EGF time correction toolbox

Codes for estimating the Green's function and measuring timing errors using ambient seismic noise

Karina Løviknes

1 Introduction

The codes presented here were initially developed as a part of the master thesis Løviknes (2019) where the main objective was to create a tool that estimates the Green's function from ambient seismic noise and use the Green's function to measure instrumental timing errors in seismic data. These codes have been rewritten to functions and are put together as toolbox. This manual explains how to use this toolbox.

The toolbox contains two main functions, one for estimating the Green's function and one for measuring time shifts. In addition, sub functions for preparing, processing and cross correlating the input data, and side functions for analysing, inverting and plotting the results, are included.

Contents

2 Main functions

The main functions of the toolbox estimates the Green's function and measures timing errors. These functions use several sub-functions for preparing, processing and cross correlating the input data. Additional side functions are included for analysing, inverting and plotting the results. The input values are all defined and can be changed in a settings text file, it is therefore only necessary to specify the name of the settings file as input to the main functions. The flowchart in figure 1 shows the main workflow starting from the settings file to estimating the Green's function and using the Green's function to measure time shifts and inversion to find the final time shift.

Most of the functions are written in MATLAB, but some are also written with SAC and Linux shell. See list 5 in appendix for overview of the functions.

2.1 Settings file

All input values are defined in a text file, here called settings file. The values are divided into different categories for specifying the stations and time period, for estimating the Green's function, for measuring time delays, for inverting and for plotting in different ways. Some of the values are mandatory, while others are optional. The following list gives all the possible input values, the mandatory values are indicated with (m) behind the value name:

- Station and time period specifications
  - **network:** (m) network name
  - **stations:** (m) list of names of the stations to be cross correlated, must include at-least two stations
  - **channels:** (m) channel code item **location:** (m) location code

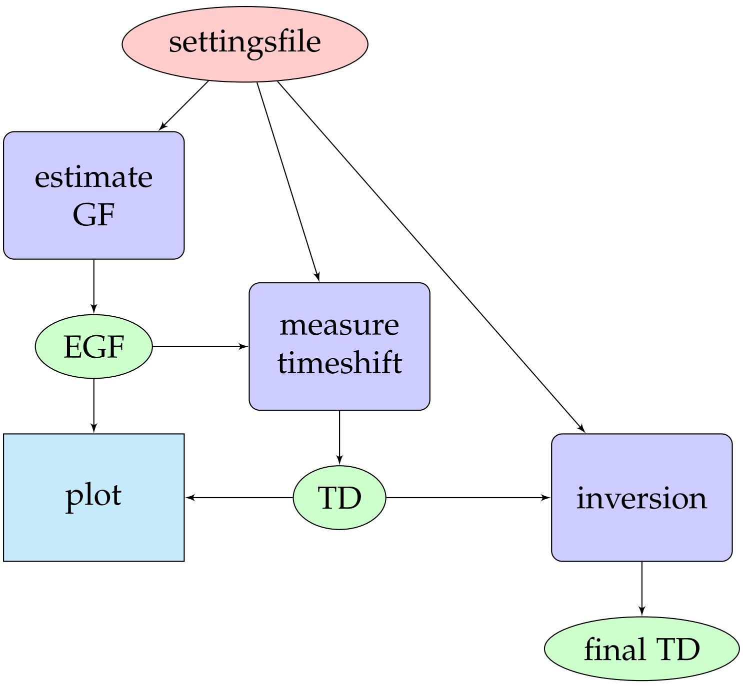

Figure 1: Flowchart showing the main workflow for estimating the Green's function from ambient seismic noise and measuring timing errors using the estimated Green's function

   - **first_day:** (m) must be in the format 'yyyy-mm-dd'
   - **last_day:** (m) must be in the format 'yyyy-mm-dd'
   - **num_stat_cc:** (m) number of stations each station is cross correlated with
   - **Fq:** (m) sampling frequency

- Values for estimating the Green's function ('EGF')
   - **filename:** must contain station name, date and folder (if in another folder), the default is [network.stationname.location.HHchannels.D.yyyy.ddd.SAC]
   - **fileformat:** can either be sac (default) or miniseed
   - **pz_file:** pole-zero file, the default format is [SAC PZs NS stationname HHchannels.pz]
   - **dateformat:** the date format used in the filename, default is 'yyyy-mm-dd'
   - **deci:** specifies if the data should be downsampled and by how much, if not leave blank or out
   - **bpf:** (m) bandpass filter to apply during preprocessing
   - **norm:** normalization specification, default is onebit before spectral whitening, the options are 'onebit' for only onebit normalization, 'sw' for spectral whitening before onebit, and 'onebit sw' for default
   - **wl:** cross correlation window length, wl must be given as an integer between 1 and 24 hours, the default is 24 hours, when wl 24 the signals are cut into smaller time windows before cross correlation and stacked together after, this is done to save time
   - **swl:** stack window length, the number of cross correlation windows to be stacked for output, swl must be given as an integer between 1 and 24 hours, the default is 24 hours

- Values for measuring time delay ('TD')
   - **bpfm:** bandpass filter to apply on the daily noise correlations and reference trace before measuring time delays
   - **iterations:** specify the number of iterations the measuring process should be run, default is 3, must be an integer
   - **lag_red:** specify over what time lag the measuring process should be run, default is 2000
   - **stackperiod:** specification about the time period the reference is stacked over. The options are 'whole' for whole time period (default), 'monthly' for

each month, where the first and last day of the correlation period must be specified and 'days' or 'firstdays' over a specified number of days, number of days must be specified

- Values for inversion ('INVERT')
  - **xaxis:** x-axis for plotting the cross correlations, units in seconds, default is [-150 150]
  - **reference_clock_station** station with reliable clock to use as reference during the inversion
- Values for plotting ('PLOT')
  - **xaxis:** x-axis for plotting the cross correlations, units in seconds, default is [-150 150]
  - **yaxis**: y-axis for plotting the measured time delays, units in seconds, default is [-100 100]
  - **bpfp:** bandpass filter to apply before plotting, when left out the signals are not filtered before plotting
  - **lag_red:** specify over what time lag the measuring process should be run, default is 2000
- Values for filtering and comparing the signal over various frequency bands ('FILTER')
  - **xaxis:** x-axis for plotting the cross correlations, units in seconds, default is [-150 150]
  - **lag_red:** specify over what time lag the measuring process should be run, default is 2000
  - **cutoff_freq1**: lower cut off frequencies, if more than one value the signal will subsequently be filtered over the different frequency bands.
  - **cutoff_freq2**: upper cut off frequencies, should contain as many values as **cutoff_freq1**
- Values for calculating and plotting with distance ('DISTANCE')
  - **xaxis:** x-axis for plotting the cross correlations, units in seconds, default is [-150 150]
  - **pz_file:** pole-zero file containing the coordinates of the station, if not spesified the default format is [SAC PZs NS stationname HHchannels.pz]
  - **dateformat:** the date format used in the file name, default is 'yyyy-mm-dd'

