Tesseroids

Release v1.2.1

2017

Contents

Tesseroids

A collection of **command-line programs** for modeling the **gravitational potential, acceleration, and gradient tensor**. *Tesseroids* supports models and computation grids in Cartesian and spherical coordinates.

Developed by Leonardo Uieda in cooperation with Carla Braitenberg.

Official site: http://tesseroids.leouieda.com

License: *BSD 3-clause*

Source code: https://github.com/leouieda/tesseroids

Latest release: v1.2.1 (doi:10.5281/zenodo.582366)

Note: *Tesseroids* is **research software**. Please consider *citing* it in your publications if you use it for your research.

Warning: See the list of *known issues* for things you should be aware of.

The geometric element used in the modeling processes is a **spherical prism**, also called a **tesseroid**. *Tesseroids* also contains programs for modeling using **right rectangular prisms**, both in **Cartesian** and **spherical coordinates**.

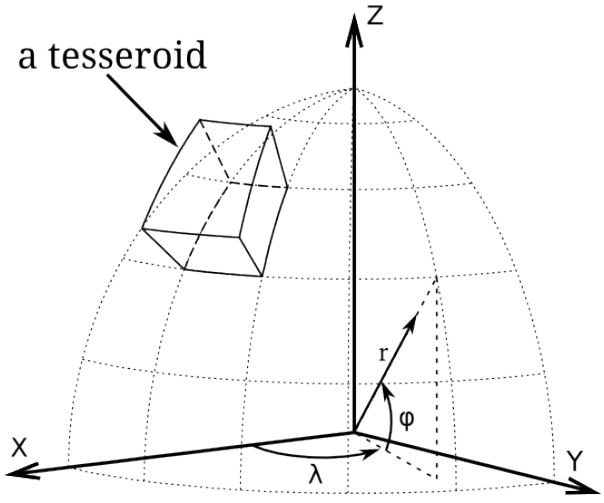


Fig. 1: View of a tesseroid (spherical prism) in a geocentric coordinate system. Original image (licensed CC-BY) at doi:10.6084/m9.figshare.1495521.

CHAPTER 1

Getting started

Take a look at the examples in the *Cookbook*. They contain scripts that run *Tesseroids* and some Python code to plot the results.

If you're the kind of person who likes to see the equations (who doesn't?), see the *Theoretical background* and the references cited there.

For a more detailed description of the software, options, and conventions used, see the *usage instructions*.

Also, all programs accept the `-h` flag to print the instructions for using that particular program. For example:

```
$ tessgrd -h
Usage: tessgrd [PARAMS] [OPTIONS]

Make a regular grid of points.

All units either SI or degrees!

Output:
  Printed to standard output (stdout) in the format:
    lon1    lat1    height
    lon2    lat1    height
    ...     ...     ...
    lonNLON lat1    height
    lon1    lat2    height
    ...     ...     ...
    ...     ...     ...
    lonNLON latNLAT height

  * Comments about the provenance of the data are inserted into
    the top of the output

Parameters:
  -r            W/E/S/N: Bounding region of the grid.
  -b            NLON/NLAT: Number of grid points in the
                longitudinal and latitudinal directions.
  -z            HEIGHT: Height of the grid with respect to the
```

```
               mean Earth radius.
  -h           Print instructions.
  --version    Print version and license information.

Options:
  -v           Enable verbose printing to stderr.
  -lFILENAME   Print log messages to file FILENAME.

Part of the Tesseroids package.
Project site: <http://fatiando.org/software/tesseroids>
Report bugs at: <http://code.google.com/p/tesseroids/issues/list>
```

CHAPTER 2

Getting help

Write an e-mail to Leonardo Uieda, or tweet, or Google Hangout. **Even better**, submit a bug report/feature request/question to the Github issue tracker.

CHAPTER 3

Contents

Citing

Geophysics paper

To cite *Tesseroids* in publications, please use our paper published in *Geophysics*:

> Uieda, L., V. Barbosa, and C. Braitenberg (2016), Tesseroids: Forward-modeling gravitational fields in spherical coordinates, GEOPHYSICS, F41-F48, doi:10.1190/geo2015-0204.1.

You can download a copy of the paper PDF and see all source code used in the paper at the Github repository.

Please note that **citing the paper is prefered** over citing the previous conference proceedings.

If you're a BibTeX user:

```
@article{uieda2016,
  title = {Tesseroids: {{Forward}}-modeling gravitational fields in spherical␣
↪coordinates},
  author = {Uieda, L. and Barbosa, V. and Braitenberg, C.},
  issn = {0016-8033},
  doi = {10.1190/geo2015-0204.1},
  url = {http://library.seg.org/doi/abs/10.1190/geo2015-0204.1},
  journal = {GEOPHYSICS},
  month = jul,
  year = {2016},
  pages = {F41--F48},
}
```

Source code

You can refer to individual versions of Tesseroids through their DOIs. However, please **also cite the Geophysics paper**.

For example. if you want to mention that you used the 1.1.1 version, you can go to *the Releases page* of the documentation and get the DOI link for that version. This link will not be broken, even if I move the site somewhere else.

You can also cite the specific version instead of just providing the link. If you click of the DOI link for 1.1.1, the Zenodo page will recommend that you cite it as:

> Uieda, Leonardo. (2015). Tesseroids v1.1.1: Forward modeling of gravitational fields in spherical coordinates. Zenodo. 10.5281/zenodo.15800

Conference proceeding

The previous way citation for Tesseroids was a conference proceeding from the 2011 GOCE User Workshop:

> Uieda, L., E. P. Bomfim, C. Braitenberg, and E. Molina (2011), Optimal forward calculation method of the Marussi tensor due to a geologic structure at GOCE height, Proceedings of the 4th International GOCE User Workshop.

Download a PDF version of the proceedings. You can also see the poster and source code at the Github repository.

Known issues

- Prism and tesseroid calculations are **only valid outside** of the mass elements. If you calculate on top or inside of the prism/tesseroid, there is no guarantee that the result will be correct.
- The gravity gradient components of tesseroids suffer from increased numerical error as the computation point gets closer to the tesseroid. It is not recommened to compute the effects at distances smaller than 1km above the tesseroid.

Changelog

Changes in version 1.2.1

- Binaries for Windows 64bit are now available for download as well. (PR 28)
- Validate order of boundaries for input tesseroids. Errors if boundaries are switched (e.g, W > E). (PR 27)
- Ignore tesseroids with zero volume from the input file (i.e., W == E, S == N, or top == bottom). These elements can cause crashes because of infinite loops during adaptive discretization. (PR 27)

Changes in version 1.2.0

- General improvements to the adaptive discretization (described in the upcoming method paper). (PR 21)
- Better error messages when there is a stack overflow (computation point too close to the tesseroid). (PR 21)
- Replace the recursive algorithm with a stack-based algorithm for adaptive discretization of tesseroids. This makes the computations faster, specially for gravity acceleration and gradient tensor components. (PR 21)
- Divide the tesseroids only along the necessary dimensions. This provides speedups when dealing with flattened or elongated tesseroids. (PR 21)
- Speedup tesseroid computations by moving some trigonometric functions out of loops. (PR 22)

- BUG fix: Singularities when calculating around a prism. Due to wrong quadrant returned by atan2 and log(0) evaluations. Fix by wrapping atan2 in a safe_atan2 that corrects the result. log(0) error happened only in cross components of the gravity gradient when the computation is aligned with the vertices of a certain face (varies for each component. Fix by displacing the point a small amount when that happens. (PR 12)

Changes in version 1.1.1

- BUG fix: Wrong results when calculating fields below a prism in Cartesian coordinates (PR 1)

Changes in version 1.1

- the tesseroids license was changed from the GNU GPL to the more permissive BSD license (see *the license text*).
- tess2prism has a new flag –flatten to make the prism model by flattening the tesseroids (i.e., 1 degree = 111km) into Cartesian coordinates (so that they can be used with the prismg* programs).
- tessg* programs have a new flag -t used to control the distance-size ratio for the automatic recursive division of tesseroids.
- **NEW PROGRAMS** prismpots, prismgs, and prismggts, to calculate the prism effects in spherical coordinates. These programs are compatible with the output of tess2prism (see *this recipe* for an example).
- **NEW PROGRAM** tesslayers to generate a tesseroid model of a stack of layers from grids of the thickness and density of each layer. tesslayers complements the functionality of tessmodgen and can be used to generate crustal models, sedimentary basin models, etc. (see *this recipe* for an example).
- tesseroids now strictly follows the ANSI C standard.
- Bug fix: prismpot, prismgx, prismgy, prismgz, and prismgxy had problems with a log(z + r) when the computation point was bellow the top of the prism (zp > prism.z1). Fixed by calculating on top of the prism when this happens, then changing the sign of the result when needed (only for gz).
- Bug fix: the tessg and prismg family of programs was crashing when the model file is empty. Now they fail with an error message.

Changes in version 1.0

Tesseroids 1.0 was completely re-coded in the C programming language and is much faster and more stable than the 0.3 release. Here is a list of new features:

- tesspot and tessg* programs now take the computation points as input, allowing for custom grids.
- tesspot and tessg* programs now automatically subdivide a tesseroid if needed to maintain GLQ precision (this makes computations up to 5x faster and safer).
- Automated model generation using program tessmodgen.
- Regular grid generation with program tessgrd.
- Total mass calculation with program tessmass.
- Programs to calculate the gravitational fields of right rectangular prisms in Cartesian coordinates.
- HTML User Manual and API Reference generated with Doxygen.
- Easy source code compilation with SCons.

Installing

We offer binaries for Windows (32 and 64 bit) and GNU/Linux (32 and 64 bit). You can download the latest version for your operating system from Github:

https://github.com/leouieda/tesseroids/releases/latest

Once downloaded, simply unpack the archive in the desired directory. The executables will be in the `bin` folder. For easier access to the programs, consider adding the bin folder to your PATH environment variable.

Tesseroids is permanently archived in Zenodo. Each release is stored (source code and binaries) and given a DOI. The DOIs, source code, and compiled binaries for previous versions can be found on the *Releases* page.

If we don't provide the binaries for your operating system, you can compile the source code (download a source distribution from Github) by following the instructions below.

Compiling from source

If you want to build *Tesseroids* from source, you'll need:

- A C compiler (preferably GCC)
- The build tool SCons

Setting up SCons

Tesseroids uses the build tool SCons. A `SConstruct` file (`Makefile` equivalent) is used to define the compilation rules. The advantage of SCons over Make is that it automatically detects your system settings. You will have to download and install SCons in order to easily compile Tesseroids. SCons is available for both GNU/Linux and Windows so compiling should work the same on both platforms.

SCons requires that you have Python installed. Follow the instructions in the SCons website to install it. Python is usually installed by default on most GNU/Linux systems.

Under Windows you will have to put SCons on your `PATH` environment variable in order to use it from the command line. It is usually located in the `Scripts` directory of your Python installation.

On GNU/Linux, SCons will generally use the GCC compiler to compile sources. On Windows it will search for an existing compiler. We recommend that you install GCC on Windows using MinGW.

Compiling

Download a source distribution and unpack the archive anywhere you want (e.g., `~/tesseroids` or `C:\tesseroids` or whatever). To compile, open a terminal (or `cmd.exe` on Windows) and go to the directory where you unpacked (use the `cd` command). Then, type the following and hit `Enter`:

```
scons
```

If everything goes well, the compiled executables will be placed on a `bin` folder.

To clean up the build (delete all generated files), run:

```
scons -c
```

If you get any strange errors or the code doesn't compile for some reason, please submit a bug report. Don't forget to copy the output of running `scons`.

