

EtaGen v0.1

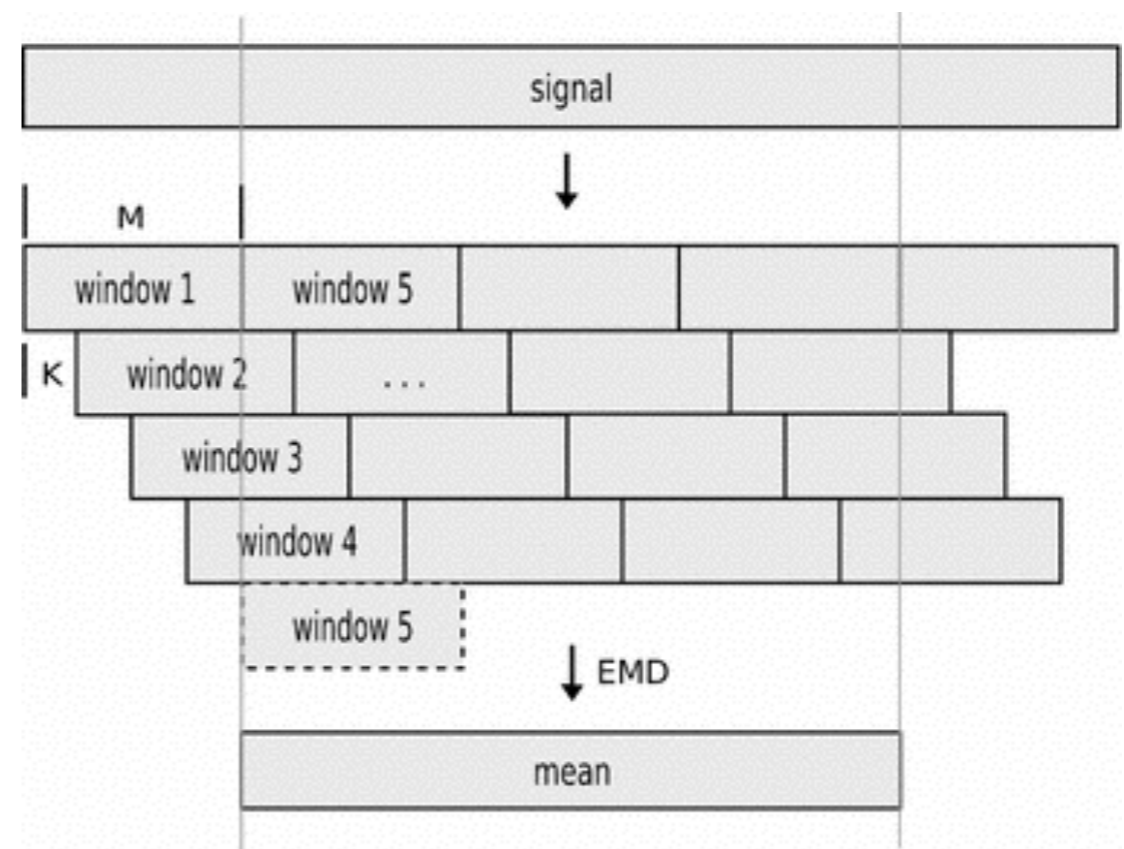
Edwin J. Son (NIMS)

Boot Camp Hands-on Tutorial on Dec. 9

What is EtaGen

- EtaGen is an event trigger generator based on Hilbert-Huang Transform (HHT)
- HHT = Empirical Mode Decomposition (EMD) + Hilbert Spectral Analysis (HSA) (for a review, see Huang et al., Rev. Geophys. 2008)
- To reduce computing cost, we are using weighted sliding EMD (wSEMD) instead of Ensemble EMD (EEMD)

