2017/07/14

- My new monitor has arrived! So excited to see the big picture of things (pun intended).

- Began reading "**Developing iPhone Development with Molly K Maskrey**".
- now on pg 29

2017/07/12

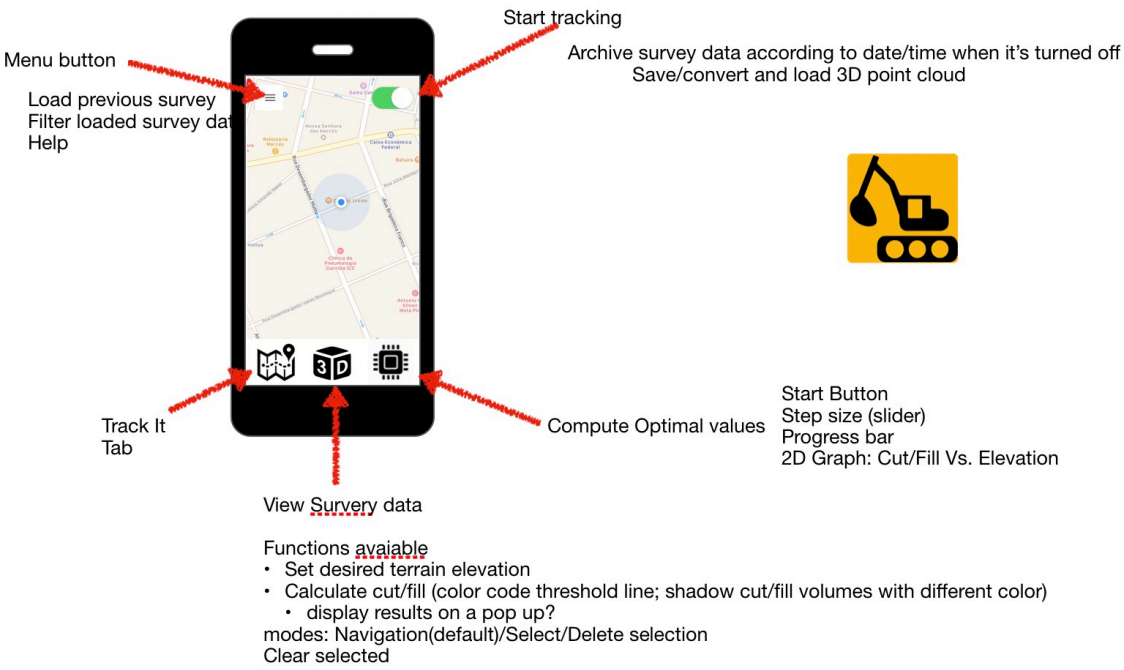
- Successfully managed to add my AppleID as an authorized one for deployment to my iPhone. App ran successfully on the iPhone for the TrackIt tab.
- 15:00: Eyeballing thru the book. Nothing that I hadn't already imagined. Now on page 216 reading about Table and Collection Views.
- 16:00: Finished going thru the book. Commencing Earthworks implementation for macOS (easier to deploy / no upload to iPhone required / great computer power).

Earthworks - macOS

- Is it a document based App?
    - Yes. We want to store .gpx files in the cloud and enable the application to load it for use.

Earthworks - iOS

- Putting some thought about why I came up with the iOS version of the app instead of the macOS one:
    - Higher audience: there's more people out there carrying iPhones than there are ones carrying macOS machines.
    - The appeal of an iPhone app kills the need for even having a computer.
- App skeleton:



2017/07/11

- After doing some thinking, seeing actual real case scenario for the program and looking at the workflow process for data collection, cleanup and display, I've decided to create a **macOS app** out of the prototype model of the Earthworks program. Reasons for choosing the platform:
  - Robust IDE (Xcode)
    - Includes resources monitor while code is running.
  - Free of charge for development
  - Potential of nice looks and performance (with SceneKit and Metal)
  - Easy monetization thru Apple's App Store.
  - Potential to extend to the iOS platform.
- At this point, it feels as if I have all of the mathematics for the program to work figured out, but still need to get comfortable developing for the Apple ecosystem. Now reading a book called **"Beginning Xcode"** in order to met these learning requirements (currently at page 62).
- 15:00: Completed Chapter 3. Now on page 83. What have I accomplished so far: built a multi-view application and managed to send data from one view to the other. I've been

introduced to the concept of **segue,** which is basically a linkage between views; prepareForSegue is called just before a segue linking view is executed. This code is placed in the source view to send data to the target view that's about to replace it; **Pushing a view** means that the navigation controller has replaced one view in the app for another.
- **New idea**: use iPhone's built in GPS unit to survey new terrain data!!! Ditch the macOS app for now and you're up for great profits given the GPS is accurate enough.
- 15:30: pg.89 trying to find those darn icons.
- 17:30: Using the CoreLocation framework in order to retrieve GPS information.
- Stopped on page 98 of the document
- A **framework** is a collection of classes and functions that provides additional functionalities to your project.
- 18:10: now on page 104. Just edited the plist file to allow location services to be enabled by the user with a prompt message.

2017/07/07

- Question posted on facebook:Preciso de uma dica de experiência: consigo usar o XCode pra desenvolver uma interface (Cocoa App) pra um gráfico tipo scatter plot 3D que me permita manipular o gráfico (rotacionar, zoom, pan), selecionar pontos indesejados e eliminá-los via interface (DEL key)? Alternativas em Python são ainda mais bem vindas. Encontrei formas de plotar, mas não de manipular pontos para deleção (Plotly). Meus dados fonte são coordenadas cartesianas (x,y,z). :) K, thks, bye!
    - Answer from Vitor Gimenes: Não mexi tão a fundo com composições em interfaces que não fossem os elementos padrão de interface ou rotinas de desenho 2D. E faz tempo que não mexo com isso. Mas, se eu tivesse esse problema pra resolver hoje e tivesse que ser de fato uma aplicação desenvolvida no Xcode no sentido literal, eu provavelmente usaria OpenGL ES, ou talvez Metal - isso, correndo o risco (que considero meu default) de ser overkill. Agora, se tiver a opção de desenvolver em qualquer outro lugar e só compilar usando Xcode, provavelmente eu tentaria alguma lib em WebGL e usaria HTML5, carregando tudo via Web View na interface criada no Xcode (estilo CocoonJS). Como eu disse, nunca fui atrás de uma solução pra isso - porque nunca precisei fazer uma solução pra isso - mas é onde eu começaria pesquisando.

