SigMT - User Manual

v. 1

K.S. Ajithabh

June 11, 2022

Introduction

SigMT is a python package designed for the processing of the raw magnetotelluric (MT) data to obtain the MT impedance and tipper estimates. It works in an automated way, so that manual time series inspection and editing are not required. Mahalanobis based data selection tool is implemented in the package to avoid the manual editing of time series. The final impedance estimation is done using the robust estimation method. Different data selection tools such as coherency threshold, polarization direction are included in this package. This document is a guide to use the SigMT package. At present, only '.ats' file formats from ADU07 (Metronix Geophysics) is supported.

Contents

1 Supported file formats

- *.ats file format from ADU-07 by Metronix Geophysics

2 Getting started

The program is written as a project in Spyder IDE with the support of 'Anaconda' Python Distribution. I suggest installation of the Anaconda first to start with this package. You can download Anaconda for Windows/Linux/Mac from Anaconda website. Please install Anaconda3-2021.11 version from the Anaconda archives using the link https://repo.anaconda.com/archive/. Because newer version is showing some errors. Anaconda provides all necessary python modules for the scientific computations.

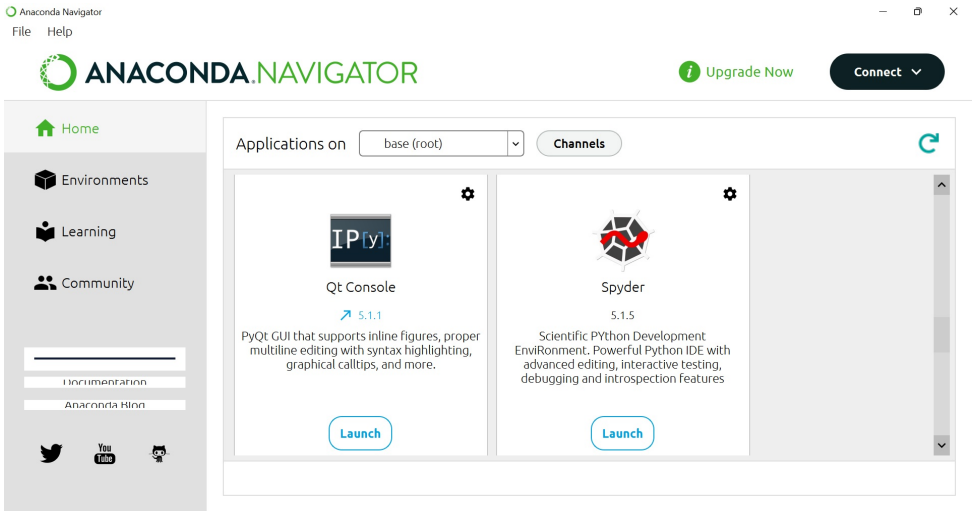

Figure 1: Anaconda Navigator

After installation of Anaconda, you can check in Anaconda Navigator (Figure 1) whether Spyder is installed or not. If not installed by default, please install Spyder using install button in Anaconda Navigator.

Once Anaconda and Spyder is ready, you can download the SigMT package as zip from GitHub (Figure 2). Extract the compressed folder to your local drive. Then open 'Spyder IDE', select Projects and click 'Open Project' (Figure 3).

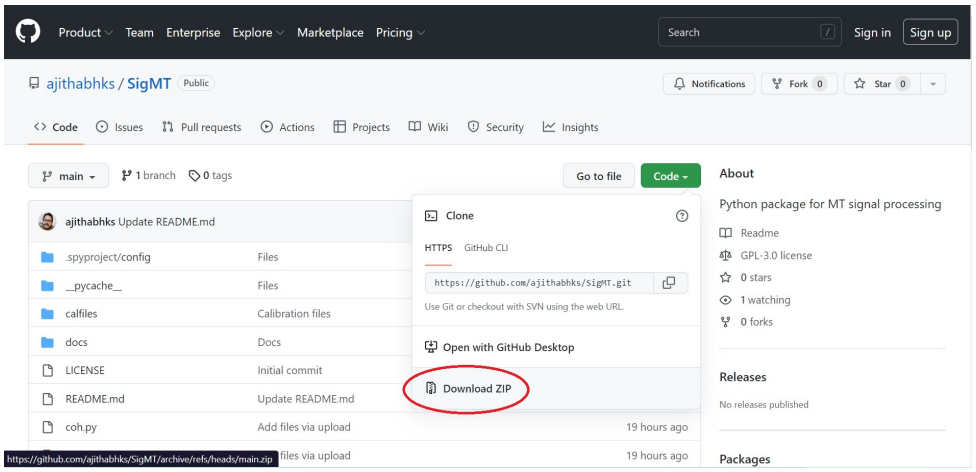

Figure 2: Download the package

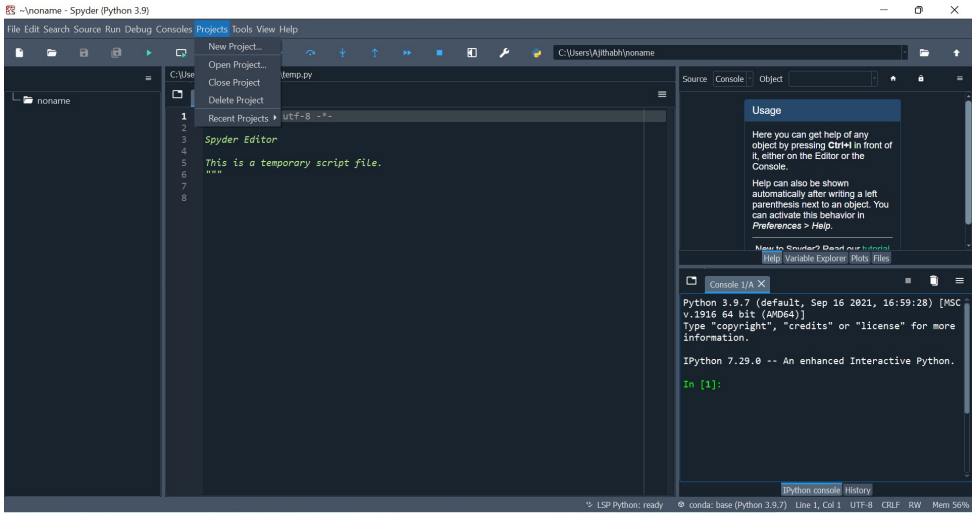

Figure 3: Open Project in Spyder

Navigate to extracted SigMT project folder and select it. Now you are in SigMT package. Before running the package, you have to install the 'mne' package using the pip. Run following command in Spyder console to install it.

```
pip install mne
```

Now add the calibration files of your MFS06 sensors to the 'calfiles' folder in SigMT package folder. Please use sensor number as file name (as in Figure 4), otherwise program will not read it. Some sample files are given by default in 'calfiles' folder. Please try to make your files similar to it.

