

SigMT - User Manual

v. 1

K.S. Ajithabh

November 7, 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	Setting of parameters	8
5	An example	8
5.1	Running the program	8
5.2	Creating EDI file	10
5.3	Data selection tools	15
5.4	Remote reference	18

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.

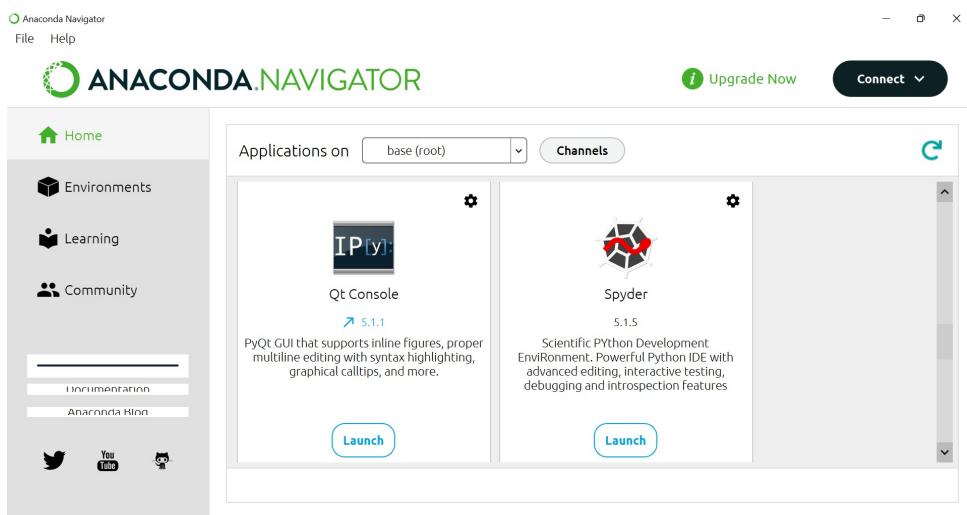


Figure 1: Anaconda Navigator

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).

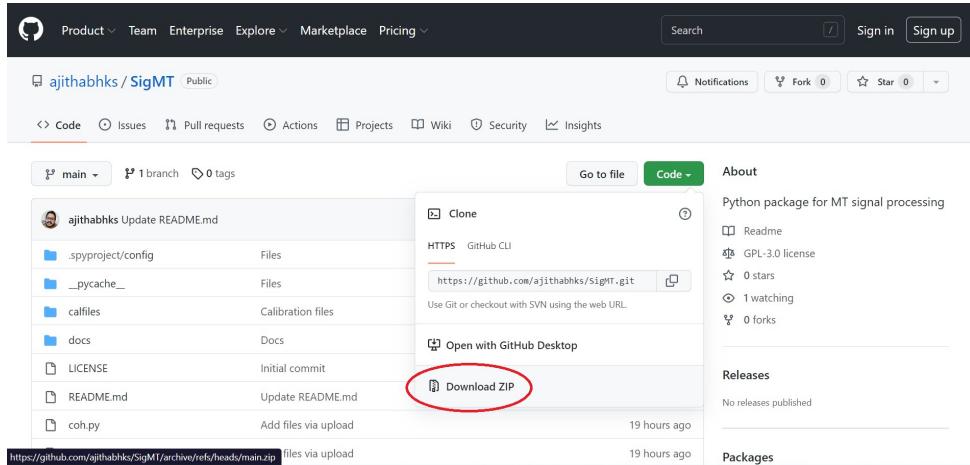


Figure 2: Download the package

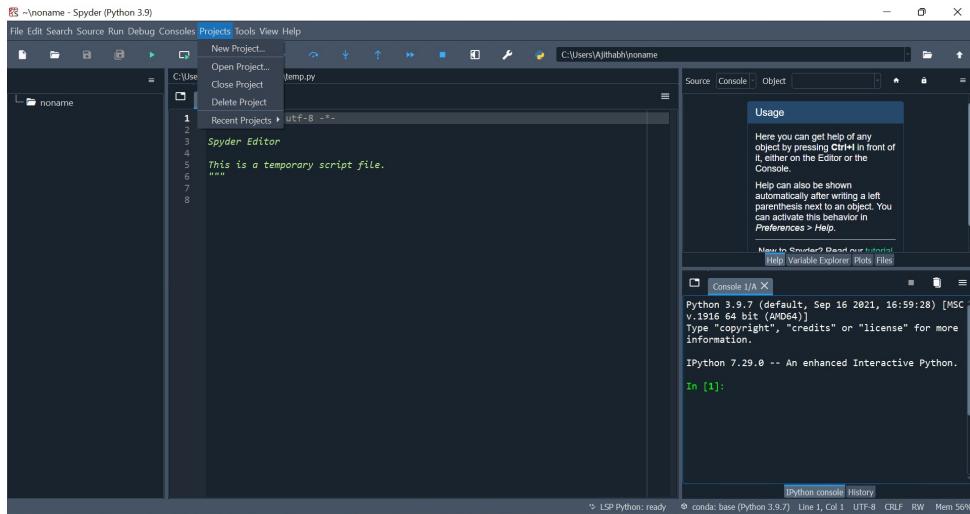


Figure 3: Open Project in Spyder

Navigate to extracted SigMT project folder and select it. Now you are in SigMT package.

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.

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 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, var.py, mahaDist.py, data_sel.py, tipper.py are written to perform different tasks in package and need not to be edited.

I suggest editing of main_script.py, main_script_RR.py and config.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).

This PC > DATA (D:) > NGRI > FIELD RAW DATA > demo			
Name	Date modified	Type	Size
KL33A	27-04-2022 16:06	File folder	
LU16	27-04-2022 16:06	File folder	
LU33A	27-04-2022 16:06	File folder	

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.

