires, which are given by *three numbers* defined in *module parameter_limites_cov*. This may be done once for all or for each given case - if one wants to optimize the used core on the computer (this implies the re-compilation of the module). The *three* corresponding instructions in the module are:

```
integer , parameter :: lim = 151     ! >= max. degree/order of spherical harmonic model

integer , parameter :: nlon1 = 361  ! >= max. number of longitudes in the grid

integer, parameter  :: nlign = 181   !  = number of lines  in a block
```

The last number (nlign) sets the (exact) size of the block of latitudes lines (or bands if mean values are computed) which are processed all together. It impacts greatly the used core, for the (usually) largest array in the software is *faux*, which is dimensioned (*laux,nlign*), with

laux = [(lim+1)*(lim+2)]/2

3.1. Directing file (input commands); example

We define a case identifier XX...X (15 characters max., input from keyboard in the PC version); this is the same identifier already encountered in naming the VC matrix file.

The directing file is generated by the user. Its name is built with the identifier XX…X, it is 'covhsmp_dir_XX...X.txt' , it is placed on unit nu5 by the software. Here is an example.

```
directing file for covhsmp
GRACE150                (eigen_gl04s) : name given to model = 15 first characters (max.)
meanponc=1          0 : grid of mean values  ;  1 : grid of point values
gm=0.39860044150000e+15,a=0.63781364600000e+07,uapl=0.29825765000000E+03,om=0.72920905111492E-04
lmin=002            min. degree taken into account
lsup=150            max. degree ...
mmin=000            min. order ...
msup=150            max. order ...
m=-99,l_beg=000,l_end=000  for specific orders (m=...) : min. and max. degree (end if m=-99)
s0=+1.48548e+00             variance factor, will multiply the covariance matrix (read in e12.5)
kf=1                function type : 1=n(geoid),2=deltag(FA),3=dg=trr,4=d2T/dr2,5=dFA/dr,6=water eq.,0=other
kse=2               key for type of latitudes (1:geoc. , 2:ellip.)
h=+0.000000000    altitude (m): in effect according to function type (kf=3,4,5) ,read in f12.0
iunit=0             iunit for lat./lon. steps   (0:degree  , 1:minute)
fimin=-90.00,fimax=+90.00,dfi=+10.00,xlmin=-180.00,xlmax=+180.00,dxl=+10.00  (grid limits in deg.)
f0=0.0000000000 factor depending on function type (effective or not) , read in f12.0
kfilter=00,dfilter=300000.00,psi0=5.000,fract0=0.500 filter parameters (no filtering if kfilter=0)
l1=002,l2=150,lstp=00  step by step cumulated errors from deg. l1 to l2, by step lstp (if =0 : l1 to l2)
0           end of file (for PC)
```

Most entries are self-explanatory. Parameters which need clarification are:

- uapl = inverse of flattening.
- m=-99,l_beg=000,l_end=000 : these are the modifications of the limiting degree, per order, of some spherical harmonics. In this example, there is none. If one would start at degree 2 for *m*=1, one would have:

```
m=001,l_beg=002,l_end=150
m=-99,l_beg=000,l_end=000  :  to mark the end of such rec.
```

- f0 : this is applied only if kf = 0 (other function than geodetic); in other cases f0 is computed (and written on output).

- kfilter=00,dfilter=300000.00,psi0=5.000,fract0=0.500 : filter parameters
  kfilter = 0 no filter
  kfilter = 1 : Meissl-Pellinen
  kfilter = 2 : Hanning
  kfilter = 3 : Gauss (global)
  kfilter = 4 : cap-limited Gauss
  dfilter = Δ : distance in meter
  psi0 = ψ_0 : angular distance, in degree
  fract0 = Ω : between 0 and 1

(psi0 is ignored when kfilter = 3, dfilter and fract0 are needed only when kfilter = 3 or 4)

- l1=002,l2=150,lstp=00 : computation of cumulated errors from degree ℓ =l1 to degree ℓ =l2, by step of lstp (always starting at degree l1):
  . if one grid only (total error), take lstp = 0, or define l1=l2 and lstp=1.
  . if not, one computes grids of cumulated errors (from l1, up to l2 [at most], and by step of lstp)
  N.B. this implies reductions of the covariance matrix (by s/p redhsmp)

3.2. Output files

3.2.1. Output controls ("prints"); example

The file 'covhsmp_out_XX...X_.txt' : on unit nu6, contains everything written on output.
We give here an example of geoid gridding with the *GRACE eigen_gl04s* model, complete to degree and order 150; the grid is worldwide, with steps of 10° in latitude and longitude.
Computation is done in one block.

```
                    point synthesis of variances

      calculation : m=   0    ldeb=   2   lfin= 150
      calculation : m=   1    ldeb=   2   lfin= 150
      calculation : m=   2    ldeb=   2   lfin= 150
      calculation : m=   3    ldeb=   3   lfin= 150
      calculation : m=   4    ldeb=   4   lfin= 150
      calculation : m=   5    ldeb=   5   lfin= 150
        ...
      calculation : m= 145    ldeb= 145   lfin= 150
      calculation : m= 146    ldeb= 146   lfin= 150
      calculation : m= 147    ldeb= 147   lfin= 150
      calculation : m= 148    ldeb= 148   lfin= 150
      calculation : m= 149    ldeb= 149   lfin= 150
      calculation : m= 150    ldeb= 150   lfin= 150

      total number of parameters= 22797

      sigma0=   1.485478156765300

      kse= 2 ----> latitudes are geodetic (ellipsoidal)

       1    n(geoid)          altitude=      0.000 metres

      zone :     latitude    : min=  -90.0000     max=  90.0000    step=  10.00000
                 longitude   : min=-180.0000     max= 180.0000    step=  10.00000

reference ellipsoid defined by              gm= 398600.44150000E+09
                                             a = 6378.1364600000E+03
                                             f = 1./      298.25765
                                            om= 7.2920905111492E-0
```

geometrical parameters

b = 6356.7518066306E+03
ge= 521.85359180424E+03
e = 8.1819132449895E-02
ep= 8.2094378965670E-02

dynamical parameters

u0= 72.105843211372E+03
pm= 3.4497624792635E-03
ga= 9.7803272087811E+00 (gravity at equator)
gb= 9.8321865256238E+00 (gravity at poles)

gravity as function of fi

(f2= 5.2788883556304E-03 (coef. of $\sin^2$ fi)
(
(f4= 2.3295306130934E-05 (coef. of $\sin^4$ fi)

j2= 1.0826375455676E-03
j4=-2.3836835418358E-06

the generated grid, on scratch file (tape 2) has 19 records of 37 words

verification: kf= 1 f0= 6.249481239541E+07

kfilter= 0 dfilter= 300000. m psi0= 5. deg. fract0= 0.500

date/time : 2009 1 8 60 16 23 17 78 ==> tbegin_sec = 58997.078

No modification of the degrees, therefore matrix is identical

n0. block = 0 date/time : 2009 1 8 60 16 23 17 125 ==> t0_sec = 58997

control : block 1 lat. = 90.000
control : block 1 lat. = 80.000
control : block 1 lat. = 70.000
•••
control : block 1 lat. = -80.000
control : block 1 lat. = -90.000

n0. block = 1 date/time : 2009 1 8 60 16 23 47 ==> t1_sec = 59027
nrec_nucov = 1000 time = 15.096 sec

n0. block = 1 date/time : 2009 1 8 60 16 24 02 ==> t1_sec = 59042
nrec_nucov = 2000 time = 14.148 sec

n0. block = 1 date/time : 2009 1 8 60 16 24 17 ==> t1_sec = 59057
nrec_nucov = 3000 time = 14.235 sec

n0. block = 1 date/time 2009 1 8 60 16 24 31 ==> t1_sec = 59071
nrec_nucov = 4000 time = 13.627 sec

•••

n0. block = 1 date/time : 2009 1 8 60 16 28 18 ==> t1_sec = 59298
nrec_nucov = 21000 time = 13.539 sec

n0. block = 1 date/time : 2009 1 8 60 16 28 32 ==> t1_sec = 59312
nrec_nucov = 22000 time = 13.336 sec

degree min. : 2 degree max. : 150 r.m.s. of errors : 6.486457E-02

grid extreme values : min= 0.024041 max= 0.071373 metre

date/time : 2009 1 8 60 16 28 32 ==> t (lp)_sec = 59312.484

lp = 150 time = 315.406 sec

3.2.2. Generated grid(s)

The file named 'grid_err_XX...X' , with the identifier XX…XX previously defined, contains the final grid(s). It is a formatted file, on unit nu4, with the following structure:

. when errors are cumulated, from degree l1 (fixed) to l2 (i.e. from l1 with stepsize lstp), one finds ng grids on nu4 (each one has a header) with ng = [(l2-l1)/lstp] + 1 ; the k.th grid contains the errors cumulated from degree l1 up to degree l1 + (k-1) * lstp.

. when lstp = 0 (or if l1=l2 and lstp=1), there is only one grid on nu4 (with its header).

. each header record specifies the min. and the max. degree (this is also the case if one has one grid only) written with the format ('l1=',i3,',l2=',i3).

. the grid is written in matrix form: following the header one has N_φ records, each of N_λ words, where N_φ and N_λ have been defined in section 2.1.

4. Using covhs2p

There are lots of similarities with the use of *covhsmp*. They are repeated here so as not to force the user to read this section and the previous one at the same time.

The first thing to do is to set the maximum dimensions which the problem requires, which are given by:

- *three numbers* defined in *module parameter_limites_cov*. This may be done once for all or for each given case - if one wants to optimize the used core on the computer (this implies the re-compilation of the module). The *three* corresponding instructions in the module are:

```
integer , parameter :: lim = 201     ! >= max. degree/order of spherical harmonic model

integer , parameter :: nlon1 = 361   ! >= max. number of longitudes in the grid

integer, parameter  :: nlign = 181   ! = number of lines  in a block
```

- *three other numbers* defined in *module parameter_covhs2p*. They are: nmaxcov, lathmax, lonkmax, and they are defined in the following instructions in the module:

```
integer, parameter :: nmaxcov = 181     ! max. dim. of tables of isotropic,
                                        ! N-S and E-W covariances

integer, parameter :: lathmax = 20      ! Hmax = max. value of  H

integer, parameter :: lonkmax = 20      ! Kmax = max. value of  K
```

(Hmax, Kmax define the maximum size of the window centered at each grid node)

The last number (nlign) in the first module and the second number in the second module (Hmax) set the (exact) size of the block of latitudes lines (or bands if pseudo-mean values are computed) which are processed all together. It impacts greatly the used core, for the (usually) largest array in the software is *faux*, which is dimensioned (*laux, nlign* + 2 *Hmax*), with

laux = [(lim+1)*(lim+2)]/2

ex : lim = 200 (GOCE), therefore *laux* = 20301
nlign=181 (the whole sphere, with a step of 1 deg. in latitude)
lathmax = 20 : half-"height" of windows (in latitude)
=> *nlign* + 2 *Hmax* = 221 => *faux* has dimension. 4 486 521 (!)

4.1. Directing file (input commands); example

We define a case identifier XX...X (15 characters max., input from keyboard in the PC version); this is the same identifier already encountered in naming the VC matrix file.

The directing file is generated by the user. Its name is built with the identifier XX…X, it is 'covhs2p_dir_XX...X.txt' , it is placed on unit nu5 by the software. Here is an example.

```
directing file for covhs2p
GOCE200                 (simulated recovery) : name given to model = 15 first characters (max.)
typgrid=0            0 : direct access     1 : sequential  (binary files)
meanponc=1           0 : grid of ( pseudo) mean values  ;  1 : grid of point values
gm=0.39860044150000e+15,a=0.63781364600000e+07,uapl=0.29825765000000E+03,om=0.72920905111492E-04
lmin=002             min. degree taken into account
lsup=200             max. degree ...
mmin=000             min. order ...
msup=200             max. order ...
m=-99,l_beg=000,l_end=000  for specific orders (m=...) : min. and max. degree (end if m=-99)
s0=+1.00000e+00   variance factor, will multiply the covariance matrix (read in e12.5)
kf=1                 function type : 1=n(geoid),2=deltag(FA),3=dg=trr,4=d2T/dr2,5=dFA/dr,6=water eq.,0=other
kse=2                key for type of latitudes (1:geoc. , 2:ellip.)
h=+0.000000000    altitude (m): in effect according to function type (if kf=3, 4 or 5), read in f12.0
iunit=0               iunit for lat./lon. steps   (0:degree  , 1:minute)
fimin=+20.00,fimax=+80.00,dfi=+01.00,xlmin=-060.00,xlmax=+030.00,dxl=+01.00 (limits of inner zone Z in deg.)
H=lath=020,K=lonk=020   window size : half-height, halh-width (in number of grid points)
f0=1.0000000000     factor depending on function type (effective or not) , read in f12.0
kfilter=00,dfilter=300000.00,psi0=5.000,fract0=0.500 filter parameters (no filtering if kfilter=0)
l1=001,l2=200        computation for degree between l1 and l2 (eventually: reduction of cov. matrix)
dpsi=01.000000       stepsize (in degree) for tables of covariance functions) , read in f9.0
kverif=0             key for verification by "brute force" at a few points (if cov. matrix fits in core), 0: no
interp_ex=1          key for testing the interpolation procedure (if DA file), 0: no; 1:yes, for pair below
zi_lat=+40.50,zi_lon=+000.50,v_lat=+48.50,v_lon=+003.50  pair of points (1 in Z ; 2 in W [1]) for interp.
0            end of file   (for PC)
```

Most entries are self-explanatory. Parameters which need clarification are:

- typgrid: this keyword decides on the type (and structure) of the output grid file.
0 : direct access ; 1 : sequential. The file is always binary.
This was implemented at the request of some users. Note that the choice "sequential" forbids the use of software *extra_cov2p* and *inter_cov2p* (see section 5).
- uapl = inverse of flattening.
- m=-99,l_beg=000,l_end=000 : these are the modifications of the limiting degree, per order, of some spherical harmonics. In this example, there is none. If one would start at degree 2 for *m*=1, one would have:
m=001,l_beg=002,l_end=150
m=-99,l_beg=000,l_end=000 : to mark the end of such rec.