- **lag_red:** specifies over what time lag the measuring process should be run, default is 2000

The values are read into MATLAB using the function *read_settings*. The inputs of this function are the name of the settings file and an optional specification about the type of values to be read. These options are 'EGF' for reading values for estimating the Green's function, 'TD' for reading values for measuring time shifts, 'PLOT' for reading plotting values, 'INVERT' for reading values for inversion, 'FILTER' for reading values for applying bandpass filters and 'DISTANCE' for values used to calculate and plot with distance between the stations.

The default values for the example stations are given in the text file *settings.txt* (figure 8).

2.1.1 File format

The actual files containing the raw data are loaded from within the main functions based on station codes and dates. The file format must be either SAC or miniseed and specified in the settingsfile. The format must be the same for all the stations. The filename can be of any structure, but must contain the station name and dates. If the files are located in another folder than the functions, the folder name must also be included in the filename. The date should be specified as year 'yyyy' and day of the year 'ddd', or (DATE) with an optional date format specification. The date format must be valid for the inbuilt MATLAB function *datestr* (see figure 2), default is 'yyyy-mm-dd'.

Numeric Identifier	Date and Time Format
-1 (default)	'dd-mmm-yyyy HH:MM:SS' or 'dd-mmm-yyyy' if 'HH:MM:SS'= 00:00:00
0	'dd-mmm-yyyy HH:MM:SS'
1	'dd-mmm-yyyy'
2	'mm/dd/yy'
3	'mmm'
4	'm'
5	'mm'
6	'mm/dd'
7	'dd'
8	'ddd'
9	'd'
10	'yyyy'
11	'yy'
12	'mmmyy'
13	'HH:MM:SS'
14	'HH:MM:SS PM'
15	'HH:MM'
16	'HH:MM PM'
17	'QQ-YY'
18	'QQ'
19	'dd/mm'
20	'dd/mm/yy'
21	'mmm.dd,yyyy HH:MM:SS'
22	'mmm.dd,yyyy'
23	'mm/dd/yyyy'
24	'dd/mm/yyyy'
25	'yy/mm/dd'
26	'yyyy/mm/dd'
27	'QQ-YYYY'
28	'mmmyyyy'
29	'yyyy-mm-dd'
30	'yyyymmddTHHMMSS'

Figure 2: Dateformat for MATLAB function *datestr*

Examples of filename for a station SSSS of network NN and channel C on day ddd of year yyyy and on month mm, day dd and year yyyy:

filename = [folder/NN.SSSS.00.HHC.D.yyyy.ddd.000000.SAC]

filename = [folder/NN.SSSS.00.HHC.D.(DATE).mseed] dateformat='yyyy.mm.dd'

Both when using miniseed and SAC files the files should be organised as daily files. If there are more than one file for each day, the files can be merged using the Linux bash function *preparefiles.sh*. This function merge the files into one day using SAC. The function can also be used to downsample the files and convert between SAC and miniseed format. When running the *preparefiles.sh*- function, the settings file is not taken as input, but asks the user for input file format, network, station names, dates and whether to downsample or not and with how much. The functions permanently alters the original files, and the files and functions should therefore be copied into a new folder before executing.

2.2 estimate_GF

The function *estimate_GF* is used to estimate the Green's function from daily noise recordings. The input is the name of the settings file and the output is a struct giving the daily cross correlations, the time lag, the number of days and the name of the station pair. The estimated Green's functions are also saved as mat-files (daily EGF) and SAC-files (stacked EGF) with filenames:

[Egf_*pair_dates*.mat] [Egf_*pair_dates*.SAC]

The *estimate GF* function use the functions *read_daily*, *prepros* and *cross_conv* for retrieving, preprocessing and cross correlate the data, respectively (figure 3). In addition, the functions *gen_generate* is used to generate a response file, from a pole-zero file, for removal of instrument response in the *prepros* function, and the inbuilt MATLAB function *designfilt* is used to design a 4th order zero-phase Butterworth filter to be applied in the *prepros* function. The default values and inputs of these functions are specified in the settings file: *settings.txt*.

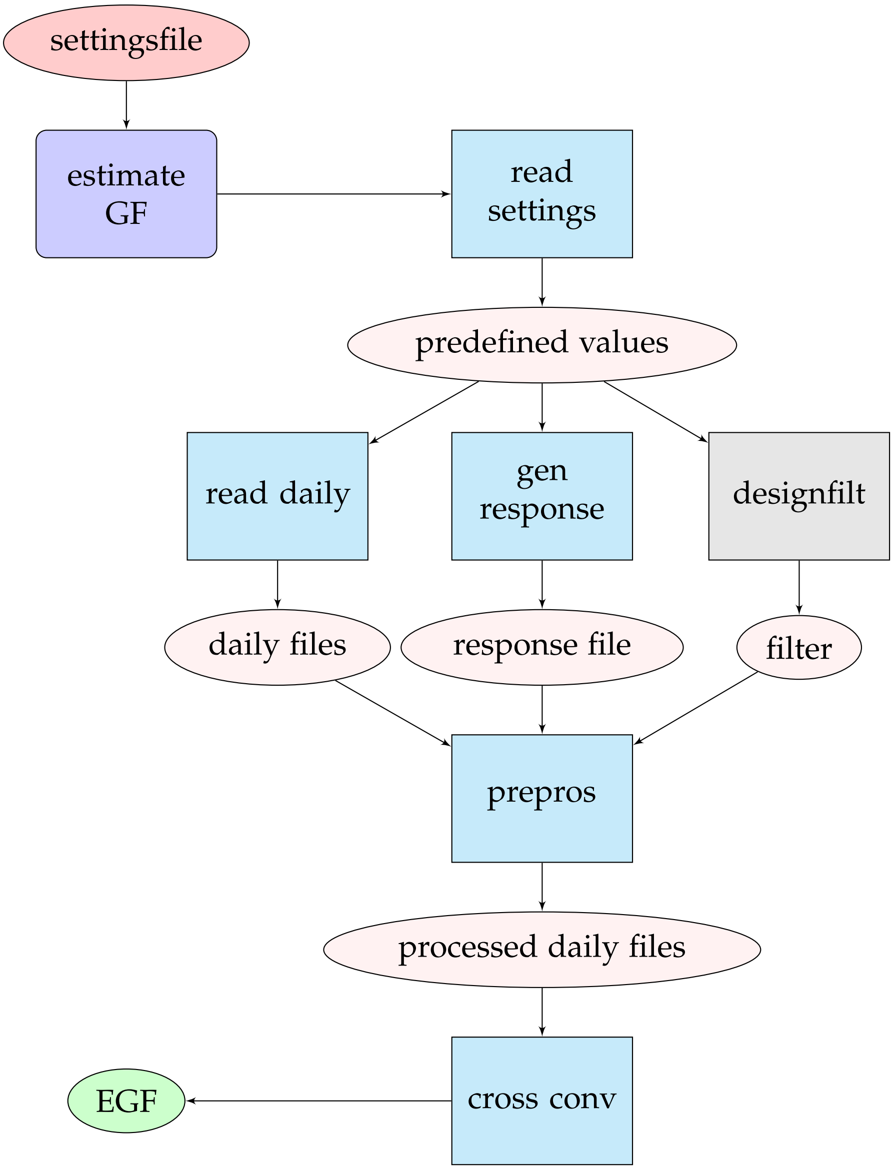

Figure 3: Flowchart.

2.2.1 read_daily

The *read_daily* function loads the daily data from sac or miniseed files and checks that the given sampling and timing is correct. The inputs are the station network, name, channels and location code, a vector containing all the dates, the file name and format, the date format, sampling frequency and a downsampling specification. The inputs are defined in the settingsfile and/or created in the *estimate_GF* function. The output is the raw daily data. In addition a text-file with information about the each day is created.

The file format can be either sac or miniseed. The function *rdsac* is used to read SAC-files, while *ReadMSEEDFast* reads mseed-files. Both functions were originally developed by others (https://se.mathworks.com/matlabcentral/fileexchange/46356-rdsac-and-mksac-read-and-write-sac-seismic-data-file and https://se.mathworks.com/matlabcentral/fileexchange/46532-readmseedfast-filename), and must be downloaded and placed in the toolbox before the functions can be used.

The filename format is specified in the settings file, see section 2.1.1, and the function *str2filename* is used to create the actual filenames. *str2filename* takes the filename format, station name, dates and other optional variables as input and insert these into the filename. The output is the actual filename.

Warning messages The *read_daily* function reads through the daily files and verifies that the sampling frequency and timing are correct. If the start time in hours, minutes and seconds (HH:MM:SS.SSS)is not exactly 00:00:00.000, zeroes are padded in the beginning of the trace and a warning message is given. The function then checks the end time of the daily file, and equally pads with zeroes and exerts a warning message if the end time is not correct. The functions also checks that the sampling frequency given in the settings file is the same as the sampling frequency given in the file header, and decimates the data if requested in the settings file. In addition to being executed directly in the command window, the warning messages are saved in a text-file together with other information about the daily files. The text-file is saved as ['stationname' - 'firstday' - 'lastday' .txt].

2.2.2 prepros

The *prepros* function consists of several preprocessing steps (figure 4). The inputs are the raw data to be processed, sampling frequency, filter designed using the inbuilt MATLAB function *designfilter*, a pole-zero file and an optional specification about the normalization processes. The different normalization options available are onebit normalization before spectral whitening (default), only onebit normalization or spectral whitening before onebit normalization. Other options can be added. The output of the *prepros* function is the processed data ready for cross correlation.

The different steps of the *prepros* function can be seen in figure 4. The mean and trend of the raw data is first removed using the inbuilt MATLAB function *detrend*, before a cosine taper is applied using the function *costap_filter*, where the ends for the signal are smoothly decayed to zero. The function *rm_iresp* then removes the input instrument response from the signal using deconvolution. The inbuilt *filtfilt* function is used to filter the signal, both before and after the removal of instrument response. The *filtfilt* function filters the input data in both directions, and therefore does not alter the phase of the signal. The signal is normalized in the time domain using onebit normalization and in the frequency domain using spectral whitening. Onebit normalization sets all positive amplitudes to 1 and all negative amplitudes to -1, this is done by dividing the signal with its absolute value. The spectral whitening is done using the function *spectral_whitening*, where the amplitude spectrum is flattened by dividing the Fourier transform of the signal by a smoothed version of the amplitude spectrum.

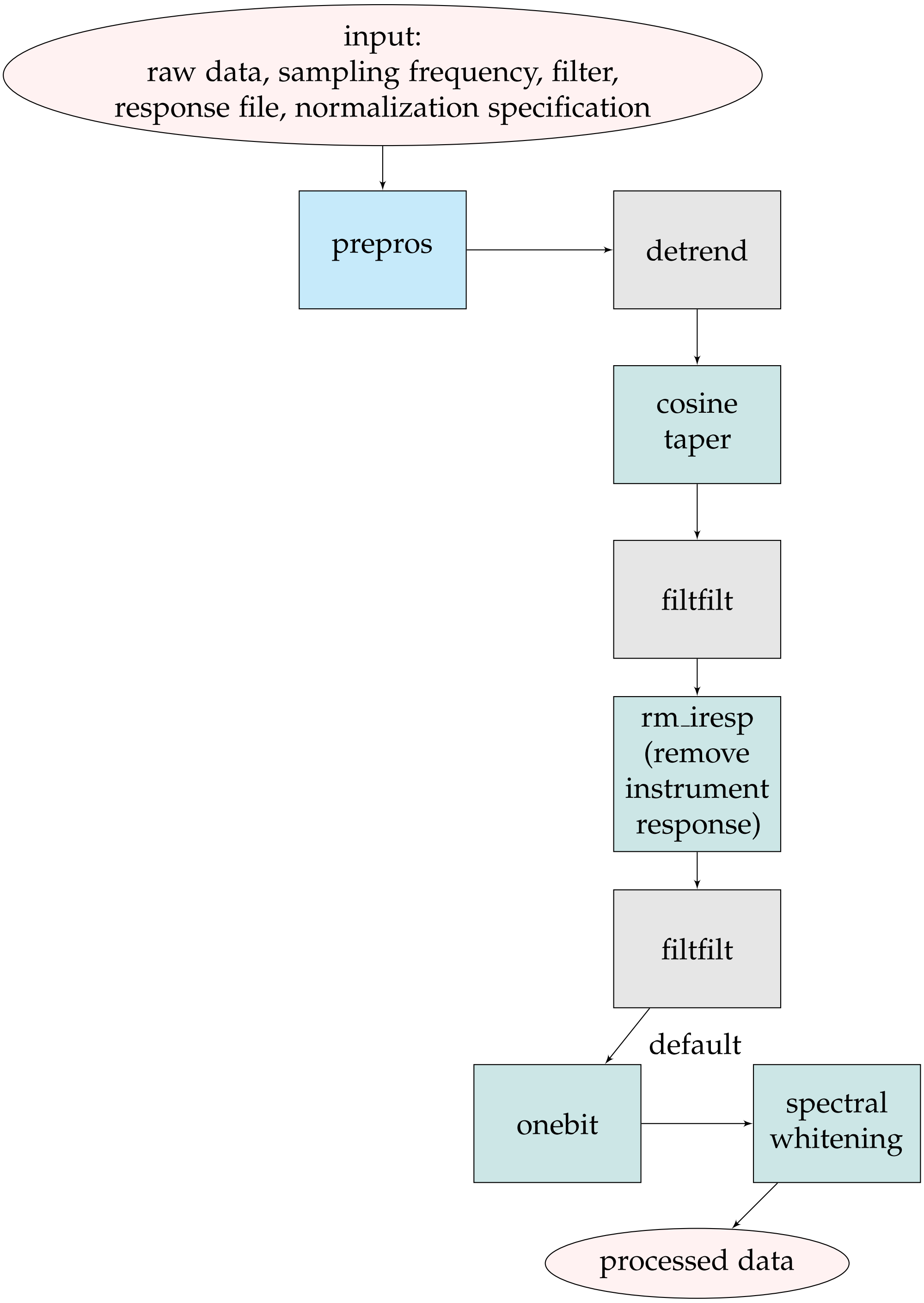

Figure 4: Flowchart.

2.2.3 cross_conv

After pre-processing, the daily traces are cross correlated and stacked. The cross correlation is done using the *cross_conv* function where the first of the two traces is flipped right to left in time domain, before both traces are convolved together This is possible since the only difference between cross correlation and convolution is the sign of the time shift. The convolution is done in time domain using the inbuilt MATLAB function *conv*.

The inputs of the *cross_conv* function are the two segments to be cross correlated, the sampling frequency and the optional time window length (wl) in hours for cross correlation over shorter time windows and a stacking time window length (swl) for stacking the shorter time windows. Cross correlating over shorter time windows is mainly done to save time. After cross correlation, the shorter time windows are stacked together using the inbuilt MATLAB function *sum*. The shorter time windows are either stacked over 24 hours (default), or a shorter time period if specified. The length of the cross correlation and stacking time windows are specified in the settings file.

2.3 measure_timeshift

The function *measure_timeshift* measures the timeshift of the Green's function by comparing daily cross correlations to a reference trace. The input is the name of the settings file and the output is a struct giving the found daily time delays, the name of the station pair, the number of days and information about how the timeshift is measured

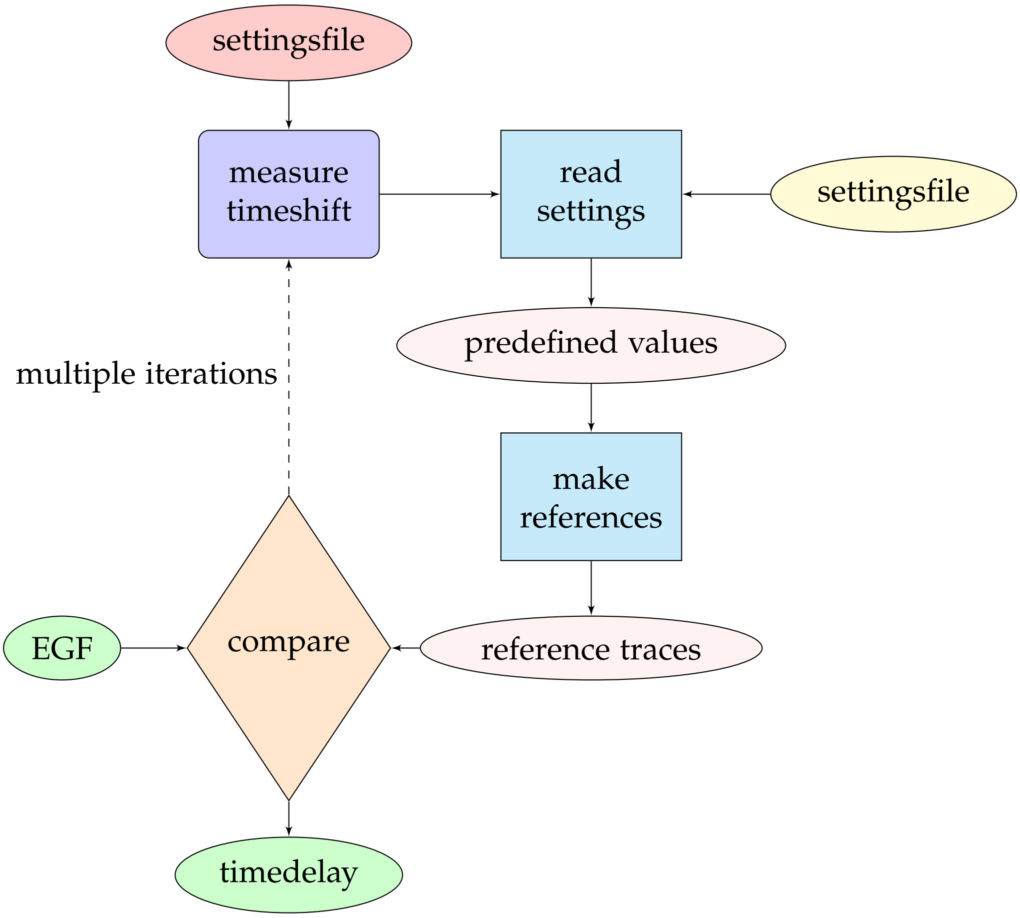

Figure 5: Flowchart.

To find time errors specifically caused by instruments, the time delay is measured for the causal and acausal part of the signal separately. The time delay of the negative side $d^-(t)$ and positive side $d^+(t)$ are further compared and only used when following the condition:

$$d^+(t) \approx -d^-(t) \tag{2.1}$$

The final time delay d(t) is set as the average of the time shift found for each side, following the condition of equation 2.1.

$$d(t) = \frac{d^+(t) + (-d^-(t))}{2} \tag{2.2}$$

The daily correlations are compared to the reference using cross correlation. To ensure that the daily cross correlations have converged towards the Green's functions, only the comparing cross correlations with correlation coefficient above a certain threshold are used to measure the time shift. The correlation coefficient is found by normalizing the cross correlation using the autocorrelations:

$$\overline{d^+(t)} = \frac{d^+(t)}{max(\sqrt((r^+(t) * r^+(t))(s^+(t) * s^+(t))))} \tag{2.3}$$

This threshold is set to 0.4 following Sens-Schönfelder (2008). This is a relatively arbitrary constant, mainly chosen as an extra step to ensure quality measurements.

The time shift is measured as the time lag where the comparing correlation is maximum.

$$d(t) = max(r(t) * s(t)) \tag{2.4}$$

Due to uneven noise directivity, some noise correlations have much stronger amplitudes on one side compared to the other. Such ″one-sided″ signals are used when the correlation coefficient of the signal side is above 0.4 while below 0.4 on the non-signal side. In this case the conditions of equation 2.1 is disregarded. If the correlation coefficient of both sides of the signal are below the threshold, the time shift can be measured over the entire signal as one, if it's correlation coefficient is above the threshold. If no parts of the signal gives a correlation coefficient above the threshold the time shift is either set to NaN (not a number) or equal the time shift of the previous day. This gives two different measured time shifts, where the vector ″timedelay″ gives a continuous time shift and the vector ″timedelay0″ gives the actual measured time shifts. During the measuring process a vector giving information about whether the time shift is measured over the separated sides ″s″, positive side ″p″, negative side ″n″, the whole signal ″w″ or not measured ″0″, is created. This vector is called ″type″ and given in the output struct.

Since the reference trace is found from potentially erroneous data, the found time delays are used to correct the reference trace before running the measuring process again. The cross correlations are corrected using the Fourier transform and the shift theorem (McClellan et al., 2007):

$$y(n) = x(n - n_d) \leftrightarrow Y(\omega) = X(\omega)exp(-j\omega n_d) \tag{2.5}$$

The new reference trace is the stack of the corrected daily traces. The measuring process is run a specified number of times (iterations) to improve the final result.

2.3.1 make_reference

The reference trace used to measure the time delays are made from stacking the daily cross correlations over a specified time period. The reference trace is made using the *make_reference* function, where the inputs are all the daily cross correlations and an optional specification about the stacking period. The default is the entire available time period, this can also be achieved by the option 'whole'. The other options are stacking the reference monthly or over a specified number of days, either the last days prior to the day to be compared (option 'days'), or the first days of the time period (option 'firstdays'). When stacking the reference monthly the first and last recoding day of the station pair must be specified. When stacking the reference over a number of days, the number of days must be specified.

2.4 invert_TD

The measured time delays of the station pairs are inverted to find the time delays of each station. The input of the function *invert_TD* is the name of the settings file and the outputs are the inverted time delays of each station, both uncorrected and corrected for absolute time.

The function *invert_TD* loads the measured time delays of each stations pair and inverts to find the time delay of each station using the relation between the stations in matrix form, as explained in the master thesis (Løviknes, 2019). The generalized inverse of the matrix is found using the inbuilt MATLAB function *pinv*, which calculates the Moore-Penrose pseudo inverse of a matrix.

The relation matrix is a matrix of rank 3, meaning that a constant offset can be added to all stations without changing the time differences (Sens-Schönfelder, 2008). The found timing error Δ is therefore still relative and must be adjusted to a reliable station time. The station with the most reliable clock can either be specified in the settings file, which is recommended, or defined in the function as the station with the least fluctuation. The most reliable station can be found from studying the GPS logs of each station or the noise correlations of the different station pairs.

2.5 Correction function

3 Side functions

The functions *plot_egf_td*, *apply_filterband* and *plot_distance* analyse and plots the estimated Green's function and measured time delays in various ways. The inputs of these side functions are the name of the settingsfile and an optional specification about what to plot.

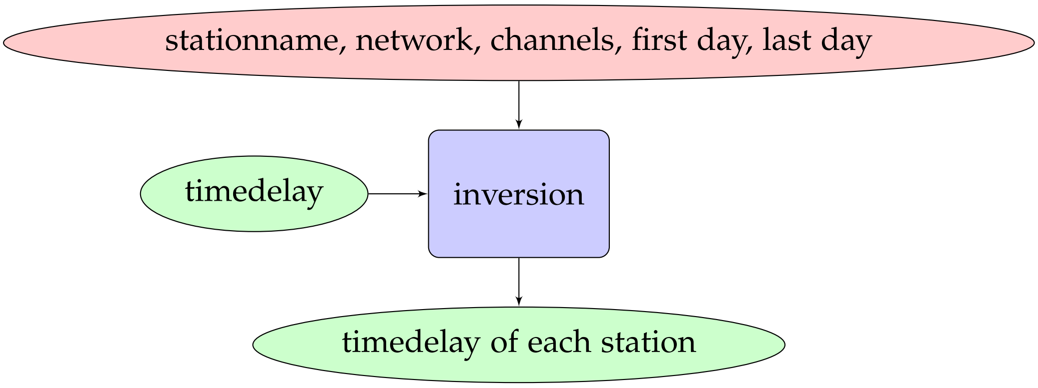

Figure 6: Flowchart.

3.0.1 plot_egf_td

The *plot_egf_td* - function plots the daily noise correlations as a function of amplitude scaled in colours using the inbuilt MATLAB function *imagesc*, the stacked and corrected reference traced used in the measuring process and the measured time delays are both plotted using *plot*. Optionally, only the daily correlations, 'EGF', or the measured time delays, 'TD', can be plotted. When using the option 'DAILY' the daily noise correlations are plotted in one figure using *plot* instead of *imagesc*. The noise correlations are normalized daily by setting the maximum to one, but otherwise not altered before plotting. Both the continuous and the actually measured time delays are plotted.

3.0.2 apply_filterband

The *apply_filterband* - function filters the estimated Green's function over the different frequency bands given in the settingsfile, and plots the results. The function either filters an plots both the daily noise correlations, using *imagescc*, option 'daily', or the stacked noise correlations using *plot*, option 'stack'.