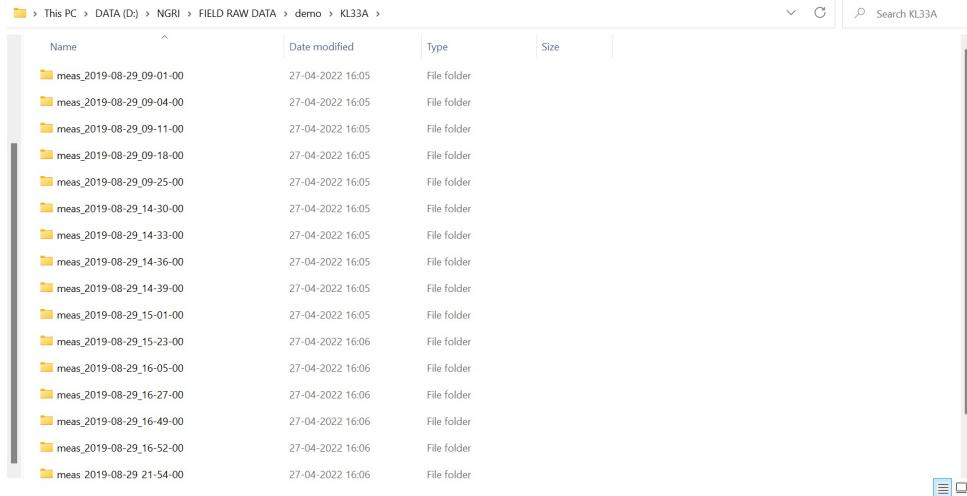


Figure 6: Site folder

There are four 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 of all provide the path of folder in the ‘cal_path’ variable where calibration files are kept. For example: cal_path = ‘D:/Pyth/SigMT/calffiles/’

Second 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.

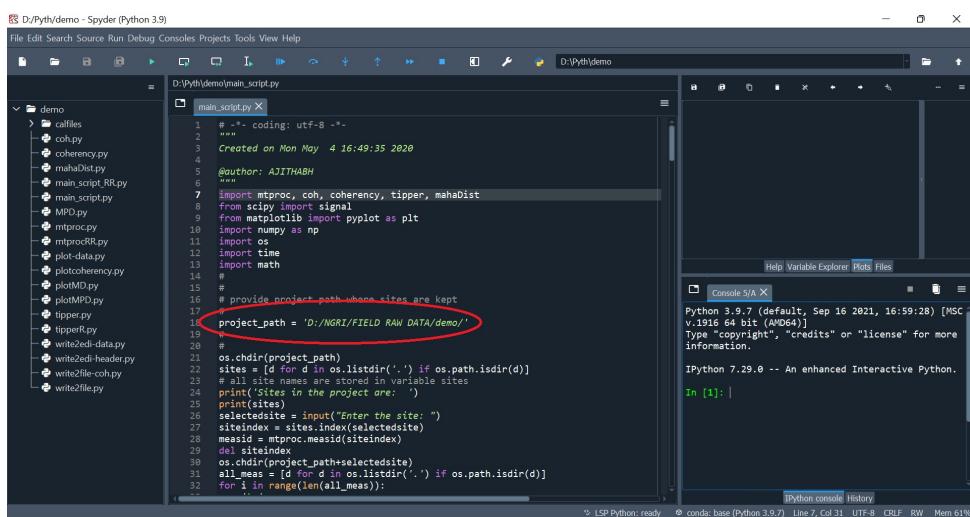


Figure 7: Give project path (sites folder)

Third 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.

```

D:\Pyth\SigMT-Distribution - Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Pyth\SigMT-Distribution
main_script.py

43 # #
44 # ===== Selection of sites and setting a path =====
45 sites, selectedsite, mesid, all_meas, select_meas, proc_path = mtproc.makeproxpath(project_pat
46 #===== Site is selected and path is created =====
47 timer_start = time.time()
48 #
49 # ===== Time series reading starts =====
50 procinfo = {}
51 procinfo['fs'] = procinfo['sensor_no'], timeline, procinfo['ChoppStat'], loc = mtproc.ts(proc
52 #===== Decimation section =====
53 # Keep dfFlag = 0 if decimation is not required
54 dfFlag = 0
55 if dfFlag == 1:
56     decimate = [0, 8, 4]
57     for d in decimate:
58         ts['tsEx'] = signal.decimate(ts.get('tsEx'), d, n=None, ftype='fir')
59         ts['tsEy'] = signal.decimate(ts.get('tsEy'), d, n=None, ftype='fir')
60         ts['tsHx'] = signal.decimate(ts.get('tsHx'), d, n=None, ftype='fir')
61         ts['tsHy'] = signal.decimate(ts.get('tsHy'), d, n=None, ftype='fir')
62         ts['tsHz'] = signal.decimate(ts.get('tsHz'), d, n=None, ftype='fir')
63         procinfo['fs'] = procinfo.get('fs')/d
64 #===== Decimation section =====
65 #
66 # Some calculations and printing some information
67 # Some calculations to edit
68 procinfo['CoHTh'] = len(ts['tsEx'])
69 procinfo['Notch'] = 0 # Notch flag 1 - On, 0 - Off
70 print("-----")
71 print(MT site: + selectedsite)
72 print(Measurement directory: + all_meas[select_meas])
73 print(fs + str(procinfo.get('fs')) + Hz)
74 print("-----")

```

Figure 8: Decimation

Fourth 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 5.3.

```

D:\Pyth\SigMT-Distribution - Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Pyth\SigMT-Distribution
main_script.py

121 alpha_deg0, alpha_deg1 = data_sel.pdvalues(bandevg)
122 #
123 # ===== Coherency threshold based selection =====
124 if ctflag == 1:
125     CoHThre = [0.9, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1]
126     for i in range(np.shape(AllocEx)[0]):
127         for j in range(np.shape(AllocEx[i])):
128             if AllocEx[i][j] < CoHThre[i]:
129                 coMatriEx[i][j] = 0
130             else:
131                 coMatriEx[i][j] = 1
132             if coMatriEx[i][j] == 1:
133                 coMatriEx[i][j] = 0
134             else:
135                 coMatriEx[i][j] = 1
136 #
137 #===== Polarization direction =====
138 pdflag = 0 # Give '1' to perform polarization direction based selection
139 if pdflag == 1:
140     pdlim = [40, 60]
141     alpha_deg0, alpha_deg1 = data_sel.pdvalues(bandevg)
142     for i in range(np.shape(pdmat)[0]):
143         for j in range(np.shape(pdmat)[1]):
144             if alpha[i][j] > pdlim[0] and alpha[i][j] < pdlim[1]:
145                 pdmat[i][j] = 0
146             else:
147                 pdmat[i][j] = 1
148 #
149 pdflag = 0 # Give '1' to perform polarization direction based selection
150 if pdflag == 1:
151     timwindow_limits = [0, 5]
152     for i in range(np.shape(pdmat)[0]):
153         for j in range(np.shape(pdmat)[1]):
154             if pdmat[i][j] > timwindow_limits[0] and j < timwindow_limits[1]:
155                 pdmat[i][j] = 0
156             else:
157                 pdmat[i][j] = 1
158 #

