PS6

November 8, 2021

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import h5py
import json
from scipy import signal, interpolate
```

0.1 Question 1

0.1.1 a.)

[2]:
```
directory = 'LOSC_Event_tutorial/'

def read_template(filename):
    dataFile=h5py.File(directory+filename,'r')
    template=dataFile['template']
    th=template[0]
    tl=template[1]

    return th,tl

def read_file(filename):
    dataFile=h5py.File(directory+filename,'r')
    dqInfo = dataFile['quality']['simple']
    qmask=dqInfo['DQmask'][...]

    meta=dataFile['meta']
    #gpsStart=meta['GPSstart'].value
    gpsStart=meta['GPSstart'][()]
    #print meta.keys()
    #utc=meta['UTCstart'].value
    utc=meta['UTCstart'][()]
    #duration=meta['Duration'].value
    duration=meta['Duration'][()]
    #strain=dataFile['strain']['Strain'].value
    strain=dataFile['strain']['Strain'][()]
    dt=(1.0*duration)/len(strain)
```

```
    dataFile.close()

    return strain,dt,utc
```

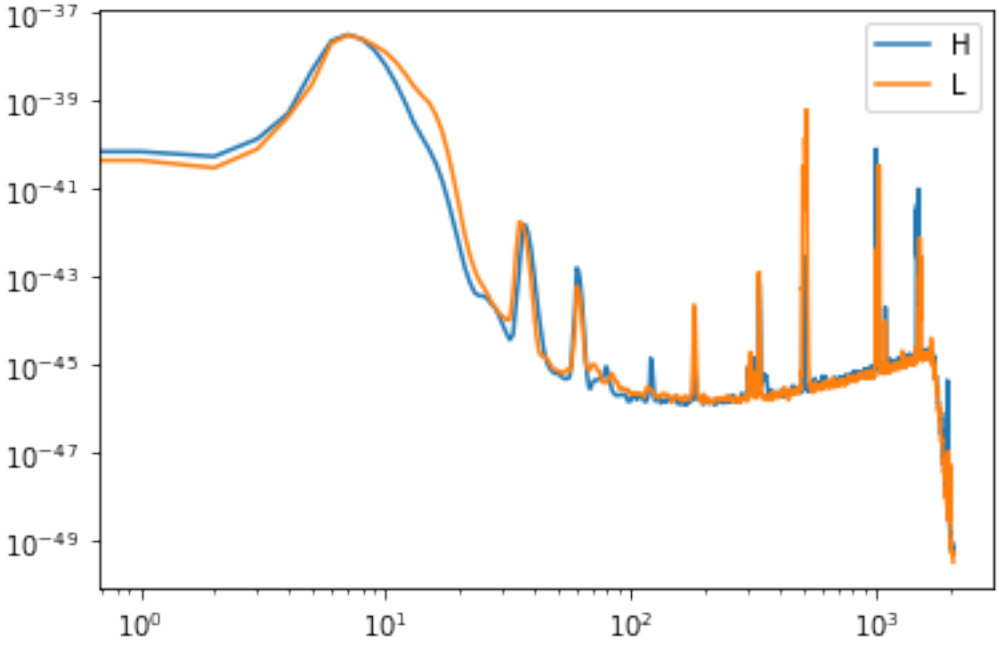
[3]:
```
with open(directory+'BBH_events_v3.json') as file:
    json_dat = json.load(file)
list_events = list(json_dat.keys())
H_event, L_event, H_template, L_template = [],[],[],[]

