Fatiando a Terra: Open-source tools for geophysics

Santiago R. Soler\* and Lindsey J. Heagy

Department of Earth, Ocean and Atmospheric Sciences, University of British Columbia

Summary

In this talk, we will present the Fatiando a Terra project, a collection of open-source Python libraries designed for geophysical applications. We will describe how the project has grown from its start as a simple library part of a PhD Thesis in South America to a production-quality codebase, and how it created a community around it that actively collaborates to its development and its current state. We will introduce the tools available in the project and show examples with field data of how they can be used to solve geophysical problems. Finally we will discuss some of the current challenges and mention the upcoming features and development plans for the future.

Introduction

Advancements in computing, from the origins of silicon-based chips, the development of numerical solvers and the advent of multiprocessing have each enabled new opportunities to solve larger and more complex problems in geophysics. Software developed by researchers and industry professionals enable us to process large amount of data, generate visualizations for interpretation, and ultimately to perform inversions to build models of the subsurface. All of which are key to improve decision making processes. Although much of this software was built as in-house tools, the appearance of open-source software for geosciences happened as early as the 70s and 80s. Projects like Seismic Unix (Stockwell, 1999) and GMT (Wessel et al., 2019) are pioneer examples.

In recent decades, the popularity of the Python language amongst every scientific field planted the seeds for an ever-growing open-source geoscientific ecosystem. Python tools aimed at solving geoscientific problems have proliferated, and there are now open-source tools that span geologic modelling, data processing, inversion, visualization, and much more.

In this setting, Leonardo Uieda created the Fatiando a Terra project (Uieda et al., 2013; https://www.fatiando.org) aimed at developing open-source Python tools for geophysics. It started in 2010 as a single Python library as part of his PhD Thesis in South America. Being available under an open-source license and its clear and comprehensive documentation have facilitated its use by researchers and industry professionals from different regions of the world. Some of these users have since become contributors by writing new features, fixing bugs, adding more tests, improving the documentation or having an active participation in its community. Now, the Fatiando project has over 30 contributors, and it continues to grow using a community-driven model.

The main goal of the project is to provide open-source software tools that are easy to use and also well designed, tested, and documented. Nowadays it consists in a set of Python libraries, each one of them with a very specific scope of application. These libraries offer software solutions for downloading and caching data from the web, handling and interpolating spatial data, computing normal gravity of reference ellipsoids using analytic solutions, processing potential fields data with frequency-domain filters, forward modelling gravity and magnetic fields of different geometrical bodies, and gridding harmonic data with equivalent sources.