```

Figure 9: Data selection tools section

4 Setting of parameters

The screenshot shows the Spyder IDE interface with the file 'D:\Pyth\SigMT-Distribution\config.py' open. The code defines a configuration function that sets parameters like FFT length, parzen radius, and Mahalanobis distance thresholds. The configuration file is timestamped as 'Created on Mon Jul 25 11:46:56 2022'. The Spyder interface includes a sidebar with project files, a central code editor, and a right-hand panel for help and variable exploration.

```
config.py
config = {}
config['FFT_Length'] = 0
# Give '0' (default value) to automatically select an FFT length according to Borah & Patro, 2015
# Otherwise try 256, 512, 1024, 2048, 4096, 8192, 16384
config['parzen_radius'] = 1.5
# Give '0' (default value) to select parzen radius according to Borah & Patro, 2015
config['MD_threshold_impedance'] = 1.5
# Mahalanobis distance threshold value for impedance calculations
config['MD_threshold_tipper'] = 1.0
# Mahalanobis distance threshold value for tipper calculations
return config
```

Figure 10: Configuration file

The user can define parameters such as FFT length, parzen radius, mahalanobis distance for impedance calculation, mahalanobis distance for tipper calculation. The configuration file is shown in the Figure 10. Keep FFT_Length and parzen_radius as 0 to choose default values. Or else, provide values as per the need.

5 An example

5.1 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 11

The screenshot shows the Spyder IDE interface with the file 'D:\Pyth\demo\main_script.py' open. The code defines a script that processes data from a specified project path. A red box highlights the 'Run' button in the toolbar. The right-hand panel shows the Python console output.

```
main_script.py
project_path = 'D:/NGRI/FIELD RAW DATA/demo/'
```

Figure 11: Give path and run the program

Once we run the program, the names of all sites will be displayed in the console (Figure 12). 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:\Pyth\demo\main_script.py
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 import mproc, coh, coherence, tipper, mahaDist
8 from scipy import signal
9 from matplotlib import pyplot as plt
10 import numpy as np
11 import os
12 import time
13 import math
14 #
15 #
16 # provide project path where sites are kept
17 #
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 selectedsite = input("Enter the site: ")
26 siteindex = sites.index(selectedsite)
27 measid = mproc.measid(siteindex)
28 del siteindex
29 os.chdir(project_path+selectedsite)
30 all_meas = [d for d in os.listdir('.') if os.path.isdir(d)]
31 all_meas = [d for d in all_meas]
32 for i in range(len(all_meas)):
    for i in range(len(all_meas)):

```

Figure 12: 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 13.