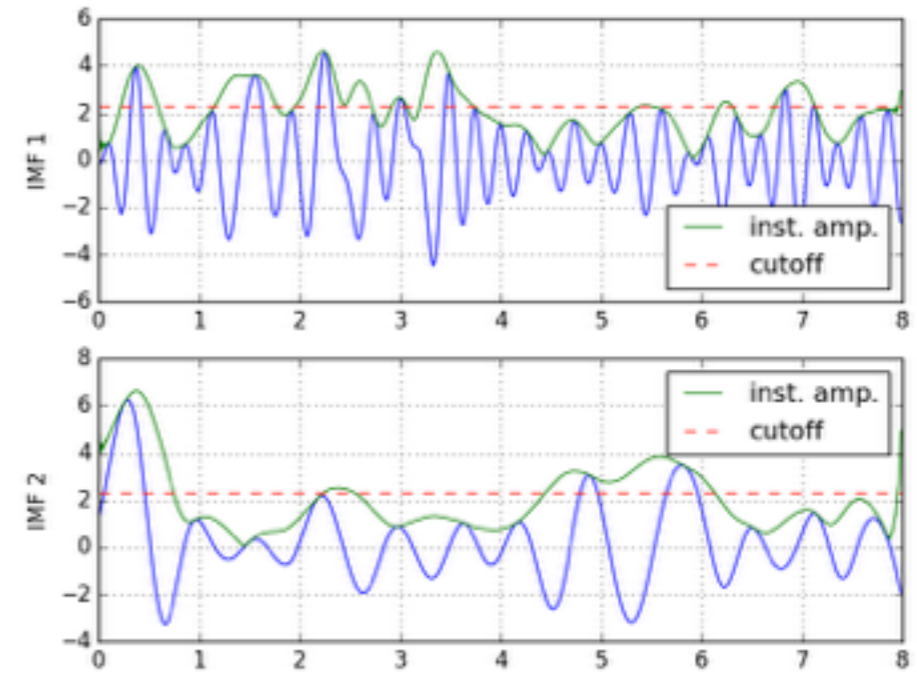
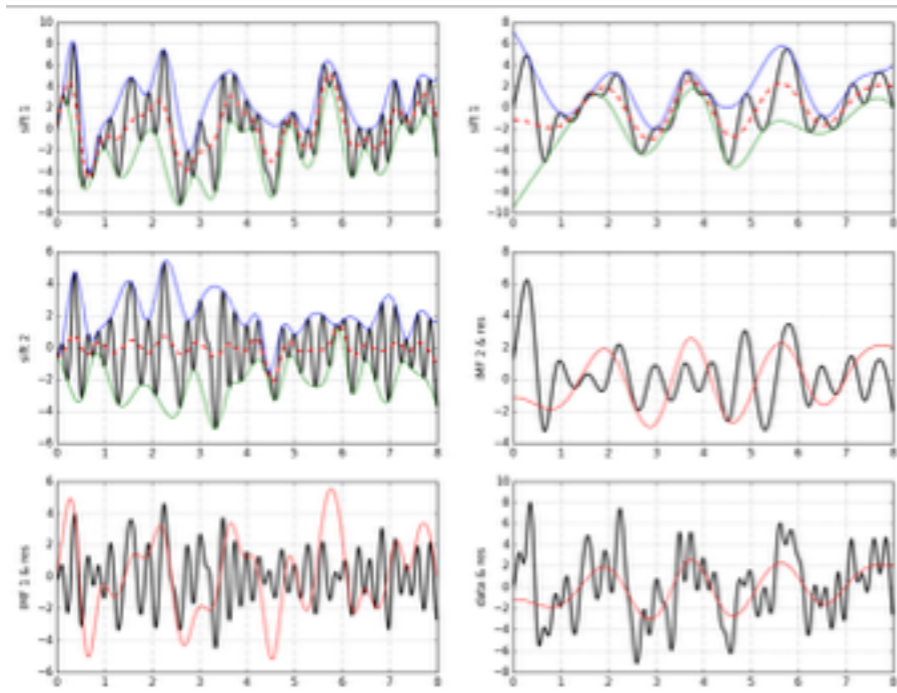
- Hopefully, EtaGen will be soon distributed to public



wSEMD Concept Image

(credit: R. Faltermeier et al. 2011)

