|  |
| --- |
| logo_eost.png  RTEmagicC_Logo_labex_01.png.png  Labex G-EAU-THERMIE Profonde |
| pyGrav |
| User manual |
|  |
| **Basile HECTOR** |
| **09/09/2015** |

|  |
| --- |
| EOST 5 rue René Descartes 67084 Strasbourg cedex France. Contact : basilehector@gmail.com |

Contents

[1. Overview 3](#_Toc429496753)

[2. Quick start 4](#_Toc429496754)

[3. pyGrav functions 5](#_Toc429496755)

[3.1. Loading data 5](#_Toc429496756)

[3.1.1. Loading raw data 5](#_Toc429496757)

[3.1.2. Loading processed data 5](#_Toc429496758)

[3.2. Data corrections 6](#_Toc429496759)

[3.2.1. Earth Tides 6](#_Toc429496760)

[3.2.2. Ocean loading 7](#_Toc429496761)

[3.2.3. Atmosphere 7](#_Toc429496762)

[3.3. Data selection 7](#_Toc429496763)

[3.4. Drift adjustment 8](#_Toc429496764)

[3.5. Compute double differences 8](#_Toc429496765)

[4. Test-case 9](#_Toc429496766)

[4.1. Using pyGrav GUI 9](#_Toc429496767)

[4.2. Using a python script and pyGrav objects 10](#_Toc429496768)

[5. Code structure 12](#_Toc429496769)

[5.1. Object-oriented programming and microgravity data 12](#_Toc429496770)

[5.2. GUI and PyQt 12](#_Toc429496771)

[5.3. pyGrav code 12](#_Toc429496772)

[5.3.1. Data object file 12](#_Toc429496773)

[5.3.2. GUI file 14](#_Toc429496774)

[6. Acquisition protocol 15](#_Toc429496775)

[6.1. Loop geometries and nomenclature 15](#_Toc429496776)

[6.2. Transportation 15](#_Toc429496777)

[6.3. Setting up the instrument 15](#_Toc429496778)

[6.4. Data acquisition 15](#_Toc429496779)

[7. Least-square inversion and error propagation 17](#_Toc429496780)

[8. Installing external programs 19](#_Toc429496781)

[8.1. ETERNA installation 19](#_Toc429496782)

[8.2. MCGRAVI installation (optional) 19](#_Toc429496783)

[9. References 21](#_Toc429496784)

1. Overview

pyGrav (Hector and Hinderer, 2016) is designed for processing microgravity data and is mainly dedicated to time-lapse gravity studies. The user may choose between a graphical user interface (GUI) or classical Python scripting by calling pyGrav functions for processing the data: tide and atmospheric corrections, data selection, and drift adjustment. The code is open-source, written in Python 2.7 and following an object-oriented scheme, which allows fast implementation of new features/options. Currently only Scintrex CG5 ASCII file format can be read but any other format may be easily added to pyGrav reading routines. Key points are summarized below:

* Provide a single interface for different processing steps (corrections, data selection, drift adjustment, double differences…), instead of worrying about managing different specific programs with associated input/output file formats.
* Provide a unique and easy to use interface for data selection, with both graphical and table display, and automatic selection criteria to favor faster processing.
* Written in open-source Python 2.7 language, and may be used to wrap other programs (such as MCGRAVI for network compensation, or ETERNA PREDICT for synthetic tides computation).
* Written in object-oriented style, appropriate for microgravity data for which objects with specific properties/functions are clearly identified, and follow an intuitive hierarchy (a gravity **campaign**, with several **surveys**, each of them consisting of different **loops**, composed of a few **stations**).
* The code structure (GUI also coded in object-oriented style, using PyQt) allows an easy implementation of additional functions (such as I/O format or interfacing other programs).

This user manual is organized as follow:

* A **“quick start”** is proposed in **section 2** to get pyGrav to run
* pyGrav **functions** are detailed in **section 3**
* **Section 4** lists the steps for running the provided **test-case**. This is the best way for a go-through to pyGrav
* The **code structure** is quickly presented in **section 5**, for who is interested into modifying the python scripts
* An **acquisition protocol** is proposed in **section 6**, based on the author’s experience and to exploit at best the current capabilities of pyGrav.
* **Section 7** reminds the **least-square adjustment equations** and error propagation
* **Section 8** provides **installation procedure for external programs** which can be used within pyGrav (ETERNA and MCGRAVI).

1. Quick start

pyGrav is a set of four python scripts with a .py extension: *pyGrav\_main.py*, *data\_objects.py*, *model\_Classes\_tree\_and\_table.py*, and *synthetic\_tides.py*. The core functions for handling and processing microgravity data are defined in *data\_objects.py*, and the Graphical User Interface (GUI) is defined in *pyGrav\_main.py*. The third script, *model\_Classes\_tree\_and\_table.py*, only hosts functions for displaying and interacting with the data in the data selection step of pyGrav while the fourth script, *synthetic\_tides.py* is only a module which hosts functions for the calculation of synthetic tides. **To launch the pyGrav user interface, run the *pyGrav\_main.py* script.** A good way to run and/or edit a python script is to use Spyder, a visual interface similar to Matlab which provides a script editor and code structure, console… Once in Spyder, open the pyGrav\_main.py script and run it using F5.

* For Windows or Mac users, a good option is to download and install Spyder together with an extensive list of Python modules. Good options are **Anaconda** (<https://store.continuum.io/cshop/anaconda/>) or **Python(x,y)** (<https://code.google.com/p/pythonxy/>).
* For Linux users, Spyder may be obtained from a package manager. It is also possible to directly run the python script with the command “python pyGrav\_main.py” from the source files directory, if python and all required modules are installed.

