Example: Calculate the gravity gradient tensor from a DEM file

Leonardo Uieda

April 11, 2011

This document intends to demonstrate 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;
5. Make nice color plots of the results (we'll be using Python for this too);

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

run_example.sh

```
#!/bin/bash

# First, insert the density information into the DEM file using a Python script
python dem_density.py dem-10min.xyz > dem-10min-dens.xyz

# Now, use tessmodgen to create a tesseroid model
tessmodgen -v -s0.166666667/0.166666667 -z0 < dem-10min-dens.xyz > dem-10min-mod.txt

# Calculate the GGT
tessgrd -r-60/-45/-30/-15 -b50/50 -z250e03 | \
tessgxx dem-10min-mod.txt -lgxx.log | tessgxy dem-10min-mod.txt -lgxy.log | \
tessgxz dem-10min-mod.txt -lgxz.log | tessgyy dem-10min-mod.txt -lgyy.log | \
tessgyz dem-10min-mod.txt -lgyz.log | tessgzz dem-10min-mod.txt -lgzz.log > dem-10min-
    ggt.xyz

# Use another Python script to make some nice plots
python mkplots.py
```

1 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 (`http://www.soest.hawaii.edu/gmt`) or other plotting program.
If you have interest in learning Python we recommend the excelent video lectures in the Software Carpentry (`http://software-carpentry.org`) 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.

2 The DEM file

For this example we'll use ETOPO1 for our DEM. The file dem-10min.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):

dem–10min.xyz

```
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 you can include comments in the files by starting a line with #. Figure 1 shows the DEM ploted in pseudocolor. The red rectangle is the area in which we'll be calculating the GGT.

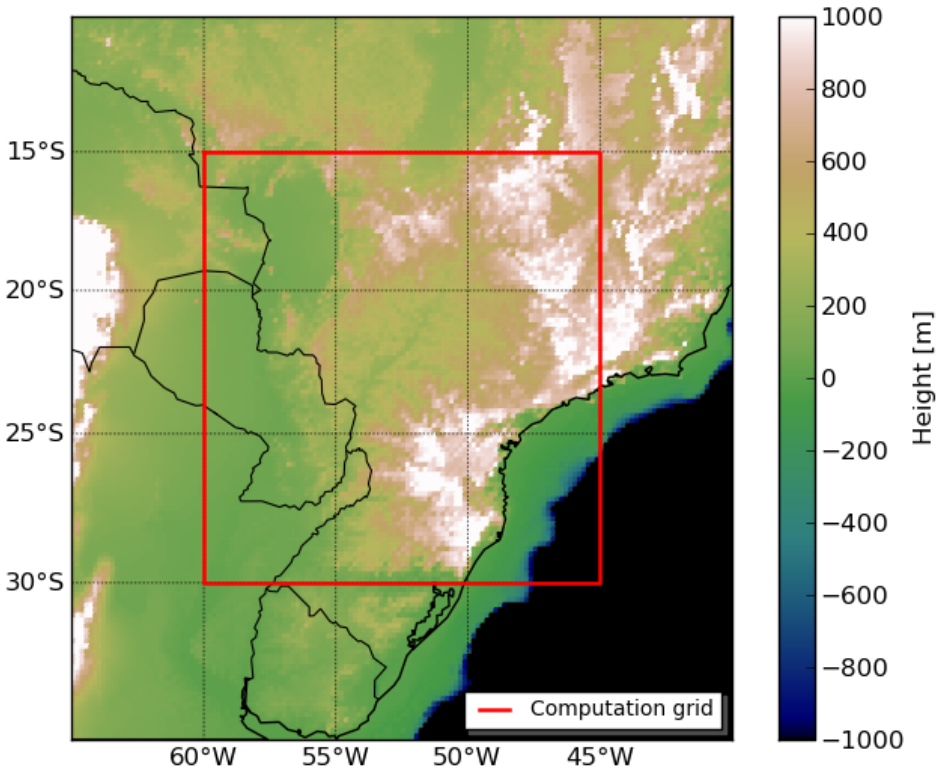

Figure 1: The ETOPO1 10" DEM of the Paraná Basin, southern Brasil.

3 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 $g.cm^{-3}$ and oceans with 1.67 $g.cm^{-3}$. 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-10min-dens.xyz:

run_example.sh

```
# First, insert the density information into the DEM file using a Python script
python dem_density.py dem-10min.xyz > dem-10min-dens.xyz
```

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:

dem_density.py

