UNIVERSITÀ DEGLI STUDI DI TRIESTE

dipartimento di matematica e geoscienze

The DMG Quick Reference Manuals

Noise cross-correlation

Aimed at the retrieval of the Green function

QR

Document name: DMGQuickNoise.pdf
Document Version: 1.0
Date Issued: October 3, 2017 5:47 PM
Author: Franco Vaccari

University of Trieste
Department of Mathematic and Geosciences
Group of Seismology
via Weiss, 4
34128 Trieste, Italy

Table of Contents

Introduction

The purpose of this manual is to describe the processing required to obtain the cross-correlation of simultaneous noise recordings obtained from a couple of instruments.

If the noise recording is properly made (proper distribution of noise sources, enough duration of the recording), the Green function of the structural model between the two instruments should appear in the cross-correlation. The longer the recording, the better the signal-to-noise ratio in the cross-correlation.

The cross-correlation obtained from the vertical component of the recorded noise can be analyzed with the XFtan2012 software for the forward and inverse modeling of the Rayleigh group velocity dispersion curve.

Data preparation

Time series file format

The two time series to be cross-correlated must be joined into a single file made of two columns containing only the amplitudes of the recordings. The sampling must be the same for the two time series, and must be indicated in the parameter file `prenoisecross.par` that is used to pre-process the file. The sampling should be known to the user from the analysis of the raw data produced by the instruments.

It is best that the two amplitudes of each record are in sync, so that they share the same absolute time. It is in any case possible to apply a time shift during the processing to bring the samples in sync.

Example of data file

This is an example of data file, with the two amplitudes listed in the two columns

```
-0.000291 0.0002180
-0.000328 -0.000655
-0.000510 -0.001056
0.0004370 -0.000837
-0.000109 -0.000765
-0.000437 -0.000146
-0.000510 0.0001820
-0.000073 0.0001820
0.0000000 0.0008740
...
...
0.0000730 0.0005820
-0.000437 0.0008740
-0.000109 0.0009830
0.0000000 -0.000073
0.0000000 -0.001201
0.0000360 -0.002184
0.0000000 -0.001820
-0.000546 -0.001238
-0.000146 -0.000364
0.0000360 0.0010920
-0.000182 0.0024390
```

Pre-processing

Before the actual cross-correlation is made, a pre-processing step is required. The purpose of the pre-processing is to divide the full time series of the noise recording into couples of windows (one file for the master and one for the second recording) where the signals are synchronized. Each window has to have a number of samples that is a power of 2.

The length of each window generated in the pre-processing has to be at least long enough so that it can contain the full Green function that will form after the cross-correlation. Usually in the pre-processing step the window length will be much longer. It will be in the cross-correlation step that the long recording will be split in smaller windows and a stacking of the results will be finally applied to improve the signal-to-noise ratio in the generated Green function.

The parameter file used to configure the pre-processing is named `prenoisecross.par`, and is used by program *prenoisecross.out*. When executed, *prenoisecross.out* prepares the input time series that will be read in by *noisecross.out*.

in the example available in /XDST/Examples/Noise, the full noise recording will be divided into windows of 65536 samples each (7948799 total duration, divided by 65536 samples per window leads to 121 windows, with the last 18943 samples of the full recording discarded

(7948799-(65536 x 121)=18943)

Example of parameter file

An example of the parameter file that has to be properly configured to deal with the user's noise data is given below:

```
PARAMETER FILE FOR PROGRAM PRENOISECROSS
1          ! output data format (ibin:0 ASCII, 1=Binary)
1          ! original frequency sampling (samples per second)
0          ! decimation factor
0          ! t start (s) for input data (if tshift>0, refers to shifted trace)
7948799    ! t stop  (s) for input data (if tshift>0, refers to shifted trace)
0.         ! time of shift(s) (0=no shift: the program run faster!)
1          ! track to shift (1=first column; 2=second column)
65536      ! number of samples for output files (power of 2)
65536      ! pts threshold to make or not the last short win
jusma274_365.dat                             ! input noise data (2 cols)
mm_test                                      ! output filename root (col 1)
ms_test                                      ! output filename root (col 2)
```

Binary output format is highly desirable when dealing with very long time series, as it reduces both disk space occupation and computational time.