- f0 : this is applied only if kf = 0 (other function than geodetic); in other cases f0 is computed (and written on output).
- kfilter=00,dfilter=300000.00,psi0=5.000,fract0=0.500 : filter parameters
    kfilter = 0 no filter
    kfilter = 1 : Meissl-Pellinen
    kfilter = 2 : Hanning
    kfilter = 3 : Gauss (global)
    kfilter = 4 : cap-limited Gauss
    dfilter = Δ : distance in meter
    psi0 = ψ_0 : angular distance, in degree
    fract0 = Ω : between 0 and 1

(psi0 is ignored when kfilter = 3, dfilter and fract0 are needed only when kfilter = 3 or 4)
- l1=002,l2=200 : computation of covariances from degree ℓ =l1 to degree ℓ =l2; this is for studying limited (truncated) cases, i.e. with l1 ≥ lmin and/or l2 < lsup (which implies a reduction of the covariance matrix - by s/p redhsmp)
- kverif: this is a keyword to decide or not on whether we make a verification by a "brute force" algorithm at a few nodes.

    This verification is made by the s/p *verif_varhs2p*, which we have especially developped, at 81 (9*9) couples of points (mostly useful for testing the software during the installation phase); it is limited to cases where the covariance matrix can be put in core (cf. value of *ncmax*, in parameter in s/p *verif_varhs2p*, to be modified as the user wants – according to the available memory). At 9 nodes (*i,j*) of the inner zone, of latitude $\varphi_I - (i\text{-}1)\,\delta\varphi$ and longitude $\lambda_I + (j\text{-}1)\,\delta\lambda$, one (re)computes the covariances C_{ij}^{hk} between (*i,j*) and (*h,k*), for 9 values of (*h,k*).

    The selected test nodes are: $i = 1, N_\varphi/2, N_\varphi$, and $j = 1, N_\lambda/2, N_\lambda$, and the associated points are $h = i\text{-H}, i, i\text{+H}$, and $k = j\text{-K}, j, j\text{+K}$.

    This computation is performed by a direct algorithm which evaluates $Y_{hk}\,\Gamma\,Y_{ij}$ with no trick at all. Results are compared to those obtained by s/p *varhs2p*.

    For this verification, the VC matrix is read only once, and it is stored in array *cov* (*ncmax,ncmax*)... but only if the total number of harmonics in the problem is less or equal to *ncmax* (declared in 'parameter' as written above). This limits the possibility of verification to relatively "small" cases (this statement depends on the computer available memory) - to the benefice of efficiency and simplicity of the strategy.

- interp_ex: keyword for testing the interpolator (0 or 1). Such a procedure has been developped for many applications; it works when typgrid = 0 (grid file in direct access). One pair of point (z,w) is chosen for the test: z must be in [Z] and w in the "window" of z, i.e. at angular distances less than H $\delta\varphi$ and K $\delta\lambda$, respectively in latitude and longitude. Coordinates of z and w are read in the following record (as examplified).

4.2. Output files

4.2.1. Output controls ("prints"); example

The file 'covhs2p_out_XX...X_.txt' : on unit nu6, contains everything written on output. We give an example of geoid covariances computation. It corresponds to a GOCE mission simulation performed for one measuring phase of 6 months, at a mean altitude of 265 km. The Earth's gravity field is recovered up to degree and order 200, which makes the full Γ matrix occupy 13 Gb on disk. The considered zone [Z] extends from 20°N to 80°N and from 60°W

to 30°E. With a step size of one degree in latitude and longitude and with 40° x 40° windows (H=K=20), the computation (done in one block) took a little less than 5000 seconds on a standard PC (the same referred to in the article of the annex), which is a gain of ~50% on previous tests (due to the use of different compiler options).

```
                point synthesis of covariances between two points

                typgrid =  0

      calculation : m=   0    lbeg=   2   lend= 200
      calculation : m=   1    lbeg=   2   lend= 200
      calculation : m=   2    lbeg=   2   lend= 200
      calculation : m=   3    lbeg=   3   lend= 200
        . . .
      calculation : m= 196    lbeg= 196   lend= 200
      calculation : m= 197    lbeg= 197   lend= 200
      calculation : m= 198    lbeg= 198   lend= 200
      calculation : m= 199    lbeg= 199   lend= 200
      calculation : m= 200    lbeg= 200   lend= 200

      total number of parameters= 40397

      sigma0=   1.00000000000000

      kse= 2 ----> latitudes are geodetic (ellipsoidal)


reference ellipsoid defined by          gm= 398600.44150000E+09
                                          a = 6378.1364600000E+03
                                          f = 1./     298.25765
                                         om= 7.2920905111492E-0

 geometrical parameters                   b = 6356.7518066306E+03
                                         ge= 521.85359180424E+03
                                          e = 8.1819132449895E-02
                                         ep= 8.2094378965670E-02

 dynamical parameters                    u0= 72.105843211372E+03
                                         pm= 3.4497624792635E-03
                                         ga= 9.7803272087811E+00    (gravity at equator)
                                         gb= 9.8321865256238E+00    (gravity at poles)

                                 (   f2= 5.2788883556304E-03    (coef. of sin^2  fi)
 gravity as function of fi       (
                                 (   f4= 2.3295306130934E-05    (coef. of sin^4  fi)

                                       j2= 1.0826375455676E-03
                                       j4=-2.3836835418358E-06

       1    n(geoid)          altitude=      0.000 metres

      inner zone :     latitude    :  min=  20.0000    max=  80.0000    step=   1.00000
                       longitude :  min= -60.0000    max=  30.0000    step=   1.00000

      the inner zone/grid has   61 lines in latitude and    91 columns in longitude   ==>    5551 nodes


      window size :     H=lath=  20     K=lonk=  20

      the generated file has    5551 records of   1683 words each

      verification:     kf=  1     f0= 6.249481239541E+07

      kfilter= 0     dfilter=  300000. m     psi0=   5. deg.     fract0=     0.500
```

dpsi = 1.000000 deg. : stepsize of tables of covariance functions (iso,n-s,e-w)

date/time : 20090113 122002.359 +0100 2009 1 13 60 12 20 2 359 ==> tbegin_sec = 44402.359

No modification of the degrees, therefore matrix is identical

Call to varhs2p

n0. block = 0 date/time : 2009 1 13 60 12 20 2 375 ==> t0_sec = 44402.375

n0. block = 1 date/time : 2009 1 13 60 12 22 5 78 ==> t1_sec = 44525.078
nrec_nucov = 1000 delta(time) = 122.703 sec

n0. block = 1 date/time : 2009 1 13 60 12 24 10 859 ==> t1_sec = 44650.859
nrec_nucov = 2000 delta(time) = 125.781 sec

...

n0. block = 1 date/time : 2009 1 13 60 13 40 16 625 ==> t1_sec = 49216.625
nrec_nucov = 39000 delta(time) = 124.250 sec

n0. block = 1 date/time : 2009 1 13 60 13 42 19 593 ==> t1_sec = 49339.593
nrec_nucov = 40000 delta(time) = 122.968 sec

Out of varhs2p

degree min. : 2 degree max. : 200 calculation finished...

Verification of covariance functions : iso , n-s , e-w

n	psi (deg)	fcov_iso (nval)	fcov_n-s (nval)	fcov_e-w (nval)
0	0.000000	2.124340E-04 (10738)	2.124340E-04 (5642)	2.124340E-04 (10283)
1	1.000000	1.233236E-04 (75985)	1.235043E-04 (11193)	1.206238E-04 (22750)
2	2.000000	1.426108E-04 (145236)	1.417969E-04 (11284)	1.477557E-04 (22750)
3	3.000000	1.151392E-04 (225498)	1.060590E-04 (11284)	1.213302E-04 (22022)
4	4.000000	1.188274E-04 (288197)	1.050238E-04 (11375)	1.402988E-04 (21294)
5	5.000000	1.016775E-04 (342069)	7.553444E-05 (11375)	1.382674E-04 (18018)
6	6.000000	9.648963E-05 (388206)	6.378022E-05 (11466)	1.626426E-04 (16562)
7	7.000000	8.045237E-05 (430430)	3.762685E-05 (11466)	1.543081E-04 (14560)
8	8.000000	8.276579E-05 (464919)	2.958263E-05 (11557)	1.639852E-04 (12922)
9	9.000000	5.522578E-05 (490581)	1.887106E-07 (11557)	1.570991E-04 (11466)
10	10.000000	4.369802E-05 (514241)	-1.575716E-05 (11557)	1.580669E-04 (10192)
11	11.000000	2.349728E-05 (527527)	-3.623165E-05 (11375)	1.555643E-04 (9464)
12	12.000000	-1.626885E-06 (537810)	-5.691606E-05 (11284)	1.475191E-04 (8008)
13	13.000000	-1.222237E-05 (541996)	-6.496271E-05 (11102)	1.465701E-04 (6916)
14	14.000000	-3.966501E-05 (548093)	-8.324447E-05 (11011)	1.308330E-04 (6006)
15	15.000000	-4.968117E-05 (545181)	-8.586595E-05 (10829)	1.266021E-04 (4914)
16	16.000000	-7.000760E-05 (538538)	-9.426018E-05 (10738)	1.096386E-04 (4186)
17	17.000000	-7.496658E-05 (530712)	-9.024622E-05 (10556)	1.041302E-04 (3094)
18	18.000000	-8.436056E-05 (511875)	-8.684648E-05 (10465)	8.656117E-05 (1638)
19	19.000000	-9.878611E-05 (492947)	-8.636886E-05 (10283)	8.149719E-05 (546)
20	20.000000	-9.759828E-05 (471744)	-7.434528E-05 (10192)	0.000000E+00 (0)
21	21.000000	-9.781373E-05 (286286)	0.000000E+00 (0)	0.000000E+00 (0)
22	22.000000	-9.270572E-05 (173628)	0.000000E+00 (0)	0.000000E+00 (0)
23	23.000000	-8.812517E-05 (109382)	0.000000E+00 (0)	0.000000E+00 (0)
24	24.000000	-7.486374E-05 (70980)	0.000000E+00 (0)	0.000000E+00 (0)
25	25.000000	-6.624610E-05 (38948)	0.000000E+00 (0)	0.000000E+00 (0)
26	26.000000	-4.872753E-05 (20202)	0.000000E+00 (0)	0.000000E+00 (0)
27	27.000000	-3.738262E-05 (7644)	0.000000E+00 (0)	0.000000E+00 (0)
28	28.000000	-2.602923E-05 (1638)	0.000000E+00 (0)	0.000000E+00 (0)