```

D:\Pyth\demo - Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Pyth\demo\main_script.py
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 import mproc, coh, coherence, tipper, mahaDist
8 from scipy import signal
9 from matplotlib import pyplot as plt
10 import numpy as np
11 import os
12 import time
13 import math
14 #
15 #
16 # provide project path where sites are kept
17 #
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 selectedsite = input("Enter the site: ")
26 siteindex = sites.index(selectedsite)
27 measid = mproc.measid(siteindex)
28 del siteindex
29 os.chdir(project_path+selectedsite)
30 all_meas = [d for d in os.listdir('.') if os.path.isdir(d)]
31 all_meas = [d for d in all_meas]
32 for i in range(len(all_meas)):
    for i in range(len(all_meas)):

```

Figure 13: 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 14). Then the processing will be started.

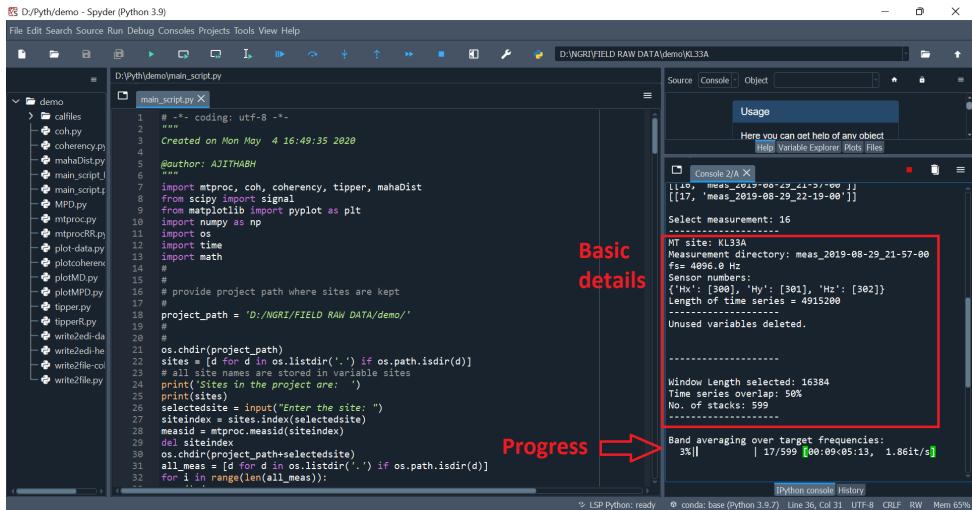


Figure 14: Processing..

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 15).

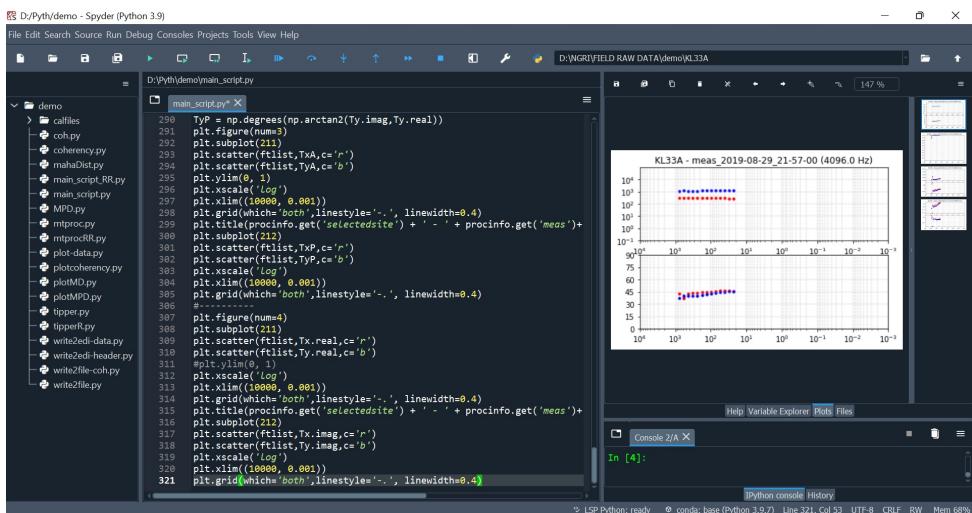


Figure 15: Output figures

5.2 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 16). Inside that folder, make a folder with site name. Create a folder named ‘bands’ inside site folder (Figure 16).

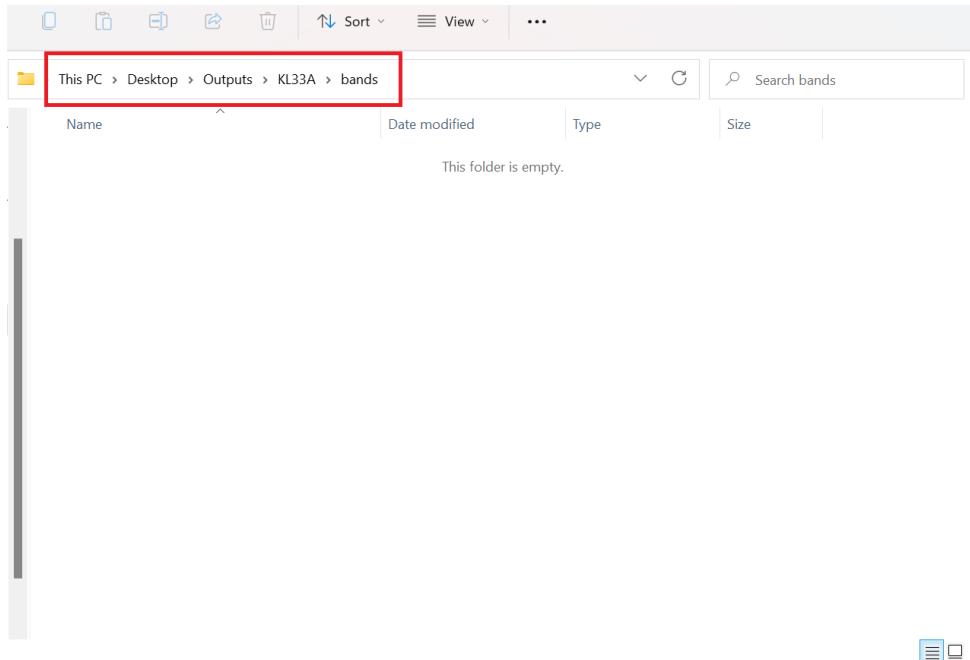


Figure 16: Create a folder to save processed data

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 17).

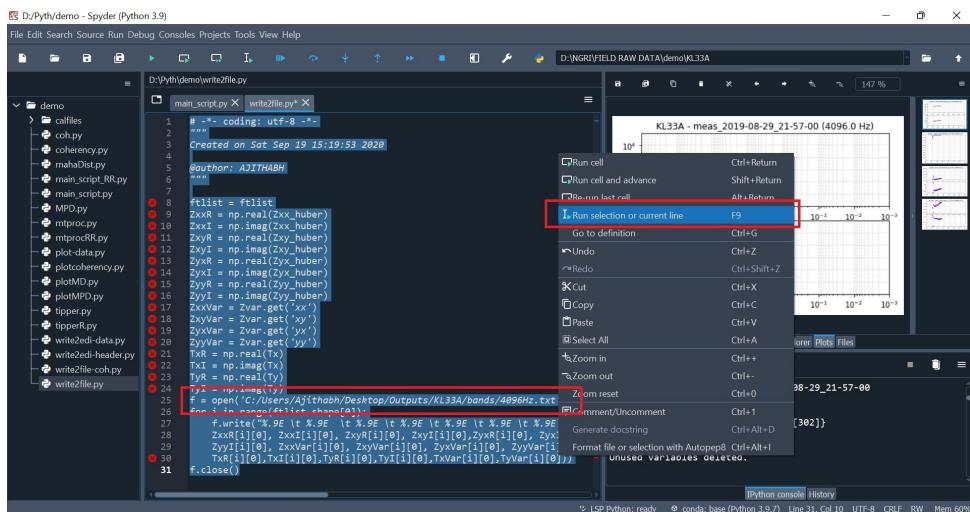


Figure 17: Save data in file

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 18)

```

D:\Pyth\demo\write2edi-header.py
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Pyth\demo\write2edi-header.py
main_script.py x write2file.py x write2edi-header.py x
10 lat_d = np.floor(lat)
11 lat_m2 = (lat - lat_d) * 60
12 lat_m = np.floor(lat_m2)
13 lon_d = np.floor(lon)
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/Ajithbab/Desktop/Outputs/KL33A/KL33A-HEADER.edt",
29         "w")
30 f.write("FILEBY=")
31 f.write(procinfo.get('selectedsite'))
32 f.write("\n ACQBY=" + "CSIR - NGR, INDIA")
33 f.write("\n FILEID=" + "719529")
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 STDOVRSL=" + "SEG 1.0")
43 f.write("\n PROGVERS= SigNet 0.4.1")
44 f.close()

```

Figure 18: Save header information

After executing processing for a measurement, always close ‘Console’ (Figure 19).

