OBS Noise **AN**alysis (obsnan) Manual

Chao An (anchao@sjtu.edu.cn, April 2024)

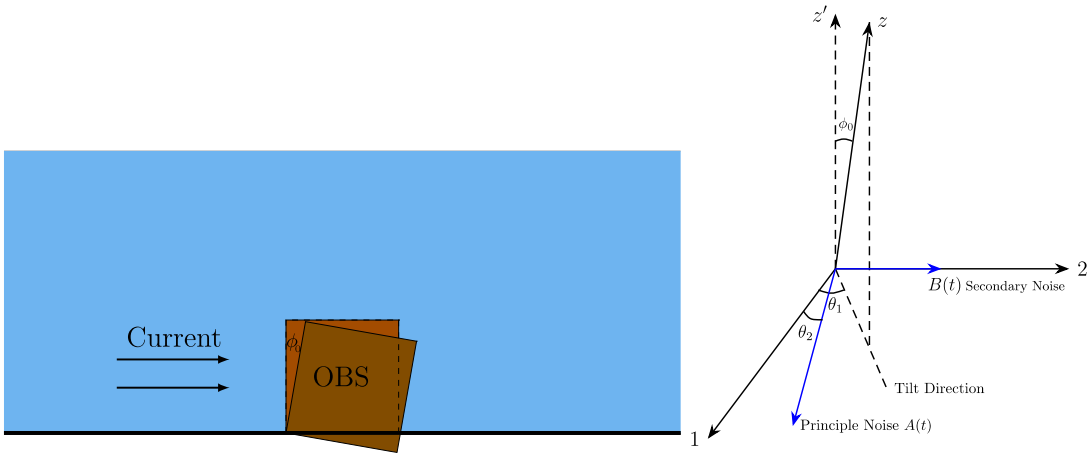
These codes are originally developed to calculate and remove the tilt noise and pressure compliance noise in OBS data. But except these two applications, the package provides a framework of cutting continuous OBS data into small segments. For each time segment, one can use a FORTRAN program or SAC commands to perform other OBS noise analysis. Thus, this code package is named OBS noise analysis. In this manual, instructions on the analysis of tilt, pressure compliance and horizontal water wave noise are given. Users can develop other noise analysis tools according to the examples based on this program framework.

Contents

1 Introduction of Tilt Noise, Pressure Compliance Noise and Horizontal Water Wave Noise

A brief summary of the algorithm to calculate and remove tilt noise, pressure compliance noise and horizontal water wave noise is given bellow. For more details, refer to Crawford et al. (2000, BSSA), Bell et al. (2015, BSSA), An et al. (2020, SRL), An et al. (2022, SRL) and Zhang and An (2024, JGR).

Tilt Noise is thought to be caused by ocean currents. Ocean-bottom currents tilt the instrument, and generate a principle noise and secondary (background) noise in the horizontal direction. The horizontal noise leaks into the vertical channel due to imperfect leveling. Thus, if the tilt of the instrument is known, the tilt noise in the vertical channel can be calculated using the horizontal records.



In order to predict and remove the tilt noise, two angles must be determined, i.e., the tilt direction of the instrument in the horizontal plane with respect to the direction of channel 1, and the tilt angle of the instrument with respect to the gravitational direction. Here we use θ_1 and ϕ_0 to denote these two angles. They are calculated based on the correlation relationships of different channels of continuous noise data. The algorithm is explained by Bell et al. (2015, BSSA) and An et al. (2022, SRL). The tilt noise in the vertical channel is calculated by

$$T = (a_1 \cos\theta_1 + a_2 \sin\theta_1) \sin\phi_0 \,,$$

where a_1 and a_2 are the noise recorded in the two horizontal channels.

Pressure Compliance Noise is generated by ocean surface water waves. Water waves of wavelength larger than approximately half of the water depth can penetrate the water column and generate ocean-bottom pressure change, which in consequence causes vertical deformation of the seafloor. A pressure transfer function (PTF) is defined as the ratio of the seafloor acceleration and bottom pressure in the frequency domain. It depends on the underlying Earth structure, and it is shown to be linearly proportional to frequency (An et al. 2020, SRL). Using continuous noise data, we can calculate the PTF at each station. Then the

pressure compliance noise in the vertical channel is calculated by multiplying the pressure records by the PTFs.

The above two types of noise is for the vertical channel. For the horizontal channels, in deep water, the major noise is the Tilt Noise (or Principle Noise) as shown in the above figure. Such noise can be suppressed in one channel by rotating the two horizontal channels to the current direction (see An et al. 2022, SRL). In shallow water (<300 m), another major noise is the water wave noise. **Water Wave Noise** has a quantitative feature which is that the horizontal acceleration is linearly proportional the time derivative of ocean-bottom pressure (dp/dt). Thus, similar to the vertical pressure compliance noise, we can calculate a horizontal pressure transfer function (HPTF) between horizontal acceleration and dp/dt to calculate and remove such noise (see Zhang and An 2024, JGR).

2 Program Structure

The file structure of the program is explained in the following table. Python programs are marked in green, parameter files are black, and folders are blue. Folders indicated by # are automatically created by the program.

obsnan.py	The main Python script
obsnan.ctl	Parameters for the main script
tiltCalculateAllFrequenciesTilt.py	Automatically calculate tilt in many frequency bands
tiltCombineAllFrequencies.py	Find the final tilt angles
ptfCalculateAllFrequencies.py	Automatically calculate PTFs in many frequency bands
ptfCombineAllFrequencies.py	Assemble the final PTFs
hptfCalculateAllFrequencies.py	Automatically calculate HPTFs in many frequency bands
parallelDivideStations.py	Divide stations into groups for parallel calculation
parallelCombineResults.py	Combine results of parallel computing
obsnan	Folder of Class obsnan, all Python functions
__init__.py	Definition of Class obsnan
GlobalFunctions.py	Global useful functions
CommonFunctions.py	Common functions of class obsnan
Tilt.py	Functions related to tilt noise analysis
PressureTransferFunction.py	Functions related to pressure transfer functions
HorizontalPressureTransferFunction.py	Functions related to horizontal PTFs
EventNoiseRemoval.py	Functions related to removal of noise for event data
FFT.py	Functions related to calculating FFT
EmptyFunction.py …	Empty function as an example for other applications
StationsWaterDepth.ctl	Water depth at all the OBS stations
AuxiliaryFilesPythonScripts	More useful Python scripts
bin	Fortran programs (calculateCorrelation.f90 …)
MatlabTools	Matlab scripts to plot results
NoiseData	Folder of OBS data ready to calculate tilt and PTF
# PTFResults	PTF results after combing all calculated PTFs
# PTFResults_f1_f2_TimeLength	PTF results in a certain frequency band
# TiltResults	Tilt results considering tilt in multiple frequency bands
# TiltResults_f1_f2_TimeLength	Tilt results in a certain frequency band
# TempFiles	Files after filtering and temporary files

# TaskXX_StaYY_ZZ	Folders for parallel calculation

3 Initiation of the Program

All the Python and Fortran files are provided in the package. Users should prepare some other files and folders, including:

StationsWaterDepth.ctl:
This file is indeed not a must. However, if it is provided, it will be more convenient. The file has two columns, the first column is the station name and the second is the water depth. See the following screenshot. Note that the line number is not part of the file.

```
 1 LT12 93.0
 2 LT10 108.0
 3 LT09 137.0
 4 LT02 153.0
 5 LT20 153.0
 6 LT14 156.0
 7 LT04 157.0
 8 LT03 158.0
 9 LT11 160.0
10 LT16 170.0
```

bin:
In folder bin, it is required to recompile the FORTRAN90 programs, because the compilation depends on the operating system and the compiler. Refer to *makefile* in the folder. Note that SAC libs are required here for FORTRAN programs to read SAC files.

NoiseData:
The continuous noise data are stored in this folder. All the data files must end with .SAC. The program checks all the files ending with .SAC, and then determine the initial time, station name and channels by reading the SAC headers via SAC software. For each station, the channel names must be specified. See function *obsnan/ GlobalFunctions.py*. The name of the folder can be changed and specified in the configure file *obsnan.ctl*.

```
 6
 7 def identifySACFiles(SortStationsBy=None, SortFilesBy=None):
 8
 9         # Add your channel names here
10         AllowedChannelsInOrder = [['BH1','BH2','BHZ','BDH'], ['LH1','LH2','LHZ','LDH'], ['HH1','HH2','HHZ','HDH'],\
11                 ['BHE','BHN','BHZ','BDH'], ['LHE','LHN','LHZ','LDH'], ['HHE','HHN','HHZ','HDH'], \
12                 ['BH1','BH2','BHZ','HYD'],\
13                 ['BHE','BHN','BHZ','BHH'],\
14                 ['SHX','SHY','SHZ','HYD']]
15
```

obsnan.ctl:
This file provides the parameters for the calculation.

4 Calculation of Instrumental Tilt

4.1 Parameters in obsnan.ctl

```
20 ================================================:=========================
21 Tilt Calculation Flow and Parameters    (Units) :  Values                  |
22 ================================================:=========================
23
24 Filter Noise Data for Tilt    (1/y/yes, 0/n/no) :  yes
25 Calculate Correlation for All Segments          :  yes
26 Find Tilt Axis for All Segments                 :  1
27 Remove Bad Correlation for Tilt Calculation     :  1
28
29 Frequency Range to Find Tilt Axis          (Hz) :  0.02, 0.05
30 Time Length to Determine Tilt Axis     (second) :  2500
31 Min Value to Determine Correlation              :  0.90
32 Average Tilt Angle Near Days           (+-days) :  2
33
34 Overwrite if Results exist tilt(1/y/yes, 0/n/no):  no
35
```

Here options 1, y and yes are equal. *yes* is used for the former two steps to indicate that the calculation takes a long time.