****** No verification by direct algo.

****** Execute one example of interpolation : use of s/p interp_cov2p

i,i1,j,j1 : 41 40 62 61
fii,fii1,xlj,xlj1 : 40.0000000000000 41.0000000000000 1.00000000000000 0.000000000000000

```
h,h1,k,k1 :  33  32  65  64
fih,fih1,xlk,xlk1 :  48.0000000000000  49.0000000000000  4.00000000000000  3.00000000000000
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  1  1  40  61  32  64
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  1  2  40  61  32  65
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  1  3  40  61  33  65
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  1  4  40  61  33  64
                    p,Ch1k1,Ch1k,Chk,Chk1 :   1  2.186820504639600E-005  1.587030203982503E-005
6.964959623725848E-005  6.647943517087066E-005  : 1-2-3-4
               ----------------p, c(p) :   1  4.346688462358754E-005
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  2  1  40  62  32  64
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  2  2  40  62  32  65
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  2  3  40  62  33  65
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  2  4  40  62  33  64
                    p,Ch1k1,Ch1k,Chk,Chk1 :   2  2.717104864196947E-005  2.093685783013844E-005
6.751527202253276E-005  6.813129318337236E-005  : 1-2-3-4
               ----------------p, c(p) :   2  4.593861791950326E-005
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  3  1  41  62  32  64
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  3  2  41  62  32  65
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  3  3  41  62  33  65
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  3  4  41  62  33  64
                    p,Ch1k1,Ch1k,Chk,Chk1 :   3 -1.594919958752995E-005 -1.668430568163937E-005
2.139664318964785E-005  2.880847094829536E-005  : 1-2-3-4
               ----------------p, c(p) :   3  4.392902217193472E-006
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  4  1  41  61  32  64
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  4  2  41  61  32  65
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  4  3  41  61  33  65
 *-*-* lath, lonk, p,q,i0,j0,h0,k0 :  20  20  4  4  41  61  33  64
                    p,Ch1k1,Ch1k,Chk,Chk1 :   4 -1.738959056494679E-005 -1.449720655042875E-005
1.659430253978638E-005  2.251611624894857E-005  : 1-2-3-4
               ----------------p, c(p) :   4  1.805905418339851E-006
 Covariance between point (in inner zone) : lat =   40.500  lon =    0.500    , and point : lat =   48.500  lon =    3.500   (inside
window of 1.st point)

       valcov (interpolated) =  2.390107754E-05        ier =   0

 date/time : 20090113 134313.921 +0100      2009   1  13    60  13  43  13  921    ==>  tend_sec =  49393.921

       time for reduc. + comput. + verif. + interp. =  4991.562 sec
```

4.2.2. Generated grid and covariance table files

(a) *the grid file of covariances* :

The generated grid file contains one record per node ; its type and structure depend on the keyword *typgrid* (cf. directing file):

- if typgrid = 1, the file is binary sequential (no format), the s/p does not make any 'rewind' at the beginning (==> header possible prior to call).
  Its name is 'covhs2p_cov_S_XX...X' and it is on unit nu4.
  One has one record per node (i,j) of the inner zone, in increasing order for i (= decreasing latitudes, $i = 1,2,\ldots N_\varphi$) and, for each i, index j is increasing (= increasing longitude) from 1 to N_λ; the record contains the components $C(i,j,h,k) = C_{ij}^{hk}$ for h increasing from i-H to i+H and, for each h, k is increasing from j-K to j+K . Thus, there are $N_\varphi * N_\lambda$ records (following, eventually, the header records), and we have (2*H+1)*(2*K+1) words per record.
  The $C(i,j,h,k)$'s for given i and j are in record $(i-1)* N_\lambda + j$ (following eventual header), and the (h,k) component is the word of rank $(2*K+1)*[h-(i-H)]+k-(j-K)+1$.

- if typgrid = 0, the file is in direct access, without format, and with no header. It is this type of file which is proper to window extraction and interpolation by our procedures.
  Its name is 'covhs2p_cov_DA_XX...X' and it is on unit nu2.
  One has one record per node (*i,j*) of the inner zone in increasing order for the latitude index *i* (*i*=1,2,... N_φ) - i.e. decreasing latitude and, for each *i*, *j* is increasing (=increasing longitude) from 1 to N_λ.
  Each record contains two integers at the beginning (*i* and *j*), then the C(*i,j,h,k*) = C_{ij}^{hk} for *h* increasing from *i*-H to *i*+H and, for each *h*, *k* increasing from *j*-K to *j*+K.
  Thus, there are N_φ * N_λ records and 2+(2*H+1)*(2*K+1) words per record.
  The C(*i,j,h,k*)'s for given *i* and *j* are in record (*i*-1)* N_λ +*j*, and the (*h,k*) component is the word of rank 2+(2*K+1)*[*h*-(*i*-H)]+*k*-(*j*-K)+1.

  (b) *the file of covariance tables* :

  The file named 'tab_fcov_XX...X' , which is on unit nu9, contains the three tables of isotropic, N-S et E-W covariance functions. It is a sequential (free-)formatted file.

  Each record contains:

  n, psi, fcov_iso(n), nbcov_iso(n), fcov_ns(n), nbcov_ns(n) , fcov_ew(n), nbcov_ew(n)

  with (maxcov having been defined in *module parameter_covhs2p* , and dpsi in the directing file):

  . fcov_iso (_ns) (_ew) : vectors (0:nmaxcov), contain the function values, discretized. fcov_...(n) corresponds to a distance psi = ψ such that (n-1)/2*dpsi ≤ psi ≤ (n+1)/2*dpsi.
  . nbcov_iso (_ns) (_ew) : vectors (0:nmaxcov) : number of valeurs of C(*i,j,h,k*) which contribute to fcov_...(n)
  . ncov_iso(_ns) (_ew) : highest rank of the last non zero component.

5. The utility software

5.1. Fullmat_form_bin2

This software reads a general square matrix C, for instance a covariance matrix (in full form), formatted, split into Nfic files, - usually generated on "mainframe" and in a Unix environment.
The "Fullmat_form_bin2" s/w is usually run in a different environment (e.g. Windows) - This is the very reason to have it !
It reads the different files, and re-writes the C matrix on a single binary file suited e.g. for the usage by the *covhsmp* and *covhs2p* s/w (or similar ones), especially when they are run on a PC or in a non Unix environment. This is why this transformation is necessary (since the binary files are incompatible).
Parameters describing the case and necessary for running the s/w are input via a "Directing file", named 'Fullmat_form_bin2_dir.txt'.

In our case, the C matrix comes from "real" calculations (modelling from observations) or from a simulation of the recovery of a global gravitationnal model expanded in spherical harmonics, e.g. in the framework of a space mission such as GOCE.
The matrix is solely for the harmonic coefficients of the model, from degree ldmin to degree ldmax (read in input) and the model is supposed to be complete, i.e. the number of parameters (= order of the C matrix) has to be N = (ldmax+1)**2 - ldmin**2 (if it is not the case, it's easy to modify it in the s/w). This means that all other parameters have been eliminated (reduced). Harmonic coefficients must also be ordered in the order which is used by the subsequent s/w (*covhsmp*, *covhs2p*, ...)

The matrix blocks are on files (F1), F(2), ... in this order. That is each (Fk), k = 1, 2, ...Nfic, corresponds to a block C(k) from the *sequential splitting* of the C matrix. Each file (Fk) bears the name: namefic_in // 'k' , where k has one or two characters (left justified), i.e. Nfich < 100. namefic_in is read in input (and has 20 characters max.). The program searches the files bearing the names ***trim*** (namefic_in) // 'k'
None of the (Fk) files has headers.
The Nfic-1 first files all have the same number of records which correspond to Nlfic rows (or "lines") of the matrix (Nlfic : read in input).
The last one has exactly N - (Nfic-1) * Nlfic records; this number is computed.

Each record = one full row of C, i.e. N real numbers (type SINGLE according to the definitions in module f90_kind).
Read-in format of (Fk) : '(N(formread))',
formread is defined in input (10 char. max.), e.g.: es20.13
Then the *exact format is built* by the program.

C is copied on a file: tape33, binary, of name 'namefic_out', defined in input (20 charact. max). This file has no header and can be used as such in the afore mentionned s/w.

N.B. 1. It is possible to test the program with (at most) the two first files (Fk) and over the first rows only (according to the *nlwr* parameter) of each file. It suffices to define the keyword Ktest to 1 in input...
Author : G. Balmino (2008)
Version : PC - Windows XP - Absoft Pro Fortran V.9
2. ATTENTION : this version uses two vectors (a, b) for reading and writing each line of the matrix, due to the limitation of the implicit do loop counter on some compilers: M= 2**15-1 . If needed the procedure needs to be modified (i.e. by using three, four,... vectors) if N > 2*M .

The directing file for the GOCE200 case is copied below. Entries are self-explanatory following the definitions given above.

```
Directing file for Fullmat_form_bin (and versions 1, 2) ******
8           Nfic  :  number of input files containing the matrix
matcov_form_         namefic_in  (20 char. max)
5000        Nlfic : number of lines (rec.) (except the last one - size will be computed)
2           ldmin  (min. degree of spherical harmonic model)
200         ldmax  (max. degree of spherical harmonic model)
e16.9          formread : format of real numbers on input files (10 char. max)
25          nlwr  : number of lines(rows) (of each input files) printed for control
10          nmwr  : number of columns printed for control
1000        ideltawr : global control of progressing exec' every ideltawr rows
```

```
matcov_GOCE200              namefic_out : name of output file (20 char. max)
0           Ktest  : for executing s/w on a small part of the input file(s) 0 : no
00000       end of this directing file  (on PC) ******
```

5.2. Ltrimat_form_bin2

This is similar to Fullmat_form_bin2, but adapted to symmetric matrices of which the lower triangle is given.

It reads a lower triangular covariance matrix C formatted, split into Nfic files, - usually generated on "mainframe" and in a Unix environment.
The "Ltrimat_form_bin2" s/w is usually run in a different environment (e.g. Windows) - this is the very reason to have it.
It reads the different files, and re-writes the C matrix on a single binary file suited for the usage by the *covhsmp_tri*, *covhsd_tri* s/w (or similar ones), especially when they are run on a PC or in a non Unix environment. This is why this transformation is necessary (since the binary files are incompatible).
The output file may also be used by the compcarmat s/w which transforms the triangular matrix into a full square one, e.g. for subsequent use by the *covhsmp* and *covhs2p* s/w.
Parameters describing the case and necessary for running the s/w are input via a "Directing file", named 'Ltrimat_form_bin2_dir.txt' .
In the present case, the C matrix comes from "real" calculations (modelling from observations) or from a simulation of the recovery of a global gravitationnal model expanded in spherical harmonics, e.g. in the framework of a space mission such as GOCE.
The matrix is solely for the harmonic coefficients of the model, from degree ldmin to degree ldmax (read in input) and the model is supposed to be complete, i.e. the number of parameters (= order of the C matrix) has to be N = (ldmax+1)**2 - ldmin**2 (if it is not the case, it's easy to modify it in the s/w). This means that all other parameters have been eliminated (reduced).
Harmonic coefficients must also be ordered in the order which is used by the subsequent s/w (*covhsmp_tri*, *covhsmp*, *covhs2p*, ...) .

The matrix blocks are on files (F1), F(2), ... in this order. That is each (Fk), k = 1, 2, ...Nfic, corresponds to a block C(k) from the sequential splitting of the C matrix.
Each file (Fk) bears the name: namefic_in // 'k' , where k has one or two characters (left justified), i.e. Nfich < 100.
namefic_in is read in input (and has 20 characters max.). The program searches the files bearing the names ***trim*** (namefic_in) // 'k'
None of the (Fk) files has headers.
The Nfic-1 first files all have the same number of records which correspond to Nlfic rows (or "lines") of the matrix (Nlfic : read in input).
The last one has exactly N - (Nfic-1) * Nlfic records; this number is computed.

Each record = one truncated row of C, i.e. the i.th row contains i real numbers (type SINGLE according to module f90_kind) : C(i,j) j = 1 to i.
Read-in format of (Fk) : '(i(formread))', i being between 1 and N.
formread is defined in input (10 char. max.), e.g.: es20.13
Then the *exact format is built* by the program.

C is copied on a file: tape33, binary, of name 'namefic_out', defined in input (20 charact. max).
This file has no header and can be used as such in the afore mentionned s/w.

N.B. (1) It is possible to test the program with (at most) the two first files (Fk) and over the first rows only (according to the *nlwr* parameter) of each file. It suffices to define t the keyword Ktest to 1 in input...
Author : G. Balmino (2008)
Version : PC - Windows XP - Absoft Pro Fortran V.9

(2) ATTENTION : This version uses up to two vectors (a, b) for reading and writing each line of the matrix, due to the limitation of the implicit do loop counter on some compilers: M= 2**15-1. If needed the procedure needs to be modified (i.e. by using three, four,... vectors) if N > 2*M.

A directing file applicable to the GOCE200 case is copied below. Entries are self-explanatory following the description above.

```
Directing file for Ltrimat_form_bin (and versions 1, 2) ******
8           Nfic  :  number of input files containing the triangular matrix
matcov_form_         namefic_in  (20 char. max)
5000        Nlfic : number of lines (rec.) (except the last one - size will be computed)
2           ldmin  (min. degree of spherical harmonic model)
200         ldmax  (max. degree of spherical harmonic model)
e16.9           formread : format of real numbers on input files (10 char. max)
25          nlwr  : number of lines(rows) (of each input files) printed for control
10          nmwr  : max. number of columns printed for control
1000        ideltawr : global control of progressing exec' every ideltawr rows
matcovt_GOCE200              namefic_out : name of output file (20 char. max)
1           Ktest  : for executing s/w on a small part of the input file(s) 0 : no
00000       end of this directing file  (for PC) ******
```

5.3. compcarmat

The VC matrix may be given by one of its triangles, we assume it is the lower one. In order to use the *covhsmp* and *covhs2p*s/w one needs to complement the triangular matrix into a square matrix. This is the case of the GOCE-VC matrices provided by the ESA/EGGC H.P.F. (High level Processing Facility). This transformation is not trivial when the matrices do not fit in core. It can be performed by the *subroutine compcar_matriang*, which may be called by the program *compcarmat*.

