Time-Frequency HOS Signal Processing programs:

Package provides a set of C and python modules for calculating higher-order statistics (HOS) and/or RMS envelope characteristic functions (CF) of the recorded seismic signals. In this version of the program only kurtosis HOS (4th moment) and envelope CFs are implemented together with the time-frequency analysis option of the Multi-Band Filter (MBF). The MBF analysis scheme is a modified version of the FilterPicker algorithm of Lomax *et al.* (2012).

- Requirements:

  C compiler (GCC)

  Python - 2.7.x< version < 3.0

  NumPy

  SciPy

  Matplotlib

  Obspy

- Installation and running example codes:

1. Compiling C libraries: run following command from the command line in your terminal:

% python setup.py install

This should create build/ and lib/ folders.

2. Running example python codes:

There are two python codes provided as examples for illustrating how the modules for CFs' and MBF signal processing can be used. Explanation of how to run the examples are given below:

% python make_CF.py config_SigProc.txt

example for calculating kurtosis CF on a local earthquake record. Outputs: figure comparing the original record and calculated CF; data file of CF in SAC format.

% python make_MBF_CF.py config_SigProc.txt

example for MBF kurtosis CF calculation on a local earthquake record. Outputs: figure illustrating MBF signal processing scheme and CF calculation step; figure comparing the

original record and final composed MBF kurtosis CF; data file of the composed MBF kurtosis CF in SAC format.

Note: Same example output figures can be found in folder figs_example/, for comparison. You can make your own examples by modifying the input data and/or parameters for CFs' and MBF. This can be done by editing configuration file: config_SigProc.txt.

- Configuration file and parameter explanation:

config_SigProc.txt: example configuration file:

```
#--settings for input data files----------------------------
data_dir = 'data_example'
data_file = 'XS.QF17.00.HHZ.D.2010.091'
#
start_time = '2010-04-01T12:52:30'
end_time = '2010-04-01T12:55:30'
#
#--settings for CF calculations-----------------------------
decay_const = 0.5
ch_function = 'kurtosis'
#
#--settings for MBF filtering -------------------------------
nr_freq_bands = 15
f_min = 0.02
f_max = 50.
band_spacing = 'log'
var_window = False
#
#--setting for outputs: plotting and data
save_CF = True
do_plot = True
do_plot_MBF = True
save_plot = True
#
out_data_format = 'sac'
plot_format = png
```

- l.2. **data_dir** - relative/absolute path to the data folder.

- l.3. **data_file** - name of the data file to be analyzed (data should be in SAC, mseed, SEED, or any other formats accepted by ObsPy). The path to the data file is composed as: data_dir/data_file.

- l.5,6. **start_time & end_time** - start and end times of the data to be read into the memory and used in signal processing. If omitted entire record will be processed (accepts start_time = None).

- l. 9. **decay_const** - Values of decay constant T_{decay} (seconds), used for calculating the CFs (same variable for MBF CF calculation).

- l.10. **ch_function** - Type of CF to be calculated: *kurtosis/envelope*.

- l.13. **nr_freq_bands** - Number of frequency bands, defining number of central frequencies for the band-pass filters in MBF analysis.

- l.14. **f_min** - Lower frequency limit in MBF scheme (defined by the length of the data or characteristics of the instrument)

- l.15. **f_max** - Upper frequency limit in MBF scheme (defined by the sampling rate of the data or characteristics of the instrument).

- l.16. **band_spacing** - Defines scale of frequency sampling between the two limit frequencies f_min and f_max. Possible options: lin/log (default value = log)

- l.17. **var_window** - Defines if the decay constant for MBF CF calculation is constant (False) or variable (True). If *True* - a frequency-dependent decay constant will be used for CFs of MBF. Only implemented for the envelope CF.

- l.20. **save_CF** - Option for saving the resulted CFs as separate data-file (if *True*). Format is defined by variable `out_data_forma` (line 25).

- l.21. **do_plot** - Option for plotting the comparison of the final CF and the original data.

- l.22. **do_plot_MBF** - Option for plotting the MBF analysis and CF calculation results.

- l.23. **save_plot** - Defines if the above plotting will be save to a file or output in the matplotlib interactive graphic window.

- l.25 **out_data_format** - Defines the format in which final CF will be saved (default = *SAC*). Can be any format supported by the ObsPy.