for i in range(len(list_events)):
    event = list_events[i]
    H = json_dat[event]['fn_H1']
    L = json_dat[event]['fn_L1']
    temp = json_dat[event]['fn_template']
    H_event.append(read_file(H))
    L_event.append(read_file(L))
    H_temp, L_temp = read_template(temp)
    H_template.append(H_temp)
    L_template.append(L_temp)
file.close()
```

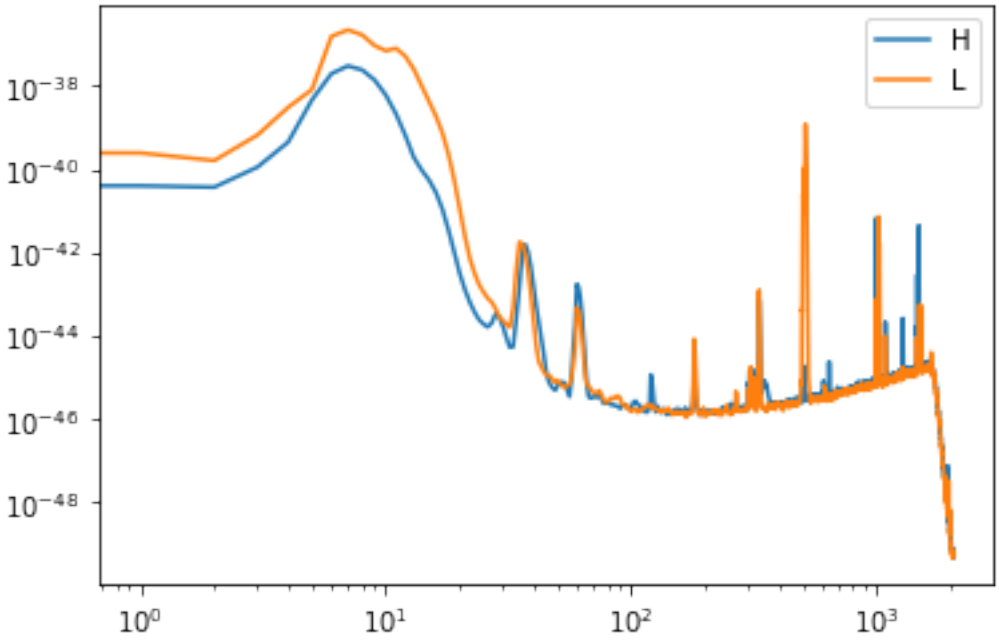
[12]:
```
def Smooth(x):
    fft = np.fft.fft(x)
    n = np.arange(len(x))
    fun = np.exp(-0.5 * (n/1.5)**2)
    funfft = np.fft.fft(fun)
    return np.abs(np.fft.ifft(fft * funfft))
```

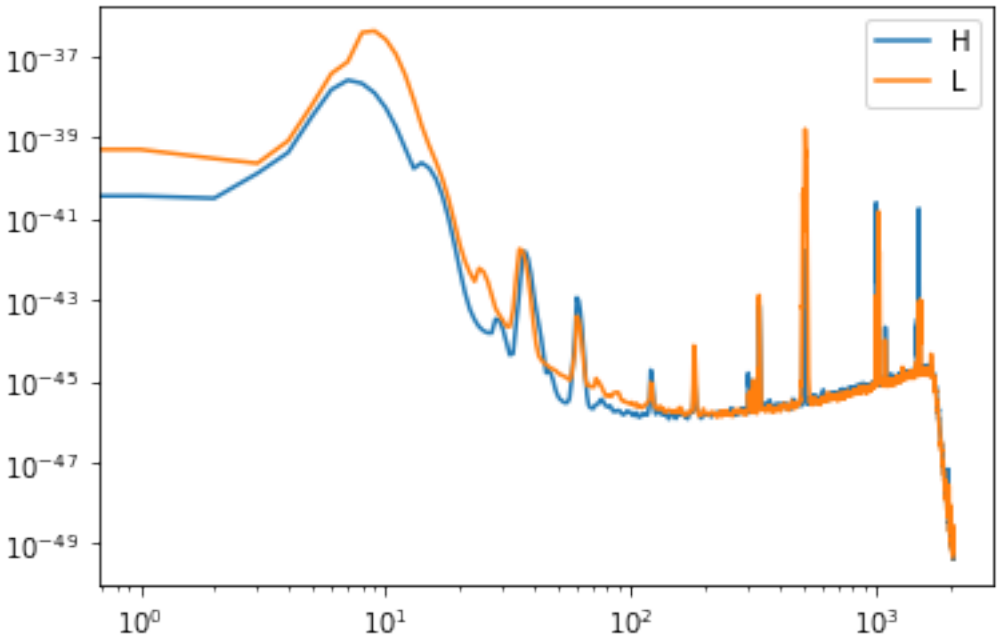
[22]:
```
def Noise():
    noise = []
    for i in range(len(L_event)):
        H_strain, H_dt = np.array(H_event, dtype = object).T[:2,i]