We first present the subroutine.

SUBROUTINE compcar_matriang (n, b ,m, numtri, numcar, x, ndimx, ier)

The two matrices: T (lower triangular), A (square) are on disc, in sequential binary form, and are written in row increasing order.

- "in" variables:
  n : order of matrix
  b : working vector
  m : dimension of vector b (the largest possible, and with m ≥ 2*n)
  numtri : unit number of file containing the triangular part, T, of A.
  If h is the header number of records (see down below) the (n+h).th record contains n words.

numcar : unit number of file containing the square matrix A
x : auxiliary (working) vector
ndimx : dimension of x (≥ n)

N.B. vectors b and x could be omitted in the calling sequence and declared as arrays internal to this s/p. We did it for verification purposes...

- "out" variable :
ier : error code . 0 = O.K., = -1 if m is too small.
- s/p: this subroutine needs : header_mat : provided by the user.

ATTENTION : files numtri and numcar are opened by the user prior to the call. If they have headers, these are managed (in reading on numtri, in writing on numcar...) by the s/p header_mat which calling sequence should be as follows:

call header_mat(numtri,numcar,kle_rw)
numtri : unit number of first file
numcar : unit number of second file
kle_rw = 0 for initialization (usually the case when called by main prog.)
= 1 for reading an header on numtri (make it work if no header !)
= 2 for writing an header on numcar,
= 3 for reading an header on numtri, and to copy it on numcar
= 4 for reading a header on numcar
other variables may be transfered via the *modules* :
- parameter_dim_mat_tri_sq (see below)
- common_tr_header_mat (user defined)

The header_mat s/p is always called, even if there is no header at all.
After a call to this s/p the file(s) is (are) positionned so as to read/write the first record of matrix A.

Program *compcarmat* uses the s/p *compcar_matriang* . We provide a version which generates its own test matrix and then transforms it with *compcar_matriang* .
It is very simple to use:

- first the user must properly define, in *module parameter_dim_mat_tri_sq*, the following integers: nmax : maximum order of matrix
mmax: maximum size of working array b(:), used in *compcar_matriang.*
maxrec_header: maximum number of header records.

- then the user must define the case by constructing the "Directing file", named 'compcarmat_dir.txt'

Here is an example corresponding to the test case generated by the provided test program:

```
Directing file for prog_mat_tri_sq (compcarmat)
95                  order of matrix
250          size of working array b (largest possible)
1                   k-header : 0= no header, 1= header
end_header     key-word (alf) , 10 characters : 1.st word of last rec. of header
0            end of file (for PC)
```

5.4. extra_cov2p

This program is usually run after the execution of *covhs2p* and generation of a 4-D grid of covariances. It extracts windows of size (2H+ 1)x(2K+1) around nodes of (or arbitrary points in) the inner zone [Z], for instance for local studies, graphic representation, etc.

The input grid file is on unit nugrid, in direct access. However it may have been generated by another software under the conditions that the context described in the following is identical and that the structure of the file is exactly the same.

Having defined a case identifier nommod = 'XX...X' (15 characters max., input from keyboard in the PC version) – which is the same identifier already encountered in naming the VC matrix file and in defining the case, the name of file nugrid is:

'covhs2p_cov_DA_' // ***trim***(nommod)

(a) Context:

We repeat some of the basic facts and definitions which are used in *covhs2p*.

Pairs of points belong to a 4-dimension domain consisting of :

- a "inner" zone, noted [Z], defined by latitudes (λ) and longitudes (φ) with min/max values as follows :
  - fimax to fimin = fimax - (N_φ -1)*δφ ,
  - xlmin to xlmax = xlmin + (N_λ -1)*δλ ;

  this zone is covered by a grid, consisting of N_φ * N_λ "nodes" z(i,j), equidistant in latitude (step = δφ) and in longitude (step = δλ), (unit of angles : degree).
  ATTENTION: here fimin, fimax, xlmin, xlmax are the coordinates of the grid limits when we deal with point values (at the corner of the grid equi-angular cells); they are the coordinates of the limiting cell centers at the NW, NE, SE, SW of the grid in the case of (pseudo) mean values.

- windows W[z(i,j)] centered on each node z(i,j) of the inner zone; the size of each window is fixed and defined by H = lath ≤ lathmax, K = lonk ≤ lonkmax , and each one consists in (2*H+1)*(2*K+1) points equidistant in latitude (step δφ) and in longitude (step δλ) - like the nodes of [Z]; points in the window of z(i,j) are said to be in its "vicinity" and are noted v(z(i,j)), or simply v if there is no ambiguity.

N.B. : lathmax and lonkmax are defined in *module parameter_covhs2p*.

The domain of the sphere (or of R**2) consisting in [Z] and of the coverage by the ensemble {W[z]} of all windows is called the "envelope of [Z]" and noted { E[Z] }, or simply {E}. It is also a grid, having the same steps as [Z], which nodes are noted e(m,n); over the part common to [Z] and {E}, the nodes e(-,-) are obviously identical to the nodes z(-,-).

At each node z(i,j) of [Z], of latitude fimax-(i-1)* δφ and longitude xlmin+(j-1)* δλ, we have computed covariance tensor components :

C(i,j,h,k) = Cov [z(i,j),v(h,k)]

between z(i,j) and points in its vicinity v(h,k), for h=i-H to i+H, and k=j-K to j+K.

N.B. Unit = [*unit of the values of studied function*]**2 *since we deal with covariances.*

The structure of file nugrid is the following :

- no header (i.e. no record before the grid records).
- there is one record per node (i,j) of the inner zone, with i increasing (i=1,2,... N_φ) and for each i, j increases too (from 1 to N_λ); i.e. latitude decreases with i and longitude increases with j; the record has two integers at the beginning: i and j, followed by the values C(i,j,h,k) with h increasing from i-H to i+H and, for each h, k increases from j-K to j+K.

So, there are N_φ* N_λ records and 2+(2*H+1)*(2*K+1) words/record.
The covariance C(i,j,h,k) for i and j fixed is in record number (i-1)*nlo+j, and this is the word of rank 2+(2*K+1)*[h-(i-H)]+k-(j-K)+1.

(b) Directing file:

Its name is 'extra_cov2p_dir.txt' ; it is located on unit nu5 and it contains the directing parameters. Here is an example corresponding to a GRACE150 case (or the GOCE200 simulated case). Entries are self-explanatory following the description above.

```
directing file for extra_cov2p
iunit=0         iunit   (0:degree  , 1 :minutes) for the lat/long stepsizes of the grid
fimin=+20.00,fimax=+80.00,dfi=+01.00,xlmin=-060.00,xlmax=+030.00,dxl=+01.00 grid [Z] limits (deg.)
H=lath=020,K=lonk=020   window parameters (half-height and half-width in grid steps)
zi_lat=+75.50,zi_lon=+000.50      center of window
zi_lat=+45.00,zi_lon=-050.00      center of window
zi_lat=99999.                  END  (... when lat > 100.)
```

(c) Output file:

It is on unit nu6. Its name is 'extra_cov2p_output.txt' . Here is an example of extraction of two windows from a GRACE150 case, run with the same parameters as in the directing file above.

```
   Extraction of covariance windows from a 4-D grid pre-generated by covhs2p  (in direct access)

      inner zone :      latitude  : min=  20.0000     max=  80.0000     step=   1.00000
                       longitude : min= -60.0000     max=  30.0000     step=   1.00000

      the inner zone has   61 lines (rows) in latitude and    91 columns in longitude   ==>    5551 nodes

      size of each window :     H=lath=  20     K=lonk=  20

      ===> on input D.A. file :   5551 records of   1683 words/rec.

Extraction : window number    1  centered on point of the inner zone : lat =   75.500  lon =    0.500
      closest node : i =   6   j=  62    lat =   75.000  lon =    1.000

Extraction : window number    2  centered on point of the inner zone : lat =   45.000  lon =  -50.000
      closest node : i =  36   j=  11    lat =   45.000  lon =  -50.000

      We have extracted    2  windows         END of program
```

(c) Generated file of windows

Its name is : 'gwindows_cov2p_' // ***trim***(nommod) ; it is on unit nu9, which is sequential and in "free" format. It contains the grids of covariance values corresponding to the extracted windows, one after the other. These grids all have the same size, and their structure is the following:

- each one starts by a record containing (all angles are in degree) :
  - . the window number (in the order of extraction),
  - . the coordinates (latitude, longitude) of the central node (as input),
  - . the latitudes min. and max., the longitudes min. et max. of the window,
  - . the steps $\delta\varphi$, $\delta\lambda$,
  - . the number of records, equal to (2*H+1), which follow (for this window), and the number of words, equal to (2*K+1), per record.

    N.B. H et K are constants, however these parameters are repeated for each grid in order to facilitate their use.

- we then have 2*H+1 records, one per latitude (in decreasing order), each one having 2*K+1 words and containing the covariances for the longitudes (in increasing order).

5.5. inter_cov2p

This is a subroutine to interpolate a grid of cross-variances of a function (or its errors) at two points : z, v. Its use is examplified at the end of *covhs2p*.

(a) Context:

We redescribe below much of it, which is more comfortable for the user.

The functional is f(lat, lon) given on a sphere: lat = latitude φ, lon = longitude λ. We may also consider that [lat,lon] is replaced by [Y,X], any set of parameters in a certain domain of R**2 , (X = longitude, Y = latitude).

The grid has been generated by s/w *covhs2p* (and its main s/p *varhs2p*), and is on unit nugrid, in direct access.

The pair of points : (z,v) for which we want to compute the covariance belongs to a 4-dimension domain consisting of :

- a "inner" zone, noted [Z], defined by latitudes (Y) and longitudes (X) with min/max values as follows :
  - . fimax to fimin = fimax - (nfi-1) *dfi ,
  - . xlmin to xlmax = xlmin+(nlo-1) *dxl ;

  this zone is covered by a grid, consisting of nfi*nlo "nodes" z(i,j), equidistant in latitude/Y (step = dfi) and in longitude/X (step = dxl), (unit of angles : degree - or units proper to X and Y).
  ATTENTION: here fimin, fimax, xlmin, xlmax are the coordinates of the grid limits when we deal with point values (at the corner of the grid equi-angular cells); they are the coordinates of the limiting cell centers at the NW, NE, SE, SW of the grid in the case of (pseudo) mean values.

- windows W[z(i,j)] centered on each node z(i,j) of the inner zone; the size of each window is fixed and defined by H = lath ≤ lathmax, K = lonk ≤ lonkmax , and each one consists in

(2*H+1)*(2*K+1) points equidistant in latitude (dfi) and in longitude (dxl) - like the nodes of [Z]; points in the window of z(i,j) are said to be in its "vicinity" and are noted v(z(i,j)), or simply v if there is no ambiguity.

N.B. : lathmax and lonkmax are defined in *module parameter_covhs2p.*

The domain of the sphere (or of R**2) consisting in [Z]} and of the coverage by the ensemble {W[z]} of all windows is called the "envelope of [Z]" and noted { E[Z] }, or simply {E}. It is also a grid, having the same steps as [Z], which nodes are noted e(m,n); over the part common to [Z] and {E}, the nodes e(-,-) are obviously identical to the nodes z(-,-).

As a consequence of these definitions and assumptions :

. z must be in [Z]

. v must be in {E} and in W[z]

At each node z(i,j) of [Z], of latitude fimax-(i-1)* $\delta\varphi$ and longitude xlmin+(j-1)* $\delta\lambda$, we have computed covariance tensor components :

C(i,j,h,k) = Cov [z(i,j),v(h,k)]

between z(i,j) and points in its vicinity v(h,k), for h=i-H to i+H, and k=j-K to j+K.

N.B. Unit = [*unit of the values of studied function*]**2 *since we deal with covariances.*

The structure of file nugrid is the following :

- no header (i.e. no record before the grid records).
- there is one record per node (i,j) of the inner zone, with i increasing (i=1,2,... N_φ) and for each i, j increases too (from 1 to N_λ); i.e. latitude decreases with i and longitude increases with j; the record has two integers at the beginning: i and j, followed by the values C(i,j,h,k) with h increasing from i-H to i+H and, for each h, k increases from j-K to j+K.

So, there are N_φ* N_λ records and 2+(2*H+1)*(2*K+1) words/record.
The covariance C(i,j,h,k) for i and j fixed is in record number (i-1)*nlo+j,
and this is the word of rank 2+(2*K+1)*[h-(i-H)]+k-(j-K)+1.

