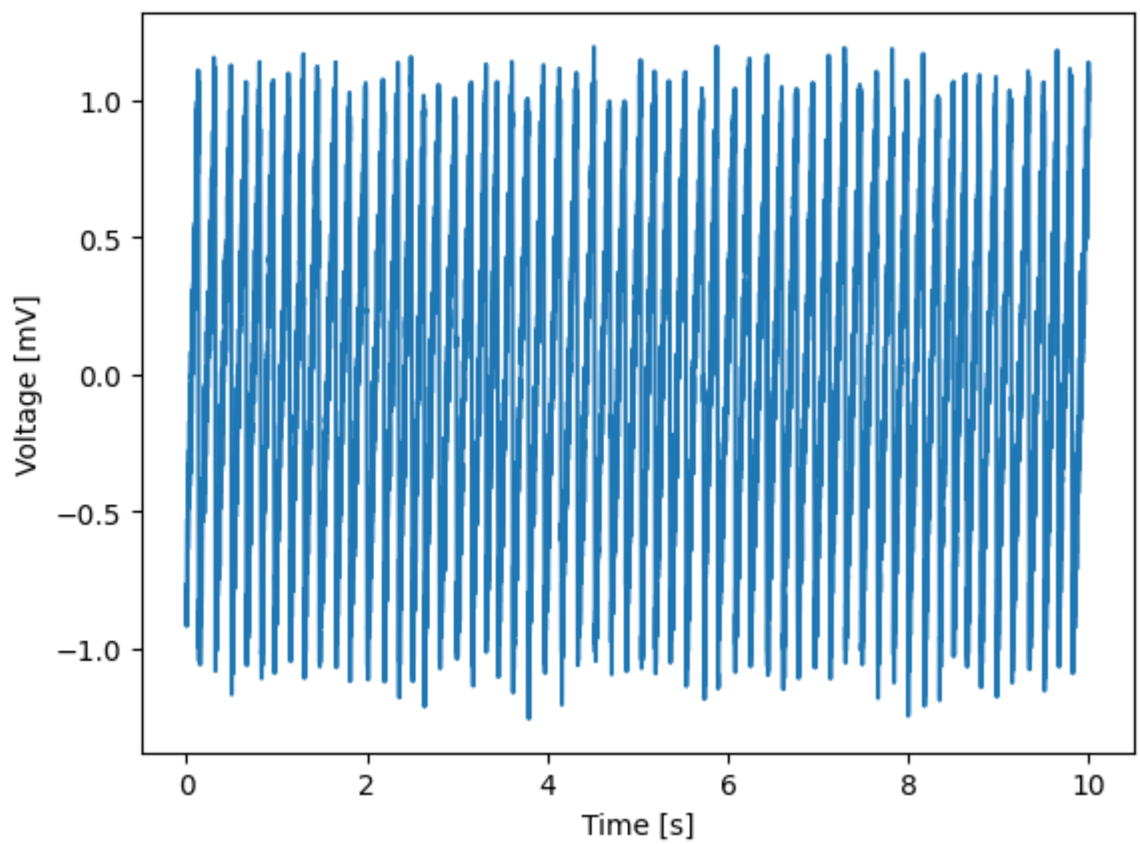
2.) LFP-3.mat

In [ ]:
```
from scipy.io import loadmat
import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import spectrogram
from scipy import signal
import random as rand
```