Testing the build

After the compilation, a program called `tesstest` will be placed in the directory where you unpacked the source. This program runs the unit tests for *Tesseroids* (sources in the `test` directory).

To run the test suite, simply execute `tesstest` with no arguments:

```
tesstest
```

or on GNU/Linux:

```
./tesstest
```

A summary of all tests (pass or fail) will be printed on the screen. If all tests pass, the compilation probably went well. If any test fail, please submit a bug report with the output of running `tesstest`.

Releases

Developement

The latest development version can be found on github.com/leouieda/tesseroids. The *master* branch is kept stable and can be used. See the *install guide* for instruction on compiling the source code.

Stable releases

- **v1.2.1:**
  - Source code
  - Download
  - Documentation
  - doi:10.5281/zenodo.16033
- **v1.2.0:**
  - Source code
  - Download
  - Documentation
  - doi:10.5281/zenodo.16033
- **v1.1.1:**
  - Source code
  - Download
  - Documentation
  - doi:10.5281/zenodo.15800
- **v1.1:**
  - Source code
  - Download
  - Documentation

  - doi:10.5281/zenodo.15801

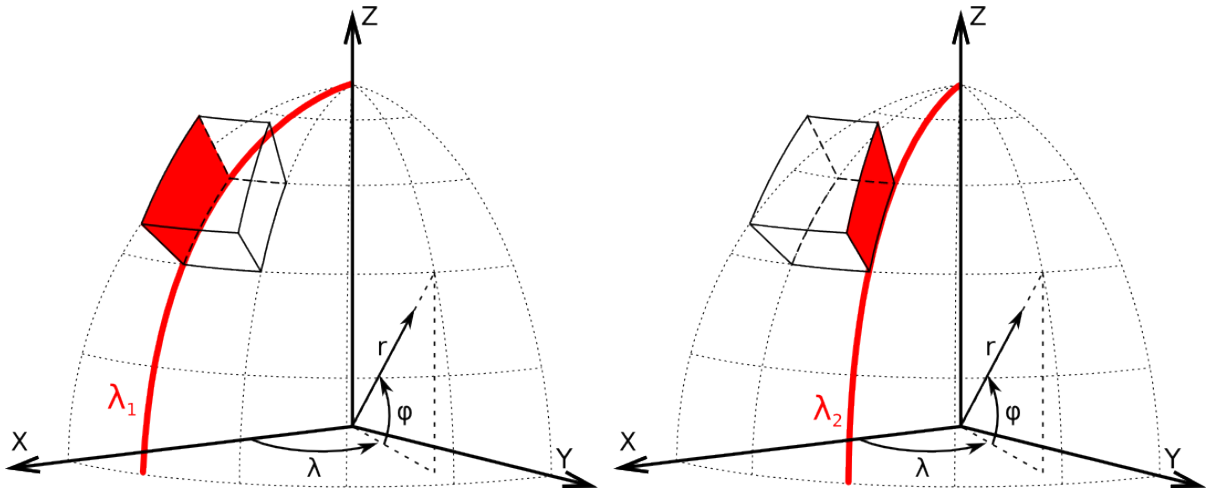
- **v1.0:**
  - Source code
  - Download
  - doi:10.5281/zenodo.15803
- **v0.3:**
  - Source code
  - Download
  - doi:10.5281/zenodo.15804
- **v0.1:**
  - Source code
  - Download
  - doi:10.5281/zenodo.15805

Theorerical background

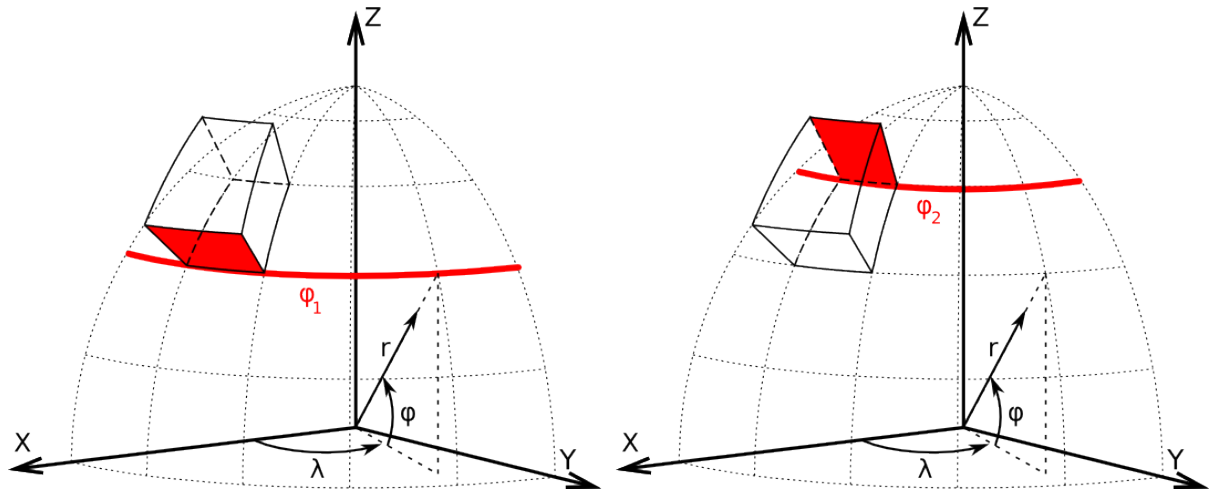
What is a tesseroid anyway?

A tesseroid, or spherical prism, is segment of a sphere. It is delimited by:

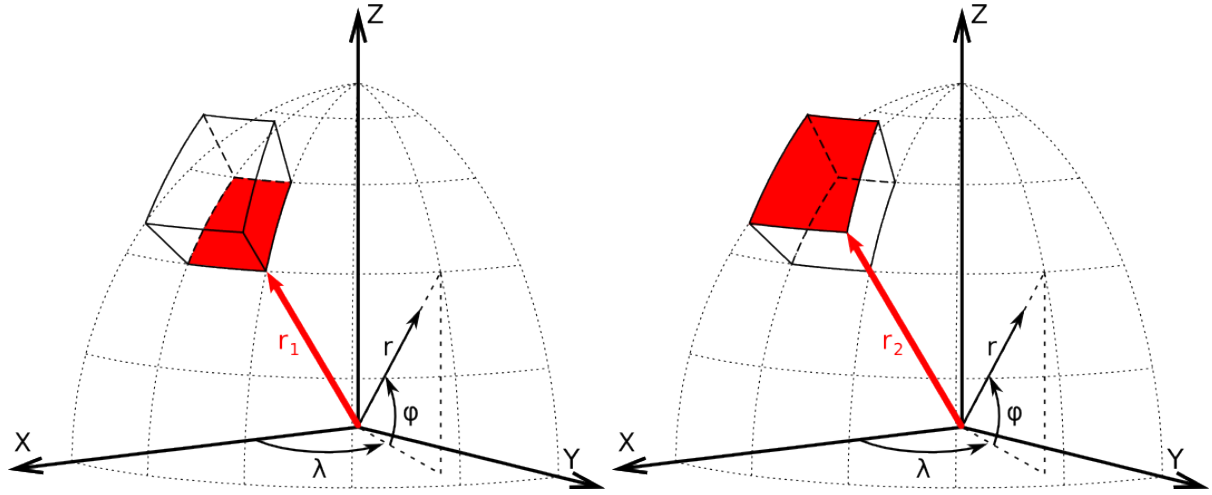
1. 2 meridians, λ_1 and λ_2



2. 2 parallels, ϕ_1 and ϕ_2



3. 2 spheres of radii r_1 and r_2



Original images (licensed CC-BY) at doi:10.6084/m9.figshare.1495537.

About coordinate systems

The figure bellow shows a tesseroid, the global coordinate system (X, Y, Z), and the local coordinate system (x, y, z) of a point P.

The global system has origin on the center of the Earth and Z axis aligned with the Earth's mean rotation axis. The X and Y axis are contained on the equatorial parallel with X intercepting the mean Greenwich meridian and Y completing a right-handed system.

The local system has origin on the computation point P. It's z axis is oriented along the radial direction and points away from the center of the Earth. The x and y axis are contained on a plane normal to the z axis. x points North and y East.

The gravitational attraction and gravity gradient tensor of a tesseroid are calculated with respect to the local coordinate system of the computation point P.

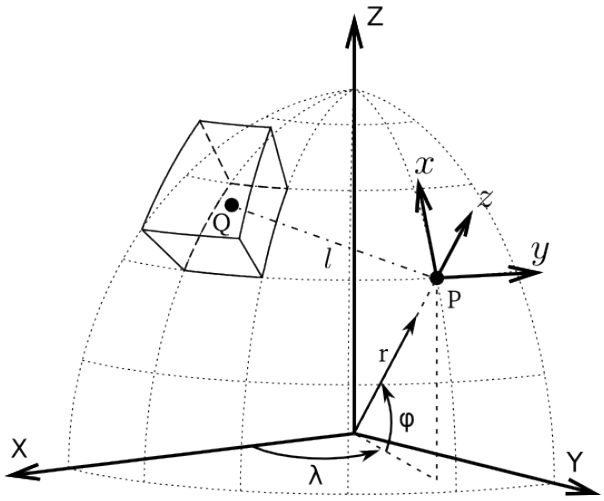


Fig. 3.1: View of a tesseroid, the integration point Q, the global coordinate system (X, Y, Z), the computation P and it's local coordinate system ($x,\ y,\ z$). $r,\ \phi,\ \lambda$ are the radius, latitude, and longitude, respectively, of point P. Original image (licensed CC-BY) at doi:10.6084/m9.figshare.1495525.

Warning: The g_z component is an exception to this. In order to conform with the regular convention of z-axis pointing toward the center of the Earth, this component **ONLY** is calculated with an inverted z axis. This way, gravity anomalies of tesseroids with positive density are positive, not negative.

Gravitational fields of a tesseroid

The gravitational potential of a tesseroid can be calculated using the formula

$$V(r, \phi, \lambda) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{1}{\ell} \kappa \, dr' d\phi' d\lambda'$$

The gravitational attraction can be calculated using the formula (Grombein et al., 2013):

$$g_\alpha(r, \phi, \lambda) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{\Delta_\alpha}{\ell^3} \kappa \, dr' d\phi' d\lambda' \quad \alpha \in \{x, y, z\}$$

The gravity gradients can be calculated using the general formula (Grombein et al., 2013):

$$g_{\alpha\beta}(r, \phi, \lambda) = G\rho \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} I_{\alpha\beta}(r', \phi', \lambda') \, dr' d\phi' d\lambda' \quad \alpha, \beta \in \{x, y, z\}$$

$$I_{\alpha\beta}(r', \phi', \lambda') = \left(\frac{3\Delta_\alpha \Delta_\beta}{\ell^5} - \frac{\delta_{\alpha\beta}}{\ell^3} \right) \kappa \quad \alpha, \beta \in \{x, y, z\}$$

where ρ is density, $\{x, y, z\}$ correspond to the local coordinate system of the computation point P (see *the tesseroid figure*), $\delta_{\alpha\beta}$ is the Kronecker delta, and

$$\begin{aligned}
\Delta_x &= r' K_\phi \\
\Delta_y &= r' \cos\phi' \sin(\lambda' - \lambda) \\
\Delta_z &= r' \cos\psi - r \\
\ell &= \sqrt{r'^2 + r^2 - 2r'r\cos\psi} \\
\cos\psi &= \sin\phi \sin\phi' + \cos\phi \cos\phi' \cos(\lambda' - \lambda) \\
K_\phi &= \cos\phi \sin\phi' - \sin\phi \cos\phi' \cos(\lambda' - \lambda) \\
\kappa &= r'^2 \cos\phi'
\end{aligned}$$

ϕ is latitude, λ is longitude, and r is radius.