```

D:\Pyth\demo\write2edi-header.py
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Pyth\demo\write2edi-header.py
main_script.py x write2file.py x write2edi-header.py x
10 lat_d = np.floor(lat)
11 lat_m2 = (lat - lat_d) * 60
12 lat_m = np.floor(lat_m2)
13 lon_d = np.floor(lon)
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/Ajithbab/Desktop/Outputs/KL33A/KL33A-HEADER.edt",
29         "w")
30 f.write("FILEBY=")
31 f.write(procinfo.get('selectedsite'))
32 f.write("\n ACQBY=" + "CSIR - NGR, INDIA")
33 f.write("\n FILEID=" + "719529")
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 STDOVRSL=" + "SEG 1.0")
43 f.write("\n PROGVERS= SigNet 0.4.1")
44 f.close()

```

Figure 19: 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 20). Create a text file named ‘all.txt’.

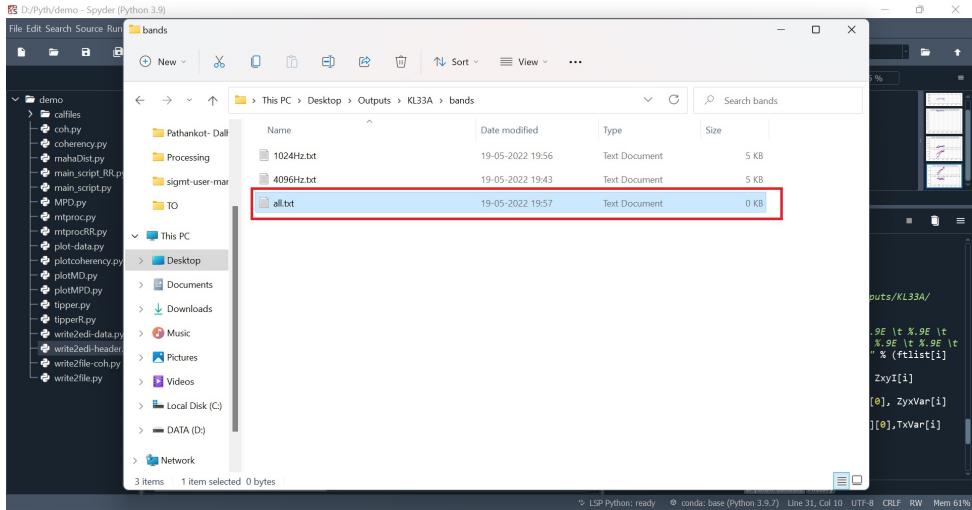


Figure 20: Create a file all.txt

Copy the data from 4096Hz.txt file and 1024Hz.txt file to all.txt as in Figure 21. Always keep higher frequency data in the top.

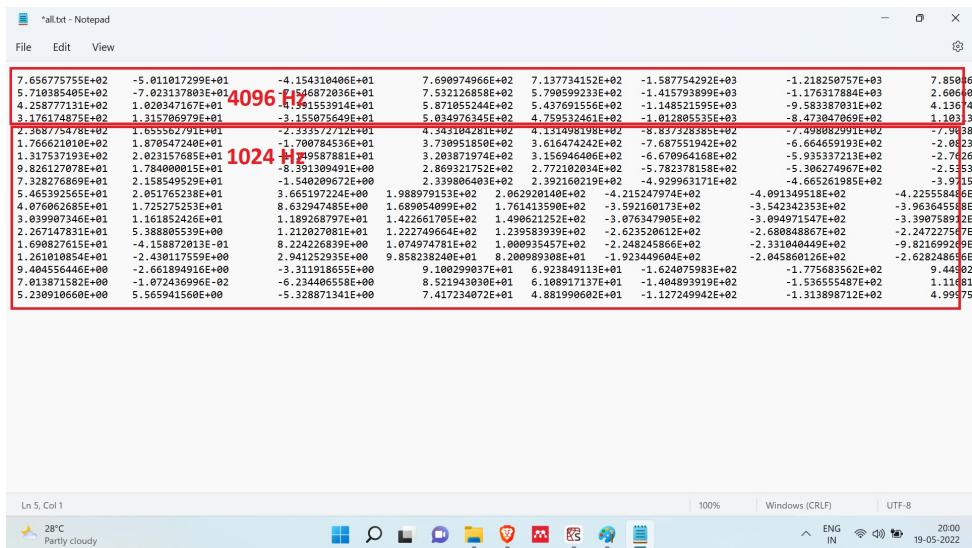


Figure 21: Copy data to all.txt file

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 22).

The screenshot shows the Spyder IDE interface with several windows open. On the left, there's a file browser window 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 of code that opens the EDI file. On the right, a 'Console 4/A' window is visible, showing command-line options and a help menu. Another red box highlights the 'Run selection or current line' option in the menu bar.

```

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/all.txt'
12 f = open("C:/Users/Ajithabh/Desktop/Outputs/KL33A/KL33A-DATA.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 TxR = data[:,13]
30 TxR = data[:,14]
31 Tyr = data[:,15]
32 TyT = data[:,16]

```

Figure 22: Write data as in EDI

Now, two files are existing in sites folder ('Figure 23). Copy data in the file KL33A- DATA.edi to the end of header information in KL33A-HEADER.edi file (Figure 24). Please correct 'NFREQ' variable with actual number of frequencies in the data (Figure 24). Now, KL33A-HEADER.edi is a standard EDI file ready to use.

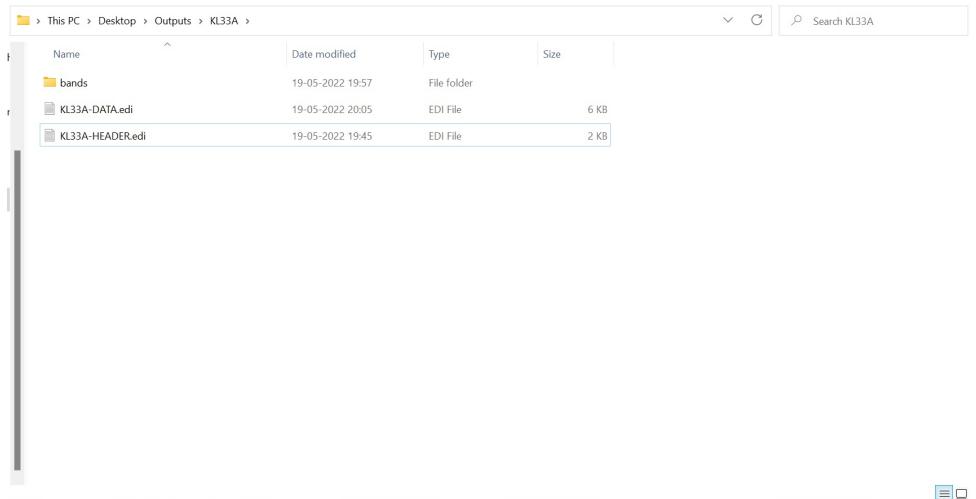


Figure 23: Files ready to create EDI