For a quick overview of pyGrav functioning, follow the test-case tutorial from section 4 of this manual.

1. pyGrav functions

This section is the actual manual for using the program, describing functions and specifications.

The program starts with the ‘start project’ only available option in the file menu. The user is requested to provide an input folder where all input data should be stored, and an output folder which will be used by the program to write output files.

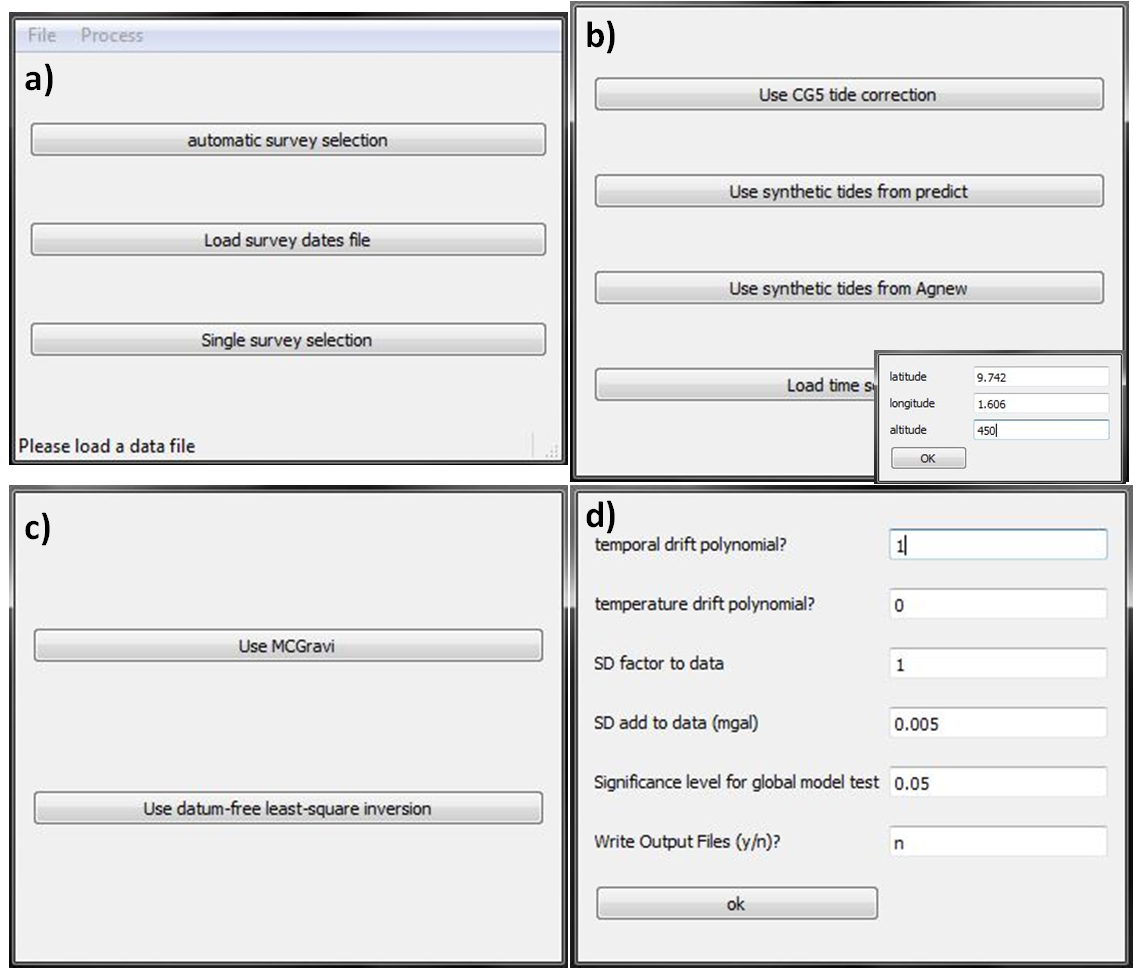


Figure 1 : Example of pyGrav snapshots : a) Loading data screen, b) Tide correction screen, c) network compensation screen and d) options for the network compensation.

* 1. Loading data

Two options are available. Loading “raw data” or “processed data” to continue/modify data processing. When choosing “processed data”, these are already sorted as survey/loop/stations hierarchy, while for “raw data”, they must be sorted. This step therefore not only loads the data, but also stores it following its hierarchy (surveys – loops – stations).

* + 1. Loading raw data

Three options are available for selecting surveys among raw data:

* Automatic survey selection: for simple survey geometries, when the base station is always the same, this option allows to automatically identify different surveys based on a simple time threshold requested by the program: if the time between two consecutive station changes exceeds the threshold, different surveys are considered. If base station and cycling station are the same but have different names, the option 0 should be selected.
* Load from survey start/end date file: this option allows to only read surveys defined between start and end dates given in an input file. The format of such input file is:

2012/07/11 05:17:00 2012/07/11 13:00:00

2012/07/13 05:00:00 2012/07/13 22:00:00

…

* Enter manually start/end date of a single survey (single survey selection)

Then, loops within surveys are identified as follows: each time the base station is encountered, a new loop is started, and the former is ended. In the later drift adjustment procedure, however, nested loops are handled as every loop is part of the same equation system which is inversed using a least-square approach.