(b) What the s/p does:

One computes valcov = Cov (z,v) = Cov([z_lat, z_lon], [v_lat, v_lon]), with :

. z_lat, z_lon : coordinates (lat/Y, lon/X) of a point z inside the inner zone - [Z] stricto senso (not enlarged, but including the boundary),

. v_lat, v_lon : cooordinates (lat/Y, lon/X) of a neighbouring point, v, i.e. in the window of point (z_lat, z_lon), i.e. such that :

$$\text{abs}[v\_lat - z\_lat] \leq H * dfi$$

$$\text{abs}[v\_lon - z\_lon] \leq K * dxl .$$

N.B. the window W(z) of an arbitrary point z is therefore defined like the window of a grid node z(i,j).

ier : this is an error code returned by the s/p; it may be:

= 0 : OK
= -1 : the provided file is not in direct access – i.e. typgrid is not equal to 0 :
interpolation can not be made with *inter_cov2p*
= +1 : point [z_lat , z_lon] is not in [Z]
= +2 : point [v_lat , v_lon] is not in W(z)
= +3 : case corresponding to ie r= +1 and ier = +2.
= +4 : (very) peculiar case : all points E(q) are outside the windows W[Z(p)]
- see below, in (c), for the definition of Z(:) and E(:).

(c) Method:

See figure 2.

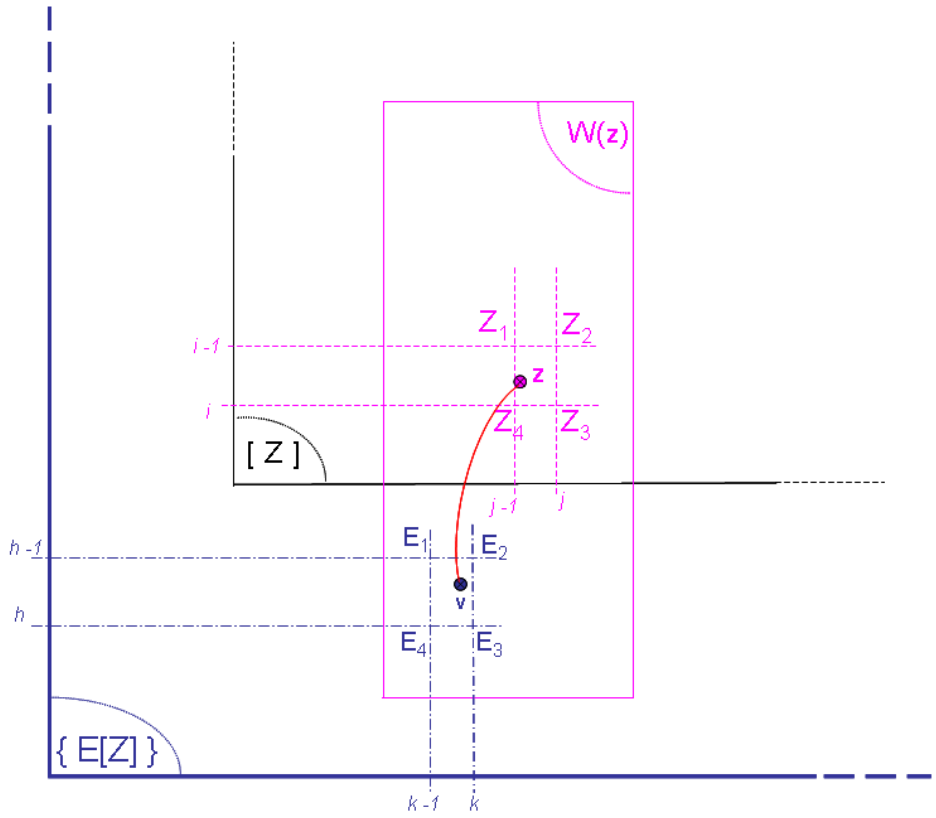

Fig. 2. Principle of interpolation in the C(*i,j,h,k*) grid:
$Cov(Z_n,v) = bilin.(v)\ [Cov(Z_n,E_1), Cov(Z_n,E_2), Cov(Z_n,E_3), Cov(Z_n,E_4)]$;
then $Cov(z,v\ ) = bilin.(z)\ [Cov(Z_1,v), Cov(Z_2,v), Cov(Z_3,v), Cov(Z_4,v)]$.

Point z is surrounded by 4 nodes Z(p), (p = 1 to 4), of the grid covering the inner zone [Z], and point v is also surrounded by 4 points, E(q), (q=1 à 4), of the grid corresponding to the envelope {E(Z)}. Each point E(q) may be considered neighbour to each Z(p), i.e. in the window W[Z(p)] - except in some peculiar case when one E(q) (or two E(q)'s at most) are ("slightly" - since v must be in W[z]) outside W[Z(p)]; in such special case one keeps only the E(q)'s which are satisfactory, and one applies a different interpolation formula.

(i) Positioning z in [Z] and v in {E} :

lat(n) and lon(m) being the latitudes and longitudes of the nodes of [Z] (and of {E}),
with: lat(n)=fimax-(n-1)*dfi, lon(m)=xlmin+(nlo-1)*dxl,
n=1 to nfi for [Z], and n=1-H to nfi+H for {E},
m=1 to nlo for [Z] , and m=1-K to nlo+K for {E},
one determines : i and i1=i-1 such that lat(i) $<$ z_lat $\leq$ lat(i1),
j and j1=j-1 such that lon(j1) $\leq$ z_lon $<$ lon(j) ,
then : h and h1=h-1 such that lat(h) $<$ v_lat $\leq$ lat(h1),
k and k1=k-1 such that lon(k1) $\leq$ v_lon $<$ lon(k) .

One then defines the surrounding points in the following order :

- for z : Z(1) = z(i1,j1) Z(2) = z(i1,j)
Z(4) = z(i ,j1) Z(3) = z(i ,j) $\Leftarrow$ p =1,2,3,4

- for v : E(1) = e(h1,k1) E(2) = e(h1,k)
E(4) = e(h ,k1) E(3) = e(h ,k) $\Leftarrow$ q =1,2,3,4

(ii) Compute, for p = 1 to 4 : c(p) = Cov(Z(p),v)

Each c(p) is obtained by bilinear interpolation, at v, of the 4 values Cov(Z(p),E(q)),
(p fixed, q=1 to 4), or by a weighted average when we are in a particular case (cf. above).
The values Cov(Z(p),E(q)) are the C(i0,j0,h1,k1) , C(i0,j0,h1,k)
C(i0,j0,h ,k1) , C(i0,j0,h ,k),
read on nugrid and corresponding to the nodes E(q) surrounding v, i.e. for (i0,j0) equal to
(respectively) : (i1,j1) : for p=1,
(i1,j) : for p=2,
(i ,j) : for p=3,
(i ,j1) : for p=4.

(iii) Compute valcov = Cov (z,v), by bilinear interpolation, at z, of :

c(1) = Cov(Z(1),v) c(2) = Cov(Z(2),v)
c(4) = Cov(Z(4),v) c(3) = Cov(Z(3),v).

Remark (for the tests): if points z and v are both centers of a cell of [Z] (and of {E}),
then valcov = [sum of the Cov(Z(p),E(q))] / 16.

References

Balmino G., J.P. Barriot, N. Valès: Non Singular Fast Formulation of the Gravity Vector and Gravity Gradient Tensor in Spherical Harmonics, *Manuscripta Geodaetica*, n°15, pp. 11-16, 1990.

Balmino G.: Gravitational potential harmonics from the shape of an homogeneous body, *Cel. Mech. and Dyn. Astr.,* Vol 60, n°3, pp. 331-364, 1994.

Gerstl M.: On the recursive computation of the integrals of the associated Legendre functions, *Manuscripta Geodaetica*, n° 5, pp. 181-199, 1980

Annex

The algorithms of covhsmp and covhs2p

" *Variances and Covariances of Global Gravity Field Models. Application to GRACE and GOCE* "
(submitted to JoG, March 2008)

Revised : Dec. 2008. Title changed to :

"*Efficient propagation of error covariance matrices of gravitational models. Application to GRACE and GOCE.*"

Submitted to Journal of Geodesy *(April 8, 2008)*
Revised Dec. 17, 2008

Efficient propagation of error covariance matrices of gravitational models. Application to GRACE and GOCE.

G. Balmino (✉)
C.N.E.S., Groupe de Recherches de Geodesie Spatiale,
Observatoire Midi-Pyrenees
14, Avenue Edouard Belin, 31400 Toulouse, France
Tel.: +33 (0)5 61 33 2980
E-mail: balmino@dtp.obs-mip.fr

Abstract. We have applied efficient methods for computing variances and covariances of functions of a global gravity field model expanded in spherical harmonics, using the full variance-covariance matrix of the coefficients. Examples are given with recent models derived from GRACE (up to degree and order 150), and with simulated GOCE derived solutions (up to degree and order 200).

Keywords: Global gravity field modelling; Spherical harmonics; Covariance matrix; Error propagation

---

1 Introduction

From the Earth's gravity field mapping missions of this decade, especially GRACE and GOCE, many global models of the field in spherical harmonics are and will be produced. Besides the current use of these models, for instance in terms of grids of geoid heights, of gravity anomalies or any other functional of the gravitational potential useful in the geosciences and various applications, there is a need for knowing the error characteristics of the model as precisely as possible and include them in data assimilation procedures, for instance in oceanography. This not only requires the knowledge of the error variances but also of the error covariances between two points (sometimes called cross-variances). Their derivation from the gravity field model involves the full variance-covariance matrix which can be very large, implying *a priori* huge computational efforts.

We show in this paper how the rigorous computation of the variances and covariances over grids can be optimized by using elementary algebra. The software has been written and tested on a personal computer, in keeping the spherical harmonics variance-covariance matrix out of core and by minimizing drastically the number of times that the program requires to access data from the disk. We show some results based on GRACE recent solutions on the one hand, and on GOCE simulations on the other hand, which demonstrate the efficiency of the algorithms.

2 Formulation of the problem

Let q be any function of two variables, expanded into surface spherical harmonics, for instance a linear (or linearized) functional of the gravity field:

$$q = \sum_{\ell \le L} \sum_{m \le \ell} f_{\ell m} \, [\, C_{\ell m} P_{\ell m}(\sin\varphi) \cos m\lambda + S_{\ell m} P_{\ell m}(\sin\varphi) \sin m\lambda \,] \qquad (1)$$

or

$$q = Y^t X \qquad (2)$$

In current applications q can be: geoid height, gravity anomaly or disturbance, or their vertical gradient, equivalent water height (accounting for the loading effect), topography, or any other function subject to this type of representation.

This is written on a surface (e.g. the Earth's surface, a reference ellipsoid) with φ, λ being the geo/planeto-centric latitude and longitude, respectively, and where the coefficients $C_{\ell m}$ and $S_{\ell m}$ have been predetermined from observations of q (or one or several functions of q) by means of a least squares adjustment (of normal matrix N). In gravitational potential problems, the $C_{\ell 0}$'s may be residual harmonics (when the normal potential of a reference ellipsoid is subtracted). The $C_{\ell m}$ and $S_{\ell m}$ coefficients are usually normalized and ordered according to a certain numbering scheme, that is $X = \{C_{\ell m}\, ; S_{\ell m}\}_{\ell,m}$.

In (2) we have $Y = \{ f_{\ell m} P_{\ell m}(\sin\varphi) \cos m\lambda \, ; f_{\ell m} P_{\ell m}(\sin\varphi) \sin m\lambda \}_{\ell,m}$: vector of the corresponding spherical harmonic functions, ordered like the X components; the $P_{\ell m}$'s are the Legendre polynomials (m=0) and functions (m>0) of the geocentric latitude φ , normalized like the $C_{\ell m}$, $S_{\ell m}$. The $f_{\ell m}$'s are constants or functions of latitude (φ) only; for instance if q is the geoid height $f_{\ell m}=R$ (the radius of the mean spherical Earth) in spherical approximation, or $f_{\ell m}=GM\, R^{\ell}/(r^{\ell+1}\gamma)$ on the ellipsoid (with G : gravitational constant, M : Earth mass, r : radius vector and γ : normal gravity at the computation point). The $f_{\ell m}$'s may also incorporate filtering coefficients such as those developed by Jekeli (1981), or tapering coefficients limiting the harmonic series to some window in the (ℓ,m) domain for specific problems.

In most cases, q is a scalar function but it may also be vectorial in which case the $f_{\ell m}$ coefficients are also vectors; an example will be given in section 4. Also some functions (e.g. the deflection of the vertical in geodesy) may involve derivatives of the $P_{\ell m}$'s, which can be treated similarly.