- l.26. **plot_format** - Format for saving the files with plotting (png/pdf).

Note: Order in which parameters are listed in the configuration file is not important. For description of the parameter's settings and their default values as defined in the code, refer to the file: SignalProcessing_modules/configspec.conf

- Description of signal processing scheme:

1. CF calculation

Here we describe the signal processing scheme providing a representation of the seismic signal by its characteristic functions (CF). There are two types of CF representations implemented here: higher-order statistics and RMS envelopes.

1.1. Higher-order statistics (HOS) CFs

Higher-order statistics (HOS) is a signal processing tool allowing detection of the transients in stationary signals. Most widely used in the signal processing algorithms HOS, are the 3rd and 4th central moments, known as skewness and kurtosis. In general, the statistic moment of any order can be used. Here we will only focus on the signal analysis based on the 4th central moment (kurtosis) defined as the 4th moment about the mean normalized by the square of its variance. Kurtosis characteristic functions $CF(t)$ can be obtained by estimating kurtosis HOS of the signal in sliding time windows. These calculations can become rather bulky on large volumes of data. The problem can be overcome by applying recursive scheme to calculation of CF of the signal. The recursive scheme used here is based on the recursive exponential average. By directly extending the definition of the recursive exponential average to the variance and kurtosis, the expression of kurtosis can be written in recursive form as:

$$CF_{kurt}(t_i) = C\left[\frac{u(t_i) - \bar{u}(t_i)}{\sigma(t_i)}\right]^4 + (1 - C)CF_{kurt}(t_{i-1}) \qquad (1)$$

were recursive average $\bar{u}(t)$ and variance $\sigma^2(t)$ are defined in following way:

$$\bar{u}(t_i) = Cu(t_i) + (1 - C)\bar{u}(t_{i-1}), \text{ and } \sigma^2(t_i) = C(u(t_i) - \bar{u}(t_i))^2 + (1 - C)\sigma^2(t_{i-1}) \qquad (2)$$

with decay constant C; $0 \leq 1 - C \leq 1$.

Decay constant C represent the degree of weighting decrease, or a smoothing factor, defining how fast the memory of older data-points will be discarded. Here decay constant is defined as $C = \frac{\Delta T}{T_{decay}}$, with ΔT and T_{decay} representing respectively the sampling interval, and the time-length of the averaging scale in seconds. Smaller values of T_{decay}, (higher values of C), will have an effect of discarding faster the older data points, and vice versa. Due to its similarity to

the effect of the classical sliding time window on the "smoothness" of the calculated CF, the values of decay constant C is set indirectly by defining the value of T_{decay} in seconds (expressed variable **`decay_const`**). The lower limit of the values for time-averaging scale is given by the sampling interval of the data ($T_{decay} \geq \Delta T$), while the upper limit is imposed by the type of the analyzed data - time scale of the targeted transients in the signal, and the total length of the signal. This parameter should be set for each particular data-set, through testing step.

The influence of T_{decay} parameter on the shape of kurtosis CF is illustrated in Figure 1. Larger values of T_{decay} will have the effect of longer decay tails in kurtosis CF given by the longer memory of previous samples. Figure 1 is an example of the output of the ***make_CF.py*** code run for the values of *decay_const* parameter (T_{decay}) equal to 0.5 and 4 seconds.