* + 1. Loading processed data

This is equivalent as loading a “project”. This allows re-loading already processed data. Data must have been previously saved using ‘save processed data’, which is therefore equivalent as saving a “project”, or arranged in the required way. The user is requested to load a file describing the data hierarchy. The format of such file is as follow:

Directory C:/Users/…/test\_case/output\_data/

Survey: 2013-09-19 nloops: 4 directory: 2013-09-19

Loop: 1 filename: fn111c13.262.txt

Loop: 2 filename: fn211c13.262.txt

Survey: 2013-09-21 nloops: 4 directory: 2013-09-21

Loop: 1 filename: fn111c13.264.txt

Loop: 2 filename: fn211c13.264.txt

Loop: 3 filename: fn311c13.264.txt

Survey: …

Loop: …

‘Survey’ lines describe available surveys, survey names (date of the first measurement), number of loops present in the surveys, and surveys folders names within the root directory. ‘Loop’ lines describe for each line in each survey, the loop name and the loop data file. Data files are alike CGxTool ‘c’ files, with no header and an extra column containing data status (1 or 0 whether the data line is kept for drift adjustment or not).

* 1. Data corrections
     1. Earth Tides

Four options are available for correcting tides:

* Use CG5 tide correction (do nothing): Geographic coordinates must have been entered within the instrument prior to the survey.
* Use synthetic tides from predict: For window users the function PREDICT from the ETERNA program package (Wenzel, 1996) is plugged to the program (see section 8 for the installation) to create a synthetic tide from tidal parameters, which is used as a correction. The choice of the HW95 tidal potential (Hartmann and Wenzel, 1995) is hard-wired in pyGrav, but this is easy to modify, if needed. Tidal parameters are either standard tidal parameters or user’s input as a single file. Geographic coordinates of the survey must be entered.
  + An instance of the predict program (.exe) should be made available under the source code folder and is copied by pyGrav to the output folder where the calculation is made.
  + Standard tidal parameters are read from a file (200D.INI) in the main code folder
  + user input tidal parameter file format should be as follows:

0.023812 0.044652 1.13344 0.5445 MM

0.060132 0.080797 1.12607 -0.1195 MF

0.096423 0.249951 1.14548 0.6833 MTM

…

(frq. start of the bandwidth (cpd) – frq. end of the bdwth (cpd) – amplitude – phase – tide name)

* Use synthetic tides from Agnew: this is a non-harmonic method, such as provided by the fortran codes of (Agnew, 2007, 2012) and later translated into MATLABTM by Cattin et al., (2015) in their GravProcess program. In this case, the correction is a direct computation of the tidal potential from Munk and Cartwright (1966). It is based on an internal ephemeris (for the position of the Moon and the Sun). Survey location must be entered.
* Load time series: if another synthetic tide is available as a time series, it can be loaded and used as a correction. Accepted file format are ETERNA or Tsoft (.TSF) files. If the file extension is not .tsf, it will be considered as an ETERNA file. By default, the data should be stored in the first channel (column).
  + 1. Ocean loading

Ocean loading can be corrected using two different approaches:

* If a tidal analysis can be made on the survey site (for instance thanks to the close location of a superconducting gravimeter), one could proceed to an earth-tide correction using PREDICT from the ETERNA package (see above) and providing tidal parameters from the tidal analysis. This empirical approach integrates both earth tides and ocean loading correction (but may also integrate other environmental effect, such as air pressure, which occur at similar frequencies).
* The ocean loading correction in the pyGrav program is the same as in the GravProcess code (Cattin et al., 2015). It is based on Agnew (2012) and on the MATLABTM transcription from Cattin et al., (2015). Ocean loading coefficients must be loaded from a formatted file originating from the Scherneck's free ocean provider (<http://holt.oso.chalmers.se/loading/>). Parameters are required for the semidiurnal (M2, S2, N2, K2), diurnal (O1, P1, Q1, K1) and long- period (MF, Mm, Ssa) tidal harmonics.
  + 1. Atmosphere

Load a single time series. Accepted file format are ETERNA or Tsoft (.TSF) files. If the file extension is not .tsf, it will be considered as an ETERNA file. By default, the data should be stored in the first channel (column).

* 1. Data selection

This is an original feature of the program. The data hierarchy is visible through a tree-type, a table and graphical representation. Survey, loops, stations, and single measurements can be checked or unchecked whether they should be kept for the drift adjustment process (and final simple difference calculations). This can be done either manually, or following automatic selection procedures based on simple threshold criteria. These currently include

* a threshold on tilts: absolute tilts values higher than the input value are unchecked
* a threshold on gravity standard deviation (SD): SD values higher than the input values are unchecked
* a threshold based on gravity values: absolute gravity values higher than the mean of the three last values + the input threshold value are unchecked
* a criterion on duration: if data duration is different than the input value, they are unchecked. This often happens when the user keeps the current data on the field when stopping CG5 acquisition.

When single threshold are selected, they are applied to the current table only. When the button “apply to all data” is clicked, all input thresholds are used, and for the whole data set.

Another option to rapidly check/uncheck data is to select several rows with the mouse and click either the “check selected” or “uncheck selected” buttons.

Each survey and loops are identified by their first measurement dates. Station numbers are given, and the number in brackets is the repetition number, because some stations are repeated (the base station for instance). Data are organized in chronological order.

Manual selection may be done based on the table values, or graphical displays, depending on the user’s preferences. In the graphical display, unchecked data are displayed in black, while checked data are in blue. For the gravity screen, the average value of selected data is indicated as a blue horizontal line.

After data is selected, the **OK** button should be pressed. This action does not do much, but may be important: it is used to check if some stations have no data selected, in which case they are removed. This may happen using the automatic selection.