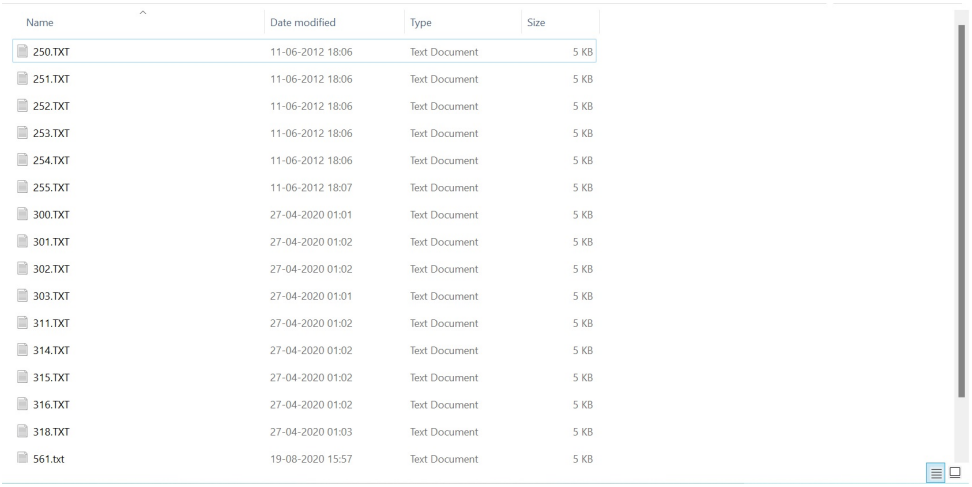

Figure 4: Calibration files

Now you are all set to start processing your MT data!!

3 Overview of the package

The file 'main_script.py' is the file you need to run the package for single site processing. But, if you need to carry out Remote reference, please use 'main_script_RR.py' file.

The files such as mtproc.py, mtprocRR.py, coh.py, coherency.py, mahaDist.py, selstack.py, tipper.py are written to perform different tasks in package and need not to be edited.

I suggest editing of main_script.py and main_script_RR.py as per your needs. The more description of these files are given below.

First of all, create a project folder and copy all your MT sites in it (as in Figure 5).

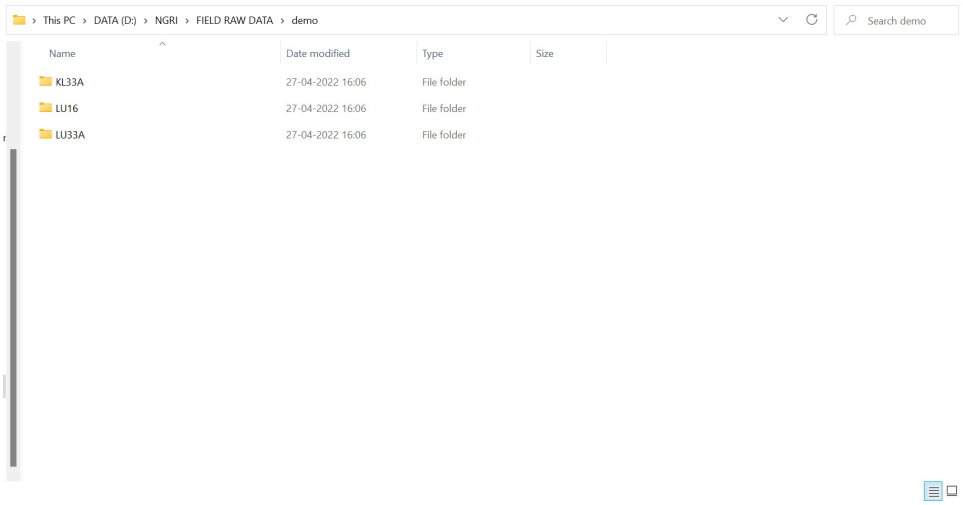

Figure 5: Project folder

The measurements should be in the site folder as in Figure 6. There should not be any other extra folders inside the site folder.

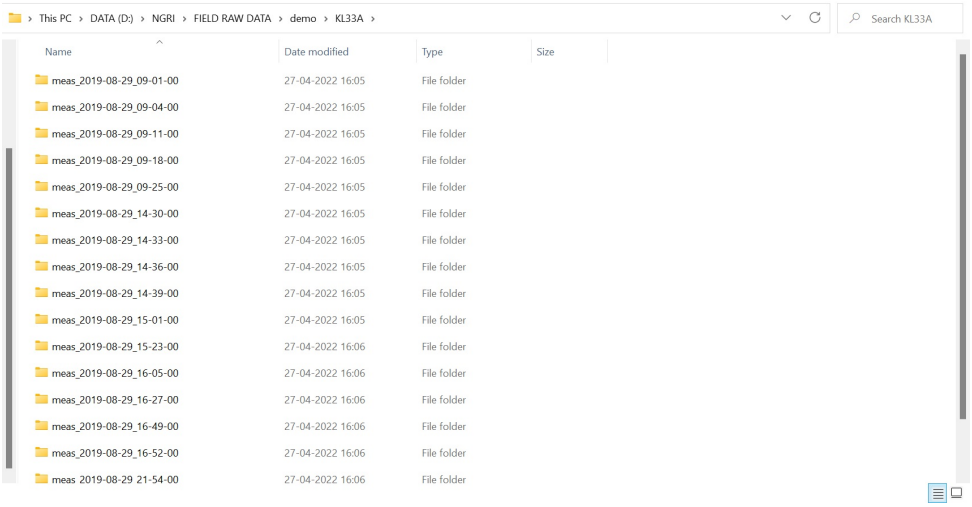

Figure 6: Site folder

There are three places, code can be modified in 'main_script.py' and 'main_script_RR.py' files to provide site folder path, decimation and data selection constraints.

First edit is required to provide the sites folder path. Copy the location of folder where sites are kept and copy to the variable 'project_path' as in Figure 7.

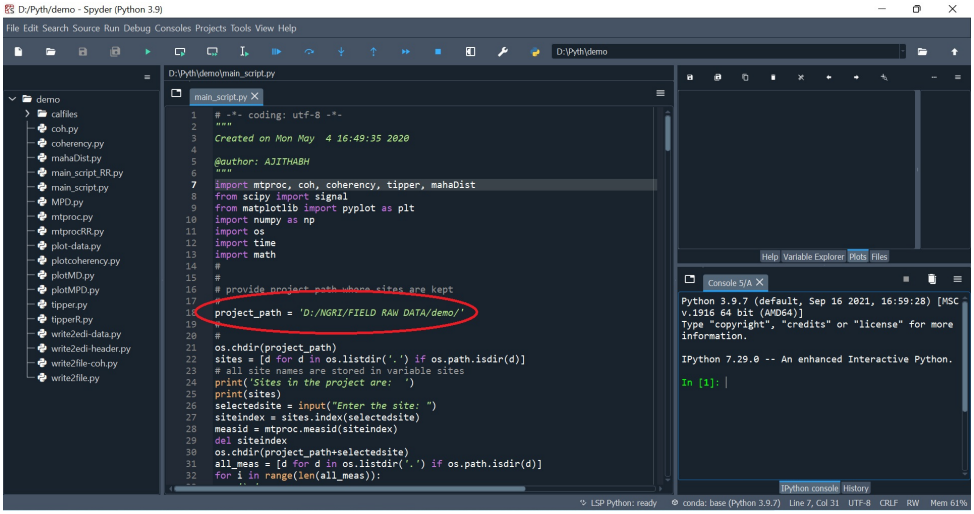

Figure 7: Give project path (sites folder)