It is generally expected that input data are already sampled with the desired step, and do not need a decimation or a synchronization. The program offers the choice to adjust that, if required. This has not been thoroughly tested though, so if the decimation option and the time shift option are activated, it is strongly suggested to use the ASCII output format, that allows to check visually the output files for possible anomalies in the resampling and/or in the relative shift between the traces. Better to resample and shift the original data with other, well established tools.

The number of samples for the output files must be a power of 2.

Cross-correlation

After the pre-processing step, the actual cross-correlation can be performed running program *noisecross.out*.

Example of parameter file

The configuration of the processing is made adjusting the parameters in file `noisecross.par`, as seen below.

```
PARAMETER FILE FOR PROGRAM NOISECROSS
1                   ! ibin: 0=ASCII,1=BINARY
65535.0             ! length of sub-window in seconds
65536.0             ! shift of sub-window in seconds
765.0               ! distance in km between the two recorders
512.0               ! time window for ftan output
10.                 ! duration for the cosine window(s) (0=NO)
0                   ! filter type(0=No, 1=Btt.,2=Gauss.)
0  6.0   0.8 0.02  ! lowpass (flag,cutoff in Hz,%,amp)
0  1.5   0.8 0.02  ! highpass (flag,max1 in Hz,%,amp)
1                   ! rbit-normalization (0=no, 1=yes) reduce amplitudes to -1 or 1
0 2                 ! efft-filtered input signals (0=no, 1=yes) and motion(1=d,2=v,3=a)
master                                                 ! output file: master name part
second                                                 ! output file: second name part
mm_test                                                ! input file root name (master)
ms_test                                                ! input file root name (second)
1                   ! number of ranges to compute
1                   ! start index of range n.1
120                 ! end index of range n.1
```

The choice whether to use ASCII or binary input files depends on what was indicated in the pre-processing step. The two choices must match or program will fail to run.

Sub-window properties

The length of the sub-window, expressed in seconds, must be chosen so that the sub-window's number of samples is a power of 2. This length defines the duration of the sub-windows into which the signals prepared by *prenoisecross.out* will be subdivided, in view of their stacking. In the example above, where the sampling interval is 1 s, note that the duration of the sub-window is 65535, so that the number of samples is 65536 (that is a power of 2). The sub-window length is obtained as *(n.of.samples - 1) x dt*, where *dt* is the sampling interval (in seconds) and *n.of.samples* must be a power of 2.

The shift of sub-window, expressed in seconds, is generally defined so that the time of first sample of the (i+1)-*th* sub-window is equal to the time of the last sample of the i-*th* sub-window, plus one sampling interval. In such a way, no sample will be skipped when stacking the sub-windows in the cross-correlation.

Cross-correlation files (*.ft)

The time window length of the stacked cross-correlation file to be passed to XFtan2012 software (file with extension .ft) must be long enough to contain the full Green function. Of course it is directly proportional to the distance between the recorders, and inversely proportional to the velocity of the structural model.

Tapering

A cosine taper can be applied to the edges of the sub-windows, by defining a duration grater than 0 in the parametyer file.

Filtering

While the filltering of the cross-correlation can be later performed in XFtan2012 software, there is a possibility to filter the data with a band-pass filter (Gaussian filter must be used; Butterworth filtering is not yet fully implemented) to remove periods not relevant to the analysis. Do not use a very selective filter: better define a large band-pass filter here and restrict the frequency range later in Xftan2012. The band-pass filter is defined as the combination of a low-pass and a high-pass filter, whose parameters can be configured in the parameter file.

For the low-pass filter, the cutoff frequency is defined (6 Hz), then the parameter that defines the frequency up to which the filter amplitude is 1, expressed as a percent of the cutoff (0.8 means 6.0 x 0.8 = 4.8 Hz), and the amplitude at the cutoff (0.02).

For the high pass filter, the frequency down to which the filter amplitude is 1 is given (1.5 Hz), then the parameter that defines the cutoff frequency, expressed as a percent of the lowest frequency with amplitude =1 (0.8 means 1.5 x 0.8 = 1.2 Hz), and the amplitude at the cutoff (0.02).

Bit-normalization

The bit normalization is usually activated, to eliminate the relevance of spurious spikes or transients that might contaminate the noise registration.

efft.out processed files (do not use!)

The option to use files prepared by *efft.out* should not be used (so let the first parameter be 0).

File naming

Output file names: only the first one matters, and will become part of the filenames generated by *noisecorr.out*.

Input file names refer to the files generated by *prenoisecross.out* in the pre-processing step.

Of the files generated in the pre-processing step, only a subset of windows can be used by defining a set of ranges. In the example file shown above, a single range is defined that accounts for the complete set of 121 windows prepared in the pre-processing step. As an example, to use two sub-sets of windows generated in the pre-processing step, parameters should be modified from:

```
1                   ! number of ranges to compute
1                   ! start index of range n.1
120                 ! end index of range n.1
```

to

```
2                   ! number of ranges to compute
20                  ! start index of range n.1
40                  ! end index of range n.1
100                 ! start index of range n.2
121                 ! end index of range n.2
```

Example execution

Once the parameter files *prenoisecross.par* and *noisecross.par* has been set, the user can perform the cross-correlation by simply calling the programs in sequence:

prenoisecross.out

noisecross.out

Below an example session, with the workflow copied from the screen. The example files for this cross-correlation can be found in /XDST/Example/Noise/Input

Run of prenoisecross.out

```
[is01:/XDST/Examples/Noise] vaccari% prenoisecross.out
 nskip1,nskip2          0           0
 nskip1,nskip2          0           0
 Input filename=.......................: jusma274_365.dat
 Output master track name................: mm_test
 Output second track name................: ms_test
 Output data format (0=ASCII;1=BIN)......:         1
 Original frequency sampling.............:   1.0000000
 Original sampling time (ms).............:   1000.0000
 Decimation sampling.....................:         0
 Start time (s)..........................:   0.0000000
 End time (s)............................:   7948799.0
 N. of samples for output subwindows.....:       65536
 Points threshold for the last subwindow.:       65536
 No shift implemented....................:         ---
 shiftpts          0
 First and last input samples (master)...:           1      7948800
 First and last input samples (second)...:           1      7948800
 Number of complete windows..............:         121
 Samples in the last uncomplete window...:       18944

 Reading data...
 Samples for subwindows:
   Subwindow       First         Last        npts        file
           1           1        65536       65536  -->  mm_test000001.yyy
                                                   -->  ms_test000001.yyy
           2       65537       131072       65536  -->  mm_test000002.yyy
                                                   -->  ms_test000002.yyy
           3      131073       196608       65536  -->  mm_test000003.yyy
                                                   -->  ms_test000003.yyy
           4      196609       262144       65536  -->  mm_test000004.yyy
                                                   -->  ms_test000004.yyy
...
...
         118     7667713      7733248       65536  -->  mm_test000118.yyy
                                                   -->  ms_test000118.yyy
         119     7733249      7798784       65536  -->  mm_test000119.yyy
                                                   -->  ms_test000119.yyy
         120     7798785      7864320       65536  -->  mm_test000120.yyy
                                                   -->  ms_test000120.yyy
         121     7864321      7929856       65536  -->  mm_test000121.yyy
                                                   -->  ms_test000121.yyy
STOP EXECUTION COMPLETE
[is01:/XDST/Examples/Noise] vaccari
```

