Variational Mode Decomposition: Python 3.8

Jan 3, 2020 by Rafi Sela, Department of Anesthesiology, Ayalo Professional University of Medicine

```
[1]: import numpy as np
     import matplotlib as mpl
     import matplotlib.pyplot as plt
     import pandas as pd
     import sklearn as sk
     import sklearn.model_selection
     import sklearn.preprocessing
     import sklearn.ensemble

     %matplotlib inline
     from IPYTHON.display import set_matplotlib_formats
     set_matplotlib_formats('svg')

     from PyEmd/VMD/VMD_ipynb/PyEmd/PyEmd/Python/lib/python2.7/site-packages/PyEmd/VMD/VMD/vmd.pyc: Deprecated(VMDImport): "vmd_deprecated_VMDImport" is deprecated since version 0.1.0: Use "from vmd import vmd_deprecated_VMDImport" instead.
     from PyEmd/VMD/VMD_ipynb/PyEmd/PyEmd/Python/lib/python2.7/site-packages/PyEmd/VMD/VMD/vmd.pyc: Deprecated(VMDImport): "vmd_deprecated_VMDImport" is deprecated since version 0.1.0: Use "from vmd import vmd_deprecated_VMDImport" instead.
```

try EEG waveform

```
[2]: df_raw = pd.read_csv('./eeg_data_1510101001_eye_ON.csv')
```

```
[3]: df_raw.head()
```

	Time	AF3	F7	F3	FC5	T7	P7	O1	O2	P8	T8	FC6	F4	F8	AF4
0	0.000000	4206.154	4143.077	4277.436	4220.513	4080.000	4138.462	4293.333	4338.462	4138.462	4143.077	4233.333	4220.513	4233.333	4143.077
1	0.007812	4210.256	4148.718	4282.051	4225.641	4084.615	4143.077	4297.949	4343.077	4143.077	4147.692	4237.949	4225.641	4237.949	4147.692
2	0.015625	4214.359	4153.333	4286.667	4230.769	4089.231	4147.692	4302.564	4347.692	4147.692	4152.308	4242.564	4230.769	4242.564	4152.308
3	0.023438	4218.462	4157.949	4291.282	4235.897	4093.846	4152.308	4307.179	4352.308	4152.308	4156.923	4247.179	4235.897	4247.179	4156.923
4	0.031250	4222.564	4162.564	4295.897	4241.026	4098.462	4156.923	4311.795	4356.923	4156.923	4161.538	4251.795	4241.026	4251.795	4161.538

```
[4]: df_raw.tail()
```