Second edit is required in case you require decimation. The highlighted portions in Figure 8 is written to perform decimation. Keep 'dflag' 1 if you need to used decimation, else always keep 0. Suppose you need to decimate the data sampled at 32 Hz. If you use [8,4], then the data will be first decimated to 32/8 = 4 Hz, then 4/4 = 1 Hz. So, the output will be 1 Hz. Direct decimation using [32] is not recommended. If you used [8,8,4], the data will be decimated to 0.125 Hz. Similarly, you can decimate to any sampling frequencies.

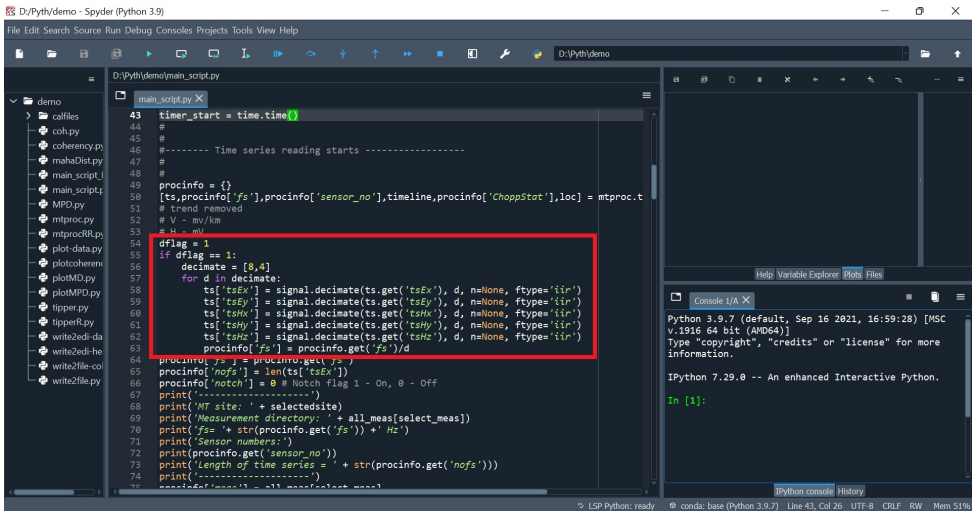

Figure 8: Decimation

Third edit is required to control data selection tools. The highlighted portions in Figure 9 is written to perform data selection. Coherency threshold and polarization direction based selections are included in the package. More detailed description is given in section 4.3.

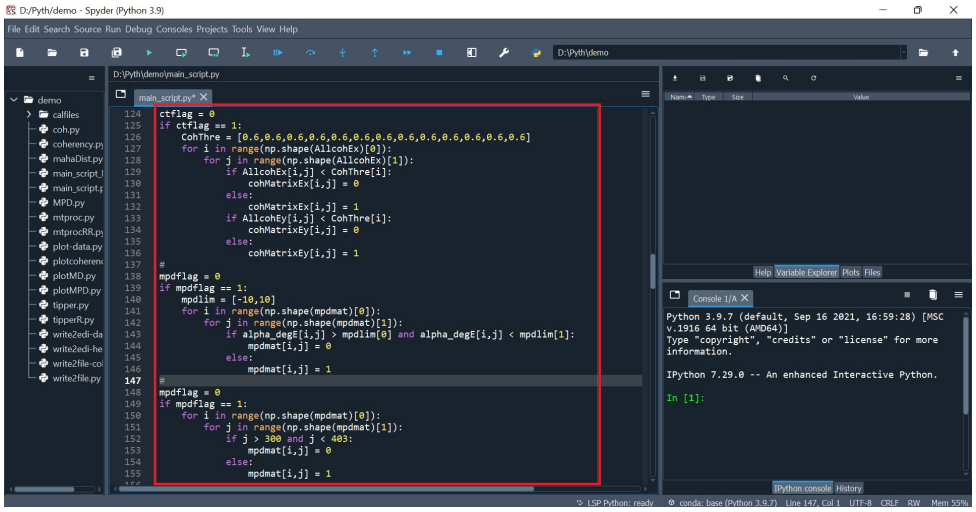

Figure 9: Data selection tools section

4 An example

4.1 Running the program

I am showing an example of processing here. Give the sites folder path in 'main_script.py' file and run the code as in the Figure 10

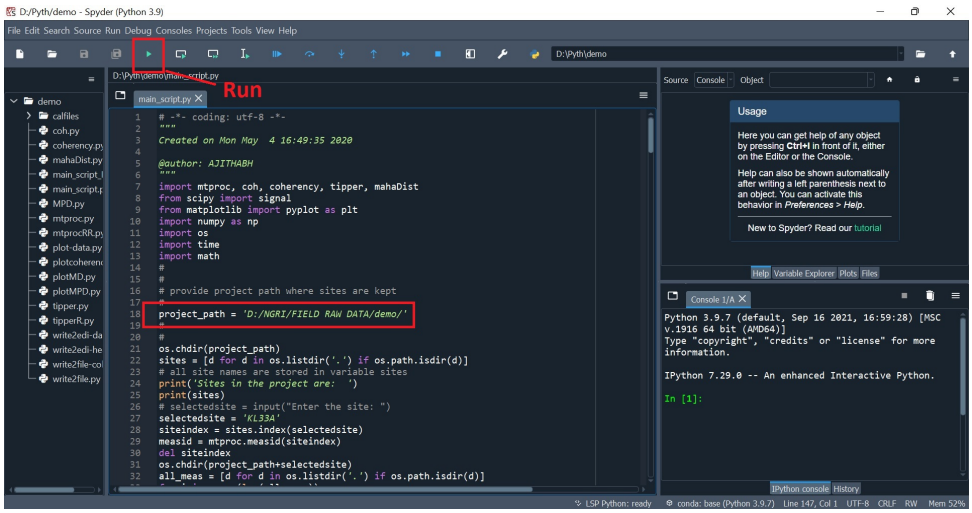

Figure 10: Give path and run the program

Once we run the program, the names of all sites will be displayed in the console (Figure 11). Then enter the site name to be processed and click 'Enter'.