How it works

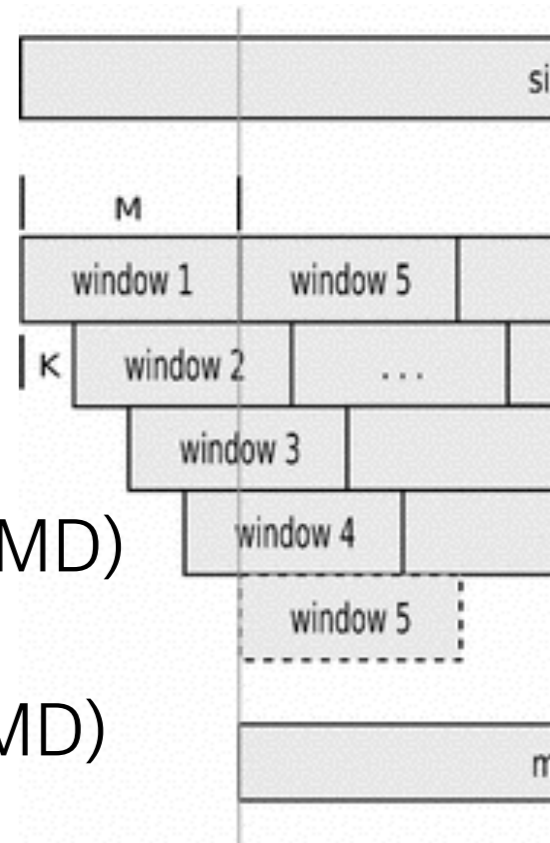


Initialize an instance

- import etagen as etg
- `h = etg.etagen(data, fsr=1.0, start_time=0.0)`
 - `h`: an instance of etagen class
 - `data`: input data which is assumed to be whitened (stored in `h.data`)
 - `fsr`: sampling frequency. If omitted, `fsr=1.0` by default (stored in `h.fsr`)
 - `start_time`: start time of data in sec. If omitted, `start_time=0.0` by default (stored in `h.start_time`)

Set Parameters for EMD or wSEDM

- `h.set_emd_param(num_imfs=0, num_sifts=15, S_number=0, emd_size=2048, num_seg=8, w_type=etg.kernel.SIN_KERNEL)`
 - `num_imfs`: Number of IMFs (stored in `h.nimfs`)
 - `num_sifts`: Maximum number of siftings
 - `S_number`: Criterion to stop sifting, $|N_{\text{ext}} - N_{\text{zero}}| \leq S$
 - `emd_size`: Number of samples in each segment (for wSEMD)
 - `num_seg`: Number of segments to average out (for wSEMD)
 - `w_type`: Type of weighting kernel (for wSEMD)



EMD or wSEMD

- You can use either `emd()` or `wsemd()` to decompose data into IMFs (stored in `h.imfs`)
- `h.emd()`
 - No arguments required
- `h.wsemd()`
 - No arguments required

Hands-on example #1

- Get the environment in the seikai server:

```
$ source ~detchar/initHasKAL.bashrc           # HasKAL environment
$ source ~eddy/bootcamp/etc/etagen-env.sh     # EtaGen environment
$ pip2.7 install --user numpy matplotlib     # required python module
```

- Run IPython:

```
$ ipython
Python 2.7.11 (default, Dec 7 2015, 11:22:37)
Type "copyright", "credits" or "license" for more information.

IPython 5.1.0 -- An enhanced Interactive Python.
?      -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help    -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

In [1]:
```

Hands-on #1 (cont'd)

- Import modules:

```
In [1]: import numpy as np
```

```
In [2]: from etagen import etagen
```

- Get IMFs of Gaussian random noise via wSEMD:

```
In [3]: Fs = 1024
```

```
In [4]: noise = np.random.randn(65536)
```

```
In [5]: h = etagen(noise, fsr=Fs)
```

```
In [6]: h.set_emd_param(num_imfs=8, num_seg=4)
```

```
In [7]: h.wsemd()
```

```
In [8]: print "# of IMFs: {0}, # of NaNs: {1}".format(h.nimfs,  
h.imfs[np.isnan(h.imfs)].shape[0])
```

```
# of IMFs: 8, # of NaNs: 0
```


HSA

- `h.hilbert(filter_length=128, stride=256)`
 - `filter_length`: Length of FIR filter for DHT
 - `stride`: Number of samples to estimate instantaneous frequency
 - Hilbert transformed IMFs will be stored in `h.hht`
 - Instantaneous amplitudes and frequencies of IMFs will be stored in `h.insa` and `h.insf`

Unclustered Triggers

- `h.get_utriggers(snr_th=3, stride=0, overlap=0, skip=0)`
 - `snr_th`: SNR threshold to generate triggers (stored in `h.min_snr`)
 - `stride`: Number of samples in a segment used to estimate background noise
 - `overlap`: Number of samples overlapped between segments
 - `skip`: Number of skipped samples in the edge of data
 - Generated triggers will be stored in `h.utrgs`

Clustered Triggers

- `h.get_triggers(snr_th=10, t_tolerance=0.0, f_tolerance=0.0, u_snr_th=3)`
 - `snr_th`: SNR threshold to generate trigger clusters (stored in `h.snr_threshold`)
 - `t_tolerance`: time tolerance for clustering [sec]
 - `f_tolerance`: frequency tolerance for clustering [ratio]
 - `u_snr_th`: SNR threshold for unclustered triggers to be clustered (stored in `h.u_snr_threshold`)
 - Clustered triggers will be stored in `h.trgs`

Hands-on #1 (cont'd)

- Get triggers:

```
In [9]: h.hilbert()
```

```
In [10]: h.get_triggers(snr_th=5)
```

```
Generating triggers with 3-snr threshold in segments of length 65536, overlapping 0 samples and skipping 0 samples from boundaries...
```

```
... generated 3104 trigger event(s)
```

```
Clustering triggers of u_snr > 3 with time tolerance=0.0, frequency tolerance=0.0 and dropping clusters of snr < 5...
```

```
... generated 583 trigger cluster(s)
```