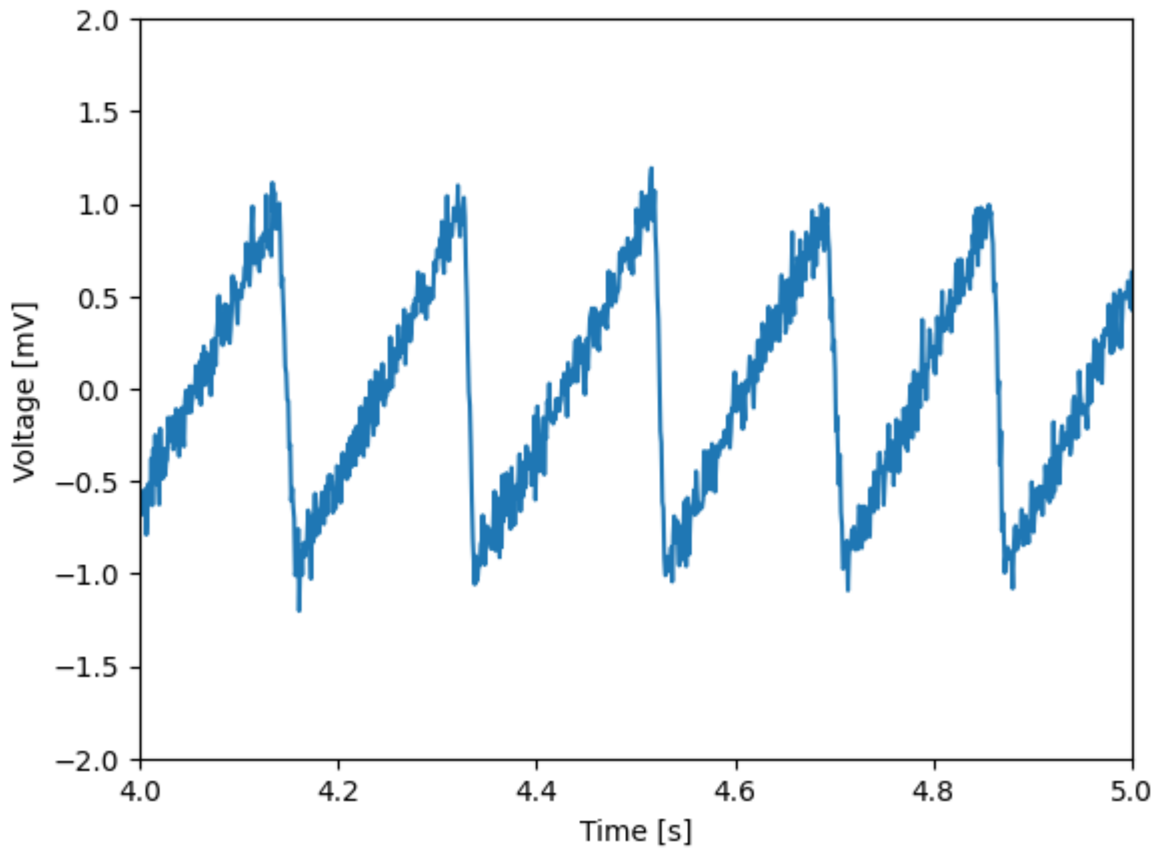
In [ ]:
```
# Load the data.
lfp2 = loadmat('LFP-3.mat')            # Load the data,
t = lfp2['t'][0]                       # ... extract t, the time variable,
LFP = lfp2['LFP'][0]                   # ... and LFP, the voltage variable.
dt = t[1] - t[0]                       # Define the sampling interval,
fNQ = 1 / dt / 2                       # ... and Nyquist frequency.
```

i. Visualize the time series data. What rhythms do you observe? Do you detect evidence for CFC in your visualizations?

In [ ]:
```
plt.plot(t,LFP)
plt.xlabel('Time [s]')        # ... with axes labeled.
plt.ylabel('Voltage [mV]');
```



In [ ]:
```
#zoom in on 1 second of activity
plt.plot(t,LFP)
plt.xlim([4,5])
plt.ylim([-2, 2])
plt.xlabel('Time [s]')      # ... with axes labeled.
plt.ylabel('Voltage [mV]');
plt.show()
```



what rhythmns are in this? is there any evidence of a CFC?

There is a low frequency rhythmn interspersed with smaller-amp blasts of high freq activity???