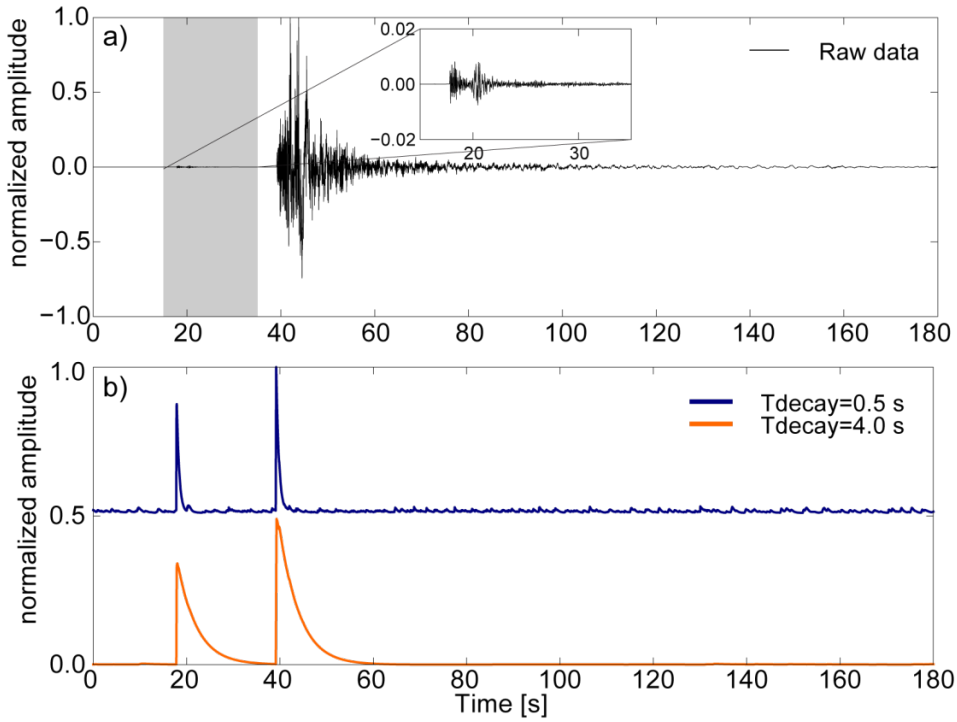

Figure 1. a) Seismogram record of a local earthquake. The inset figure shows an enlarged view of preceding smaller event. b) Kurtosis CF of the record in (a) calculated using the recursive formulation, for two different decay constants T_{decay}.

1.2. Envelope CFs

The root-mean-square (RMS) envelope provides a characterization of the energy variations within a signal. This allows to provide a CF suitable for extracting information on the emergent

signals, characterized by slow variations of energy with time, that do not have clear impulsive phases. This kind of signals are often associated to such phenomena as tectonic tremors.

Following the same idea of recursive implementation scheme, we define the envelope CF as a single component recursive RMS envelope of the seismic record $u(t)$, using the expression below:

$$CF_{env}(t_i) = \sqrt{Cu^2(t_i) + (1 - C)CF_{env}(t_{i-1})^2} \qquad (3)$$

with C - decay constant, same as in equations (1) and (2).

2. MBF analysis

The multi-band filter (MBF) algorithm provides a time-frequency decomposition of the signal by producing a set of band-pass filtered time series with different central periods (Fig. 2). Here we follow the recursive multi-band characterization scheme of FilterPicker algorithm by Lomax *et al.* (2012). It consists of obtaining a set of filtered signals $U_n(t)$ by running the original recorded signal $u(t)$ through a predefined filter-bank of narrow band-pass filers covering the interval of interest [f_{min}: f_{max}], and having $n=0$, N_{band} central frequencies f_n (Figs. 2 and 3). The number of frequency bands N_{band} (**nr_freq_bands**) are given by the desired sampling of the frequency space. One can define either linearly or logarithmically spaced central frequencies f_n for the filter-bank. The set of filtered signals $U_n(t)$ are then obtained by applying a cascade of the two high-pass and two low-pass recursive one-pole filters. This is equivalent to a two-pole band-pass filter (Fig. 3). Then, a frequency-dependent CFs ($CF^n(t)$) are calculated by running the kurtosis/envelope recursive estimations for each of the filtered signals $U_n(t)$. This yields a time-frequency dependent representation of the signal's characteristics.

A final broadband MBF CF ($CF^{TF}(t)$) can be composed by applying maximum or RMS mean operator to time-frequency CFs over the entire frequency range of the filter [*f_min*: *f_max*]:

$$CF^{TF}(t) = max\{CF^n(t); n = 0, N_{band} - 1\} \qquad (5)$$

$$CF^{TF}(t) = \sqrt{\frac{\sum_{n=0}^{N_{band}-1} CF^n(t)^2}{N_{band}}} \qquad (6)$$

In this version we use maximum operator for composing the final MBF CF in the case of kurtosis and RMS sum operator for the envelope CF.

An illustration of the MBF signal processing scheme and broadband CF calculation is provided in Fig. 2. The result corresponds to the output figure generated by the ***make_MBF_CF.py*** code. In this case, the MBF is performed using a filter-bank of 15 (*nr_freq_bands* = 15) logarithmically spaced (*band_spacing*=log) band-pass filters, covering the frequency interval of [*f_min:f_max*] equivalent to [0.02:50] Hz. The frequency-dependent kurtosis CFs are calculated assuming a constant T_{decay} parameter (*decay_const*) equal to 0.5 seconds.

