1 GRACE and GRACE-FO Gradiometer Mode Toolkit

We produce the results presented in Chapter 3, 4, and 5 using our GRACE and GRACE-FO Gradiometer Mode Toolkit (GGMT). In the research motivation of this thesis, we identified the need for a user-friendly and usable software package to produce gradients using the gradiometer mode (GM). Before this contribution, there has been no available software package for the GM. In this chapter, we provide a very-brief overview of the GGMT.

1.1 Software Manual Overview

The routines are hosted in an online repository, and the full manual is in APPENDIX A. The routines include detailed in-line comments with explanations. In addition to supporting GM, the GGMT offers a great foundation to support research objectives outside the scope of the work presented herein. Specifically, the GGMT is the first and only open-source GRACE and GRACE-FO suite of processing tools for Level 1A and 1B data products. The routines we have developed for, e.g., data product coordinate transformations, accelerometer calibration, and multi-resolution analyses (MRA), are general-purpose routines which can be used to process GRACE (and GOCE) gravity space mission data products for a wide variety of Earth and space science research efforts.

The format of the software manual in APPENDIX A follows a similar style as the "Algorithm Theoretical Basis Document for GRACE Level-1B Data ProcessingV1.2" document (Wu et al 2006). The GGMT is organized into three main processing modules: (1) **GGM1B_compute;** (2) **GGM2B_grid**; and (3) **GGM3B_visualize.** Each of these executable modules are façade modules. We use façade pattern to provide abstraction (encapsulation) for the more complex underlying code in order to improve code readability and ensure code reusability. The relevant data products for GGMT must be retrieved from NASA's Physical Oceanography Distributed Active Archive Center (PO.DAAC) via podaac.jpl.nasa.gov.

1.1.1 GGM1B_compute

The **GGM1B_compute** module calculates the gravitational gradient tensor (GGT) measurements in the leading Science Reference Frame (SRF); and provides GRACE GGT measurements with Global Position System (GPS) time-tags, geodetic (latitude/longitude) coordinates, data quality

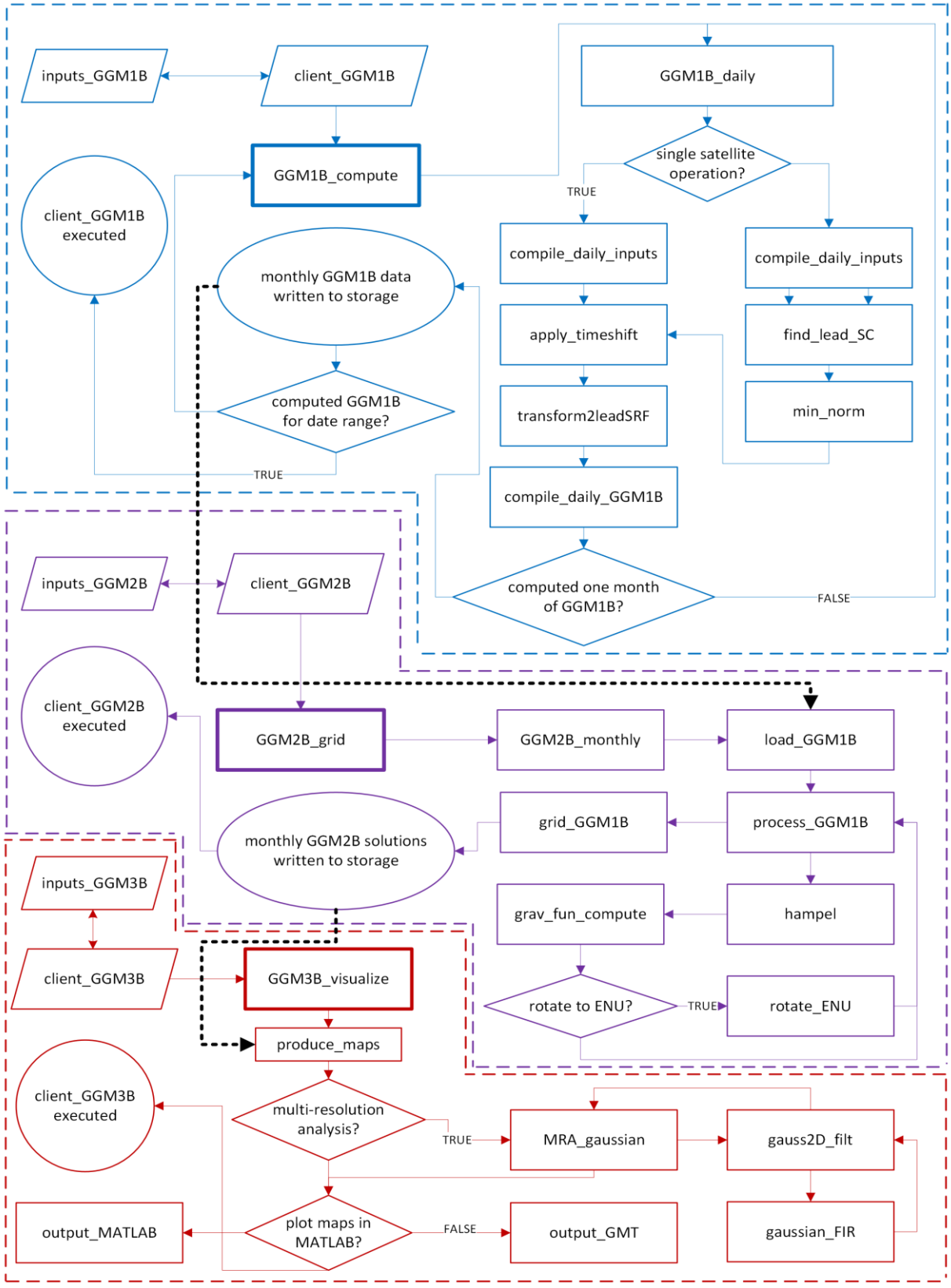
flags, and satellite vector separations at closest approach (in leading SRF). This is the GGM1B data product.