2017/07/05

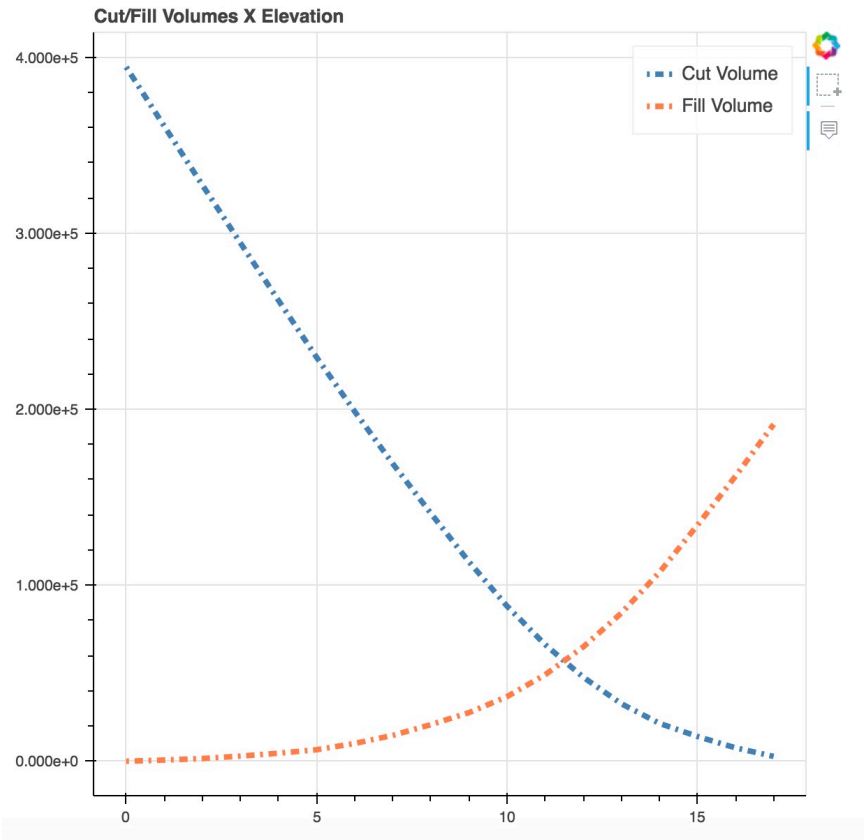
- Test run with data collected from our company's back terrain.
- Discovery: GPS can trigger measurements by pre-established time windows while in track mode. This will allow higher density point clouds to be captured.
- Must have features while working with a real world example:

    1. Load data from .gpx (separate option)
    2. Plot data and manually eliminate outliers (GPS sometimes collects points in absurd elevations possibly due to temporary signal loss).
    3. 3D Scatter plots of terrain should have the same scale in all axis (Python's Plotly is distorting scale).

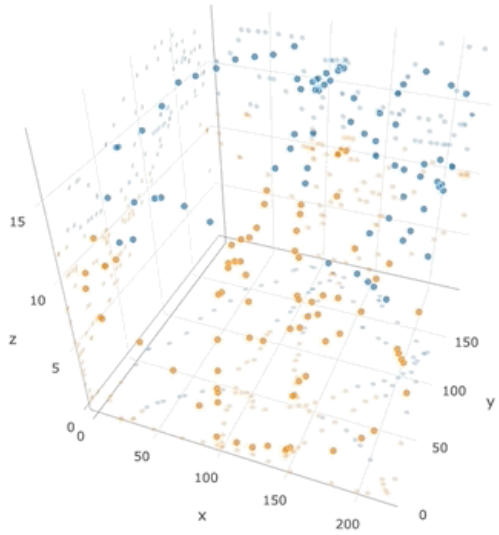
2017/06/27

- Program implementation successful.
- Next step: validate against real world scenario, in which actual volumes are known.
- Document computation methodology for publishing.

Real world data >> Volume curves.



Data plots for elevation = 11.5m



2017/06/23

- Program implementation seems to have been successful. Validating volume calculations for different elevations.

- Failed: Computation values for elevation different than 0 doesn't work properly.

2017/06/21

- Calculation successful after using Delauney's spicy implementation to order volume computation.

```
-----------------------------------------
Calculating amount of cut volume required.
Computing volume from triangular meshes.
Progress: |                                        | 100.0% Complete
The total cut volume is: 168.19 cubic meters.
Accumutaled integration error: 4.531926592179738e-11.
Computation process completed.
Script runtime: 0:01:08.417785
```

- Next: Compare different point cloud densities and effect on correct volume (170).

- Iteration 1: 300 points; 0.82 points/cubic meter - 8.2% error

```
The total cut volume is: 156.53 cubic meters.
Accumutaled integration error: 1.2988229728010765e-11.
Computation process completed.
Script runtime: 0:00:06.329923
```

Iteration 2: 14% error

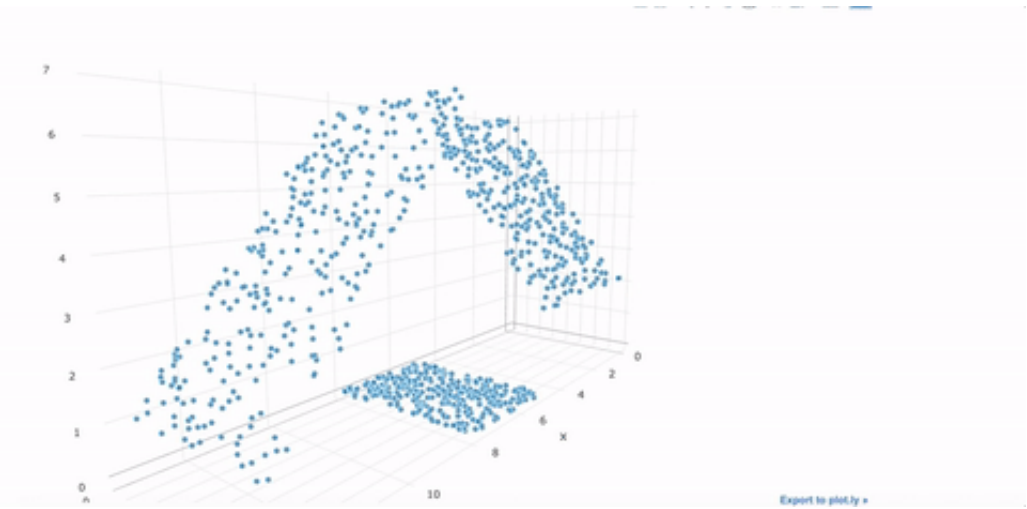
```
150 points generated thru equations.
Point cloud density: 0.41208791208791207 points per cubic meter
The total cut volume is: 146.61 cubic meters.
Accumutaled integration error: 8.870032694361646e-12.
Computation process completed.
Script runtime: 0:00:02.811927
```

Iteration 3: 1,49% error

```
1500 points generated thru equations.
Point cloud density: 4.1208791208791204 points per cubic meter
The total cut volume is: 167.46 cubic meters.
Accumutaled integration error: 3.173794571981703e-11.
Computation process completed.
Script runtime: 0:00:53.971449
```

2017/06/13

- Building test scenario:



Calculated volume: 170 units

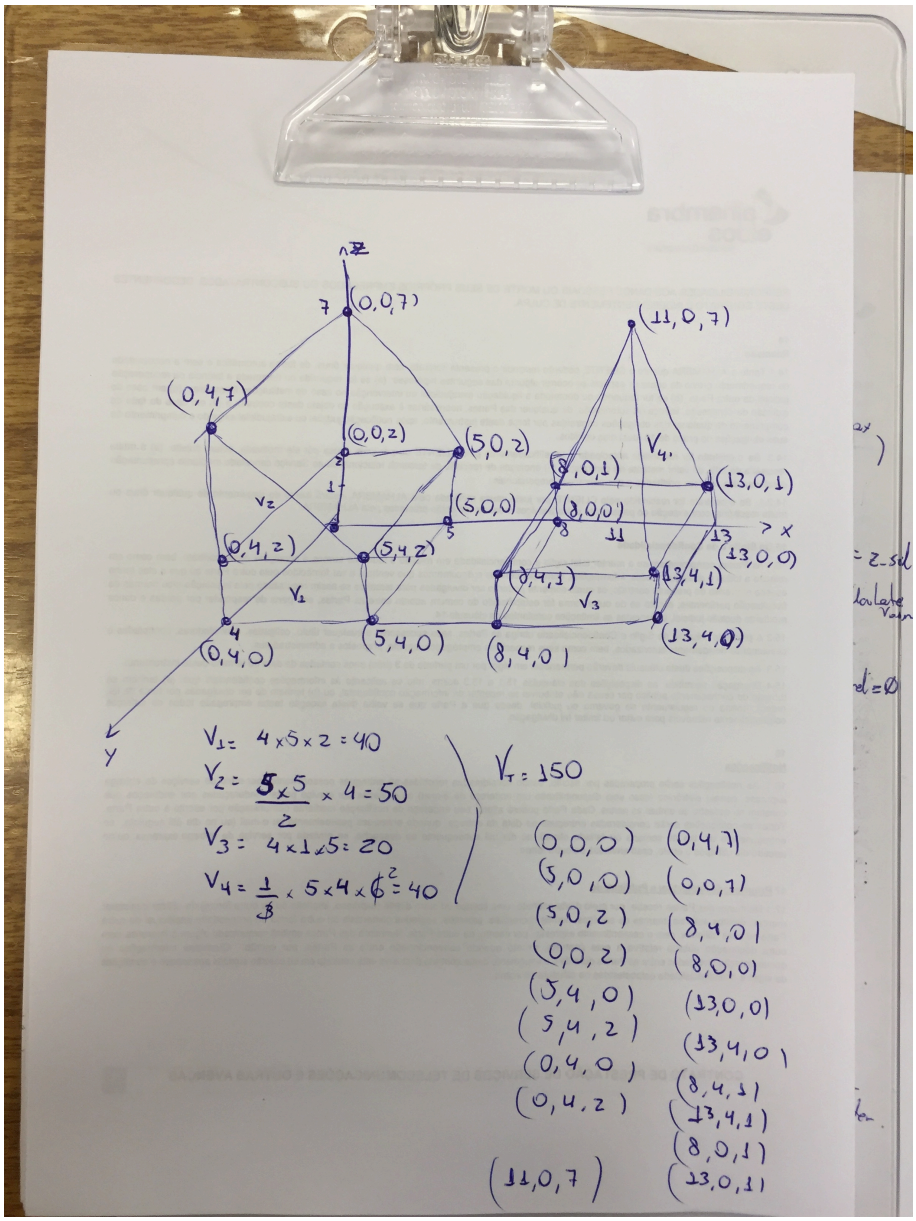
- V1=40
- V2=50
- V3=20

- V4=60

Results:
Computed volume is incorrect: 13.000. Probable cause: the ordering of points during computation. Rethink how you'll order them.

2017/06/12

- Having a program created and hoping it carries calculations accordingly isn't enough, so I'll devise a controlled scenario where I can control point cloud density and measure:
    - Accuracy: how well does the program do Vs. analytical results of known geometrical shapes
    - Performance: how much faster does the program get as point cloud density is lowered?
- Test scenario: The one devised on 2017/05/25.
    - Known total volume: 150



- Challenge 1: Create a test set and visualize it
  - Use the piece of code I created when attempting to compute volumes using a machine learning algorithm by using all plane equations defined by the point cloud (Combination of 155 x3)

2017/06/07

- While surveying terrain with the GPS, I noticed 2 issues that need addressing:

  1. Even setting up the GPS to compute points with higher

frequency, it just doesn't seem enough. Maybe the solution is to keep walking around the terrain several times until enough points have been collected.

2. The GPS occasionally computes altitudes incorrectly. These are easily spotted by eyeballing the survey data in a 3D Scatter plot. Find a way to eliminate these outliers from the calculations.

2017/06/01

- Calculating if point is inside a rectangular shape:

Let $P(x, y)$, and rectangle $A(x_1, y_1), B(x_2, y_2), C(x_3, y_3), D(x_4, y_4)$

Calculate the sum of areas of $\triangle APD$, $\triangle DPC$, $\triangle CPB$, $\triangle PBA$.

1. If this sum is greater than the area of the rectangle, then $P(x, y)$ is outside the rectangle.
2. Else if this sum is equal to the area of the rectangle (observe that this sum cannot be less than the latter),
    1. if area of any of the triangles is 0, then $P(x, y)$ is on the rectangle (in fact on that line corresponding to the triangle of area= 0). Observe that the equality of the sum is necessary; it is not sufficient that area= 0),
    2. else $P(x, y)$ is is inside the rectangle.

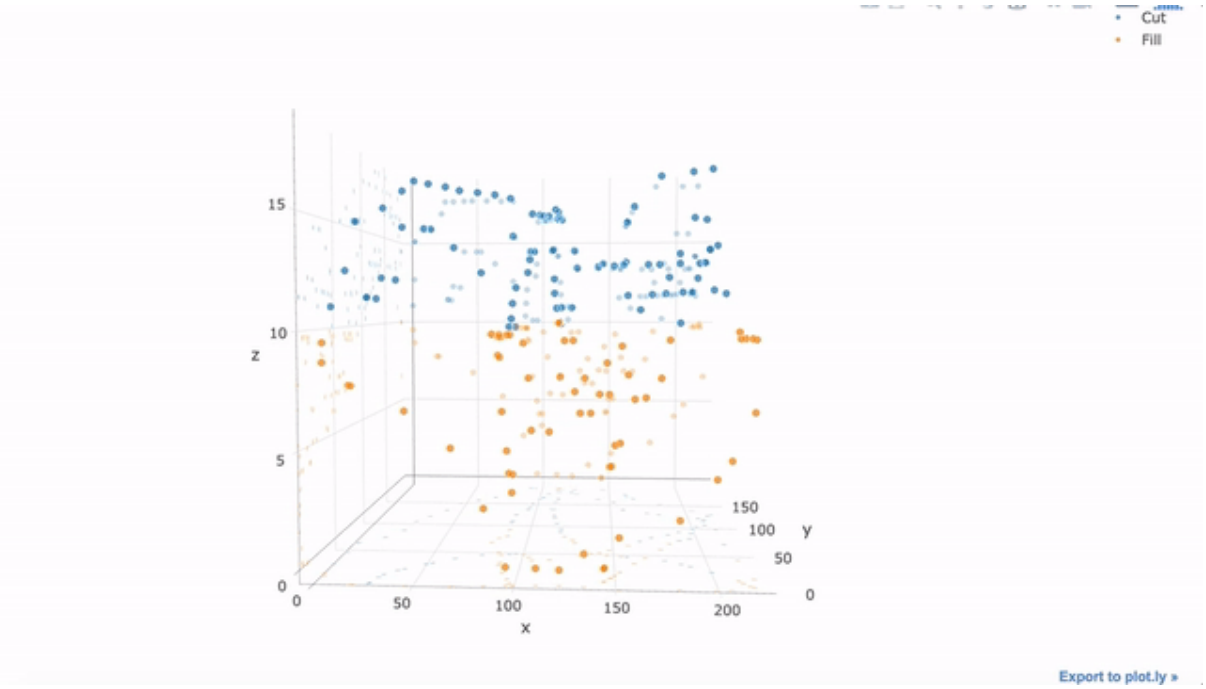
Acceptably this approach needs substantial amount of computation. This approach can be employed to any irregular polygon, too.