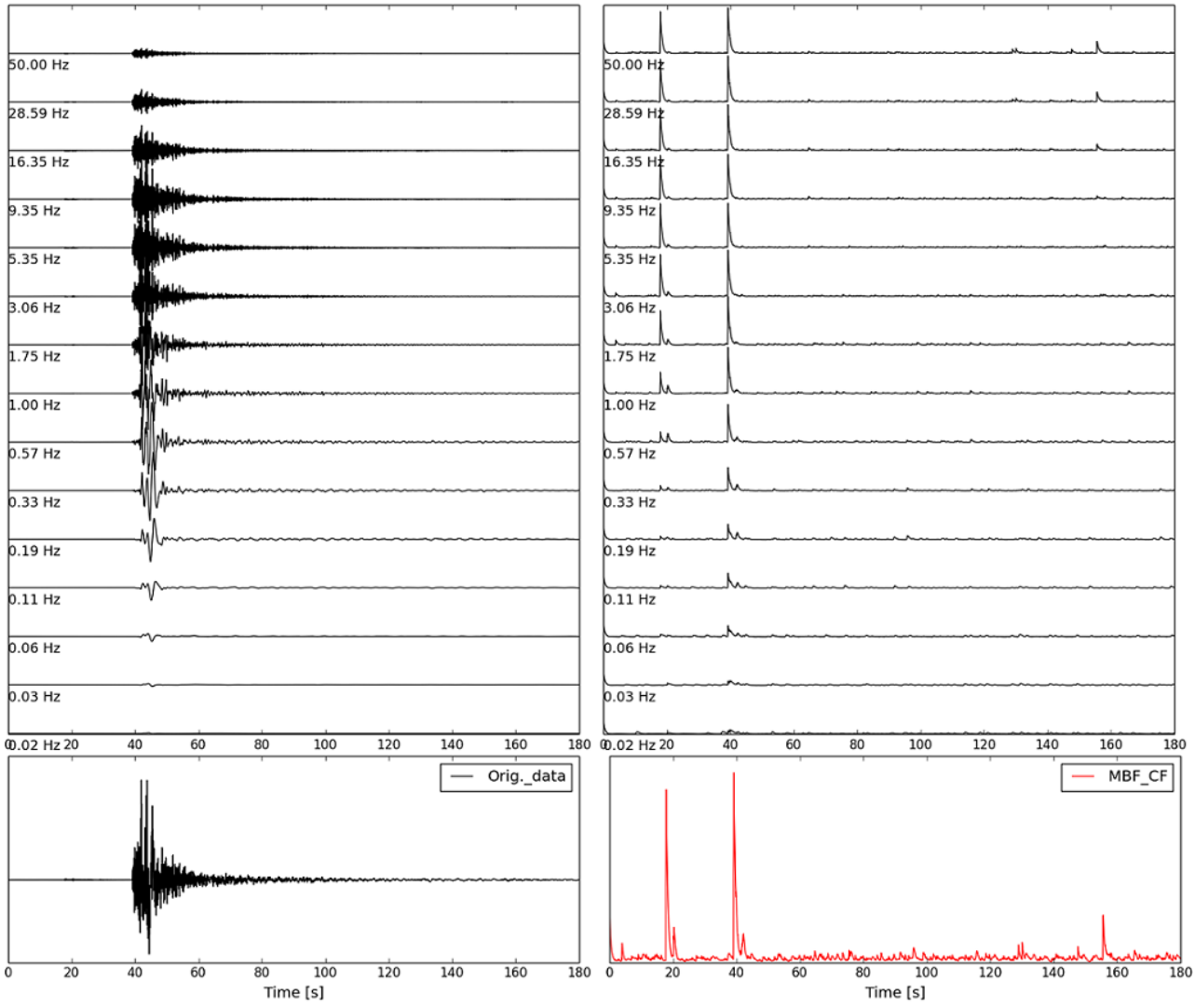

Figure 2. Example of the MBF time-frequency representation scheme using the kurtosis CF, applied to a record of typical local earthquake. Compose final MBF kurtosis CF is show in red. All the parameters used for calculation are provided in *config_SigProc.txt* example control file.