```
*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
5.710385459E+02 4.258777131E+02 3.176174875E+02 2.368775478E+02 1.766621018E+02
1.797532163E+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 24: Edit NFREQ and save EDI

5.3 Data selection tools

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

- Title Bar:** D:\Pyth\SigMT-Distribution - Spyder (Python 3.9)
- Menu Bar:** File Edit Search Source Run Debug consoles Projects Tools View Help
- Toolbar:** Standard file operations like Open, Save, Print, etc.
- Code Editor:** The main window displays the file `D:\Pyth\SigMT-Distribution\main_script.py`. The code is written in Python and defines two functions: `alpha_degE` and `pdmat`. The `alpha_degE` function performs a selection based on a threshold of 0.7. The `pdmat` function performs a selection based on a threshold of 0.6. Both functions use NumPy arrays and matrix operations.
- Source Control:** Shows the status as "Clean".
- Console:** A small preview of the console output.
- Object Browser:** Shows the current objects in memory.
- Help:** A large panel titled "Usage" provides information on how to use the IDE, including keyboard shortcuts for selection and automatic help.
- File Explorer:** Shows the project structure with files like `main_script.RP.py`, `main_script.py`, and `mtprocR.R`.
- Variable Explorer:** Shows variables like `pdmat` and `alpha_degE`.
- Merge Files:** Shows the status as "No changes".

