

SigMT - User Manual

v. 1

K.S. Ajithabh

May 24, 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	3
2	Getting started	3
3	Overview of the package	5
4	An example	7
4.1	Running the program	7
4.2	Creating EDI file	9
4.3	Data selection tools	14
4.4	Remote reference	17

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.

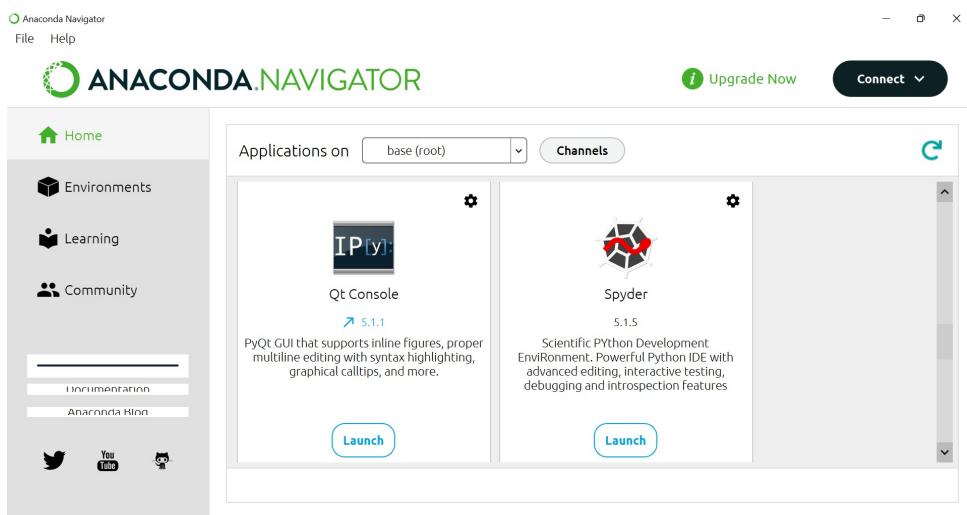


Figure 1: Anaconda Navigator

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. Extract the compressed folder to your local drive. Then open ‘Spyder IDE’, select Projects and click ‘Open Project’ (Figure 2).

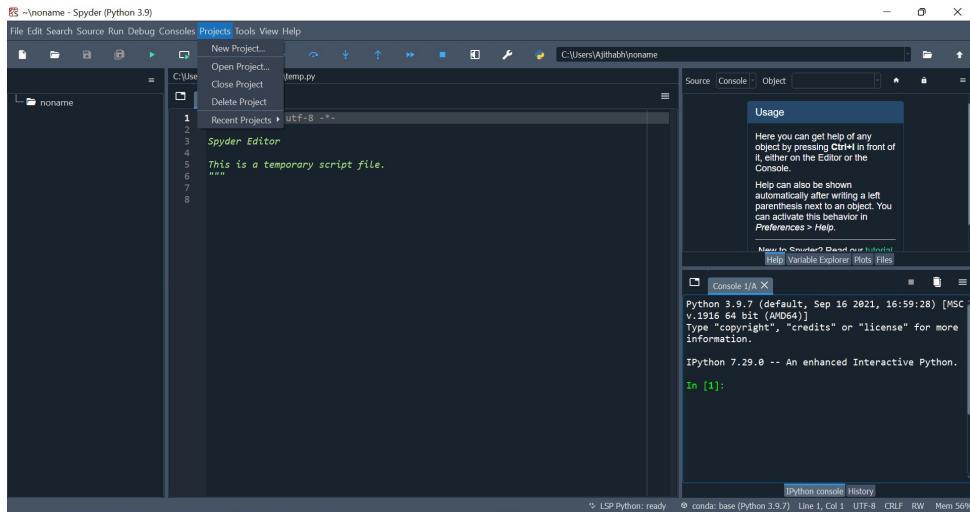


Figure 2: Open Project in Spyder

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 3), 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.

Name	Date modified	Type	Size
250.TXT	11-06-2012 18:06	Text Document	5 KB
251.TXT	11-06-2012 18:06	Text Document	5 KB
252.TXT	11-06-2012 18:06	Text Document	5 KB
253.TXT	11-06-2012 18:06	Text Document	5 KB
254.TXT	11-06-2012 18:06	Text Document	5 KB
255.TXT	11-06-2012 18:07	Text Document	5 KB
300.TXT	27-04-2020 01:01	Text Document	5 KB
301.TXT	27-04-2020 01:02	Text Document	5 KB
302.TXT	27-04-2020 01:02	Text Document	5 KB
303.TXT	27-04-2020 01:01	Text Document	5 KB
311.TXT	27-04-2020 01:02	Text Document	5 KB
314.TXT	27-04-2020 01:02	Text Document	5 KB
315.TXT	27-04-2020 01:02	Text Document	5 KB
316.TXT	27-04-2020 01:02	Text Document	5 KB
318.TXT	27-04-2020 01:03	Text Document	5 KB
561.txt	19-08-2020 15:57	Text Document	5 KB

Figure 3: 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, self-stack.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 4).

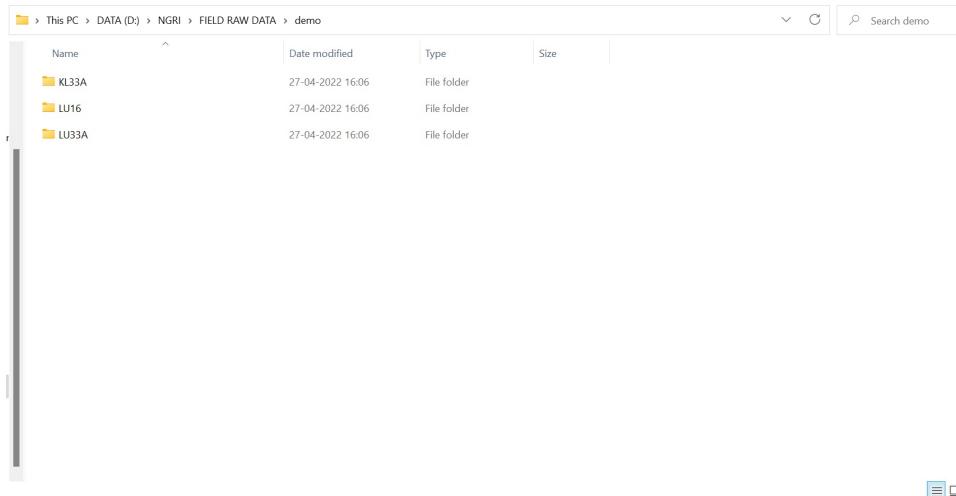


Figure 4: Project folder

The measurements should be in the site folder as in Figure 5. There should not be any other extra folders inside the site folder.