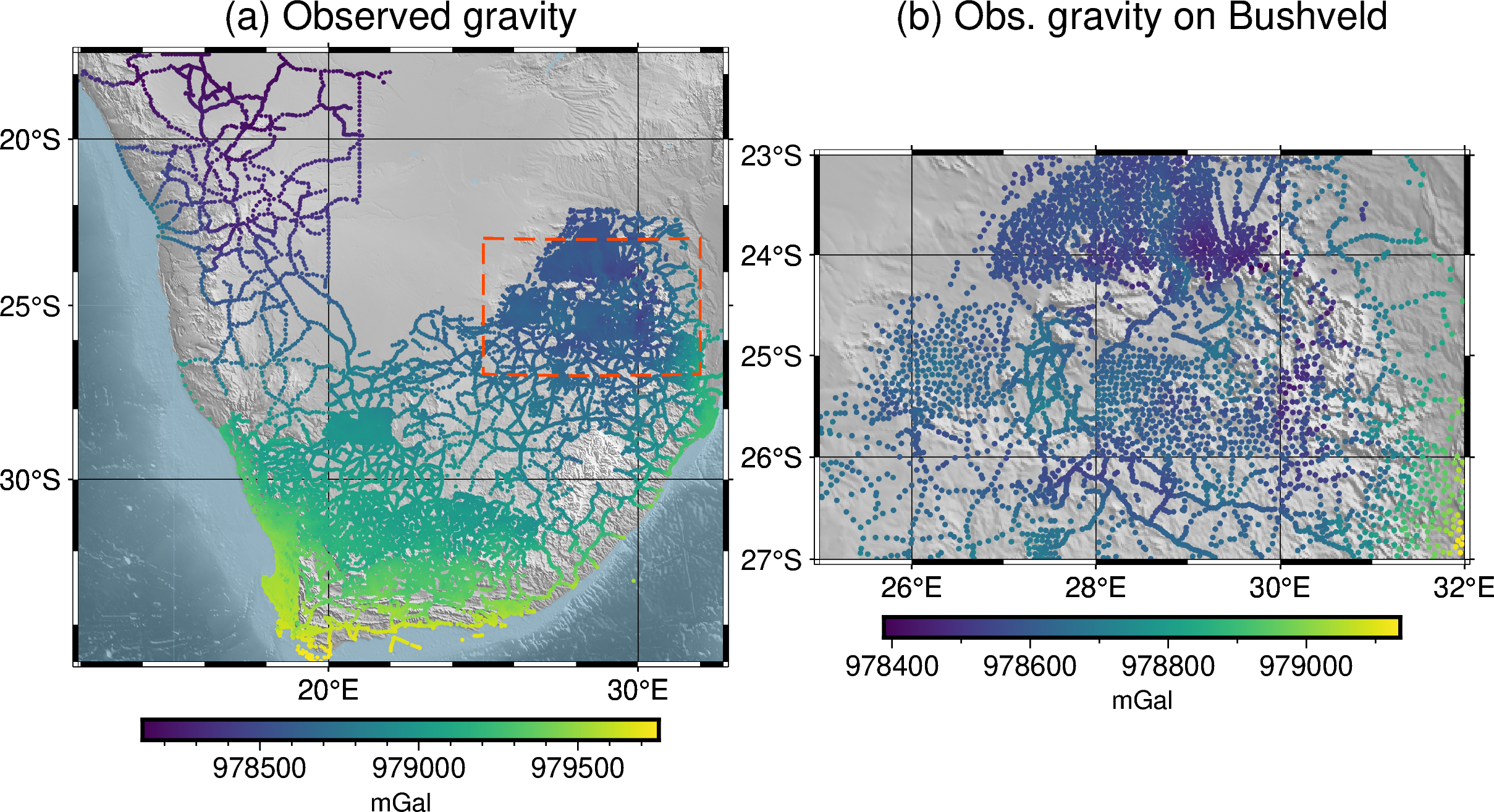
During this talk we will provide an overview of the tools available in the project and demonstrate their functionalities using examples from research and industry applications.

The Project

In its origins, Fatiando a Terra consisted of a single Python library named fatiando (https://legacy.fatiando.org) that used to host all the features that the project offered. After years of development, this design approach proved to be flawed. The large size of the code base made it difficult to maintain and to extend its functionality. Further, the code contained in this library varied greatly in scope and maturity, ranging from toy problems meant to be used for educational purposes to production ready tools.

In 2018 the community decided to redesign the code base by splitting the old fatiando library into several smaller libraries, each one with a very specific scope. This simplifies both the adoption of the libraries and also their development. Most users look for only a subset of the tools offered by the project, and having them divided in libraries reduces the size of the libraries they depend on. As a side effect, anyone interested in changing the code of one of our libraries now needs to familiarize themselves with a smaller code base, making it easier for new contributors to collaborate to the project.

In parallel, the broader geoscientific Python ecosystem has seen a major growth. Libraries like SimPEG (Cockett et al., 2015), GemPy (de la Varga et al., 2019), pyGIMLi (Rücker et al., 2017) and ObsPy (Obspy, 2019) were established. These projects provide scientists and industry with a wide range of tools for research and exploration. As a project, we decided to invest in the strengths of the Fatiando libraries in light of the broader ecosystem. The introduction of smaller, and narrowly scoped libraries has again been advantageous as it allows other projects to use them without introducing a large set of unnecessary dependencies.

Figure 1. Gravity data over Southern Africa: (a) observed gravity data and box delimiting the boundaries of a region that contains the Bushveld Igneous Complex, (b) observed data limited to the region around the Bushveld Igneous Complex. Data made available by NOAA NCEI and downloaded using Ensaio.

The project is currently formed by five libraries: Verde, Boule, Harmonica, Pooch and Ensaio.

**Verde**

Verde hosts tools for spatial data processing, interpolation, and gridding. Its core interpolation methods are inspired by machine learning, hence its interface reassembles one of the popular machine learning Python packages, Scikit-learn (Pedregosa et al., 2011). Additionally, it offers analysis tools that accompany the interpolators, such as trend removal, windowing operations, cross-validation, k-folding, grid projection, and more coordinate manipulation utilities.

**Boule**

Boule is a very lightweight library that hosts classes for representing geodetic reference ellipsoids for the Earth and for celestial bodies of the solar system like the Moon, Mars, Venus, and Mercury. These classes also offer methods to perform coordinate conversions between geodetic and geocentric spherical systems, and to compute the normal gravity generated by these ellipsoids on any external point through a closed-form analytic solution (Li and Götze, 2001).

**Harmonica**

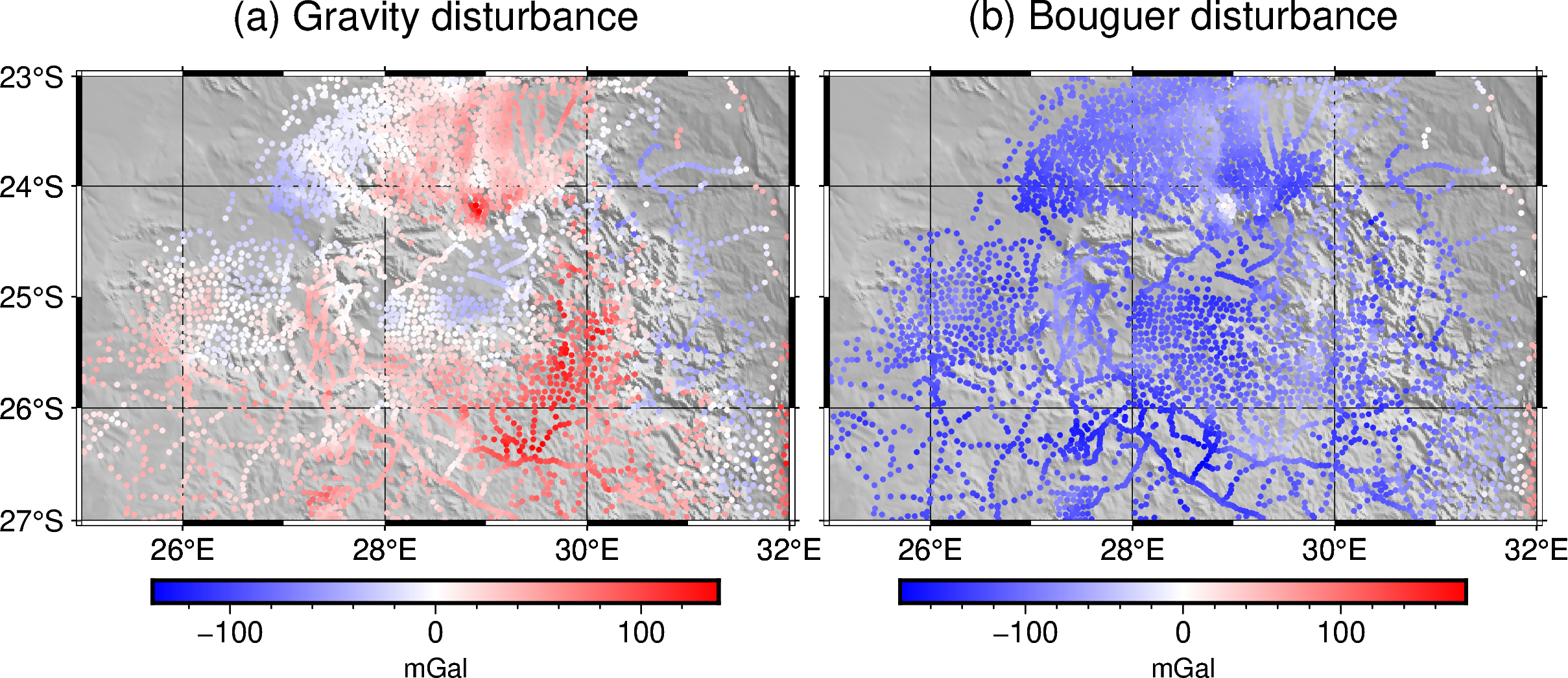
Harmonica offers functions and classes for processing and modelling gravity and magnetic data. It hosts functions for forward modelling the gravity fields of point sources, rectangular prisms and also tesseroids (a.k.a. spherical prisms). It can perform gravity corrections, from a simple Bouguer correction to a full terrain correction, by forward modelling digital elevation models with prisms. Regular grids can be transformed using FFT-based filters including upward derivative, upward continuation, and reduction to the pole, among others. It also offers ways to perform interpolation, gridding, and upward continuation through the equivalent sources technique. Finally, it can also read data stored in popular formats like .gdf files provided by the ICGEM Calculation Service and .grd files from Oasis Montaj©.

**Pooch**

The most general purpose library in the project is Pooch, which offers an easy-to-use interface for downloading and caching data from the web. Originally designed for scientific applications and to be used by other software packages, Pooch can download data from the web through a large range of protocols, cache it locally at a desired location, and also check the integrity of those files. This simple but powerful library is currently being used by other projects in the scientific Python stack, like SciPy (Virtanen et al., 2020), scikit-image (van der Walt et al., 2014), MetPy (May et al., 2016) and icepack (Shapero et al., 2020), among others.

**Ensaio**

Lastly, we introduce Ensaio, a small library that hosts open licensed datasets that are useful for running examples and tutorials, for teaching, and for probing our codes. It uses Pooch under the hood to download and cache those datasets locally so its codebase ends up being very slim.

Figure 2. (a) Gravity disturbance over the Bushveld Igneous Complex. (b) Bouguer gravity disturbance obtained after removing the terrain effect from the gravity disturbance.

Example: Gravity data over South Africa

We can use Ensaio to easily fetch some curated datasets. For example, let's download a gravity dataset over Southern Africa (made available by NOAA NCEI):

**import** **ensaio**

ensaio.fetch\_southern\_africa\_gravity(

version=1

)

Figure 1 shows plots of the downloaded gravity data over Southern Africa (Fig. 1a), and a cropped region focusing on the Bushveld Igneous Complex (Fig. 1b).

Using Boule we can define an object that represents the WGS84 reference ellipsoid and use it to compute the normal gravity, i.e. the gravity acceleration of the reference ellipsoid on every observation point:

**import** **boule** **as** **bl**

ellipsoid = bl.WGS84

ellipsoid.normal\_gravity(

data.latitude, data.height

)

By removing the normal gravity from the observed gravity we can obtain the gravity disturbance (see Fig. 2a).

In order to obtain the Bouguer gravity disturbance we need to remove the terrain effect from the gravity disturbance we already computed. Harmonica allows us to compute the gravity effect of the topographic masses on every observation point by approximating them with rectangular prisms with a specified density and forward modelling them (see Fig. 2b)

**import** **harmonica** **as** **hm**

density = np.where(

topography > 0, 2670, 1040 - 2670

)

model = hm.prism\_layer(

coordinates=(

topography.easting,

topography.northing

),

surface=topography,

reference=0,

properties={"density": density},

)

terr\_eff = model.prism\_layer.gravity(

coordinates, field="g\_z"

)

To achieve the goal of obtaining the gravity effect of the shallower masses we need to split the residual field from the regional field. We can use Harmonica to generate the regional field using deep equivalent sources (see Fig. 3a).

deep\_sources = hm.EquivalentSources(

damping=1000, depth=500e3

)

deep\_sources.fit(

coordinates, gravity\_bouguer

)

regional = deep\_sources.predict(

coordinates

)