Figure 25: Data selection tools section

In Figure 25, 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 provided in '`CohTre`' variable for each target frequency. You can see the coherency values for each target frequency using the '`plot-coherency.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 27. By analysing the values in the figure, you can set coherency threshold values.

```

D:\Pyth\demo - Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Pyth\demo\plotcoherency.py
main_script.py X plotcoherency.py X
1  # coding: utf-8 -
2  #
3  # Thu Aug 20 15:52:42 2020
4  #
5  # AJITHABH
6  #
7  #
8  import matplotlib
9  from matplotlib import pyplot as plt
10 from matplotlib import colors
11 from matplotlib import cm
12 from matplotlib import colorbar
13 from matplotlib import figure
14 from matplotlib import axes
15 from matplotlib import font_manager
16 from matplotlib import style
17 from matplotlib import colors
18 from matplotlib import cm
19 from matplotlib import colorbar
20 from matplotlib import figure
21 from matplotlib import axes
22 from matplotlib import font_manager
23 from matplotlib import style
24 from matplotlib import colors
25 from matplotlib import cm
26 from matplotlib import colorbar
27 from matplotlib import figure
28 from matplotlib import axes
29 from matplotlib import font_manager
30 from matplotlib import style
31 from matplotlib import colors
32 from matplotlib import cm
33 from matplotlib import colorbar
34 from matplotlib import figure
35 from matplotlib import axes
36 from matplotlib import font_manager
37 from matplotlib import style
38 from matplotlib import colors
39 from matplotlib import cm
40 from matplotlib import colorbar
41 from matplotlib import figure
42 from matplotlib import axes
43 from matplotlib import font_manager
44 from matplotlib import style
45 from matplotlib import colors
46 from matplotlib import cm
47 from matplotlib import colorbar
48 from matplotlib import figure
49 from matplotlib import axes
50 from matplotlib import font_manager
51 from matplotlib import style
52 from matplotlib import colors
53 from matplotlib import cm
54 from matplotlib import colorbar
55 from matplotlib import figure
56 from matplotlib import axes
57 from matplotlib import font_manager
58 from matplotlib import style
59 from matplotlib import colors
60 from matplotlib import cm
61 from matplotlib import colorbar
62 from matplotlib import figure
63 from matplotlib import axes
64 from matplotlib import font_manager
65 from matplotlib import style
66 from matplotlib import colors
67 from matplotlib import cm
68 from matplotlib import colorbar
69 from matplotlib import figure
70 from matplotlib import axes
71 from matplotlib import font_manager
72 from matplotlib import style
73 from matplotlib import colors
74 from matplotlib import cm
75 from matplotlib import colorbar
76 from matplotlib import figure
77 from matplotlib import axes
78 from matplotlib import font_manager
79 from matplotlib import style
80 from matplotlib import colors
81 from matplotlib import cm
82 from matplotlib import colorbar
83 from matplotlib import figure
84 from matplotlib import axes
85 from matplotlib import font_manager
86 from matplotlib import style
87 from matplotlib import colors
88 from matplotlib import cm
89 from matplotlib import colorbar
90 from matplotlib import figure
91 from matplotlib import axes
92 from matplotlib import font_manager
93 from matplotlib import style
94 from matplotlib import colors
95 from matplotlib import cm
96 from matplotlib import colorbar
97 from matplotlib import figure
98 from matplotlib import axes
99 from matplotlib import font_manager
100 from matplotlib import style
101 from matplotlib import colors
102 from matplotlib import cm
103 from matplotlib import colorbar
104 from matplotlib import figure
105 from matplotlib import axes
106 from matplotlib import font_manager
107 from matplotlib import style
108 from matplotlib import colors
109 from matplotlib import cm
110 from matplotlib import colorbar
111 from matplotlib import figure
112 from matplotlib import axes
113 from matplotlib import font_manager
114 from matplotlib import style
115 from matplotlib import colors
116 from matplotlib import cm
117 from matplotlib import colorbar
118 from matplotlib import figure
119 from matplotlib import axes
120 from matplotlib import font_manager
121 from matplotlib import style
122 from matplotlib import colors
123 from matplotlib import cm
124 from matplotlib import colorbar
125 from matplotlib import figure
126 from matplotlib import axes
127 from matplotlib import font_manager
128 from matplotlib import style
129 from matplotlib import colors
130 from matplotlib import cm
131 from matplotlib import colorbar
132 from matplotlib import figure
133 from matplotlib import axes
134 from matplotlib import font_manager
135 from matplotlib import style
136 from matplotlib import colors
137 from matplotlib import cm
138 from matplotlib import colorbar
139 from matplotlib import figure
140 from matplotlib import axes
141 from matplotlib import font_manager
142 from matplotlib import style
143 from matplotlib import colors
144 from matplotlib import cm
145 from matplotlib import colorbar
146 from matplotlib import figure
147 from matplotlib import axes
148 from matplotlib import font_manager
149 from matplotlib import style
150 from matplotlib import colors
151 from matplotlib import cm
152 from matplotlib import colorbar
153 from matplotlib import figure
154 from matplotlib import axes
155 from matplotlib import font_manager
156 from matplotlib import style
157 from matplotlib import colors
158 from matplotlib import cm
159 from matplotlib import colorbar
160 from matplotlib import figure
161 from matplotlib import axes
162 from matplotlib import font_manager
163 from matplotlib import style
164 from matplotlib import colors
165 from matplotlib import cm
166 from matplotlib import colorbar
167 from matplotlib import figure
168 from matplotlib import axes
169 from matplotlib import font_manager
170 from matplotlib import style
171 from matplotlib import colors
172 from matplotlib import cm
173 from matplotlib import colorbar
174 from matplotlib import figure
175 from matplotlib import axes
176 from matplotlib import font_manager
177 from matplotlib import style
178 from matplotlib import colors
179 from matplotlib import cm
180 from matplotlib import colorbar
181 from matplotlib import figure
182 from matplotlib import axes
183 from matplotlib import font_manager
184 from matplotlib import style
185 from matplotlib import colors
186 from matplotlib import cm
187 from matplotlib import colorbar
188 from matplotlib import figure
189 from matplotlib import axes
190 from matplotlib import font_manager
191 from matplotlib import style
192 from matplotlib import colors
193 from matplotlib import cm
194 from matplotlib import colorbar
195 from matplotlib import figure
196 from matplotlib import axes
197 from matplotlib import font_manager
198 from matplotlib import style
199 from matplotlib import colors
200 from matplotlib import cm
201 from matplotlib import colorbar
202 from matplotlib import figure
203 from matplotlib import axes
204 from matplotlib import font_manager
205 from matplotlib import style
206 from matplotlib import colors
207 from matplotlib import cm
208 from matplotlib import colorbar
209 from matplotlib import figure
210 from matplotlib import axes
211 from matplotlib import font_manager
212 from matplotlib import style
213 from matplotlib import colors
214 from matplotlib import cm
215 from matplotlib import colorbar
216 from matplotlib import figure
217 from matplotlib import axes
218 from matplotlib import font_manager
219 from matplotlib import style
220 from matplotlib import colors
221 from matplotlib import cm
222 from matplotlib import colorbar
223 from matplotlib import figure
224 from matplotlib import axes
225 from matplotlib import font_manager
226 from matplotlib import style
227 from matplotlib import colors
228 from matplotlib import cm
229 from matplotlib import colorbar
230 from matplotlib import figure
231 from matplotlib import axes
232 from matplotlib import font_manager
233 from matplotlib import style
234 from matplotlib import colors
235 from matplotlib import cm
236 from matplotlib import colorbar
237 from matplotlib import figure
238 from matplotlib import axes
239 from matplotlib import font_manager
240 from matplotlib import style
241 from matplotlib import colors
242 from matplotlib import cm
243 from matplotlib import colorbar
244 from matplotlib import figure
245 from matplotlib import axes
246 from matplotlib import font_manager
247 from matplotlib import style
248 from matplotlib import colors
249 from matplotlib import cm
250 from matplotlib import colorbar
251 from matplotlib import figure
252 from matplotlib import axes
253 from matplotlib import font_manager
254 from matplotlib import style
255 from matplotlib import colors
256 from matplotlib import cm
257 from matplotlib import colorbar
258 from matplotlib import figure
259 from matplotlib import axes
260 from matplotlib import font_manager
261 from matplotlib import style
262 from matplotlib import colors
263 from matplotlib import cm
264 from matplotlib import colorbar
265 from matplotlib import figure
266 from matplotlib import axes
267 from matplotlib import font_manager
268 from matplotlib import style
269 from matplotlib import colors
270 from matplotlib import cm
271 from matplotlib import colorbar
272 from matplotlib import figure
273 from matplotlib import axes
274 from matplotlib import font_manager
275 from matplotlib import style
276 from matplotlib import colors
277 from matplotlib import cm
278 from matplotlib import colorbar
279 from matplotlib import figure
280 from matplotlib import axes
281 from matplotlib import font_manager
282 from matplotlib import style
283 from matplotlib import colors
284 from matplotlib import cm
285 from matplotlib import colorbar
286 from matplotlib import figure
287 from matplotlib import axes
288 from matplotlib import font_manager
289 from matplotlib import style
290 from matplotlib import colors
291 from matplotlib import cm
292 from matplotlib import colorbar
293 from matplotlib import figure
294 from matplotlib import axes
295 from matplotlib import font_manager
296 from matplotlib import style
297 from matplotlib import colors
298 from matplotlib import cm
299 from matplotlib import colorbar
299

```

Figure 26: Run codes

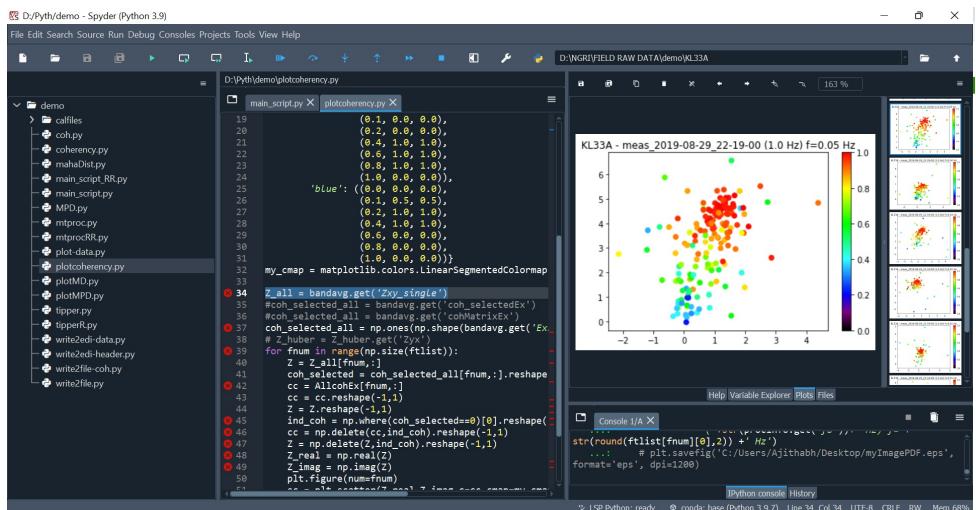


Figure 27: Coherency values

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 28). 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 29).