- Get some infos. of the max_snr trigger:

```
In [11]: indx = h.trgs['snr'].argmax()
```

```
In [12]: print "MAX SNR: {0:.2f} at {1:.2f} sec. and {2:.2f} Hz".format(h.trgs[indx]['snr'], h.trgs[indx]['p_time'], h.trgs[indx]['p_freq'])
```

```
MAX SNR: 15.92 at 17.85 sec. and 222.37 Hz
```

Hands-on example #2

- Add a sine-Gaussian signal over the noise and generate triggers:

```
In [13]: from bootcampHelper import sineGaussian
```

```
In [14]: signal_time = np.linspace(-1.5, 1.5, 3*Fs, endpoint=False)
```

```
In [15]: signal = 3 * sineGaussian(50, 100, signal_time)
```

```
In [16]: data = noise.copy()
```

```
In [17]: data[30*Fs:33*Fs] += signal
```

```
In [18]: g = etagen(data, Fs)
```

```
In [19]: # Follow the procedure of hands-on example #1 by using “g” instead of “h”
```

- Following the procedure of #1, you may see the loudest trigger near 31.5 sec.

A script for KAGRA data in the seikai server

- A script for KAGRA data (an example):

```
$ kagetagen.py --out-dir output --gps-start-time 1145621536 \  
> --gps-end-time 1145621600 --channel K1:LSC-MICH_CTRL_CAL_OUT_DQ \  
> --sampling-frequency 16384 --emd-method wsemd --number-of-imfs 10 \  
> --snr-threshold 5.5 /data/kagra/raw/full/K-K1_C-1145621[56]*.gwf
```

- This script use `FrameDataDump` in `HasKAL` to extract data
- You don't need to know Python to run this script
- [Hands-on example #3](#): find triggers in KAGRA data of channel `K1:LSC-MICH_CTRL_CAL_OUT_DQ` from `1145621536` to `1145621568` by using the above script

Some options for the script

- `-h, --help`
 - show the help message
- `--quiet`
 - reduce stdout messages
- `-o OUTDIR, --out-dir OUTDIR`
 - Directory path where output files will be located

- `-s GPS_START_TIME, --gps-start-time GPS_START_TIME`
 - GPS start time of KAGRA data in sec.
- `-e GPS_END_TIME, --gps-end-time GPS_END_TIME`
 - GPS end time of KAGRA data in sec.
- `-c CHANNEL, --channel CHANNEL`
 - Channel name
- `-f FSR, --sampling-frequency FSR`
 - Sampling frequency in Hz