A zone on the working surface (e.g. the Earth) and a grid covering this zone being defined, our goal is to compute:

(a) the variance of the error on q at each grid point; when q is a vector, we also want the covariances between its components at the same point;

(b) the error-covariances between a given point and all other points in the grid (up to a certain distance).

$\Gamma = \sigma_0^2\, N^{-1}$ being the covariance matrix of X, equal to the inverse of the normal matrix N, scaled by the unit variance factor σ_0, one has:

- for problem (a), at any point P:

$$\sigma^2(q) = Y^t \, \Gamma \, Y \qquad (3)$$

- for problem (b):

$$\begin{bmatrix} \sigma^2(q_1) & \mathrm{cov}(q_1,q_2) \\ \mathrm{cov}(q_1,q_2) & \sigma^2(q_2) \end{bmatrix} = \begin{bmatrix} Y_1^t \Gamma Y_1 & Y_1^t \Gamma Y_2 \\ Y_2^t \Gamma Y_1 & Y_2^t \Gamma Y_2 \end{bmatrix} \qquad (4)$$

where the subscripts 1, 2 refer to a pair of points P_1 and P_2 ; the diagonal terms can be computed as in problem (a), making (a) a particular case of problem (b).

We have developed efficient methods to compute over any equiangular grid:

- (i) at each node (i,j) point or mean values of $\sigma^2(q)$ for a Γ matrix of any size (Γ being either in core – when it can fit, or stored sequentially on disc); as it will be shown in section 3, this uses recursive evaluations of partial sums at longitudinal nodes for a fixed latitude (equivalent to a Fourier approach - FFT); for mean values, we evaluate the Legendre functions' integrals $I_{\ell m}$ rigorously by using a variant of the Gerstl (1980) formulation and described in Balmino (1994).
- (ii) at each node (i,j) of the regular grid the tensor $C_{ij}^{hk} = \mathrm{cov}(q_{ij}, q_{hk}) = Y_{(hk)}^t \, \Gamma \, Y_{(ij)}$ for all neighbouring nodes (h,k) – up to chosen distances in latitude and longitude, with $Y_{(\alpha\beta)}$ = value of harmonic functions vector components at node (α,β). This makes use of algorithms similar to the ones in (i) and also takes advantage of symmetries: $C_{ij}^{hk} = C_{hk}^{ij}$. Besides we also compute tables of the isotropic, North-South and East-West covariance functions over the zone, which provides the user the means to evaluate the covariances beyond the prescribed distances: an empirical anisotropic covariance function (depending on both distance ψ and azimuth) may be fitted to the $\mathrm{cov}^s(\psi)$ values where the superscript *s* stands for *isotropic*, *N-S* or *E-W*.

3 The core of the method

The main improvement over the standard computation of matrix and dot products involved in expressions such as $Y_1{}^t \, \Gamma \, Y_2$ consists, in the regular grid case, in accelerating some operations by adapting what we call the partial sums – longitude recursion algorithm (or PSLR) introduced by Bosch (1983). Fourier methods may also be used as in Haagmans and van Gelderen (1991) or Sneeuw and Bun (1996) but we favoured the simplicity of the PSLR. We will see later the similarities between the two approaches.

We want to evaluate the summations involved in (3) or (4) where the points P, or P_1 and P_2 are the nodes N_{ij} of the grid, defined by:

$$\begin{aligned} \varphi_i &= \varphi_0 - i\,\Delta\varphi \qquad (i = 0,1,...I) \\ \lambda_j &= \lambda_0 + j\,\Delta\lambda \qquad (j = 0,1,...J) \end{aligned} \tag{5}$$

The grid may be parameterized in geodetic latitude, in which case φ_i is transformed into geocentric latitude when needed. Also $\Delta\lambda$ could be made latitude dependant in the case of a quasi-equivalent area decomposition of the unit sphere, which does not change the basis of the algorithm.

[φ_I , φ_0] x [λ_0 , λ_J] defines the working zone [Z], called the "inner zone" in the covariance computation case.

When mean values are computed, the bins are centered at $(\varphi_i + \varphi_{i+1})/2$ and at $(\lambda_j + \lambda_{j+1})/2 = \overline{\lambda}_j$, and we replace the quantities $(\cos m\lambda_j, \sin m\lambda_j)$ by $(\theta_m \cos m\overline{\lambda}_j, \theta_m \sin m\overline{\lambda}_j)$ where $\theta_0 = 1$, $\theta_m = \sin(m\,\Delta\lambda/2) \,/\, (m\,\Delta\lambda/2)$; in addition the $P_{\ell m}$'s (or their derivatives in the case of some geodetic functions) are replaced by their mean values over the latitude extent of each bin.

For the covariances we define a moving window $W_{HK}(N_{ij})$ around each node (i, j) which consists of all points (φ_h , λ_k) such that i-H ≤ h ≤ i+H and j-K ≤ k ≤ j+K. The union of [Z]

and all W_{HK} 's is a domain called the envelope {E[Z]} of the inner zone. {E[Z]} may go beyond a pole or have a λ- extension larger than 2π, of which we have taken care: when $|\varphi_h| > \pi/2$ we replace [φ_h , λ_k] by [sgn(φ_h).π-φ_h , λ_k+π] and in the latter particular case we extend the range of the longitude index.

With the notations of equation (4) let us detail the computation of $Y_1{}^t \Gamma\, Y_2$ for two points P_1 and P_2 which are nodes of {E[Z]}, with for instance P_2 being in [Z] and P_1 in $W_{HK}(P_2)$.

We first compute $\Gamma Y_2 = V_2$. A component V_2^n corresponds to one row of Γ, therefore:

$$\left(\Gamma Y_2\right)^n = \sum_{\ell=\ell_1(0)}^{\ell_2(0)} \Gamma^n_{j_c(\ell,o)} F_{\ell_o} + \sum_{m=1}^{L} \sum_{\ell=\ell_1(m)}^{\ell_2(m)} \left[\Gamma^n_{j_c(\ell,m)} F_{\ell m} \cos m\lambda + \Gamma^n_{j_s(\ell,m)} F_{\ell m} \sin m\lambda\right] \tag{6}$$

where $F_{\ell m} = f_{\ell m}\ P_{\ell m}$ or $f_{\ell m}\ I_{\ell m}\ \theta_m$ in the case of mean values. In this equation (and subsequently) we dropped the subscript "2" from the $F_{\ell m}$'s and λ 's since there should be no confusion on where these quantities are evaluated.

$\ell_1(m)$, $\ell_2(m)$ determine the range of the columns which limit the considered blocks of the covariance matrix through which the effective columns $j_c(\ell,m)$, $j_s(\ell,m)$ of Γ are selected. These arrays are determined once and for all at the beginning, and correspond to the adopted numbering scheme. Usual values are $\ell_1(m) = \sup(1,m)$ and $\ell_2(m) = L$.

(i) Partial sums (PS)

Equation (6) can be rewritten as follows:

$$\begin{aligned}\left(\Gamma Y_2\right)^n &= A_0^n + \sum_{m=1}^{L} A_m^n \cos m\lambda + B_m^n \sin m\lambda \\ &= \sum_{m=0}^{L} h_m^n(\lambda)\end{aligned} \tag{7}$$

where:

$$A_m^n = \sum_{\ell=\ell_1(m)}^{\ell_2(m)} \Gamma^n_{j_c(\ell,m)} F_{\ell m} \quad , \quad (m = 0,1,...L)$$

$$B_m^n = \sum_{\ell=\ell_1(m)}^{\ell_2(m)} \Gamma^n_{j_s(\ell,m)} F_{\ell m} \quad , \quad (m = 1,2,...L)$$

These sums, independent of λ, can be computed without the $\cos^m\varphi$ factor from the Legendre functions (or from the integrals $I_{\ell m}$ with a modification of the algorithm which evaluate them), also in applying an empirical scaling factor so as to gain in accuracy and to cope with the allowed magnitude of real numbers in the used computer (a necessity when L becomes large); in doing so we have followed Holmes and Featherstone (2002).

(ii) Longitude recursion (LR)

Since $\lambda_j = \lambda_o + j\ \Delta\lambda$ (or $\lambda_o + (j + \frac{1}{2})\ \Delta\lambda$ in the case of mean values) it is easy to show (see appendix A) that:

$$h^n_{m,j} = h^n_m\left(\lambda_j\right) = \gamma^n_m \cos\left(mj\ \Delta\lambda\right) + \sigma^n_m \sin\left(mj\ \Delta\lambda\right) \tag{8}$$

with

$$\gamma_m^n = A_m^n \cos m\lambda_o + B_m^n \sin m\lambda_o \quad , \quad (m > o)$$
$$\gamma_o^n = A_o^n$$
$$\sigma_m^n = B_m^n \cos m\lambda_o - A_m^n \sin m\lambda_o \quad , \quad (m > o)$$

Expressing cos ($mj\ \Delta\lambda$) and sin ($mj\ \Delta\lambda$) in terms of sine and cosine of m (j-*1*) $\Delta\lambda$ and m (j-*2*) $\Delta\lambda$, we find the following recursion formula:

$$h_{m,j}^n = 2\ \cos m\,\Delta\lambda\ \ h_{m,j-1}^n - h_{m,j-2}^n \tag{9}$$

to be initialized by:

$$h_{m,o}^n = \gamma_m^n$$
$$h_{m,1}^n = \gamma_m^n\ \cos m\,\Delta\lambda + \sigma_m^n \sin m\,\Delta\lambda$$

Needless to say that it is not necessary, in programming this algorithm, to keep the superscript n as an array index; so not only this trick works fast but it is also economical memory-wise.

Finally we perform the summation over m of the $h_{m,j}^n$ for each λ_j, in applying a Horner scheme of order m in $\cos\varphi$ and dividing by the previously applied scaling factor as in Holmes and Featherstone (*ibid.*).

To achieve the computation, it suffices to calculate the dot product $Y_1^t\, V_2$ that is $\sum_i Y_1^n \left(\Gamma Y_2\right)^n$ a writing which shows that this can be done by successive accumulation as one browses the Γ matrix forward and for all points at a time in $W_{HK}(P_2)$, with a similar PSLR technique.

Because the variance-covariance matrix Γ is rather large it generally cannot be in core. Since the efficiency of the algorithm also comes from the use of the full square Γ matrix we first – and once for all, complement (into a square matrix) the covariance matrix when it is given as a (lower) triangular one, on disc; the software which performs this transformation does not (can not) put the triangular matrix in core but instead uses a limited size working array. Optimization is also enforced in minimizing the number of times the program accesses data from the disk. A revolving buffer zone and (relatively small) auxiliary arrays help reducing the amount of re-reading of Γ; we work simultaneously with the largest possible number of parallels (of latitude φ_i) in [Z] and also optimize the storage of the Legendre functions and integrals over these parallels.

The Fourier approach, used for instance by Haagmans and van Gelderen, is in principle similar to ours: the first gain in computer time is in the PS part of the PSLR and results, as in our case, from their reversal of the summation sequence over degree ℓ and order m (a trick which has been commonly applied); then the second gain comes from their use of fast Fourier transform (FFT) instead of our longitude recursions (LR). The technical differences lie: (i) in the covariance matrix which needs to be in core in their approach whereas it is brought in core one row at a time in ours; (ii) in the isolation of the $\cos^m\varphi$ factors and the use of an Horner summation scheme.

4 Performances and examples

The software, written in Fortran 90, has been developed and run on a standard (3 GHz single processor) personal computer with 1 Gb memory and 180 Gb disk space. It has been validated thanks to a built-in verification procedure by brute force (direct method) at a few grid nodes, also by extensive comparisons with results from an independent software written in the 1980's at the French Space Centre. The latter, which uses only the PS part of the PSLR algorithm and runs on mainframe had been validated against another software independently developed at GFZ (GeoForschungZentrum). Besides, comparisons have been made in the diagonal matrix case (i.e. when we do not consider correlations between the coefficients) with another independent software (Knudsen, *personal communication,* 2008) and good agreement was obtained.

An interpolation procedure (needed in most applications) has also been developed, which allows the fast computation of cov(q_1, q_2) from the 4-D grid of the C_{ij}^{hk} values; it uses two bilinear interpolators involving 8 points associated in 16 pairs.

Figure 1 shows the errors on the geoid associated with the EIGEN-GL04S1 solution, a (static, mean) model derived from four years of GRACE data and Lageos 1 and 2 data (over the same time period), complete to degree and order 150 (Foerste et al., 2007). The grid resolution is 1°x1°. The full Γ matrix occupies 4 Gb in this case. The elapsed time on the computer referred to above was 15 minutes.

********** FIG. 1 **********

Fig. 1 : Map of the geoid errors (in centimeter) of the EIGEN-GL04S1 model derived from GRACE and Lageos 1 and 2 satellites (the r.m.s. error is 6.2 cm). The errors are slightly asymmetric with respect to the equator which may be attributed to the errors on the odd zonal coefficients.