2017/06/08

- The program is now able to filter on 4 sided plateaus in order to calculate only a specific part of the terrain that has been surveyed. The main.py doesn't look as good as I think it could, but it can get the job done.
- Next essential feature: remove outliers
    - Statistical analysis? Doesn't require advanced interface.
    - Human based?
- Improvement idea: plot is square shaped and x,y,z axis are therefore all in their own scale.

2017/05/31

- Plotting data:
- Edson says, terrain x,y sweep is useless. Use 4 coordinates and calculate for various altitudes.

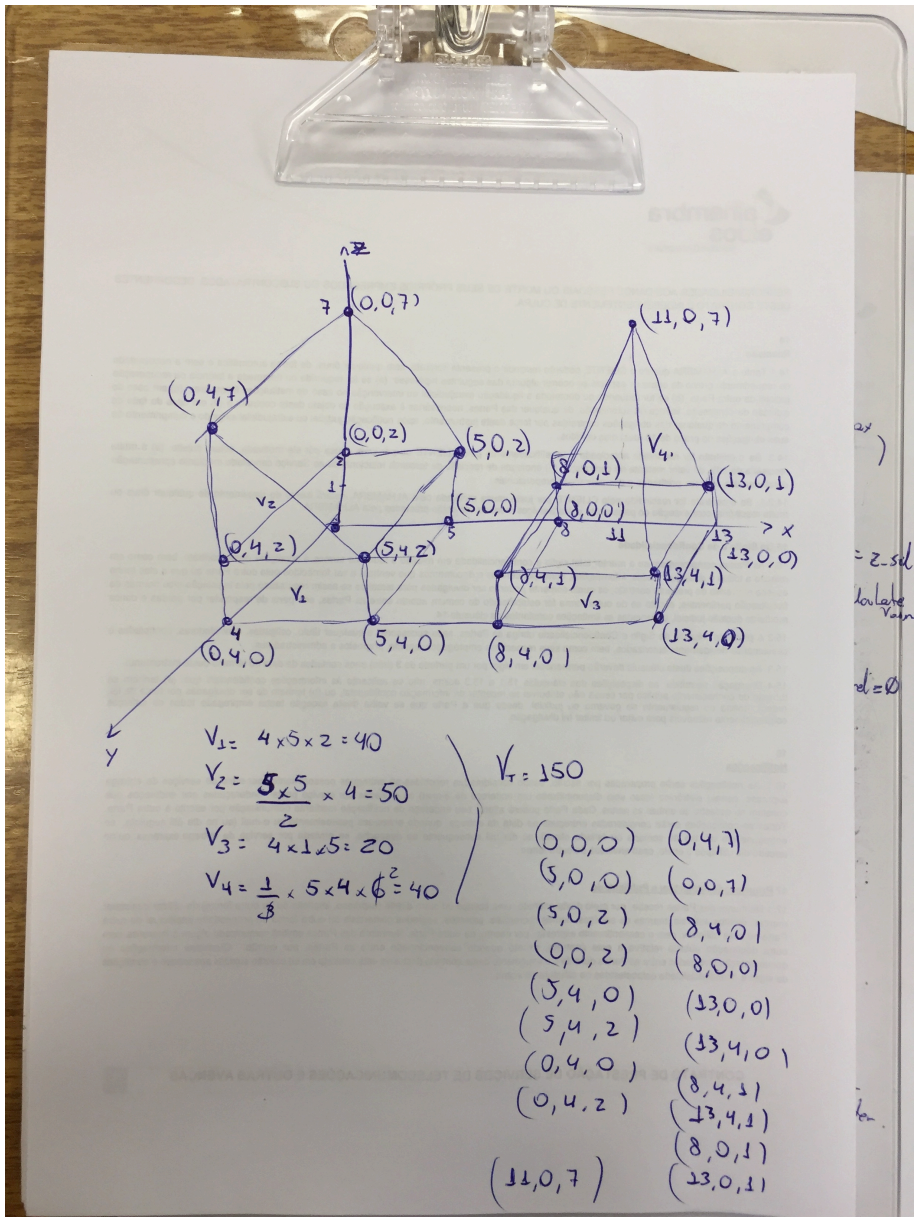


2017/05/26

- iterations for sweeping thru an area in one dimension:
    - iterations = np.floor((max_dim-window_size)/step_size + 1)

2017/05/25

- To implement:
    - make main just a skeleton of function calls and menu control (DONE)
-
- Before proceeding, I'll test the known point see to make sure it behaves as expected.
    - I was blind, but now I see: if there are 2 pointy peaks in the terrain, the algorithm on one peak will eventually create a shape that doesn't exist. Again, this may be associated with low point cloud density. If density is sufficiently high, then there may be no risk (to be validated).
-
- Test set for validating volume calculation on several elevations

- 

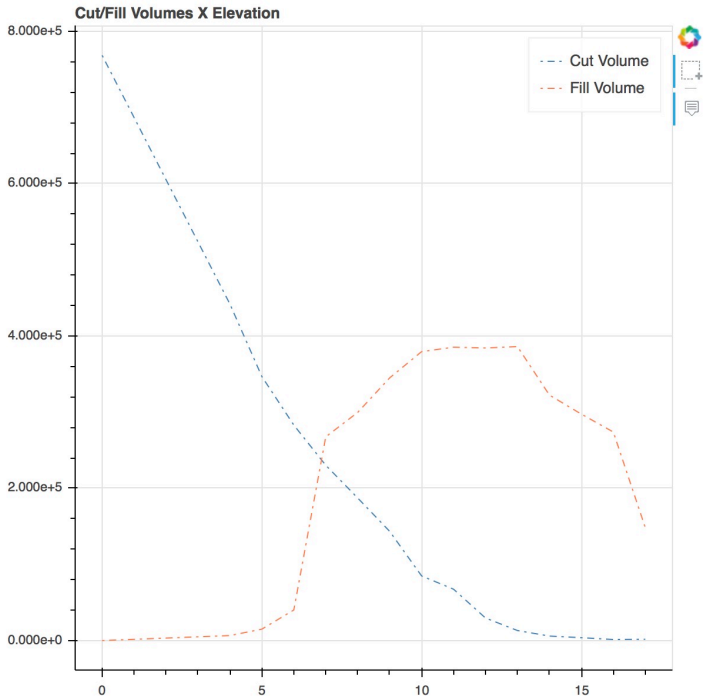
- Sample area:
  - -24.936552, -51.391311
  - -24.936947, -51.391291
  - -24.936967, -51.390895
  - -24.936655, -51.390899

- Time to add a functionality to limit the terrain size to analyze
  - 4 points will delimit construction area (4 x latitude + longitude)
    - Check if points are within the analyzed area's range.
    - Delete all outside points from the analysis.

2017/05/24
03:20pm

- Check data frame reuse in pandas. It may be causing incorrect calculations on reuse due to referenced copy.
    - NOPE: single entry calculation shows the same value.
- Next: plot elevation Vs. Cut/Fill for various iterations
- Plot the cut and fill dataframes that look weird
- Hypothesis: low point cloud density

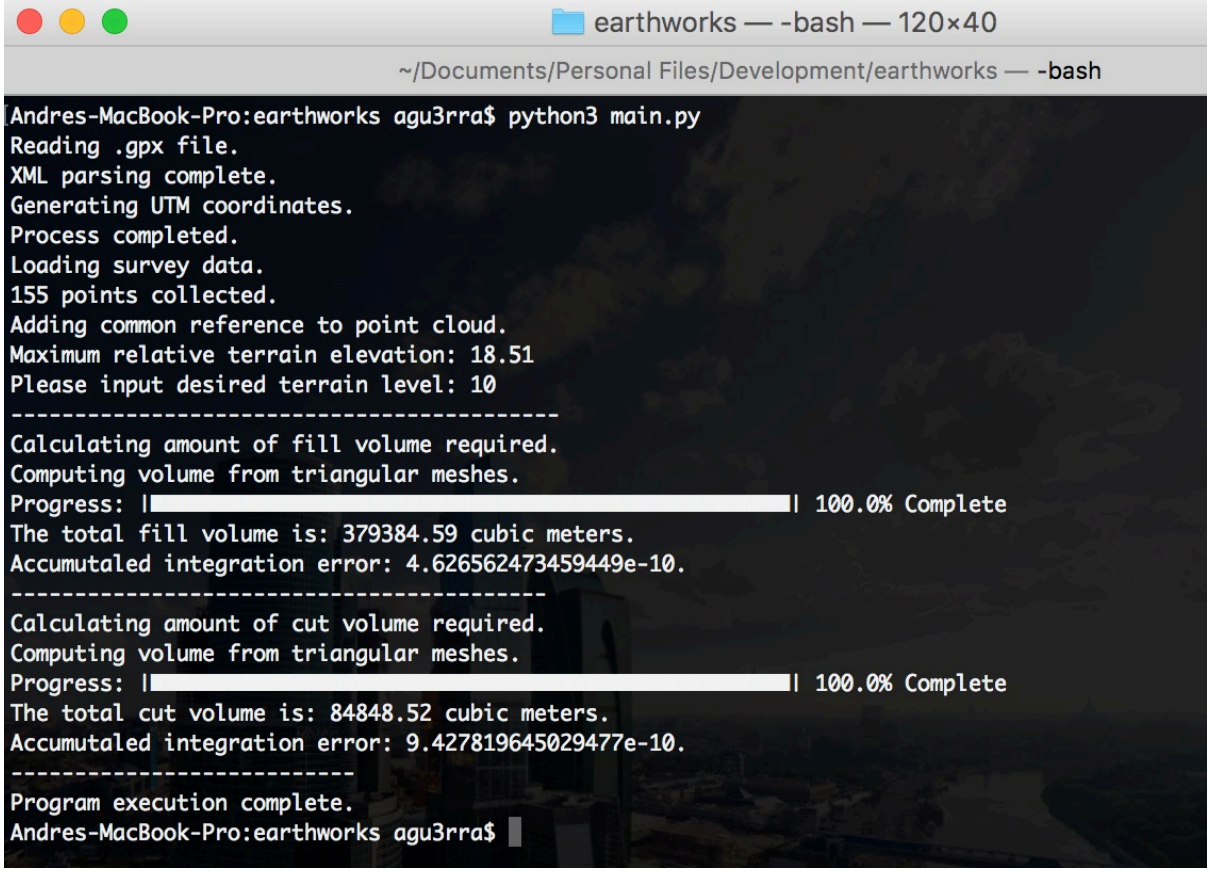
Weird looking plot:



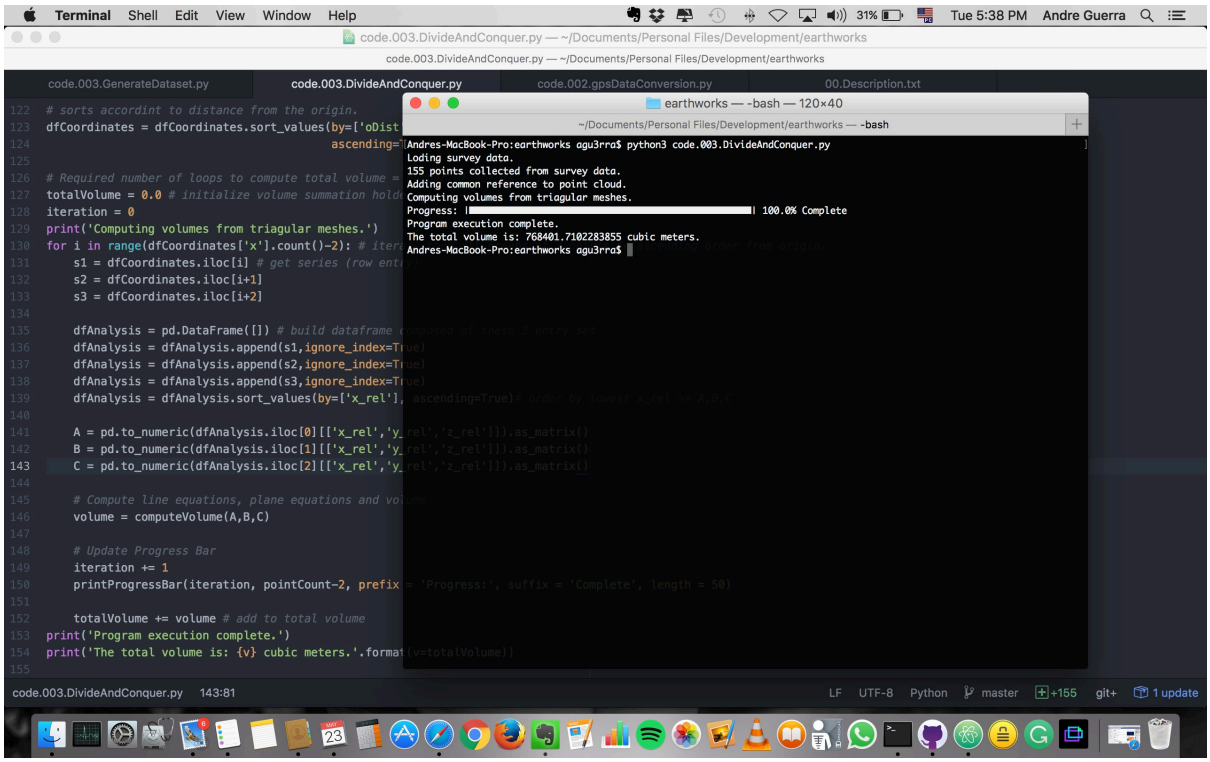
Generic File

01:00pm

- Computed cut and fill volumes required for leveling the whole terrain to a given relative elevation.
- Next step. Calculate level for cut-fill >> 0 (zero) - optimal scenario



