Computers & Geosciences 91 (2016) 90–97

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Research paper

pyGrav, a Python-based program for handling and processing relative gravity data

Basile Hector*[,1], Jacques Hinderer

Institut de Physique du Globe de Strasbourg UMR 7516 CNRS/Université de Strasbourg, 5 rue Descartes, 67084 Strasbourg, France

ARTICLE INFO

Article history:
Received 11 September 2015
Received in revised form
14 March 2016
Accepted 17 March 2016
Available online 18 March 2016

Keywords:
Relative-gravity
Time-variable gravity
Object-oriented
Python

ABSTRACT

pyGrav is a Python-based open-source software dedicated to the complete processing of relative-gravity data. It is particularly suited for time-lapse gravity surveys where high precision is sought. Its purpose is to bind together single-task processing codes in a user-friendly interface for handy and fast treatment of raw gravity data from many stations of a network. The intuitive object-based implementation allows to easily integrate additional functions (reading/writing routines, processing schemes, data plots) related to the appropriate object (a station, a loop, or a survey). This makes pyGrav an evolving tool. Raw data can be corrected for tides and air pressure effects. The data selection step features a double table-plot graphical window with either manual or automatic selection according to specific thresholds on data channels (tilts, gravity values, gravity standard deviation, duration of measurements, etc.). Instrumental drifts and gravity residuals are obtained by least square analysis of the dataset. This first step leads to the gravity simple differences between a reference point and any point of the network. When different repetitions of the network are done, the software computes then the gravity double differences and associated errors. The program has been tested on two specific case studies: a large dataset acquired for the study of water storage changes on a small catchment in West Africa, and a dataset operated and processed by several different users for geothermal studies in northern Alsace, France. In both cases, pyGrav proved to be an efficient and easy-to-use solution for the effective processing of relative-gravity data.

1. Introduction

Relative-gravity surveys are used in a wide range of applications, such as geothermics (e.g. Allis and Hunt, 1986; Hinderer et al., 2015), oil and gas exploration (e.g. Ferguson et al., 2008), volcanology (e.g. Greco et al., 2012; Jousset et al., 2000), or hydrology (e.g. Hector et al., 2015; Piccolroaz et al., 2015; Pfeffer et al., 2013; Jacob et al., 2010). They are classically used as an exploration tool for static imagery of the subsurface. Time-lapse gravity surveys are applied to the monitoring of mass redistributions.

Relative-gravity measurements are affected by several natural or instrumental processes such as earth tides and barometric pressure, long-term instrumental drift, transportation-induced drift and noise, temperature, etc. which must be corrected when a specific signal (e.g. water storage changes) is sought (Crossley et al., 2013). Furthermore, field data must be selected based on stability indicators (instruments tilts, measurements errors, presence of spikes in the data, etc.) for obtaining high-quality gravity residuals. Due to the intrinsic drifting behavior of relative spring gravity meters, field studies focus on gravity differences between stations, rather than readings at single stations. Processing (both corrections and selection steps) of a dataset acquired within the same period leads to what is known as “simple differences”, which are gravity differences between each station of the network and a base station. From this result, static imagery proceeds to subsequent so-called “static corrections” for analyzing these differences in terms of Bouguer anomalies. On the other hand, from temporal repetitions of surveys, time-lapse gravimetry studies leads to “double differences” (Pfeffer et al., 2013), which are temporal differences of simple differences results. Static corrections are not needed in such studies as static effects cancel out in survey-to-survey differences.

Many codes and routines exist for processing relative-gravity data (e.g. CgxTool (Gabalda et al., 2003), Gravnet (Hwang et al., 2002), MCGravi (Beilin, 2006)), but they are mostly dedicated to static surveys, and are focused on single-tasks, usually network adjustment. They also lack user-friendly data selection interfaces particularly needed for time-lapse investigations when high precisions are sought. Several research teams have developed their own processing chain which usually consists of sequential calls to

* Corresponding author.
E-mail address: basilehector@gmail.com (B. Hector).
[1] Present address: CNES/IRD/UJF-Grenoble 1/CNRS/G-INP, LTHE, UMR 5564, 38041 Grenoble, France.

http://dx.doi.org/10.1016/j.cageo.2016.03.010
0098-3004/

different programs having specific I/O and formalisms. This complexity makes the whole processing chain very slow and not much friendly to new users and further hinders reproducible research. The time required for data processing also prevents rapid on-field check of the data and subsequent data reacquisition in case of problems. There is therefore a strong need for a flexible tool that would allow binding all processing steps together and still authorizing easy incorporation of new functions or function updates for teams who wish to integrate some of their own codes.

The recent Gravprocess MATLAB™-based software (Cattin et al., 2015) is dedicated to the whole processing of complex gravity surveys and does not include an explicit data selection procedure. It is particularly suitable for large static surveys with several different base stations and allows the calculation of static corrections. It does not allow the simultaneous inversion of stations gravity values and drifts for overlapping or repeated lines within a short period where gravity changes (other than tides- and barometric-related ones) are assumed negligible (typically one or two days depending on environmental conditions). The study of time-lapse gravity changes usually requires high precision in the data acquisition and it is also highly important that stations and loops are repeated several times (e.g. Christiansen et al., 2011a, 2011b, 2011c; Jacob et al., 2010; Pfeffer et al., 2013).

