HOWTO g04: Methods to compute the Legendre functions

You will learn about the pros and cons of the three supported methods to compute Legendre functions.

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

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

Brief summary

The standard forward column method (SFCM, "fnALFs = 1") can be used up to degree 1800.

The modified forward column method (MFCM, "fnALFs = 2") can be used up to degree 2700.

The extended range arithmetic approach (ERA, "fnALFs = 3") can be used up to almost an arbitrary degree.

Let's do some grid-wise and point-wise syntheses with all three methods to see which one should be preferred in case more than one method is stable within your range of harmonic degrees.

Benchmarks for grid-wise computations

Define all GrafLab input parameters except the method to compute the Legendre functions.

```
GM                = 3986004.415E+8;
R                 = 6378136.3;
nmin              = 0;
nmax              = 360;
ellipsoid         = 1;
GGM_path          = '../data/input/EGM96.mat';
crd               = 0;
point_type        = 0;
lat_grd_min       = -90.0;
lat_grd_step      =   0.1;
lat_grd_max       =  90.0;
lon_grd_min       =   0.1;
lon_grd_step      = lat_grd_step;
lon_grd_max       = 360.0;
h_grd             =   0.0;
out_path          = '../data/output/howto-g04-grd';
quantity_or_error = 0;
quantity          = 5;
export_data_txt   = 0;
export_report     = 0;
export_data_mat   = 0;
display_data      = 0;
status_bar        = 1;
```

Do the tests by looping over the methods to compute Legendre functions. Note that the same number of points is use for all grid computations.

```
time_grd = zeros(3, 1);
for fnALFs = [1, 2, 3]
```

```
    fprintf("fnALFs method: %d\n", fnALFs);

    tic
    out_grd = GrafLab('OK', ...
        GM, ...
        R, ...
        nmin, ...
        nmax, ...
        ellipsoid, ...
        GGM_path, ...
        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);
    time_grd(fnALFs) = toc;

end
```

Benchmarks for point-wise computations

Now let's define the scattered points

```
[lon_sctr, lat_sctr] = meshgrid(0.0:5.0:360.0, ...
                                -90.0:5.0:90.0);
lat_sctr = lat_sctr(:);
lon_sctr = lon_sctr(:);
h_sctr   = zeros(length(lat_sctr), 1);
```

Update some GrafLab input parameters.

```
point_type = 2;  % Synthesis at scattered points defined by MATLAB variables
out_path   = '../data/output/howto-g04-sctr';
```

Do the tests by looping over the methods to compute Legendre functions. Again, the same number of scattered points is use for all point-wise computations.

```
time_sctr = zeros(3, 1);
for fnALFs = [1, 2, 3]

    fprintf("fnALFs method: %d\n", fnALFs);


    % Do the synthesis
    tic
    out_grd = GrafLab('OK', ...
        GM, ...
        R, ...
        nmin, ...
        nmax, ...
        ellipsoid, ...
        GGM_path, ...
        crd, ...
        point_type, ...
        [], ...
        [], ...
        [], ...
        [], ...
        [], ...
        [], ...
        [], ...
        [], ...
        lat_sctr, ...
        lon_sctr, ...
        h_sctr, ...
        out_path, ...
        quantity_or_error, ...
        quantity, ...
        fnALFs, ...
        [], ...
        export_data_txt, ...
        export_report, ...
        export_data_mat, ...
        display_data, ...
        [], ...
        [], ...
        [], ...
        [], ...
        status_bar);
    time_sctr(fnALFs) = toc;

end
```

Results of the benchmarks

Grid-wise computation times in sec:

```
fprintf("SFCM: %0.1f\n", time_grd(1));
fprintf("MFCM: %0.1f\n", time_grd(2));
fprintf(" ERA: %0.1f\n", time_grd(3));
```

Point-wise computation times in sec:

```
fprintf("SFCM: %0.1f\n", time_sctr(1));
fprintf("MFCM: %0.1f\n", time_sctr(2));
fprintf(" ERA: %0.1f\n", time_sctr(3));
```

Conclusions

For grid computations up to degree 1800, use SFCM. Beyond that degree, use ERA. Avoid using MFCM.

For point-wise synthesis up to degree 1800, use SFCM or MFCM. For maximum degrees from 1801 to 2700, use MFCM. Beyond degree 2700, use ERA.