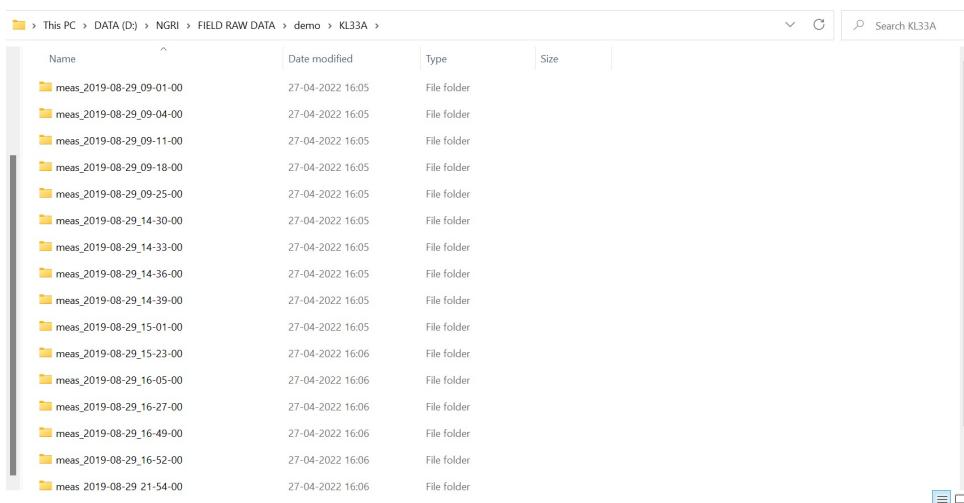


Figure 5: Site folder

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 6.

The screenshot shows the Spyder IDE interface with the following details:

- Title Bar:** D:\Pyth\demo - Spyder (Python 3.9)
- Menu Bar:** File Edit Search Source Run Debug Consoles Projects Tools View Help
- File Explorer:** demo, main_script.py, main_script_RR.py, MPD.py, mptproc.py, mptprocRR.py, plot-data.py, plotcoherency.py, plotMD.py, plotMDP.py, tipper.py, write2edi-data.py, write2file-header.py, write2file-coh.py, write2file.py
- Code Editor:** The file `main_script.py` is open. The code is as follows:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"""
Created on Mon May  4 16:49:35 2020
@author: AJITHABH

import mptproc, coh, coherency, tipper, mahaDist
from scipy import signal
from matplotlib import pyplot as plt
import numpy as np
import os
import time
import math

#
# provide project path where sites are kept
#
project_path = 'D:/NGRI/FIELD RAW DATA/demo/'

#
os.chdir(project_path)
sites = [d for d in os.listdir('.') if os.path.isdir(d)]
# all site names are stored in variable sites
print('sites in the project are: ')
for i in sites:
    print(i)

selectedsites = input("Enter the site: ")
siteindex = sites.index(selectedsite)
measid = mptproc.measid(siteindex)
del siteindex

all_measures = project_path+selectedsite
all_meas = [d for d in os.listdir('.') if os.path.isdir(d)]
for i in range(len(all_meas)):
    if
```

A red oval highlights the line `project_path = 'D:/NGRI/FIELD RAW DATA/demo/'`.

Console: IPython console

Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1929 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information
IPython 7.29.0 -- An Enhanced Interactive Python.

In [1]: |

Figure 6: Give project path (sites folder)

Second edit is required in case you require decimation. The highlighted portions in Figure 7 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.

The screenshot shows the Spyder Python IDE interface. The left pane displays a file tree for a 'demo' folder containing various Python files like 'main_script.py', 'main_script_1.py', 'main_script_2.py', etc. The main code editor window shows 'main_script.py' with several lines of Python code. A red box highlights a section of the code from line 43 to 63. The right pane features a plot area with multiple subplots showing data over time, and below it is a 'Console I/A X' tab showing the output of the Python session.

```
D:\Pyth\demo>python main_script.py
43     timer_start = time.time()
44
45     #----- Time series reading starts -----
46     #
47     #
48     #
49
50     procinfo = {}
51     ts,procinfo['fs'],procinfo['sensor_no'],timeline,procinfo['ChoppStat'],loc = mtproc.t
52     # trend removed
53     # V = mv/km
54     # H = mV/km
55
56     dfflag = 1
57     if dfflag == 1:
58         decimate = [8,4]
59         for d in decimate:
60             ts['tsEx'] = signal.decimate(ts.get('tsEx'), d, nNone, ftype='fir')
61             ts['tsEy'] = signal.decimate(ts.get('tsEy'), d, nNone, ftype='fir')
62             ts['tsHz'] = signal.decimate(ts.get('tsHz'), d, nNone, ftype='fir')
63             ts['tsHs'] = signal.decimate(ts.get('tsHs'), d, nNone, ftype='fir')
64             procinfo['fs'] = procinfo.get('fs')/d
65
66     procinfo['js'] = procinfo.get('js')
67     procinfo['nofs'] = len(ts['tsEx'])
68     print('Nr site: ', selectedsite)
69     print('Measurement directory: ', + all_meas[select_meas])
70     print('fs= ' + str(procinfo.get('fs')) + Hz)
71     print('Sens numbers: ')
72     print(procinfo.get('sensor_no'))
73     print('Length of time series = ' + str(procinfo.get('nofs')))
74     print('.....')
75     procinfo['meas'] = all_meas[selected_meas]
```

Figure 7: Decimation

Third edit is required to control data selection tools. The highlighted portions in Figure 8 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.