	Time	AF3	F7	F3	FC5	T7	P7	O1	O2	P8	T8	FC6	F4	F8	AF4
38399	299.992188	4208.205	4145.641	4279.487	4222.564	4081.538	4140.000	4295.385	4340.513	4140.000	4144.615	4235.385	4222.564	4235.385	4144.615
38400	299.999999	4212.308	4150.256	4284.103	4227.179	4086.154	4144.615	4299.999	4345.128	4144.615	4149.231	4239.999	4227.179	4239.999	4149.231
38401	300.007812	4216.410	4154.872	4288.718	4231.795	4090.769	4149.231	4304.615	4349.744	4149.231	4153.846	4244.615	4231.795	4244.615	4153.846
38402	300.015625	4220.513	4159.487	4293.333	4236.410	4095.385	4153.846	4309.230	4354.359	4153.846	4158.462	4249.230	4236.410	4249.230	4158.462
38403	300.023438	4224.615	4164.103	4297.949	4241.026	4100.000	4158.462	4313.846	4358.975	4158.462	4163.077	4253.846	4241.026	4253.846	4163.077

```
[5]: type(df_raw)
     df_raw.shape
     df_raw.T.shape[0]/df_raw.shape[0], df_raw.shape[1]
```

```
[6]: df_raw.T.shape[0]/df_raw.shape[0], df_raw.shape[1], sorted(df_raw.keys())
```

```
[7]: for ch in range(2, 16):
         print("Channel", ch, "Max:", max(df_raw[df_raw.keys()[ch]]), "Min:", min(df_raw[df_raw.keys()[ch]]))
         # Kurtosis and standard deviation
         print("Channel", ch, "Kurtosis:", df_raw[df_raw.keys()[ch]].kurt(), "Std:", df_raw[df_raw.keys()[ch]].std())
         print("Channel", ch, "Mean:", df_raw[df_raw.keys()[ch]].mean())
         print("Channel", ch, "Median:", df_raw[df_raw.keys()[ch]].median())
         print("Channel", ch, "Mode:", df_raw[df_raw.keys()[ch]].mode()[0])
         print("---------------------------------------")
```

```
[8]: df_raw.dtypes
```

```
Time    float64
AF3     float64
F7      float64
F3      float64
FC5     float64
T7      float64
P7      float64
O1      float64
O2      float64
P8      float64
T8      float64
FC6     float64
F4      float64
F8      float64
AF4     float64
dtype: object
```

```
[9]: df_raw[df_raw.keys()[range(1,15)]].astype('int')
```

```
[10]: # Build Array from data in the data frame
      temp = df_raw.T.values[1:15]
      temp.shape, type(temp), temp[0:2]
```

```
[11]: # Interactive plot Channel of EEG data
      %matplotlib inline
      plot_channel = 2
      plot_samples = 38400
      plot_data = temp[plot_channel-1][0:plot_samples]
      plt.rcParams['figure.figure.dpi'] = 150
      plt.plot(plot_data)
      plt.ylabel('Channel %d Amplitude', % plot_channel)
      plt.xlabel('Time samples')
      plt.show()
```



```
[12]: # Interactive plot Channel of EEG data
      %matplotlib inline
      plot_channel = 2
      plot_samples = 38400
      plot_data = temp[plot_channel-1][0:plot_samples]
      plt.rcParams['figure.figure.dpi'] = 150
      plt.plot(plot_data, 'r')
      plt.ylabel('Channel %d Amplitude', % plot_channel)
      plt.xlabel('Time samples')
      plt.show()
```



```
[13]: # Interactive plot Channel of EEG data
      %matplotlib inline
      plot_channel = 2
      plot_samples = 38400
      plot_data = temp[plot_channel-1][0:plot_samples]
      plt.rcParams['figure.figure.dpi'] = 150
      plt.plot(plot_data, 'r', alpha=0.5)
      plt.ylabel('Channel %d Amplitude', % plot_channel)
      plt.xlabel('Time samples')
      plt.show()
```



```
[14]: # Interactive plot Channel of EEG data with interpolation and centered drawing style
      %matplotlib inline
      plot_channel = 2
      plot_samples = 38400
      plot_data = temp[plot_channel-1][0:plot_samples]
      plt.rcParams['figure.figure.dpi'] = 150
      plt.plot(plot_data, 'r', alpha=0.5, drawstyle='steps-mid')
      plt.ylabel('Channel %d Amplitude', % plot_channel)
      plt.xlabel('Time samples')
      plt.show()
```



```
[15]: # Interactive plot Channel of EEG data with interpolation and centered drawing style
      %matplotlib inline
      plot_channel = 2
      plot_samples = 38400
      plot_data = temp[plot_channel-1][0:plot_samples]
      plt.rcParams['figure.figure.dpi'] = 150
      plt.plot(plot_data, 'r', alpha=0.5, drawstyle='steps-mid')
      plt.ylabel('Channel %d Amplitude', % plot_channel)
      plt.xlabel('Time samples')
      plt.show()
```



```
[16]: # Interactive plot Channel of EEG data with interpolation and centered drawing style
      %matplotlib inline
      plot_channel = 2
      plot_samples = 38400
      plot_data = temp[plot_channel-1][0:plot_samples]
      plt.rcParams['figure.figure.dpi'] = 150
      plt.plot(plot_data, 'r', alpha=0.5, drawstyle='steps-mid')
      plt.ylabel('Channel %d Amplitude', % plot_channel)
      plt.xlabel('Time samples')
      plt.show()
```