        L_strain, L_dt = np.array(L_event, dtype = object).T[:2,i]
        H_freq, H_pxx = signal.welch(H_strain, 1/H_dt, nperseg = 1/H_dt, window␣
 ↪= 'tukey')
        L_freq, L_pxx = signal.welch(L_strain, 1/L_dt, nperseg = 1/L_dt, window␣
 ↪= 'tukey')
        H_noise = Smooth(H_pxx)
        L_noise = Smooth(L_pxx)
        plt.loglog(H_noise, label = 'H')
        plt.loglog(L_noise, label = 'L')
        plt.legend()
        plt.show()
        noise.append({"H_noise": H_noise, "L_noise": L_noise,
                      "H_freq": H_freq, "L_freq": L_freq,
                      "H_strain": H_strain, "L_strain": L_strain,
                      "H_dt": H_dt, "L_dt": L_dt,
                      "H_template": H_template[i], "L_template": L_template[i]})
```

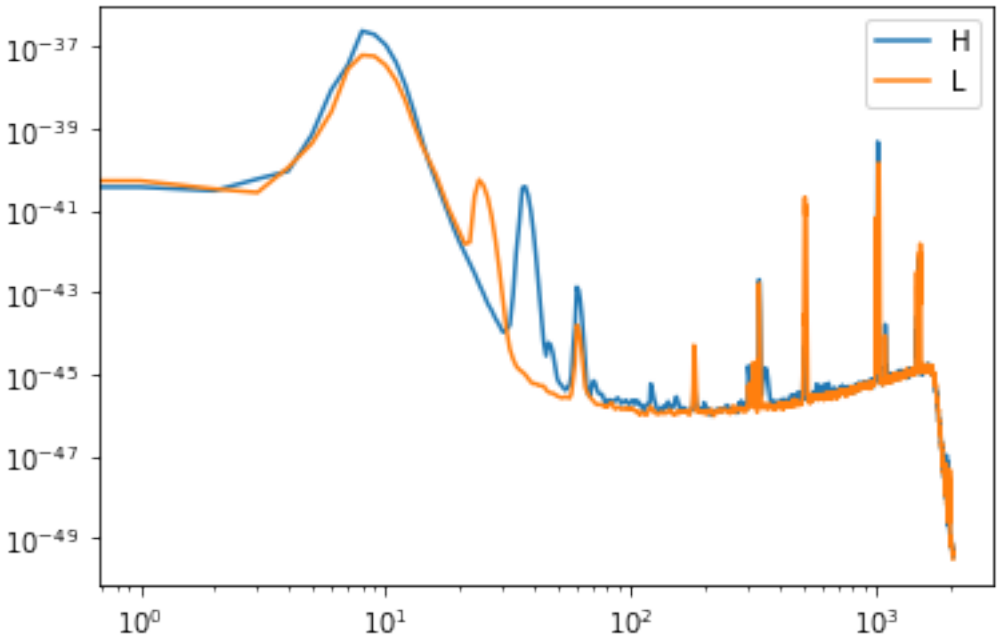
```
    return noise
```

[23]:
```
data = Noise()
```






H
L
10^{-37}
10^{-39}
10^{-41}
10^{-43}
10^{-45}
10^{-47}
10^{-49}
10^{0}
10^{1}
10^{2}
10^{3}

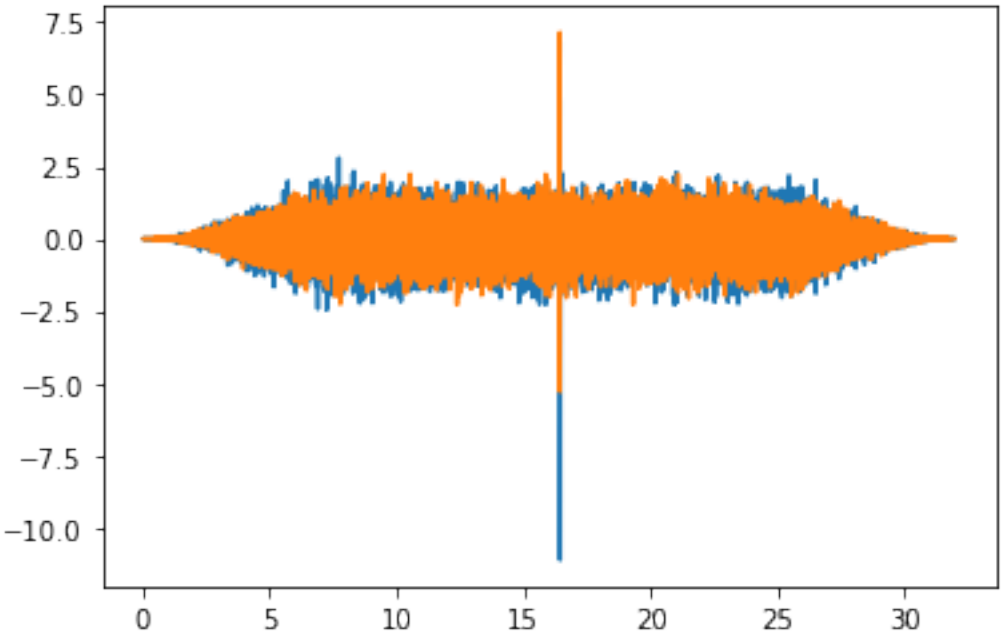

H
L
10^{-37}
10^{-39}
10^{-41}
10^{-43}
10^{-45}
10^{-47}
10^{-49}
10^{0}
10^{1}
10^{2}
10^{3}

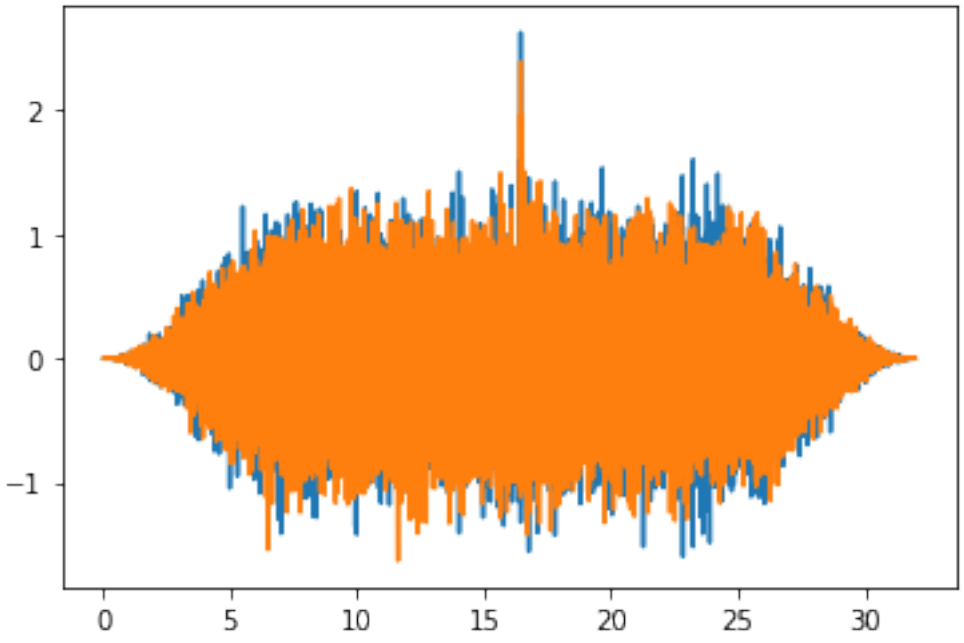
0.2 b.)

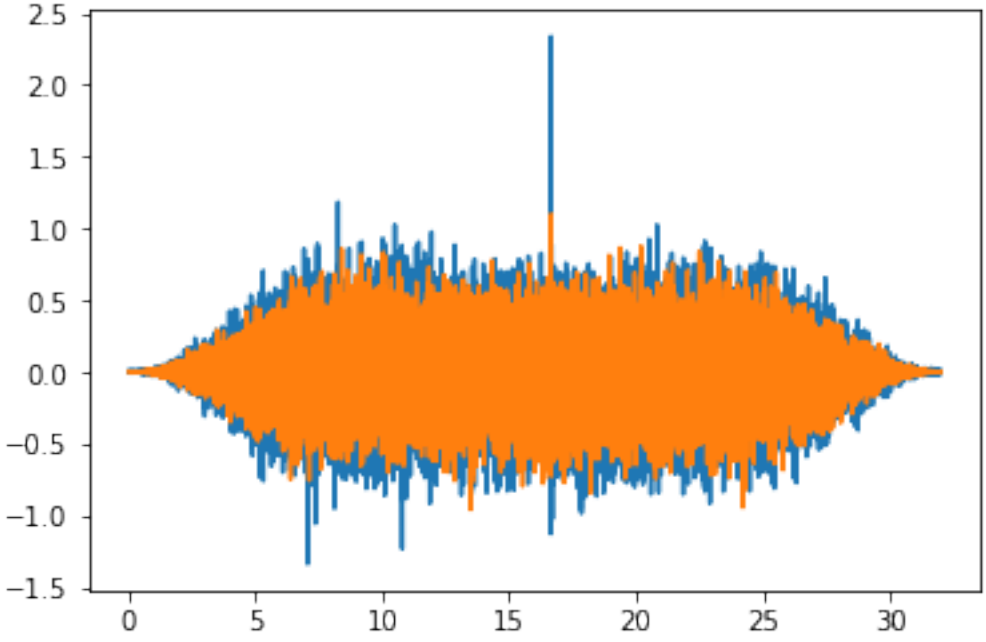
[24]:
```
def white(event):
    Hnoise_spectrum_intp = interpolate.
 ↪interp1d(event['H_freq'],event['H_noise'],kind='linear')
    Hspectr1 = np.fft.fft(event['H_strain'] * signal.get_window(window =␣
 ↪'tukey', Nx = len(event['H_strain'])))
    Hspectr2 = np.fft.fft(event['H_template'] * signal.get_window(window =␣
 ↪'tukey', Nx = len(event['H_template'])))

