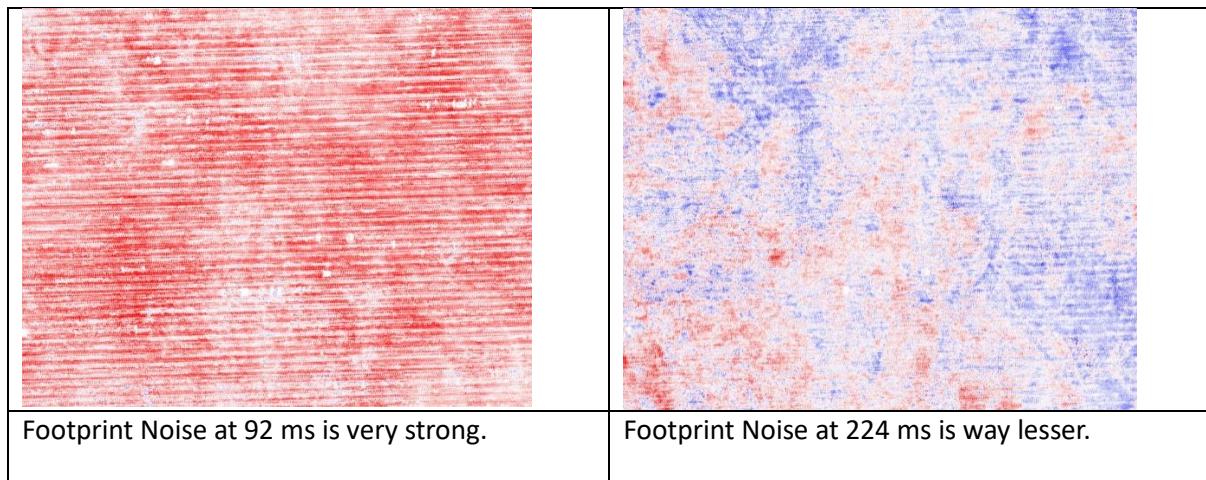


## Post Stack Seismic Acquisition Footprint Reduction Using Empirical Mode Decomposition on “Constant Time” Traces

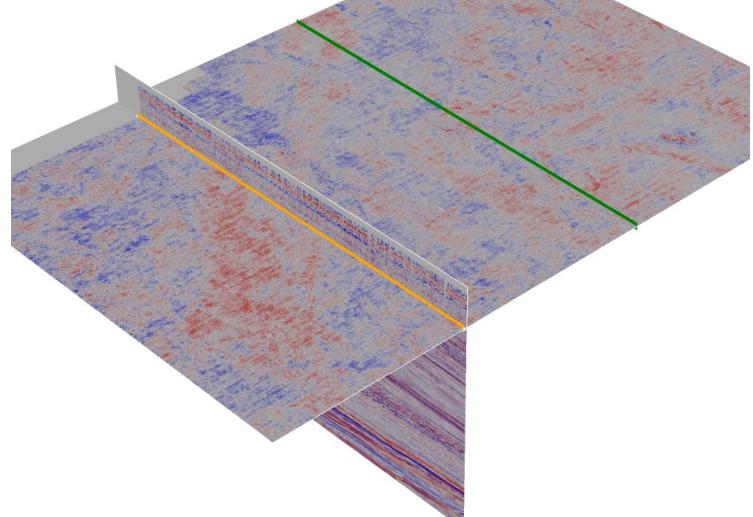
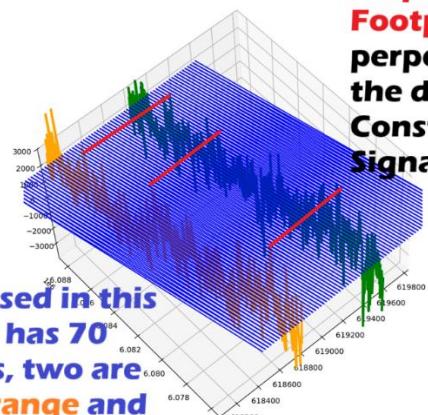
If you look at any seismic cube with acquisition footprint noise, you will see that Shallow Time Slices have more noise compared to deeper ones. This noise varies from one time slice to another and is actually visualized on time slices itself.

