moho-inversion-example

March 23, 2016

1 Implementation of the Moho inversion algorithm

This notebook presents a Python class that implements the proposed method. We'll use the inverse problems framework of the library Fatiando a Terra. The class `MohoGravityInvSpherical` is defined in the `mohoinv.py` file.

1.1 Package imports

```
In [1]: # Insert plots into the notebook
        %matplotlib inline
```

```
In [2]: from __future__ import division, unicode_literals
        import numpy as np
        import multiprocessing
        from IPython.display import Image
        import matplotlib.pyplot as plt
        import seaborn  # Makes the default style of the plots nicer
```

```
/home/leo/bin/anaconda/envs/moho/lib/python2.7/site-packages/matplotlib/__init__.py:872: UserWarning: axe
  warnings.warn(self.msg_depr % (key, alt_key))
```

Load the required modules from Fatiando a Terra and show the specific version of the library used.

```
In [3]: from fatiando.vis import myv, mpl
        from fatiando.gravmag import tesseroid
        from fatiando import utils, gridder
        import fatiando
```

```
In [4]: print("Using Fatiando a Terra version: {}".format(fatiando.__version__))
```

```
Using Fatiando a Terra version: 3c4953c170e1e9d964325ccd133a5ef28e319e89
```

```
In [5]: from mohoinv import TesseroidRelief, MohoGravityInvSpherical, make_mesh
```

Get the number of cores in the computer to run the forward modeling in parallel.

```
In [6]: ncpu = multiprocessing.cpu_count()
        ncpu
```

```
Out[6]: 4
```

1.2 Test the class on simple synthetic data

We can test and show how the class works on some simple synthetic data. We'll use the example model from the `tesseroid-relief-example.ipynb` notebook.

First, make the model of the Moho.

In [7]:
```
# shape is nlat, nlon = the number of points in the grid
shape = (30, 30)
# Make a regular grid inside an area = (s, n, w, e)
area = (20, 60, -40, 40)
lat, lon, h = gridder.regular(area, shape, z=250e3)
# Make a checkerboard relief undulating along the -35km height reference
f = 0.15
reference = -35e3
relief = 10e3*np.sin(1.5*f*lon)*np.cos(f*lat) + reference
# The density contrast is negative if the relief is below the reference
density = 600*np.ones_like(relief)
density[relief < reference] *= -1
```

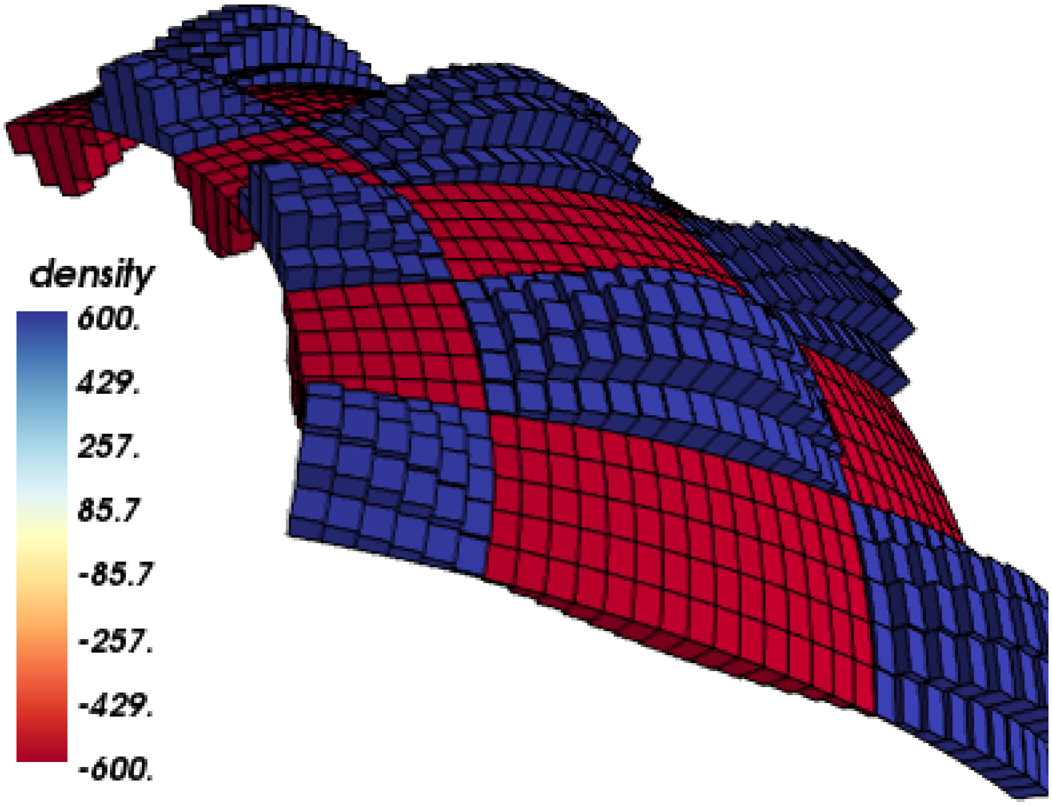
In [8]:
```
model = make_mesh(area, shape, relief, reference)
model.addprop('density', density)
```