Note: The **gravitational attraction** and **gravity gradient tensor** are calculated with respect to (x, y, z), the **local coordinate system** of the computation point P.

Numerical integration

The above integrals are solved using the Gauss-Legendre Quadrature rule (Asgharzadeh et al., 2007):

$$g_{\alpha\beta}(r, \phi, \lambda) \approx G\rho \frac{(\lambda_2 - \lambda_1)(\phi_2 - \phi_1)(r_2 - r_1)}{8} \sum_{k=1}^{N^\lambda} \sum_{j=1}^{N^\phi} \sum_{i=1}^{N^r} W_i^r W_j^\phi W_k^\lambda I_{\alpha\beta}(r'_i, \phi'_j, \lambda'_k) \; \alpha, \beta \in \{1, 2, 3\}$$

where W_i^r, W_j^ϕ, and W_k^λ are weighting coefficients and N^r, N^ϕ, and N^λ are the number of quadrature nodes (i.e., the order of the quadrature), for the radius, latitude, and longitude, respectively.

Tesseroids implements a modified version the adaptive discretization algorithm of Li et al (2011). This helps guarantee that the numerical integration will achieve a maximum error of 0.1%.

Warning: The integration error may be larger than this if the computation points are closer than 1km of the tesseroids. This effect is more significant in the gravity gradient components.

Gravitational fields of a prism in spherical coordinates

The gravitational potential and its first and second derivatives for the right rectangular prism can be calculated in Cartesian coordinates using the formula of Nagy et al. (2000).

However, several transformations have to made in order to calculate the fields of a prism in a global coordinate system using spherical coordinates (see *this figure*).

The formula of Nagy et al. (2000) require that the computation point be given in the Cartesian coordinates of the prism (x^*, y^*, z^* in *this figure*). Therefore, we must first transform the spherical coordinates (r, ϕ, λ) of the computation point P into x^*, y^*, z^*. This means that we must convert vector $\bar{e}$ (from *this other figure*) to the coordinate system of the prism. We must first obtain vector $\bar{e}$ in the global Cartesian coordinates (X, Y, Z):

$$\bar{e}^g = \bar{E} - \bar{E}^*$$

where $\bar{e}^g$ is the vector $\bar{e}$ in the global Cartesian coordinates and

$$\bar{E} = \begin{bmatrix} r\cos\phi\cos\lambda \\ r\cos\phi\sin\lambda \\ r\sin\phi \end{bmatrix}$$

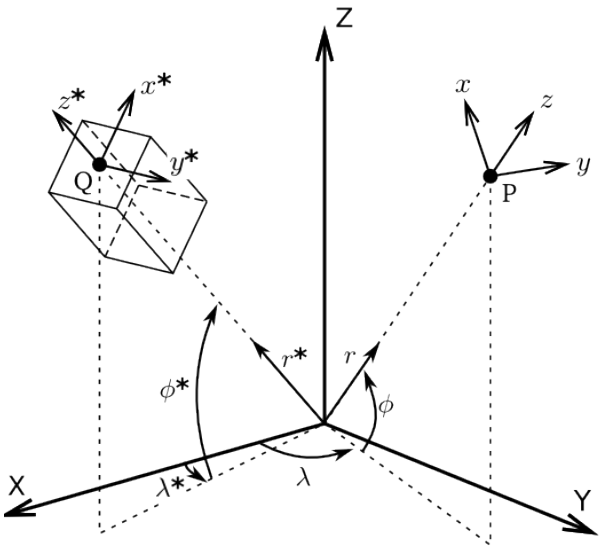


Fig. 3.2: View of a right rectangular prism with its corresponding local coordinate system ($x^*,\ y^*,\ z^*$), the global coordinate system (X, Y, Z), the computation P and it's local coordinate system ($x,\ y,\ z$). $r,\ \phi,\ \lambda$ are the radius, latitude, and longitude, respectively.

$$\bar{E}^* = \begin{bmatrix} r^* \cos\phi^* \cos\lambda^* \\ r^* \cos\phi^* \sin\lambda^* \\ r^* \sin\phi^* \end{bmatrix}$$

Next, we transform $\bar{e}^g$ to the local Cartesian system of the prism by

$$\bar{e} = \underbrace{\bar{\bar{P}}_y \bar{\bar{R}}_y(90^\circ - \phi^*) \bar{\bar{R}}_z(180^\circ - \lambda^*)}_{\bar{\bar{W}}} \bar{e}^g$$

where $\bar{\bar{P}}_y$ is a deflection matrix of the y axis, $\bar{\bar{R}}_y$ and $\bar{\bar{R}}_z$ are counterclockwise rotation matrices around the y and z axis, respectively (see Wolfram MathWorld).

$$\bar{\bar{P}}_y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\bar{\bar{R}}_y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

$$\bar{\bar{R}}_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\bar{\bar{W}} = \begin{bmatrix} \cos(90^\circ - \phi^*)\cos(180^\circ - \lambda^*) & -\cos(90^\circ - \phi^*)\sin(180^\circ - \lambda^*) & \sin(90^\circ - \phi^*) \\ -\sin(180^\circ - \lambda^*) & -\cos(180^\circ - \lambda^*) & 0 \\ -\sin(90^\circ - \phi^*)\cos(180^\circ - \lambda^*) & \sin(90^\circ - \phi^*)\sin(180^\circ - \lambda^*) & \cos(90^\circ - \phi^*) \end{bmatrix}$$

Which gives us

$$\bar{e} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

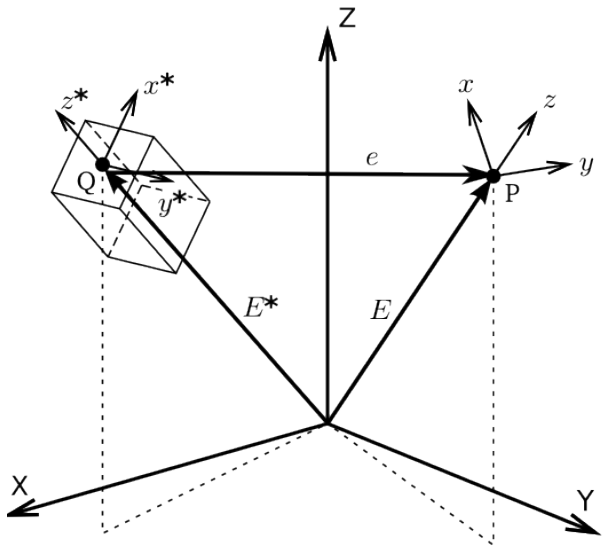


Fig. 3.3: The position vectors involved in the coordinate transformations. $\bar{E}^*$ is the position vector of point Q in the global coordinate system, $\bar{E}$ is the position vector of point P in the global coordinate system, and $\bar{e}$ is the position vector of point P in the local coordinate system of the prism ($x^*,\ y^*,\ z^*$).

Note: Nagy et al. (2000) use the z axis pointing down, so we still need to invert the sign of z.

Vector $\bar{e}$ can then be used with the Nagy et al. (2000) formula. These formula give us the gravitational attraction and the gravity gradient tensor calculated with respect to the coordinate system of the prism (i.e., $x^*,\ y^*,\ z^*$). However, we need them in the coordinate system of the observation point P, where they are measured by GOCE and calculated for the tesseroids. We perform these transformations via the global Cartesian system (tip: the rotation matrices are orthogonal). $\bar{g}^*$ is the gravity vector in the coordinate system of the prism, $\bar{g}^g$ is the gravity vector in the global coordinate system, and $\bar{g}$ is the gravity vector in the coordinate system of computation point P.

$$\bar{g}^g = \bar{\bar{R}}_z(\lambda^* - 180°)\bar{\bar{R}}_y(\phi^* - 90°)\bar{\bar{P}}_y\bar{g}^*$$

$$\bar{g} = \bar{\bar{P}}_y\bar{\bar{R}}_y(90° - \phi)\bar{\bar{R}}_z(180° - \lambda)\bar{g}^g$$

$$\bar{g} = \bar{\bar{P}}_y\bar{\bar{R}}_y(90° - \phi)\underbrace{\bar{\bar{R}}_z(180° - \lambda)\bar{\bar{R}}_z(\lambda^* - 180°)}_{\bar{\bar{R}}_z(\lambda^* - \lambda)}\bar{\bar{R}}_y(\phi^* - 90°)\bar{\bar{P}}_y\bar{g}^*$$

$$\bar{g} = \bar{\bar{R}}\bar{g}^*$$

$$\bar{\bar{R}} = \bar{\bar{P}}_y\bar{\bar{R}}_y(90° - \phi)\bar{\bar{R}}_z(\lambda^* - \lambda)\bar{\bar{R}}_y(\phi^* - 90°)\bar{\bar{P}}_y$$

$$\bar{\bar{R}} = \begin{bmatrix} \cos\beta\cos\alpha\cos\gamma - \sin\alpha\sin\gamma & \sin\beta\cos\alpha & \cos\beta\cos\alpha\sin\gamma + \sin\alpha\cos\gamma \\ -\sin\beta\cos\gamma & \cos\beta & -\sin\beta\sin\gamma \\ -\cos\beta\sin\alpha\cos\gamma - \cos\alpha\sin\gamma & -\sin\beta\sin\alpha & -\cos\beta\sin\alpha\sin\gamma + \cos\alpha\cos\gamma \end{bmatrix}$$

$$\bar{\bar{R}} = \begin{bmatrix} \cos\beta\sin\phi\sin\phi^* + \cos\phi\cos\phi^* & \sin\beta\sin\phi & -\cos\beta\sin\phi\cos\phi^* + \cos\phi\sin\phi^* \\ -\sin\beta\sin\phi^* & \cos\beta & \sin\beta\cos\phi^* \\ -\cos\beta\cos\phi\sin\phi^* + \sin\phi\cos\phi^* & -\sin\beta\cos\phi & \cos\beta\cos\phi\cos\phi^* + \sin\phi\sin\phi^* \end{bmatrix}$$

where

$$\alpha = 90^\circ - \phi$$
$$\beta = \lambda^* - \lambda$$
$$\gamma = \phi^* - 90^\circ$$
$$\cos\alpha = \sin\phi$$
$$\sin\alpha = \cos\phi$$
$$\cos\gamma = \sin\phi^*$$
$$\sin\gamma = -\cos\phi^*$$

Likewise, transformation for the gravity gradient tensor T is

$$\bar{\bar{T}} = \bar{\bar{R}}\bar{\bar{T}}^*\bar{\bar{R}}^T$$

Recommended reading

- Smith et al. (2001)
- Wild-Pfeiffer (2008)

References

Asgharzadeh, M. F., R. R. B. von Frese, H. R. Kim, T. E. Leftwich, and J. W. Kim (2007), Spherical prism gravity effects by Gauss-Legendre quadrature integration, Geophysical Journal International, 169(1), 1-11, doi:10.1111/j.1365-246X.2007.03214.x.

Grombein, T.; Seitz, K.; Heck, B. (2013), Optimized formulas for the gravitational field of a tesseroid, Journal of Geodesy, doi: 10.1007/s00190-013-0636-1

Li, Z., T. Hao, Y. Xu, and Y. Xu (2011), An efficient and adaptive approach for modeling gravity effects in spherical coordinates, Journal of Applied Geophysics, 73(3), 221–231, doi:10.1016/j.jappgeo.2011.01.004.

Nagy, D., G. Papp, and J. Benedek (2000), The gravitational potential and its derivatives for the prism, Journal of Geodesy, 74(7-8), 552-560, doi:10.1007/s001900000116.