residual = gravity\_bouguer - regional

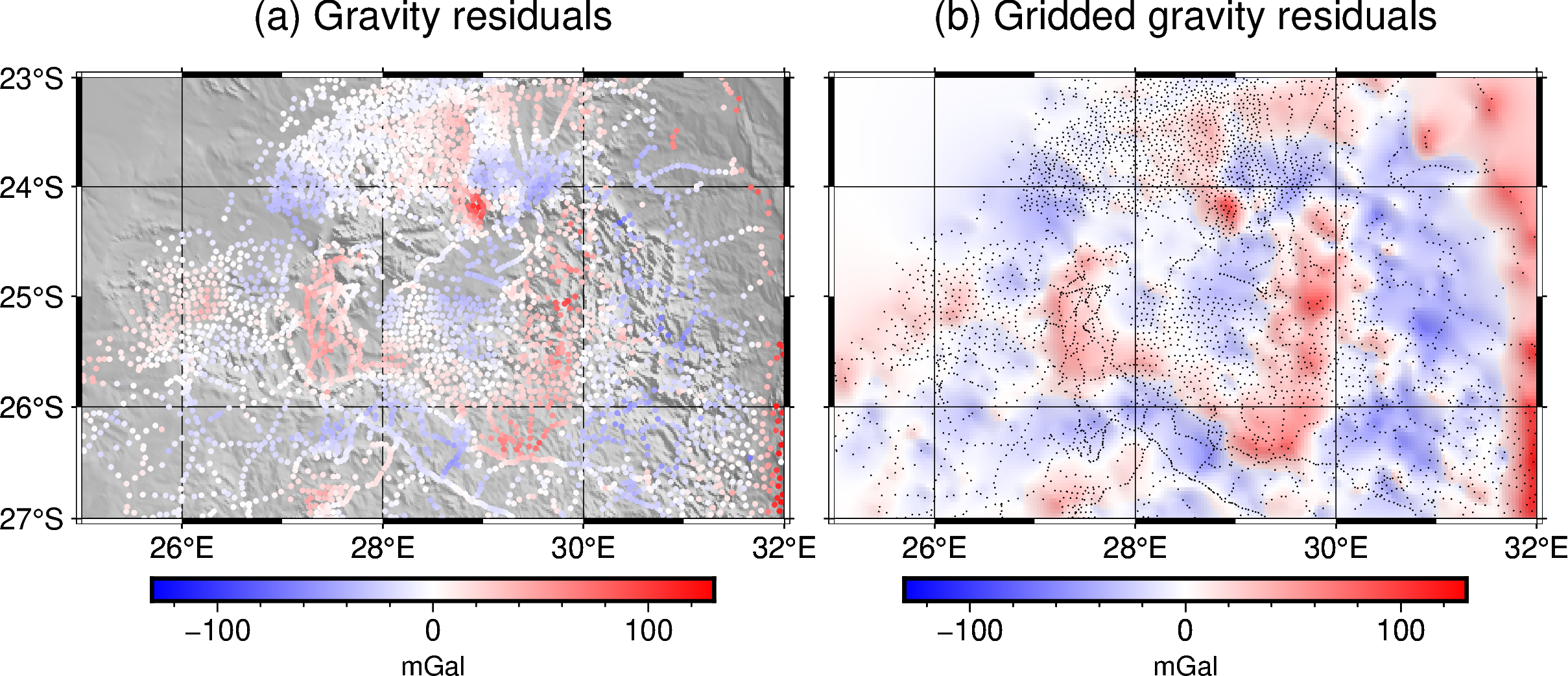
In a similar way, we can use Harmonica's equivalent sources to grid the residual field at a constant height (see Fig. 3b):

eq\_sources = hm.EquivalentSources(

damping=10, depth=10e3

)

eq\_sources.fit(

Figure 3. (a) Residual gravity field split using deep equivalent sources. (b) Gridded product of the residual gravity field at a constant height of 2200m above the ellipsoid.

coordinates, gravity\_residual

)

grid = eq\_sources.grid(

upward=2200,

spacing=grid\_spacing,

)

This example showcases how the tools available in the aforementioned libraries can be used together to process gravity data. The full code for running this example, along with a detailed explanation of the steps, can be found in https://www.fatiando.org/tutorials. Figures were produced using pyGMT (Uieda, 2022).

Conclusions

After more than a decade since the Fatiando a Terra project was started, the project continues to grow thanks to a network of collaborators and users that actively participate in its development. The success of the project has been enabled by having the code be openly-licensed and by promoting a community-driven model of software development. In addition to serving direct users, the benefits of the project extend more broadly as Fatiando libraries are relied upon by other projects in the scientific Python ecosystem. Furthermore, we have contributed to the advancement and adoption of best-practices in open-source software development within this broader community.