In [9]:
```
def plot_result_3d(moho, fname):
    "Plot the tesseroid model in 3D."
    scale = (1, 1, 40) # Exaggerate the radial dimension
    datarange = np.abs([moho.props['density'].max(),
                        moho.props['density'].min()]).max()
    scene = myv.figure(zdown=False)
    plot = myv.tesseroids(moho, 'density', scale=scale)
    plot.module_manager.scalar_lut_manager.show_legend = True
    plot.module_manager.scalar_lut_manager.lut_mode = 'RdYlBu'
    plot.module_manager.scalar_lut_manager.data_range = [-datarange, datarange]
    plot.module_manager.scalar_lut_manager.scalar_bar_representation.minimum_size = \
        np.array([1, 1])
    plot.module_manager.scalar_lut_manager.scalar_bar_representation.position2 = \
        np.array([ 0.13741855,  0.64385382])
    plot.module_manager.scalar_lut_manager.scalar_bar_representation.position = \
        np.array([ 0.03303258,  0.07342193])
    plot.module_manager.scalar_lut_manager.scalar_bar_representation.maximum_size = \
        np.array([100000, 100000])
    plot.module_manager.scalar_lut_manager.label_text_property.color = (0, 0, 0)
    plot.module_manager.scalar_lut_manager.title_text_property.color = (0, 0, 0)
    scene.scene.camera.position = [2252864.9143914036, -5202911.2574882135,
                                   8495162.9722945951]
    scene.scene.camera.focal_point = [3135763.9476126051, 1056258.4985192744,
                                      829277.18542720564]
    scene.scene.camera.view_angle = 30.0
    scene.scene.camera.view_up = [0.6164057832087273, 0.57367112225287575,
                                  0.53939350563383837]
    scene.scene.camera.clipping_range = [783483.44437851617, 16078402.004277557]
    scene.scene.camera.compute_view_plane_normal()
    scene.scene.render()
    myv.savefig(fname)
    #myv.show()
    myv.mlab.close()
    return Image(filename=fname)
```

In [10]:
```
plot_result_3d(model, 'simple-synthetic-model.png')
```

```
WARNING:traits.has_traits:DEPRECATED: traits.has_traits.wrapped_class, 'the 'implements' class advisor h
```

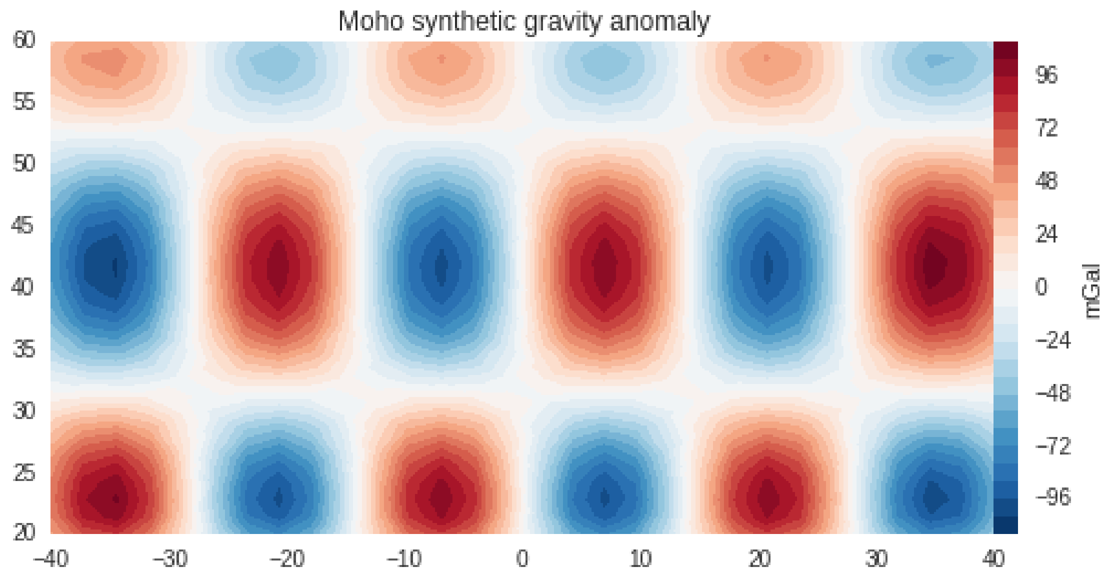
Out[10]:



Now, generate some synthetic data by forward modeling.

In [11]:
```
gz = tesseroid.gz(lon, lat, h, model, njobs=ncpu)
```

In [12]:
```
plt.figure(figsize=(9, 4))
plt.title('Moho synthetic gravity anomaly')
plt.tricontourf(lon, lat, gz, 30, cmap='RdBu_r')
plt.colorbar(pad=0).set_label('mGal')
```



1.3 Run the inversion

For this test, we'll use a mesh with the same dimensions and the original model.

```
In [13]: mesh = model.copy(deep=True)
         mesh.props['density'] = 600*np.ones(mesh.size)
```

Create the solver object.

```
In [14]: solver = MohoGravityInvSpherical(lat, lon, h, gz, mesh, njobs=ncpu)
```

Configure the optimization method to Gauss-Newton and set the initial estimate.

```
In [15]: initial = np.ones(solver.nparams)*(mesh.reference - 30e3)
         solver.config('newton', initial=initial, tol=0.2, maxit=10)

Out[15]: <mohoinv.MohoGravityInvSpherical at 0x7f7346642490>
```

Run the inversion and time the computation.

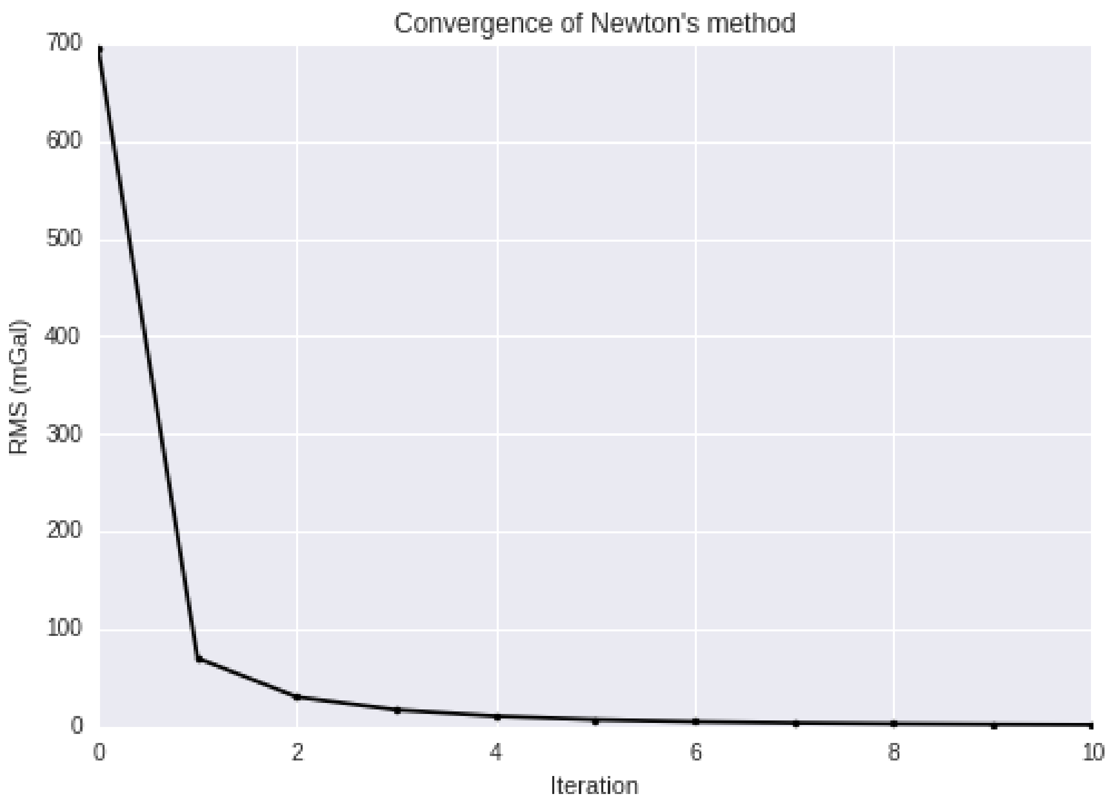
```
In [16]: %time solver.fit()

CPU times: user 56 ms, sys: 28 ms, total: 84 ms
Wall time: 9.25 s

/home/leo/bin/anaconda/envs/moho/lib/python2.7/site-packages/fatiando/inversion/optimization.py:185: Ru
  RuntimeWarning)
```