```
"""
Assign density values for the DEM points.
"""
import sys
import numpy

lons, lats, heights = numpy.loadtxt(sys.argv[1], unpack=True)

for i in xrange(len(heights)):
    if heights[i] >=0:
        print "%lf %lf %lf %lf" % (lons[i], lats[i], heights[i], 2670.0)
    else:
        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 (Figure 2):

dem–10min–dens.xyz

```
-65.000000 -10.000000 157.000000 2670.000000
-64.833333 -10.000000 168.000000 2670.000000
-64.666667 -10.000000 177.000000 2670.000000
-64.500000 -10.000000 197.000000 2670.000000
-64.333333 -10.000000 144.000000 2670.000000
-64.166667 -10.000000 178.000000 2670.000000
-64.000000 -10.000000 166.000000 2670.000000
-63.833333 -10.000000 164.000000 2670.000000
-63.666667 -10.000000 189.000000 2670.000000
-63.500000 -10.000000 210.000000 2670.000000
```

4 Making the tesseroid model

Next, we'll use our new file dem-10min-dens.xyz and program tessmodgen to create a tesseroid model of the DEM:

run_example.sh

```
# Now, use tessmodgen to create a tesseroid model
tessmodgen -v -s0.166666667/0.166666667 -z0 < dem-10min-dens.xyz > dem-10min-mod.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.

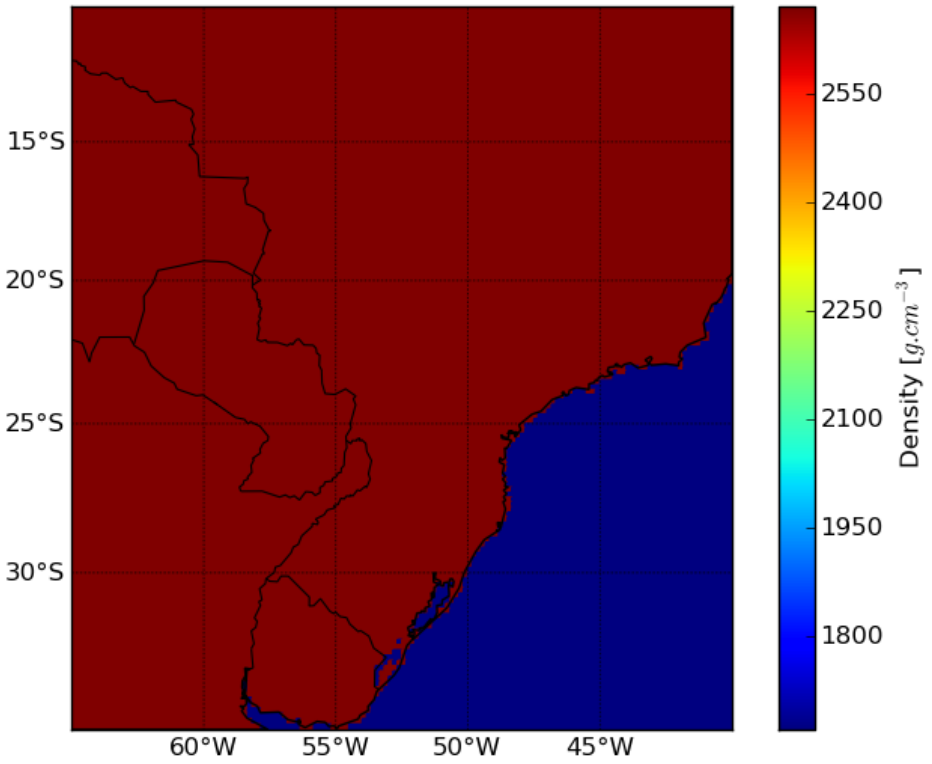

Figure 2: Density values. 2.67 $g.cm^{-3}$ in continents and 1.67 $g.cm^{-3}$ in the oceans.

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:

dem–10min–mod.txt

```
1  # Tesseroid model generated by tessmodgen 1.0:
2  #   local time: Thu Apr  7 15:30:33 2011
3  #   grid spacing: 0.166667 deg lon / 0.166667 deg lat
4  #   reference level (depth): 0
5  #   density: read from input
6  -65.0833333335 -64.9166666665 -10.0833333335 -9.9166666665 157 0 2670
7  -64.9166663335 -64.7499996665 -10.0833333335 -9.9166666665 168 0 2670
8  -64.7500003335 -64.5833336665 -10.0833333335 -9.9166666665 177 0 2670
9  -64.5833333335 -64.4166666665 -10.0833333335 -9.9166666665 197 0 2670
10 -64.4166663335 -64.2499996665 -10.0833333335 -9.9166666665 144 0 2670
```

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

dem–10min–mod.txt

```
9065 -40.0833333335 -39.9166666665 -19.9166663335 -19.7499996665 0 -19 -1670
```

5 Calculating the GGT

Tesseroids now 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 tessg* 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 to the set of tools the program tessgrd to create 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 tessg* programs:

run_example.sh