* 1. Drift adjustment

Once corrections are applied and data is selected, the drift adjustment can be undertaken. Two options are available. Either the MCGravi (Beilin, 2006) can be launched (if installed, see section 8), or the simple datum-free least-square inversion scheme (Hwang et al., 2002) can be used. The only interest in using MCGravi is when a network compensation is undertaken and several fixed absolute (a priori) values should be maintained (using the option “weighted constraint”). This requires mcgravi.exe to be available. The program writes mcgravi input files in the output directory, and reads mcgravi output files (\*.gra file in mix… folders).

The user may choose parameters to apply to the data prior to the inversion:

* SD factor: a multiplicative factor on observed standard deviation (SD)
* SD\_add: a constant added to each observed SD

Time and temperature drift polynomial degrees should also be indicated.

Simple differences are displayed in the console, and may be saved as a SimpleDifferences.dat file in each survey folder of the output directory by selecting file -> save simple differences.

* 1. Compute double differences

Once simple differences are calculated, double differences may be computed using Process->compute double differences. Then, double differences may be saved in the output folder by selecting file -> save double differences. This will create gravity and SD double difference files in the output directory. Two formats (dates vs stations or stations vs date) are available.

1. Test-case

Two ways of processing a test-case are detailed in this section. The first one uses the pyGrav GUI for a fast and user-friendly handling of microgravity data. The second uses a python script which calls the pyGrav functions. This is most suitable for who is interested into developing pyGrav and adding new functionalities.

* 1. Using pyGrav GUI

A test-case is available in the test\_case directory. This is a set of 4 surveys acquired during the West-African Campaign in a small catchment in North-Benin. Each survey is composed of usually four loops for a total of 13 stations and a base station (station #1).

To process this dataset from the raw data file, proceed as follows:

* Launch pyGrav: execute the python file *pyGrav\_main.py* using python(x,y) for instance in Windows, or by executing “python *pyGrav\_main.py”* in a Linux shell.
* Start a project: select *input\_data/raw* and *output\_data* sub-directories from the *test\_case* directory
* Select **Load raw data** from the **File** menu : Open *Atest\_Raw\_data.txt*
* Select **Load Survey dates file** and load *Atest\_start\_end\_dates.txt*, or alternatively, select a single survey by entering start and end dates (found in the *Atest\_start\_end\_dates.txt* file). Base station number is 1.
* Process tides: Select **Tide correction** from the **Process** menu, and select **Use CG5 tide correction** or alternatively **Use Synthetic tides from predict** (if installed, see section 8): Latitude = 9.742; Longitude = 1.606 ; Altitude = 450. In this case, select **Load Tidal parameters** and load *tide\_param.txt* from the *input\_data* folder. This parameter list has been obtained from a tidal analysis of superconducting gravimeter available on the survey site, and therefore empirically includes oceanic loading. Last option, select **Use Synthetic tides from Agnew**: Latitude = 9.742; Longitude = 1.606 ; Altitude = 450.
* Process ocean loading: select **Ocean Loading correction** from the **Process** menu, and select the file *oceantidal.txt* from the *input\_data* folder. This file has been obtained from <http://holt.oso.chalmers.se/loading/> with Latitude = 9.742; Longitude = 1.606 ; Altitude = 450.
* Data selection: Select **Data selection** from the **Process** menu.
  + Navigate into the tree in the left panel to check time series for some stations, by clicking on a station within a loop. At first, all measurements are checked, and display in blue on the plot windows (right panel).
  + Try to uncheck some data from the table view. Plots are refreshed by clicking again on the station on the tree (left panel), or by clicking **update plots** from the upper panel. Black points appear where data is unchecked. These measurements will not be retained for network compensation. The horizontal blue line on the gravity plot is the average gravity value of checked values.
  + Automatic selection. Fill in following automatic selection criteria and select **apply to all data**:
    - **auto uncheck tilts >** : 5
    - **auto uncheck g >** : 4
    - **auto uncheck SD > :** 20
    - **auto uncheck dur <>**: 60
  + Browse the selection to complete the actual selection (if needed)
  + Push the **OK** button. This action does not do much, but may be important: it is used to check if some stations have no data selected, in which case they are removed. This may happen using the automatic selection.
* Now it is possible to save processed data: under the **File** menu, select **Save processed data**. In this *output\_data* folder, this will create subdirectories which names are survey names, and which contain loop files similar to CGxTool ‘c’ files, with an extra 0-or-1 column to indicate whether the station should be retained or not. The whole data hierarchy is written in the file *gravity\_data\_hierarchy.txt* in the *output\_data* folder. This file can be opened in another project to load the processed dataset (by selecting **load processed data** under the **File** menu).
* Drift correction: select **Drift Adjustment** from the **Process** menu and **Use datum-free least-square inversion**, with the options:
  + - **Temporal drift polynomial ?:** 1
    - **temperature drift polynomial?:** 0
    - **SD factor to data:** 1
    - **SD add to data (mgal):** 0.005
    - **Significance level for global model test:** 0.05 (i.e. 5%)
    - **Write Output Files (y/n)?:** y
* Drift is now adjusted and it is possible to check output files to see whether the drift retrieval is satisfying or not. In each survey subfolder (which has been created if it was not already by the **Save processed data** function), from the *output-data* folder, two files starting with *LSresults* display the simple difference results for the first one and the complete LS procedure (input, outputs and global test) for the second one.
* Save simple differences: select **Save simple differences** from the **File** menu. As with the **Save processed data** function, this will create simple difference files in subfolders of the *output\_data* folder together with a hierarchy file starting by *simple\_diff\_data\_hierarchy…txt*
* Compute double difference: select **Compute double differences** from the **Process** menu. Select the reference survey, the first one for instance. Then choose **Classic double differences.**
* It is now possible to save double differences: select **Save double differences** from the **File** menu. This will create gravity and SD double difference files in the *output\_data* directory. Two formats (dates vs stations or stations vs date) are available.
* An alternative here is to load already processed data (in this case, data already selected), and to compare results of drift adjustments, simple and double differences with those resulting from your own selection. In this case, start by modifying the data path in the first line of the *gravity\_data\_hierarchy.txt* file in the *input\_data/preprocessed* directory. Then select **Load processed data** from the **File** menu, and select the *gravity\_data\_hierarchy.txt* file in the *input\_data/preprocessed* directory. This file is the hierarchy file describing where to find the loop files in each survey directory (as is originated from the **Save processed data** function.
  1. Using a python script and pyGrav objects

Another way to use pyGrav functions (in the data\_objects.py file) is to write a python script and sequentially call these functions. This is particularly suitable if one wants to add a function in the main code and test it without going through the whole GUI programming. Once the function is properly set up, extra GUI buttons/actions may be easily added in the pyGrav\_main.py by adapting existing code.

The example script (*example\_script.py*) can be found in the main\_code directory, and should be executed as a python script. The script’s purpose is to load raw data, extract single surveys based on start and end dates, select data to keep based on simple thresholds, adjust drifts for every surveys and eventually compute both simple and double differences. By commenting/uncommenting some parts of the code, it is also possible to load already selected and organized data.

1. Code structure
   1. Object-oriented programming and microgravity data

This powerful way of programming, for which Python is particularly suited, is based on a collection of definitions of objects (classes) which contain both attributes (or properties), such as a time series, a name, a dictionary of other objects… and methods (or functions/definitions), which describe what the object can do. This programming method is therefore not linear but favors code readability once the programmer is used to such concepts. A method, for instance to write new output formats, can be easily added under the appropriate object, and called from the main program by adding only a few lines and keeping the general source code frame.

This is particularly suited for micro-gravity data, mostly because data can be arranged as structures (objects) following a hierarchical definition (a campaign which includes several surveys, which include several loops, which include several station, which include time series of acquired data). Objects can be physically identified (a campaign, a survey, a loop, a station), as well as associated methods (reading CG5 ascii file should be defined under the broadest object, a campaign; writing mcgravi input file –‘c’ file- for drift adjustment should be defined under the loop object, while adjusting the drift should be called, or based, on a survey object; data selection will mainly concern stations). This logical hierarchy allows storing the data as structures.

* 1. GUI and PyQt

PyQt uses the powerful python programming language and its suitability for object-oriented programming together with the Qt GUI (Graphical User Interface) libraries. PyQt is composed of several modules, such as the fundamental QtCore (for non-gui functionality, such as using files, threads, processes or time) and QtGui (for the graphical components), among which are defined tens to hundreds of classes containing numerous functions and properties.

* 1. pyGrav code

The code is based on object-oriented programming (OOP). It can be seen as two parallel sets of definitions:

* Data storage and manipulation functions: the core of the program, where classes and functions for operations on data are defined. This is the data\_object.py source file.
* GUI functions: Graphical user interface functions that link user’s demands to data operation defined in the previous set of definitions. This is the pyGrav\_main.py source file.

In general, the code is abundantly commented, and each class, subclass, functions and properties are described.

* + 1. Data object file

This module contains the main classes of the program:

* The base class is an object of type ChannelList, which basically contains channel lists such as found in CG5 ascii output files (gravity, tilts, temperature, standard deviation, time...).

Derived classes follow a logical hierarchy, where each 'subclass' is instantiated as an element of a dictionary from the parent class:

* + class Campaign has a dictionary of 'Survey'-types objects
    - class Survey has a dictionary of 'Loop'-types objects
      * class Loop has a dictionary of 'Station'-types objects
        + class Station contains time series for each stations

Each of these objects is derived from a ChannelList object. The instance of the base class 'Campaign' contains all the data set. Each class also have specific properties and populating, writing, handling and processing functions, called from the main program (Figure 1).