Plot the RMS error (Root Mean Square) per iteration to get an idea of the convergence of the method.

```
In [17]: rms = np.sqrt(solver.stats_['objective'])/np.sqrt(solver.ndata)
```

```
In [18]: plt.figure()
         ax = plt.subplot(111)
         ax.set_title('Convergence of {}'.format(solver.stats_['method']))
         ax.plot(rms, '.k-')
         ax.set_ylabel('RMS (mGal)')
         ax.set_xlabel('Iteration')
```

```
Out[18]: <matplotlib.text.Text at 0x7f7347757e50>
```

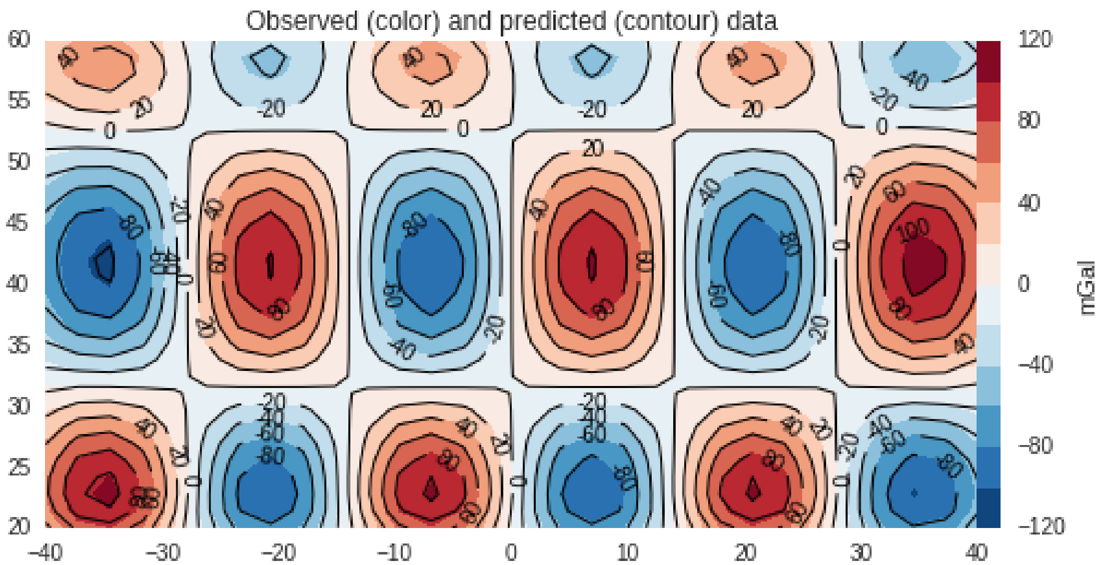


Plot the data misfit and residuals

```
In [19]: predicted = solver.predicted()
         residuals = solver.residuals()
```