**Hence the filtering to reduce or remove this noise would make better sense if done on a time slice basis and not on the entire volume or vertical trace wise.**



All data used in this article is from F3 Block, Netherlands, Courtesy dGB Earth Sciences.

All the samples of the seismic cube which lie at the intersection of any one inline and a time slice can be considered as what I call as **constant time traces**.

	<p>I propose the term Constant Time Trace as the Intersection between any Inline and any Time Slice (for that matter also, the intersection between any crossline and any Time Slice) Here, the vertical axis is time.</p>
 <p><b>Acquisition Footprints are perpendicular to the direction of Constant Time Signals.</b></p> <p><b>Time Slice used in this experiment has 70 such signals, two are shown in orange and green</b></p>	<p>Same information as above but here the <b>Vertical Axis is Amplitude</b>. Acquisition Footprints appear as Impulse Noises superimposed on the Geological Signal on both signals shown. Also, these signals/ Constant Time Traces are non-stationary in nature (mean has a trend, that is mean is not zero) and we can use EMD to filter out the noise.</p>

So, I have developed a new method to filter out Acquisition Footprint Noise, which runs **perpendicular to the acquisition footprint noise (shown in red above)** and filters one by one, each of the “constant time” traces in the time slice.

### What is EMD?

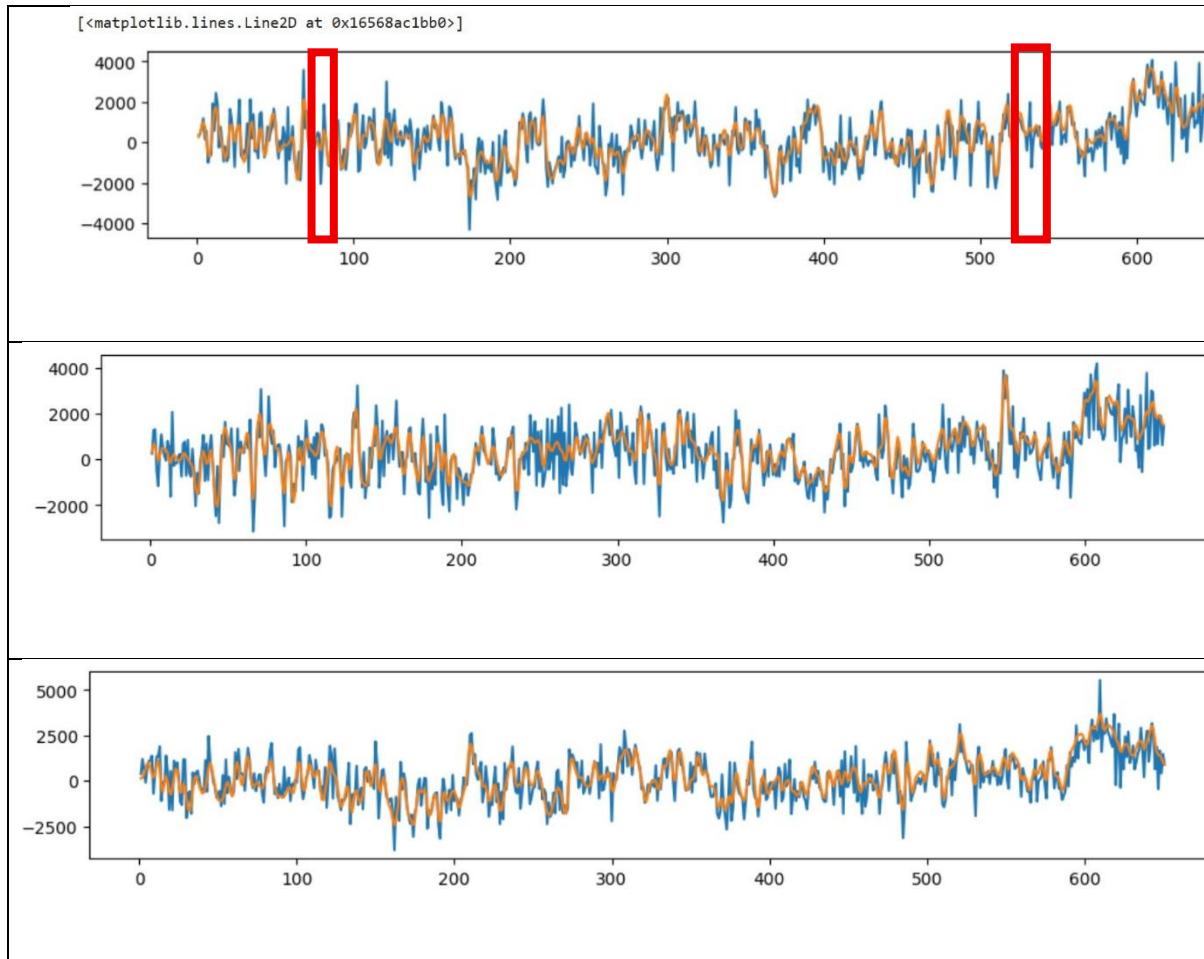
Just like Fourier Transform breaks down a signal using sin and cosine “basis”, EMD breaks down a signal into IMFs (Intrinsic Mode Functions) and a residual trend. Unlike Fourier Transform, in EMD, the **basis is extracted FROM THE DATA ITSELF**. In this decomposition generally either the first IMF or the sum of (First and Second IMFs) is the noisy part of the data. EMD can decompose nonlinear, nonstationary data effectively.