ii.) Plot the spectrum vs the frequency for this data

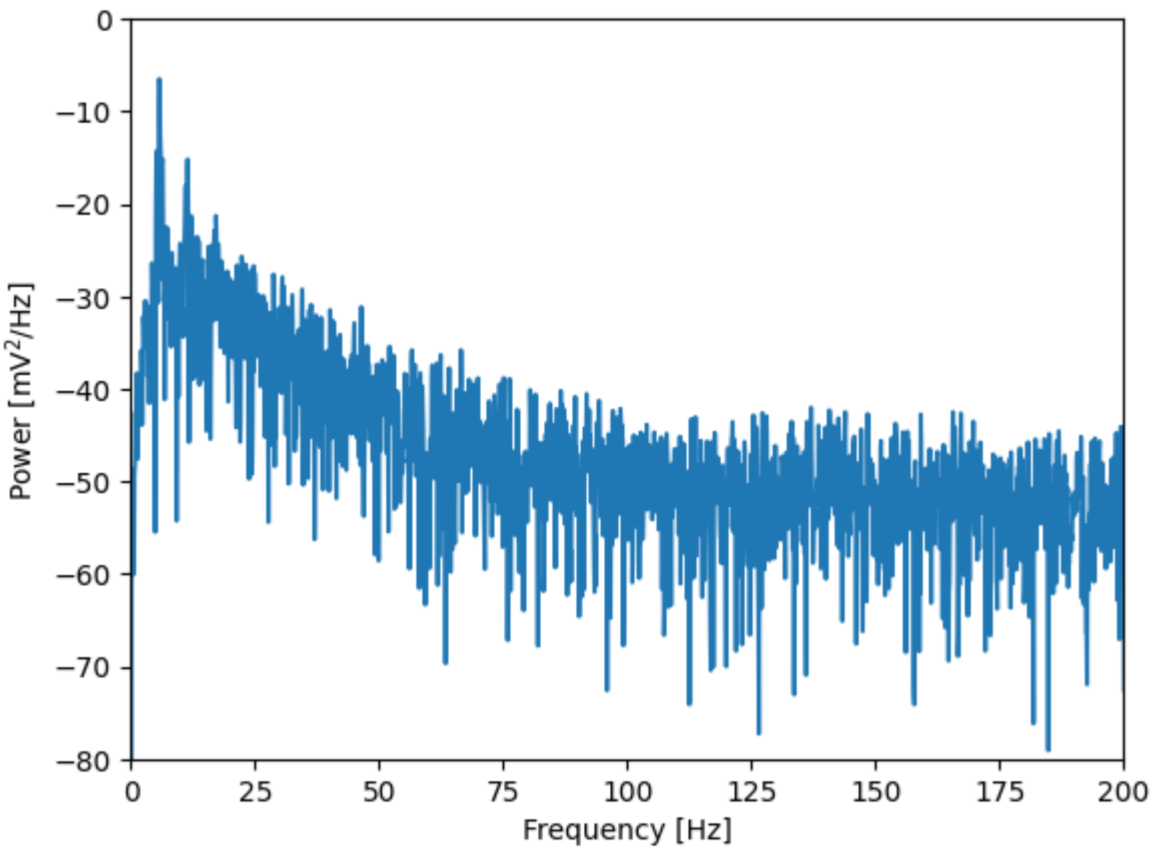
In [ ]:

```
dt = t[1] - t[0]                    # Define the sampling interval,
T = t[-1]                           # ... the duration of the data,
N = len(LFP)                        # ... and the no. of data points

x = np.hanning(N) * LFP               # Multiply data by a Hanning taper
xf = np.fft.rfft(x - x.mean())            # Compute Fourier transform
Sxx = 2*dt**2/T * (xf*np.conj(xf)) # Compute the spectrum
Sxx = np.real(Sxx)                    # Ignore complex components

df = 1 / T                          # Define frequency resolution,
fNQ = 1 / dt / 2                    # ... and Nyquist frequency.

faxis = np.arange(0, fNQ + df, df) # Construct freq. axis
plt.plot(faxis, 10 * np.log10(Sxx))     # Plot spectrum vs freq.
plt.xlim([0, 200])                      # Set freq. range,
plt.ylim([-80, 0])                      # ... andplt. decibel range
plt.xlabel('Frequency [Hz]')            # Label the axes
plt.ylabel('Power [mV$^2$/Hz]');
```