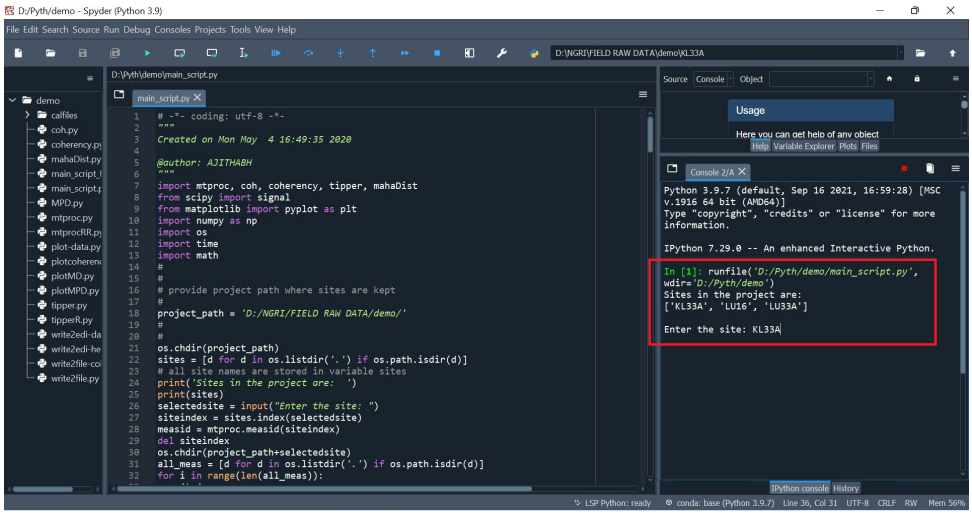

Figure 11: Enter the site name from the list

Then all measurements in the selected site will be displayed. Give the measurement number as input. For example, I selected number 16 for the processing in Figure 12.

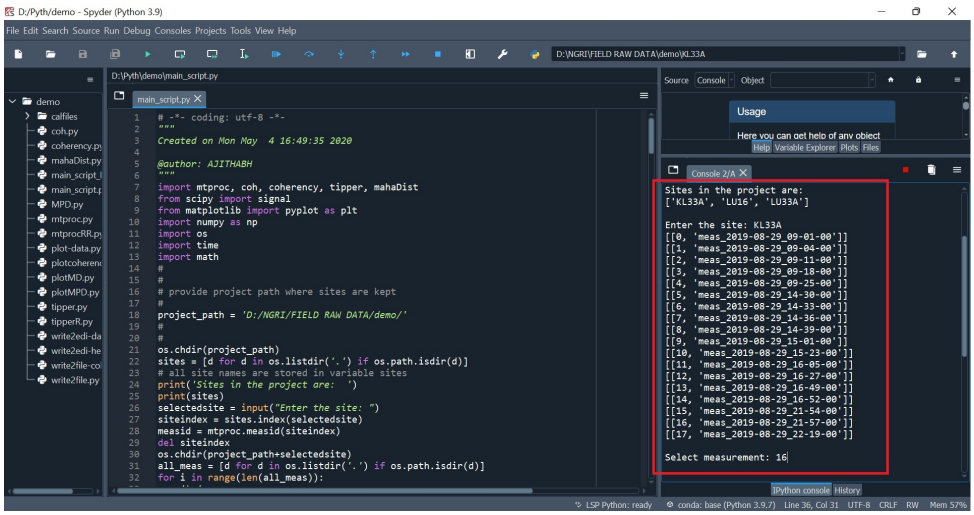

Figure 12: Enter the measurement number

Then some details about the measurement such as sampling frequency, coil numbers, length of time series, window length, and number of stacks will be displayed Figure 13. Then the processing will be started.

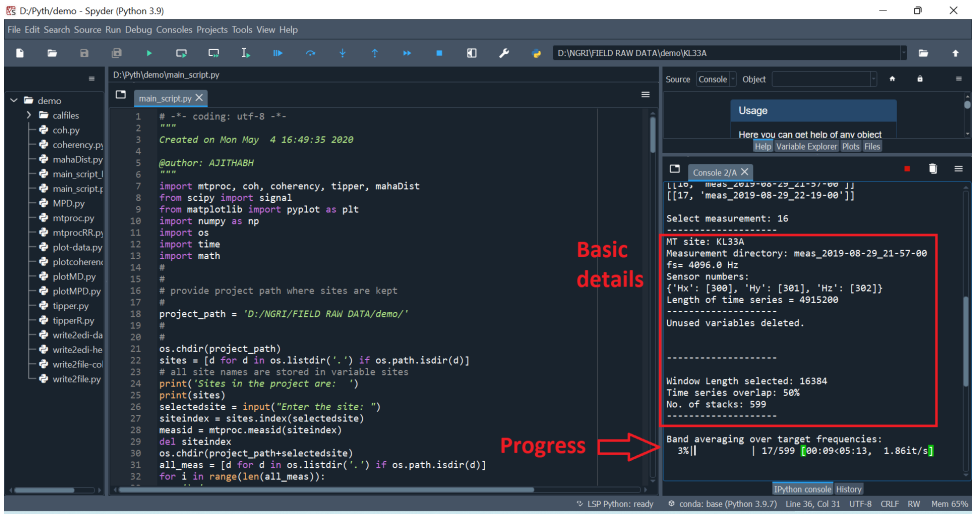

Figure 13: Processing..

After completing the processing, figure showing apparent resistivity and phase curves, coherency values and tipper data will be plotted and can be seen in 'Plots' section (Figure 14).

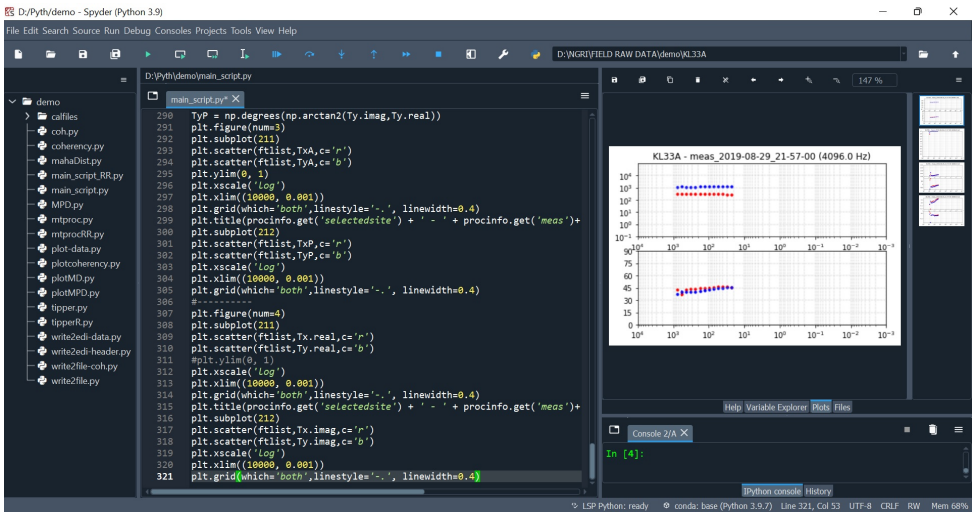

Figure 14: Output figures

4.2 Creating EDI file

The data is processed band wise. After completing processing for all measurements, the data should be joined to create the EDI files. One site may have many measurements. So, we have to save data in text files.