    Hfreq = np.fft.fftfreq(len(event['H_template']), event['H_dt'])

    Hspectr1 = Hspectr1/np.sqrt(Hnoise_spectrum_intp(np.abs(Hfreq)))
    Hspectr2 = Hspectr2/np.sqrt(Hnoise_spectrum_intp(np.abs(Hfreq)))
    Hwhite1 = np.fft.ifft(Hspectr1)
    Hwhite2 = np.fft.ifft(Hspectr2)

    Lnoise_spectrum_intp = interpolate.
 ↪interp1d(event['L_freq'],event['L_noise'],kind='linear')
    Lspectr1 = np.fft.fft(event['L_strain'] * signal.get_window(window =␣
 ↪'tukey', Nx = len(event['L_strain'])))
    Lspectr2 = np.fft.fft(event['L_template'] * signal.get_window(window =␣
 ↪'tukey', Nx = len(event['L_template'])))

    Lfreq = np.fft.fftfreq(len(event['L_template']), event['L_dt'])

    Lspectr1 = Lspectr1/np.sqrt(Lnoise_spectrum_intp(np.abs(Lfreq)))
    Lspectr2 = Lspectr2/np.sqrt(Lnoise_spectrum_intp(np.abs(Lfreq)))
    Lwhite1 = np.fft.ifft(Lspectr1)
    Lwhite2 = np.fft.ifft(Lspectr2)

    return Hwhite1, Hwhite2 ,Lwhite1, Lwhite2, Hfreq, Lfreq

def match_filter(event):
    Hfs = 1/event['H_dt']
    Lfs = 1/event['L_dt']

    Hsignal_white, H_white_spect, Lsignal_white, L_white_spect, Hfreq, Lfreq =␣
 ↪white(event)

    Hsignal_white = np.fft.fft(Hsignal_white) / Hfs
    H_white_spect = np.fft.fft(H_white_spect) / Hfs
    Lsignal_white = np.fft.fft(Lsignal_white) / Lfs
    L_white_spect = np.fft.fft(L_white_spect) / Lfs

    Hmf = Hsignal_white * H_white_spect.conjugate()
    Hmf = np.fft.ifft(Hmf)
```

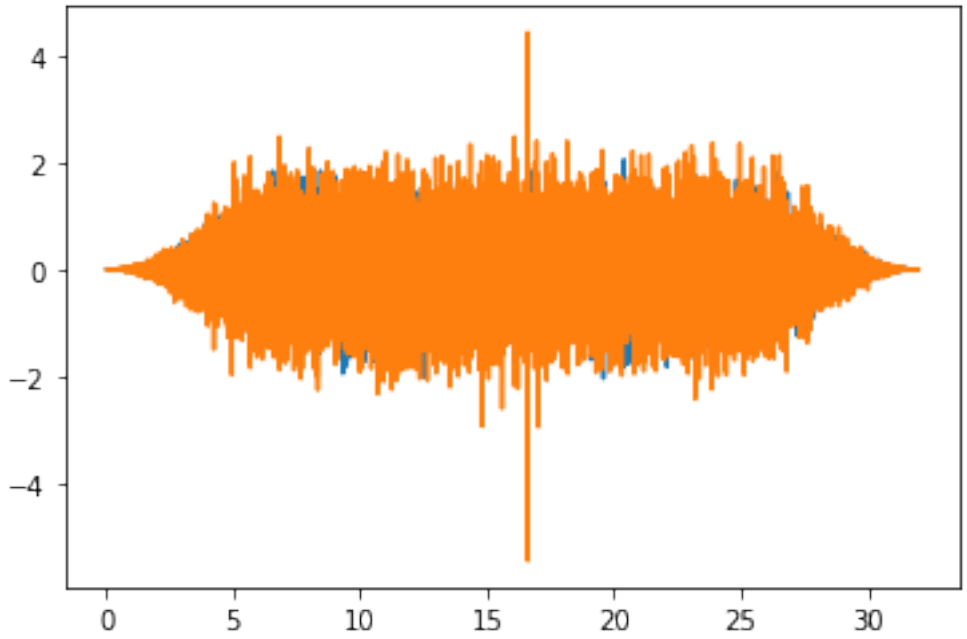
```
    Lmf = Lsignal_white * L_white_spect.conjugate()
    Lmf = np.fft.ifft(Lmf)