The program will first calculate the tilt for all the data segments (steps 1 to 3, Line 24 to 26). Then, the program discards some results (step 4, Line 27) based on the minimum correlation specified (0.90 in this example, Line 31). Note that results of tilt angle larger than 5 degrees are discarded in the program. Refer to function *removeTiltAxisBadCorrelation* around Line 355 in *obsnan/Tilt.py*.

The frequency range and the segment time length can be changed. Our experience is that the results will be approximately the same for frequency below 0.1 Hz, which is the dominant frequency band of tilt noise.

The time length can also be changed. Here the length is roughly determined to be about 50 times the longest period.

How exactly minimum correlation works is a little complex, and can be seen in *obsnan/Tilt.py*. Generally speaking, a higher parameter means more reliable but less results. If this parameter is 1.0, all the results are discarded; if it is 0, all the tilt results are reserved. If only this parameter is changed, it is not necessary to recalculate the first three steps. Thus, one can use options *no, no, 0, 1* for the four steps (Line 24 to 27) to only re-run step 4, and update the results. Results are saved in files *TiltAxis_STA_Selected.dat*. Remember to overwrite existing results (option yes in Line 34).

Line 32 gives a time length to average the tilt results. When using the tilt angles to calculate and remove tilt noise, tilt angles are averaged over a certain time period to avoid fluctuations.

4.2 Calculation of tilt

Run **obsnan.py**. The program will analyze all the SAC files in the OBS data folder, and then calculate the instrument tilt. The calculation takes a couple of steps:

Step 1 (Line 24): The noise data are filtered between the specified frequency range. Resulted SAC files are named as *filt_STA_FileNum_CHN_f1_f2*, and stored in folder *TempFiles*. These filtered files are deleted after the calculation is completed.

Step 2 (Line 25): The filtered data are cut into small segments, and for each segment, the maximum/minimum correlation parameters between different channels when rotating the horizontal channels are calculated. Results are saved in files *Correlation_STA.dat* in folder *TiltResults_f1_f2_TimeLength*. Files *Correlation_STA.dat* have 100 columns.

Step 3 (Line 26): Using the results in *Correlation_STA.dat*, θ_1 and ϕ_0 are calculated. Results are saved in files *TiltAxis_STA.dat*. *TiltAxis_STA.dat* has 106 columns, of which the first 100 columns are the same as *Correlation_STA.dat*, and 101 and 102 are θ_1 and ϕ_0. Four more other parameters are also calculated and saved.

Step 4 (Line 27): Based on files TiltAxis_STA.dat, some tilt results are discarded due to low correlation. Tilt results after selection are saved in files *TiltAxis_STA_Selected.dat*. Files *TiltAxis_STA_Selected.dat* have the same file structure as *TiltAxis_STA.dat* except that some rows are discarded.

One can run **tiltCalculateAllFrequencies.py** to automatically calculate the tilt in a few frequency bands. Note that tiltCalculateAllFrequencies.py only changes the parameters for the frequency bands and time length in file obsnan.ctl, so other parameters must be correctly provided in obsnan.ctl before you run the program.

4.3 Parallel calculation

The calculation takes a relatively long time, so sometimes it is useful to speed up the calculation via parallel techniques. The parallel method is actually very simple: it divides the stations into groups and conducts the calculation in different folders at the same time.

Firstly, change the parameters in **parallelDivideStations.py** and run it. The script will divide the stations into groups and create working folders *TaskXX_StaYY_ZZ*. The noise data files will

be copied into each folder. Because there will be temporary files in the data folder during calculation, parallel calculation must be done in separate folders.

```
#! /usr/bin/python3

import os, subprocess, datetime, math, sys, re, shutil
from inspect import currentframe, getframeinfo
from obsnan.GlobalFunctions import identifySACFiles, saveSACFilesListResult, readSACFilesListResult
from obsnan.GlobalFunctions import excuteCommand, excuteSACCommand, obtainSACFileHeaders, printOverPreviousLine


nTasks = 2

FromDataFolderName = './NoiseData'
ToDataFolderName = 'NoiseData'
OverWriteData = False
ReanalyzeFromPathFiles = False

CopyFiles = ['obsnan.ctl','obsnan.py','tiltCalculateAllFrequencies.py','ptfCalculateAllFrequencies.py','ptfCombineAllFrequencies.py','StationsWaterDepth.ctl']

CopyFolders = ['bin','obsnan']
#CopyFolders = ['bin','obsnan','TiltResults']

```

Run **obsnan.py** or **tiltCalculateAllFrequencies.py** or **ptfCalculateAllFrequencies.py** or **hptfCalculateAllFrequencies.py** in each task folder.

Run **parallelCombineResults.py** to move all the results into the results folder of the main path. (*Paralleling calculation is not tested at the latest update*).

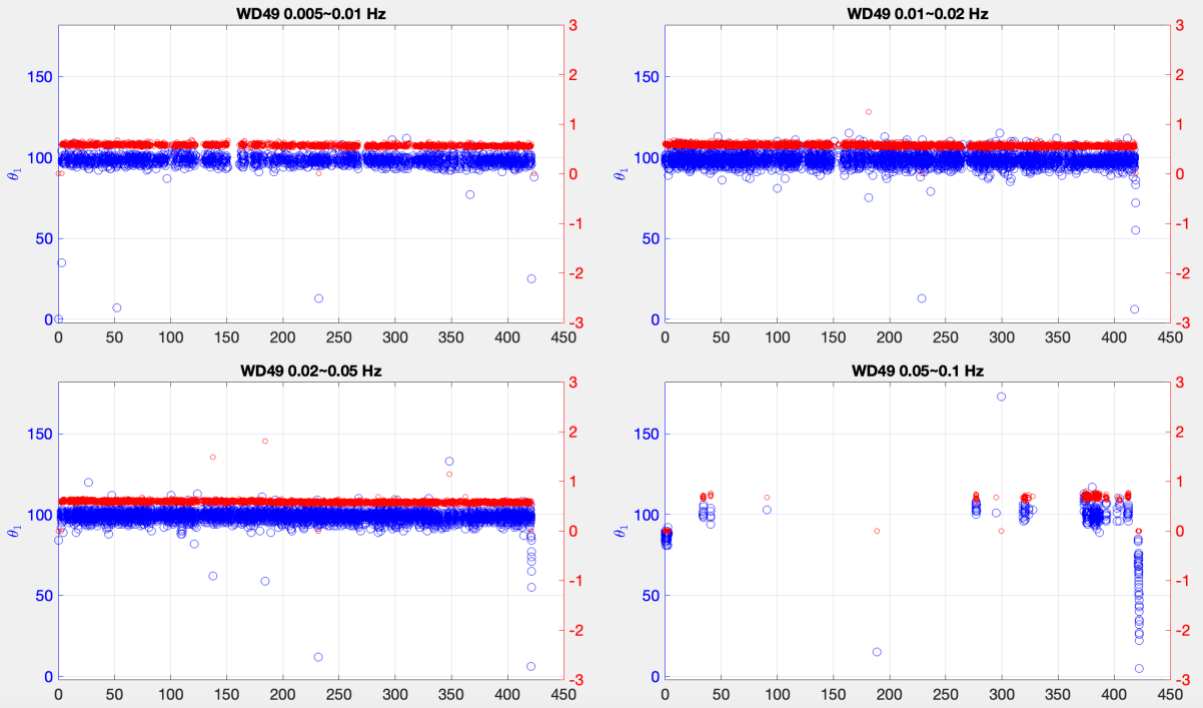
4.4 Postprocess of the tilt results

Use Matlab script *calculateTiltMatDataAll.m* to convert the tilt results to Matlab mat files. It also calculates the average of the results over several prescribed time lengths.

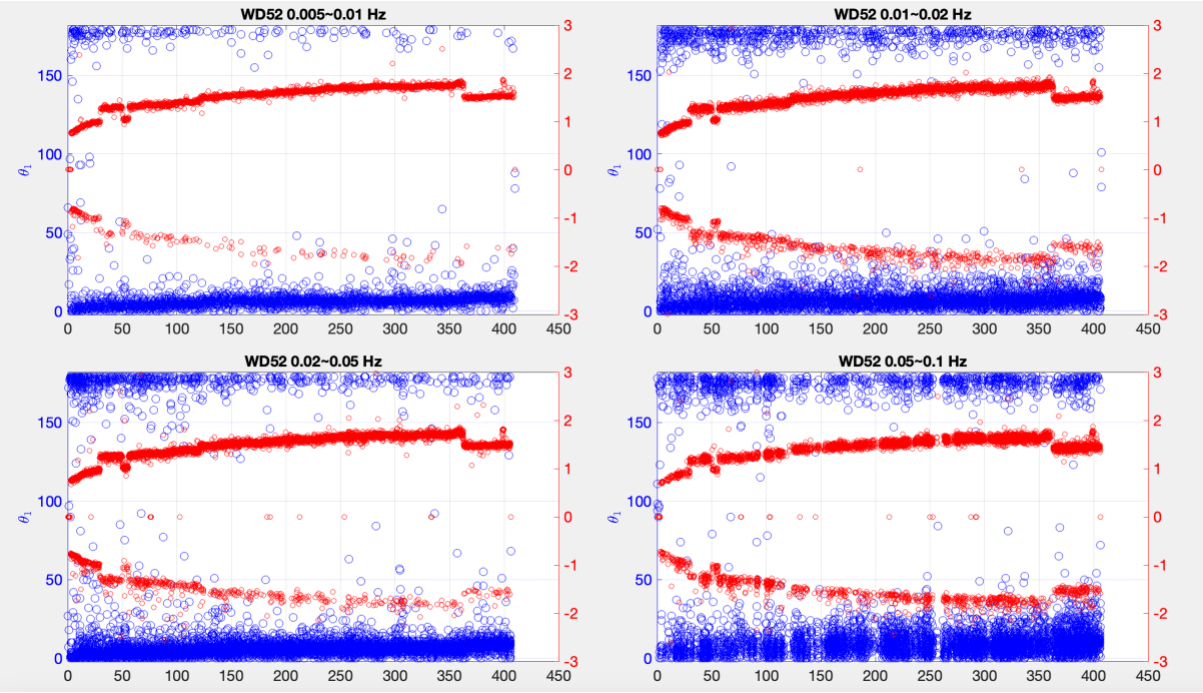
```
function calculateTiltMatDataAll
% 0.0 0.0 to represent TiltResults

%Frequencies = [0.0 0.0]
%SuffixSegmentLength = [0];

Frequencies = [0.005 0.01; 0.01 0.02; 0.02 0.05; 0.05 0.1];
SuffixSegmentLength = [10000, 5000, 2500, 1000];

```

Use Matlab script *plotTheta1Phi0VaryInTimeSingleStationAllFrequencies.m* to check the results. For example, run *plotTheta1Phi0VaryInTimeSingleStationAllFrequencies* ('wd49') in Matlab command window. There are generally three possibilities: good, correction-required and bad/no results. Examples are shown below.

[1] Good: very constant tilt direction (θ_1) and tilt angel (ϕ_0) are obtained.



[2] Correction-required: Tilt angel (ϕ_0) has two branches, one is positive and the other one is negative. Tilt direction (θ_1) is close to 0 or 180. This is because θ_1 is obtained by a global search in range 0 to 180. When it is near 0, it can be 0 or 180, which leads to opposite title angle (ϕ_0).



To correct these angles, create a file with name *StationsTheta1Theta2Range.ctl* and specify a different range of θ_1. Below shows an example of the format of

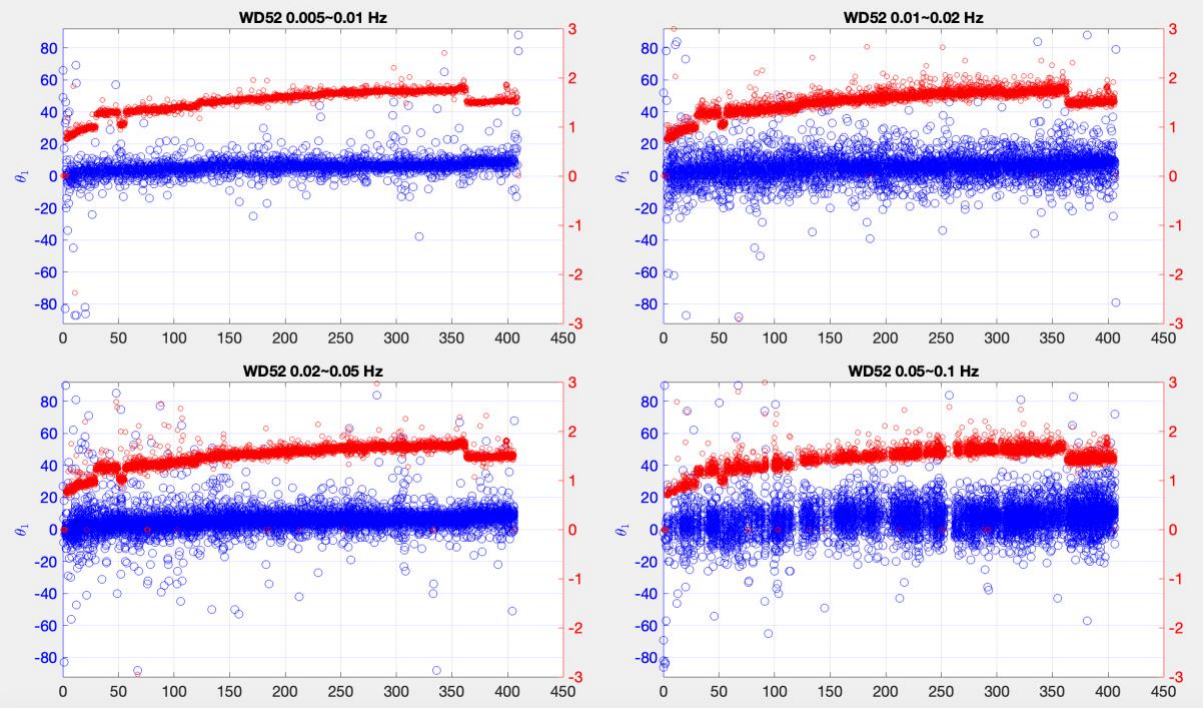
StationsTheta1Theta2Range.ctl. Ignore the second column which is used in the program for other purposes.

```
 1 2 1 XX00     # 1: 0~180; 2: -90~90
 2
 3 2 1 LT20
 4 2 1 WS74
 5 2 1 WD55
 6 2 1 WD48
 7 2 1 LA30
 8 2 1 LD45
 9 2 1 WD52
10 2 1 WD70
```

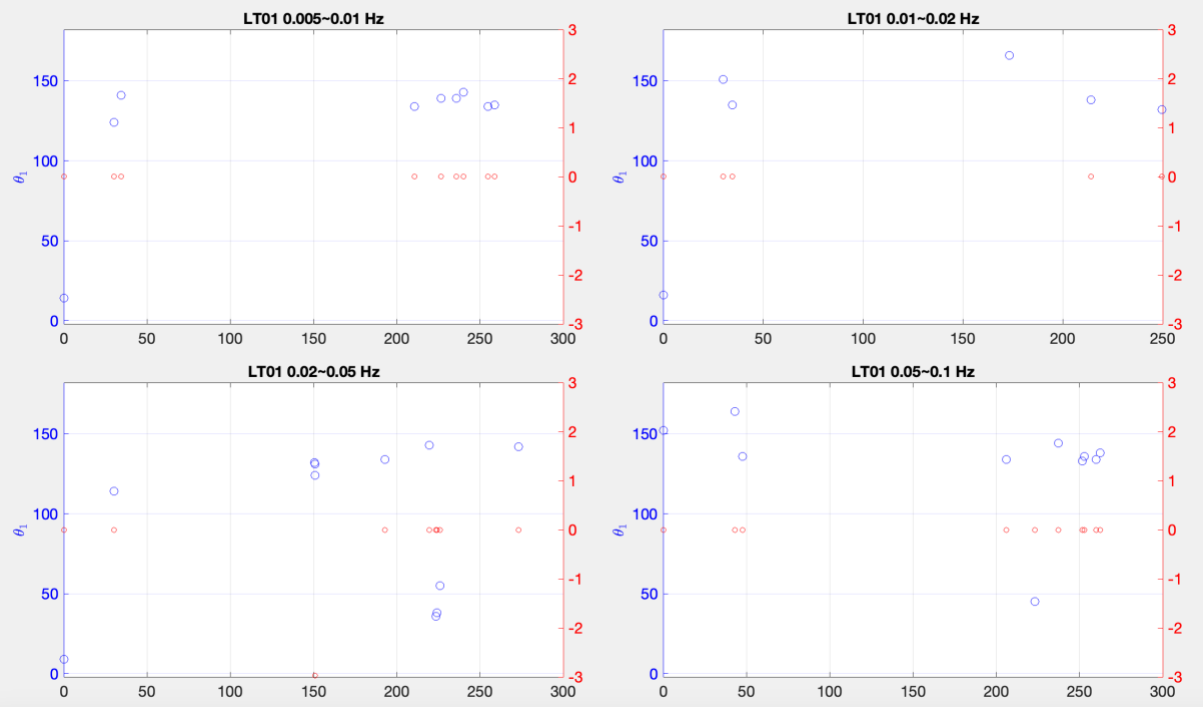
Then re-calculate the tilt direction (θ_1) and tilt angle (ϕ_0). Use the following parameters in *obsnan.ctl*. It does not re-calculate the correlation parameters from the seismic data, but only re-calculate the tilt direction and tilt angle. Run **tiltCalculateAllFrequencies.py.**

```
20 ================================================:============================
21 Tilt Calculation Flow and Parameters     (Units) :  Values                    |
22 ================================================:============================
23
24 Filter Noise Data for Tilt     (1/y/yes, 0/n/no) :  no
25 Calculate Correlation for All Segments           :  no
26 Find Tilt Axis for All Segments                  :  1
27 Remove Bad Correlation for Tilt Calculation      :  1
28
29 Frequency Range to Find Tilt Axis           (Hz) :  0.02, 0.05
30 Time Length to Determine Tilt Axis      (second) :  2500
31 Min Value to Determine Correlation               :  0.90
32 Average Tilt Angle Near Days           (+-days) :  2
33
34 Overwrite if Results exist tilt(1/y/yes, 0/n/no):  yes
35
```

Then in Matlab, run *calculateTiltMatDataAll.m* to re-convert the tilt results to Matlab mat files. Then use *plotTheta1Phi0VaryInTimeSingleStationAllFrequencies.m* to check the results.



[3] Bad: very few result points are obtained. Possibly because the data have problems (e.g. spikes) or the correlation is low because the dominant noise is not tilt noise.



4.5 Combine tilt results in different frequency bands

Generally, the calculated tilt angles are good only in certain frequency bands at different stations. Thus, it is necessary to choose the tilt results of a certain frequency band as the final results. Use script **tiltCombineAllFrequencies.py** and change some of the parameters in the script.

```
#! /usr/local/bin/python3

import os, subprocess, datetime, math, sys, shutil, re
from inspect import currentframe, getframeinfo



DefaultResultFolder = './TiltResults_0.02_0.05_2500'

StationsExceptions = [['LT20','WS74','WS75','LD36','WD55','WD48','WD52'],\
        ['LT01','LD40','LA32','LA28','LD35','WS73','LA25','WD56','WS72','WD46','WD63','WD64','LD37','LA26','LA21']];
FoldersExceptions = ['./TiltResults_0.005_0.01_10000','']

```

By default, tilts results in folder DefaultResultFolder will be used. For some other stations indicated by StationsExceptions, results in folders FoldersExceptions will be used. Empty folders indicate that at these stations we do not obtain good results and zero tilt will be used.

The final tilt results will be stored in folder **TiltResults**. This will be the tilt parameters used for calculating and removing tilt noise in the vertical channel.

Now one can use *calculateTiltMatDataAll.m* to calculate the mat files, and then use *plotTheta1Theta2Phi0VaryInTimeSingleStation.m* to verify the final tilt results. Remember to change the parameters in *calculateTiltMatDataAll.m.*

```
function calculateTiltMatDataAll(varargin)
% 0.0 0.0 to represent TiltResults

Frequencies = [0.005 0.01; 0.01 0.02; 0.02 0.05; 0.05 0.1];
SuffixSegmentLength = [10000, 5000, 2500, 1000];

Frequencies = [0.0 0.0];
SuffixSegmentLength = [0];
```

5 Calculation of Pressure Transfer Functions (PTFs)

5.1 Calculating PTFs in different frequency bands

The procedure is very similar to the calculation of tilt. In **obsnan.ctl**, use the following parameters. Tilt calculation is skipped since the results are already obtained.

```
20 ================================================:============================
21 Tilt Calculation Flow and Parameters    (Units) :  Values                     |
22 ================================================:============================
23
24 Filter Noise Data for Tilt    (1/y/yes, 0/n/no) :  no
25 Calculate Correlation for All Segments          :  no
26 Find Tilt Axis for All Segments                 :  0
27 Remove Bad Correlation for Tilt Calculation     :  0
28
29 Frequency Range to Find Tilt Axis          (Hz) :  0.05, 0.1
30 Time Length to Determine Tilt Axis     (second) :  1000
31 Min Value to Determine Correlation              :  0.90
32 Average Tilt Angle Near Days           (+-days) :  2
33
34 Overwrite if Results exist tilt(1/y/yes, 0/n/no):  no
35
36
37 ================================================:============================
38 PTF Calculation Flow and Parameters     (Units) :  Values                     |
39 ================================================:============================
40
41 Filter Noise Data for PTF     (1/y/yes, 0/n/no) :  yes
42 Find Pressure Transfer Function for All Segments:  yes
43 Remove Bad Correlation for PTF                   :  yes
44 Calculate Average PTF over Time                 :  yes
45 Fit Average PTF with Polynomials                :  1
46
47 Frequency Range to Find Pressure TF        (Hz) :  0.18, 0.2
48 Time Length to Determine Pressure TF   (second) :  10000
49 Min Value to Determine P/Z Correlation          :  0.80
50 Polynomial Order for PTF Fitting                :  1
51
52 Overwrite if Results exist PTF(1/y/yes, 0/n/no) :  no
53
```

Use **ptfCalculateAllFrequencies.py** to calculate PTFs in a few frequency bands. Frequency bands have small intervals in order to ensure high correlation between the vertical channel and pressure channel, and also to find out the upper frequency limit of the PTFs.

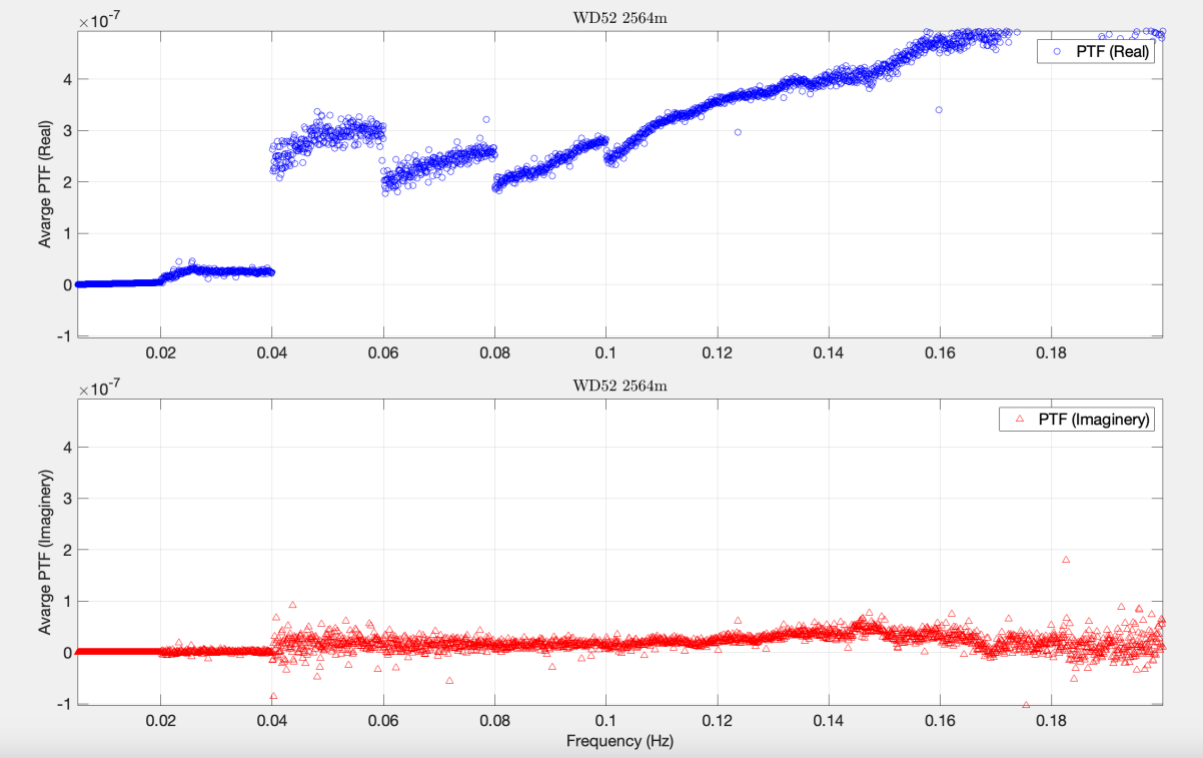
```
1  #! /usr/bin/python3
2
3  import os, subprocess, shutil, re, sys, datetime
4
5
6
7  freqs = [0.005,0.01,0.02,0.04,0.06,0.08,0.10,0.12,0.14,0.16,0.18,0.20]
8  #freqs = [0.10,0.12,0.14,0.16]
9
10 for i in range(0, len(freqs)-1):
11         f1 = freqs[i]
12         f2 = freqs[i+1]
```

The script simply modifies the frequency limits in *obsnan.ctl* and calls *obsnan.py*. After the calculation is finished, there will be a series of folders containing the PTF results, e.g, *PTFResults_Freqf1_f2_TimeLength*. Similar to the calculation of instrumental tilt, you can use the parallel programs.

5.2 Combining PTFs in different frequency bands

Use **ptfCombineAllFrequencies.py** to combine all the PTF results in different frequency bands and create a single PTF. By default, *ptfCombineAllFrequencies.py* combines the PTFs in all the frequencies, and perform polynomial fit in the frequency band.

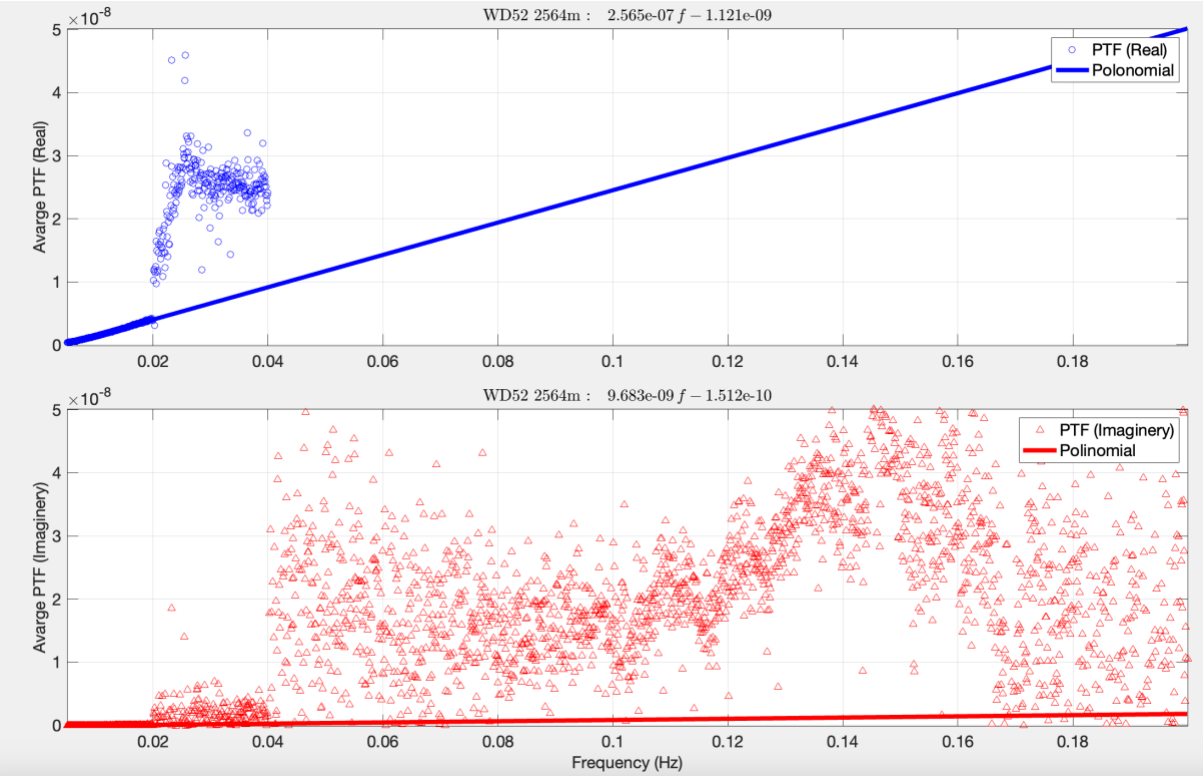
Use Matlab script **plotPressureTransferFunctionAveragePolyFit.m** to check all the PTFs. A typical computed PTF looks like the following plot. In a low frequency band, the PTF is very small, and it is roughly linearly proportional to frequency. This band is the real PTF we want. In the high frequency bands, the PTF is very large, and it is the seismic PTF (An et al. 2020, SRL), which is caused by microseism accelerating the overlying water column.



Create a file of name **StationsPTFFitFrequencies.ctl** to specify the frequency limits of the PTFs. The file format is displayed below. The first column is the station name, the second and third columns are the frequency limits, and there can be a fourth column to indicate the polynomial order of fitting. If one of the frequency limits is negative, then the program will use zero PTF at this station. At some stations in shallow water, it might be necessary to use a second order polynomial fit.

```
STA f1 f2 n # f1,f2 Poly Fit Frequency Range (negative to use zero PTF); n: poly fit order (default 1)


LT01 0.02 0.1 2
WD55 -0.01 0.04
WD52 0.005 0.02
WD49 0.005 0.02
LT20 0.02 0.1 2
WS74 0.01 0.04
LA39 0.005 0.02
WS75 0.005 0.03
LD40 0.005 0.04
LD36 0.005 0.04
WD48 0.01 0.02
```

After *StationsPTFFitFrequencies.ctl* is provided, re-run **ptfCombineAllFrequencies.py**. Now the PTFs are ready to use. Use Matalb script plotPressureTransferFunctionAveragePolyFit.m to check all the PTFs. The command can be like **plotPressureTransferFunctionAveragePolyFit('WD52','poly')** in order to show the polynomial fit. The polynomials are stored in files **PTFResults/PTF_STA_Average_Poly.dat**, and they will be used to remove the pressure compliance noise.



6 Calculation of Horizonal Pressure Transfer Functions (HPTFs)

6.1 Calculating HPTFs in different frequency bands

For stations in shallow water (<300 m), water-wave-induced noise is dominant around 0.07 Hz. Such noise can be removed in both the horizontal and vertical channels by calculating and using the pressure transfer functions (PTFs and HPTFs). The procedure of calculating HPTFs is very similar to the calculation of PTFs. In **obsnan.ctl**, use the following parameters. Tilt and PTF calculations are kipped since the results are already obtained.

```
55 ================================================:============================
56 HPTF Calculation Flow and Parameters    (Units) :  Values                    |
57 ================================================:============================
58
59 Filter Noise Data for HPTF    (1/y/yes, 0/n/no) :  yes
60 Find Horizontal PTF for All Segments            :  yes
61 Remove Bad Correlation for HPTF                 :  1
62 Calibrate Wave Direction                        :  1
63
64 Frequency Range to Find Horizontal PTF     (Hz) :  0.15, 0.2
65 Time Length to Determine Horizontal PTF(second) :  2000
66 Min Value to Determine dpdt/A Correlation       :  0.80
67
68 For Denoising: Average HPTF near Days  (+-days) :  2
69 For Denoising: Fit HPTF with Polynomial Order   :  1
70 For Denoising: Fit HPTF in Frequency Range      :  0.05,0.12
71 For Denoising: Use Wave Direction In Freq Range :  0.05, 0.1
72 For Denoising: Average Wave Direction (+-hours) :  2
73
74 Overwrite if Results exist HPTF(1/y/yes, 0/n/no):  no
75
```

Use **hptfCalculateAllFrequencies.py** to calculate HPTFs in a few frequency bands. Frequency bands have small intervals in order to ensure high correlation between the horizontal channel and pressure channel, and also to find out the upper frequency limit of the HPTFs.

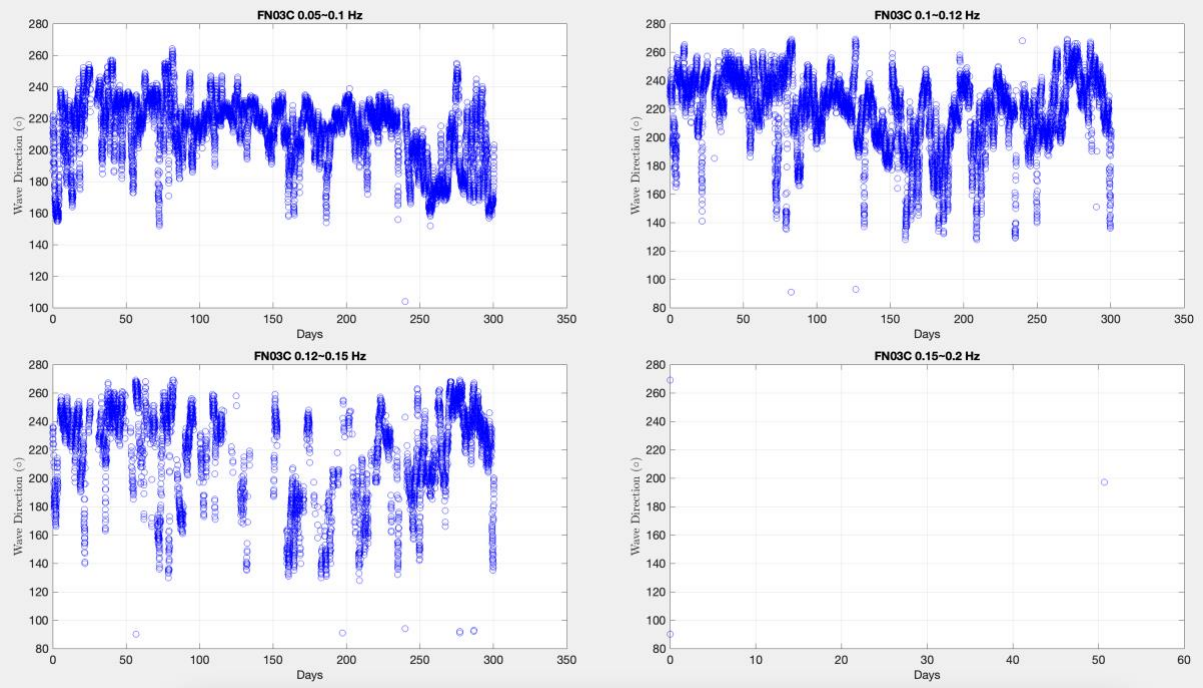
```
 1 #! /usr/bin/python3
 2
 3 import os, subprocess, shutil, re, sys, datetime
 4
 5
 6
 7 freqs = [0.02,0.05,0.10,0.12,0.15,0.20]
 8
 9 for i in range(0, len(freqs)-1):
10         f1 = freqs[i]
11         f2 = freqs[i+1]
12
13         print('\n f1, f2 == '+str(f1)+', '+str(f2)+'\n')
```

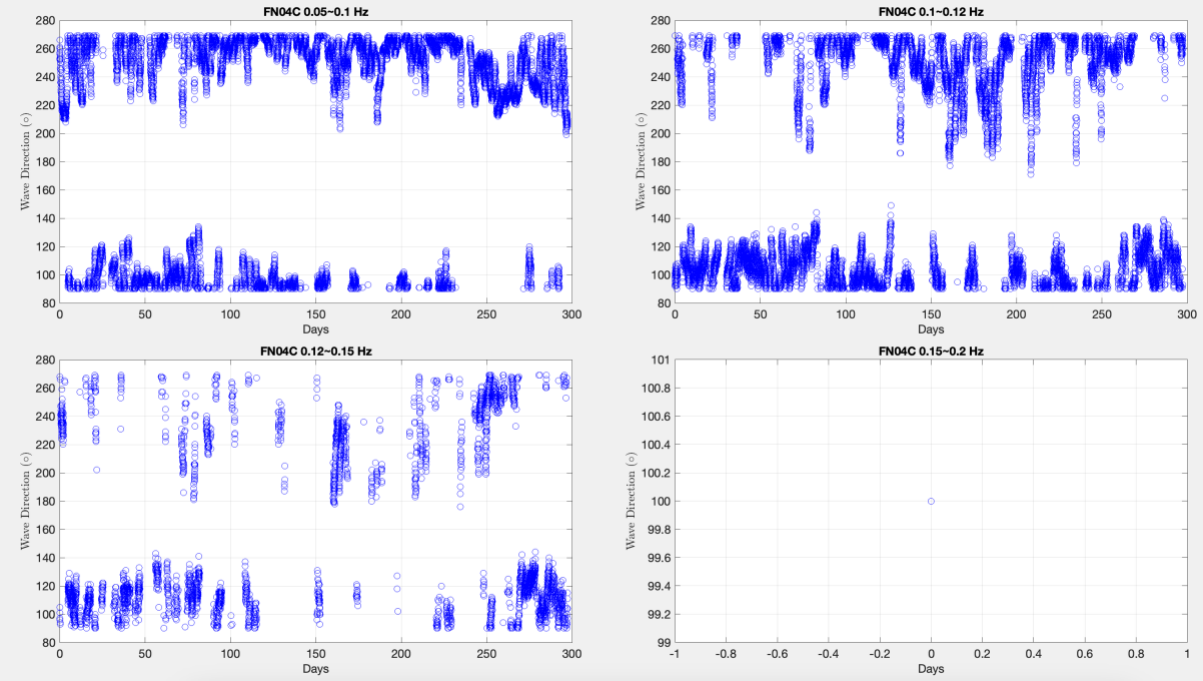
The script simply modifies the frequency limits in *obsnan.ctl* and calls *obsnan.py*. After the calculation is finished, there will be a series of folders containing the HPTF results, e.g, *HPTFResults_Freqf1_f2_TimeLength*.

In each result folder, there is a subfolder named *HPTF_STA*, where all the HPTFs calculated for data segments with high coherence are saved. There is a file named *HPTF_STA.dat*, saving the starting and ending time, coherence coefficient between the horizontal acceleration and dpdt, direction of water waves, and the index of each data segment. There is a file named *HPTF_STA_Selected.dat*, which is the same as *HPTF_STA.dat* but removing data segments of low coherence. These two files are actually the same, because HPTFs for data segments of low coherence are not calculated at all. There is a file named *HPTF_STA_Selected_Calibrated.dat*, which has an extra column with value of either 0 or -180, indicating the calibration angle for the propagation direction of water waves. (STA = StationName)

6.2 Calibrating Wave Direction

Now we go to folder **MatlabTools** and use script **plotWaveDirectionAllFrequencies.m** to view the propagation direction of water waves at each station. For example, in Matlab command window, use command *for i=1:3;plotWaveDirectionAllFrequencies(i);input('');end*. Similar to the calculation of tilt angles, it is possible that at some stations the angle is consistent over time, while at other stations it splits into two branches. An example is shown below.





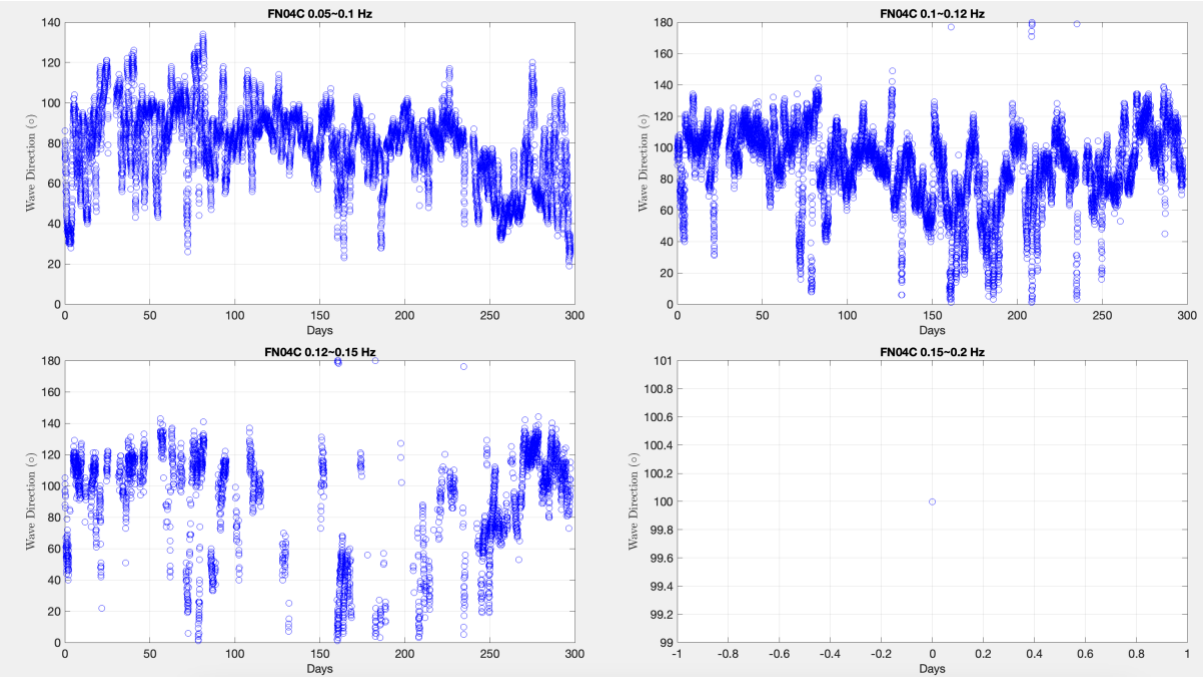
For the second station, we have to calibrate the wave direction so that it is also consistent over time. To do that, we create a file named **StationsHPTFWaveDirectionRange.ctl** in the main path, and specify an option of 2 at station FN04C.

```
1 2 XX00    # 1: Original; 2: if > 180, -180
2
3 2 FN02C
4 2 FN04C
5
```

Then we change the parameters in **obsnan.ctl** (Line 59-62 and Line 74), and use **hptfCalculateAllFrequencies.py** or **obsnan.py** to do the calculation again.

```
55 ================================================:============================
56 HPTF Calculation Flow and Parameters    (Units) :  Values                     |
57 ================================================:============================
58
59 Filter Noise Data for HPTF    (1/y/yes, 0/n/no) :  no
60 Find Horizontal PTF for All Segments            :  no
61 Remove Bad Correlation for HPTF                 :  0
62 Calibrate Wave Direction                        :  1
63
64 Frequency Range to Find Horizontal PTF     (Hz) :  0.15, 0.2
65 Time Length to Determine Horizontal PTF(second) :  2000
66 Min Value to Determine dpdt/A Correlation       :  0.80
67
68 For Denoising: Average HPTF near Days  (+-days) :  2
69 For Denoising: Fit HPTF with Polynomial Order   :  1
70 For Denoising: Fit HPTF in Frequency Range      :  0.05,0.12
71 For Denoising: Use Wave Direction In Freq Range :  0.05, 0.1
72 For Denoising: Average Wave Direction (+-hours) :  2
73
74 Overwrite if Results exist HPTF(1/y/yes, 0/n/no):  yes
```

The calculation updates the calibration angles in **HPTF_STA_Selected_Calibrated.dat**. Now in folder **MatlabTools** re-run plotWaveDirectionAllFrequencies, and we will see that the angles are calibrated.



6.3 Averaging and Combining HPTFs in different frequency bands

Because HPTFs may change significantly over time, so we do not average the HPTFs over time but save all the HPTFs for each time segment. When removing noise, we will look for HPTFs that are close to the earthquake time to do the average. Thus, the following parameters in *obsnan.ctl* (Line 68-72) are used in the denoising process instead of calculating HPTFs. Averaging and combining HPTFs are also done in the denoising process.

```
67
68 For Denoising: Average HPTF near Days  (+-days) :  2
69 For Denoising: Fit HPTF with Polynomial Order   :  1
70 For Denoising: Fit HPTF in Frequency Range      :  0.05,0.12
71 For Denoising: Use Wave Direction In Freq Range :  0.05, 0.1
72 For Denoising: Average Wave Direction (+-hours) :  2
73
```

7 Removal of Noise

Provide SAC files in folder EventXXXX, and the program will remove the noise for these SAC files. For example, at station WD52, WD55 and FN03C, data segments of length of about 5,000 s are cut for a few earthquakes, and stored in folder EventTest.

```
Chao-MacBook-Pro:EventTest ac$ ls *-1.SAC *-2.SAC *-3.SAC
FN03C.7D..HDH.2013.11.17.08.48.15-2013.11.17.09.54.55-3.SAC WD52.XO..HDH.2018.07.25.22.35.33-2018.07.26.00.07.13-1.SAC
FN03C.7D..HDH.2014.04.01.23.30.07-2014.04.02.00.36.47-1.SAC WD52.XO..HDH.2018.07.25.23.34.26-2018.07.26.01.06.06-2.SAC
FN03C.7D..HDH.2014.04.03.02.26.33-2014.04.03.03.33.13-2.SAC WD52.XO..HDH.2018.08.12.19.33.14-2018.08.12.21.04.54-3.SAC
FN03C.7D..HH1.2013.11.17.08.48.15-2013.11.17.09.54.55-3.SAC WD52.XO..HH1.2018.07.25.22.35.33-2018.07.26.00.07.13-1.SAC
FN03C.7D..HH1.2014.04.01.23.30.07-2014.04.02.00.36.47-1.SAC WD52.XO..HH1.2018.07.25.23.34.26-2018.07.26.01.06.06-2.SAC
FN03C.7D..HH1.2014.04.03.02.26.33-2014.04.03.03.33.13-2.SAC WD52.XO..HH1.2018.08.12.19.33.14-2018.08.12.21.04.54-3.SAC
FN03C.7D..HH2.2013.11.17.08.48.15-2013.11.17.09.54.55-3.SAC WD52.XO..HH2.2018.07.25.22.35.33-2018.07.26.00.07.13-1.SAC
FN03C.7D..HH2.2014.04.01.23.30.07-2014.04.02.00.36.47-1.SAC WD52.XO..HH2.2018.07.25.23.34.26-2018.07.26.01.06.06-2.SAC
FN03C.7D..HH2.2014.04.03.02.26.33-2014.04.03.03.33.13-2.SAC WD52.XO..HH2.2018.08.12.19.33.14-2018.08.12.21.04.54-3.SAC
FN03C.7D..HHZ.2013.11.17.08.48.15-2013.11.17.09.54.55-3.SAC WD52.XO..HHZ.2018.07.25.22.35.33-2018.07.26.00.07.13-1.SAC
FN03C.7D..HHZ.2014.04.01.23.30.07-2014.04.02.00.36.47-1.SAC WD52.XO..HHZ.2018.07.25.23.34.26-2018.07.26.01.06.06-2.SAC
FN03C.7D..HHZ.2014.04.03.02.26.33-2014.04.03.03.33.13-2.SAC WD52.XO..HHZ.2018.08.12.19.33.14-2018.08.12.21.04.54-3.SAC
```

Stations in different water depths have different types of noise. For deep water (>300 m), try to remove tilt and compliance noise in the vertical channel, and rotate the horizontal channels to suppress the noise. For shallow water (<300 m), try to remove tilt and compliance noise in the vertical channel (tilt noise can be very negligible), and remove water-wave noise in the horizontal channels around 0.07 Hz.

7.1 Deep Water Stations

WD52 and WD55 are stations in deep water (water depth 2563.6 m and 1283.5 m respectively)**.** Change parameters in **obsnan.ctl**, and run **obsnan.py**.

```
103
104 ================================================:==========================
105 Event Data Noise Removal Parameters     (Units) :  Values                   |
106 ================================================:==========================
107
108 Remove Noise for Event Data   (1/y/yes, 0/n/no) :  1
109 Remove Noise for Vertical Channel Option        :  2 #0/no; 1/only tilt; 2/tilt+pressure
110 Remove Noise for Horizontal Channel Option      :  1 #0/no; 1/rotate current noise; 2/wave noise
111 Frequency Range for Event Data Processing       :  0.01, 0.02
```

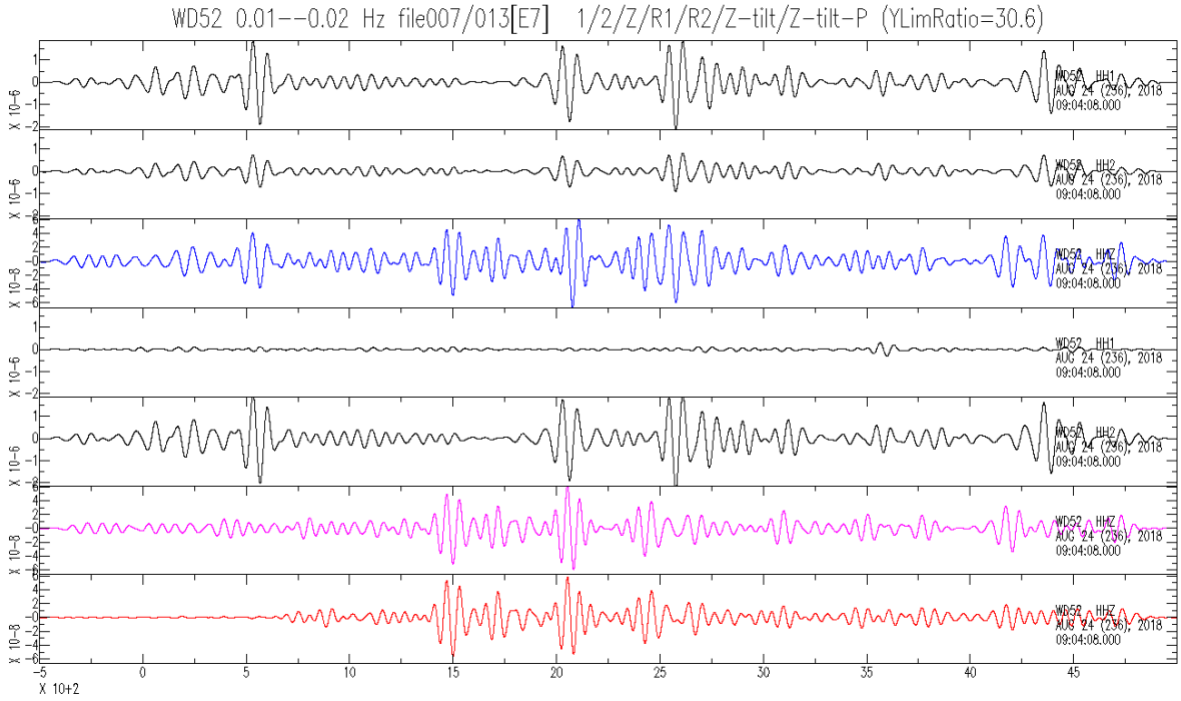
```
--> Remove noise of event data ...

    Identify event data folders ...
        /Users/ac/work/OceanBottomSeismicData/codes/CodesToPost_AddHPTF/EventTest

    Processing event data ...

        (1/1) /Users/ac/work/OceanBottomSeismicData/codes/CodesToPost_AddHPTF/EventTest:

                Identify SAC files in the folder [3 Stations; 36 Event Pairs]                  [DONE]
                Filter event data (OldName_filt) [0.01, 0.02 Hz]                               [DONE]
                Remove tilt noise (OldName_filt_TR)                                            [DONE]
                Remove pressure noise (OldName_filt_TR_PR)                                     [DONE]
                Rotate horizontal data (OldName_filt_RT)                                       [DONE]


--> Complete.
```

The program removes the tilt and pressure noise, and generates new SAC files named *****.SAC_filt*, *****.SAC_filt_TR*, *****.SAC_filt _TR_PR* and *****.SAC_filt _RT*, which are

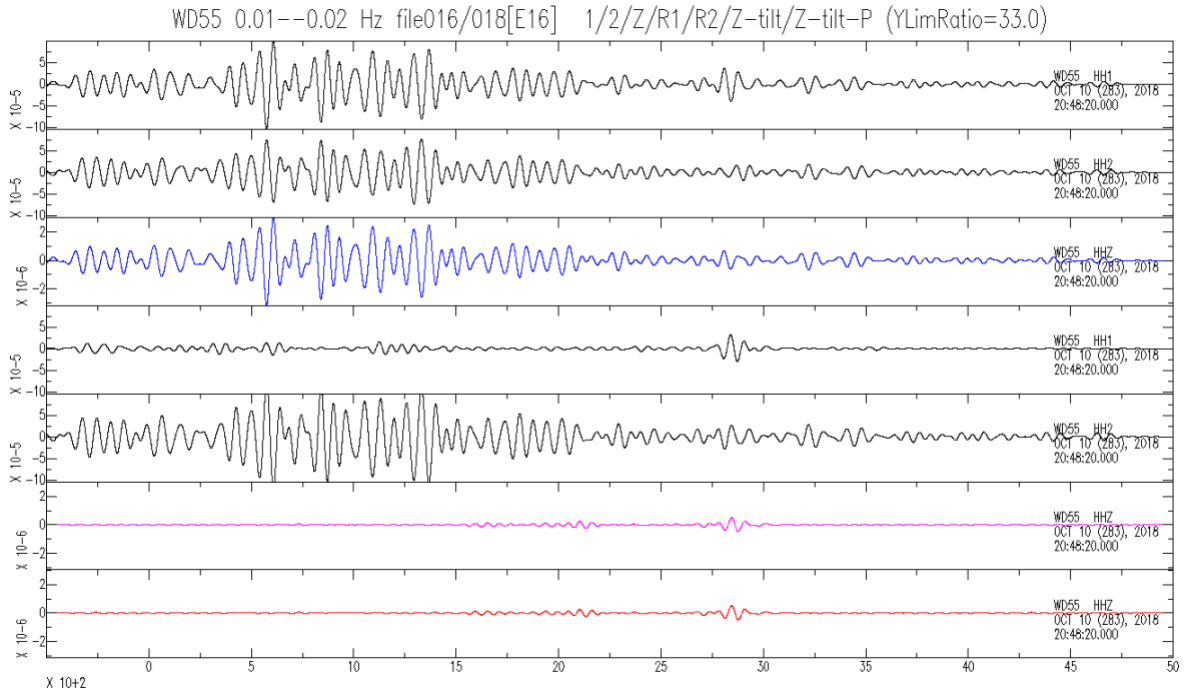
vertical channels filtered, vertical channels with tilt noise removed, vertical channels with both tilt and pressure compliance noise removed, and horizontal channels rotated to the current direction. Here TR stands for “Tilt Removed”, PR for “Pressure Removed” and RT for “Rotated”.

The program also generates two SAC MACRO files,
sacCommandToViewResults_EventTest.macro and
sacCommandToViewResults_EventTest.macro_SACInput
in the main path to view the results.

It is expected that the noise reduction is significant in the vertical channel:



At some stations, the noise reduction of the horizontal channels might also be helpful:



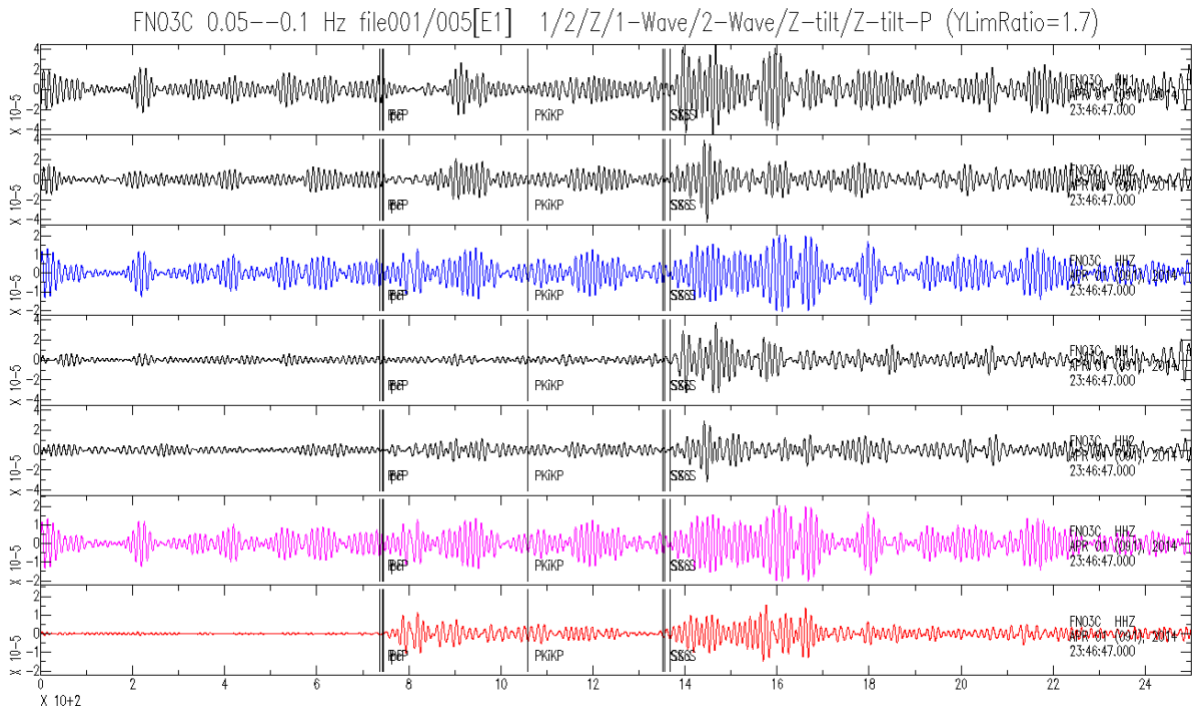
7.2 Shallow Water Stations

FN03C is a station in shallow water (93 m). The noise reduction is more significant around 0.07 Hz. The horizontal noise can be reduced using the HPTFs. Change parameters in **obsnan.ctl** and run **obsnan.py**. Note that HPTFs are not calculated for stations in deep water (WD52 and WD55), so I temporally removed SAC files of WD52 and WD55 in *EventTest*.

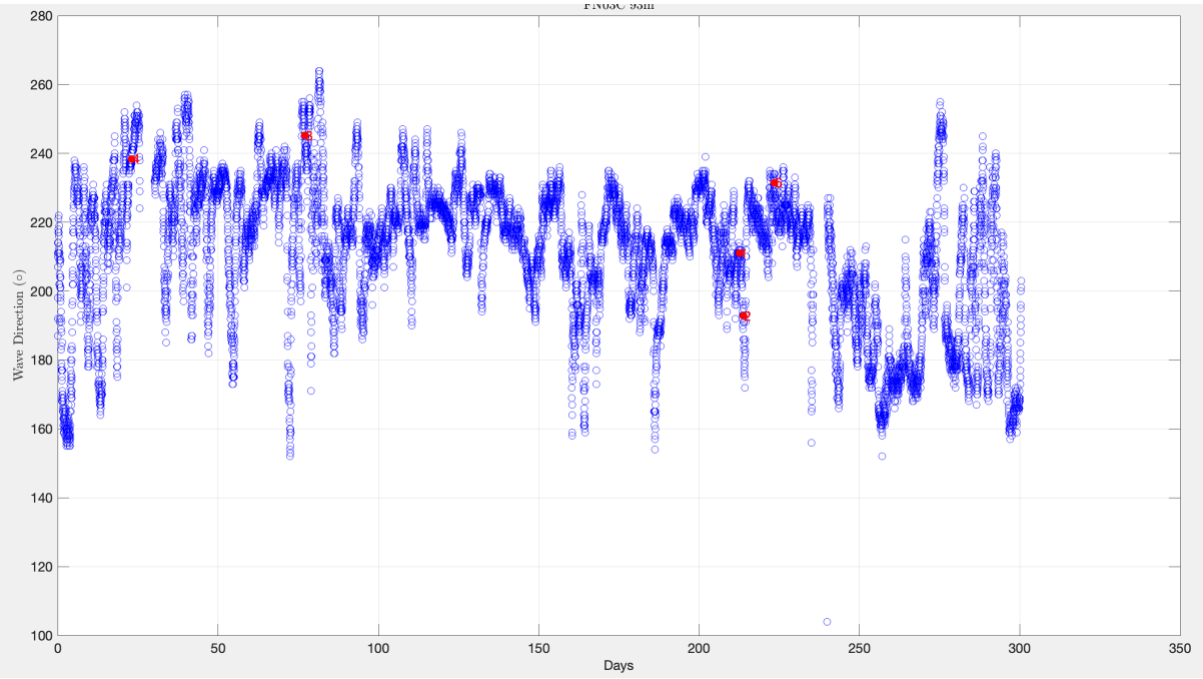
```
103
104 ================================================:=========================
105 Event Data Noise Removal Parameters     (Units) :  Values                  |
106 ================================================:=========================
107
108 Remove Noise for Event Data   (1/y/yes, 0/n/no) :  1
109 Remove Noise for Vertical Channel Option        :  2 #0/no; 1/only tilt; 2/tilt+pressure
110 Remove Noise for Horizontal Channel Option      :  2 #0/no; 1/rotate current noise; 2/wave noise
111 Frequency Range for Event Data Processing      :  0.05, 0.1
```

```
--> Remove noise of event data ...

    Identify event data folders ...
        /Users/ac/work/OceanBottomSeismicData/codes/CodesToPost_AddHPTF/EventTest

    Processing event data ...

        (1/1) /Users/ac/work/OceanBottomSeismicData/codes/CodesToPost_AddHPTF/EventTest:

                Identify SAC files in the folder [1 Stations; 5 Event Pairs]          [DONE]
                Filter event data (OldName_filt) [0.05, 0.1 Hz]                       [DONE]
                Remove tilt noise (OldName_filt_TR)                                   [DONE]
                Remove pressure noise (OldName_filt_TR_PR)                            [DONE]
                Remove horizontal wave noise (OldName_filt_WV)                        [DONE]


--> Complete.
```

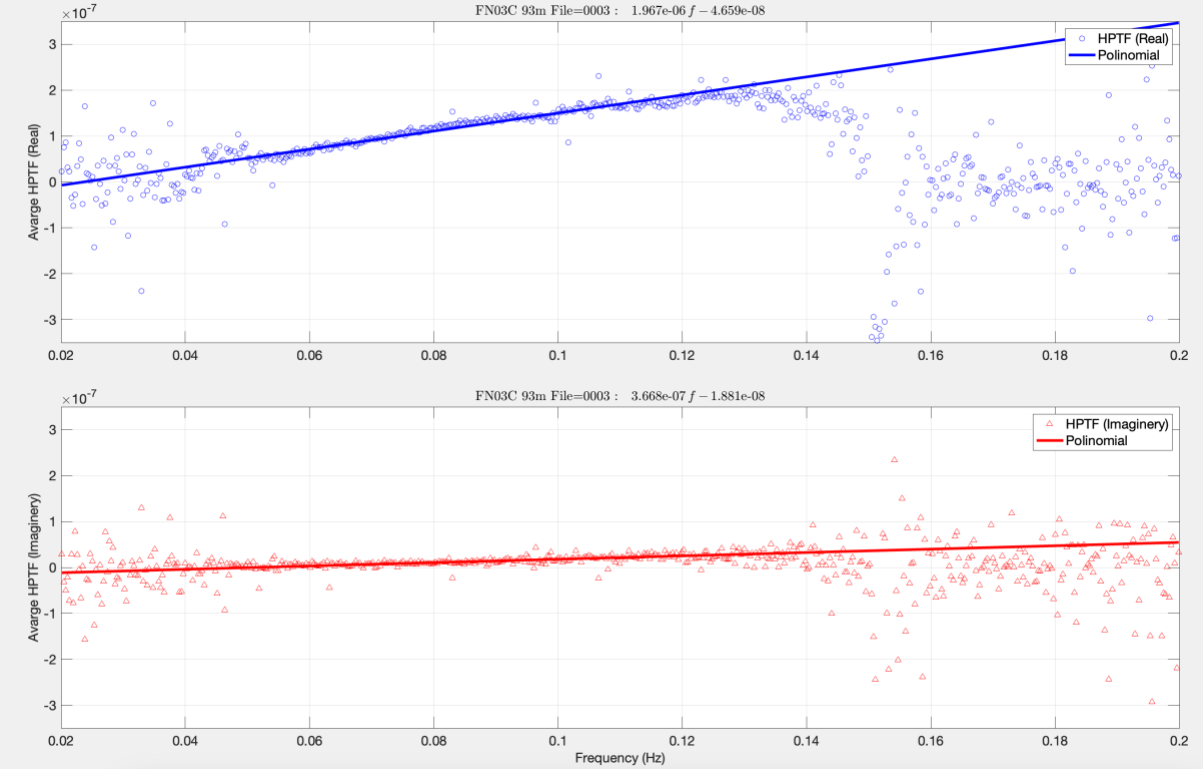
New horizontal files are ended by “WV” meaning water-wave noise is removed.



Now in MatlabTools you can use **plotWaveDirectionUsedForEvent.m** and **plotHPTFUsedForEvent.m** to view the wave direction and HPTF used to remove the noise. For example, in Matlab command window: *plotHPTFUsedForEvent(1)* shows



plotHPTFUsedForEvent(1,'file',3,'poly') or *plotHPTFUsedForEvent('fn03c','file',3,'poly')* shows



Note that if the HPTFs are not satisfactory, it is necessary to change the frequency range for polynomial fitting. Change parameters in **obsnan.ctl** (Line 70), and re-run **obsnan.py** to remove the noise.

References

Crawford, W. C., & Webb, S. C. (2000). Identifying and removing tilt noise from low-frequency (< 0.1 Hz) seafloor vertical seismic data. Bulletin of the Seismological Society of America, 90(4), 952-963.

Bell, S. W., Forsyth, D. W., & Ruan, Y. (2015). Removing noise from the vertical component records of ocean-bottom seismometers: Results from year one of the Cascadia Initiative. *Bulletin of the Seismological Society of America*, *105*(1), 300-313.

An, C., Wei, S., Cai, C., & Yue, H. (2020). Frequency Limit for the Pressure Compliance Correction of Ocean-Bottom Seismic Data. *Seismological Research Letters*, *91*(2A), 967-976.

An, C., Cai, C., Zhou, L., & Yang, T. (2022). Characteristics of Low-frequency Horizontal Noise of Ocean-Bottom Seismic Data . *Seismological Research Letters*, 93(1), 257-267.

Zhang, C., & An, C. (2024). Water-wave-induced OBS noise: theories, observations and potential applications. Journal of Geophysical Research: Solid Earth, 129(4), e2023JB027787.