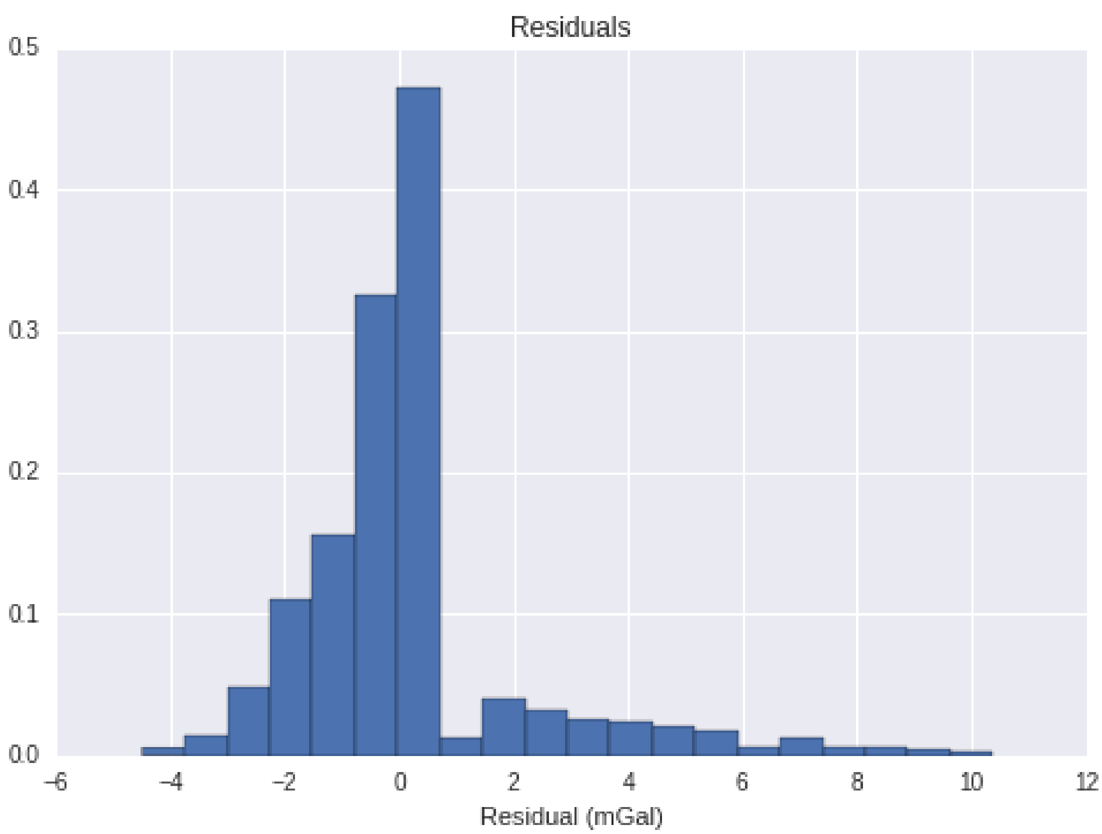
```
In [20]: plt.figure(figsize=(9, 4))
         levels = mpl.contourf(lon, lat, gz, shape, 12, cmap='RdBu_r')
         plt.colorbar(pad=0).set_label('mGal')
         mpl.contour(lon, lat, predicted, shape, levels)
         plt.title('Observed (color) and predicted (contour) data')
```

```
Out[20]: <matplotlib.text.Text at 0x7f7347648fd0>
```




```
In [21]: plt.title('Residuals')
         plt.hist(residuals, bins=20, normed=True)
         plt.xlabel('Residual (mGal)')
         print('Mean: {}   std: {}'.format(residuals.mean(), residuals.std()))
```

```
Mean: 0.214036548333   std: 2.02563704196
```

