NEROGRAV

New Refined Observations of Climate Change from Spaceborne Gravity Missions

International Spring School

10.-14. March 2025

Marius Schlaak, Mail: marius.schlaak@tum.de   Grigorios Kalimeris, Mail: greg.kalimeris@tum.de

Lab 2: Filtering/De-striping

Purpose

In this Lab the technique of gravity field filtering in the spectral domain is presented. For this purpose, a synthetically generated gravity field signal superimposed with typical aliasing effects (striping) is given which shall then be treated with different types of filters. In order to understand how the various filters affect the gravity solution an error-free reference is provided as well. Finally, the filtering will be applied to two real GRACE-based monthly solutions.

Material

- Data
  - Static gravity field model *ITSG-Grace2018s* in ICGEM format
  - Synthetic monthly gravity solutions
    - synthetic_n90_signal (true signal, ICGEM format)
    - synthetic_n90_signal_noise (true signal with correlated noise, ICGEM format)
  - Real monthly gravity field solutions
    - GSM-2_200404_0027 (April 2004, good month)
    - GSM-2_200409_0030 (September 2004, bad month)
- Python functions (found in *SpringSchoolLib12*)
  - `readicgem`, `cs_format`, `shs`, `triplot`
  - Gaussian filter
    - `filter_sh_gaussian`
    - `gaussian`
  - Swenson-Wahr filter
    - `filter_sh__swenson`
    - `cs_format`

- DDK filter
  - `cs2ddkformat`
  - `filter_sh__ddk`
  - `ddk2csformat`
  - `cs_format`
  - `read_BIN`
  - DDK filter coefficient files

Tasks

1. The file *synthetic_n90_signal* contains a certain "true" monthly variation signal, while in model *synthetic_n90_signal_noise* the same signal is superimposed by correlated noise similar to what can be found in real GRACE data. Import both models and visualize them in terms of global EWH grids. Further, visualize the corresponding SH coefficients with `triplot` and select a reasonable colorbar range. Interpret your results.

2. Apply different filters (or combinations of filters) to the noise-polluted model, e.g.

   - Gaussian filter (experiment with different filter radii, e.g. 150-800 km)
   - Swenson-Wahr decorrelation filter
   - DDK filter (experiment with the different strengths 1 [strongest] to 8 [weakest])

   Compute global EWH grids from the filtered difference model and compare them to the true signal. Plot the corresponding SH coefficient triangles as well. Discuss which filter resp. combination of filters yields the best result.

3. Import the gravity field models of April (*GSM-2_200404_0027*) and September 2004 (*GSM-2_200409_0030*) derived from GRACE observations and reduce the static gravity field ITSG-Grace2018s from them. Visualise the residual signal in terms of EWH.

4. Apply different filters (cf. task 2) to both residual fields. In your opinion, which filter (combination) yields the best result?

Python functions:

<table>
<tr><th colspan="2">function [scs, ncs, header, scst, ncst] = readicgem(filename)</th></tr>
<tr><td colspan="2">Reads potential coefficients in ICGEM-format from ASCII file</td></tr>
<tr><td>Input</td><td>• <code>filename</code>: full path and file name [string]</td></tr>
<tr><td>Output</td><td>• <code>scs</code>: potential coefficients in cs-format; size [n,n]
• <code>ncs</code>: formal errors of potential coefficients in cs-format (if available); size [n,n]
• <code>header</code>: structure containing header information of the ICGEM file
• <code>scst</code>: dot-coefficients in cs-format (if available); array size [n,n]
• <code>ncst</code>: formal errors of dot-coefficients in cs-format (if available); size [n,n]</td></tr>
<tr><td>Requires</td><td>• —</td></tr>
</table>

<table>
<tr><th colspan="2">function global_grid = shs(gco, fun, colat, lon, GM, ae, alt)</th></tr>
<tr><td colspan="2">Computes a spherical harmonic synthesis of a gravity functional on a global grid</td></tr>
<tr><td>Input</td><td>• <code>gco</code>: disturbing potential coefficients given in cs-format; size [n,n]
• <code>func</code>: gravity functional to be computed; list or array
– <code>1</code>: geoid heights $[m]$
– <code>2</code>: gravity anomaly $[mGal] = [1^{-5}\ m/s^2]$
– <code>3</code>: vertical gravity gradient $[E] = [1^{-9}\ m/s^2/m]$
– <code>4</code>: total water storage $[mm\ EWH] = [kg/m^2]$
– <code>5</code>: no dimensioning
– <code>6</code>: gravity disturbance $[mGal] = [1^{-5}\ m/s^2]$
– <code>7</code>: pressure $[Pa]$
– <code>8</code>: vertical crustal deformation $[m]$
• <code>colat</code>: co-latitude vector for global grid
• <code>lon</code>: longitude vector for global grid
• <code>ae</code>: radius resp. semi-major axis of Earth $[m]$
• <code>alt</code>: altitude above earth surface for computation of synthesis $[m]$
• <code>kwargs</code>:
– <code>lmax</code>: maximum degree of expansion
– <code>GM</code>: gravity constant times Earth mass $[m^3/s^2]$</td></tr>
<tr><td>Output</td><td>• <code>global_grid</code>: global grid of the computed functional</td></tr>
<tr><td>Requires</td><td>• <code>legnorm</code>
• <code>loadlove_farrell</code></td></tr>
</table>

function [c, s] = cs_format(cs, s)	
Transforms coefficients in cs-format into sc-format or separate c/s matrices and vice versa. cs-format: $\|C\backslash S\|$ sc-format: $\|/SC\backslash\|$ c, s separate: $\|C\backslash\|, \|S\backslash\|$	
Input	• `cs`: if only input, then – size [n,n] implies input in cs-format – size [n,2n] implies input in sc-format • `s`: if specified, then cs are c-coefficients while s are s-coefficients, both are of size [n,n]
Output	• `c`: if only output, then – cs-format, i.e. size [n,n] if only input is cs of size [n,2n] – cs-format, i.e. size [n,n] if two inputs are specified – sc-format, i.e. size [n,2n] if only input is cs of size [n,n] • `s`: if specified, c contains c-coefficients and s contains s-coefficients, both are of size [n,n]
Requires	• —

function [fig] = triplot(scs, nmax)	
Plots a SH coefficient triangle (logarithmic scale)	
Input	• `scs`: potential coefficients in cs-format; size [n,n] • `nmax`: maximum harmonic degree
Output	• `fig`: figure handle
Requires	• —

How to: Gaussian filter

```
cs_fltr = filter_sh_gaussian(field, radius_filter)
```

function cs_fltr = filter_sh_gaussian(field, radius_filter)	
Applies Gaussian smoothing to spherical harmonic coefficients	
Input	• `field`: spherical harmonic coefficients in cs-format, size [n,n] • `radius_filter`: radius of Gaussian bell [km]
Output	• `scs`: filtered spherical harmonic coefficients in cs-format; size [n,n]
Requires	• `gaussian`

How to: Swenson-Wahr filter

```
scnew = filter_sh_swenson(sc)
```

function cs_fltr = filter_sh_swenson(field)	
Destriping in spectral domain based on Swenson & Wahr (2006)	
Input	• `sc`: spherical harmonic coefficients in either sc- or cs-format; size [n,2n] if sc-format, size [n,n] if cs-format
Output	• `sc_new`: filtered spherical harmonic coefficients in sc-format; size [n,2n]
Requires	• `cs_format`

How to: DDK filter

```
shc_ddkformat = cs2ddkformat(cs)
dataDDK = filter_sh_ddk(x,shc_ddkformat)
scs_filt = ddk2csformat(dataDDK)
```

function shc_ddkformat = cs2ddkformat(cs)	
Transforms spherical harmonic coefficients in cs-format to DDK-format	
Input	• `cs`: spherical harmonic coefficients in cs-format; size [n,n]
Output	• `shc_ddkformat`: structure of spherical harmonic coefficients in DDK-format
Requires	• —

function dataDDK = filter_sh_ddk(number, data)	
Performs DDK filtering on spherical harmonic coefficients in DDK-format	
Input	• `number`: order of DDK filter, values 1-8 (strongest=1, weakest=8) • `data`: structure file of spherical harmonic coefficients in DDK-format
Output	• `dataDDK`: structure of filtered spherical harmonic coefficients in DDK-format
Requires	• `read_BIN`

function cs = ddk2csformat(shc_ddkformat)	
Transforms spherical harmonic coefficients in DDK-format to cs-format	
Input	• `shc_ddkformat`: structure of spherical harmonic coefficients in DDK-format
Output	• `cs`: spherical harmonic coefficients in cs-format; size [n,n]
Requires	• `cs_format`