Generated files

The generated files will have extension .yyy and the sub-window index included in the filenames

```
[is01:/XDST/Examples/Noise] vaccari% ls -l *.yyy
```

```
-rw-r--r-- 1 vaccari dstguest 786456 Jun 20 15:58 mm_test000001.yyy
-rw-r--r-- 1 vaccari dstguest 786456 Jun 20 15:58 mm_test000002.yyy
-rw-r--r-- 1 vaccari dstguest 786456 Jun 20 15:58 mm_test000003.yyy
-rw-r--r-- 1 vaccari dstguest 786456 Jun 20 15:58 mm_test000004.yyy
...
...
-rw-r--r-- 1 vaccari dstguest 786456 Jun 20 16:00 ms_test000118.yyy
-rw-r--r-- 1 vaccari dstguest 786456 Jun 20 16:00 ms_test000119.yyy
-rw-r--r-- 1 vaccari dstguest 786456 Jun 20 16:00 ms_test000120.yyy
-rw-r--r-- 1 vaccari dstguest 786456 Jun 20 16:00 ms_test000121.yyy
[is01:/XDST/Examples/Noise] vaccari%
```

Run of noisecross.out

```
[is01:/XDST/Examples/Noise] vaccari% noisecross.out

 dati range........=           1           1         121
 ncouples=         121
 Duration window to correlate =    65535.000
 Shifting duration window =    65536.000
 Master to receiver distance  =    765.00000
 Time window output =    512.00000
 Time croping for cosine window =    10.000000
 Filter type chosen =           0
 Output master name = master
 Output receiver name = second
 Number of data couples =         121
 loop:           1
 index=           1
 mm_test000001.yyy                       ms_test000001.yyy
 Sampling (s)..............................:   1.0000000
 Original n. of samples....................:       65536
 Samples really used (pow of 2)............:       65536
 of course...
       65536       65536           1
 Samples asked for each window.............:       65536
 Length asked for each window (s)..........:   65535.000
 Samples used for each window (pow of 2)....:       65536
 True length (s)...........................:   65535.000
 Window shift for stacking (s).............:   65536.000
 nshift           0
 PLEASE WAIT! Cross-correlating... & Stacking...
 ISTART           0
 appling cosine window to windows
 Stacking the cross-correlation...           1
 73.4***************178.1265.8154.8218.2 -2.4********** 96.4299.0146.3319.7-26.2
 ANSTACK
 73.4***************178.1265.8154.8218.2 -2.4********** 96.4299.0146.3319.7-26.2
 Number of windows stacked.................:           1


...
...


 index=         121
 mm_test000121.yyy                       ms_test000121.yyy
 Sampling (s)..............................:   1.0000000
 Original n. of samples....................:       65536
 Samples really used (pow of 2)............:       65536
 of course...
       65536       65536           1
 Samples asked for each window.............:       65536
 Length asked for each window (s)..........:   65535.000
 Samples used for each window (pow of 2)....:       65536
 True length (s)...........................:   65535.000
 Window shift for stacking (s).............:   65536.000
 nshift           0
 PLEASE WAIT! Cross-correlating... & Stacking...
 ISTART           0
 appling cosine window to windows
 Stacking the cross-correlation...           1
********** 80.1149.1295.6*****125.6238.1*****-65.2*****421.7-53.2100.0*****-62.0
 ANSTACK
**************170.9*************************************************************
 Number of windows stacked.................:           1

 Stacking the whole data       65536
```

```
 STAKPOS
***************170.9********************
 STAKNEG
*****178.8*************************810.5
 STAKFOL
**********************************-98.9
 time window wanted :   512.00000
 number of points taken :          513
 creating the .xpos.ft file
 creating the .xneg.ft file
 creating the .xfold.ft file
 creating the .xpos. file
 creating the .xneg. file
 creating the .cross. file
 creating post filtering GF file xpos
 creating post filtering GF file xneg
 creating post filtering GF file xfold
 OK - Execution completed
[is01:/XDST/Examples/Noise] vaccari%
```