3.0.3 plot_distance

The *plot_distance* - function calculates and plots the estimated Green's functions with distance. The function reads the coordinates of each station from the pole zero file defined in the settingsfile. The distance between the stations of a station pair is then calculated and the noise correlations are plotted with increasing inter-station pair distance.

4 Example - NEONOR2 data

In addition to the functions, the toolbox contains a folder with some example stations, a settings file, *settings.txt* (figure 8), with default values, and an example script for running the functions. The example script *RUN.m* runs the process from estimating the Green's function to measuring time delays, plotting and finally inverting to find the time delay of each station (figure **??**). The stations used in the example are data recorded between 1st of September 2015 until 16th of November 2015 on stations NBB15, NBB14, N2ST and N2TV from the temporary network NEONOR2 (Michálek et al., 2018).

RUN.m

```
settingsfile='settingsfile.txt';

% ESTIMATE THE GREEN'S FUNCTION:
egfs = estimate_GF(settingsfile);

% MEASURE THE TIME SHIFT
delays = measure_timeshift(settingsfile);

% PLOT DAILY CROSS CORRELATION AND TIME SHIFTS
h = plot_egf_td(settingsfile);
h = plot_egf_td(settingsfile,'all');


% INVERT TO FIND THE TIME DELY OF EACH STATION
[fdelay fdelayc] = invert_TD(settingsfile,'plot');

% Plot the results with distance, filters and ampliude spectrum
h = plot_distance(settingsfile)
h = apply_filterband(settingsfile,'stack')
```

Figure 7: Example

```
%%% Settings file %%%

%% Station and time period specifications (mandatory values):
% Example stations:
network = 2D
stations = NBB15 NBB14 N2ST N2TV
channels = Z
location = 00
first_day = 2015-09-01
last_day = 2015-11-16

% Number of stations each station is cross correlated with, default is number of stations - 1
num_stat_cc = 3

% Sampling frequency (must be the same as the sampling frequency given in the file header)
Fq = 10

% The filename format, must include station name, date and folder (if in another folder), if not specified default is
% [network.stationname.location.HHchannels.D.yyyy.ddd.SAC]
filename = stationname/network.stationname.location.HH.D.(DATE).mseed

% Input file format, can be either sac (default) or miniseed
%fileformat = sac
fileformat = miniseed

% The date format used in the filename, default is 'yyyy-mm-dd'
dateformat = yyyymmdd

%% EGF:
% Specify the values specifically for loading the files and for estimating the Green's function
% Pole-zero file format, if not specified the default format is [SAC_PZs_NS_stationname_HHchannels.pz]
pz_file = stationname/SAC_PZs_NS_stationname_HHchannels.pz

% Decimate factor, if no downsampling is needed leave out or blank
% decimate =

% Tolerance number of missing files in row (default = 5)
missingfiles = 5

% Bandpass filter to be applied during preprocessing (mandatory value):
bpf = [0.01 1.25]

% Normalization specification, the options are 'onebit sw'  for onebit before spectral whitening (default), 'onebit' for only onebit normalization and
% 'sw' for spectral whitening before onebit
norm = onebit sw

% Cross correlation window length in hours, must be given as an integer between 1 and 24, the default is 24 hours
wl = 1

% Stack window length in hours, must be given as an integer between 1 and 24, the default is 24 hours
swl = 24

%% TD:
% Specify the values for measuring time shifts
% Bandpass filter to apply on the daily noise correlations and reference trace before measuring time delays, if not specified no filter will be applied
bpfm = [0.1429 0.5000]

% Number of iterations to run the measuring process, must be an integer, default is 3
iterations = 3

% Only use +-lag_red time lag to reduce computational effect, default is 2000
lag_red = 2000

% Specification about the time period the reference is stacked over. The options are the whole period (default), months (the first and last
% day of the correlation period must be specified), days and firstdays (number of days must be specified);
% stackperiod = firstdays 80

%% INVERT:
% Specify values for inversion:
% Station with reliable clock to use as reference during the invertion
reference_clock_station = N2ST

%% PLOT:
% Specify the values for plotting:
xaxis = -150 150
yaxis = -150 150

%% FILTER
% Specify values for applying different bandpass filters:
cutoff_freq1 = [0.25 0.2 1/7 0.1 0.05]
cutoff_freq2 = [1.25 1 0.5 0.2 0.1]
```

Figure 8: An example of a settings file for station NBB15, NBB14, N2ST and N2TV

The most important things to consider when using the functions is to define the mandatory values in the settings file and to make sure the daily files are in the correct format, sac or miniseed, with file names, and/or folder name, containing both the station name and dates. When this is the case, the main functions can easily be run with the name of settings file as input.

The result of running the example script in figure **??** with the settingsfile in figure 8, can be seen in the figures 9.

The estimated Green's function of the station pair NBB15-N2ST shown in figure 9 (left) are stable and symmetric for the first 35 days of the time period. The time shifts around day 35 and day 70 and the time drift of station NBB15 can both seen directly in the estimated Green's function and are detected by the measuring method, as shown in figure 9 (bottom right).

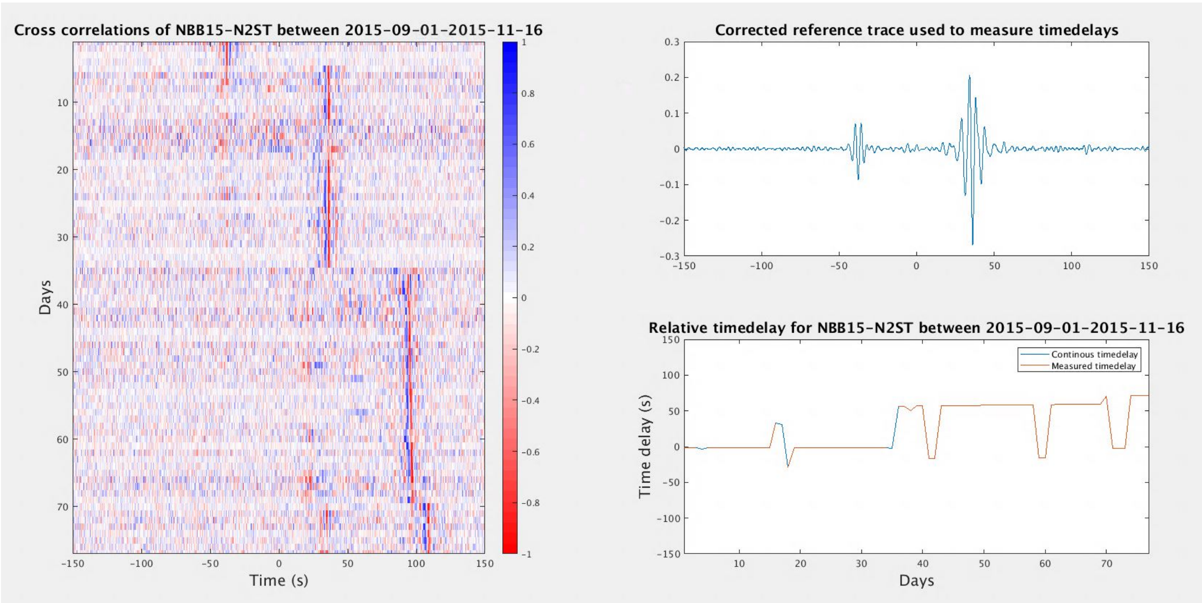

Figure 9: The daily estimated Green's function of the station pair NBB15-N2ST plotted as a function of amplitude (left), the reference trace used to measure the time shifts (top, right), and the measured time shifts (bottom left)

The time shift of each station can be estimated from comparing the cross correlations of the different station pairs (figure 10) and from inversion (figure 11). The inversion is done using the timing of N2ST as reference, this was defined in the settings file.

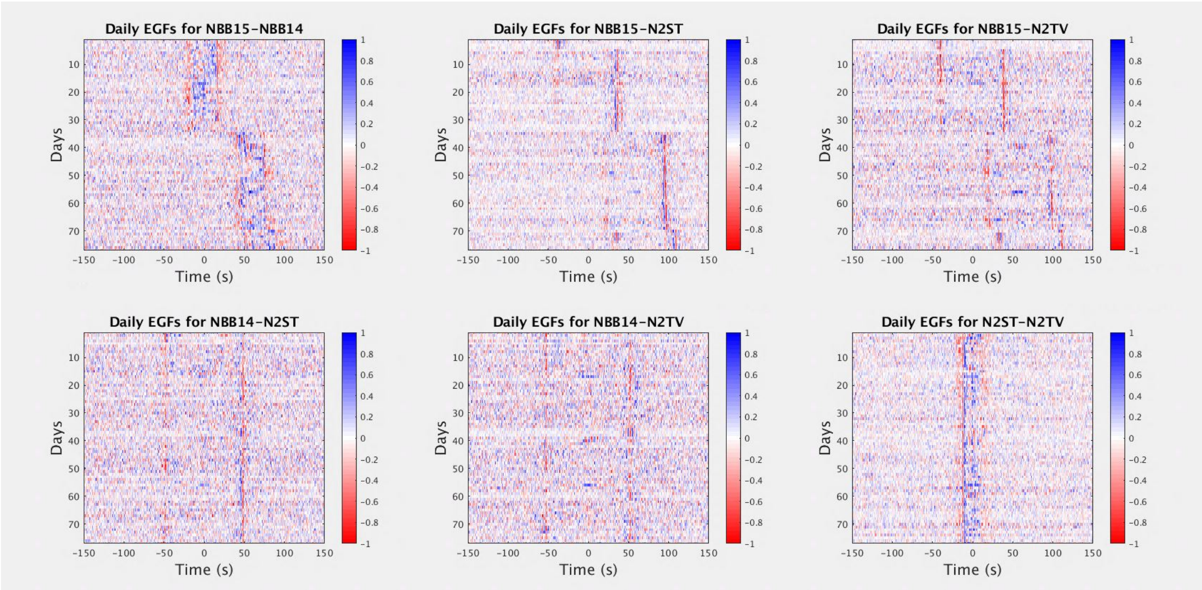

Figure 10: The origin of the time shft can be estimated from comparing the noise correlations of all the station pairs

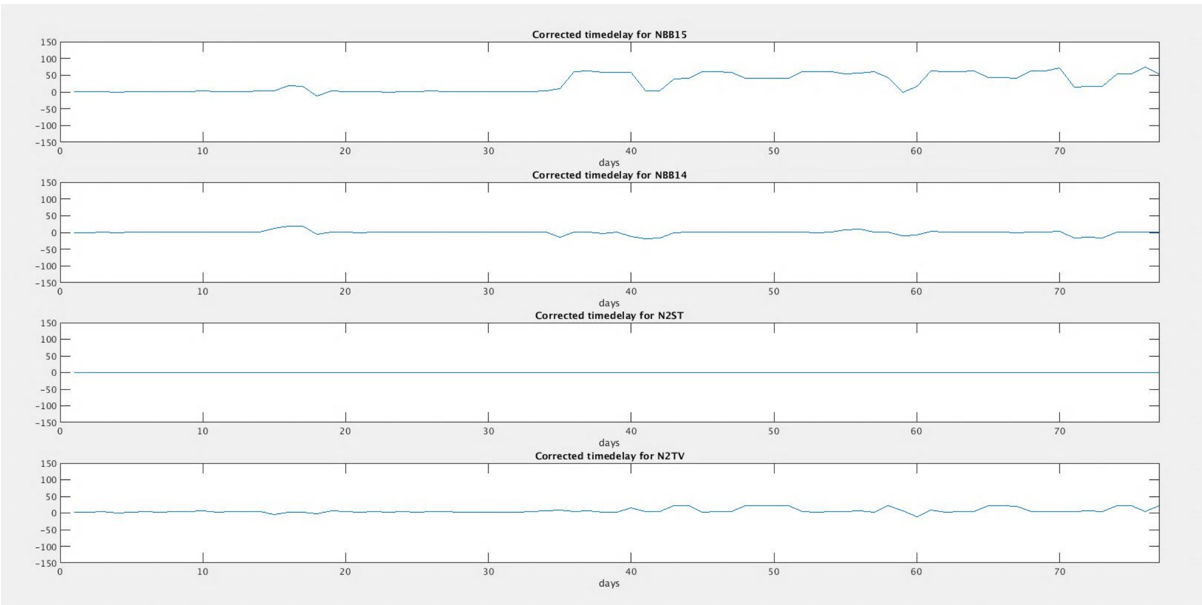

Figure 11: The result of the inversion and absolute clock correction

The estimated Green's functions can be further investigated by applying different bandpass filters (figure 12) and plotting with distance figure 13.

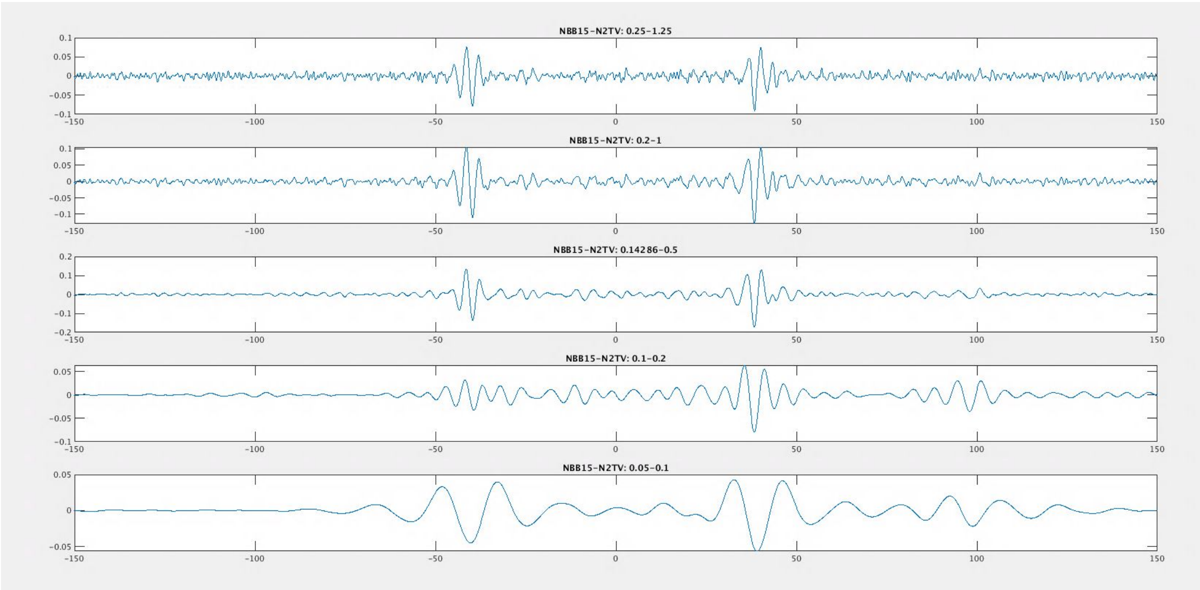

Figure 12: The stacked noise correlation of the station pair NBB15-N2TV filtered over different narrow bandpass filters

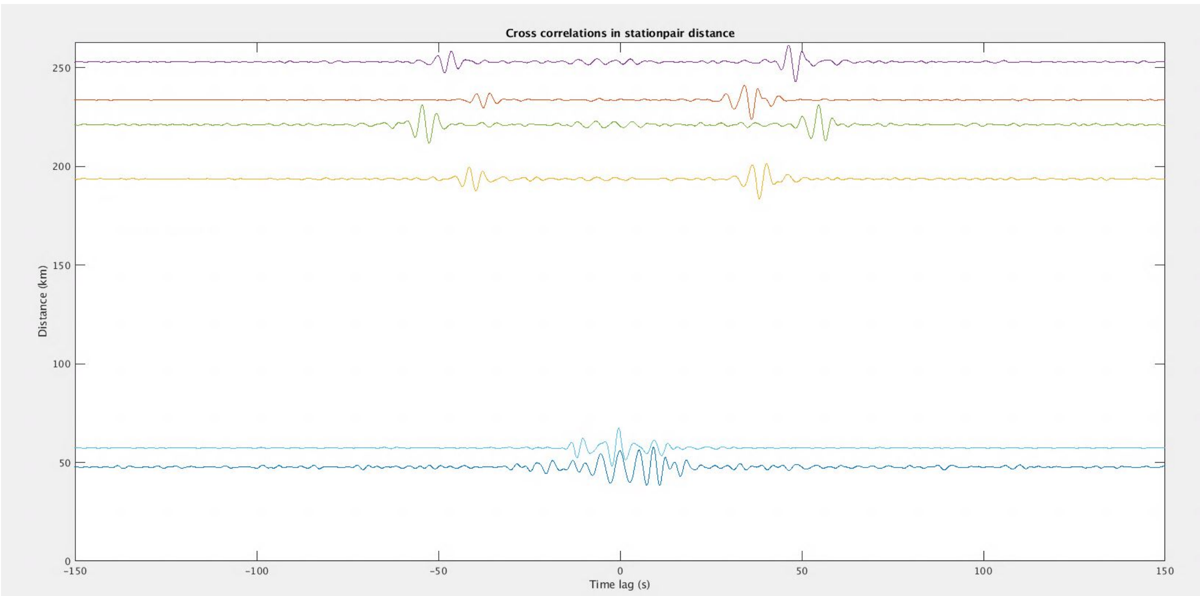

Figure 13: The noise correlations of each station pair plotted with distance between the stations

5 Appendix - list of functions

5.1 Functions

- estimate_GF.m 2.2
  - read_settings.m 2.1
  - read_daily.m 2.2.1
  - str2filename.m 2.1.1
  - gen_response.m
  - read_info_pz.m
  - prepros.m 2.2.2
    * costap_filter.m
    * rm_resp.m
    * spectral_whitening.m
  - cross_conv.m 2.2.3
- measure_td.m 2.3
  - make_reference.m 2.3.1
- invert_td.m 2.4
- plot_egf_td.m 3.0.1
- apply_filterband.m 3.0.2
- plot_distance.m 3.0.3
  - calc_distance.m
- preparefiles.sh 2.1.1

5.2 Example files

- Run.m 4
- Daily miniseed files from station NBB15
- Pole-zero files from station NBB15
- Daily miniseed files from station NBB14

- Pole-zero files from station NBB14
- Daily miniseed files from station N2ST
- Pole-zero files from station N2ST
- Daily miniseed files from station N2TV
- Pole-zero files from station N2TV

——————————————— —- BIBLIOGRAPHY ———————————————

References

Løviknes, K. (2019). Measuring seismic station timing errors from ambient noise. *Master's thesis, Department of Earth Science, University of Bergen.*

McClellan, J. H., Schafer, R. W., and Yoder, M. A. (2007). *DSP First (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Michálek, J., Tjåland, N., Drottning, A., Strømme, M. L., Storheim, B. M., Rondenay, S., and Ottemöller, L. (2018). Report on seismic observations within the NEONOR2 project in the Nordland region, Norway. Technical report, University of Bergen, Bergen.

Sens-Schönfelder, C. (2008). Synchronizing seismic networks with ambient noise. *Geophysical Journal International*, 174(3):966–970.