HOWTO g02: Gravitational field models

You will learn about the formats of gravitational field models that are supported by GrafLab.

All the GrafLab input parameters are explained in ../docs/graflab.md.

```
clear; clc; init_checker();
```

Structure of a global geopotential model (GGM) file

The structure of the GGM coefficients table must follow either "Table 1" or "Table 2". Any other ordering scheme may not be processed correctly by GrafLab, so is not recommended.

Table 1: GGM coefficients table up to degree 2. The order of the columns is harmonic degree, harmonic order, Cnm and Snm

```
0   0    1.00000E+00    0.00000E+00
1   0    0.00000E+00    0.00000E+00
1   1    0.00000E+00    0.00000E+00
2   0   -0.48417E-03    0.00000E+00
2   1   -0.20662E-09    0.13844E-08
2   2    0.24394E-05   -0.14003E-05
```

Table 2: GGM coefficients table up to degree 2. The order of the columns is harmonic degree, harmonic order, Cnm and Snm

```
0   0    1.00000E+00    0.00000E+00
1   0    0.00000E+00    0.00000E+00
2   0   -0.48417E-03    0.00000E+00
1   1    0.00000E+00    0.00000E+00
2   1   -0.20662E-09    0.13844E-08
2   2    0.24394E-05   -0.14003E-05
```

GGM as a MATLAB's binary file

The "mat" file must store one variable only (there is an exception to be explained in "HOWTO g11"). The variable must be a matrix with the structure as shown in "Table 1" or "Table 2" above. An example of a valid MATLAB binary file format of a GGM can be found in "../data/input/EGM96.mat".

Let's define the GrafLab input parameters.

```
GM                = 3986004.415E+8;
R                 = 6378136.3;
nmin              = 0;
nmax              = 'nmaxGGM';
ellipsoid         = 1;
GGM_mat           = '../data/input/EGM96.mat';  % This is the MATLAB binary
                                                % file with GGM coefficients
crd               = 0;
point_type        = 0;
lat_grd_min       = -90.0;
lat_grd_step      =   1.0;
lat_grd_max       =  90.0;
lon_grd_min       =   0.0;
lon_grd_step      =   1.0;
```

```
lon_grd_max       = 360.0;
h_grd             =   0.0;
out_path          = '../data/output/howto-g02-table-mat';
quantity_or_error = 0;
quantity          = 5;
fnALFs            = 1;
export_data_txt   = 1;
export_report     = 1;
export_data_mat   = 1;
display_data      = 0;
status_bar        = 1;
```

Do the synthesis

```
out_mat = GrafLab('OK', ...
    GM, ...
    R, ...
    nmin, ...
    nmax, ...
    ellipsoid, ...
    GGM_mat, ...
    crd, ...
    point_type, ...
    lat_grd_min, ...
    lat_grd_step, ...
    lat_grd_max, ...
    lon_grd_min, ...
    lon_grd_step, ...
    lon_grd_max, ...
    h_grd, ...
    [], ...
    [], ...
    [], ...
    [], ...
    out_path, ...
    quantity_or_error, ...
    quantity, ...
    fnALFs, ...
    [], ...
    export_data_txt, ...
    export_report, ...
    export_data_mat, ...
    display_data, ...
    [], ...
    [], ...
    [], ...
    [], ...
    status_bar);
```

GGM in a text format

This example shows how to import a text format of GGM that obeys the structure of "Table 1" or "Table 2". To this end, we save the "GGM_mat" file to a text file. Next, we update some GrafLab input parameters.

```
GGM = load(GGM_mat);
GGM = GGM.EGM96;
GGM_txt  = '../data/output/EGM96.txt';  % This will be the text version of
                                        % "GGM_mat"
save(GGM_txt, 'GGM', '-ascii', '-double');
out_path = sprintf('../data/output/howto-g02-table-txt');
```

Do the synthesis

```
out_txt = GrafLab('OK', ...
    GM, ...
    R, ...
    nmin, ...
    nmax, ...
    ellipsoid, ...
    GGM_txt, ...
    crd, ...
    point_type, ...
    lat_grd_min, ...
    lat_grd_step, ...
    lat_grd_max, ...
    lon_grd_min, ...
    lon_grd_step, ...
    lon_grd_max, ...
    h_grd, ...
    [], ...
    [], ...
    [], ...
    [], ...
    out_path, ...
    quantity_or_error, ...
    quantity, ...
    fnALFs, ...
    [], ...
    export_data_txt, ...
    export_report, ...
    export_data_mat, ...
    display_data, ...
    [], ...
    [], ...
    [], ...
    [], ...
    status_bar);
```

Now check the synthesis from the MATLAB's binary file and from the text file.

```
fprintf("The RMS of the difference is %0.16e\n", rms(out_mat(:, end) - ...
                                                    out_txt(:, end)));
```

GGM in the GFC format

The "gfc" format is defined by ICGEM (http://icgem.gfz-potsdam.de/ICGEM-Format-2011.pdf).

GrafLab should be able to process most of the *static* models found on the ICGEM website. It *cannot* process temporal models. The temporal models can easily be identified, as they use terms such as "gfct", "trnd", "acos", "asin", etc. If you attempt import a temporal model, you should get an error message.

Whenever you import a "gfc" file, GrafLab reads its header and takes the "GM" and "R" constants from that file. The "GM" and "R" values that we pass to GrafLab are *not* used in case of "gfc" files, so we can use any positive real number, it simply does not matter. GrafLab will print a warning if your "GM" and/or "R" values are different from those found in the "gfc" file. You may verify that GrafLab took correctly the "GM" and "R" values from the "gfc" file by inspecting the report file exported by GrafLab.

In this example, we import some more or less random static model downloaded from the ICGEM website.

Now we need to define the GrafLab input parameters. We intentionally set "GM" and "R" to a wrong value "1.0" to demonstrate that GrafLab will ignore our choice and will use the correct values from the "gfc" file.

```
GM       = 1.0;
R        = 1.0;
GGM_gfc  = '../data/input/GO_CONS_GCF_2_TIM_R6.gfc';  % This is the "gfc" file
out_path = '../data/output/howto-g02-gfc';
```

Do the synthesis

```
out_txt = GrafLab('OK', ...
    GM, ...
    R, ...
    nmin, ...
    nmax, ...
    ellipsoid, ...
    GGM_gfc, ...
    crd, ...
    point_type, ...
    lat_grd_min, ...
    lat_grd_step, ...
    lat_grd_max, ...
    lon_grd_min, ...
    lon_grd_step, ...
    lon_grd_max, ...
    h_grd, ...
    [], ...
    [], ...
    [], ...
    [], ...
    out_path, ...
    quantity_or_error, ...
    quantity, ...
    fnALFs, ...
    [], ...
    export_data_txt, ...
```

```
    export_report, ...
    export_data_mat, ...
    display_data, ...
    [], ...
    [], ...
    [], ...
    [], ...
    status_bar);
```

You may now want to inspect the report file to see that GrafLab indeed used the correct "GM" and "R" values from the "gfc" file.

```
fprintf("The name of the report file is ""%s_Report.txt"".\n", out_path);
```