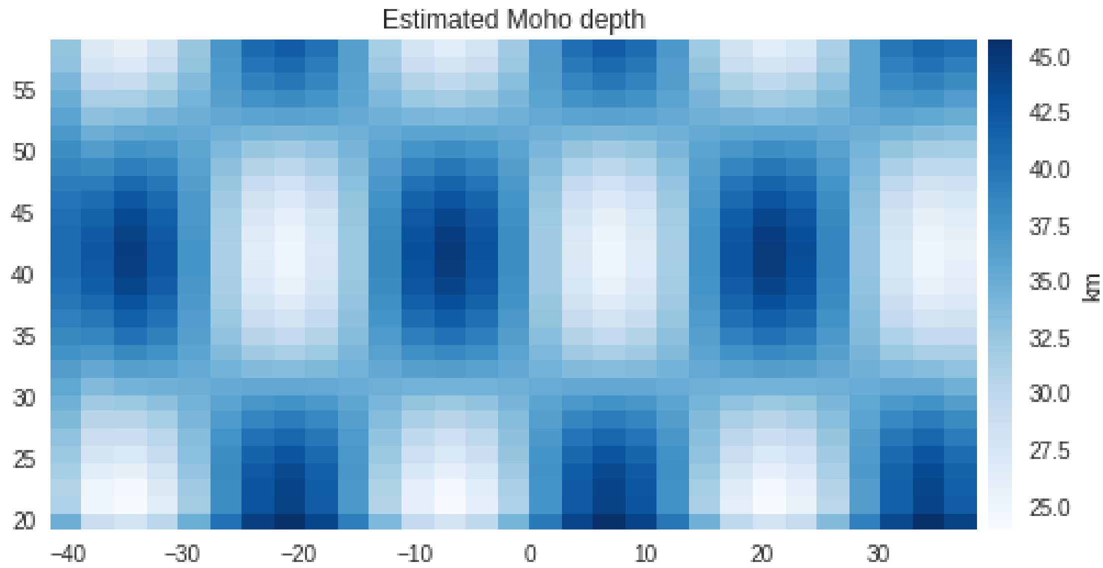


Map the estimated Moho depth.

In [22]:
```
moho = solver.estimate_
```

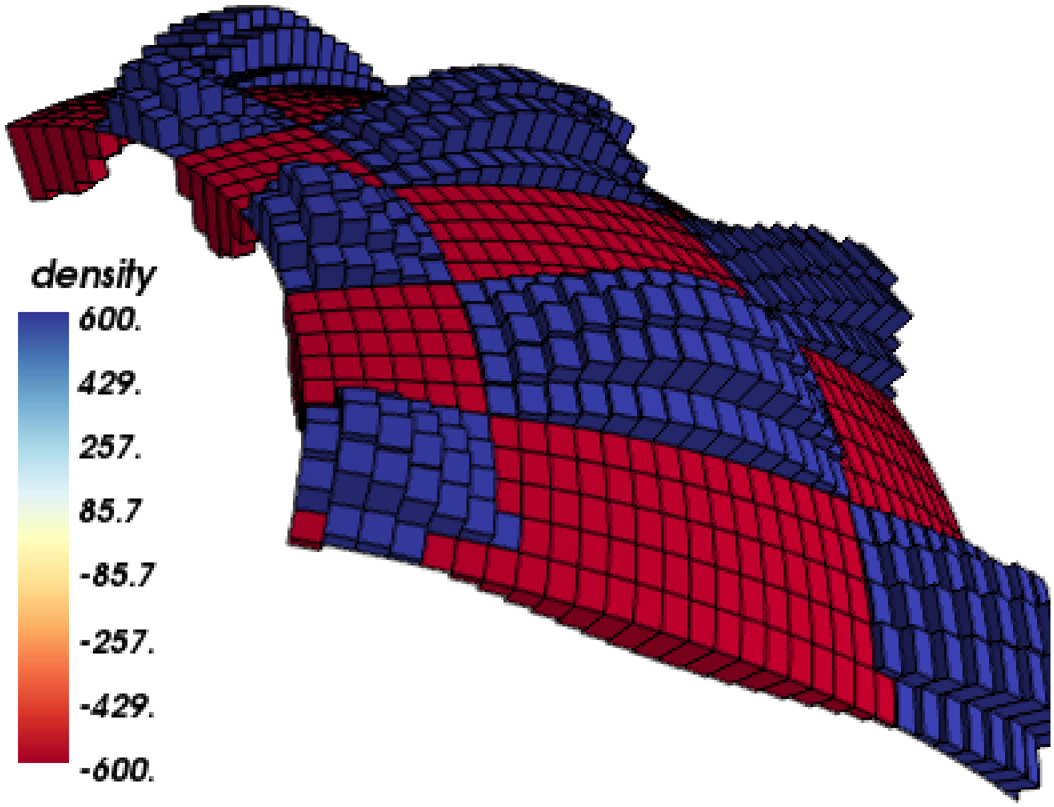
In [23]:
```
plt.figure(figsize=(9, 4))
plt.title("Estimated Moho depth")
plt.pcolormesh(moho.lons, moho.lats, -0.001*moho.relief.reshape(moho.shape),
               cmap='Blues')
plt.colorbar(pad=0.01).set_label('km')
plt.xlim(moho.lons.min(), moho.lons.max())
plt.ylim(moho.lats.min(), moho.lats.max())
```

Out[23]: (19.310344827586206, 59.310344827586206)



In [24]:
```
plot_result_3d(moho, 'mohoinv-example.png')
```

Out[24]:


density
600.
429.
257.
85.7
-85.7
-257.
-429.
-600.