```
9  # Calculate the GGT
10 tessgrd -r-60/-45/-30/-15 -b50/50 -z250e03 | \
11 tessgxx dem-10min-mod.txt -lgxx.log | tessgxy dem-10min-mod.txt -lgxy.log | \
12 tessgxz dem-10min-mod.txt -lgxz.log | tessgyy dem-10min-mod.txt -lgyy.log | \
13 tessgyz dem-10min-mod.txt -lgyz.log | tessgzz dem-10min-mod.txt -lgzz.log > dem-10min-
       ggt.xyz
```

The end result of this is file dem-10min-ggt.xyz 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 Figure 3.

dem–10min–ggt.xyz

```
26 # gxx component calculated with tessgxx 1.0:
27 #   local time: Thu Apr  7 15:30:33 2011
28 #   model file: dem-10min-mod.txt (22801 tesseroids)
29 #   GLQ order: 2 lon / 2 lat / 2 r
30 #   Use adaptative algorithm: True
31 # Grid generated with tessgrd 1.0:
32 #   local time: Thu Apr  7 15:30:33 2011
33 #   args: -r-60/-45/-30/-15 -b50/50 -z250000
34 #   grid spacing: 0.3061224490 lon / 0.3061224490 lat
35 #   total 2500 points
36 -60 -30 250000 0.0610445591649299 0.125941812061822 -0.115028827048309
       -0.0913138725698717 0.0752780977843118 0.0302693134049447
37 -59.6938775510204 -30 250000 0.0627400803886871 0.130584463354345 -0.119451177918124
       -0.0927917856885737 0.0600651687877179 0.0300517052998942
```

6 Making the plots

The plots were generated using the powerfull Python library Matplotlib (`http://matplotlib.sourceforge.net/index.html`). The script mkplots.py is somewhat more complicated that 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 (`http://matplotlib.sourceforge.net/basemap/doc/html/`).

7 Conclusions

To wrap up our example, I would like to note that this same processes can be applied for any sort of interface, like Moho and sediment thickness maps. The difference is all in the choice of the reference level. For Moho it could be, say, h=-35km. Another thing to note is that the interface relief should be given as heights. So sediment thickness would have to be converted into height of the basin's basement (which would be negative values).

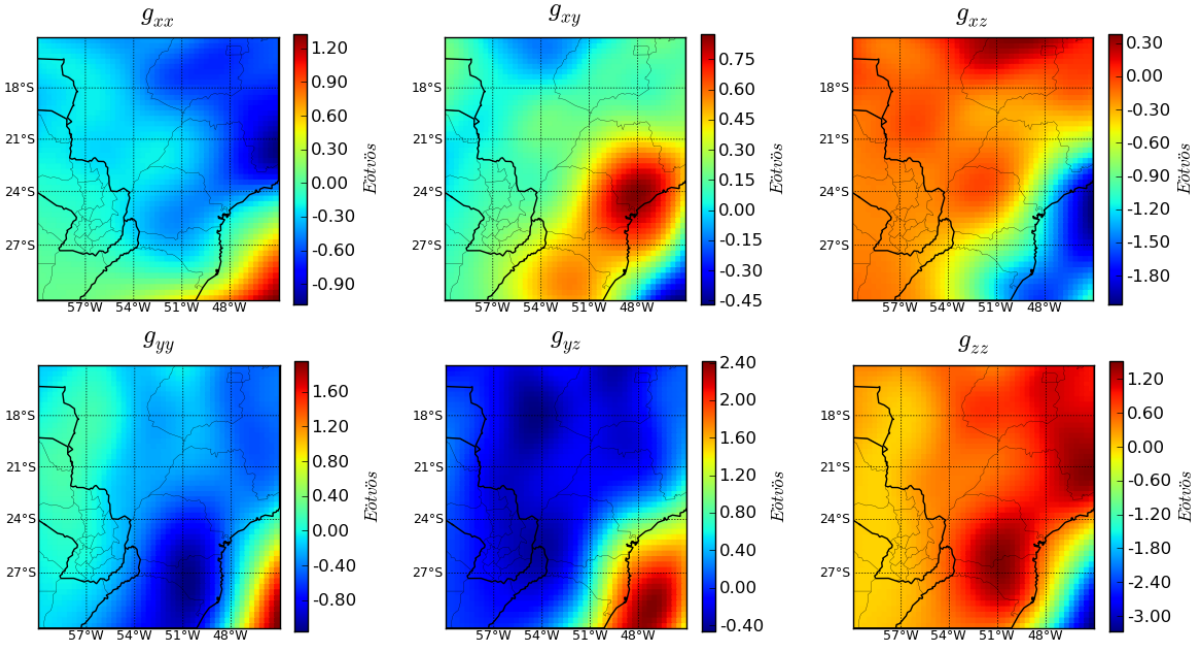

Figure 3: GGT of caused by the topographic masses.