Create a folder anywhere. For example, a folder named 'Outputs' (as in Figure 15). Inside that folder, make a folder with site name. Create a folder named 'bands' inside site folder (Figure 15).

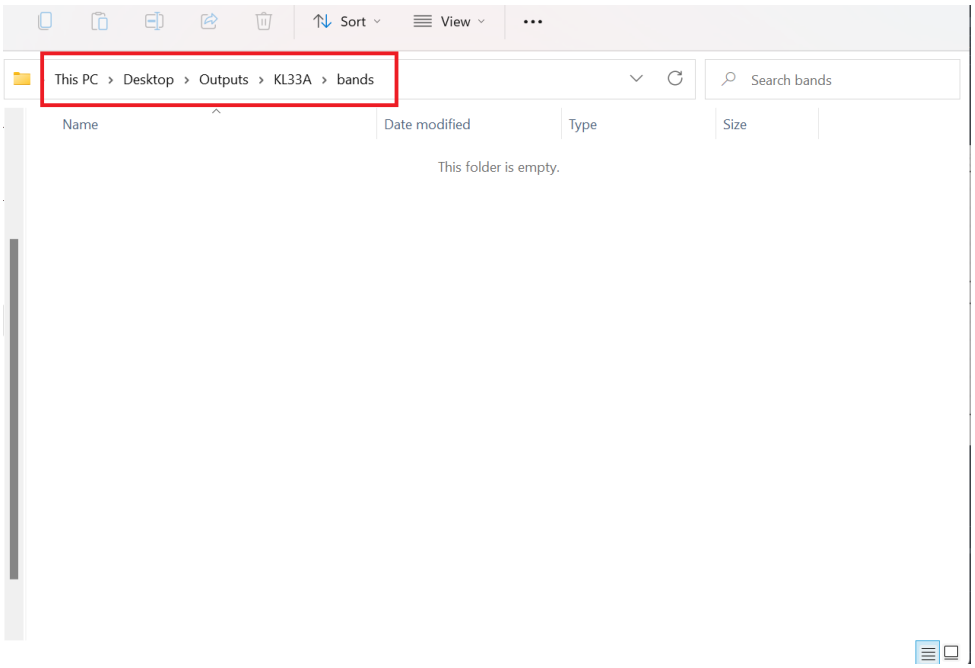

Figure 15: Create a folder to save processed data

Now come to Spyder and open 'write2file.py' script. Give the path to bands folder in

the variable ‘f’. Give sampling freqency as file name. For example ‘4096Hz.txt’. Select all script lines using ‘Ctrl+A’ and right click. Select ‘Run section or current line’ (Figure 16).

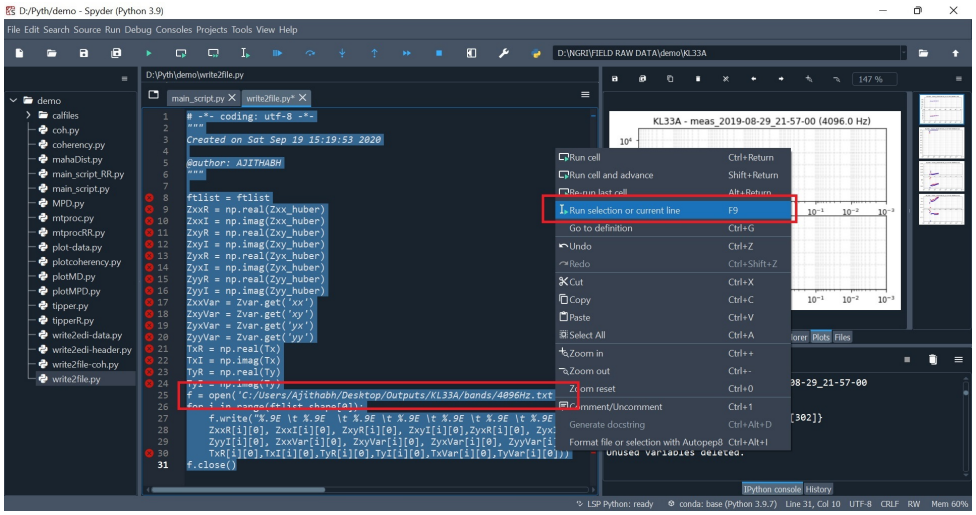

Figure 16: Save data in file

Now save header information to a file in site output folder. It is required for EDI file creation. Open ‘write2edi-header.py’ file and give path to sites folder (output) in variable ‘f’. Select all lines, right click and select ‘Run section or current line’ (Figure 17)

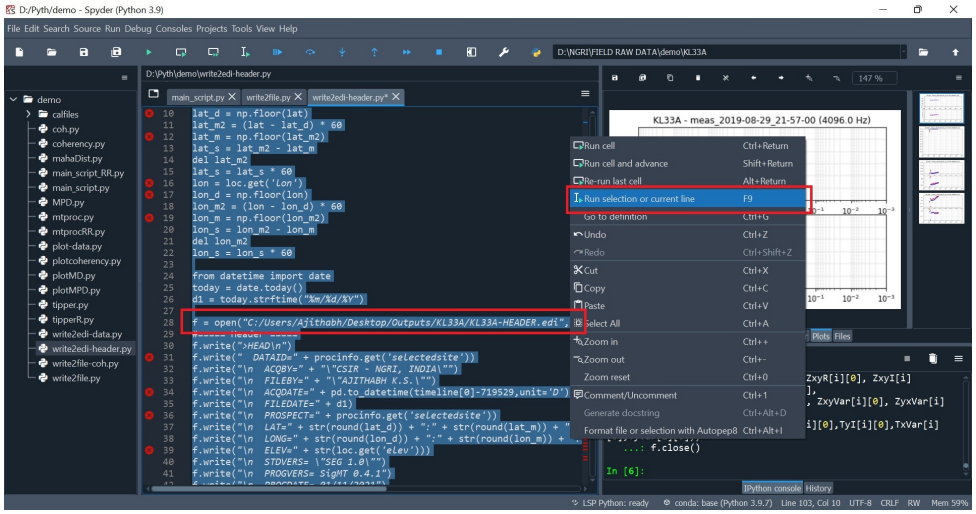

Figure 17: Save header information

After executing processing for a measurement, always close ‘Console’ (Figure 18).