    Ht = np.arange(len(event['H_strain']) * event['H_dt'], step = event['H_dt'])
    Lt = np.arange(len(event['L_strain']) * event['L_dt'], step = event['L_dt'])



    return Ht, Hmf, Lt, Lmf
```

[25]:
```
for i in range(len(list_events)):
    Ht, Hmf, Lt, Lmf = match_filter(data[i])
    Htshift = np.fft.fftshift(Ht)
    Ltshift = np.fft.fftshift(Lt)

    plt.plot(Htshift, Hmf)
    plt.plot(Ltshift, Lmf)
    plt.show()
```




2
1
0
−1
0
5
10
15
20
25
30


2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
5
10
15
20
25
30



0.3 c.)

[26]:
```
for i in range(len(list_events)):
    Ht, Hmf, Lt, Lmf = match_filter(data[i])
    Hnoise = np.std(Hmf[5000:35000])
    HSNR = np.max(np.abs(Hmf))/Hnoise
    Lnoise = np.std(Lmf[5000:35000])
    LSNR = np.max(np.abs(Lmf))/Lnoise
    print('Estimated Hanford SNR = ', HSNR, 'and Estimated Livingston SNR =',␣
 ↪LSNR)
```

```
Estimated Hanford SNR =  17.31784555379502 and Estimated Livingston SNR =
12.086529276907937
Estimated Hanford SNR =  6.479458532989562 and Estimated Livingston SNR =
6.106278698553358
Estimated Hanford SNR =  8.475530386869139 and Estimated Livingston SNR =
4.923803099806829
Estimated Hanford SNR =  8.329249273008111 and Estimated Livingston SNR =
7.992943327731226
```

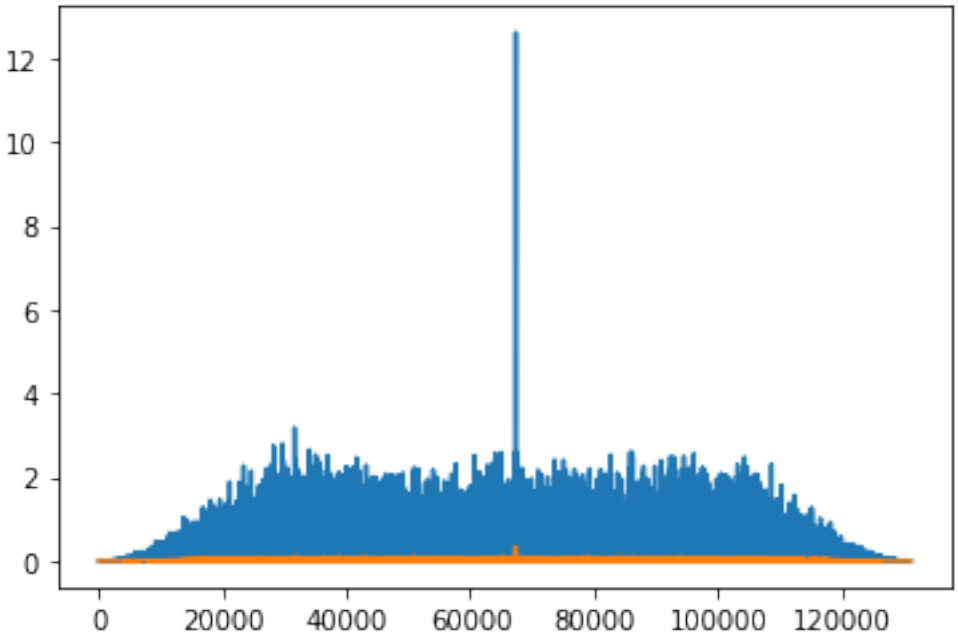
0.4 d.)

[27]:
```
def SNR(event):
    Hfs = 1/event['H_dt']
    Lfs = 1/event['L_dt']
```