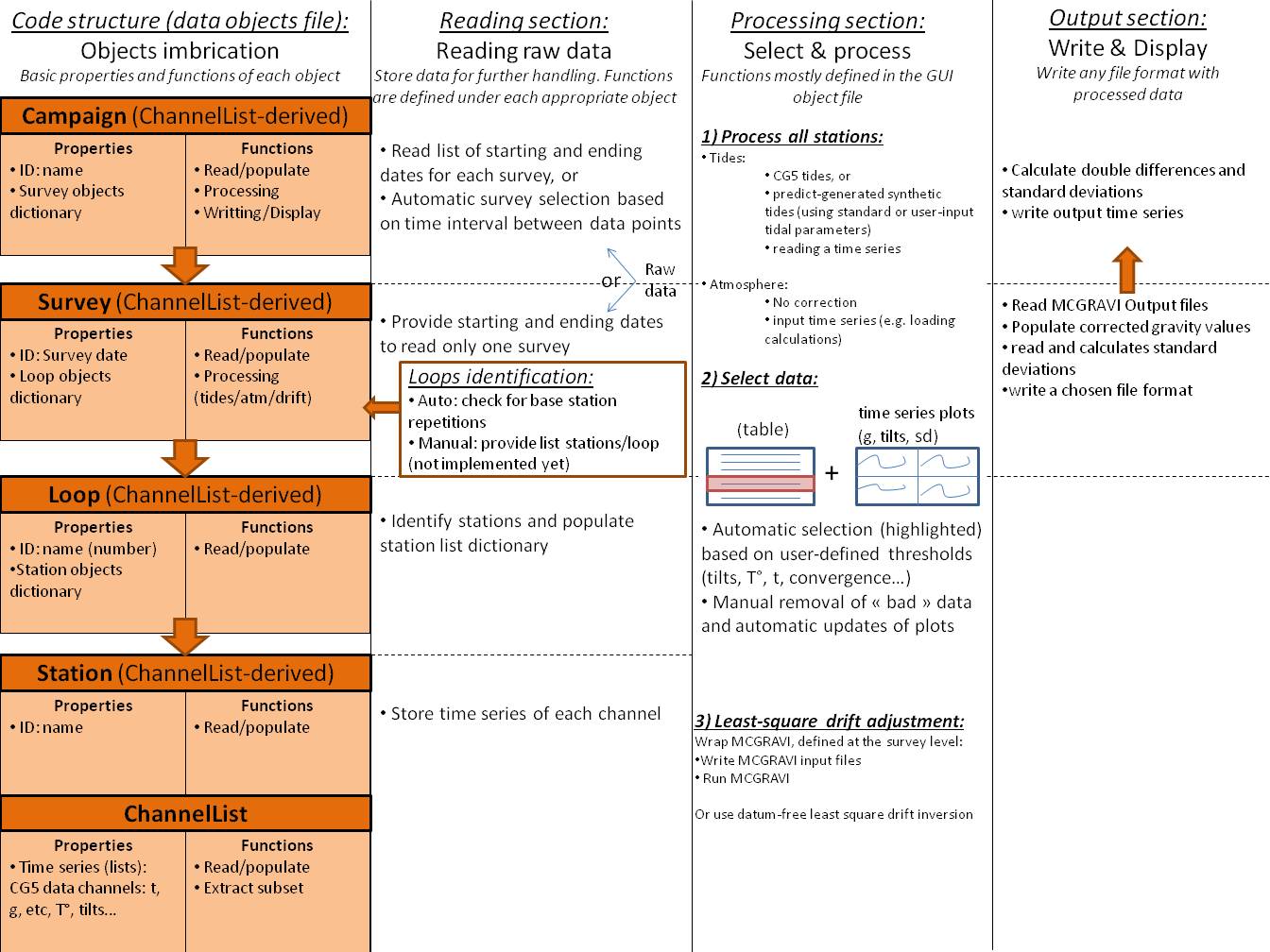


Figure 2 : pyGrav chart and structures (objects) imbrications : in a campaign object there is a dictionary of several surveys. In a Survey object there is a dictionary of several loops. In a Loop object, there is a dictionary of several Stations. Each one of these objects are derived from the ChannelList object (i.e. they contain several time series).

* + 1. GUI file

This is the file which should be executed for running pyGrav. It contains a single class called mainProg, which is a QMainWindow object derived from Qt. The most important mainProg class properties are a Campaign object which contains the whole dataset and both data and output directories. Most functions of the mainProg class link the user interface (defined within the functions) with the processing code written in the data object file, to modify states of the Campaign object (the data set).

**Links:**

A few links for who is interested into modifying the code

Some Python tutorials:

<https://docs.python.org/2/tutorial/> Python tutorial for version 2.7

<http://zetcode.com/lang/python/>

<http://marvin.cs.uidaho.edu/Teaching/CS515/pythonTutorial.pdf> (The G. Van Rossum tutorial)

A PyQt tutorial :

<http://zetcode.com/gui/pyqt4/>

PyQt class references:

<http://pyqt.sourceforge.net/Docs/PyQt4/classes.html>

PyQt plotting demos:

<http://eli.thegreenplace.net/2009/05/23/more-pyqt-plotting-demos/>

Model view programming tutorials:

<http://www.yasinuludag.com/blog/?p=98>

Matplotlib tutorial:

<http://web.archive.org/web/20100830233506/http://matplotlib.sourceforge.net/leftwich_tut.txt>

1. Acquisition protocol

Some basic steps concerning the acquisition procedure are required, in order to have the program working properly. Some acquisition procedures should also be applied in a more general sense, to get more accurate results and efficient surveys. For a detailed analysis and description of gravity networks, see for instance (Lambert and Beaumont, 1977; Torge, 1980). Requirements and advices for high quality survey designs can be found in Seigel et al., (1995).