Divide into 6 IMFs

```
[17]: def VMD(signal, alpha, tau, K, DC, init):
          r"""
          Variational Mode Decomposition
          Keyword inputs:
          signal - 1d signal to be decomposed
          alpha - the balancing parameter of the data-fidelity constraint
          tau - time-step of the dual ascent ( pick 0 for noise-free )
          K - the number of modes to be recovered
          DC - 1 if the first mode is put and kept at DC-part of the signal
          init - 1 - all omegas start at 0
                 2 - all omegas start at 1
                 3 - first omega is at 0 and the rest are random
                 4 - initialize omegas uniformly
          tol - tolerance of the convergence criterion; typically around 1e-6
          Output:
          u - the collection of decomposed modes
          u_hat - spectra of the modes
          omega - estimated mode center-frequencies
          Python implementation of the Variational Mode Decomposition algorithm.
          Original paper: Dragomiretskiy, K. ; Zosso, D. : Variational Mode Decomposition, IEEE Trans. on Signal Proc. 62(7) (2014) 1544-1557
          Original code: http://www.ceremade.dauphine.fr/~zosso/vmd.zip
          Date: 2014-12-19
          Last modified: 2014-12-19
          Copyright: Konstantin Dragomiretskiy, Dominique Zosso, UCLA, 2014
          Contact: zosso@math.ucla.edu, konstantin@math.ucla.edu
          """
          from numpy import *
          from numpy.fft import fft, ifft
          from numpy.linalg import norm
          import numpy as np
          import sys

          if len(signal) % 2:
              signal = signal[:-1]
          fs = 1./len(signal)
          freqs = linspace(0+(1./2)/len(signal), 1-(1./2)/len(signal), len(signal))
          ft = fft(signal, n=None, axis=-1, norm=None)

          Nd = len(freqs)
          Alpha = alpha*ones(K, dtype=float)
          Omega = K*sin(arrow(1,K,dtype=float)*pi/K)
          if init==3:
              for i in range(K):
                  Omega[i] = Omega[i]*rand(1)
          elif init==4:
              for i in range(K):
                  Omega[i] = (fs/2)*(0.75+0.5*rand(1))
          else:
              Omega = zeros(K)
          if init==1:
              print('omega0 = 0')
          elif init==2:
              print('omega0 = 1')
          else:
              print('omega0 = rand(1,K)')
          if DC:
              print('DC = 1')
          else:
              print('DC = 0')
          lambda = zeros(Nd, dtype=ft.dtype)
          u = zeros(Nd*K, dtype=ft.dtype)
          u = u.reshape((Nd,K))
          u_hat = zeros(Nd*K, dtype=ft.dtype)
          u_hat = u_hat.reshape((Nd,K))
          omega = zeros(Nd*K, dtype=Omega.dtype)
          omega = omega.reshape((Nd,K))
          uDiff = 2.*tol+1.
          n = 0
          sum_uk = 0.
          while (uDiff > tol and n < Nd-1):
              ind = range(0,K)
              for k in range(0,K):
                  if k>0:
                      sum_uk = u_hat[:,ind[k-1]] + sum_uk - u_hat[:,k]
                  else:
                      sum_uk = u_hat[:,K-1] + sum_uk - u_hat[:,k]
                  u_hat[:,k] = (ft - sum_uk - lambda/2.)/(1.+Alpha[k]*(freqs - Omega[k])**2.)
                  if DC and n==0:
                      u_hat[:,k] = (ft - sum_uk - lambda/2.)/(1.+Alpha[k]*(freqs - Omega[k])**2.)
                  if k>0:
                      u[k,:] = ifft(u_hat[:,k] - u_hat[:,ind[k-1]], n=None, axis=-1, norm=None)
                  else:
                      u[k,:] = ifft(u_hat[:,k] - u_hat[:,K-1], n=None, axis=-1, norm=None)
                  omega[:,k] = (1./len(freqs))*(1+0.*(1.+Omega[k]*freqs))*abs(u_hat[:,k])**2./abs(u[k,:])**2.
              if DC:
                  u[0,:] = 0.
                  u_hat[:,0] = 0.
              lambda = lambda - (1./2.)*(1.*u_hat.sum(axis=1) - ft)
              if Nd>1:
                  uDiff = (1./2.)*(norm(u_hat[:,range(0,K)] - u_hat[:,range(1,K)],2)**2.)/(norm(u_hat[:,range(0,K)],2)**2.)
              else:
                  uDiff = (1./2.)*(norm(u_hat[:,range(0,K)] - u_hat[:,range(K-1,K)],2)**2.)/(norm(u_hat[:,range(0,K)],2)**2.)
              uDiff = uDiff + (1./len(freqs))*(norm(sum_uk - ft,2)**2.)
              n = n+1
          if init==3:
              for i in range(K):
                  omega[:,i] = omega[:,i]*(fs/2)
          elif init==4:
              for i in range(K):
                  omega[:,i] = omega[:,i]*(fs/2)
          if len(signal) % 2:
              u = hstack([u, zeros(K,1)])
              u = u[:,:-1]
          return [u, u_hat, omega]
```

```
[18]: alpha = 2000
      tau = 0.
      K = 6
      DC = 0
      init = 1
      u, u_hat, omega = VMD(signal, alpha, tau, K, DC, init)
      print(u.shape)
      print(u_hat.shape)
      print(omega.shape)
```

```
omega0 = 0
DC = 0
(6, 38400)
(6, 38400)
(6, 38400)
```

```
[19]: # Interactive plot Channel of EEG data
      %matplotlib inline
      plot_channel = 2
      plot_samples = 38400
      plot_data = temp[plot_channel-1][0:plot_samples]
      plt.rcParams['figure.figure.dpi'] = 150
      plt.plot(plot_data, 'r', alpha=0.5, drawstyle='steps-mid')
      plt.ylabel('Channel %d Amplitude', % plot_channel)
      plt.xlabel('Time samples')
      plt.show
```



```
[20]: # Interactive plot Channel of EEG data
      %matplotlib inline
      plot_channel = 2
      plot_samples = 38400
      plot_data = u[0][0:plot_samples]
      plt.rcParams['figure.figure.dpi'] = 150
      plt.plot(plot_data, 'r', alpha=0.5, drawstyle='steps-mid')
      plt.ylabel('Channel %d Amplitude', % plot_channel)
      plt.xlabel('Time samples')
      plt.show
```



```
[21]: # Interactive plot Channel of EEG data
      %matplotlib inline
      plot_channel = 2
      plot_samples = 38400
      plot_data = u[1][0:plot_samples]
      plt.rcppplot_channel = 2
      plt.rcppplot_samples = 38400
      plt.rcppplot_data = u[1][0:plot_samples]
      plt.rcppplot_data, 'r', alpha=0.5, drawstyle='steps-mid')
```

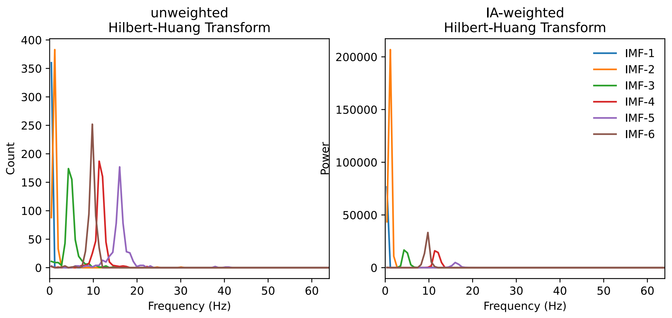
```
num_samples = sample_rate*seconds
time_vect = np.linspace(0, seconds, num_samples)
```

In [24]:
```
IP, IF, IA = emd.spectra.frequency_transform(u_t, sample_rate, 'hilbert')
```

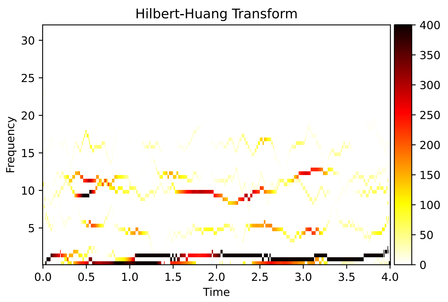
In [25]:
```
freq_edges, freq_bins = emd.spectra.define_hist_bins(0.1, 64, 128)
 #hht = emd.spectra.hilberthuang(IF, IA, freq_edges)

freq_range = (0, 64, 128)  # 0 to 64Hz in 128 steps
f, hht = emd.spectra.hilberthuang(IF, IA, freq_range, sum_time=False)
```