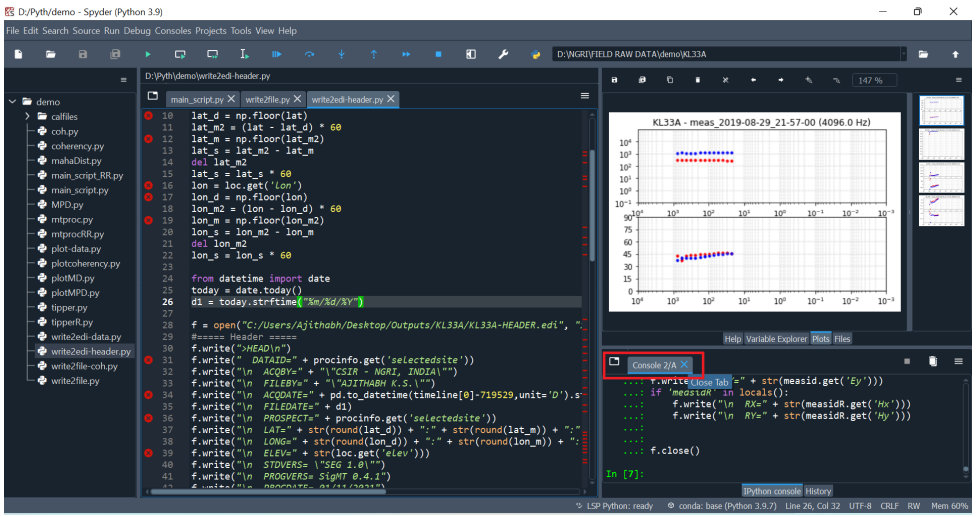

Figure 18: Close console after processing a measurement

Similarly, I processed for 1024 Hz measurement and save data in bands folder of KL33A. Now the measurement data can be seen in the folder (Figure 19). Create a text file named 'all.txt'.

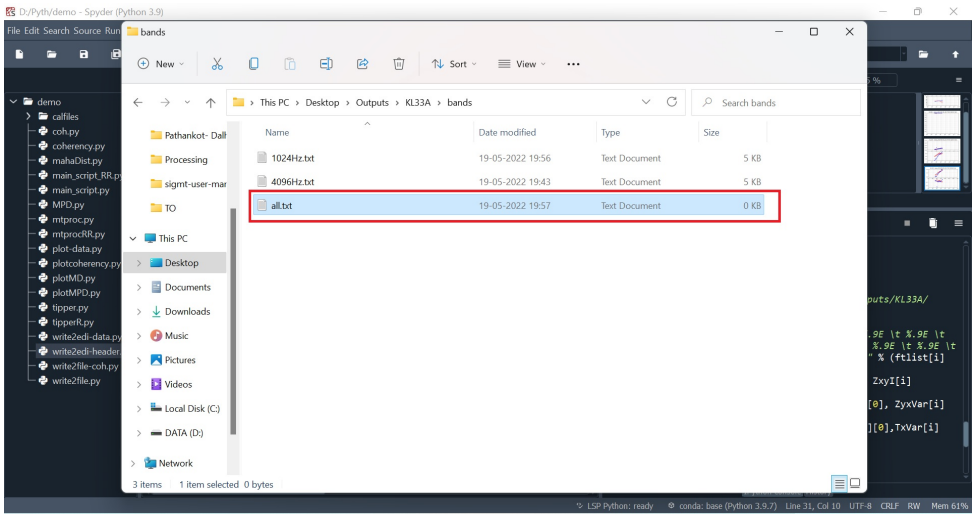

Figure 19: Create a file all.txt

Copy the data from 4096Hz.txt file and 1024Hz.txt file to all.txt as in Figure 20. Always keep higher frequency data in the top.

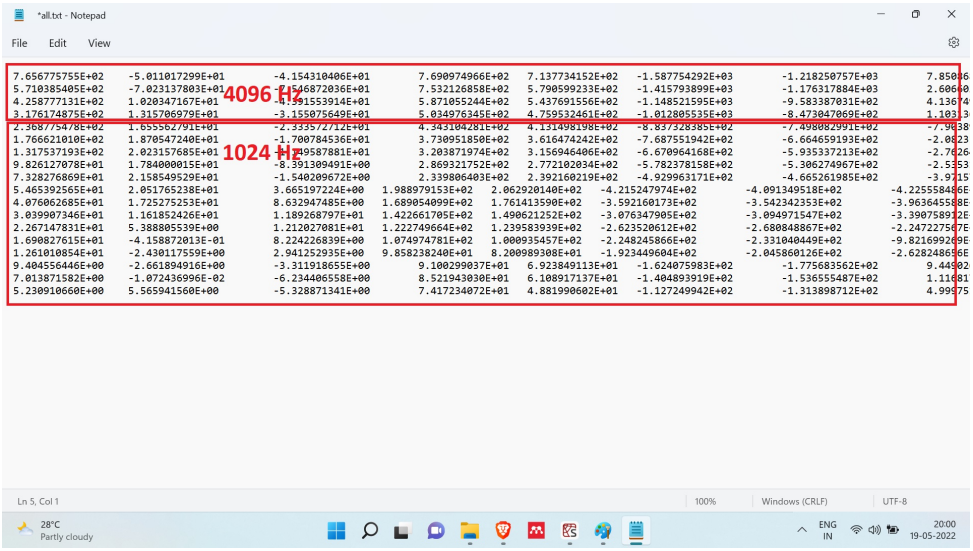

Figure 20: Copy data to all.txt file

Now, again go to Spyder and open 'write2edi-data.py' file. Give file path to 'all.txt' and path of sites folder in variables 'edifilname' and 'f' (Figure 21).

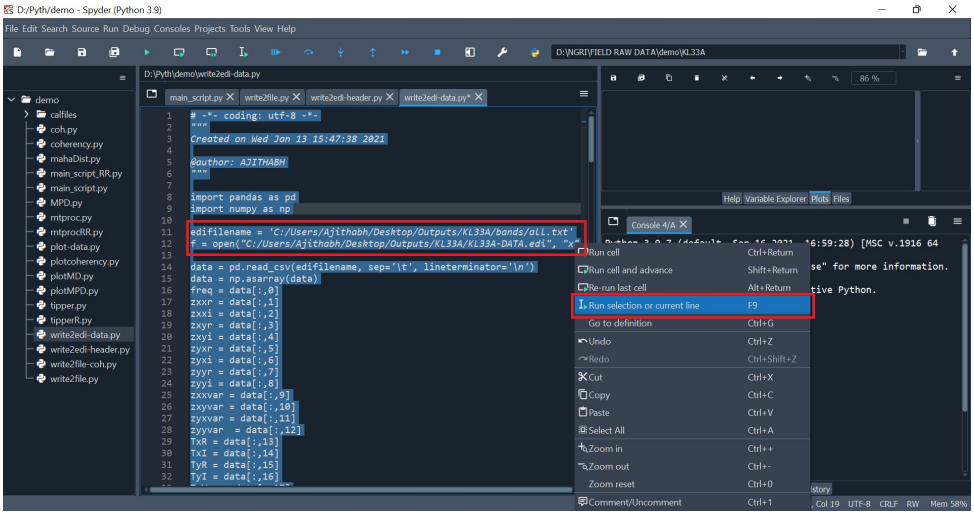

Figure 21: Write data as in EDI

Now, two files are existing in sites folder ('Figure 22). Copy data in the file KL33A-DATA.edi to the end of header information in KL33A-HEADER.edi file (Figure 23). Please correct 'NFREQ' variable with actual number of frequencies in the data (Figure 23). Now, KL33A-HEADER.edi is a standard EDI file ready to use.

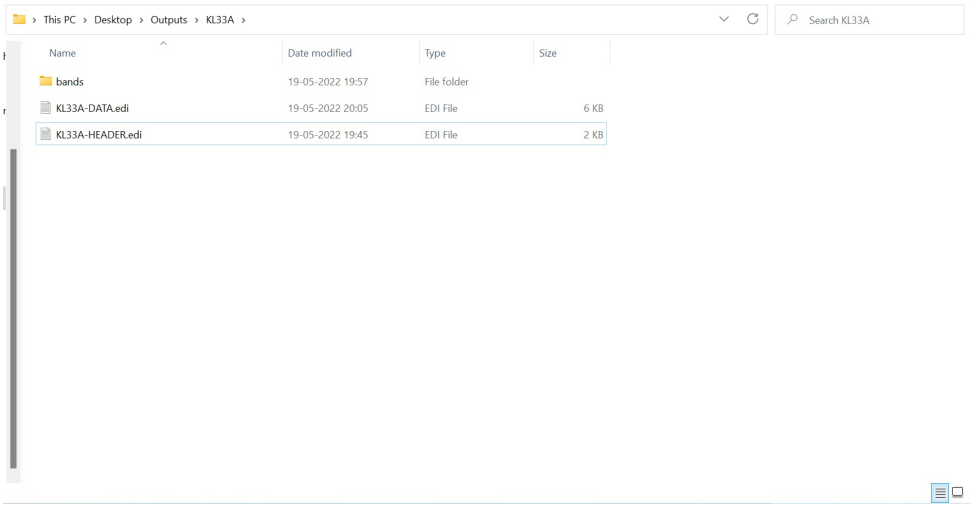

Figure 22: Files ready to create EDI

Figure 23: Edit NFREQ and save EDI

4.3 Data selection tools

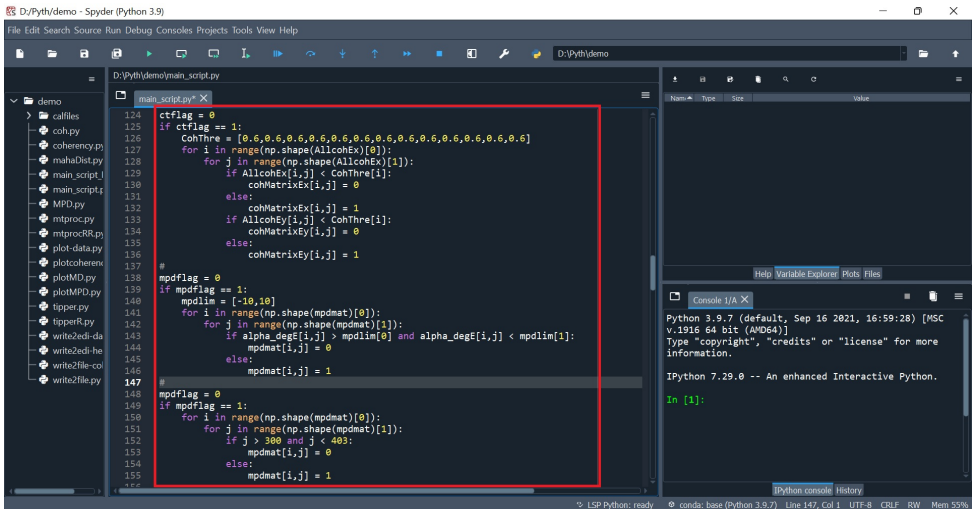

Figure 24: Data selection tools section

In Figure 24, we can see the data selection tools part of package. If ctflag is 1, the coherency threshold will be activated. The coherency threshold value can be provide in 'CohTre' variable for each target frequency. You can see the coherency values for each target frequency using the 'plotcoherency.py' script. Give Z_all = bandavg.get('Zxy_single') to see coherency of xy component and Z_all = bandavg.get('Zyx_single') for yx component. Once you run the script, coherency plots will be displayed for each target frequency as in Figure 26. By analysing the values in the figure, you can set cohrency threshold values.

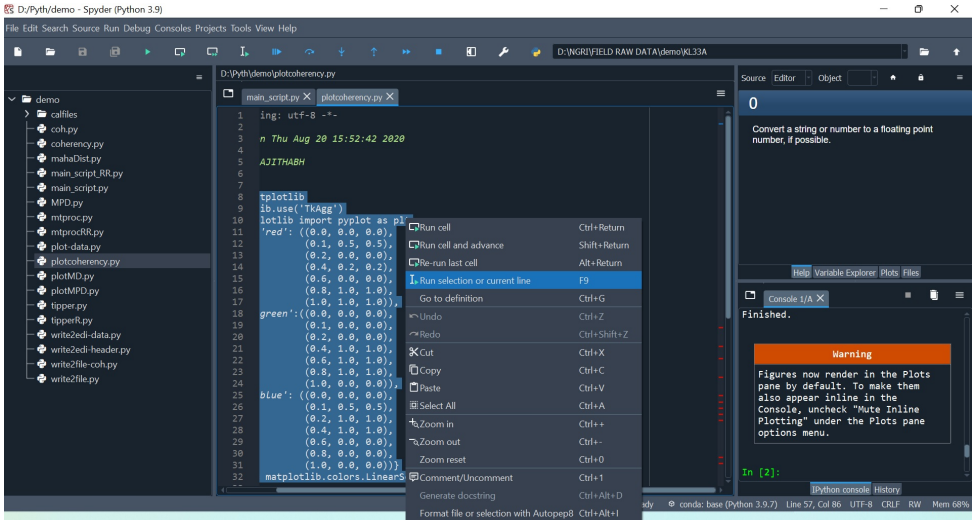

Figure 25: Run codes

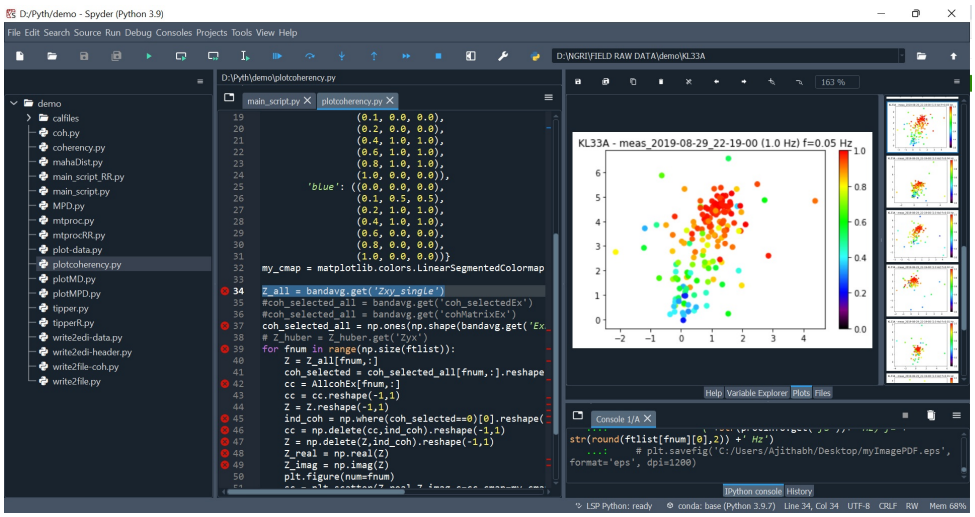

Figure 26: Coherency values

As similar in the case of coherency threshold, run scipts in 'plot-pd.py' to plot polarization directions for each stacks for all target frequencies (Figure 27). The top panel in the plot shows magnetic polarization directions and bottom panel shows electric polarization directions. Analysing the plots, we can identify the polarized segments. Polarization directions can be selected in two ways. We can select a range of polarization direction to be discarded in the case 1. All stacks with polarization direction values [-10,10] will be discarded in this case. In case you need to use magnetic polarization direction, use 'alpha_degH' and for electric polarization direction, use 'alpha_degE' (as in Figure 28).

Next is to select stacks, give a range of stacks to be discarded in this case. In the example (Figure 28), the stacks between 300 and 403 will be discarded from processing.

'mpdflag' should be 1 to perform the tasks.

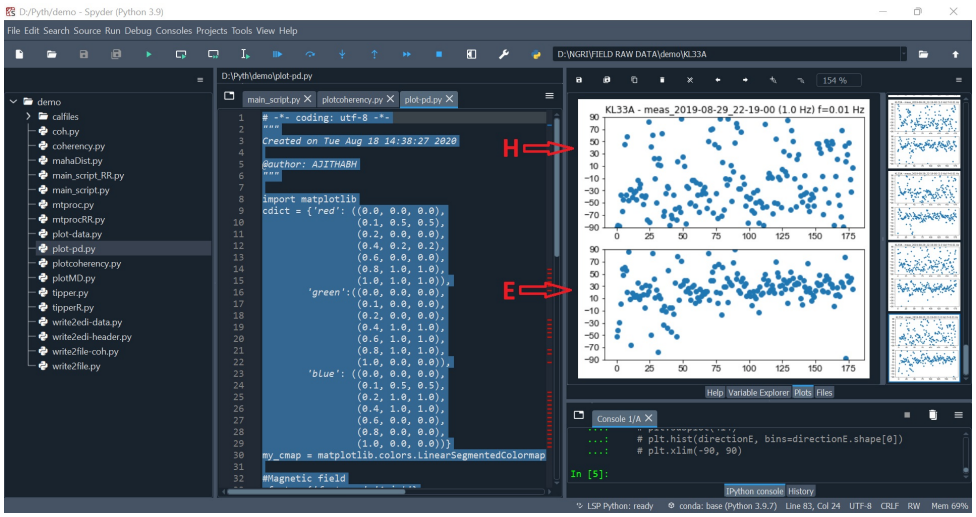

Figure 27: Polarization directions

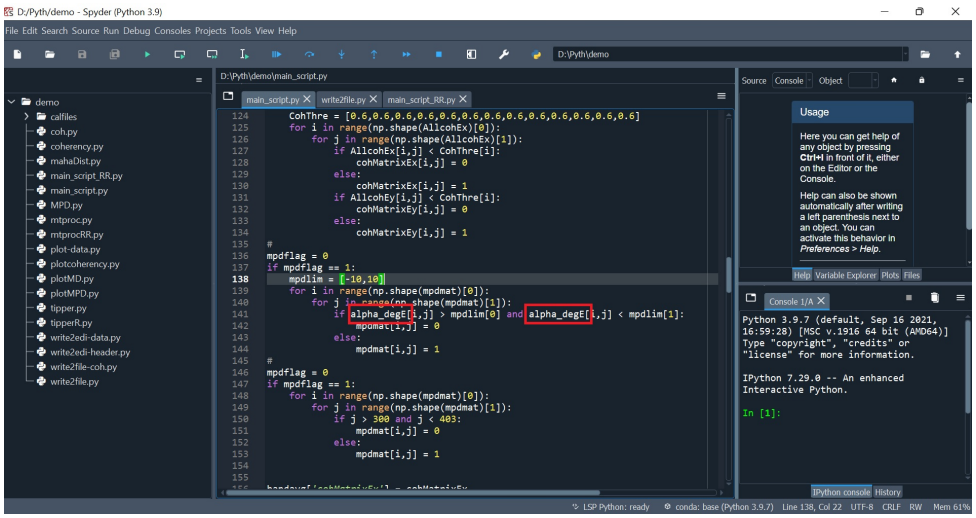

Figure 28: Magnetic or Electric Polarization direction

Once data selection constraints are set, just run a part of code as highlighted (to end of the script) in figure 29.

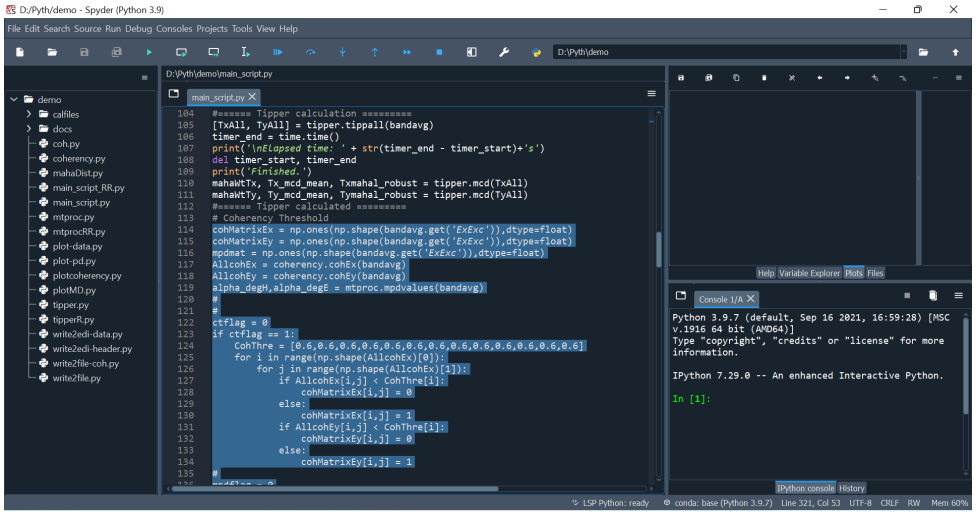

Figure 29: Run this part of script

4.4 Remote reference

The remote reference can be done similar as above example. But run 'main_script_RR.py' file. First you have to enter the site you need to process and enter the measurement number. Then enter the remote site name and measurement number.