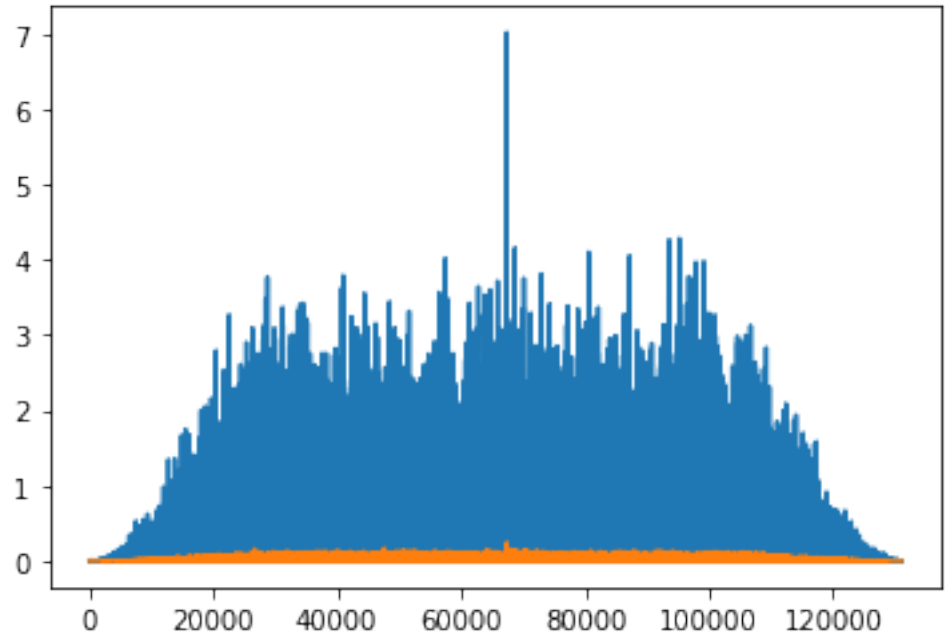
```
    Hsignal_white, H_white_spect, Lsignal_white, L_white_spect, Hfreq, Lfreq =␣
↪white(event)
    Hsignal_white = np.fft.fft(Hsignal_white) / Hfs
    H_white_spect = np.fft.fft(H_white_spect) / Hfs
    Lsignal_white = np.fft.fft(Lsignal_white) / Lfs
    L_white_spect = np.fft.fft(L_white_spect) / Lfs

    Hdf = np.abs(Hfreq[1] - Hfreq[0])
    Ldf = np.abs(Lfreq[1] - Lfreq[0])

    Hnoise = np.interp(np.abs(Hfreq), Hfreq, Hsignal_white)
    Lnoise = np.interp(np.abs(Lfreq), Lfreq, Lsignal_white)

    HSNR = Hsignal_white * H_white_spect.conjugate() / Hnoise
    LSNR = Lsignal_white * L_white_spect.conjugate() / Lnoise
    HSNR = 2*np.fft.ifft(HSNR)*Hfs
    LSNR = 2*np.fft.ifft(LSNR)*Lfs

    Hsigma = np.sqrt(np.abs(1*(H_white_spect * H_white_spect.conjugate() /␣
↪Hnoise).sum() * Hdf))
    Lsigma = np.sqrt(np.abs(1*(L_white_spect * L_white_spect.conjugate() /␣
↪Lnoise).sum() * Ldf))

    HnormSNR = HSNR/Hsigma
    LnormSNR = LSNR/Lsigma

    Hmax = round(len(event['H_strain']) / 2)
    Lmax = round(len(event['L_strain']) / 2)
    HnormSNR = np.abs(np.roll(HnormSNR,Hmax))
    LnormSNR = np.abs(np.roll(LnormSNR,Lmax))

    return HnormSNR, LnormSNR
```

[28]:
```
for i in range(len(list_events)):
    HSNR, LSNR = SNR(data[i])
    HSNRmax = HSNR[np.argmax(HSNR)]
    LSNRmax = LSNR[np.argmax(LSNR)]
    print('Hanford SNR = ', HSNRmax, 'and Livingston SNR =', LSNRmax)