Next is to select stacks, give a range of stacks to be discarded in this case. In the example (Figure 29), the stacks between 300 and 403 will be discarded from processing.

‘pdflag’ should be 1 to perform the tasks.

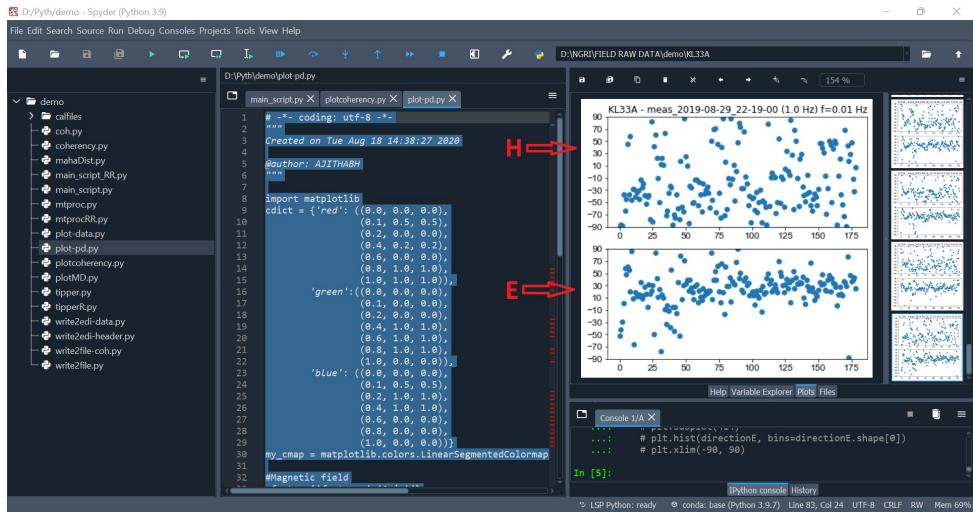


Figure 28: Polarization directions

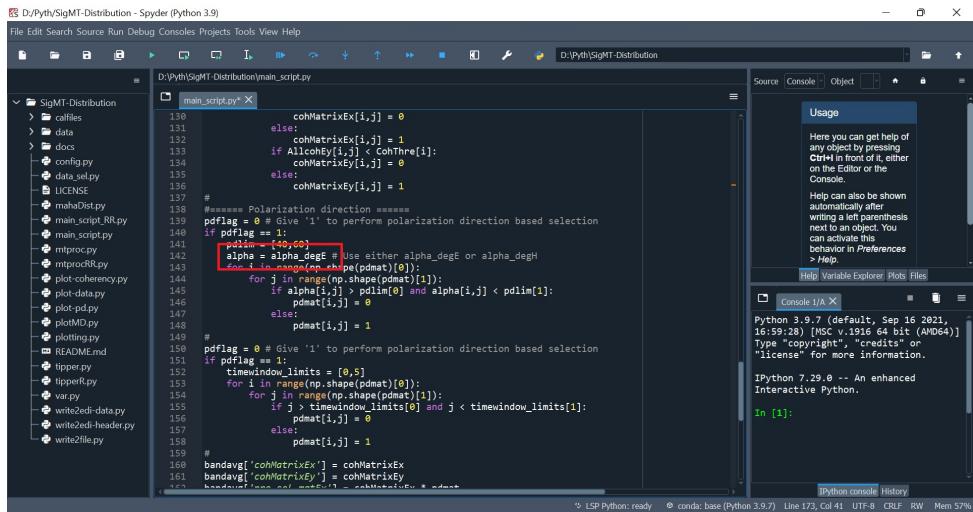


Figure 29: Magnetic or Electric Polarization direction

Once data selection constraints are set, just run a part of code as highlighted (to end of the script) in figure 30.

```

D:\Pyth\SigMT-Distribution - Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Pyth\SigMT-Distribution\main_script.py
main_script.py
101 #===== Start band averaging =====
102 # No need to edit
103 # Average value after calibration and averaging using parzen window
104 ffile,bandavg = mproc.bandavg(tts,procinfo,config)
105
106 #===== Band averaging finished =====
107 timer_end = time.time()
108 print('Elapsed time: ' + str(timer_end - timer_start) +'s')
109 del timer_start, timer_end
110 print('Finished.')
111
112 #
113 #####Data selection tools section. Coherency threshold & Polarization direction
114 cohMatrixEx = np.ones(np.shape(bandavg.get('ExEx')),dtype=float)
115 cohMatrixEx *= np.ones(np.shape(bandavg.get('ExEx')),dtype=float)
116 cohMatrixEx *= np.ones(np.shape(bandavg.get('ExEx')),dtype=float)
117 # Calculation of coherency values for all time windows
118 AllcohEx = data_sel.cohEx(bandavg)
119 AllcohEy = data_sel.cohEy(bandavg)
120 # Calculation of polarization directions for all time windows
121 alpha_degM,alpha_degP = data_sel.pdvalues(bandavg)
122
123 ##### Coherency threshold =====
124 ctflag = 0 # Give '1' to perform coherency threshold based selection
125 if ctflag == 1:
126     CohThre = [0.7,0.7,0.7,0.7,0.7,0.7,0.7,0.7,0.7,0.9,0.9,0.9]
127     for i in range(np.shape(AllcohEx)[0]):
128         for j in range(np.shape(AllcohEx)[1]):
129             if AllcohEx[i,j] < CohThre[i]:
130                 cohMatrixEx[i,j] = 0
131             else:
132                 cohMatrixEx[i,j] = 1
133
134

```

Figure 30: Run this part of script

5.4 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.