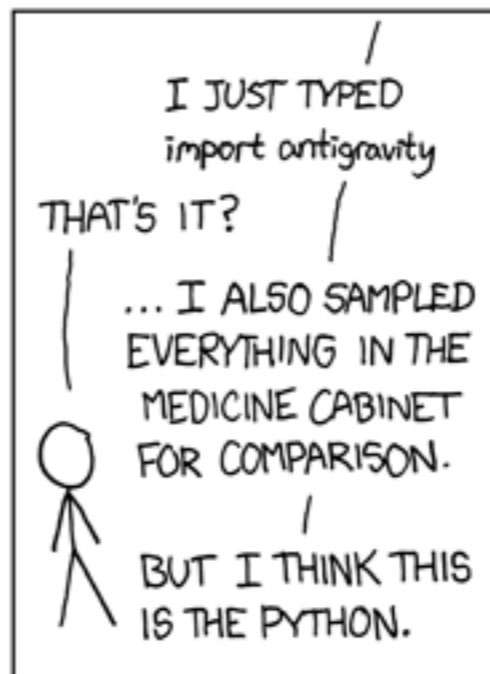
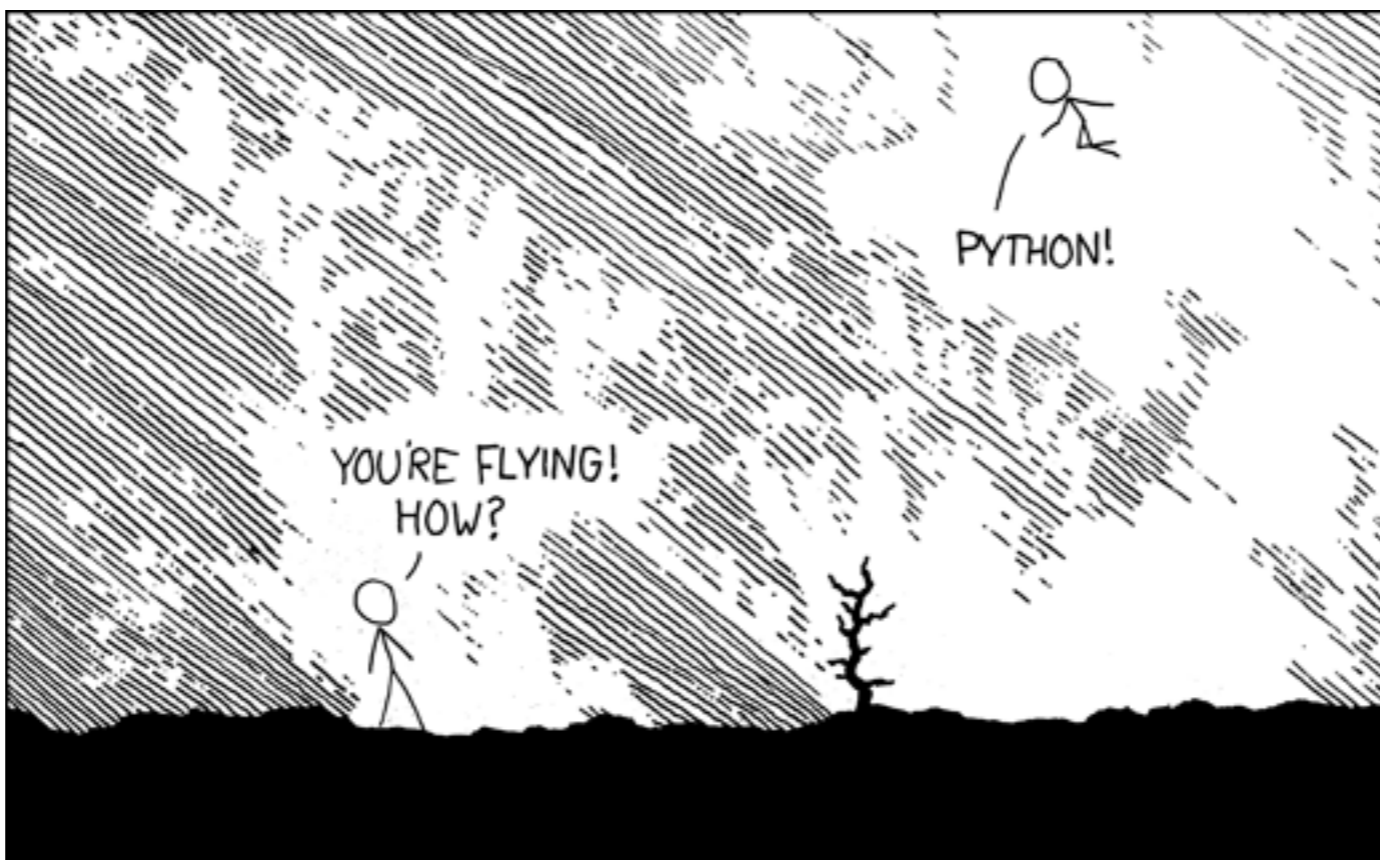
- `-m {emd,wsemd}, --emd-method {emd,wsemd}`
 - EMD method
- `-I NUM_IMFS, --number-of-imfs NUM_IMFS`
 - Number of IMFs
- `-T SNR, --snr-threshold SNR`
 - SNR threshold for trigger generation
- `-G, --plot-trigger`
 - If set, trigger distribution will be plotted

Trigger file from the script

- Filename convention:
TRG_{channel name}-{start time}-{duration}.txt
- An example of trigger file:

```
# s_time    e_time c_time c_freq c_energy    p_time p_freq p_amp p_imf_idx  
p_snr f_min f_max npts snr  
  
1145621549.39 1145621549.64 1145621549.5 32.0978011138 0.79732707563 1145621549.46  
30.8421454391 0.90591291609 6 18.0275844935 5.35106740598 126.819252762 6  
24.2931992713  
  
1145621555.14 1145621555.23 1145621555.17 168.530469161 0.435129056977 1145621555.16  
349.234736403 0.759307323296 3 3.96260428079 39.0329795337 405.324007045 5  
7.50064722872  
  
1145621553.49 1145621553.72 1145621553.56 96.3266691716 0.225746843968  
1145621553.51 90.8877928668 0.199682669844 5 3.17691573596 28.5208729623  
271.918467305 6 7.39301000967
```

お疲れ様
でした



[<https://xkcd.com/353/>]

Thank you

Scatter plot of triggers