```

D:\Pyth\demo - Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Pyth\demo
main_script.py X
124     cohMatrixEx[i,j] = 0
125     if cohMatrixEx[i,j] < CohThre[i]:
126         cohMatrixEx[i,j] = 0
127     else:
128         cohMatrixEx[i,j] = 1
129     for j in range(np.shape(AllcohEx)[1]):
130         if AllcohEx[i,j] < CohThre[i]:
131             cohMatrixEx[i,j] = 0
132         else:
133             cohMatrixEx[i,j] = 1
134     if AllcohEx[i,j] < CohThre[i]:
135         cohMatrixEx[i,j] = 0
136     else:
137         cohMatrixEx[i,j] = 1
138
139     mpdflag = 0
140     if mpdflag == 1:
141         mpdlim = [-10,10]
142         for i in range(np.shape(mpdmat)[0]):
143             for j in range(np.shape(mpdmat)[1]):
144                 if alpha_degE[i,j] > mpdlim[0] and alpha_degE[i,j] < mpdlim[1]:
145                     mpdmat[i,j] = 0
146                 else:
147                     mpdmat[i,j] = 1
148
149     mpdflag = 0
150     if mpdflag == 1:
151         mpdlim = [-10,10]
152         for i in range(np.shape(mpdmat)[0]):
153             for j in range(np.shape(mpdmat)[1]):
154                 if j > 300 and j < 403:
155                     mpdmat[i,j] = 0
156                 else:
157                     mpdmat[i,j] = 1
158
159     mpdflag = 0
160     if mpdflag == 1:
161         mpdlim = [-10,10]
162         for i in range(np.shape(mpdmat)[0]):
163             for j in range(np.shape(mpdmat)[1]):
164                 if j > 300 and j < 403:
165                     mpdmat[i,j] = 0
166                 else:
167                     mpdmat[i,j] = 1
168
169     mpdflag = 0
170     if mpdflag == 1:
171         mpdlim = [-10,10]
172         for i in range(np.shape(mpdmat)[0]):
173             for j in range(np.shape(mpdmat)[1]):
174                 if j > 300 and j < 403:
175                     mpdmat[i,j] = 0
176                 else:
177                     mpdmat[i,j] = 1
178
179     mpdflag = 0
180     if mpdflag == 1:
181         mpdlim = [-10,10]
182         for i in range(np.shape(mpdmat)[0]):
183             for j in range(np.shape(mpdmat)[1]):
184                 if j > 300 and j < 403:
185                     mpdmat[i,j] = 0
186                 else:
187                     mpdmat[i,j] = 1
188
189     project_path = 'D:/NGRI/FIELD RAW DATA/demo/'
190
191     os.chdir(project_path)
192     sites = [d for d in os.listdir('.') if os.path.isdir(d)]
193     # all site names are stored in variable sites
194     print('Sites in the project are: ')
195     print(sites)
196     selectedsite = input("Enter the site: ")
197     selectedsite = 'L3SA'
198     siteindex = sites.index(selectedsite)
199     measid = mtproc.measid(siteindex)
200     del siteindex
201     os.chdir(project_path+selectedsite)
202     all_meas = [d for d in os.listdir('.') if os.path.isdir(d)]

```

Figure 8: 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 9

```

D:\Pyth\demo - Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Pyth\demo
main_script.py X
1  # -*- coding: utf-8 -*-
2  """
3      Created on Mon May  4 16:49:35 2020
4
5      @author: AJITHABH
6      """
7
8      import mtproc, coh, coherency, tipper, mahaDist
9      from scipy import signal
10     from matplotlib import pyplot as plt
11     import numpy as np
12     import os
13     import time
14     import math
15
16     #
17     # provide project path where sites are kept
18     project_path = 'D:/NGRI/FIELD RAW DATA/demo/'
19
20     #
21     os.chdir(project_path)
22     sites = [d for d in os.listdir('.') if os.path.isdir(d)]
23     # all site names are stored in variable sites
24     print('Sites in the project are: ')
25     print(sites)
26     selectedsite = input("Enter the site: ")
27     selectedsite = 'L3SA'
28     siteindex = sites.index(selectedsite)
29     measid = mtproc.measid(siteindex)
30     del siteindex
31     os.chdir(project_path+selectedsite)
32     all_meas = [d for d in os.listdir('.') if os.path.isdir(d)]

```

Figure 9: Give path and run the program

Once we run the program, the names of all sites will be displayed in the console (Figure 10). Then enter the site name to be processed and click ‘Enter’.

```

D:\Pyth\demo - Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\NGRI\FIELD RAW DATA\demo\KL33A
Source Console Object
Usage
Here you can get help of any object
Help Variable Explorer Plots Files
Console 2/A
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.
IPython 7.29.0 -- An enhanced Interactive Python.
In [1]: runfile('D:/Pyth/demo/main_script.py', wdir='D:/Pyth/demo')
Sites in the project are:
['KL33A', 'LU16', 'LU33A']
Enter the site: KL33A

```

Figure 10: 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 11.

```

D:\Pyth\demo - Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\NGRI\FIELD RAW DATA\demo\KL33A
Source Console Object
Usage
Here you can get help of any object
Help Variable Explorer Plots Files
Console 2/A
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.
IPython 7.29.0 -- An enhanced Interactive Python.
In [1]: runfile('D:/Pyth/demo/main_script.py', wdir='D:/Pyth/demo')
Sites in the project are:
['KL33A', 'LU16', 'LU33A']
Enter the site: KL33A
[0, 'meas_2019-08-29_09-01-00']
[1, 'meas_2019-08-29_09-04-00']
[2, 'meas_2019-08-29_09-11-00']
[3, 'meas_2019-08-29_09-18-00']
[4, 'meas_2019-08-29_09-25-00']
[5, 'meas_2019-08-29_10-00-00']
[6, 'meas_2019-08-29_14-33-00']
[7, 'meas_2019-08-29_14-36-00']
[8, 'meas_2019-08-29_14-39-00']
[9, 'meas_2019-08-29_15-00-00']
[10, 'meas_2019-08-29_15-03-00']
[11, 'meas_2019-08-29_16-05-00']
[12, 'meas_2019-08-29_16-27-00']
[13, 'meas_2019-08-29_16-49-00']
[14, 'meas_2019-08-29_16-52-00']
[15, 'meas_2019-08-29_21-54-00']
[16, 'meas_2019-08-29_21-57-00']
[17, 'meas_2019-08-29_22-19-00']
Select measurement: 16

```

Figure 11: 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 12. Then the processing will be started.

Basic details

Progress

```

D:\Pyth\demo - Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\NGRI\FIELD RAW DATA\demo\KL33A
Source Console Object
Usage
Here you can get help of any object
Help Variable Explorer Plots Files
Console 2/A X
[140, 'meas_2019-08-29_21-57-00']
[17, 'meas_2019-08-29_22-19-00']

Select measurement: 16
-----
MT site: KL33A
Measurement directory: meas_2019-08-29_21-57-00
f=4096.0 Hz
Number of measurements:
{'Hx': [300], 'Hy': [302]}
Length of time series = 4915200
-----
Unused variables deleted.

-----
Window Length selected: 16384
Time series overlap: 50%
No. of stacks: 599
-----
Band averaging over target frequencies:
3% | 17/599 [00:09-05:13, 1.86it/s]

```

LSP Python: ready conda: base (Python 3.9.7) Line 36, Col 31 UTF-8 CRLF RW Mem 65%

Figure 12: 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 13).

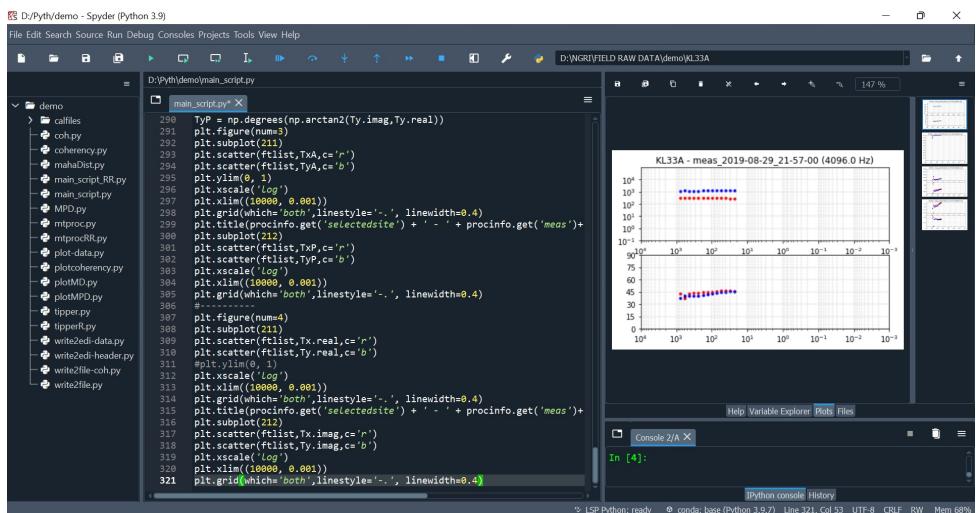


Figure 13: Output figures

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 14). Inside that folder, make a folder with site name. Create a folder named ‘bands’ inside site folder (Figure 14).

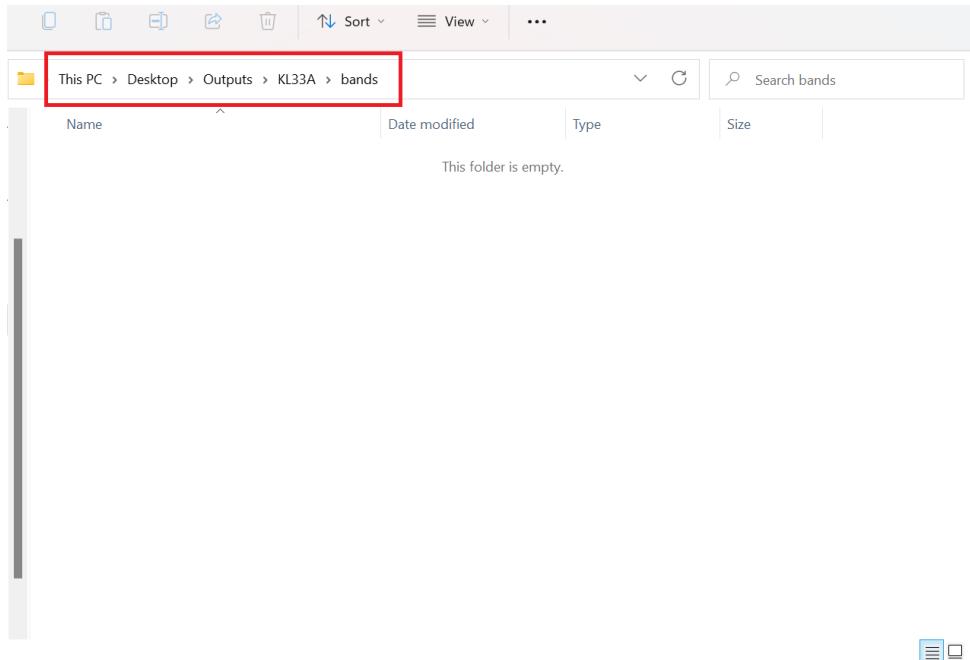


Figure 14: Create a folder to save processed data

Now come to Spyder and open ‘write2file.py’ script. Give the path to bands folder in the variable ‘f’. Give sampling frequency as file name. For example ‘4096Hz.txt’. Select all script lines using ‘Ctrl+A’ and right click. Select ‘Run section or current line’ (Figure 15).

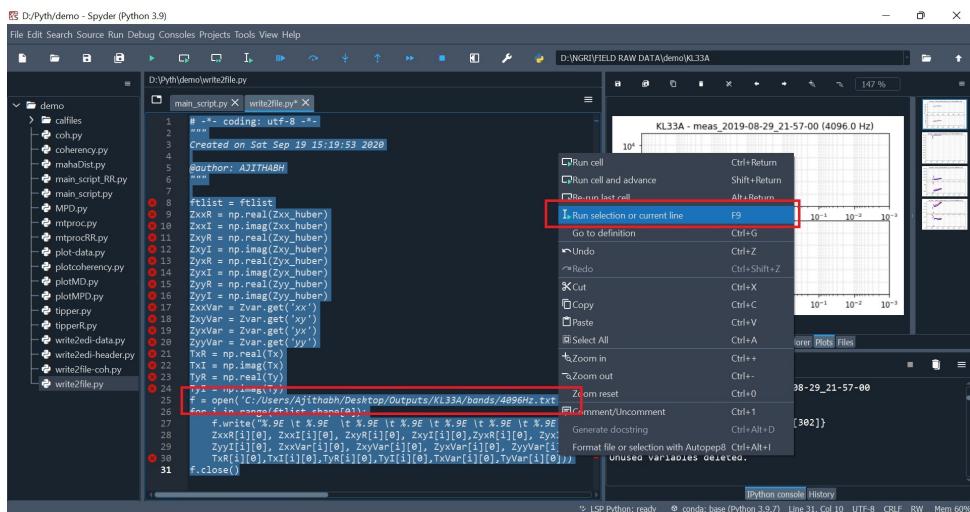


Figure 15: Save data in file

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 16)

```

D:\Pyth\demo\writeZedi-header.py
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Pyth\demo\writeZedi-header.py
main_script.py writefile.py writeZedi-header.py
10 lat_d = np.floor(lat)
11 lat_m2 = (lat - lat_d) * 60
12 lat_m = np.floor(lat_m2)
13 lat_s = lat_m2 - lon_m
14 del_lat_m2
15 lat_s = lat_s * 60
16 lon = loc.get('lon')
17 lon_d = np.floor(lon)
18 lon_m2 = (lon - lon_d) * 60
19 lon_m = np.floor(lon_m2)
20 lon_s = lon_m2 - lon_m
21 del_lon_m2
22 lon_s = lon_s * 60
23
24 from datetime import date
25 today = date.today()
26 d1 = today.strftime('%m/%d/%Y')
27
28 f = open("C:/Users/Ajithab/Desktop/Outputs/KL33A/KL33A-HEADER.edt",
29         "w")
30 f.write("FILEID=")
31 f.write(procinfo.get('selectedsite'))
32 f.write("\n ACQBY=" + "CSIR - NGR, INDIA")
33 f.write("\n FILEBY=" + "AJITHABH K.S.")
34 f.write("\n ACQDATE=" + pd.to_datetime(timeline[0]-719529,unit='D'))
35 f.write("\n ACQTIME=" + d1)
36 f.write("\n PROJECT=" + procinfo.get('selectedsite'))
37 f.write("\n LAT=" + str(round(lat_d)) + ":" + str(round(lat_m)) + ":"
38 f.write(str(lat_s)))
39 f.write("\n LONG=" + str(round(lon_d)) + ":" + str(round(lon_m)) + ":"+
40 f.write(str(lon_s)))
41 f.write("\n ELEV=" + str(loc.get('elev')))
42 f.write("\n STVER=" + "SEG 1.0")
43 f.write("\n PROGVERS= SigNet 0.4.1")
44 f.close()

```

Figure 16: Save header information

After executing processing for a measurement, always close ‘Console’ (Figure 17).

```

D:\Pyth\demo\writeZedi-header.py
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Pyth\demo\writeZedi-header.py
main_script.py writefile.py writeZedi-header.py
10 lat_d = np.floor(lat)
11 lat_m2 = (lat - lat_d) * 60
12 lat_m = np.floor(lat_m2)
13 lat_s = lat_m2 - lon_m
14 del_lat_m2
15 lat_s = lat_s * 60
16 lon = loc.get('lon')
17 lon_d = np.floor(lon)
18 lon_m2 = (lon - lon_d) * 60
19 lon_m = np.floor(lon_m2)
20 lon_s = lon_m2 - lon_m
21 del_lon_m2
22 lon_s = lon_s * 60
23
24 from datetime import date
25 today = date.today()
26 d1 = today.strftime('%m/%d/%Y')
27
28 f = open("C:/Users/Ajithab/Desktop/Outputs/KL33A/KL33A-HEADER.edt",
29         "w")
30 f.write("FILEID=")
31 f.write(procinfo.get('selectedsite'))
32 f.write("\n ACQBY=" + "CSIR - NGR, INDIA")
33 f.write("\n FILEBY=" + "AJITHABH K.S.")
34 f.write("\n ACQDATE=" + pd.to_datetime(timeline[0]-719529,unit='D'))
35 f.write("\n ACQTIME=" + d1)
36 f.write("\n PROJECT=" + procinfo.get('selectedsite'))
37 f.write("\n LAT=" + str(round(lat_d)) + ":" + str(round(lat_m)) + ":"
38 f.write(str(lat_s)))
39 f.write("\n LONG=" + str(round(lon_d)) + ":" + str(round(lon_m)) + ":"+
40 f.write(str(lon_s)))
41 f.write("\n ELEV=" + str(loc.get('elev')))
42 f.write("\n STVER=" + "SEG 1.0")
43 f.write("\n PROGVERS= SigNet 0.4.1")
44 f.close()

```

Figure 17: 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 18). Create a text file named ‘all.txt’.

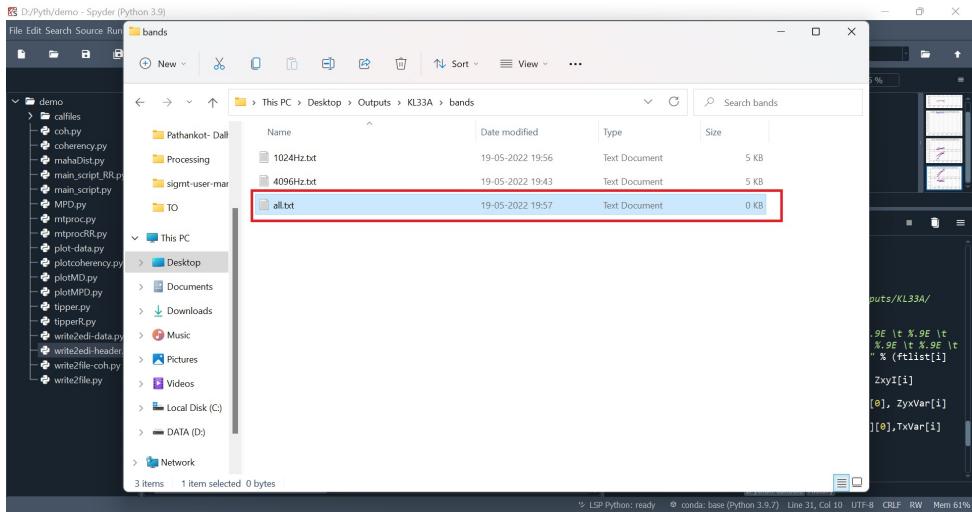


Figure 18: Create a file all.txt

Copy the data from 4096Hz.txt file and 1024Hz.txt file to all.txt as in Figure 19. Always keep higher frequency data in the top.

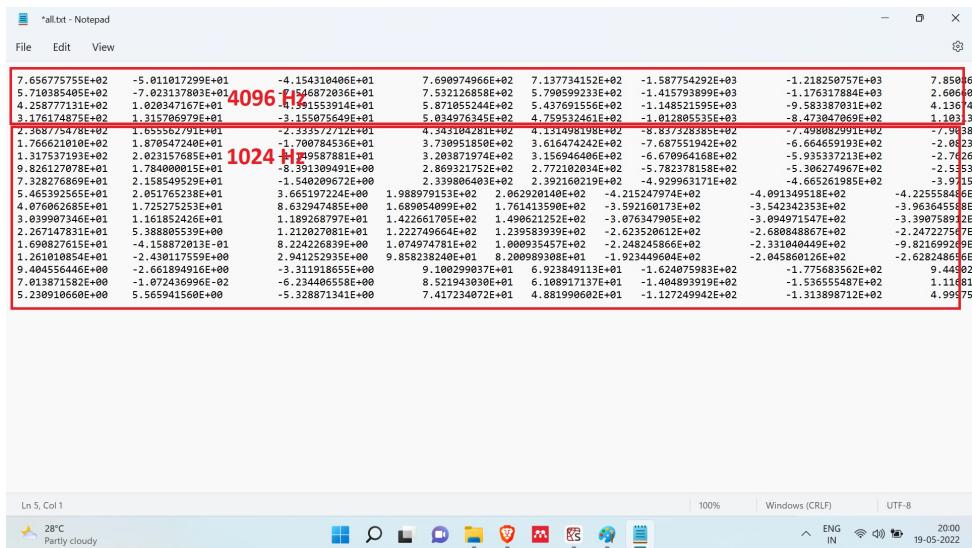


Figure 19: Copy data to all.txt file

Now, again go to Spyder and open ‘write2edi-data.py’ file. Give file path to ‘all.txt’ and path of sites folder in variables ‘edifilename’ and ‘f’ (Figure 20).

The screenshot shows the Spyder IDE interface with several windows open. On the left, there's a file browser showing a directory structure under 'demo'. In the center, a code editor window displays a Python script named 'write2edi-data.py'. The script contains code for reading a CSV file, processing it with pandas and numpy, and then writing it to an EDI file. A red box highlights the line 'f = open("C:/Users/Ajithabh/Desktop/Outputs/KL33A/bands/edi", "x")'. To the right of the code editor is a 'Console 4/A' window showing the command line interface with various options like 'Run cell', 'Run cell and advance', etc. Another red box highlights the 'Run selection or current line' option under the 'Cell' menu.

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Jan 13 15:47:38 2021
4
5 @author: AJITHABH
6 """
7
8 import pandas as pd
9 import numpy as np
10
11 edifilename = 'C:/Users/Ajithabh/Desktop/Outputs/KL33A/bands/edi.txt'
12 f = open("C:/Users/Ajithabh/Desktop/Outputs/KL33A/bands/edi", "x")
13
14 data = pd.read_csv(edifilename, sep='\t', lineterminator='\n')
15 data = np.asarray(data)
16 zxxi = data[:,0]
17 zxxi = data[:,1]
18 zxxi = data[:,2]
19 zxyr = data[:,3]
20 zxyr = data[:,4]
21 zyxv = data[:,5]
22 zyxv = data[:,6]
23 zyyr = data[:,7]
24 zyyr = data[:,8]
25 zxvar = data[:,9]
26 zxvar = data[:,10]
27 zxvar = data[:,11]
28 zxvar = data[:,12]
29 Tyr = data[:,13]
30 Tx1 = data[:,14]
31 Tyr = data[:,15]
32 TyT = data[:,16]

```

Figure 20: Write data as in EDI

Now, two files are existing in sites folder ('Figure 21). Copy data in the file KL33A- DATA.edi to the end of header information in KL33A-HEADER.edi file (Figure 22). Please correct 'NFREQ' variable with actual number of frequencies in the data (Figure 22). Now, KL33A-HEADER.edi is a standard EDI file ready to use.

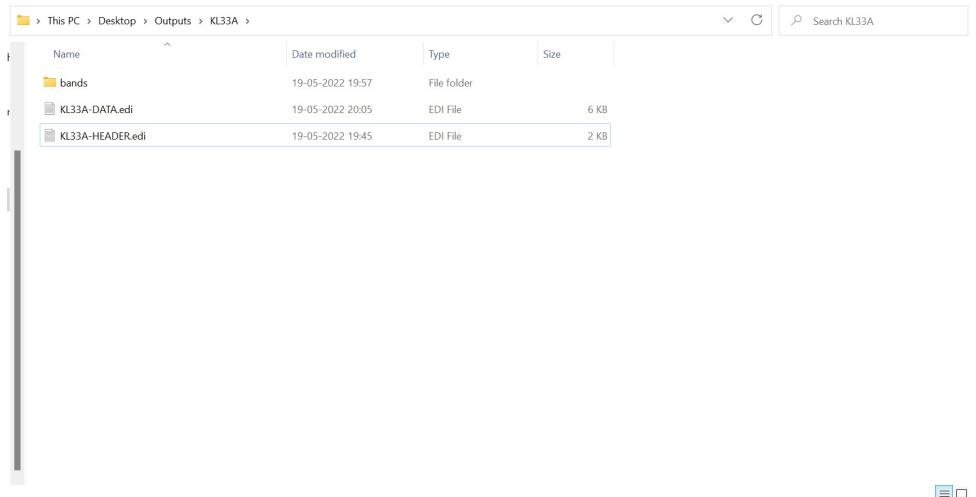


Figure 21: Files ready to create EDI

```
*KL33A-HEADER.edt - Notepad
File Edit View

>HEMAS ID=4_001 CHTYPE=HY X=0.00000000E+00
Y=0.00000000E+00 Z=0.00000000E+00
ACQCHAN="ADU07 /MFS06 0/" GAIN=1 MEASDATE=08/29/2019
AZH=0.00000000E+01 DIP=0.00000000E+00
SENSOR=MFS06 /0

>HEMAS ID=4_001 CHTYPE=HZ X=0.00000000E+00
Y=0.00000000E+00 Z=0.00000000E+00
ACQCHAN="ADU07 /MFS06 0/" GAIN=1 MEASDATE=08/29/2019
AZH=0.00000000E+00 DIP=0.00000000E+00
SENSOR=MFS06 /0

=>MTSECT
SEGMENT=KL33A
MTSEG=13
HX=3_001
HY=4_001
HZ=5_001
EX=1_001
EV=2_001

>FREQ //17
5.710385405E+02 4.258777131E+02 3.176174875E+02 2.368775478E+02 1.766621018E+02
1.797532103E+02 9.02127072E+01 7.328276869E+01 5.465392565E+01 4.076062685E+01
3.03927346E+01 2.267147831E+01 1.090827615E+01 1.261010854E+01 9.404556446E+00
0.138715182E+00 5.230818650E+00

>XXX //17
-7.203137083E+01 1.020347167E+01 1.315706979E+01 1.655562791E+01 1.870547240E+01
0.23157685E+01 1.784000015E+01 2.158549529E+01 2.851765238E+01 1.725275253E+01
1.161852426E+01 5.388805539E+00 -1.458872013E-01 -2.430117559E+00 -2.661894916E+00
-1.072436996E-02 5.565941560E+00

>XXX //17
-7.546872036E+01 -4.391553914E+01 -3.155075649E+01 -2.333572712E+01 -1.700784536E+01

Ln 58, Col 11 100% Windows (CRLF) UTF-8
```

Figure 22: Edit NFREQ and save EDI

4.3 Data selection tools

Figure 23: Data selection tools section

In Figure 23, 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 25. By analysing the values in the figure, you can set cohrency threshold values.

The screenshot shows the Spyder Python IDE interface. The left sidebar displays a project structure under 'demo' with files like 'coh.py', 'coherency.py', 'mahaDist.py', 'main_script.RP.py', 'main_script.py', 'MPD.py', 'nptrope.py', 'nptropeRPy', 'plot-data.py', 'plotcoherency.py', 'plotMD.py', 'plotMDP.py', 'tipper.py', 'tipperF.py', 'write2edi-data.py', 'write2edi-header.py', 'write2file-coh.py', and 'write2file.py'. The main editor window shows a Python script named 'main_script.py' with code related to 'plotcoherency.py'. A context menu is open over the line 'blue': ((0.0, 0.0, 0.0),'. The menu items include 'Run cell' (Ctrl+Return), 'Run cell and advance' (Shift+Return), 'Run last cell' (Alt+Return), 'Run selection or current line' (F9), 'Go to definition' (Ctrl+G), 'Undo' (Ctrl+Z), 'Redo' (Ctrl+Shift+Z), 'Cut' (Ctrl+X), 'Copy' (Ctrl+C), 'Paste' (Ctrl+V), 'Select All' (Ctrl+A), 'Zoom in' (Ctrl++), 'Zoom out' (Ctrl+-), 'Zoom reset' (Ctrl+0), 'Comment/Uncomment' (Ctrl+I), 'Generate docstring' (Ctrl+Alt+D), and 'Format file or selection with Autopaste' (Ctrl+Alt+A). The status bar at the bottom indicates the file is 'D:\INGR\FIELD RAW DATA\demo\kL33A'. On the right, there are panes for 'Source', 'Editor', 'Object', and 'Plots'. A 'Warning' box in the Plots pane states: 'Figures now render in the Plots pane by default. To see them also appear inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane options menu.'

Figure 24: Run codes

D:\Pyth\demo\plotcoherency

```
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\INGRIFIELD RAW DATA\demo\KL33A
19         (0.1, 0.0, 0.0),
20         (0.2, 0.0, 0.0),
21         (0.4, 0.0, 0.0),
22         (0.6, 1.0, 1.0),
23         (0.8, 1.0, 1.0),
24         (1.0, 0.0, 0.0),
25         ('blue'): ((0.0, 0.0, 0.0),
26             (0.2, 0.0, 0.0),
27             (0.4, 0.0, 0.0),
28             (0.6, 0.0, 0.0),
29             (0.8, 0.0, 0.0),
30             (1.0, 0.0, 0.0)),
31     my_cmap = matplotlib.colors.LinearSegmentedColormap
32
33     # 34
34     Z_all = bandavg.get('Zxy_single')
35     coh_selected_all = bandavg.get('coh_selectedEx')
36     coh_selected_ex = bandavg.get('cohMatrixEx')
37     coh_selected_ex = np.reshape(np.shape(bandavg.get('Ex'))[0], 7, 7)
38     Z_huber = Z_huber.get('7x7')
39     for fnum in range(np.size(ftlist)):
40         Z = Z_all[fnum,:]
41         coh_selected_ex = coh_selected_ex[fnum,:,:].reshape(7,7)
42         cc = coh_selected_ex[0,0]
43         cc = cc.reshape(-1,1)
44         Z = Z.reshape(-1,1)
45         ind_coh = np.where((coh_selected==0)[0].reshape(-1,1))
46         c = np.delete(cc, ind_coh).reshape(-1,1)
47         Z = np.delete(Z, ind_coh).reshape(-1,1)
48         Z[0] = 0
49         Z_imag = np.imag(Z)
50         plt.imshow(numz)
51         plt.colorbar()
52         plt.title('KL33A - meas_2019-08-29_22-19-00 (1.0 Hz) f=0.05 Hz')
53         plt.savefig('C:/Users/Ajithabh/Desktop/myImagePDF.eps',
54             format='eps', dpi=1200)
```

Figure 25: Coherency values

As similar in the case of coherency threshold, run scripts in ‘plot-pd.py’ to plot polarization directions for each stacks for all target frequencies (Figure 26). 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 27).

Next is to select stacks, give a range of stacks to be discarded in this case. In the example (Figure 27), the stacks between 300 and 403 will be discarded from processing.

‘mpdflag’ should be 1 to perform the tasks.

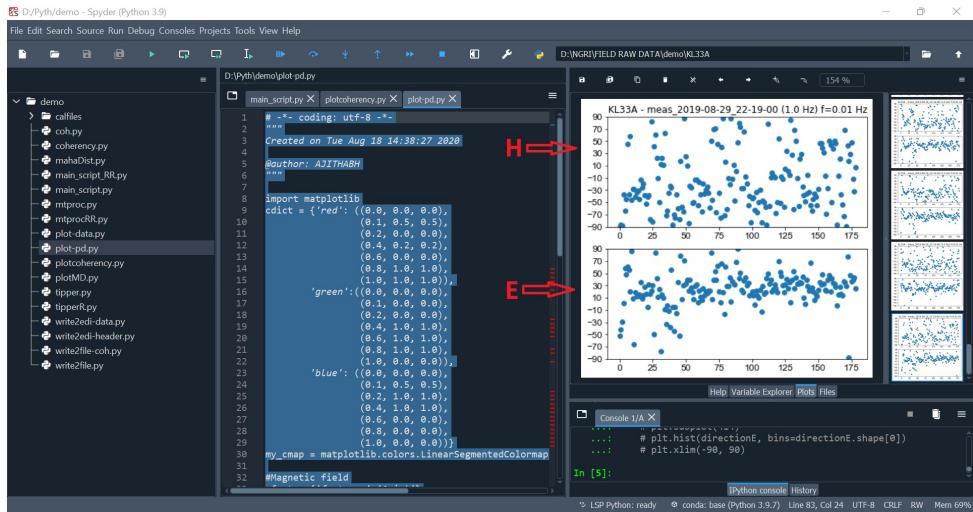


Figure 26: Polarization directions

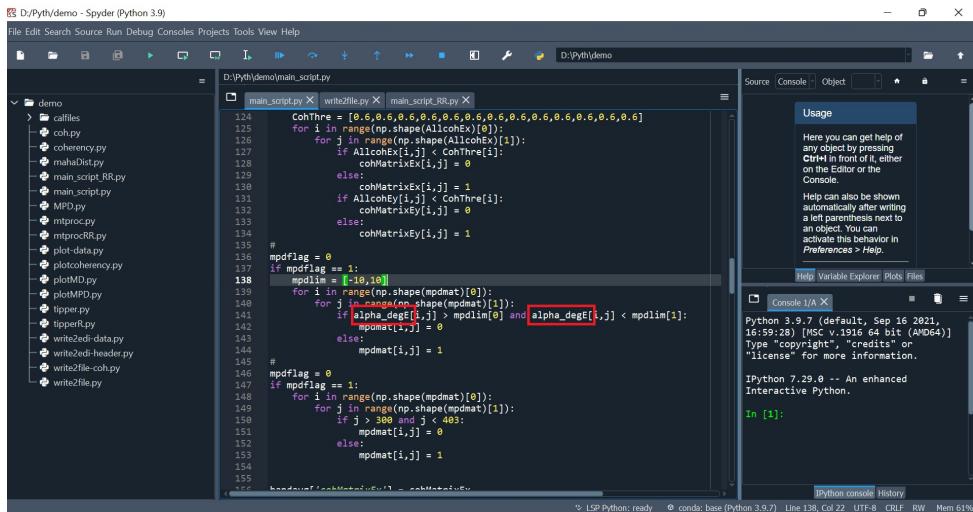


Figure 27: Magnetic or Electric Polarization direction

Once data selection constraints are set, just run a part of code as highlighted (to end of the script) in figure 28.

```

D:\Pyth\demo - Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Pyth\demo\main_script.py
D:\Pyth\demo\main_script.py
104     ##### Tipper calculation #####
105     [TxAll, TyAll] = tipper.tippall(bandavg)
106     timer_end = time.time()
107     print('Time taken: ' + str(timer_end - timer_start) + 's')
108     del timer_start, timer_end
109     print('Finished.')
110     mahaTx, Tx_mcd_mean, Tymahal_robust = tipper.mcd(TxAll)
111     mahaTy, Ty_mcd_mean, Tymahal_robust = tipper.mcd(TyAll)
112     print('Tipper calculated')
113     # Coherence Threshold
114     cohMatrixEx = np.ones(np.shape(bandavg.get('ExExc')), dtype=float)
115     cohMatrixEy = np.ones(np.shape(bandavg.get('ExExc')), dtype=float)
116     mpdmat = np.ones(np.shape(bandavg.get('ExExc')), dtype=float)
117     AllcohEx = coherence.coheX(bandavg)
118     AllcohEy = coherence.coheY(bandavg)
119     alpha_deg, alpha_degE = mpdproc.mpdvalues(bandavg)
120     #
121     #
122     ctflag = 0
123     if ctflag == 1:
124         Cohere = [0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6]
125         for i in range(np.shape(AllcohEx)[0]):
126             for j in range(np.shape(AllcohEx)[1]):
127                 if AllcohEx[i,j] < Cohere[i]:
128                     cohMatrixEx[i,j] = 0
129                 else:
130                     cohMatrixEx[i,j] = 1
131                 if AllcohEx[i,j] < Cohere[i]:
132                     cohMatrixEy[i,j] = 0
133                 else:
134                     cohMatrixEy[i,j] = 1
135

```

Figure 28: 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.