Nagy, D., G. Papp, and J. Benedek (2002), Corrections to "The gravitational potential and its derivatives for the prism," Journal of Geodesy, 76(8), 475-475, doi:10.1007/s00190-002-0264-7.

Smith, D. A., D. S. Robertson, and D. G. Milbert (2001), Gravitational attraction of local crustal masses in spherical coordinates, Journal of Geodesy, 74(11-12), 783-795, doi:10.1007/s001900000142.

Wild-Pfeiffer, F. (2008), A comparison of different mass elements for use in gravity gradiometry, Journal of Geodesy, 82(10), 637-653, doi:10.1007/s00190-008-0219-8.

Using Tesseroids

This is a tutorial about how to use the Tesseroids package. It is a work-in-progress but I have tried to be as complete as possible. If you find that anything is missing, or would like something explained in more detail, please submit a bug report (it's not that hard).

Any further questions and comments can be e-mail directly to me (leouieda [at] gmail [dot] com).

If you don't find what you're looking for here, the *cookbook* contains several example recipes of using Tesseroids.

A note about heights and units

In order to have a single convention, the word "height" means "height above the Earths surface" and are interpreted as positive up and negative down (i.e., oriented with the z axis of the Local coordinate system). Also, all input units are in SI and decimal degrees. Output of tesspot is in SI, tessgx, tessgy, and tessgz are in mGal, and the tensor components in Eotvos. All other output is also in SI and decimal degrees.

Getting help information

All programs accept the -h and –version flags. -h will print a help message describing the usage, input and output formats and options accepted. –version prints version and license information about the program.

Program tessdefaults prints the default values of constants used in the computations such as: mean Earth radius, pi, gravitational constant, etc.

Computing the gravitational effect of a tesseroid

The tesspot, tessgx, tessgy, tessgz, tessgxx, etc. programs calculate the combined effect of a list of tesseroids on given computation points. The computation points are passed via standard input and do NOT have to be in a regular grid. This allows, for example, computation on points where data was measured. The values calculated are put in the last column of the input points and printed to standard output.

For example, if calculating gz on these points:

```
lon1 lat1 height1 value1 othervalue1
lon2 lat2 height2 value2 othervalue2
...
lonN latN heightN valueN othervalueN
```

the output would look something like:

```
lon1 lat1 height1 value1 othervalue1 gz1
lon2 lat2 height2 value2 othervalue2 gz2
...
lonN latN heightN valueN othervalueN gzN
```

The input model file should contain one tesseroid per line and have columns formatted as:

```
W E S N HEIGHT_OF_TOP HEIGHT_OF_BOTTOM DENSITY
```

HEIGHT_OF_TOP and HEIGHT_OF_BOTTOM are positive if the above the Earth's surface and negative if bellow.

Note: Remember that HEIGHT_OF_TOP > HEIGHT_OF_BOTTOM!

Use the command line option -h to view a list of all commands available.

Example:

Calculate the field of a tesseroid model having verbose printed and logged to file `gz.log` and GLQ order 3/3/3. The computation points are in `points.txt` and the output will be placed in `gz_data.txt`:

```
tessgz modelfile.txt -v -lgz.log -o3/3/3 < points.txt > gz_data.txt
```

The -a flag

The -a flag on tesspot, tessgx, tessgxx, etc., programs **disables** the automatic recursive dividing of tesseroids to maintain the GLQ accuracy. As a general rule, the tesseroid should be no bigger than a ratio times the distance from the computation point (program tessdefaults prints the value of the size ratios used). The programs automatically break the tesseroids when this criterion is breached. This means that the computations can be performed with the default GLQ order 2/2/2, which is much faster, and still maintain correctness.

Warning: It is strongly recommended that you don't use this flag unless you know what you are doing! It is also recommended that you keep 2/2/2 order always.

Verbose and logging to files

The -v flag enables printing of information messages to the default error stream (stderr). If omitted, only error messages will appear. The -l flag enables logging of information and error messages to a file.

Comments and provenance information

Comments can be inserted into input files by placing a "#" character at the start of a line. All comment lines are ignored. All programs pass on (print) the comment lines of the input to the output. All programs insert comments about the provenance of their results (where they came from) to their output. These include names of input files, version of program used, date, etc.

Generating regular grids

Included in the package is program tessgrd, which creates a regular grid of points and prints them to standard output.

Example

To generate a regular grid of 100 x 100 points, in the are -10/10/-10/10 degrees, at a height of 250 km:

```
tessgrd -r-10/10/-10/10 -b100/100 -z250e03 -v > points.txt
```

Automatic model generation

As of version 1.0, Tesseroids includes program tessmodgen for automatically generating a tesseroid model from a map of an interface. The interface can be any surface deviating from a reference level. For example, topography (a DEM) deviates from 0, a Moho map deviates from a mean crustal thickness, etc. This program takes as input a **REGULAR** grid with longitude, latitude and height values of the interface. Each tesseroid is generated with a grid point at the center of it's top face. The top and bottom faces of the tesseroid are defined as:

- Top = Interface and Bottom = Reference if the interface is above the reference
- Top = Reference and Bottom = Interface if the interface is bellow the reference

The density RHO of the tesseroids can be passed using the -d option. This will assign a density value of RHO, when the interface is above the reference, and a value of -RHO if the interface is bellow the reference. Alternatively, the density of each tesseroid can be passed as a forth column on the input grid. As with the -d option, if the interface is bellow the reference, the density value will be multiplied by -1! Also, an error will occur if both a forth column and the -d option are passed!

Example:

To generate a tesseroid model from a Digital Elevation Model (DEM) with 1 x 1 degree resolution using a density of 2670 km/m^3:

```
tessmodgen -s1/1 -d2670 -z0 -v < dem_file.txt > dem_tess_model.txt
```

Calculating the total mass of a model

The tessmass program can be used to compute the total mass of a given tesseroid model. If desired, a density range can be given and only tesseroids that fall within the given range will be used in the calculation.

Example:

To calculate the total mass of all tesseroids in `model.txt` with density between 0 and 1 g/cm^3:

```
tessmass -r0/1000 < model.txt
```

Computing the effect of rectangular prisms in Cartesian coordinates

Tesseroids 1.0 also introduced programs to calculate the gravitational effect of right rectangular prisms in Cartesian coordinates. This is done using the formula of Nagy et al. (2000). The programs are prismpot, prismgx, prismgy, prismgz, prismgxx, etc. Input and output for these programs is very similar to that of the tesspot, tessgx, etc., programs. Computation points are read from standard input and the prism model is read from a file. The model file should have the column format:

```
X1 X2 Y1 Y2 Z1 Z2 DENSITY
```

Note: As in Nagy et al. (2000), the coordinate system for the rectangular prism calculations has X axis pointing North, Y axis pointing East and Z axis pointing Down. This is important to note because it differs from the convention adopted for the tesseroids. In practice, this means that the g_{xz} and g_{yz} components of the prism and tesseroid will have different signs. This will not be such for the g_z component, though, because the convention for tesseroids is to have Z axis Down for this component only. See the *Theoretical background* section for more details on this.

Piping

Tesseroids was designed with the Unix philosophy in mind:

```
Write programs that do one thing and do it well.
Write programs to work together.
Write programs to handle text streams, because that is a universal interface.
```

Therefore, all tessg* programs and tessgrd can be piped together to calculate many components on a regular grid.

Example:

Given a tesseroids file `model.txt` as follows:

```
-1 1 -1 1 0 -10e03 -500
```

Running the following would calculate gz and gradient tensor of tesseroids in `model.txt` of a regular grid from -5W to 5E and -5S to 5N on 100x100 points at 250 km height. And the best of all is that it is done in parallel! If your system has multiple cores, this would mean a great increase in the computation time. All information regarding the

computations will be logged to files gz.log, gxx.log, etc. These should include the information about how many times the tesseroid had to be split into smaller ones to guarantee GLQ accuracy:

```
tessgrd -r-5/5/-5/5 -b100/100 -z250e03 | \
tessgz model.txt -lgz.log | \
tessgxx model.txt -lgxx.log | \
tessgxy model.txt -lgxy.log | \
tessgxz model.txt -lgxz.log | \
tessgyy model.txt -lgyy.log | \
tessgyz model.txt -lgyz.log | \
tessgzz model.txt -lgzz.log > output.txt
```

Cookbook

The following recipes can be found in the `cookbook` folder that comes with your *Tesseroids* download (along with shell and batch scripts and sample output):

Calculate the gravity gradient tensor from a DEM

This example demonstrates how to calculate the gravity gradient tensor (GGT) due to topographic masses using tesseroids.

To do that we need:

1. A DEM file with lon, lat, and height information;
2. Assign correct densities to continents and oceans (we'll be using a little Python for this);
3. Convert the DEM information into a tesseroid model;
4. Calculate the 6 components of the GGT;

The file `dem_brasil.sh` is a small shell script that executes all the above (we'll be looking at each step in more detail):

```
#!/bin/bash

# First, insert the density information into
# the DEM file using the Python script.
python dem_density.py dem.xyz > dem-dens.txt

# Next, use the modified DEM with tessmodgen
# to create a tesseroid model
tessmodgen -s0.166667/0.166667 -z0 -v < dem-dens.txt \
> dem-tess.txt

# Calculate the GGT on a regular grid at 250km
# use the -l option to log the processes to files
# (usefull to diagnose when things go wrong)
# The output is dumped to dem-ggt.txt
tessgrd -r-60/-45/-30/-15 -b50/50 -z250e03 | \
tessgxx dem-tess.txt -lgxx.log | \
tessgxy dem-tess.txt -lgxy.log | \
tessgxz dem-tess.txt -lgxz.log | \
tessgyy dem-tess.txt -lgyy.log | \
```

```
21 tessgyz dem-tess.txt -lgyz.log | \
22 tessgzz dem-tess.txt -lgzz.log -v > dem-ggt.txt
```

Why Python

Python is a modern programming language that is very easy to learn and extremely productive. We'll be using it to make our lives a bit easier during this example but it is by no means a necessity. The same thing could have been accomplished with Unix tools and the Generic Mapping Tools (GMT) or other plotting program.

If you have interest in learning Python we recommend the excelent video lectures in the Software Carpentry course. There you will also find lectures on various scientific programming topics. I strongly recommend taking this course to anyone who works with scientific computing.

The DEM file

For this example we'll use ETOPO1 for our DEM. The file `dem.xyz` contains the DEM as a 10' grid. Longitude and latitude are in decimal degrees and heights are in meters. This is what the DEM file looks like (first few lines):

```
 1 # This is the DEM file from ETOPO1 with 10' resolution
 2 # points in longitude: 151
 3 # Columns:
 4 #  lon          lat       height(m)
 5 -65.000000 -10.000000    157
 6  -64.833333 -10.000000    168
 7  -64.666667 -10.000000    177
 8  -64.500000 -10.000000    197
 9  -64.333333 -10.000000    144
10  -64.166667 -10.000000    178
```

Notice that *Tesseroids* allows you to include comments in the files by starting a line with `#`. *This figure* shows the DEM ploted in pseudocolor. The red rectangle is the area in which we'll be calculating the GGT.

Assigning densities

Program tessmodgen allows us to provide the density value of each tesseroid through the DEM file. All we have to do is insert an extra column in the DEM file with the density values of the tesseroids that will be put on each point. This way we can have the continents with 2.67 g/cm3 and oceans with 1.67 g/cm3. Notice that the density assigned to the oceans is positive! This is because the DEM in the oceans will have heights bellow our reference (h = 0km) and tessmodgen will automatically invert the sign of the density values if a point is bellow the reference.