Here I show a few of the:

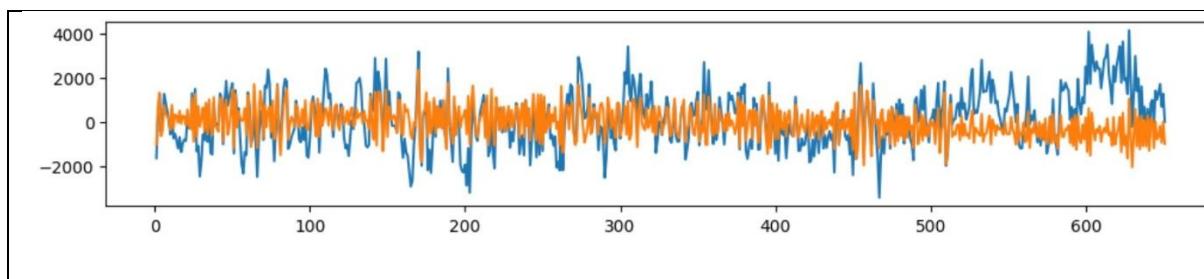
1. Constant Time Traces,
2. Their Denoised Version (Original – First IMF) and
3. The noise itself (First IMF is noise)

In the following plots, I give 3 examples of running EMD on Constant Time Traces. Notice how the **footprint affected signal** is filtered with EMD and the **denoised CT Trace avoids Spiky Impulses**. It more or less retains the original frequency of the data also.

**Notice inside the red boxes, how beautifully it avoids the Footprint Spikes.**



The second plot shows **the noise (First IMF)** superimposed on the **original noisy signal**.



Conventional EMD suffers from a problem known as Mode Mixing, and to avoid it Ensemble EMD is used. **Ensemble EMD (EEMD)**.

In EEMD, you run many trials (many times, typically 100 by default) of EMD by adding different noise sequence to the original signal. Then average the IMFs from all the trials to get the average IMF for cleaner decomposition. The important parameters in EEMD are:

1. `noise_width`: amplitude of white noise added to the original signal. I took `noise_width = 0.6` after testing, it means my noise is 60% of the standard deviation of original signal.
2. `ensemble_size`: my number of trials in this experiment is 300. By default, the parameter is 100.