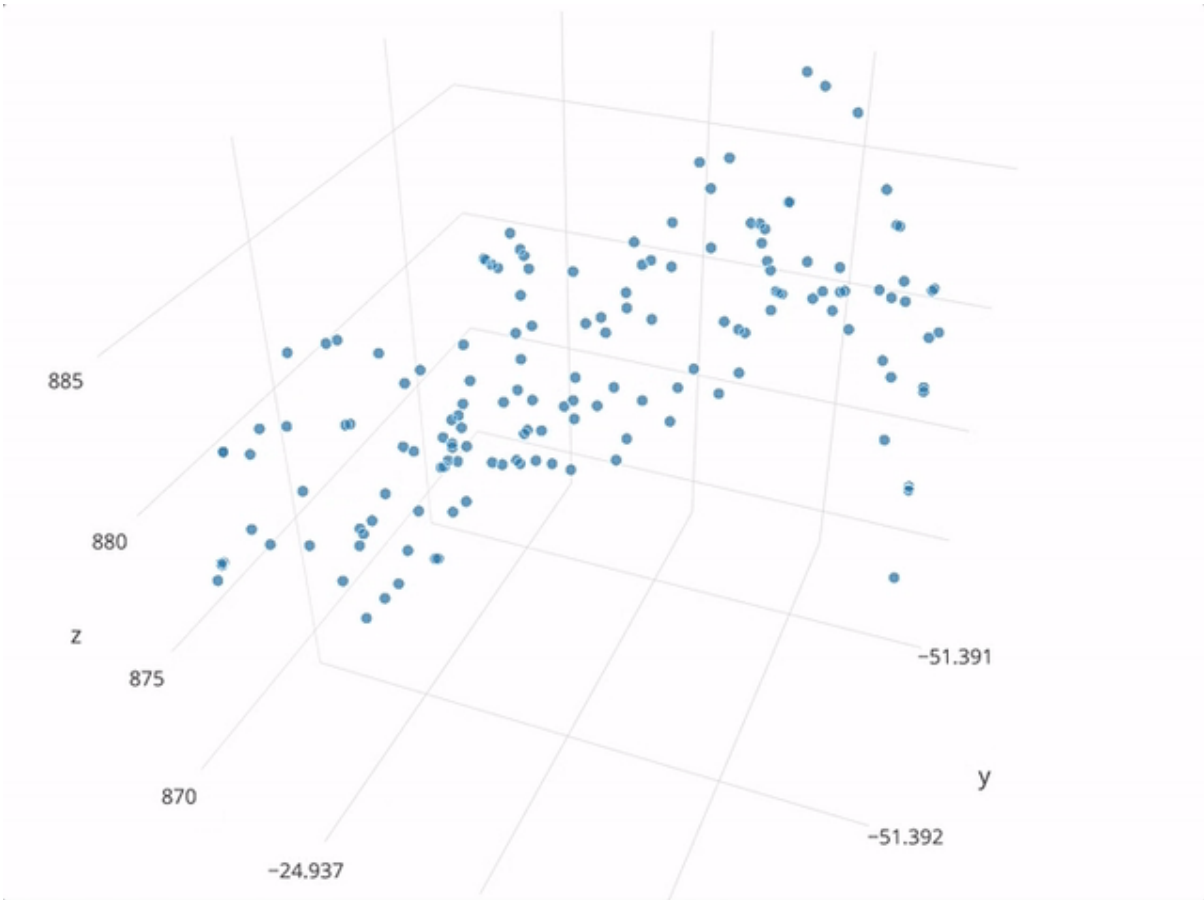
2017/05/23
05:38pm
IT WORKED!



11:10am

- So at the end of this chapter, we know how to calculate the volume of a solid object formed by a set of 3 points in space. Now it is a matter of ordering how this computations are to be made for a point cloud and sum all of them for total volume.
- Remember: mind the ordering of A,B,C >> lowest to highest x

Saving gif of our target example:



10:00am

- Determining ordering of integration factors:
    - A,B,C from lowest x to highest.

- Yesterday's accomplishments:
    - Successfully calculated volume delimited by a triangular area using double integral on line equations.
    - I need to take care of the zero divide issue, when the line equation is parallel to the y axis. In that case, volume = 0 and the total volume is to be given by a single double integral.
- Next steps:
    - Check volume calculations again known volume shapes.
    - Solve the zero divide issue [SOLVED]
    - Test inverting min/max from double integrals
        - ANS: Signal toggle, zero sum and different volume result. Think about ordering points.
            - Assumption1: x determines ordering. Define slopes using that order.
        - Inverting ordering of the y range causes signal

toggle.

- Assumption: don't worry about ordering y ranges, simply take abs() of the result. Mind x ordering only to determine which points are A,B and C.

In [9]:
```
# Defining a generic function to comput the above
def computeLineEquation(A,B):
    # computes line equation given 2 points: A and B
    # Both A and B need to be numpy arrays

    if (B[0]-A[0]) == 0: # Check if line is parallel to y axis (zero divide)
        print("Zero division. Line is parallel to y axis. x={x}".format(x=A[0]))
        return np.array([None,None])
    else: # detect slope and intersection
        m = (B[1]-A[1])/(B[0]-A[0]) # slope
        xo = A[0]
        yo = A[1]
        c = -m*xo+yo
        print("The line equation is y = {m}x + {c}".format(m=m,c=c))
        return np.array([m,c]) # returns slope and constant in numpy array format
```

In [11]:
```
slope, constant = computeLineEquation(np.array([1,2]), np.array([1,15]))
```

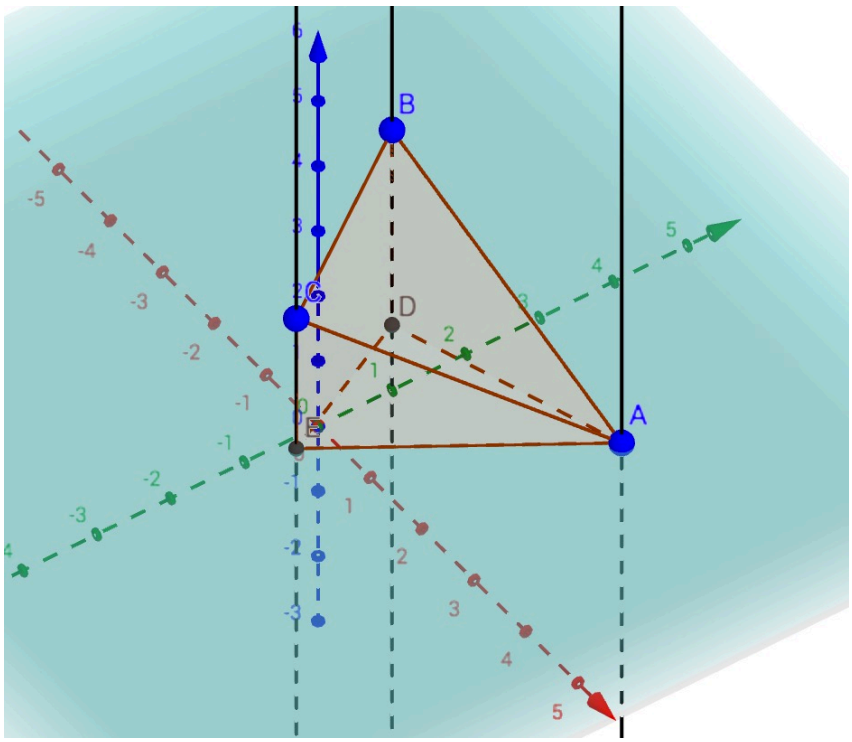
```
Zero division. Line is parallel to y axis. x=1
```

In [5]:
```
# the same but different...
slope, constant = computeLineEquation(np.array([1,2]), np.array([4,5]))
```

```
The line equation is y = 1.0x + 1.0
```

2017/05/22

- Suggestion from Renato Gasoto: Calculate volume based on pyramid and 3 sided polyhedron.