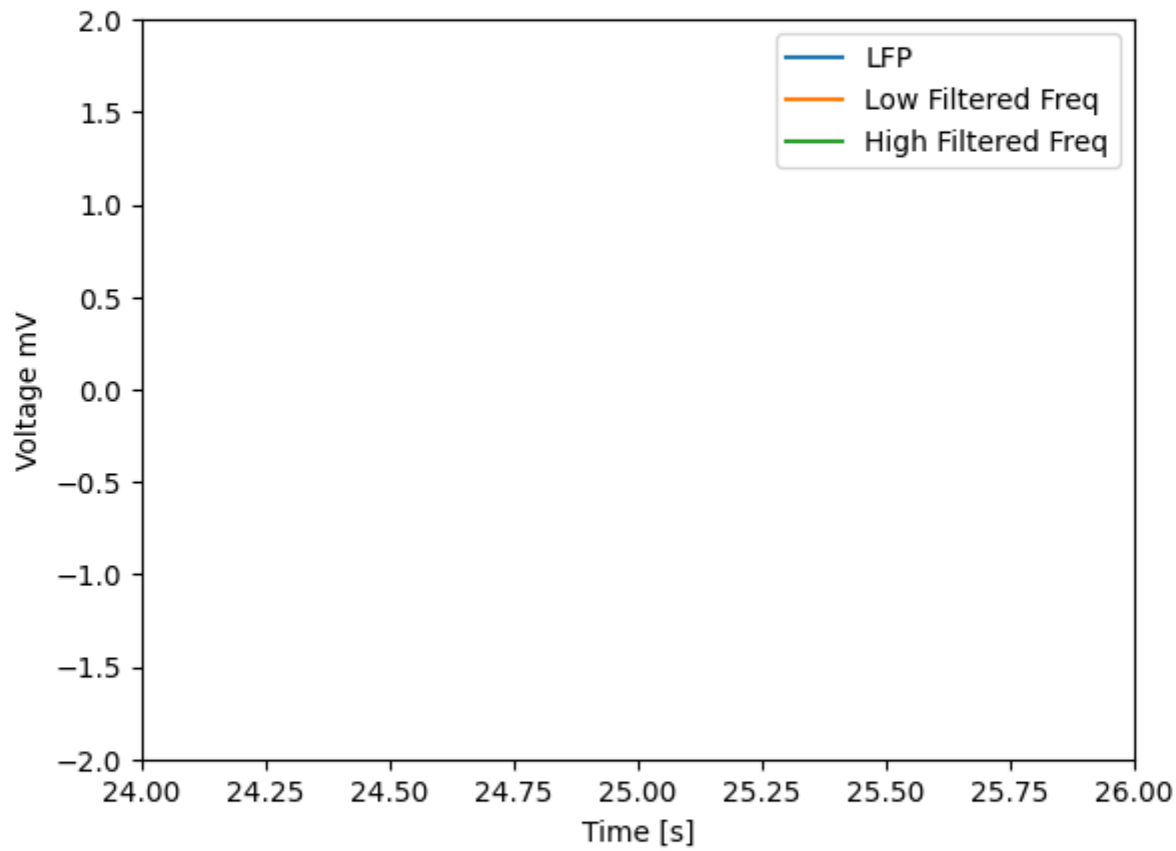
peak around 5? then something from like 25-200 hz....

iii.) APPLY the CFC method. What choices will you make, and why? What if any CFC do you find?

```
In [ ]: # Low frequency band. (the dominant)
        Wn = [5,7];                          # Set the passband for low frequency band
        n = 100;                               # ... and filter order,
                                               # ... build the bandpass filter,
        b = signal.firwin(n, Wn, nyq=fNQ, pass_zero=False, window='hamming');
        Vlo = signal.filtfilt(b, 1, LFP);    # ... and apply it to the data.

        # High frequency band.
        Wn = [75, 120];                        # Set the passband for high frequency band
        n = 100;                              # ... and filter order,
                                              # ... build the bandpass filter,
        b = signal.firwin(n, Wn, nyq=fNQ, pass_zero=False, window='hamming');
        Vhi = signal.filtfilt(b, 1, LFP);    # ... and apply it to the data.
```

```
In [ ]: plt.plot(t, LFP)
        plt.plot(t, Vlo) #plot the low peak
        plt.plot(t, Vhi) #plot the high freq peak
        plt.xlabel('Time [s]')
        plt.ylabel('Voltage mV')
        plt.xlim([24, 26]); #zoom in a specific time interval to get a more detailed graph,
        plt.ylim([-2,2]);
        plt.legend(['LFP', 'Low Filtered Freq', "High Filtered Freq"])
        plt.show()
```



why doesn't this work....