Now, I will walk you through the method, with one constant time trace and you may repeat the SAME process on the other 69 traces. Then merge all the results into one dataframe and visualize the result.

- 1.) First, we have to install PyEMD library. Then import the following:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PyEMD import EEMD
```

PyEMD simplifies and automates a lot of computations for you. For every signal, it does something called as “sifting”:

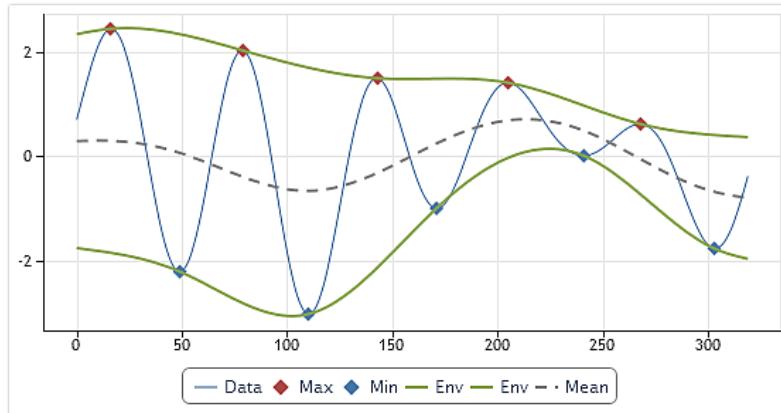


Fig Courtesy: <https://www.mql5.com/en/articles/439>

#### Sifting:

- Finds all **max** points
- Finds all **min** points
- Then draws upper and lower **envelopes**
- Find the average of both envelopes
- Subtract the average from the signal to see if an IMF has been reached
- Repeat till IMF found

- 2.) Then we import the .csv file for any constant time trace say trace 31 and save it as a dataframe. Fill the green box below, with your file path for any trace:

```
df = pd.read_csv(r'')
```

	X	Y	Z	amplitude
0	619055.375	6.073926e+06	264.0	-731.0
1	619054.625	6.073951e+06	264.0	-996.0
2	619054.000	6.073976e+06	264.0	156.0
3	619053.250	6.074001e+06	264.0	-1917.0

Now, there has to be a sample number to serve as X-axis for plotting of these signals. Hence, I add a new column to the dataframe called “sample number”. I know there are 651 samples in this direction. Then I plot the amplitude on the vertical axis and sample number on horizontal axis.

```
df['sampleNumber'] = np.linspace(1, 651, len(df))

signal = df['amplitude']

plt.figure(figsize=(12, 2))
plt.plot(df['sampleNumber'], signal)

[<matplotlib.lines.Line2D at 0x2945e457050>]
```

This signal by its appearance looks non stationary and perfect to be analysed by a method like EMD.

- 3.) Then we instantiate the EMD class by calling the constructor EEMD(). We use Ensemble EMD because the EMD method has a short coming of mode mixing. Ensemble EMD runs a number of trials and averages results. And for you and me to have the same result we should have the same noise seed. I randomly chose it

```
: myemd = EEMD()
myemd.noise_seed(3999)
```

- 4.) To see the currently used number of trials and noise characteristics, we do the following

```
myemd. trials
100

myemd.noise_width
0.05
```

5.) Now, I experimented and changed these two parameters. So, we run the following:

```
myemd = EEMD()
myemd.noise_seed(3999)
myemd.trials = 300
myemd.noise_width = 0.6
```

6.) Now, variables need to be in array format but currently they are dataframe columns which are Pandas Series. So we change these to arrays to use them in EEMD() method

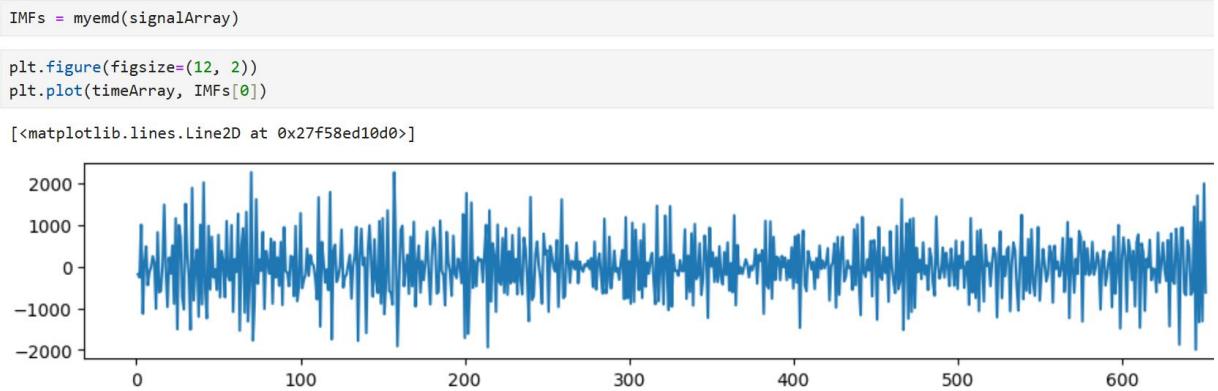
```
type(df.sampleNumber)
pandas.core.series.Series

timeArray = df['sampleNumber'].to_numpy()

signalArray = signal.values
```

7.) Then the main step below, to compute the Intrinsic Mode Functions. The First IMF, is IMF[0].

Generally, the first IMF or IMF[0] is the noise extracted from the signal. (As in our case)



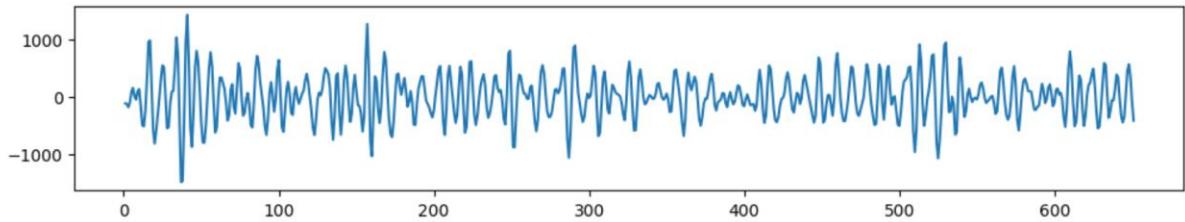
8.) You can also know how many IMFs have been computed. As you see below, there are 9 IMFs and each have 651 samples, the same number of samples as in the original CT Trace:

```
IMFs.shape
```

```
(9, 651)
```

9.) Let us check the second IMF, how it looks:

```
plt.figure(figsize=(12, 2))
plt.plot(timeArray, IMFs[1])
[<matplotlib.lines.Line2D at 0x27f59bf8140>]
```

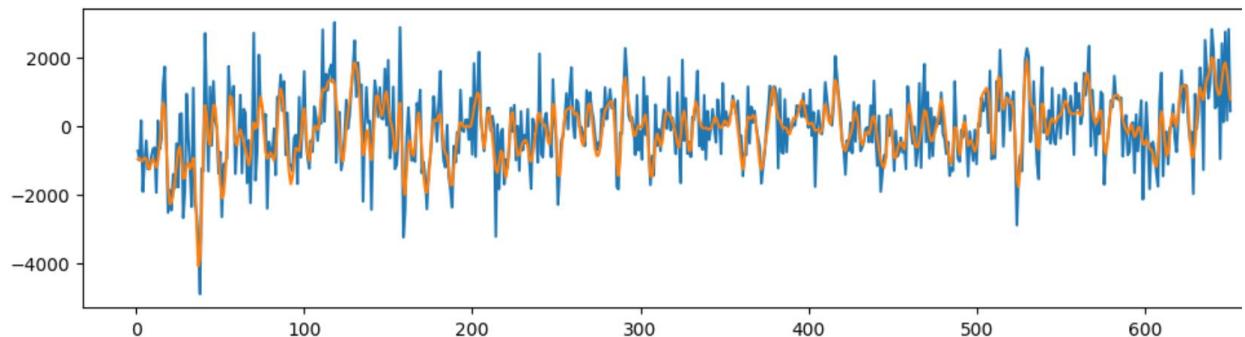


10.) Our Denoised Signal is the sum of all IMFs except the First IMF, which is IMFs[0]. Below, OP stands for Optimized Parameters (remember I optimized trials = 300 and noise\_width = 0.6)

```
DenoisedSignal_OP = np.sum(IMFs[1:], axis=0) # Skip first 1 IMFs
```

11.) Now, we plot this Optimized Parameter, Denoised Signal:

```
plt.figure(figsize=(12, 3))
plt.plot(timeArray, signal)
plt.plot(timeArray, DenoisedSignal_OP)
[<matplotlib.lines.Line2D at 0x27f59c45040>]
```



All the blue spikes are actually seen as Footprint Noise in 2D. (Align all 1D Spikes in a line from all Constant Time Traces and you will have the Footprint Noise Signature in 2D). Seems like this is working.

12.) All the files provided with names “ConstTimeTraceX.csv” where X is from 1 to 70, may have dataframe columns of

- X
- Y
- Z
- amplitude

- sampleNumber
- DenoisedSignal
- difference

I followed the given method for default parameters of trials = 100 and noise\_width = 0.05.  
Now in this exercise, I repeat the processes with Optimized Parameters (OP) and **hence we generate new columns like DenoisedSignal\_OP and difference\_OP**

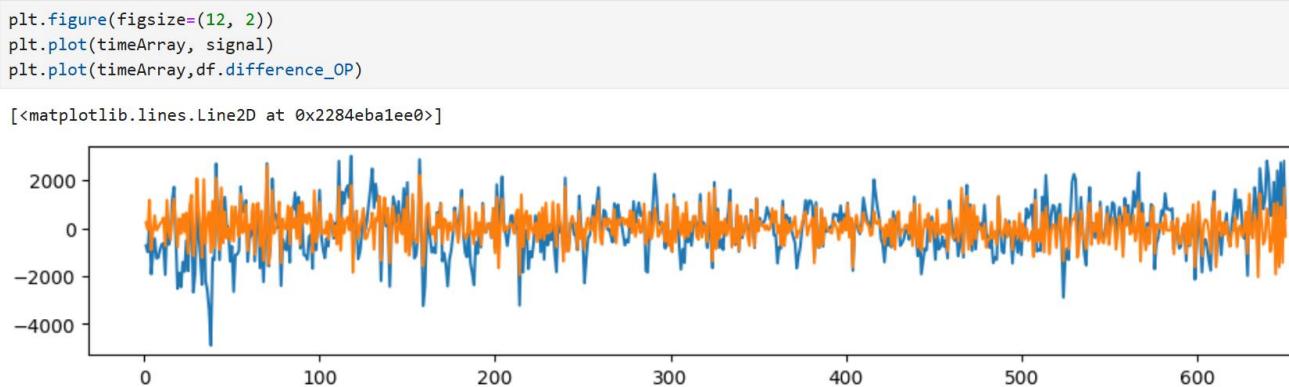
```
df['DenoisedSignal_OP'] = DenoisedSignal_OP
df['difference_OP'] = df.amplitude - df.DenoisedSignal_OP
df
```

	Unnamed: 0	X	Y	Z	amplitude	sampleNumber	DenoisedSignal	difference	DenoisedSignal_OP	difference_OP
0	0	619055.375	6.073926e+06	264.0	-731.0	1.0	-880.110568	149.110568	-970.371079	239.371079
1	1	619054.625	6.073951e+06	264.0	-996.0	2.0	-681.995067	-314.004933	-939.886351	-56.113649
2	2	619054.000	6.073976e+06	264.0	156.0	3.0	-923.822898	1079.822898	-1021.380251	1177.380251
3	3	619053.250	6.074001e+06	264.0	-1917.0	4.0	-1281.751101	-635.248899	-1010.933258	-906.066742
4	4	619052.625	6.074026e+06	264.0	-1020.0	5.0	-1386.007148	366.007148	-922.020936	-97.979064

13.) I will export all 70 traces (later I merge them to get the entire time slice):