Figures 2 and 3 show results of covariances computation. They correspond to a GOCE mission simulation (Abrikosov et al., 2006). It was performed for one measuring phase of 6 months, at a mean altitude of 265 km. The Earth's gravity field is recovered up to degree and order 200, which makes the full Γ matrix occupy 13 Gb on disk. The considered zone [Z] extends from 20°N to 80°N and from 60°W to 30°E. With a step size of one degree in latitude and longitude and with H=K=20, the computation took three hours, of which 1h20mn were needed to read the matrix (this was determined separately). Three windows of 40° x 40° are shown on figure 2. The covariances are in m^2.

The isotropic, north-south and east-west covariance functions which are simultaneously computed are shown on figure 3. They demonstrate an increasing anisotropy for distances larger than a few degrees; the east-west component in particular shows larger error covariances which may be attributed to the limited measurement bandwidth of the gradiometer and some characteristics of the processing (arc length, weighting and regularizing strategies, etc).

********** FIG. 2 **********

Fig. 2 : Example of 40°x40° windows of geoid error covariances (in cm^2) for a 6 months GOCE simulation. Longitudes are on the horizontal axes, latitudes on the vertical ones.

********** FIG. 3 **********

Fig. 3 : Isotropic, N-S and E-W covariance functions determined over a 60° x 90° area for a six months GOCE simulation (values of the N-S and E-W functions beyond 20 and 15 degrees respectively are not computed due to the window size, the limited area and the equiangular geometry). Values are in cm^2. The larger error of the E-W component compared to the N-S one may have to do with the along-track data sampling interval (about 8 km) versus the across-track separation of the ground tracks (about 35 km for the simulated orbit).

Finally, to illustrate the case of a vectorial function q, we give an example of covariances of the geoid model induced error on the geostrophic currents.

Let $h\ (\varphi,\lambda)$ be the sea surface topography at any place on the oceans, $N\ (\varphi,\lambda)$ the geoid height and $H\ (\varphi,\lambda)$ the ellipsoid height at this point. The geostrophic current velocity vector $q = (\dot{x}, \dot{y})$ is given by:

$$\begin{aligned} \dot{x} &= -\frac{g}{f\,R}\frac{\partial h}{\partial \varphi} \\ \dot{y} &= \frac{g}{f\,R}\frac{1}{\cos\varphi}\frac{\partial h}{\partial \lambda} \end{aligned} \tag{10}$$

with R the mean Earth's radius as before, $f = 2\ \omega \sin\ \varphi$ (ω = Earth's rotation rate in rad/s), g the mean surface gravity. $\{x, y\}$ is a local system with x being eastward and y northward. This equation may be used about 5 to 10 degrees off the equator.

On the other hand, we have:

$$h = H - N \tag{11}$$

Assuming no correlation between H – which comes from satellite altimetric measurements, and N – which comes from a global gravity field model, we write:

$$\mathrm{cov}\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \frac{g^2}{f^2}\left[\mathrm{cov}\begin{pmatrix} \frac{1}{R} & \frac{\partial H}{\partial \varphi} \\ \frac{1}{R\cos\varphi} & \frac{\partial H}{\partial \lambda} \end{pmatrix} + \mathrm{cov}\begin{pmatrix} \xi \\ \eta \end{pmatrix} \right] \tag{12}$$

where (ξ, η) are the components of the deflection of the vertical.

The first term in (12) is attributed to sea surface slope errors coming from instrumental errors (i.e. from the altimeter measuring system) while the second term comes from the error on the geoid. It is this second part which can be easily computed and visualized by its error ellipse at any point (see appendix B), from the gravity harmonics covariance matrix. Figure 4 shows such error ellipses for the EIGEN-GL04S1 model at grid points of a limited area in the north-west Atlantic Ocean, super-imposed on the Levitus current velocities.

********** FIG. 4 **********

Fig. 4: Error ellipses of the geostrophic current velocity vector, induced by errors on the global gravity field model EIGEN-GL04S1. The arrows show the velocity vector itself according to Levitus model. The size of the circle and of the vertical arrow at the bottom indicate the scale (in cm). Ellipses are elongated in the N-S direction because this component (y) corresponds to errors on the derivative of the geoid height in longitude, and because the GRACE mission produces the best accuracy in the N-S direction.

Such an analysis is complementary to the computation and analysis of the covariances; it is useful for quickly assessing the isotropic character of errors on any gravity model, or departure from it over a given area.

5 Conclusion

We have found and implemented simple, though efficient algorithms which allow to propagate geodetic function error variances and covariances from the complete covariance matrix associated to a global gravitational potential model expanded in spherical harmonics. The software currently runs on a standard personal computer and is used in applications relating to the GRACE and GOCE gravity mapping missions.

Acknowledgments. I thank my colleagues from the CNES Toulouse Space Center: R. Biancale and J.M. Lemoine for providing the GRACE covariance matrix, J.C. Marty and S. Bruinsma for providing the GOCE simulated covariance matrix, and N. Vales who helped in running the graphical software part. I also thank the reviewers who made very constructive comments and helped improving the paper.

Appendix A: The LR algorithm

We rewrite equation (7) of the text in dropping superscript n:

$$h = \sum_{m=0}^{L} h_m$$

with

$$h_m = A_m \cos m\lambda + B_m \sin m\lambda$$

which has to be evaluated for each $\lambda_j = \lambda_0 + j\Delta\lambda$.
The LR algorithm is a recursive relation between the $h_m(\lambda_j) = h_{m,j}$ with m fixed.
We have:

$$\begin{aligned} h_{m,j} &= A_m \cos(m\lambda_0 + mj\Delta\lambda) + B_m \sin(m\lambda_0 + mj\Delta\lambda) \\ &= (A_m \cos m\lambda_0 + B_m \sin m\lambda_0)\cos mj\Delta\lambda + (B_m \cos m\lambda_0 - A_m \sin m\lambda_0)\sin mj\Delta\lambda \\ &= \gamma_m \cos mj\Delta\lambda + \sigma_m \sin mj\Delta\lambda \end{aligned}$$

with obvious notations; note that $\gamma_0 = A_0$ and $\sigma_0 = 0$.

We then use the following trigonometric identities:

$$\cos jx = 2\cos x \ \cos(j-1)x - \cos(j-2)x$$
$$\sin \ jx = 2\cos x \ \sin(j-1)x - \sin(j-2)x$$

with $x = m\Delta\lambda$.

Therefore:

$$\begin{aligned} h_{m,j} = \ & \gamma_m .2\cos m\Delta\lambda \cos m(j-1)\Delta\lambda - \gamma_m \cos m(j-2)\Delta\lambda \\ & + \sigma_m .2\cos m\Delta\lambda \ \sin m(j-1)\Delta\lambda - \sigma_m \sin m(j-2)\Delta\lambda \\ = \ & 2\cos m\Delta\lambda \ [\, \gamma_m \cos m(j-1)\Delta\lambda + \sigma_m \sin m(j-1)\Delta\lambda \,] \\ & - [\, \gamma_m \cos m(j-2)\Delta\lambda + \sigma_m \sin m(j-2)\Delta\lambda \,] \end{aligned}$$

that is

$$h_{m,j} = 2\cos m\Delta\lambda \ \ h_{m,j-1} - h_{m,j-2}$$

which is initialized with $h_{m,0}$ and $h_{m,1}$.
This is the recursive formula found by Bosch (*ibid.*) and which we have used in this paper.

Appendix B: Error ellipse of the geostrophic current velocity vector

The covariances are gridded on the Earth's surface, using the algorithm described in the text. At each grid node, we have:

$$C = \mathrm{cov}\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \sigma_{\dot{x}}^2 & \sigma_{\dot{x}\dot{y}} \\ \sigma_{\dot{x}\dot{y}} & \sigma_{\dot{y}}^2 \end{pmatrix}$$

The equation of the ellipse of error at this point in the local coordinate system $\{x, y\}$ is:

$$(xy)^t \ C^{-1} \ (xy) = 1$$

that is:

$ax^2 + 2bxy + cy^2 = 1$

where:

$a = \sigma_{\dot{y}}^2 / \Delta$
$b = -\sigma_{\dot{x}\dot{y}} / \Delta$
$c = \sigma_{\dot{x}}^2 / \Delta$
with: $\Delta = \sigma_{\dot{x}}^2 \sigma_{\dot{y}}^2 - \sigma_{\dot{x}\dot{y}}^2 > 0$ (since C is symmetric, positive definite)

- If $a \neq c$, let us define

$$\theta = \frac{1}{2} Arctg \frac{b}{a-c}$$
$$a' = a\cos^2\theta + 2b\sin\theta\cos\theta + c\sin^2\theta$$
$$c' = a\sin^2\theta - 2b\sin\theta\cos\theta + c\cos^2\theta$$

. if $a' \leq c'$, then θ is the polar angle, with respect to the West-East direction, of the semi-major axis of the ellipse, which value is: $\sqrt{1/a'}$; the semi-minor axis is $\sqrt{1/c'}$.

. if $a' > c'$, θ is replaced by $\theta + \frac{\pi}{2}$ and the axes are inter-changed.

- If $a = c$

. if $b = o$: the ellipse degenerates into a circle

. if $b \neq o$: $\theta = \frac{\pi}{4}$, $a' = a + b$, $c' = a - b$. If $b < o$, the semi-major axis of the ellipse is $\sqrt{1/(a+b)}$, the semi-minor axis is $\sqrt{1/(a-b)}$. In the other case ($b \geq o$), $\theta = \frac{3\pi}{4}$ and the axes are interchanged.

References

Abrikosov O, Foerste C, Rothacher M, Bruinsma S, Marty JC, Balmino G (2006) Gravity Field Recovery with Simulated GOCE Observations, EGU, Vienna (A), Session G8

Balmino G (1994) Gravitational potential harmonics from the shape of an homogeneous body, Cel Mech and Dyn Astr, Vol 60, N°3: 331-364

Bosch W (1983) Effiziente Algorithmen zur Berechnung von Raster-Punkwerten von Kugelfunktionsentwicklungen, Memorandum, D.G.F.I. , Munich

Foerste C, Schmidt R, Stubenvoll R, Flechtner F, Meyer U, Konig R, Neumayer H, Biancale R, Lemoine JM, Bruinsma S, Loyer S, Barthelmes F, Esselborn S (2007) The GFZ/GRGS Satellite-Only and Combined Gravity Field Models : EIGEN-GL04S1 and EIGEN-GL04C.

J Geod, doi: 10.1007/s00190-007-0183-8
Gerstl M (1980) On the recursive computation of the integrals of the associated Legendre functions, Manuscripta Geodaetica 5: 181-199
Haagmans RHN and van Gelderen M (1991) Error Variances-Covariances of GEM-T1: Their Characteristics and Implications in Geoid Computation. J. Geophys. Res., Vol. 96 (B12), 20,011-20,022
Holmes SA and Featherstone WE (2002) A unified approach to the Clenshaw summation and the recursive computation of very high degree and order normalised associated Legendre functions. J Geod 76: 279-299
Jekeli C (1981) Alternative methods to smooth the Earth's gravity field, Rep 310, Department of Geodetic Science and Surveying, Ohio State University, Columbus
Sneeuw N and Bun R (1996) Global spherical harmonic computation by two-dimensional Fourier methods. J Geod 70: 224-232

---------- end of text

4 figures follow (one per page) ----------

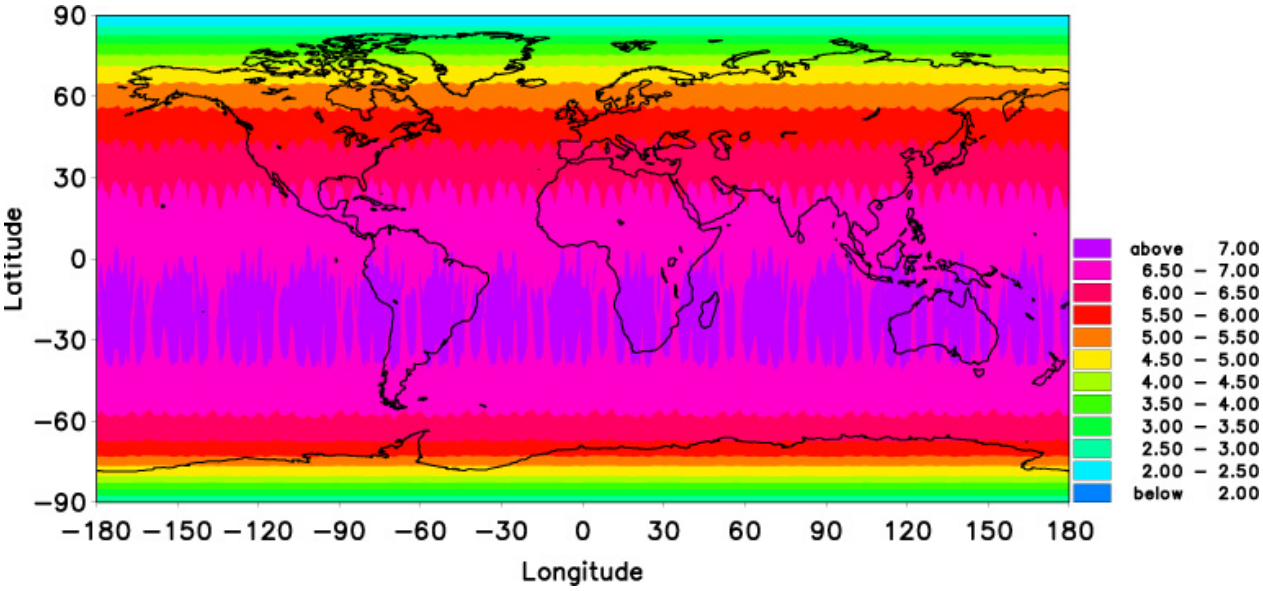

Fig. 1 : Map of the geoid errors (in centimeter) of the EIGEN-GL04S1 model derived from GRACE and Lageos 1 and 2 satellites (the r.m.s. error is 6.2 cm). The errors are slightly asymmetric with respect to the equator which may be attributed to the errors on the odd zonal coefficients.

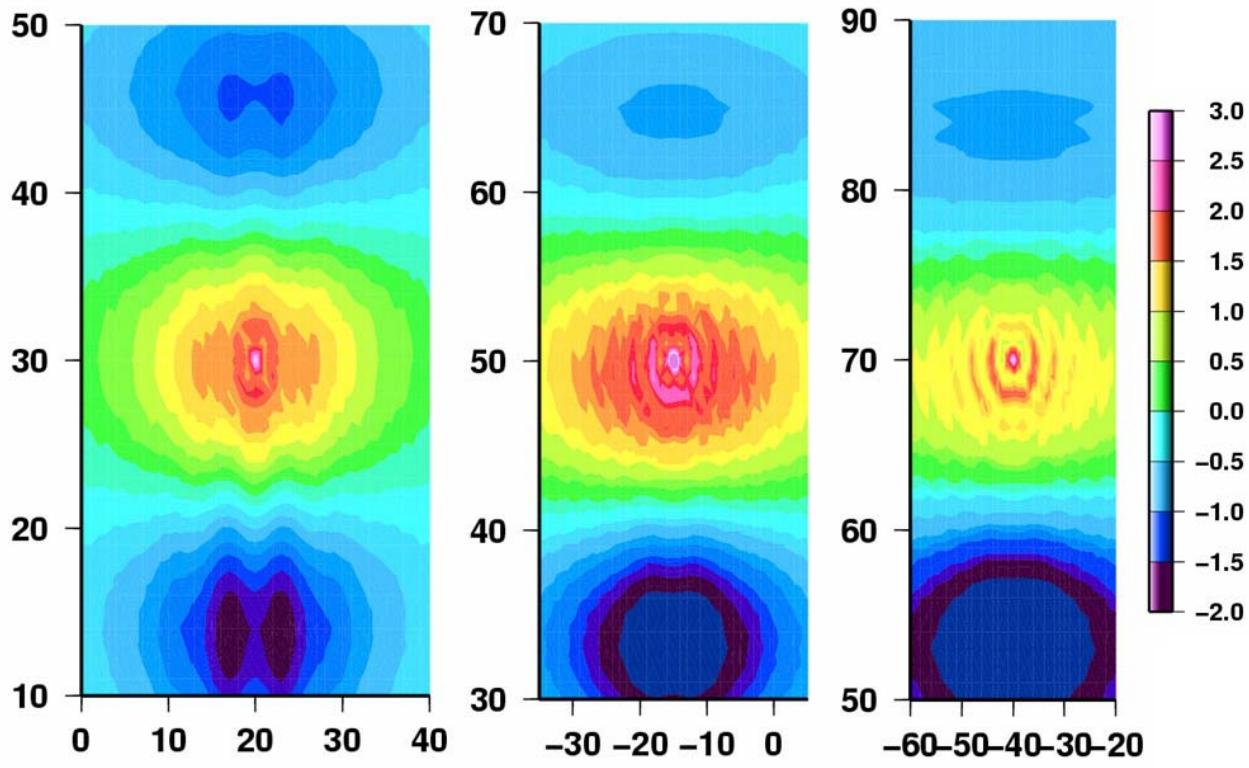

Fig. 2 : Example of 40°x40° windows of geoid error covariances (in cm^2) for a 6 months GOCE simulation. Longitudes are on the horizontal axes, latitudes on the vertical ones.

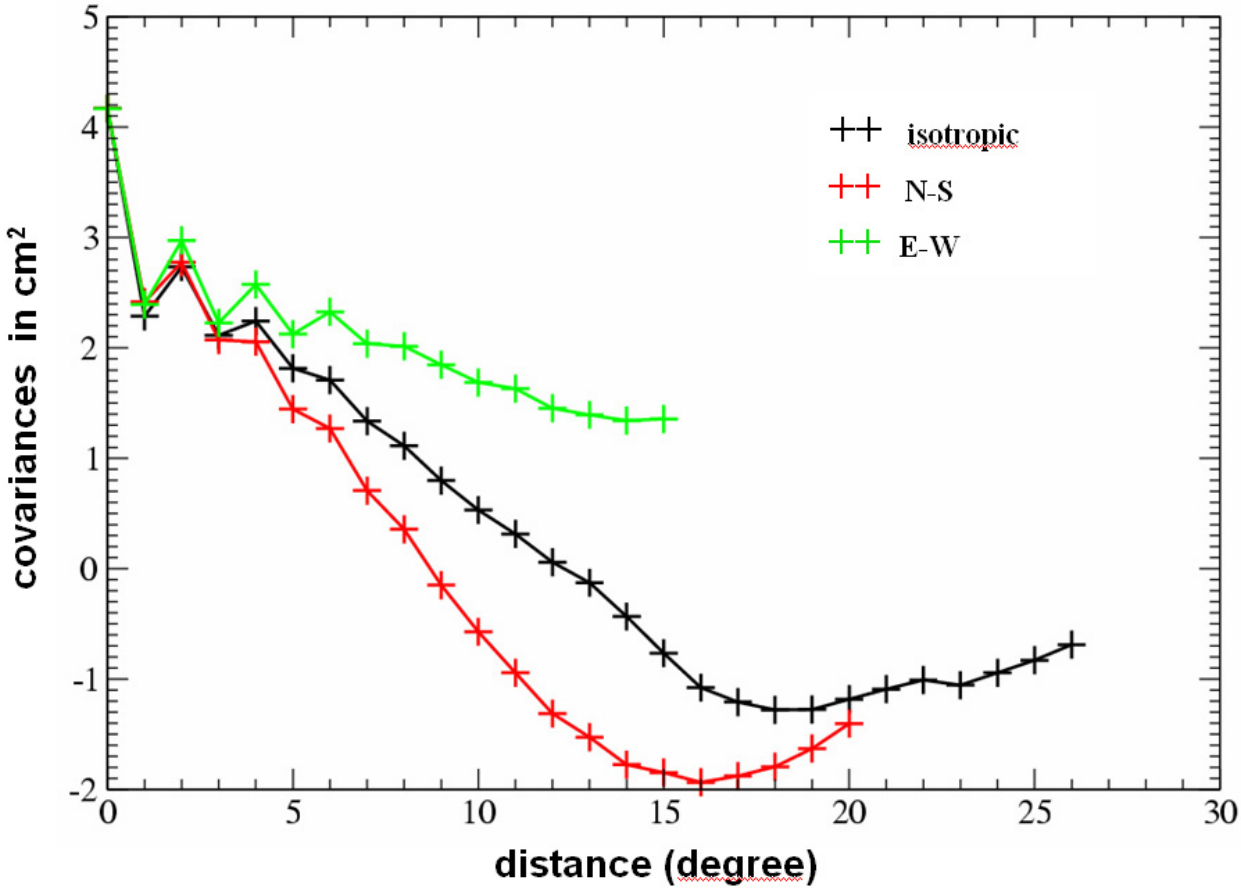

Fig. 3 : Isotropic, N-S and E-W covariance functions determined over a 60° x 90° area for a six months GOCE simulation (values of the N-S and E-W functions beyond 20 and 15 degrees respectively are not computed due to the window size, the limited area and the equiangular geometry). Values are in cm^2. The larger error of the E-W component compared to the N-S one may have to do with the along-track data sampling interval (about 8 km) versus the across-track separation of the ground tracks (about 35 km for the simulated orbit).

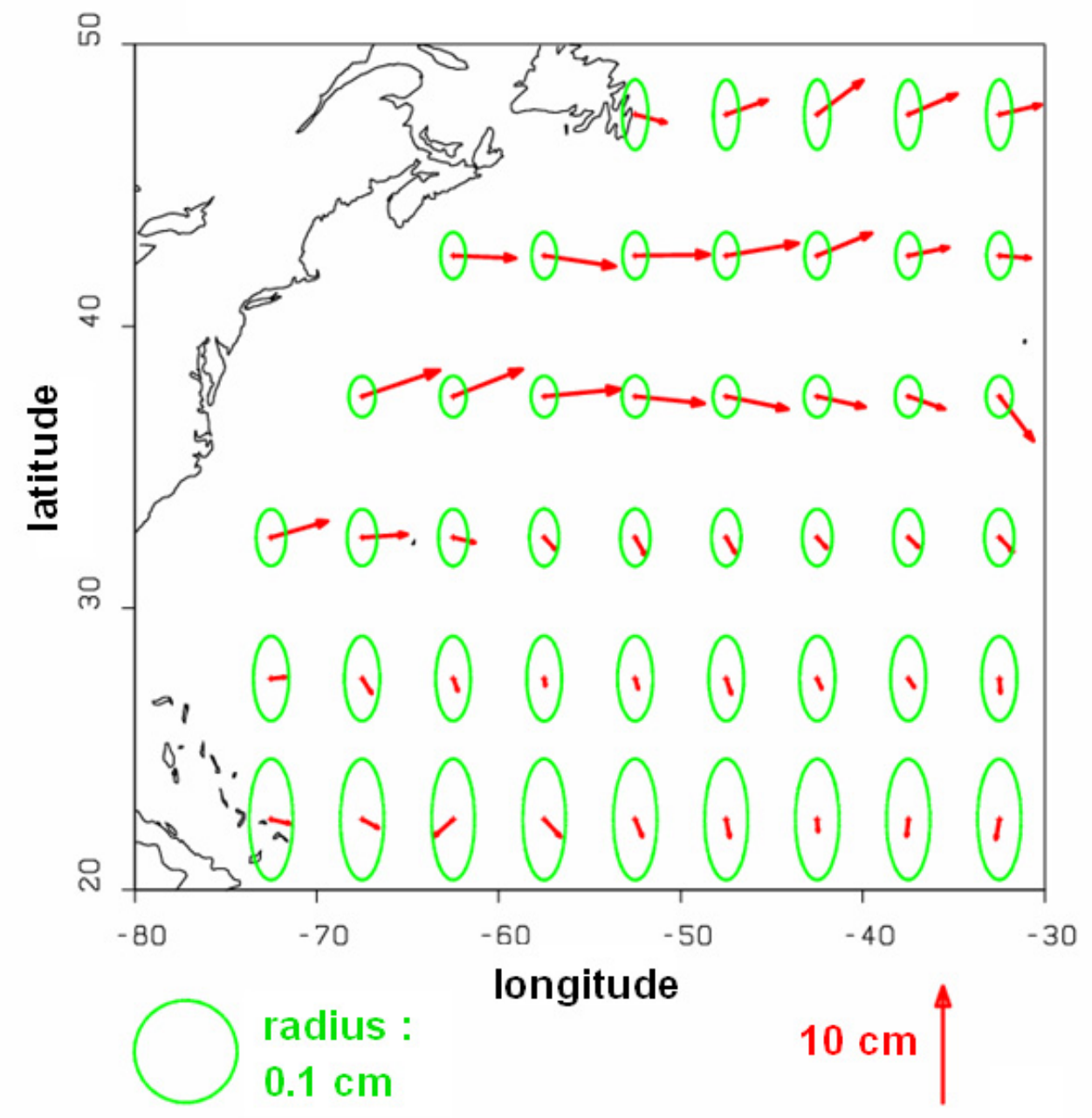

Fig. 4: Error ellipses of the geostrophic current velocity vector, induced by errors on the global gravity field model EIGEN-GL04S1. The arrows show the velocity vector itself according to Levitus model. The size of the circle and of the vertical arrow at the bottom indicate the scale (in cm). Ellipses are elongated in the N-S direction because this component (y) corresponds to errors on the derivative of the geoid height in longitude, and because the GRACE mission produces the best accuracy in the N-S direction.