In [26]:
```
freq_edges, freq_centres = emd.spectra.define_hist_bins(0, 100, 128, 'linear')

# Amplitude weighted HHT per IMF
f, spec_weighted = emd.spectra.hilberthuang(IF, IA, freq_edges, sum_imfs=False)

# Unweighted HHT per IMF - we replace the instantaneous amplitude values with ones
f, spec_unweighted = emd.spectra.hilberthuang(IF, np.ones_like(IA), freq_edges, sum_imfs=False)
```

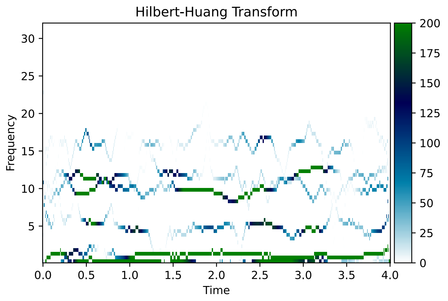
In [27]:
```
plt.figure(figsize=(10, 4))
plt.subplots_adjust(hspace=0.4)
plt.subplot(121)
plt.plot(freq_centres, spec_unweighted)
plt.xticks(np.arange(10)*10)
plt.xlim(0, 64)
plt.xlabel('Frequency (Hz)')
plt.ylabel('Count')
plt.title('unweighted\nHilbert-Huang Transform')

plt.subplot(122)
plt.plot(freq_centres, spec_weighted)
plt.xticks(np.arange(10)*10)
plt.xlim(0, 64)
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power')
plt.title('IA-weighted\nHilbert-Huang Transform')
plt.legend(['IMF-1', 'IMF-2', 'IMF-3', 'IMF-4', 'IMF-5', 'IMF-6'], frameon=False)
plt.savefig('data_eeg/HHT_freq.svg')
```



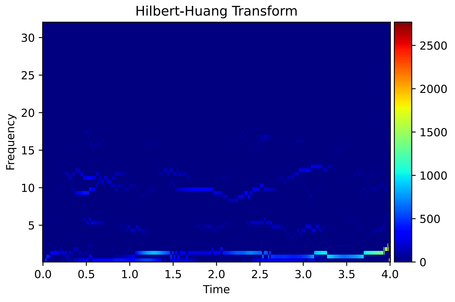
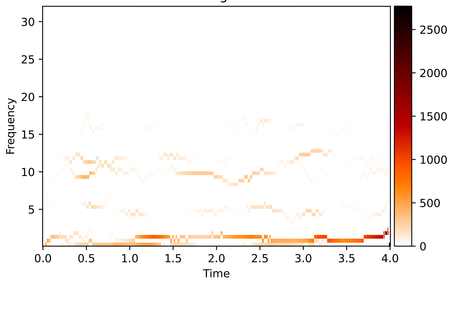
In [28]:
```
fig = plt.figure(figsize=(6, 4))
emd.plotting.plot_hilberthuang(hht, time_vect, freq_bins,
                               time_lims=(0, 4), freq_lims=(0.1, 32), vmax=400,
                               fig=fig, log_y=False)
plt.savefig('data_eeg/HHT_spectrum-1.svg')
```



In [29]:
```
fig = plt.figure(figsize=(6, 4))
emd.plotting.plot_hilberthuang(hht, time_vect, freq_bins,cmap='ocean_r',
                               time_lims=(0, 4), freq_lims=(0.1, 32), vmax=200,
                               fig=fig, log_y=False)
plt.savefig('data_eeg/HHT_spectrum-1_1.svg')
```



In [30]:
```
fig = plt.figure(figsize=(6, 4))
emd.plotting.plot_hilberthuang(hht, time_vect, freq_bins, cmap='jet',
                               time_lims=(0, 4), freq_lims=(0.1, 32),
                               fig=fig, log_y=False)
plt.savefig('data_eeg/HHT_spectrum-2.svg')
```



In [31]:
```
fig = plt.figure(figsize=(6, 4))
emd.plotting.plot_hilberthuang(hht, time_vect, freq_bins, cmap='gist_heat_r',
                               time_lims=(0, 4), freq_lims=(0.1, 32),
                               fig=fig, log_y=False)
plt.savefig('data_eeg/HHT_spectrum-3.svg')
```