```
df.to_csv(r'C:\\\ConstTimeTrace31_output_OP.csv')
```

14.) You will find 70 such files which I have merged to get the time slice. We now will plot the original signal and the noise extracted:



15.) Next let us plot the entire time slice with the difference plot to see how the method works:

```

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PyEMD import EEMD

df = pd.read_csv(r'C:\\output_OptimizedParams.csv')

df

```

	X	Y	Z	amplitude	denoisedSignal	difference	DENOISED_SIGNAL_OP	DIFF_OP
0	618305.625	6.073905e+06	264.0	351.0	361.498810	-10.498797	263.099426	87.900574
1	618305.000	6.073930e+06	264.0	409.0	525.027588	-116.027565	471.265167	-62.265163
2	618304.250	6.073955e+06	264.0	786.0	685.015198	100.984818	796.624084	-10.624114

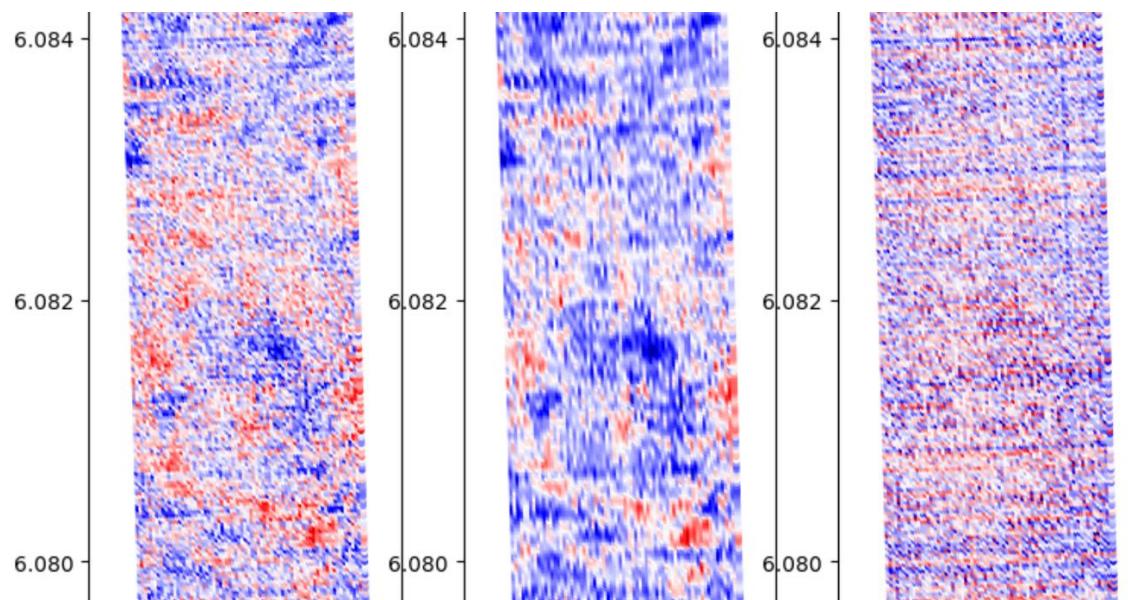
16.) Finally, it is important to plot all the three plots with same minimum and maximum values.  
So input, denoised output and noise all 3 are plotted with same min and max.

```

vmin = -3000
vmax = 3000
fig, axes = plt.subplots(ncols=3, figsize=(9,20))

axes[0].scatter(df.X, df.Y, c = df.amplitude, s=5, cmap = 'seismic')
axes[1].scatter(df.X, df.Y, c = df.DENOISED_SIGNAL_OP, s=5, cmap = 'seismic')
axes[2].scatter(df.X, df.Y, c = df.DIFF_OP, s=5, cmap = 'seismic')

```



Thanks.