1.1.2 GGM2B_grid

The **GGM2B_grid** module computes the GGT in the GM reference frame (GRF); grids the GGT time-series; and then computes the average gradients in each grid cell. Optionally, it flags, detects, and removes outliers; applies additional filtering in time domain; separates ascending/descending track solutions; and references gradient solutions to the local East-North-Up (ENU) reference frame. This is the GGM2B data product.

1.1.3 GGM3B_visualize

The **GGM3B_visualize** module interpolates sparsely located values using the inverse distance as weights, if a grid cell is empty from **GGM2B_grid**; and then provides the GGT solutions in map or text file format for visualization in Generic Mapping Tools (GMT). Optionally, it band-pass filters/performs a multi-resolution analysis (MRA). This is the GGM3B data product.



Flow diagram of the GRACE and GRACE-FO Gradiometer Mode Toolkit (GGMT). Blue is GGM1B_compute; purple is GGM2B_grid; and red is GGM3B_visualize.

2 SOFTWARE PACKAGE: GRACE AND GRACE-FO GRADIOMETER MODE TOOLKIT

PROCESSING FLOW DOCUMENTATION

2.1 GGM1B_compute

The GGM1B data product is produced using five processing modules in sequence: **compile_daily_inputs, find_lead_SC, apply_timeshift, transform2leadSRF, compile_daily_GGM1B**. Each of these modules are described in the following sub-sections. The five modules are called in sequence in the **GGM1B_daily** module. For single-satellite GM (SS-GM), **find_lead_SC** is not used. Each of the modules depend on low-level routines, and we refer to the low-level routines (aka foundational blocks) as toolkit dependencies.

2.1.1 GGM1B_inputs

The processing parameters for **GGM1B_compute** (and sub-modules) is defined in the **GGM1B_inputs** object. Parse the object into to **GGM1B_compute** to generate the GGM1B data product.