* 1. Loop geometries and nomenclature
* Single stations should always keep the same ID number (name), regardless of their status (base station/network station). If it is changed, it can be manually modified using a search/replace procedure on the raw data file in a text editor.
* Different base stations within a single processing scheme are not handled up to now, but the program may be launched sequentially for each subset of data.
* It is preferable, but not necessary, that base station has a different name than cycling station (if it corresponds to the same location).
  1. Transportation
* CG5 is very sensitive to transportation: for short distances, it could be carried by two operators, who hold a rod on which the gravimeter bag is hung by the handles. For longer distances, it should be isolated at best from vehicle vibrations.
* Transportation usually imply a so-called short time ‘transportation drift’, and spring stabilization once the station is reached, may take some time. The data selection step in pyGrav allows to select data after the instrument has stabilized.
  1. Setting up the instrument
* As vertical movements strongly impact gravity changes (≈0.3µgal/mm using a nominal free air gradient), stations should be pillars made of concrete to limit the consequences of a soft ground or other unstable features. Alternatively, drive rods could be used for limiting the role of soil shrink/swell. For each survey (in a time-lapse application), the instrument should be located exactly at the same point, and kept at the same azimuth thanks to screws stuck within the concrete pillar for instance. The height with respect to the pillar should be kept constant with the use of a brass ring fixed to one foot of the instrument’s tripod, as suggested by Montgomery, (1971).
* The CG5 and its tripod should be shielded from wind and direct sun exposure, for instance by a dustbin covered with insulating material or an appropriate umbrella.
  1. Data acquisition
* There is (currently) no way to apply different tide corrections within the same survey. Therefore, if a survey is composed of stations separated from a large distance (several tens of km, not recommended for high precision time-lapse gravity surveys), the user should take care of entering correct station locations in the CG5 on the field, so that the CG5 earth tide correction can be applied.
* Duration: CG5 is sampling at 6Hz, and averaging over a user-defined duration to produce one single measurement. Merlet et al., (2008) showed in their study that the Allan deviation of their CG5 reached 1µgal after 40s, to a minimum of 0.8µgal after 85s but further increased because of tidal influence. Gettings et al., (2008) also show an example from raw 1s data (their figure 2), where running average converge at about 40s. Furthermore, after 60s, standard error on the CG5 display screen operating on the field always seems to have converged. Keeping 60s data often allow for a dynamic and fast inference of instrument stability (see for instance Hector et al., (submitted)).
* Stabilization identification: There are as many options as users and scientific objectives for defining whether the acquired time series is long enough (stabilized) or not. A rigorous procedure for small amplitude signal identification in time-lapse studies could be the following: At a station, a first set of 5 measurements are taken while the operators stay a few meters away from the instrument, before leveling is checked a first time. Another set of measurements are then taken, and operators come to check for instrument stability roughly every 5 minutes. At any check, if tilts are out of the 0 ± 5’’ range (or any other chosen threshold), the instrument is leveled again. This follows Merlet et al., (2008), who found the CG5 internal tilt correction to be accurate at the 1µgal level at ± 20‘’. They further managed to keep the tilts within 0 ± 3’’ in their indoor and stable pier conditions. Measurements may be considered achieved (and the gravimeter stable) when all the following criteria are reached: - a minimum of 10 relevant measurements are taken,
* - gravity changes are 3 µgal or less within 5 consecutive measurements, -there is no visible drift in the 5 last measurements (drift<1µgal/mn). All this imply that the long term CG5 drift is internally corrected in the instrument, say with a remaining drift < 100µgal/day (about 1µgal/15mn).

1. Least-square inversion and error propagation

Gravity observations at a station are time series of several gravity measurements, each measurement being the mean of several samples. For instance, a typical CG-5 measurement results from acquiring samples for more than a minute (85s is considered best, (Merlet et al., 2008)) at 6 Hz. Standard error () on each measurement is therefore:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

where is measurement standard deviation and is the number of samples.

After data have been selected, a single observation and a standard deviation are derived from the remaining time series, for each station using variance-weighted means:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

The standard deviation, for a relative gravity observation between station and is given by

|  |  |  |
| --- | --- | --- |
|  |  |  |

while is given by

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

Where are the residuals, is the gravity value at station , is the degree of the polynomial of coefficients for the gravimeter drift and is the degree of the polynomial of coefficients for the temperature drift. The equation system is therefore

|  |  |  |
| --- | --- | --- |
|  |  |  |

where is a vector containing relative gravity observations, is a vector with residuals, is the design matrix and is the vector of unknowns (gravity values and drift parameters). To obtain a solution of , one must hold fixed at least one gravity value during the adjustment (the so-called gravity datum). This is done here by adding absolute gravity observation (usually equal to zero) at the base station:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Introducing the vector with (number of gravity values) first values equal to one and last values equal to zero, which satisfies:

|  |  |  |
| --- | --- | --- |
|  |  |  |

one may find the least-square solution of :

|  |  |  |
| --- | --- | --- |
|  |  |  |

The weight matrix is composed of inverse of variance terms for observations.

The a posteriori variance of unit weight () is given by

|  |  |  |
| --- | --- | --- |
|  |  |  |

and the a posteriori covariance matrix is given by covariance propagation:

|  |  |  |
| --- | --- | --- |
|  |  |  |

The code also proceeds to a global model test, to test if the mathematical model is adequate, or if there are outliers in the data. If the following condition is met, then the model adjustment can be considered correct and complete, to the significance level :

|  |  |  |
| --- | --- | --- |
|  |  |  |

where is the a priori variance of unit weight and is the critical value of the distribution when the confidence level is and the degree of freedom of adjustment is m.

|  |  |  |
| --- | --- | --- |
|  |  |  |

and for ,

|  |  |  |
| --- | --- | --- |
|  |  |  |

and

|  |  |  |
| --- | --- | --- |
|  |  |  |

1. Installing external programs

pyGrav provides a frame for microgravity data processing. As such, it is easily feasible to include calls to external programs within pyGrav. Currently, two programs may be called, the PREDICT function from the ETERNA package (Wenzel, 1996) for the calculation of synthetic tides, and MCGRAVI (Beilin, 2006) for network adjustment. To add any other call to external programs, the best way is to go through pyGrav scripts and being inspired by the way these two programs are being interfaced. Here is described the installation of such programs, if they are to be used within pyGrav.