We will use the Python script `dem_density.py` to insert the density values into our DEM and save the result to `dem-dens.txt`:

```
3 # First, insert the density information into
4 # the DEM file using the Python script.
5 python dem_density.py dem.xyz > dem-dens.txt
```

If you don't know Python, you can easily do this step in any other language or even in Excel. This is what the `dem_density.py` script looks like:

```
 1 """
 2 Assign density values for the DEM points.
 3 """
 4 import sys
 5 import numpy
```

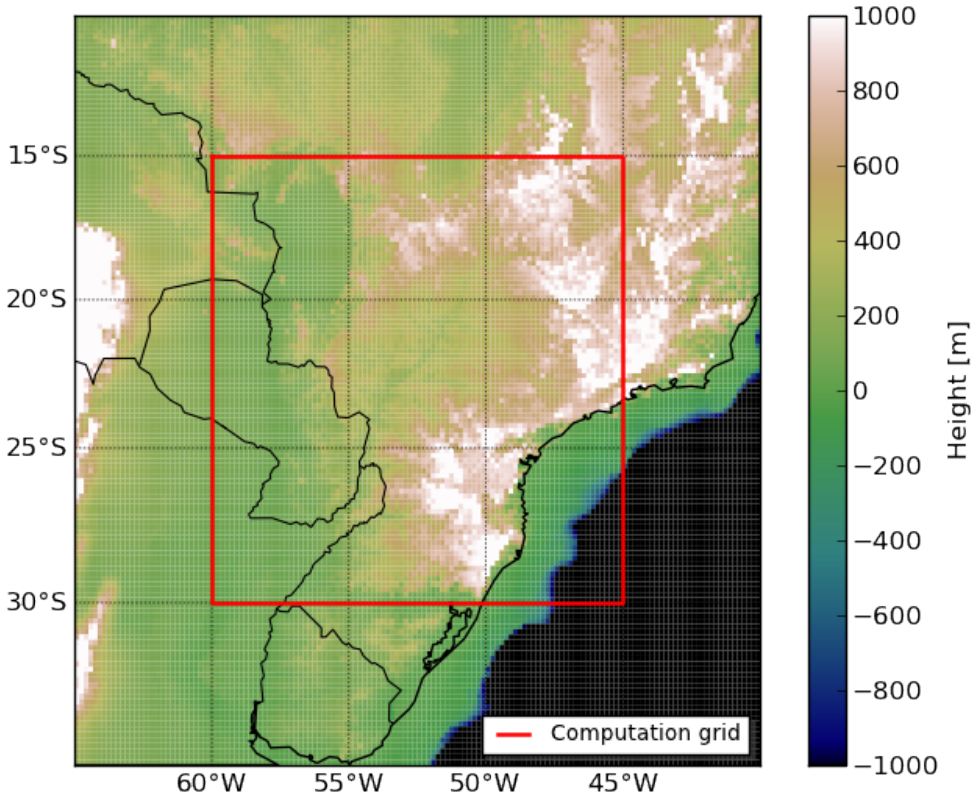


Fig. 3.4: The ETOPO1 10' DEM of the Parana Basin, southern Brasil.

```
 6
 7 lons, lats, heights = numpy.loadtxt(sys.argv[1], unpack=True)
 8
 9 for i in xrange(len(heights)):
10     if heights[i] >=0:
11         print "%lf %lf %lf %lf" % (lons[i], lats[i], heights[i], 2670.0)
12     else:
13         print "%lf %lf %lf %lf" % (lons[i], lats[i], heights[i], 1670.0)
```

The result is a DEM file with a forth column containing the density values (see *this figure*):

```
 1 -65.000000 -10.000000 157.000000 2670.000000
 2 -64.833333 -10.000000 168.000000 2670.000000
 3 -64.666667 -10.000000 177.000000 2670.000000
 4 -64.500000 -10.000000 197.000000 2670.000000
 5 -64.333333 -10.000000 144.000000 2670.000000
 6 -64.166667 -10.000000 178.000000 2670.000000
 7 -64.000000 -10.000000 166.000000 2670.000000
 8 -63.833333 -10.000000 164.000000 2670.000000
 9 -63.666667 -10.000000 189.000000 2670.000000
10 -63.500000 -10.000000 210.000000 2670.000000
```

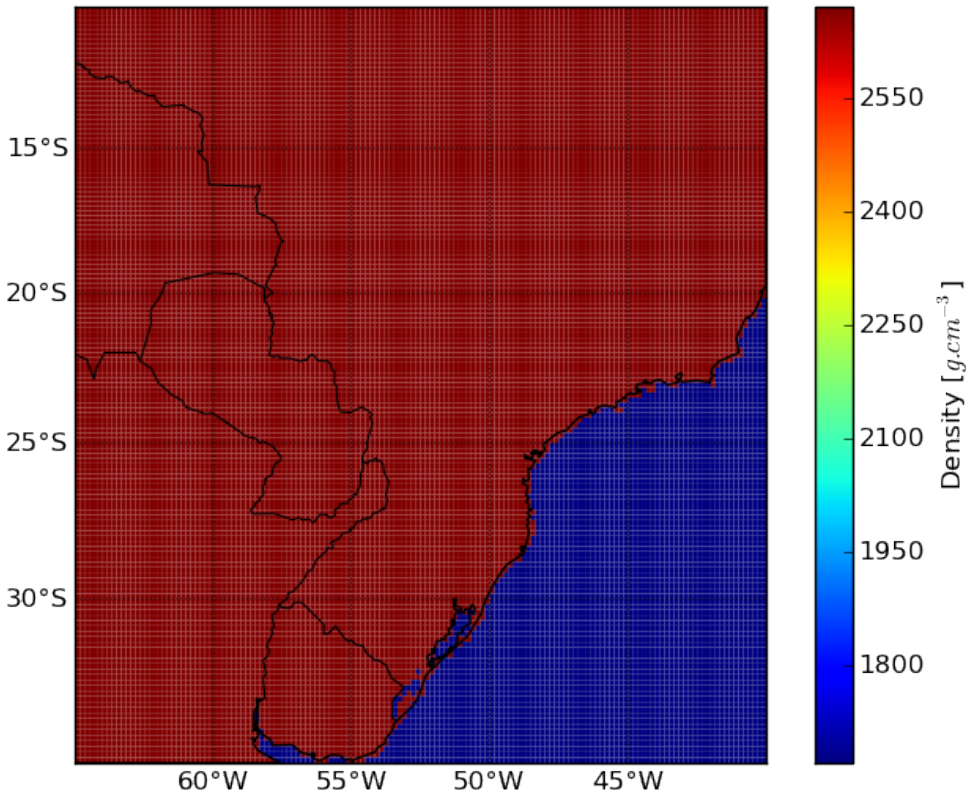


Fig. 3.5: Density values. 2.67 g/cm3 in continents and 1.67 g/cm3 in the oceans.

Making the tesseroid model

Next, we'll use our new file `dem-dens.txt` and program tessmodgen to create a tesseroid model of the DEM:

```
 7 # Next, use the modified DEM with tessmodgen
 8 # to create a tesseroid model
 9 tessmodgen -s0.166667/0.166667 -z0 -v < dem-dens.txt \
10 > dem-tess.txt
```

tessmodgen places a tesseroid on each point of the DEM. The bottom of the tesseroid is placed on a reference level and the top on the DEM. If the height of the point is bellow the reference, the top and bottom will be inverted so that the tesseroid isn't upside-down. In this case, the density value of the point will also have its sign changed so that you get the right density values if modeling things like the Moho. For topographic masses, the reference surface is h = 0km (argument `-z`). The argument `-s` is used to specify the grid spacing (10') which will be used to set the horizontal dimensions of the tesseroid. Since we didn't pass the `-d` argument with the density of the tesseroids, tessmodgen will expect a fourth column in the input with the density values.

The result is a tesseroid model file that should look somthing like this:

```
 1 # Tesseroid model generated by tessmodgen 1.1dev:
 2 #   local time: Wed May  9 19:08:12 2012
 3 #   grid spacing: 0.166667 deg lon / 0.166667 deg lat
 4 #   reference level (depth): 0
 5 #   density: read from input
 6 -65.0833335 -64.9166665 -10.0833335 -9.9166665 157 0 2670
 7 -64.9166665 -64.7499995 -10.0833335 -9.9166665 168 0 2670
 8 -64.7500005 -64.5833335 -10.0833335 -9.9166665 177 0 2670
 9 -64.5833335 -64.4166665 -10.0833335 -9.9166665 197 0 2670
10 -64.4166665 -64.2499995 -10.0833335 -9.9166665 144 0 2670
```

and for the points in the ocean (negative height):

```
9065 -40.0833335 -39.9166665 -19.9166665 -19.7499995 0 -19 -1670
```

Calculating the GGT

Tesseroids allows use of custom computation grids by reading the computation points from standard input. This way, if you have a file with lon, lat, and height coordinates and wish to calculate any gravitational field in those points, all you have to do is redirect stardard input to that file (using <). All tess* programs will calculate their respective field, append a column with the result to the input and print it to stdout. So you can pass grid files with more than three columns, as long as the first three correspond to lon, lat and height. This means that you can pipe the results from one tessg to the other and have an output file with many columns, each corresponding to a gravitational field. The main advantage of this approach is that, in most shell environments, the computation of pipes is done in parallel. So, if your system has more than one core, you can get parallel computation of GGT components with no extra effort.

For convience, we added the program tessgrd to the set of tools, which creates regular grids and print them to standard output. So if you don't want to compute on a custom grid (like us), you can simply pipe the output of tessgrd to the tess* programs:

```
12 # Calculate the GGT on a regular grid at 250km
13 # use the -l option to log the processes to files
14 # (usefull to diagnose when things go wrong)
15 # The output is dumped to dem-ggt.txt
16 tessgrd -r-60/-45/-30/-15 -b50/50 -z250e03 | \
17 tessgxx dem-tess.txt -lgxx.log | \
18 tessgxy dem-tess.txt -lgxy.log | \
19 tessgxz dem-tess.txt -lgxz.log | \
20 tessgyy dem-tess.txt -lgyy.log | \
21 tessgyz dem-tess.txt -lgyz.log | \
22 tessgzz dem-tess.txt -lgzz.log -v > dem-ggt.txt
```

The end result of this is file `dem-ggt.txt`, which will have 9 columns in total. The first three are the lon, lat and height coordinates generated by tessgrd. The next six will correspond to each component of the GGT calculated by tessgxx, tessgxy, etc., respectively. The resulting GGT is shown in *this figure*.

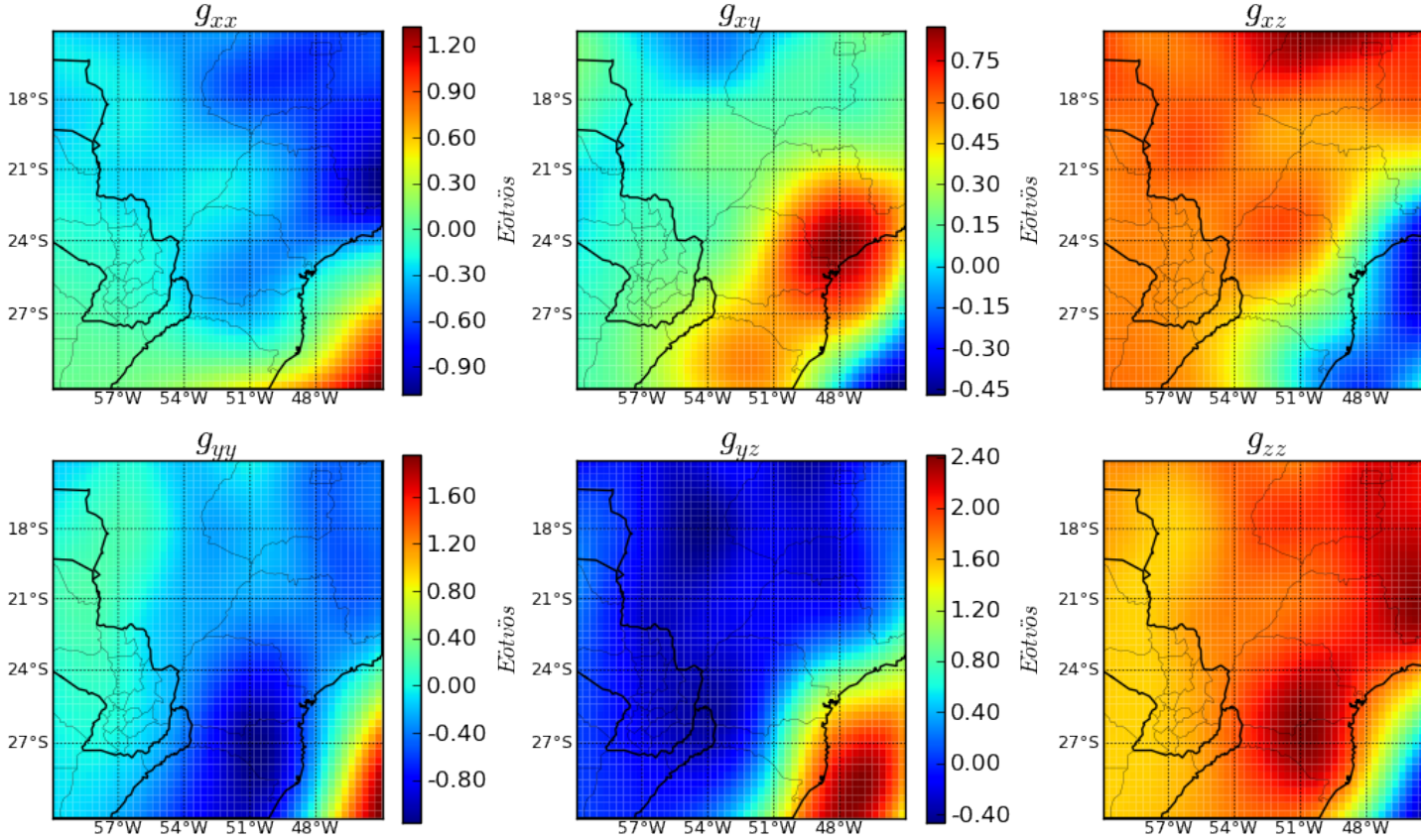


Fig. 3.6: GGT of caused by the topographic masses.

Making the plots

The plots were generated using the powerfull Python library Matplotlib. The script `plots.py` is somewhat more complicated than `dem_density.py` and requires a bit of "Python Fu". The examples in the Matplotlib website should give some insight into how it works. To hanble the map projections, we used the Basemap toolkit of Matplotlib.

Simple prism model in Cartesian coordinates

The `simple_prism.sh` script calculates the gravitational potential, gravitational attraction, and gravity gradient tensor due to a simple prism model in Cartesian coordinates:

```
#!/bin/bash

# Generate a regular grid, pipe it to all the computation programs,
# and write the result to output.txt

tessgrd -r0/20000/0/20000 -b50/50 -z1000 | \
prismpot model.txt | \
```

```
prismgx model.txt | prismgy model.txt | prismgz model.txt | \
prismgxx model.txt  | prismgxy model.txt  | \
prismgxz model.txt  | prismgyy model.txt  | \
prismgyz model.txt  | prismgzz model.txt > output.txt
```

The model file looks like this:

```
# Test prism model file
2000 5000 2000 15000 0 5000 1000
10000 18000 10000 18000 0 5000 -1000
```

The result should look like the *following* ("column" means the column of the output file).

Simple tesseroid model

The files in the folder `cookbook/simple_tess` show how to calculate the gravitational fields of a simple 2 tesseroid model at 260 km height.

For this simple setup, the model file looks like this:

```
# Test tesseroid model file
10 20 10 20 0 -50000 200
-20 -10 -20 -10 0 -30000 -500
```

The `simple_tess.sh` script performs the calculations and calls the `plot.py` script to plot the results:

```
#!/bin/bash

# Generate a regular grid, pipe it to all the computation programs,
# and write the result to output.txt

tessgrd -r-45/45/-45/45 -b101/101 -z260e03 | \
tesspot model.txt | \
tessgx model.txt | tessgy model.txt | tessgz model.txt | \
tessgxx model.txt  | tessgxy model.txt  | \
tessgxz model.txt  | tessgyy model.txt  | \
tessgyz model.txt  | tessgzz model.txt -v -llog.txt > output.txt

# Make a plot with the columns of output.txt
python plot.py output.txt 101 101
```

`tessgrd` generates a regular grid and prints that to standard output (`stdout`). The scripts pipes the grid points to `tesspot` etc. to calculate the corresponding fields. Option `-v` tells `tessgzz` to print information messages (to stderr). Option `-llog.txt` tells tessgzz to log the information plus debug messages to a file called `log.txt`.

The columns of the output file will be, respectively: longitude, latitude, height, potential, gx, gy, gz, gxx, gxy, gxz, gyy, gyz, and gzz. The result should look like this ("column" means the column of the output file):

Convert a tesseroid model to prisms and calculate in spherical coordinates

The `tess2prism.sh` script converts a tesseroid model to prisms (using `tessmodgen`) and calculates the gravitational potential, gravitational attraction, and gravity gradient tensor in spherical coordinates:

```
#!/bin/bash
```

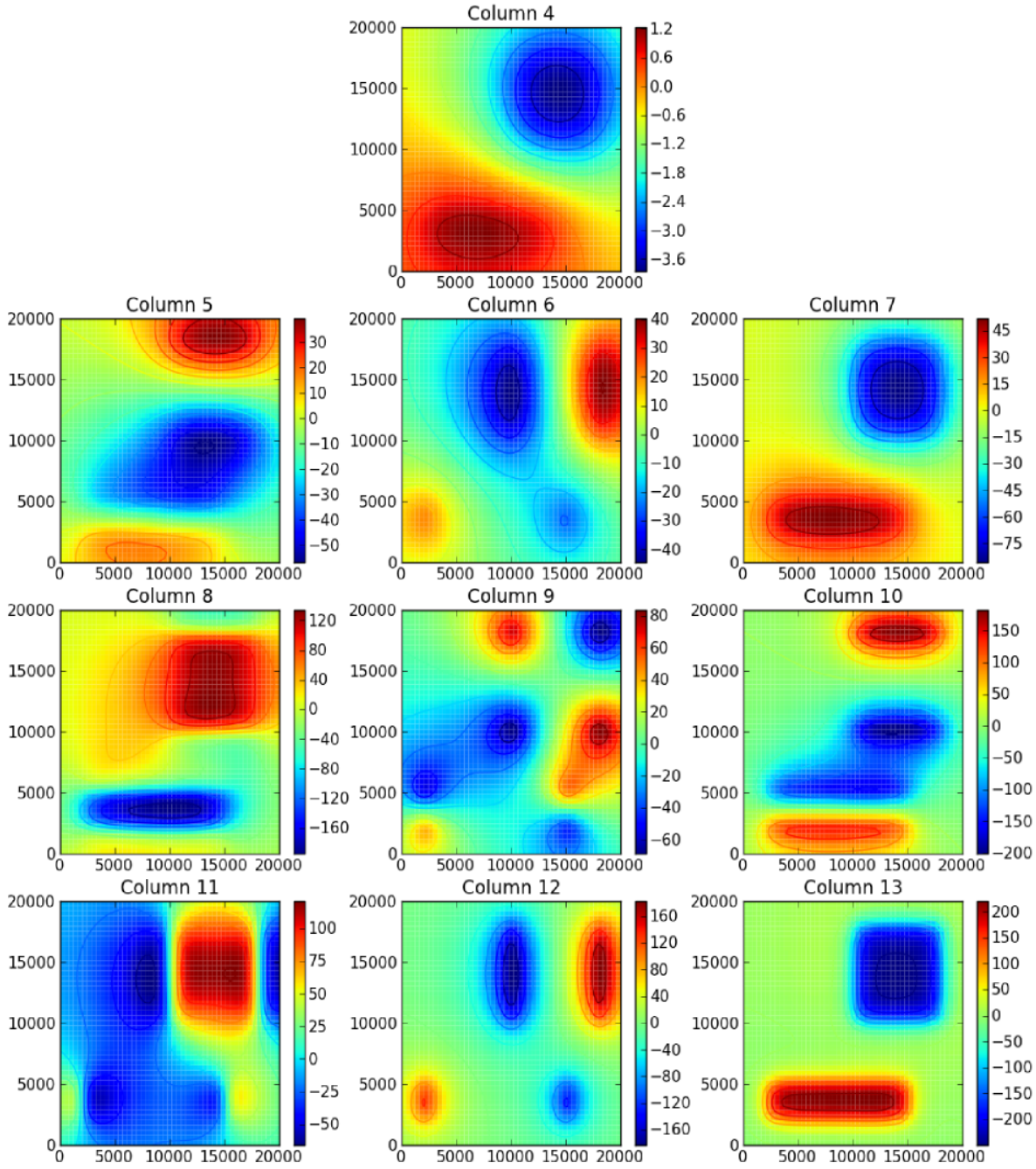


Fig. 3.7: Plot of the columns of `output.txt` generated by `simple_prism.sh`. The x and y axis are longitude and latitude, respectively.

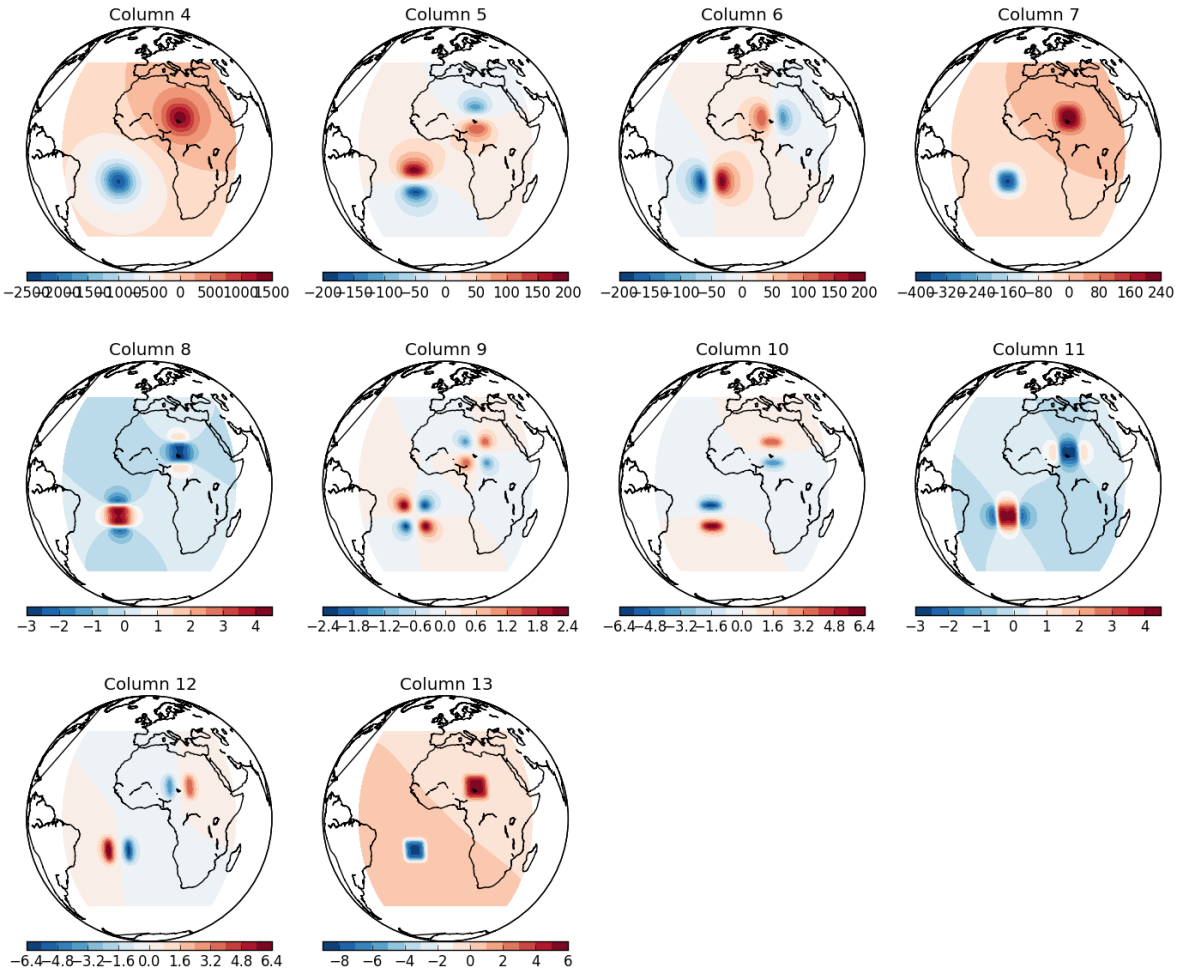


Fig. 3.8: Plot of the columns of `output.txt` generated by `simple_tess.sh`. Orthographic projection (thanks to the Basemap toolkit of matplotlib).

```
# Generate a prism model from a tesseroid model.
# Prisms will have the same mass as the tesseroids and
# associated spherical coordinates of the center of
# the top of the tesseroid.

tess2prism < tess-model.txt > prism-model.txt

# Generate a regular grid in spherical coordinates,
# pipe the grid to the computation programs,
# and dump the result on output.txt
# prismpots calculates the potential in spherical
# coordinates, prismgs calculates the full
# gravity vector, and prismggts calculates the full
# gravity gradient tensor.

tessgrd -r-160/0/-80/0 -b100/100 -z250e03 | \
prismpots prism-model.txt | \
prismgs prism-model.txt | \
prismggts prism-model.txt -v > output.txt
```

The tesseroid model file looks like this:

```
# Test tesseroid model file
-77 -75 -41 -39 0 -50000 500
-79 -77 -41 -39 0 -50000 500
-81 -79 -41 -39 0 -50000 500
-83 -81 -41 -39 0 -50000 500
-85 -83 -41 -39 0 -50000 500
```

and the converted prism model looks like this:

```
# Prisms converted from tesseroid model with tess2prism 1.1dev
#   local time: Wed May 16 14:34:47 2012
#   tesseroids file: stdin
#   conversion type: equal mass|spherical coordinates
#   format: dx dy dz density lon lat r
# Test tesseroid model file
221766.31696055 169882.854778591 50000 499.977196258595 -76 -40 6378137
221766.31696055 169882.854778591 50000 499.977196258595 -78 -40 6378137
221766.31696055 169882.854778591 50000 499.977196258595 -80 -40 6378137
221766.31696055 169882.854778591 50000 499.977196258595 -82 -40 6378137
221766.31696055 169882.854778591 50000 499.977196258595 -84 -40 6378137
```

Note that the density of prisms is altered. This is so that the tesseroid and corresponding prism have the same mass.

The result should look like the *following* ("column" means the column of the output file).

Convert a tesseroid model to prisms and calculate in Cartesian coordinates

The `tess2prism_flatten.sh` script converts a tesseroid model to prisms (using the `--flatten` flag in `tessmodgen`) and calculates the gravitational potential, gravitational attraction, and gravity gradient tensor in Cartesian coordinates:

```
#!/bin/bash

# Generate a prism model from a tesseroid model by
# flattening the tesseroids (1 degree = 111.11 km).
```

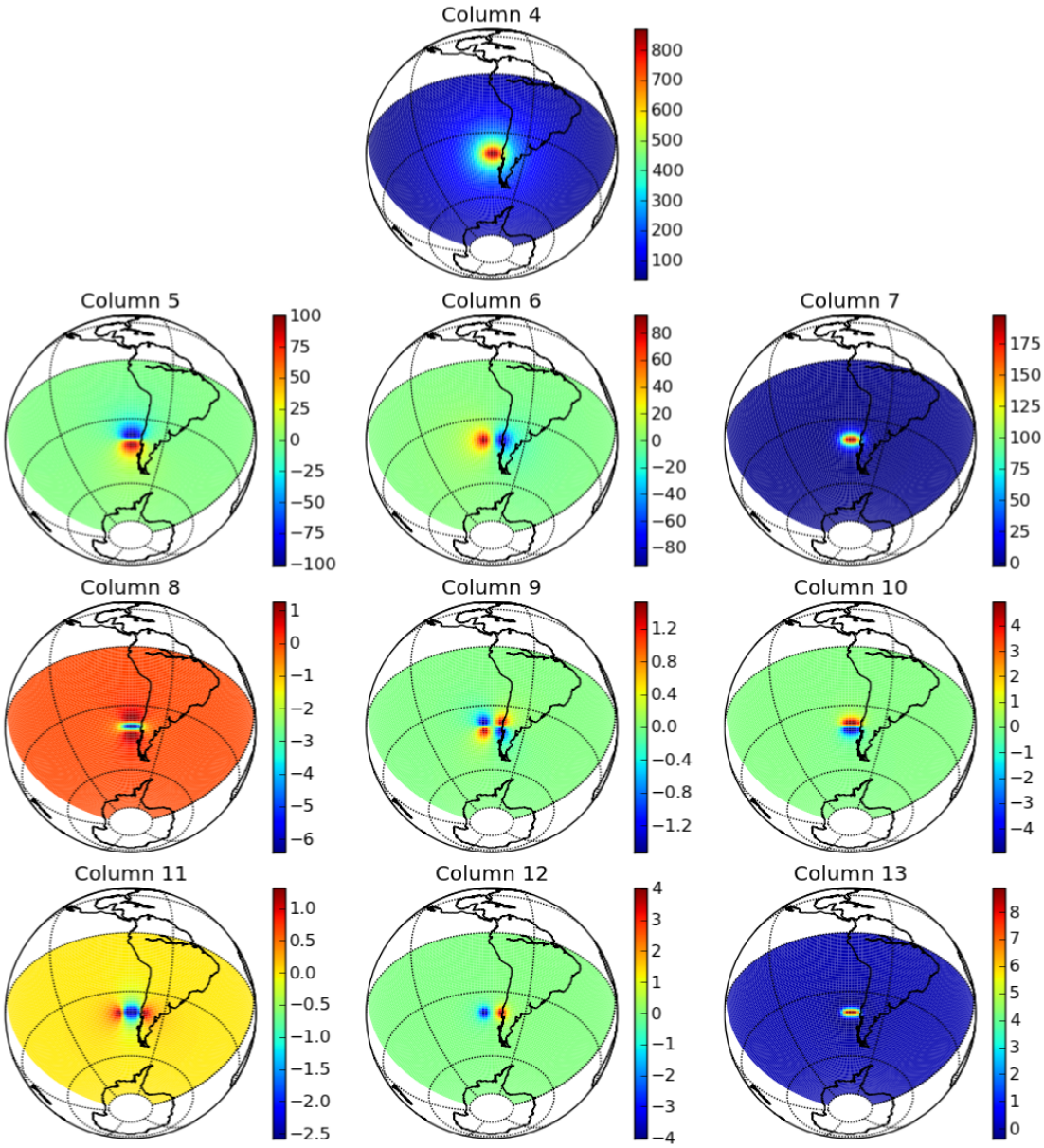


Fig. 3.9: Plot of the columns of `output.txt` generated by `tess2prism.sh`. Orthographic projection (thanks to the Basemap toolkit of matplotlib).

```
# This way the converted prisms can be used
# with the prism* programs in Cartesian coordinates.

tess2prism --flatten < tess-model.txt > prism-model.txt

# Generate a regular grid in Cartesian coordinates,
# pipe the grid to the computation programs,
# and dump the result on output.txt

tessgrd -r-3e06/3e06/-3e06/3e06 -b50/50 -z250e03 | \
prismpot prism-model.txt | \
prismgx prism-model.txt | \
prismgy prism-model.txt | \
prismgz prism-model.txt | \
prismgxx prism-model.txt  | prismgxy prism-model.txt  | \
prismgxz prism-model.txt  | prismgyy prism-model.txt  | \
prismgyz prism-model.txt  | prismgzz prism-model.txt > output.txt
```

The tesseroid model file looks like this:

```
# Test tesseroid model file
10 15 10 15 0 -30000 500
-15 -10 -10 10 0 -50000 200
-15 5 -16 -10 0 -30000 -300
```

and the converted prism model looks like this:

```
# Prisms converted from tesseroid model with tess2prism 1.1dev
#   local time: Tue May  8 14:55:02 2012
#   tesseroids file: stdin
#   conversion type: flatten
#   format: x1 x2 y1 y2 z1 z2 density
# Test tesseroid model file
1111100 1666650 1111100 1666650 0 30000 487.534658568521
-1111100 1111100 -1666650 -1111100 0 50000 198.175508383774
-1777760 -1111100 -1666650 555550 0 30000 -291.9029748328
```

Note that the density of prisms is altered. This is so that the tesseroid and corresponding prism have the same mass.

The result should look like the *following* ("column" means the column of the output file).

Using tesslayers to make a tesseroid model of a stack of layers

The `tesslayers.sh` script converts grids that define a stack of layers into a tesseroid model. It then calculates the gravitational attraction and gravity gradient tensor due to the tesseroid model:

```
#!/bin/bash

# Convert the layer grids in layers.txt to tesseroids.
# The grid spacing passed to -s is used as the size of the tesseroids,
# so be careful!
tesslayers -s0.5/0.5 -v < layers.txt > tessmodel.txt

# Now calculate the gz and tensor effect of this model at 100km height
tessgrd -r-8/8/32/48 -b50/50 -z100000 | \
tessgz tessmodel.txt | \
tessgxx tessmodel.txt | tessgxy tessmodel.txt | \
```

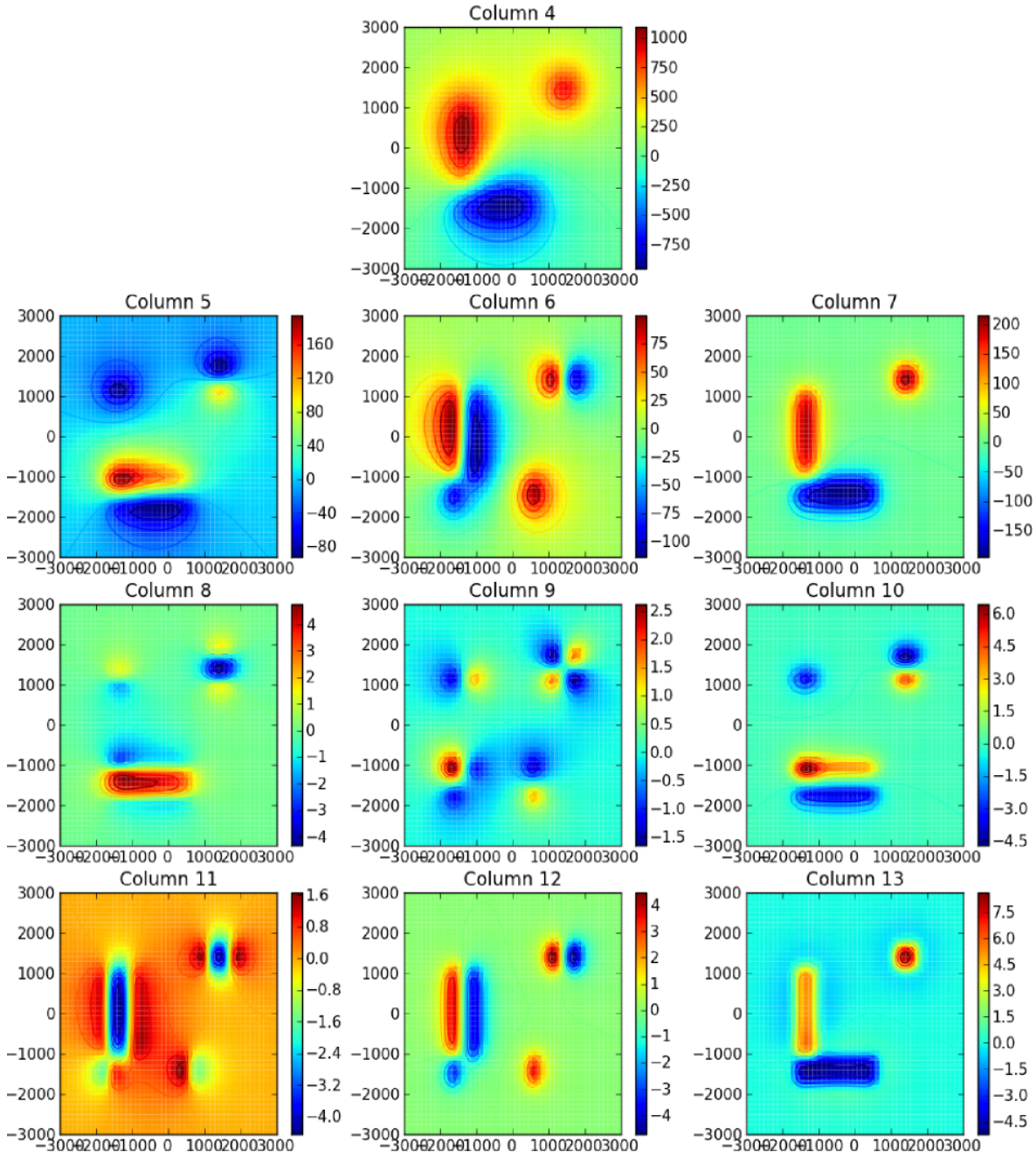


Fig. 3.10: Plot of the columns of `output.txt` generated by `tess2prism_flatten.sh`. The x and y axis are West-East and South-North, respectively, in kilometers.

```
tessgxz tessmodel.txt | tessgyy tessmodel.txt | \
tessgyz tessmodel.txt | tessgzz tessmodel.txt -v > output.txt
```

The input file `layers.txt` contains the information about the stack of layers. It is basicaly regular grids in xyz format (i.e., in columns). The first 2 columns in the file are the longitude and latitude of the grid points. Then comes a column with the height of the first layer. This is the height (with respect to mean Earth radius) of the top of stack of layers. Then comes the thickness and density of each layer. Our layer file looks like this:

```
 1 # Synthetic layer model of sediments and topography
 2 #   lon   lat   height   thickness   density
 3 -10 30 800 800.002 1900
 4 -9.5 30 800 800.006 1900
 5 -9 30 800 800.016 1900
 6 -8.5 30 800 800.042 1900
 7 -8 30 800 800.105 1900
 8 -7.5 30 800 800.248 1900
 9 -7 30 800 800.554 1900
10 -6.5 30 800 801.173 1900
```

...

```
500 -7 36 798.411 814.357 1900
501 -6.5 36 796.635 830.394 1900
502 -6 36 793.262 860.866 1900
503 -5.5 36 787.236 915.303 1900
504 -5 36 777.127 1006.62 1900
505 -4.5 36 761.226 1150.26 1900
506 -4 36 737.823 1361.66 1900
507 -3.5 36 705.685 1651.98 1900
508 -3 36 664.665 2022.53 1900
509 -2.5 36 616.299 2459.43 1900
```

This is a synthetic layer model generated from two gaussian functions. *This* is what the topography (height column) and the thickness of the sediments look like:

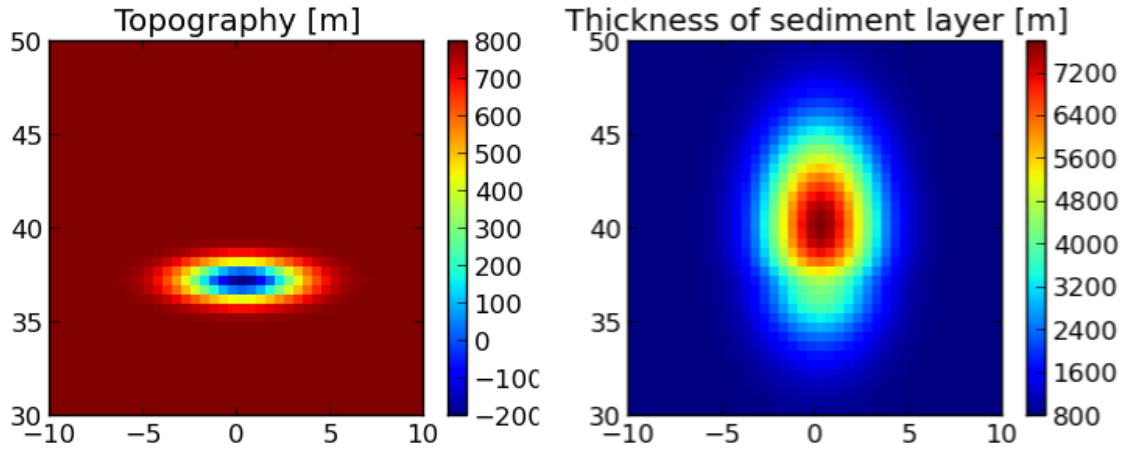


Fig. 3.11: Plot of the third and forth columns of `layers.txt`. The x and y axis are longitude and latitude, respectively.

The model file generated looks like this:

```
 1 # Tesseroid model generated by tesslayers 1.1dev:
 2 #   local time: Fri Jul 20 18:02:45 2012
 3 #   grid spacing (size of tesseroids): 0.5 deg lon / 0.5 deg lat
 4 -10.25 -9.75 29.75 30.25 800 -0.00200000032782555 1900
 5 -9.75 -9.25 29.75 30.25 800 -0.00600000005215406 1900
 6 -9.25 -8.75 29.75 30.25 800 -0.0159999998286366 1900
 7 -8.75 -8.25 29.75 30.25 800 -0.0420000003650784 1900
 8 -8.25 -7.75 29.75 30.25 800 -0.105000000447035 1900
 9 -7.75 -7.25 29.75 30.25 800 -0.247999999672174 1900
10 -7.25 -6.75 29.75 30.25 800 -0.553999999538064 1900
```

...

```
500 -7.75 -7.25 35.75 36.25 799.290000000037 -7.125 1900
501 -7.25 -6.75 35.75 36.25 798.411000000313 -15.9459999995306 1900
502 -6.75 -6.25 35.75 36.25 796.634999999776 -33.7590000005439 1900
503 -6.25 -5.75 35.75 36.25 793.262000000104 -67.6040000002831 1900
504 -5.75 -5.25 35.75 36.25 787.235999999568 -128.067000000738 1900
505 -5.25 -4.75 35.75 36.25 777.127000000328 -229.492999999784 1900
506 -4.75 -4.25 35.75 36.25 761.225999999791 -389.033999999985 1900
507 -4.25 -3.75 35.75 36.25 737.822999999858 -623.837000000291 1900
508 -3.75 -3.25 35.75 36.25 705.68499999959 -946.295000000857 1900
509 -3.25 -2.75 35.75 36.25 664.665000000037 -1357.86500000022 1900
```

The result should look like the *following* ("column" means the column of the output file).

License

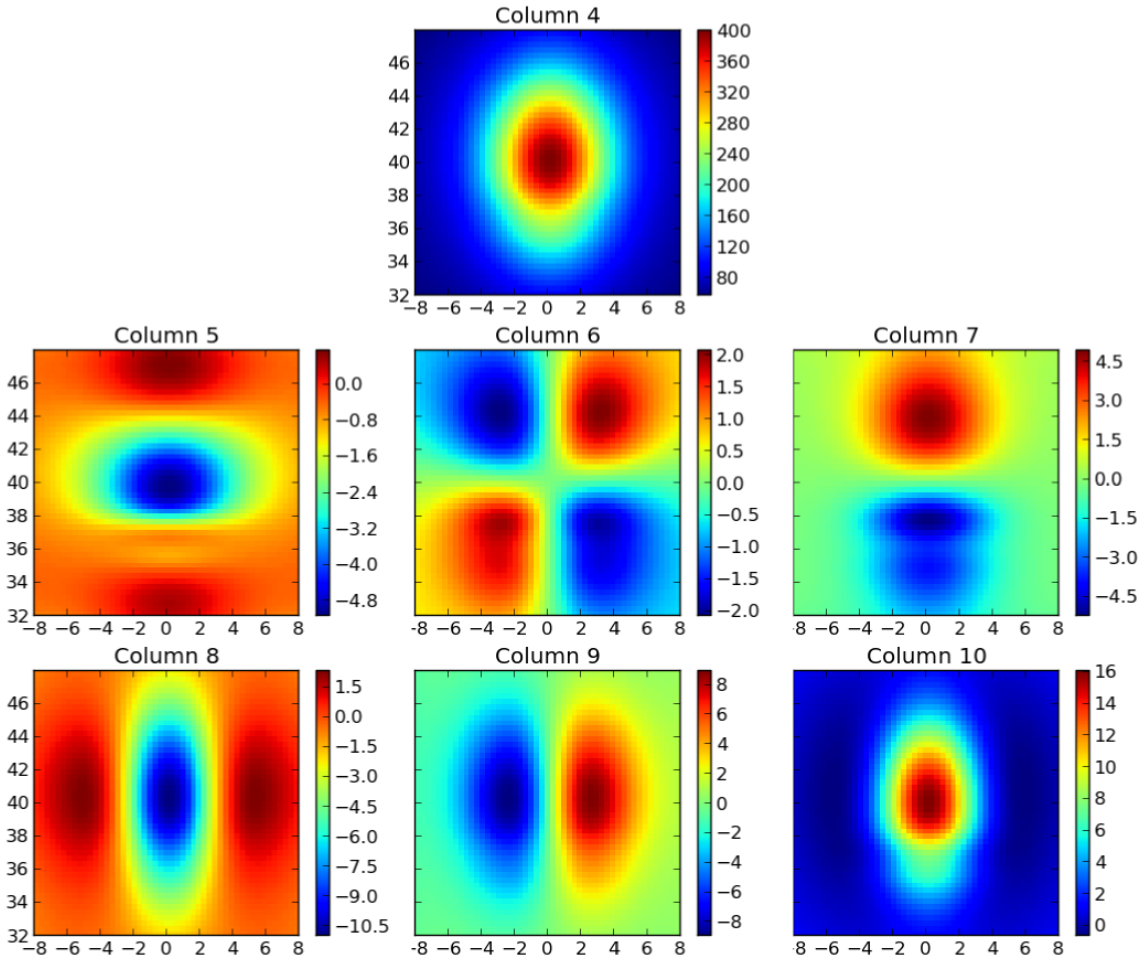


Fig. 3.12: Plot of the columns of `output.txt` generated by `tesslayers.sh`. The x and y axis are longitude and latitude, respectively.