Required and Optional Object Parameters:

- Path of Level 1B GRACE data products - required
- Current directory – required
- GGM1B path out (to write .MAT files) - required
- GRACE-ID – required
- Compute Start Date – required
- Compute End Date – required
- Accelerometer data product selection – required
    - See **compile_daily_inputs** for more information.
- Gradiometer Mode Configuration (i.e., single-satellite or dual-satellite) – required
- Accelerometer filtering type – optional

    - If selected, the filtering routine requires filter shape, filter cut-off(s), and number of filtering stages as input. See **gaussian_FIR.**
- Path of Level 1A GRACE-FO data products - optional
- Debug Mode – optional
    - Debug mode does not run in parallel processing mode, so the user can use the MATLAB debugger tool.

2.1.2 GGM1B_compute

GGM1B_compute calls **GGM1B_daily** for every day in the range (inclusive) of compute start date and compute end date. Every month of **GGM1B** data product generation is computed in parallel using MATLAB's parallel computing toolbox. After one month has been completed, the total outputs of **GGM1B_daily** for the entire month are returned to **GGM1B_compute** and then output to a .MAT file. An associated file specifying the input parameters for **GGM1B_compute** is also written out to storage. See **GGM1B_inputs**. After these two files are written out, the next month will begin computation until the processing date range for GGM1B data product generation has reached its upper temporal limit.

2.1.3 GGM1B_daily

Calls five processing modules in sequence: **compile_daily_inputs, find_lead_SC, apply_timeshift, transform2leadSRF, compile_daily_GGM1B**, with additional error checks.

Inputs:

1. Input structure. See **GGM1B_inputs.**

Outputs:

1. Daily GGM1B data product in MATLAB data table format. See **compile_daily_GGM1B**.

Dependencies:

compile_daily_inputs, find_lead_SC, apply_timeshift, transform2lead, compile_daily_GGM1B.

Algorithm:

1. SS-GM: Calls **compile_daily_inputs** for the requested date and GRACE ID.

2. DS-GM:
    a. Calls **compile_daily_inputs** for GRACE-A and GRACE-B, or GRACE-C and GRACE-D for the requested date.
    b. Determines the leading spacecraft ID along the flight trajectory using **find_lead_SC**.
3. Pipelines outputs from step (2) for DS-GM/step (1) for SS-GM into **apply_timeshift**.
4. Pipelines outputs from step (3) into **transform2leadSRF** and then into **compile_daily_GGM1B**.

2.1.4 compile_daily_inputs

Compiles and pre-processes GRACE and GRACE-FO data products to compute the GGM1B data product.

Inputs:

1. Input structure. See **GGM1B_inputs**.

Outputs:

1. Accelerometer (ACC) data product in the Science Reference Frame (SRF).
2. Position coordinates in Inertial Reference Frame (IRF) interpolated to the ACC data product time -tags.
3. SCA1B (rotational quaternions) data product interpolated to ACC data product time-tags.

Dependencies:

read_ACC, get_ACCinSRFfromGPS, read_ACT1A_thrustFREE, gaussian_FIR, read_ACT1B, read_ACC1B, truncate_data2pad, ACC_upsample, GGM1B_inputs, read_GNI1B_IRF, read_GNV1B_ITRF, ITRFtoIRF, interp_spline, read_SCA1B, flip_quats, get_GRACE_coord, find_SCA1B_gaps, avg_sample_rate, timeOBC2GPS, read_CLK1B, read_TIM1B, find_grace_header, find_gracefo_header.

Algorithm:

1. Loads the ACC linear accelerations based on user selection. See **read_ACC.**
    a. For GRACE, the ACC data product is either ACC1B accelerations with calibration factors applied, or precise orbit determination (POD) solution derived accelerations in the selected GRACE-ID SRF. Calibration factors for the ACC1B data product are determined by McCullough et at., (2019) and are applied using **calib_ACC**.

b. For GRACE-FO, the ACC data product is either ACT1A, ACT1B, 'thruster-free' ACT1A , or POD solution derived accelerations in the selected GRACE-FO ID SRF. ACT1A data products are transformed from the accelerometer frame (AF) to SRF and are also mapped from onboard computer time (OBC) to GPS time using **timeOBC2GPS**, **read_CL1K1B,** and **read_TIM1B**. These coordinate transformations are done internally in **read_ACT1A.**

c. POD derived accelerations are computed using **get_ACCinSRFfromGPS.**

2. Optionally, Performs zero-phase (forward/background) filtering to the ACC data product. **See gaussian_FIR**.
3. Interpolates the ACC data products to 10 Hz using cubic splines.
4. Loads position coordinates in IRF from the GNI1B data product using **read_GNI1B_IRF**, and then interpolates to time-tags of ACC data using cubic splines.
    a. For GRACE, the GNI1B data product is not released. The positions are only given in the ITRF in the GNV1B data product. The **read_GNI1B_IRF** routine calls **read_GNV1B_ITRF** to load the positions from the GNV1B data product, and then transforms the ITRF positions to IRF using **ITRFtoIRF** internally.
5. Loads rotational quaternions from the SCA1B data product. See **read_SCA1B.**
6. Corrects the quaternions, and then interpolates to ACC data using splines.
    a. The quaternion correction flips signs of the 4 quaternions, if needed, to maintain continuity in time to ensure reliable interpolation. See **flip_quats**. The correction is called internally in **read_SCA1B**.

2.1.5 find_lead_SC

Determines the leading and trailing satellite along the flight trajectory.

Inputs:

1. IRF position coordinates for GRACE-A or GRACE-C.
2. Rotational quaternions from the SCA1B data product interpolated to IRF positions.

Outputs:

1. The ID of the leading satellite
2. The ID of the trailing satellite.

Dependencies:

IRFtoSRF.

Algorithm

1. Transforms the positions for all epochs (input 1) in IRF to the SRF using rotational quaternions from the first epoch only (input 2).
2. Applies the discrete forward derivative operator to transformed positions.
    a. The result of step (2) is negative if GRACE-A or GRACE-C is the leading satellite and is positive if GRACE-B or GRACE-D is leading. This is because the SRF has the X_{SRF} pointing towards the other satellite. If the result from step 3) is neither all negative nor all positive (i.e., mixed), an error is thrown.
3. Returns the leading and trailing satellite ID.

2.1.6 apply_timeshift

Applies time-shifts to move the trailing satellite to be as close as possible into the air-space of the leading satellite.

Inputs:

1. Input structure. See **GGM1B_inputs**.
2. Outputs from **compile_daily_inputs**.

Outputs:

1. Time-shifted trailing and leading satellite input (2).

Dependencies:

min_norm.

Algorithm:

1. For SS-GM:
    a. Applies a relative one index offset (shift) to input (2) in order to assign the 'time-shifted trailing' and 'leading' satellites. The leading and trailing satellite are the same satellite.
2. For DS-GM:
    a. Computes time-shifts to move the trailing satellite as close as possible to the leading satellite air-space, and then uses then uses evaluated time-shifts to index input (2) to assign the 'time-shifted trailing satellite'. Computes time-shifts by minimizing the L2 (Euclidean) norm of the 3D position separation in IRF between the trailing satellite positions in the interval $[t, t + 50s]$ and the leading satellite position at epoch t. Calculates time-shifts for every epoch of the

leading satellite positions from input (2).

2.1.7 transform2leadSRF

Transforms all measurements to the leading satellite SRF.

Inputs:

1. Outputs (1) from **apply_timeshift**.

Outputs:

1. Outputs (1) from **apply_timeshift** in the leading SRF.

Dependencies:

SRFtoIRF, **IRFtoSRF.**

Algorithm:

1. Transforms the time-shifted trailing satellite ACC data from SRF to IRF using SCA1B data of the trailing satellite. SCA1B data is interpolated to the time-tags of ACC data. See **SRFtoIRF**.
2. Transforms time-shifted trailing satellite ACC data from IRF to SRF using SCA1B data of the leading satellite. SCA1B data is interpolated to the time-tags of the leading satellite for which the trailing satellite is time-shifted towards (i.e., in the airspace), determined by the **apply_timeshift** module.
3. Transforms time-shifted trailing satellite and leading satellite position coordinates in IRF to leading satellite SRF. See **IRFtoSRF**.

2.1.8 compile_daily_GGM1B

Compiles daily GGM1B data product in a MATLAB data table format.

Inputs:

1. Outputs (1) from **transform2leadSRF**.

Outputs:

1. Daily GGM1B data product.

Dependencies:

find_SCA1B_gaps, get_GRACE_coord

Algorithm:

1. Determines geodetic ITRF coordinates (latitude and longitude) of the leading satellite. See **get_GRACE_coord**, a wrapper for **ecef2geodetic**.
2. Returns quality warnings when interpolated SCA1B data is interpolated using data that has gaps larger than 5 seconds.
    a. Precise attitude determination given by the SCA1B data product is critical for the GM, and therefore gappy SCA1B data should probably not be included in gradient measurements. See **find_SCA1B_gaps**.

2.2 GGM2B_grid

The generation of the GGM2B data product (gridded gradient measurements) from the GGM1B data product involves three processing modules: **load_GGM1B**, **process_GGM1B,** **grid_GGM1B**. These 3 modules are called in sequence and wrapped by **GGM2B_monthly** which are then encapsulated by the façade **GGM2B_grid**.

2.2.1 GGM2B_inputs

The processing parameters for **GGM2B_compute** (and sub-modules) is defined in the **GGM2B_inputs** object. Parse the object into to **GGM2B_compute** to generate the GGM2B data product.

Required and Optional Object Parameters:

- Working directory – required
- Compute Start Date – required
- Compute End Date – required
- Directory of GGM1B data – required
- Rotation of GGM2B in SRF to local East-North-Up (ENU) frame – optional
- Differential accelerations filtering type – optional

2.2.2 GGM2B_grid

GGM2B_grid calls **GGM2B_monthly** for every day in range (inclusive) of compute start date and compute end date. Every month of GGM1B data product, processed using **GGM1B_compute**, is loaded and gridded using **GGM2B_monthly** until all months in the compute date range have

been processed. Once this has been completed, the total outputs of all gridded gradient measurements are returned to **GGM2B_grid** and then output to a .MAT file. An associated file specifying the input parameters for **GGM2B_grid** is written out. See **GGM2B_inputs**

2.2.3 GGM2B_monthly

Calls three processing modules in sequence: **load_GGM1B**, **process_GGM1B, grid_GGM1B**, with additional error checks.

Inputs:

1. Input structure. See **GGM2B_inputs**.

Outputs:

1. Monthly GGM2B monthly solutions. See **grid_GGM1B**.

Dependencies:

load_GGM1B, process_GGM1B, grid_GGM1B.

Algorithm:

1. Pipelines results from **load_GGM1B** to **process_GGM1B** to **grid_GGM1B**.

2.2.4 load_GGM1B

Reads in the outputs of **GGM1B_compute**. There is no processing performed in this utility.

2.2.5 process_GGM1B

Performs outlier removal; computes and filters gravity gradient derived from GGM1B data product in GMRF; rotates to an ENU frame if selected; removes flagged data; and also segregates GGM1B data into ascending track and descending tracks.

2.2.5.1 Inputs:

1. Input structure. See **GGM2B_inputs**,
2. Outputs from **load_GGM1B**.

2.2.5.2 Outputs:

1. Processed gravitational gradient tensor (GGT) in ascending track, descending track, and all track solutions.

2.2.5.3 Dependencies:

grav_fun_compute, rotate_ENU, ascending_tracks, descending_tracks, filtBYsegments, hampel (MATLAB).

2.2.5.4 Algorithm:

1. Calculates the differential mode positions and accelerations referenced in the leading SRF.
2. Applies a Hampel outlier identification algorithm (moving median) to the differential positions to identify and remove when the GRACE satellites operate in nadir-pointing mode. See **hampel** (MATLAB).
3. Filters the differential positions and optionally filters the differential accelerations. The differential positions are low-pass filtered using a corner frequency of $10^{3.5}$ Hz.
    a. The filtering processes are done in segments. A new segment of the input series is created when there is a data gap greater than 500 seconds. For every gap in the input series, the filtering procedure is restarted. If there are no gaps larger than 500 seconds, the filtering procedure assumes there is only one segment. See **filtBYsegments**. **filtBYsegments** calls **gaussian_FIR**.
    b. Differential positions less than 0.1 m are flagged because differential positions below this threshold significantly modulate the gravitational gradient signal.

4. Calculates the GGT in GMRF using the differential mode positions and accelerations.
5. Optionally transforms the output the GGT in GMRF to the local ENU frame. See **rotate_ENU**.
    a. Calculates the azimuth of the satellite ground track. See **azimuth** (MATLAB). Uses the calculated azimuths to create rotation matrices about the Z-axis of GMRF. The transformation is about the Z-axis of GMRF and aligns the X-axis of the GMRF to North, and Y-axis of the GMRF to East. The Z-axis is kept zenith pointing.
6. Removes flagged GGM1B data and isolates into ascending track and descending track solutions.

2.2.6 grid_GGM1B

Grids the GGT time-series.

2.2.6.1 Inputs:

1. Input structure. See **GGM2B_inputs**
2. Outputs from **process_GGM1B**.

2.2.6.2 Outputs:

1. Gridded global GGT in (1) ascending tracks; (2) descending tracks; and (3) all tracks.

2.2.6.3 Dependencies:

gridbin (Chad, A)

2.2.6.4 Algorithm:

1. Uses **gridbin** (Chad, A) to grid the GGT time-series with one-degree bin spacing and then computes the average value in each bin.

2.3 GGM3B_visualize

The generation of the GGM3B solution from GGM2B data involves one processing module: **produce_map**. This module is encapsulated by **GGM3B_visualize**.

2.3.1 GGM3B_inputs

The processing parameters for **GGM3B_visualize** (and sub-modules) is defined in the **GGM3B_inputs** object. Parse the object into to **GGM3B_visualize** to generate the GGM3B data product.

Required and Optional Object Parameters:

- Working directory – required
- Directory of GGM2B data – required
- Compute Start Date – required
- Compute End Date – required
- Output mode – required
    - 'MATLAB' to produce mapping solutions in MATLAB
    - 'GMT' to produce XYZ text files to be plotted in GMT
- Number of monthly GGM2B solutions to average- required
- 2D filtering wavelength – optional
- Ascending tracks solution only – optional
- Descending track solution only – optional

2.3.2 GGM3B_visualize

GGM3B_visualize calls **produce_map** in a step size equal to the number of monthly GGM2B solutions to average solutions in the range (inclusive) of compute start date and compute end date. For example, if compute start date is 2022/1 and compute end date is 2022/12, and the number of monthly solutions to average is 6 – two sets of 6-month GGM3B solutions would be produced: one from 2022/1 to 2022/6 and another set from 2022/7 to 2022/12. Outputs of **GGM3MB_visualize** are saved in the directory of the GGM2B data under a folder named "plot".

2.3.3 produce_map

Produces gravitational maps using GGM2B data product in MATLAB, or outputs XYZ data to be plotted in GMT.

Inputs:

1. Input structure. See **GGM3B_inputs**.

Outputs:

1. Gravitational map visualized in MATLAB or XYZ data output in text file.

Dependencies:

MRAgaussian, gauss2D_filt, output_MATLAB, output_GMT, filt2 (Chad, A)

Algorithm:

1. Loads computed GGM2B data product and then extracts monthly gradient solutions which correspond to the requested monthly range.
2. If necessary, performs inverse weighting to fill missing gradient grid cells. See **filt2** (Chad, A).
3. Averages monthly GGM2B solutions, if requested.
4. If requested, performs 2D Gaussian low-pass filtering. See **gauss2D_filt**.
5. If requested, performs 2D multiresolution analysis (MRA) to either:
    a. Decomposes gradient solution to 7 levels.
    b. Isolates gradient solution to specific levels of decomposition and then resynthesizes the decomposition. See **MRAgaussian**.
6. Visualizes output(s) from step 5) in MATLAB or returns a text file for GMT plotting. See **output_MATLAB, output_GMT**.