With the ongoing development of gravity measurements for time-lapse studies, such as the Critex program in France for the monitoring of water storage changes in Critical Zone observatories (http://critex.des-mondes-singuliers.coop/), there will be an increasing need for such complete processing codes. They will need to be adaptable, for integrating outcome from new research, to be easy to manipulate and to be efficient for rapidly checking the data on field. This becomes particularly relevant with the emerging concept of hybrid gravimetry (e.g. Hector et al., 2015; Hinderer et al., Submitted, 2015). Hybrid gravimetry is the combined use of relative spring gravimeters with instruments measuring gravity changes at the network base station (*i.e.* superconducting gravimeters, measuring continuously, and absolute gravimeters, measuring episodically), in order to determine the absolute changes at all points of the network, see e.g. Okubo et al. (2002) or Sugihara and Ishido (2008).

This paper presents pyGrav, a multi-platform, open-source, Python-based software for processing relative-gravity data. The code is fully object-oriented, which is highly appropriate for handling structured relative-gravity data as described in a first section. The complete processing chain of relative-gravity data, including a data selection frame that relies on an object-oriented scheme is then presented. Finally, two different case studies (hydrology and geothermics) are shown.

2. Relative gravity measurements and object oriented programming

Geophysical data sets make often object-oriented programming very intuitive, such as in seismology (e.g. the Waveloc seismic event detection and location code also written in Python Langet et al., 2014), or for magnetotelluric data (Krieger and Peacock, 2014). The climate community also widely use the Python object-based capabilities (for instance the UV-CDAT data analysis tool, http://uvcdat.llnl.gov/), and so do the geographic information system (GIS) community. For instance the widely-used QGIS open-source software makes extensive use of Python capabilities and provides Python bindings which allow users to interface their own scripts through the GIS. Such open-source initiatives contribute to the development of highly collaborative work, sometimes through git repositories which allow different contributors to help developing a specific code.

Object-oriented programming is particularly suited for handling and processing relative-gravity data as data can be arranged as structures (objects –or classes, in Python) following a hierarchical definition: A relative-gravity *campaign* refers to a specific study and designates the whole data acquisition scheme, including stations network geometry, timing of measurements, acquisition protocol (e.g. Seigel et al., 1995), etc. A relative-gravity campaign may be composed of one or several *surveys*, each survey being undertaken under similar environmental conditions (e.g. assuming no significant changes of the target variable like water storage changes, ground deformation, mass changes, etc. during the survey period). This allows to process data of single surveys simultaneously, under the assumption that station values (once corrected for tides, air pressure and drift) should not have changed. Differences between surveys are the goal of time-lapse relative-gravity studies, as each survey gives access to a snapshot of the gravity field. Surveys include several *loops*, where each loop refers to the measurements of different *station*, with at least one station repetition (the base station) for drift control.

This intrinsic hierarchical structure allows for an easy and intuitive object-oriented programming frame for the handling and processing of relative-gravity data. Specific functions may be defined at the corresponding object (class) level (Fig. 1). For instance, network adjustment should be applied to a whole single survey. Earth tides and atmospheric corrections can be applied to the whole campaign, but data selection should be undertaken at the

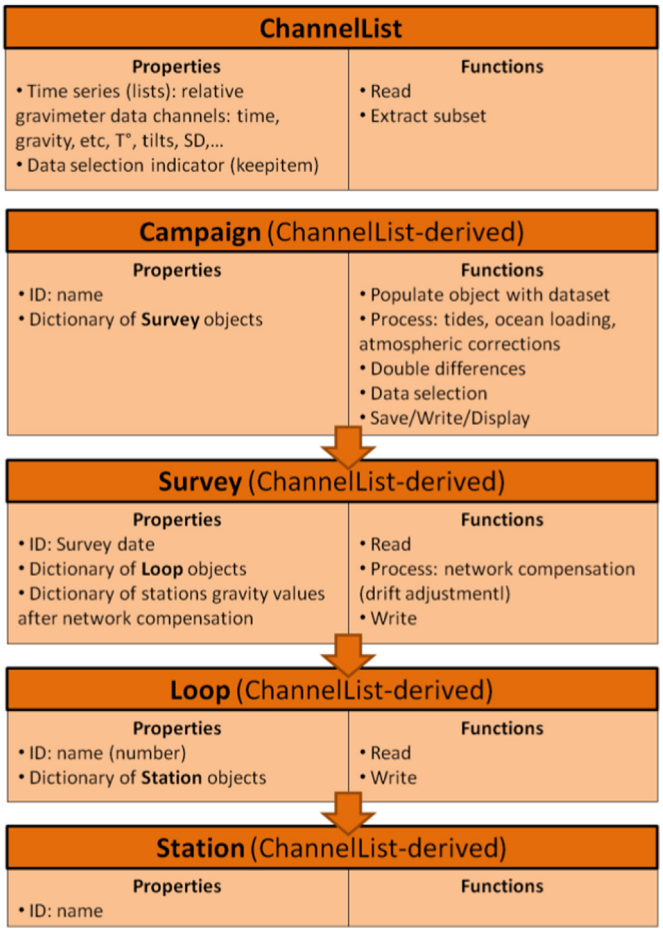

Fig. 1. Objects (classes) and objects imbrications which are the core of the pyGrav program. Each instance of each object has specific properties, and each object is associated to specific functions. All objects are derived from a Channel List-type object, and inherit its properties. Note that functions which imply graphical interactions (i.e. data selection) are defined in the main program script which includes pyQt functions.