* 1. ETERNA installation

For window users, it is possible to use the PREDICT function from the ETERNA package (Wenzel, 1996) for tidal correction. This is particularly suited when tidal parameters for a specific site are available (for instance by a tidal analysis on superconducting gravimeter data, because this also includes ocean loading effect). The pyGrav directory includes a lighter version of ETERNA with the minimum required to get the PREDICT function to work properly. Simply copy the /eterna33 folder from the main\_code/eternal\_files/ folder to the root (C:\) directory. When pyGrav is asked to launch predict, it will copy an instance of the predict.exe file present in the main\_code/eternal\_files/ folder to the survey output directory and run it. This predict.exe program will then call tidal potential data from the C:/eterna33 folder. The full ETERNA package is also available free of charge at the ICET: <http://www.upf.pf/ICET/soft/index.html>.

* 1. MCGRAVI installation

MCGravi (Beilin, 2006) can be used for least-square drift adjustment and network compensation in the case of complicated network with several known absolute points (weighted constraint least-square inversion, see Hwang et al., 2002). Alternatively, the datum-free least square inversion algorithm (Hwang et al., 2002) is coded in pyGrav.

* Copy the MCGRAVI folder under the root (C:\) directory (or elsewhere)
* MCGRAVI may need to be recompiled. In this case G95 is needed. It may be found here: <http://www.g95.org/downloads.shtml>, or <http://math.hawaii.edu/~dale/190/fortran/fortran-windows-installation.html> or <http://www.fortran.com/the-fortran-company-homepage/whats-new/g95-windows-download/>
* With intel fortran:
  + Launch Visual Studio and open Mc\_gravi.vfproj project. No specific parameters are necessary
* With g95 :
  + cmd.exe (or exec -> cmd in the start menu) to open a dos console. cd to /mcgravi. “make clean” if needed, or “del \*.o” & ”del \*.mod”
  + make all
  + to launch mcgravi, copy the mcgravi.exe in the working directory, and mcgravi conf.conf in the dos window.
* Add the executable path to the PATH environment variable. Go to advanced system parameters under the system page (configuration panel). Select environment variable. In the lower panel (system variables), select the ‘Path’ line, and ‘modify’. Copy/paste the variable value in a text editor (Ctrl+A/ Ctrl+C => Ctrl+V). Add the path, followed by ‘;’. Do not change anything else.
* Install Perl
* GMT is also required for output maps from mcgravi, but everything works fine if it is not installed.

1. References

Agnew, D. C.: 3.06 - Earth Tides, in Treatise on Geophysics, edited by G. Schubert, pp. 163–195, Elsevier, Amsterdam., 2007.

Agnew, D. C.: SPOTL: Some Programs for Ocean-Tide Loading, SIO technical report (http://dx.doi.org/http://es cholarship.org/uc/sio\_techreport)., 2012.

Beilin, J.: Apport de la gravimétrie absolue à la réalisation de la composante gravimétrique du Réseau Géodésique Français, Master thesis, Inst. Géogr. Natl., Paris, France., 2006.

Cattin, R., Mazzotti, S. and Baratin, L.-M.: GravProcess: An easy-to-use MATLAB software to process campaign gravity data and evaluate the associated uncertainties, Comput. Geosci., 81, 20–27, doi:10.1016/j.cageo.2015.04.005, 2015.

Gettings, P., Chapman, D. S. and Allis, R.: Techniques, analysis, and noise in a Salt Lake Valley 4D gravity experiment, Geophysics, 73, 71, doi:10.1190/1.2996303, 2008.

Hartmann, T. and Wenzel, H.-G.: The HW95 tidal potential catalogue, Geophys. Res. Lett., 22(24), 3553–3556, doi:10.1029/95GL03324, 1995.

Hector, B. and Hinderer, J.: pyGrav, a Python-based program for handling and processing relative gravity data, Comput. Geosci., doi:10.1016/j.cageo.2016.03.010, 2016.

Hector, B., Séguis, L., Hinderer, J., Cohard, J.-M., Wubda, M., Descloitres, M. and Benarrosh, N.: Water storage changes as a marker for baseflow generation processes in a tropical humid basement catchment (Benin): insights from hybrid gravimetry, Water Resour. Res., Accepted.

Hwang, C., Wang, C.-G. and Lee, L.-H.: Adjustment of relative gravity measurements using weighted and datum-free constraints, Comput. Geosci., 28(9), 1005–1015, doi:10.1016/S0098-3004(02)00005-5, 2002.

Lambert, A. and Beaumont, C.: Nano variations in gravity due to seasonal groundwater movements: Implications for the gravitational detection of tectonic movements, J. Geophys. Res., 82(2), 297–306, doi:10.1029/JB082i002p00297, 1977.

Merlet, S., Kopaev, A., Diament, M., Geneves, G., Landragin, A. and Santos, F. P. D.: Micro-gravity investigations for the LNE watt balance project, Metrologia, 45(3), 265, doi:10.1088/0026-1394/45/3/002, 2008.

Montgomery, E. L.: Determination of coefficient of storage by use of gravity measurements., PhD Thesis, The University of Arizona. [online] Available from: http://arizona.openrepository.com/arizona/handle/10150/190978 (Accessed 5 April 2013), 1971.

Munk, W. H. and Cartwright, D. E.: Tidal Spectroscopy and Prediction, R. Soc. Lond. Philos. Trans. Ser. A, 259, 533–581, doi:10.1098/rsta.1966.0024, 1966.

Seigel, H. O., Brcic, I. and Mistry, P.: A guide to high precision land gravimeter surveys, Scintrex LTD, Concord, Ontario., 1995.

Torge, W.: Geodesy., Walter de Gruyter, Berlin, New York., 1980.

Wenzel, H.: The nanogal software: earth tide data processing package: Eterna 3.3., Bull Inf Marées Terr, (124), 9425–9439, 1996.