Fatiando has been used by students and researchers as core parts of their Thesis and scientific articles, and also by industry. The growing reliance on open-source tools by researchers and professionals highlights the importance of continued maintenance and development of these tools.

Plans for future development include the implementation of more efficient forward modelling functions for Harmonica as well as the expansion of functionality for equivalent source calculations and FFT-based transformations. Moreover, we plan to bring back the inversion framework that the old fatiando library had, with the opportunity of improving it with modern tools and focusing on types of inversions that are missing in other packages of the geophysical Python ecosystem.

We invite everyone to try out our tools, suggest new features, bring new use cases and especially make contributions to the code and documentation. The Fatiando website (https://www.fatiando.org/contact) has resources for getting involved.

Acknowledgments

We are indebted to the developers and maintainers of all the open-source software and particularly to the users and contributors of Fatiando a Terra. Without their gratitude, effort, and contributions this project wouldn't exist.

References

Cockett, R., Kang, S., Heagy, L. J., Pidlisecky, A., & Oldenburg, D. W. (2015). SimPEG: An open source framework for simulation and gradient based parameter estimation in geophysical applications. Computers & Geosciences, 85, 142–154. https://doi.org/10.1016/j.cageo.2015.09.015

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hoyer, S., & Hamman, J. (2017). xarray: N-D labeled Arrays and Datasets in Python. Journal of Open Research Software, 5(1), 10. https://doi.org/10.5334/jors.148

Li, X., & Götze, H. (2001). Ellipsoid, geoid, gravity, geodesy, and geophysics. GEOPHYSICS, 66(6), Article 6. https://doi.org/10.1190/1.1487109

May, R., Arms, S., Marsh, P., Bruning, E., Leeman, J., Bruick, Z., & Camron, M. D. (2016). MetPy: A Python Package for Meteorological Data. UCAR/NCAR-Unidata. https://doi.org/10.5065/D6WW7G29

McKinney, W. (2010). Data Structures for Statistical Computing in Python. 56–61. https://doi.org/10.25080/Majora-92bf1922-00a

The ObsPy Development Team (2019). ObsPy 1.1.1 (1.1.1). Zenodo. https://doi.org/10.5281/ZENODO.1040770

The Pandas Development Team (2023). pandas-dev/pandas: Pandas (v1.5.3). Zenodo. https://doi.org/10.5281/ZENODO.3509134

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., y Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830.

Rücker, C., Günther, T., & Wagner, F. M. (2017). pyGIMLi: An open-source library for modelling and inversion in geophysics. Computers & Geosciences, 109, 106–123. https://doi.org/10.1016/j.cageo.2017.07.011

Shapero, D., Badgeley, J., & Hoffman, A. (2020). icepack: Glacier flow modeling with the finite element method in Python (v1.0.0). Zenodo. https://doi.org/10.5281/ZENODO.1205640

Stockwell, J. W. (1999). The CWP/SU: Seismic Un∗x package. Computers & Geosciences, 25(4), 415–419. https://doi.org/10.1016/S0098-3004(98)00145-9

Uieda, L., V. C. Oliveira Jr, and V. C. F. Barbosa (2013), Modeling the Earth with Fatiando a Terra, Proceedings of the 12th Python in Science Conference, pp. 91 - 98.

de la Varga, M., Schaaf, A., & Wellmann, F. (2019). GemPy 1.0: Open-source stochastic geological modeling and inversion. Geoscientific Model Development, 12(1), 1–32. https://doi.org/10.5194/gmd-12-1-2019

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Vázquez-Baeza, Y. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods, 17(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., & Yu, T. (2014). scikit-image: Image processing in Python. PeerJ, 2, e453. https://doi.org/10.7717/peerj.453

Wessel, P., Luis, J. F., Uieda, L., Scharroo, R., Wobbe, F., Smith, W. H. F., & Tian, D. (2019). The Generic Mapping Tools Version 6. Geochemistry, Geophysics, Geosystems, 20(11), 5556–5564. https://doi.org/10.1029/2019GC008515