

Contact: Tonia Robinson; tonianr@usf.edu

An Interactive Seismic Refraction Plot using Obspy: graphSEG2

This project focuses on creating an interactive Seismic Refraction plot to save data points related to the direct and refracted wave returns. The use of seismic refraction datasets is crucial to Near-surface geophysical explorations and are often used to delineate depths to bedrock. This project will utilize the Obspy python framework to read SEG2 refraction datasets and plot interactive stream datasets using the [matplotlib.pyplot.ginput](#) function.

This document presents various ways to use the code provided with sources of supporting materials. Possible **errors** and how to address them are discussed at the end of the document.

graphSEG2.v2

graphSEG2 uses the [Obspy](#) framework to create an interactive plot of Seismic Refraction datasets. To begin use, make sure to [install the Obspy framework](#) and initialize the environment using the command while in the anaconda command prompt: `conda activate obspy`

1. Importing function

To import the function, use the code:

```
from graphSEG2_v2 import graphSEG2
```

This allows the user script in use to have access to the graphSEG2 function and its executables.

2. graphSEG2.plot


graphSEG2.plot allows the user to plot Seismic Refraction datasets by utilizing the [obsypy.read](#) SEG2 support function. To execute the graphSEG2.plot, the user will need to input a *filename*, *gain*, *ymax*, *nclicks*, *original name*. The user will also use this function to make first arrival picks for analysis.

```
graphSEG2.plot(filename, gain, ymax, nclicks, originalname)
```

```
graphSEG2.plot('300.dat', 15, 100, 24, 'false')
```

Where:

- i. **Filename** (*str*) – should contain a string, e.g., '103.dat.'
- ii. **Gain** (*int*) – should be an integer, e.g., 1 for no gain or 10 to amplify the dataset by a factor of 10
- iii. **Ymax** (*int*) – should be an integer value related to the desired time window the user wishes to view, e.g., if first arrivals are within the first 100 ms, the user would add the

integer 100. Zoom tool  in interactive figure window also does this, however the user risks added an unwanted selection point. If the user does decides to use the zoom tool, the user should click the *right mouse* after zooming to remove any unwanted point selections.

- iv. **Nclicks** (*int*) – typically the number of geophones used within the survey, e.g., 24
- v. **Original name** (*str*) – program asks if the user wants to save the file's original name after completing the first arrival picks. 'true' indicates that the user would like to save picks with the files original name while 'false' prompts the user to enter the desirable

name after completing picks. The input filename should be *filename.txt*, with no quotations.

After running the above code, the user is prompted to read a mini guide and informed to press *ENTER* to continue or *CTRL+C* to end.

2.1 Making First Arrival Picks

First arrival picking in the module utilizes [matplotlib.pyplot.ginput](#) for interactive selections. When picking points:

- selecting → *LEFT MOUSE*
- removing point → *RIGHT MOUSE*
- end selection → *ENTER*

Note that the user defined **Nclicks** is the maximum number of selections the user is allowed but one can exit the selection mode by simply pressing *ENTER*.

3. graphSEG2.analyze

graphSEG2.analyze allows the user to plot and analyze Seismic Refraction first arrival picks using the Slope Break method (see 3.1). To execute the graphSEG2.plot, the user will need to input a *pick file and show_slopebreaks*.

```
graphSEG2.analyze(pckfilename, show_slopebreaks)
graphSEG2.analyze('300.txt', 'false')
```

Where:

- i. **Pckfilename** (*str*) – should contain a string of the text file, e.g., '103.txt.' This text file should have x,y dataset space or tab-separated. Where the x column is the distance, and the y column is the time.
- ii. **Show_slopebreaks** (*str*) – indicates whether the user would like to plot all the slope breaks on the analysis plot; 'true' shows slope breaks while 'false' plots only the analysis and final slope break.

3.1 Slope break method

The Slope Break method applies a single model (eqn.1) with the slope and intercept of two lines; this linear model is then fitted to each dataset (fig.1).

$$\text{eqn 1. } y = b_0 + b_1 + b_2D + b_3Dx$$

Slope breaks are identified by the location of the minimum slope uncertainty for slopes before and after a set kink, i.e., where both lines have a combination of the lowest slope uncertainties.

The equation is solved by computing a simple least-squares calculation and solving for the model parameters, which in this case is the *b* array:

$$y = Ab \text{ thus } b = A \backslash y$$

Where the *A* array is a combination of the *x* and the *D* arrays.

Example for 5-point array:

$$y = \begin{bmatrix} y \\ y \\ y \\ y \\ y \end{bmatrix} \quad A = \begin{bmatrix} 1 & x & D & D * x \\ 1 & x & D & D * x \\ 1 & x & D & D * x \\ 1 & x & D & D * x \\ 1 & x & D & D * x \end{bmatrix}; \quad b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

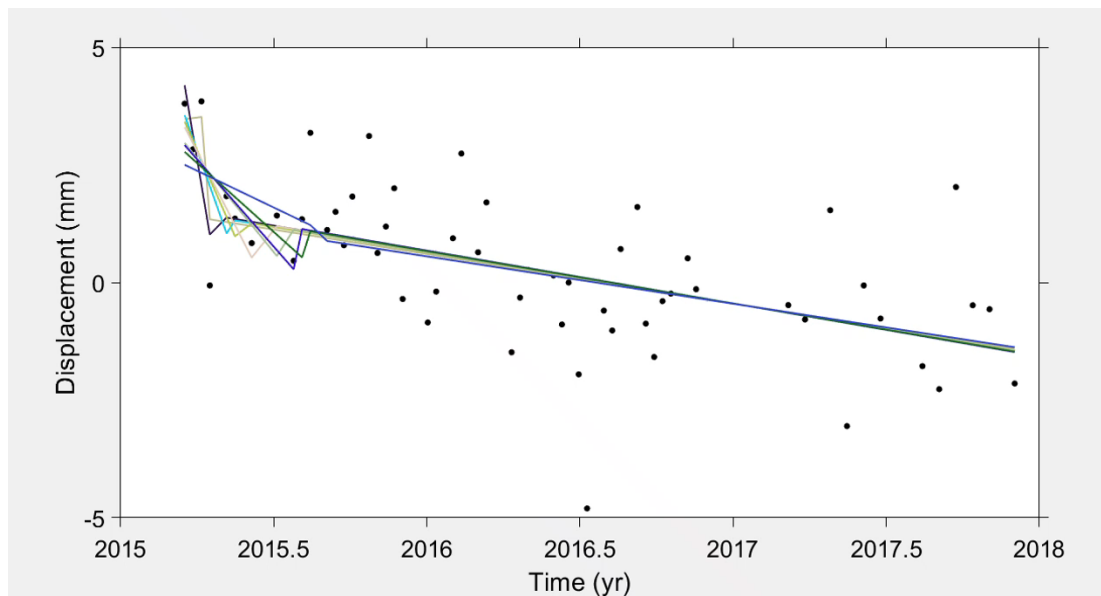


Fig.1.

3.2 Linestatistics.py

This function is implemented in the graphSEG2.analyze function, and it is used to calculate line statistics parameters for each line before and after the Slope Break analysis has been applied. Linestatistics simply uses the input of x,y variables found in a workspace and uses matrix computations. Linestatistics returns the variables: **m**, **b**, **m_uncertainty**, **b_uncertainty**, **R2**, which are the slope, intercept, slope uncertainty, and R-squared for a given linear dataset.

3.2.1 Running Linestatistics.py

To import the function, use the code:

```
from linearstatistics import linestatistics as lstats
```

Run e.g.

```
Statistics = lstats(x,y)
```

4. graphSEG2.crossover

graphSEG2.crossover allows the user to plot Seismic Refraction first arrival picks from text file and select desired crossover point. To execute the graphSEG2.crossover, the user will need to input a *filename*.

```
graphSEG2.crossover(filename)
graphSEG2.crossover ('300.txt')
```

Where:

- i. **Filename** (*str*)—should contain a string, e.g., '103.txt.' This text file should have x,y dataset space or tab-separated. Where the x column is the distance, and the y column is the time.

4.1 Findclosestpoint.py

The callable module Findclosestpoint.py was created from a code found on the following forum: <https://codereview.stackexchange.com/questions/28207/finding-the-closest-point-to-a-list-of-points>

It is used to find closest point location within a list of points, it returns the index of the closest point. In graphSEG2.crossover this code uses the user selected crossover location to find the index of the point in the input file.

4.1.1 Running Findclosestpoint.py

To import the function, use the code:

```
from findclosestpoint import closestpoint
```

Run e.g.

```
Idx_closepoint = closestpoint(pt, pts)
```

Where:

pt (*array or list*)—should be a list or array with size **2**, it is a single (x,y) point dataset

pts (*array or list*)—should be a list or array with size (**n, 2**), where **n** is the total number of points, it is a series of datapoints

4.1.2 Example code:

```
from findclosestpoint import closestpoint
import numpy as np

x = [1,2,3,4,5,6,8]
y = [1,2,3,4,5,6,8]
a = np.stack((x,y))
a = a.T
```

```
pt = [3.12, 3.12]
w = closestpoint(pt,a)
print(w)
```

****Utilize numpy.stack to ensure your array is useable with code.**

4.2

5. Other

5.1 Plotting two SEG2 files

To plot two SEG2 files, the user will need to run graphSEG2.plot twice for both files, bypass the first plot after running by pressing CTRL+C, then continue to run the 2nd line (making sure the first is commented out or in a separate cell). To create separate cells, use `#%%`

Example:

Run:

```
1 graphSEG2.plot("300.dat",15, 100,24,'false')
```

>> **CTRL + C** in the console after plotting, leave figure window open

Run:

```
2 graphSEG2.plot("301.dat",15, 100,48,'false')
```

When graphing two SEG2 files, note that the 2nd file input will overwrite the first, e.g., if you want to make 48 picks (both lines together), you will need to add 48 **nclicks** on line 2. It is also best to leave the **original name** as 'false' so that you can name our file as desired.

5.2 Analyzing two SEG2 files

To run graphSEG2.analyze for two or more SEG2 files, the user will need to run graphSEG2.plot twice for both files.

Example:

```
1 graphSEG2.analyze('102_reflexw.txt','true')
2 graphSEG2.analyze('102.txt','true')
```

5.3 Loading file from another folder

When loading file from folder make sure to source folder.

Example:

```
1 graphSEG2.plot('./foldername/300.dat',15, 100,24,'false')
```

OR

```
2 graphSEG2.analyze('./foldername/102.txt', 'true')
```

OR

```
3 graphSEG2.crossover('./foldername/102.txt')
```

5.4 Saving pick txt file to specified folder

When prompted to input output file name after picking first arrivals (graphSEG2.plot original name set to 'false') user can add specific file location.

Example when prompted in console:

```
Write the output file's name as filename.txt:  
./foldername/103.txt
```

5.5

6. Possible Hiccups

6.1 non-GUI Backend Error

6.1.1 Error message in console: *Matplotlib is currently using agg, which is a non-GUI backend, so cannot show the figure.*

Fix by changing IPython Console graphics backend to Qt5 in the Preferences window. Source: <https://stackoverflow.com/questions/36700083/in-spyder-plot-using-matplotlib-with-interactive-zoom-etc>

6.2