- Lowest z point is the top of the pyramid.
  - $\text{Volume}_{\text{Pyramid}}$ = 1/3 b*w*h

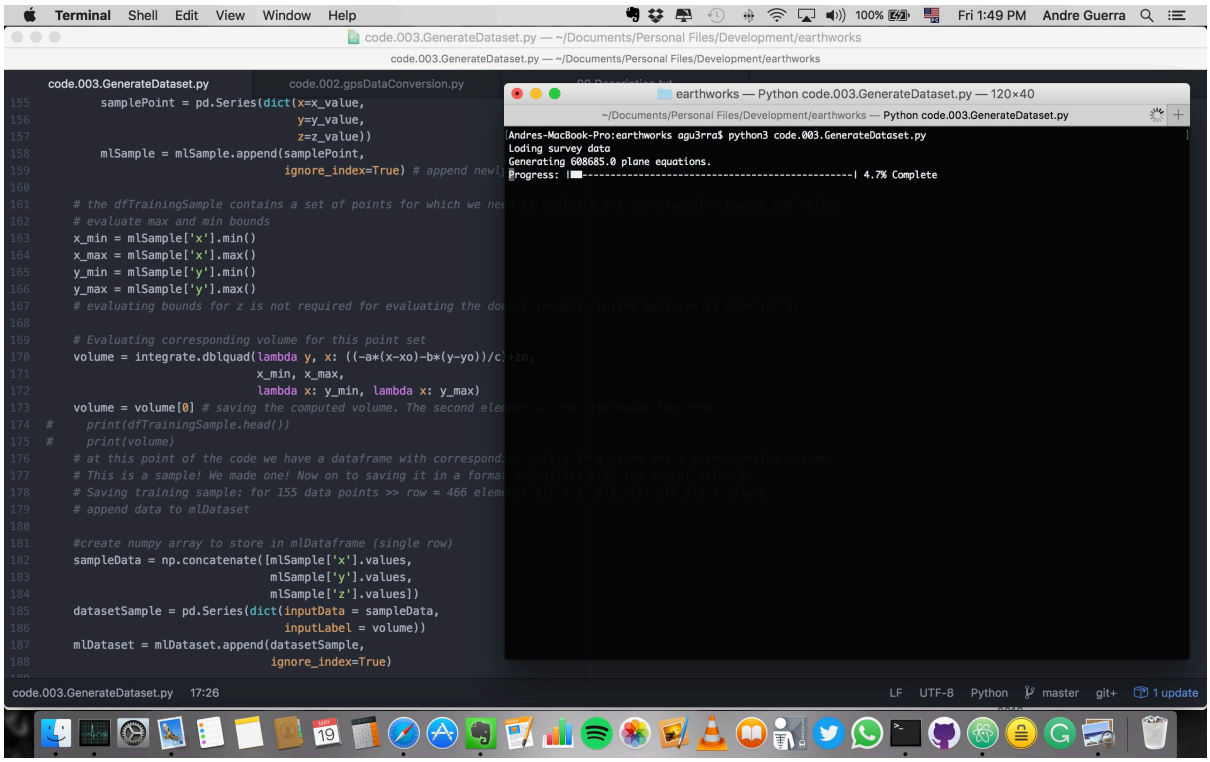
- 3 sided polyhedron: $Vol_{polyhedron}$ = Base area * Height
- Iteration design:
    - Order collection of points according to their distance from the origin (0,0,0).
    - Select first 3 entries (lowest distance from origin);
    - Project these points to the base plane (xy)
    - Calculate volume of corresponding solid
        - Option 1:
            - 1 pyramid volume+
            - 1 polyhedron volume
        - Option 2:
            - Generate plane equation from 3 points
            - Calculate double integral
    - What do I do?
        - Double integral seems to be the easiest exit as I already know how to compute a plane equation - z=f(x,y) and a double integral.
        - I can double check results agains a known set of points by building a clean visually easy to calculate set of points
        - Insight: y boundaries will be a function of x that delimits the triangle, not y_min and y_max.

2017/07/19

- Running dataset generation code for about 2 hours now and it's only 4.7% complete.
- Options:

1. Continue development stage on to Neural Network using a small sample dataset
2. Learn how to use Cloud Computing to decrease processing time.

- Amazon AWS
- Google Cloud Platform



2017/05/18

- Successfully managed to generate training example from existing data and match format to the expected neural network input.
- Next step: generate full training set based on plane equations.

2017/05/17

- Possible models to use:
  - Keras, Theano, TensorFlow, Sklearn
- http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html#sklearn.neural_network.MLPRegressor

Following Tutorial for Regression problem: http://machinelearningmastery.com/regression-tutorial-keras-deep-learning-library-python/

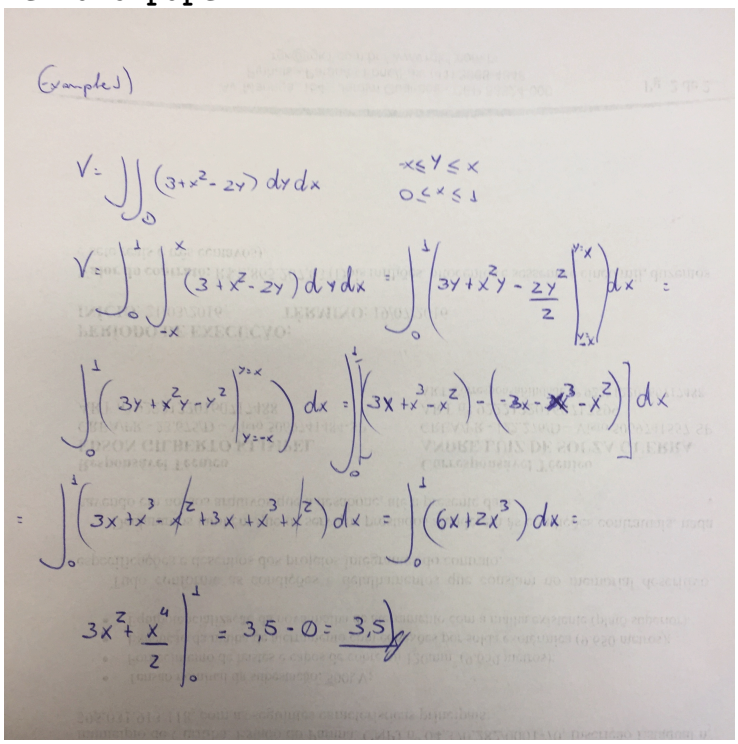
Success: I have managed to train a NN model and make predictions on new data.
Conclusion: I can indeed use a neural network to make predictions

---

2017/05/16

- Idea: Use a Machine Learning Regression model to obtain volume out of a given set of points
  - Generating training data:
    - Use double integrals to calculate for known z=f(x,y) shapes within the x,y,z ranges
      - Use cos/sin functions;
      - Use linear planar slopes;
- To do next: Calculate integrals analytically in Python
- Managed to calculate double integral in Matlab successfully:

Pen and paper



(Example 1)

$$V = \iint_D (3 + x^2 - 2y)\, dy\, dx \qquad -x \le y \le x \quad 0 \le x \le 1$$

$$V = \int_0^1 \int_{-x}^{x} (3 + x^2 - 2y)\, dy\, dx = \int_0^1 \left(3y + x^2 y - \frac{2y^2}{2} \Big|_{-x}^{x} \right) dx =$$

$$\int_0^1 \left(3y + x^2 y - y^2 \Big|_{y=-x}^{y=x} \right) dx = \int_0^1 \left[(3x + x^3 - x^2) - (-3x - x^3 - x^2) \right] dx$$

$$= \int_0^1 \left(3x + x^3 - \cancel{x^2} + 3x + x^3 + \cancel{x^2} \right) dx = \int_0^1 (6x + 2x^3)\, dx =$$

$$3x^2 + \frac{x^4}{2} \Big|_0^1 = 3{,}5 - 0 = \underline{3{,}5}$$

Matlab

```
>> fun = @(x,y) (3 + x.^2 - 2*y);
>> ymin = @(x) - x;
>> ymax = @(x) x;
>> q = integral2(fun,0,1,ymin,ymax)

q =

    3.5000
```

Python

```
integrate.dblquad(lambda x, y: 3 + x**2 - 2*y, 0, 1.0, lambda x: -x, lambda x: x)
```

```
(1.8333333333333333, 2.960584255223384e-14)
```

Successful call in Python:

```
In [28]: integrate.dblquad(lambda y, x: 3 + x**2 - 2*y, 0.0, 1.0, lambda x: -x, lambda x: x)
Out[28]: (3.4999999999999996, 8.85288647725923e-14)
```

Now on to plotting this graph!

Ok, so reaching the end of the day, I was able to:

- Calculate double integrals using Python.
- Plot corresponding graph data

Need to learn:

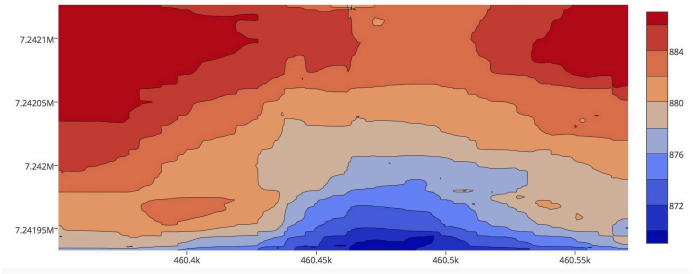
- Which machine learning algorithm to use.
  - It'll probably be a regression one, as I want the model to output a value (volume) given input data (a collection x,y,z coordinates)
- Determine how to generate training data
  - Set of mathematical functions and their limits to apply the double integral program.
- Assumption: a bunch of known well defined mathematical function will generalize well for my set of points. Maybe there's some magic that needs to be discovered as to define which functions may be the best for which type of terrain
  - sin/cos
  - linear/quadratic slope
- Regression models:
  - Ordinary Least Squares Regression (OLSR)
  - Linear Regression
  - Logistic Regression (classification)
  - Stepwise Regression
  - Multivariate Adaptive Regression Splines (MARS)
  - Locally Estimated Scatterplot Smoothing (LOESS)

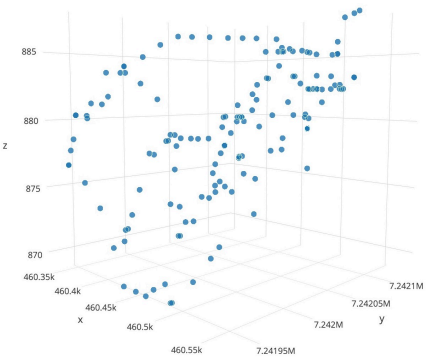
2017/05/12

- Managed to plot 3D data with acceptable display speed using Plotly
- Managed to apply numerical integration in Python using Scipy. https://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html#integrating-using-samples
- Ideas:
  - Preserve elevation data when plotting, convert only latitude/longitude
  - Choose map projection that generates the most accurate

distances.

- I've used UTM conversion with corresponding Python library
- I've managed to plot contour map and 3D scatter plot.





- Completed XML parser in Python to read gpx files and convert them into a csv file with latitude, longitude, altitude.
- These are in a spherical coordinate system so they must be converted to polar for an easy to work cartesian projection.
- What I really need is to convert from a geographic coordinate system to cartesian.

Geographic coordinates to Cartesian coordinates

These formulae can be used to convert geographic coordinates, latitude (Φ), longitude (λ), and height (h), into Cartesian coordinates (X, Y, Z):

$$X = (\nu + h)\cos\phi\cos\lambda$$
$$Y = (\nu + h)\cos\phi\sin\lambda$$
$$Z = \left[\nu(1 - e^2) + h\right]\sin\phi$$

Where a and f are obtained from the reference ellipsoid used for the respective geodetic datum and the h is the height of the computation point or approximated as zero

$$\nu = \frac{a}{\sqrt{1 - e^2\sin^2\phi}}$$
$$e^2 = 2f - f^2$$

(first) flattening	f	$\frac{a-b}{a}$	Fundamental. Geodetic reference ellipsoids are specified by giving $1/f$

From Spherical to Cartesian:

To Cartesian coordinates [edit]

From spherical coordinates [edit]

Main article: spherical coordinates

$$x = \rho \sin\phi \cos\theta$$
$$y = \rho \sin\theta \sin\phi$$
$$z = \rho \cos\phi$$

$$\frac{\partial(x,y,z)}{\partial(\rho,\theta,\phi)} = \begin{pmatrix} \sin\theta\cos\phi & \rho\cos\theta\cos\phi & -\rho\sin\theta\sin\phi \\ \sin\theta\sin\phi & \rho\cos\theta\sin\phi & \rho\sin\theta\cos\phi \\ \cos\theta & -\rho\sin\theta & 0 \end{pmatrix}$$

So for the volume element:

$$dx\ dy\ dz = \det\frac{\partial(x,y,z)}{\partial(\rho,\theta,\phi)} d\rho\ d\theta\ d\phi = \rho^2 \sin\theta\ d\rho\ d\theta\ d\phi$$

https://en.wikipedia.org/wiki/World_Geodetic_System
The World Geodetic System (WGS) is a standard for use in cartography, geodesy, and navigation including GPS. It comprises a standard coordinate system for the Earth, a standard spheroidal reference surface (the datum or reference ellipsoid) for raw altitude data, and a gravitational equipotential surface (the geoid) that defines the nominal sea level.

Ellipsoid reference	Semi-major axis *a*	Semi-minor axis *b*	Inverse flattening (1/*f*)
GRS 80	6 378 137.0 m	≈ 6 356 752.314 140 m	298.257 222 100 882 711...
WGS 84	6 378 137.0 m	≈ 6 356 752.314 245 m	298.257 223 563

Calculations from Python terminal
f = 0.003352810664775694
e = 0.08181919084296517
v=6378951.26996405 (forgot to conver to radians before applying m.sin())
v=6381935.230708307

Implementing Python program for calculation using these 2 points as reference:
point1=(-24.9361993186,-51.3906013593,883.16)
point2=(-24.9356264155,-51.3912779465,882.69)
euclidean distance computed: 93.27m
- Information checks out according to distance measurement tool on Google Earth.

Next step: plot cartesian frame data in 3D

References:
https://en.wikipedia.org/wiki/Spherical_coordinate_system

https://en.wikipedia.org/wiki/Azimuth
https://en.wikipedia.org/wiki/List_of_common_coordinate_transformations#From_spherical_coordinates
https://www.mathworks.com/help/matlab/ref/sph2cart.html#inputarg_azimuth

---

Given a set of points (cloud) x,y,z coordinates:

- Build a plane connecting all points with splines
- Define another plane that cuts thru it
- Determine cut and fill volumes.