Generated files

The relevant files generated by *noisecross.out* are those with the .ft extension, that can be read by XFtan2012 program for the analysis of the group velocity dispersion:

```
[is01:/XDST/Examples/Noise] vaccari% ls -l*.ft
-rw-r--r-- 1 vaccari dstguest 9191 Jun 20 16:43 master.xfold.second.ft
-rw-r--r-- 1 vaccari dstguest 9190 Jun 20 16:43 master.xneg.second.ft
-rw-r--r-- 1 vaccari dstguest 9190 Jun 20 16:43 master.xpos.second.ft
[is01:/XDST/Examples/Noise] vaccari%
```

where:

xfold is the folded cross-correlation

xneg is the negative part of the cross-correlation

xpos is the positive part of the cross-correlation

The xfold file is obtained as the sum of the positive part with the folded negative part. When the sources of the noise are properly distributed in the survey area it should be the best series to look at for the recognition of the Green function.

References

Campillo, M. and Paul, A. (2003). Long range correlations in the seismic coda. Science, 299, 547-549

Bensen, G. D. et al. (2007). Processing seismic ambient noise data to obtain reliable broad-band surface wave dispersion measurement. Geophysical journal international, 169, 1239-1260.

Appendix 1: Noise Acquisition with Tromino

Instrument configuration

Instrument configuration should be made before moving the instruments to their recording location, to minimize the amount of work to be done on site.

Card partitioning

For very long recordings, the memory card has to be partitioned with a single-partition scheme. If many short recordings are to be made, the memory card should be partitioned into as many partitions as required. Starting from the first partition, each new recording will be stored sequentially into the next partition.

Acquisition settings

For noise recording, typically the Tromino acquisition settings have to be configured like this:

- Sampling: 128 Hz
- Program: 4
- Min sat number: 3
- GPS Antenna: external
- Trigger channel: NO
- Acquisition length: Memory Limit
- At least High Gain channel is selected

Instrument positioning

For noise recording, two (or more) Tromino instruments have to be deployed at the ground level of two (or more) buildings, with external GPS antenna and power supply connected.

The location has to be properly selected so that:

- The instrument can not be stolen
- There is an electric plug available
- There is a way to position the GPS antenna outsidev the building, with open sky to the south.

Usually only the vertical component of motion will be used, so in principle the only requirement is that the instrument must be properly leveled. This is achieved by adjusting the three feet of the case while checking the instrument's bubble level.

If two instruments are used to acquire data for a single profile, it is suggested that the instrument's North is aligned with the profile azimuth.

Data acquisition

With the acquisition settings defined as above, after selecting the menu item "Acquire" of the "Acquisition" menu data, acquisition starts automatically as soon as the GPS locks the three required satellites. Traces are saved with GPS time information, and synchronization between traces will be later obtained using the Grilla software.

Data acquisition stops automatically when the memory card partition is filled up with data, or when the required time has expired should the instrument have been configured in “Timer” mode. The user can stop manually the acquisition at any time by pressing simultaneously the two central buttons of the instrument.

⚠️➡Do not move the instrument while data acquisition is on⬅⚠️. Always check that acquisition has ended (manually or due to expired time or memory condition) before handling the instrument, or the sensors could get damaged. Shut down the instrument after acquisition has ended.

Data downloading

After the acquisition has been made, the recorded data can be downloaded from the instruments to a computer using the Grilla software on a Windows computer. Trace synchronization will produce ASCII files with columns of data amplitudes for each active channel.

For noise cross-correlation purposes, only the vertical component of motion will be considered. Using the data recorded by a couple of instruments, a single file with two columns of amplitudes will be obtained.

File conversion

Since the synchronized data are obtained on a Windows computer, but the processing will occur in a Unix environment, line-endings of the file has to be properly converted to Unix style. So, the first thing to do after transferring the synchronized traces to is01 server, is to issue the command:

```
dos2unix filename
```

to convert the line endings of each transferred file. The resulting files are ready to be processed for noise cross-correlation.