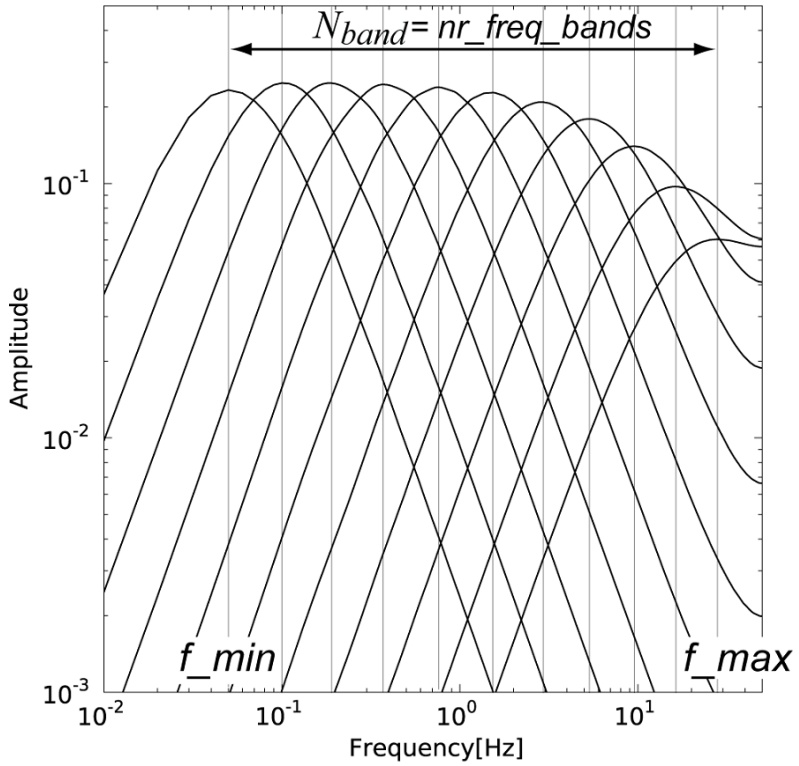

Figure 3. Frequency response of a filter-bank composed of eleven serial two high-pass and two low-pass serial recursive filters.

- Folder content:

List of folders and files in the package.

SignalProcessing/

1. build/
2. lib/
3. SignalProcessing_modules/
4. data_example/
5. figs_example/
6. Readme
7. Documentation.doc
8. config_SigProc.txt
9. make_CF.py
10. make_MBF_CF.py

11. setup.cfg
12. setup.py

- Description of the folder content:

1. & 2. folders referring to the compiled C libraries (will be created after executing "python setup.py install" command).

3. **SignalProcessing_modules/**: folder containing modules for calculating the CF and performing MBF time-frequency analysis. Consists of:

   c_libs/ - folder with C modules, implementing the kurtosis and RMS envelope recursive CFs, and MBF algorithm: *lib_rec_filter.c, lib_rec_kurtosis.c, lib_rec_rmc.c*

   rec_filter.py, rec_kurtosis.py, rec_rmc.py - python wrappers for corresponding C modules.

   Config.py, parse_config.py, configspec.conf - modules defining and describing the class of configuration parameters.

   mod_doCFcalculation.py, mod_doMBF_CFcalculation.py - modules performing the calculation of CFs and MBF analysis and broadband CFs calculation respectively (rely on rec_filter.py, rec_kurtosis.py, rec_rmc.py modules and related C libraries).

4. **data_example/**: two example data sets corresponding to one hour continuous records of seismic activity (XS.QF17.00.HHZ.D.2010.091) and tectonic tremor (N.TBEH.E.sac). First record is used for exemplifying signal processing scheme on kurtosis CF. Second - can be used for envelope CF examples

5. **figs_example/**: plots - output of the example runs of programs 9 and 10.

6. Readme - instructioin file

7. Documentation.doc/pdf - this file

8. **config_SigProc.tx**t - example control parameter file

9. **make_CF.py** - code for CF calculation. Reads parameters from example parameter file 8.

10. **make_MBF_CF.py** - code for MBF CF calculation. Reads parameters from example parameter file 8.

References:

Lomax, A., C. Satriano & Vassallo M., 2012. Automatic picker developments and optimization: FilterPicker - a robust, broadband picker for real-time seismic monitoring and earthquake early-warning, *SRL*., 83, 531-540.