now we find h

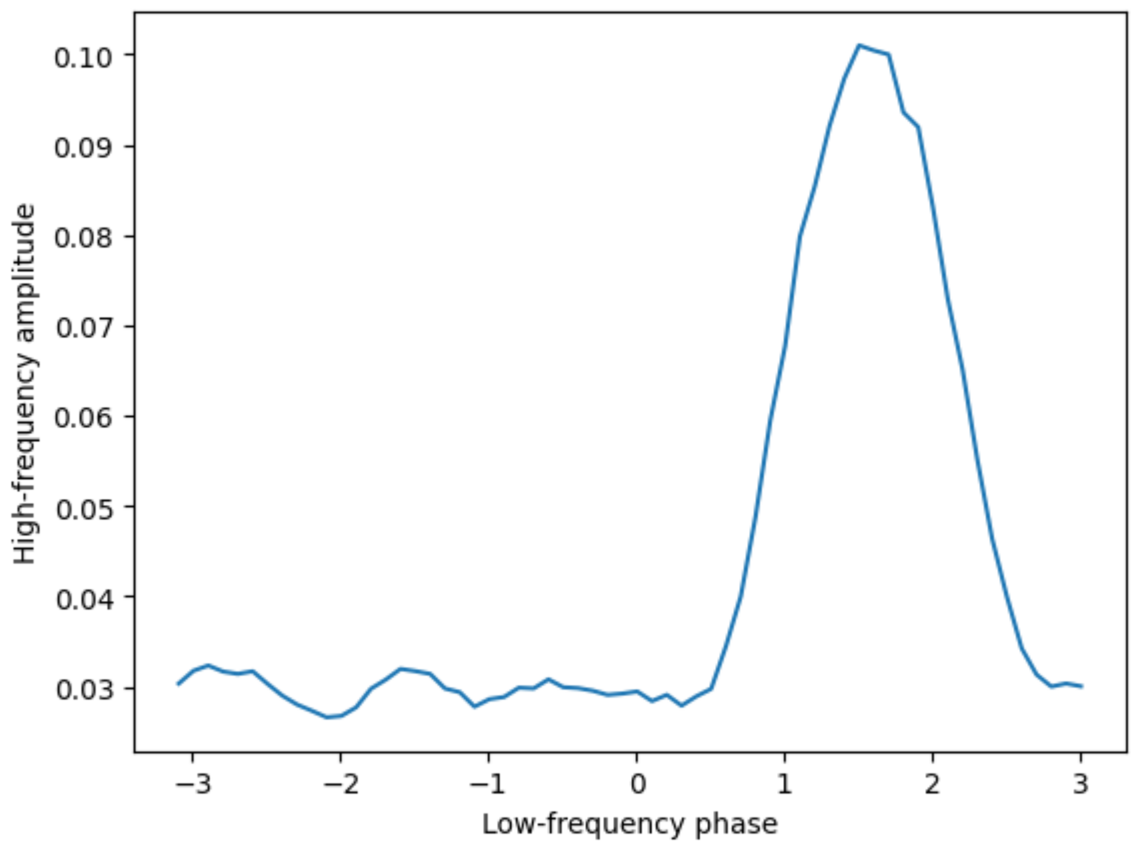
In [ ]:

```
phi = np.angle(signal.hilbert(Vlo))  # Compute phase of low-freq signal
amp = abs(signal.hilbert(Vhi))       # Compute amplitude of high-freq signal
```

In [ ]:

```
p_bins = np.arange(-np.pi, np.pi, 0.1)
a_mean = np.zeros(np.size(p_bins)-1)
p_mean = np.zeros(np.size(p_bins)-1)
for k in range(np.size(p_bins)-1):     #For each phase bin,
    pL = p_bins[k]                     #... lower phase limit,
    pR = p_bins[k+1]                   #... upper phase limit.
    indices=(phi>=pL) & (phi<pR)       #Find phases falling in bin,
    a_mean[k] = np.mean(amp[indices])  #... compute mean amplitude,
    p_mean[k] = np.mean([pL, pR])      #... save center phase.

plt.plot(p_mean, a_mean)                #Plot the phase versus amplitude,
plt.ylabel('High-frequency amplitude')  #... with axes labeled.
plt.xlabel('Low-frequency phase');
```



this suggests CFC to me

In [ ]:

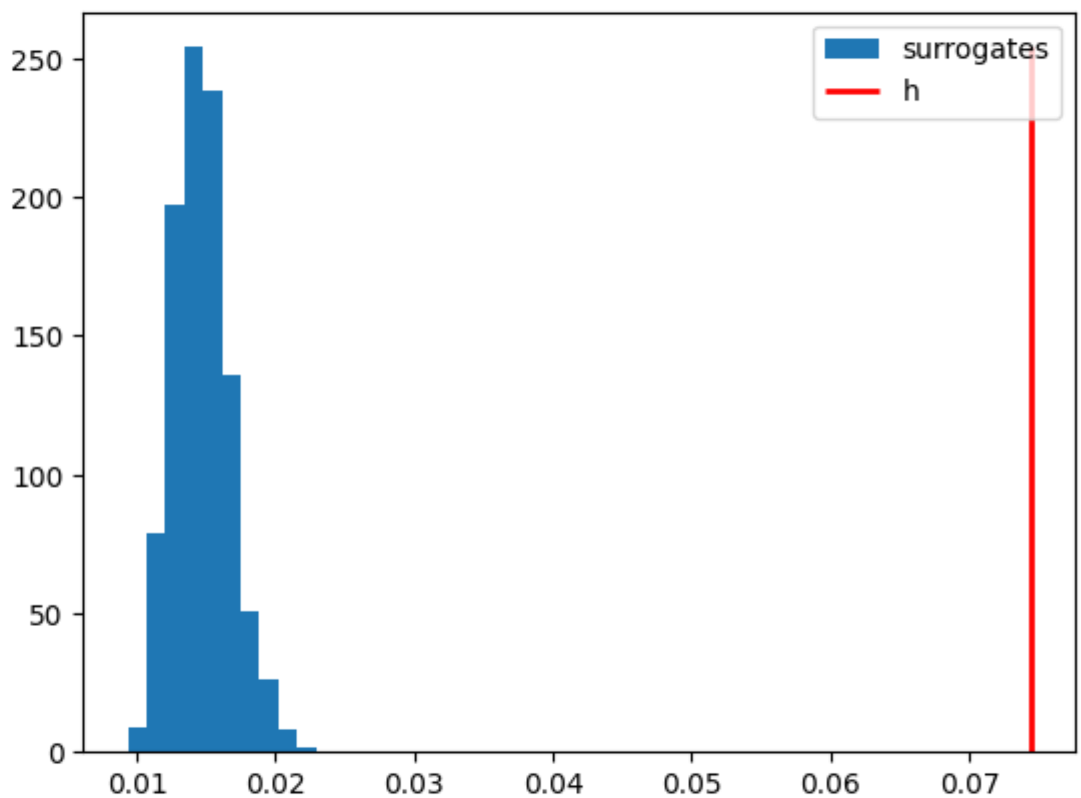
```
h = max(a_mean)-min(a_mean)
print(h)
```

```
0.07438676419936611
```

now we make surrogate h values to check h's signficance

In [ ]:
```
n_surrogates = 1000;                          #Define no. of surrogates.
hS = np.zeros(n_surrogates)                      #Vector to hold h results.
for ns in range(n_surrogates):                #For each surrogate,
    ampS = amp[np.random.randint(0,N,N)]             #Resample amplitude,
    p_bins = np.arange(-np.pi, np.pi, 0.1)         #Define the phase bins
    a_mean = np.zeros(np.size(p_bins)-1)        #Vector for average amps.
    p_mean = np.zeros(np.size(p_bins)-1)        #Vector for phase bins.
    for k in range(np.size(p_bins)-1):
        pL = p_bins[k]                        #... lower phase limit,
        pR = p_bins[k+1]                      #... upper phase limit.
        indices=(phi>=pL) & (phi<pR)          #Find phases falling in bin,
        a_mean[k] = np.mean(ampS[indices])  #... compute mean amplitude,
        p_mean[k] = np.mean([pL, pR])          #... save center phase.
    hS[ns] = max(a_mean)-min(a_mean)          # Store surrogate h.
```

In [ ]:
```
counts, _, _ = plt.hist(hS, label='surrogates')                # Plot the histogram
plt.vlines(h, 0, max(counts), colors='red', label='h', lw=2)  # Plot the observed h
plt.legend();
```



In [ ]:
```
p = sum([s > h for s in hS]) / len(hS) #what is the proportion of surrogante h valu
print(p)
```

```
0.0
```

iv.) Explain your results

We reject the null hypothesis of there being no CFC between the phase of the low frequency band and the amplitidue of the high frequency band. This suggets that there is CFC coupling between our datasets.