    plt.plot(HSNR)
    plt.plot(LSNR)
    plt.show()
```

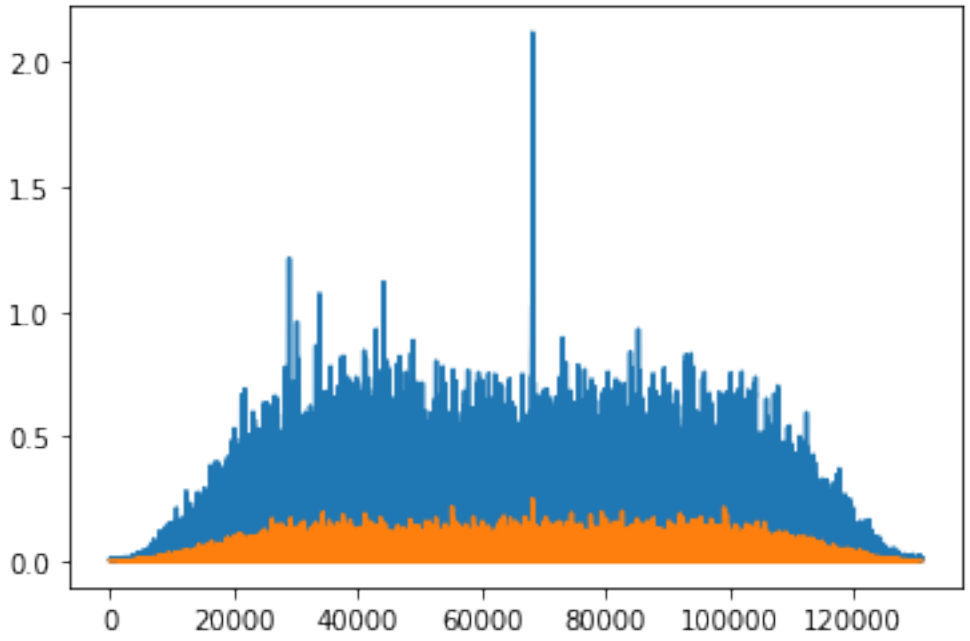
```
Hanford SNR =  12.619802711371046 and Livingston SNR = 0.33540778655434894
```



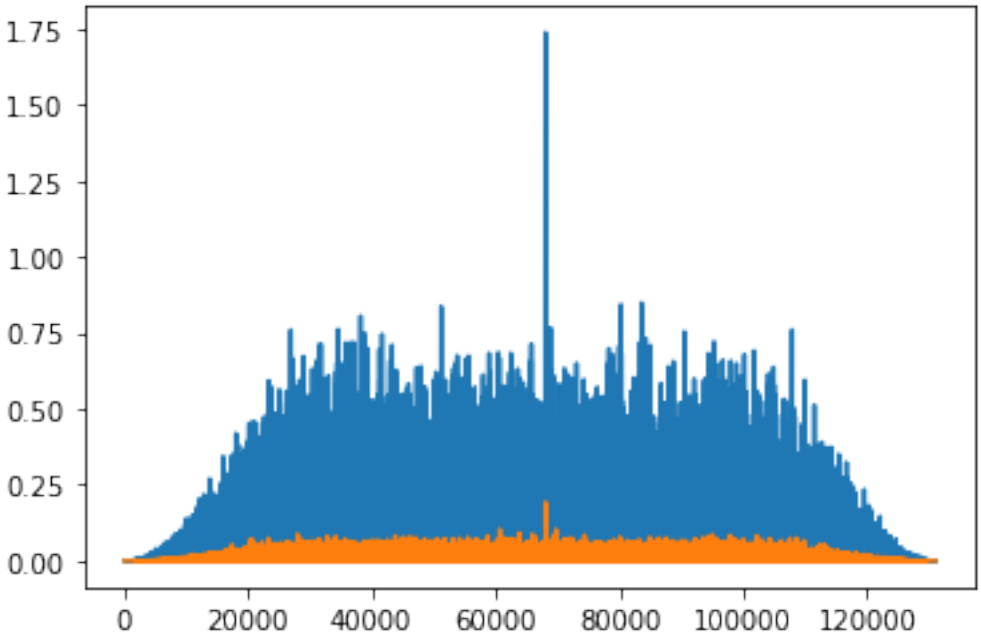

```
Hanford SNR =  7.016514806565015 and Livingston SNR = 0.24752213183869576
```




```
Hanford SNR =  2.119748411824677 and Livingston SNR = 0.24961066154886152
```




```
Hanford SNR =  1.7381431581763587 and Livingston SNR = 0.1934751864389854
```



0.5 e.)

[ ]: