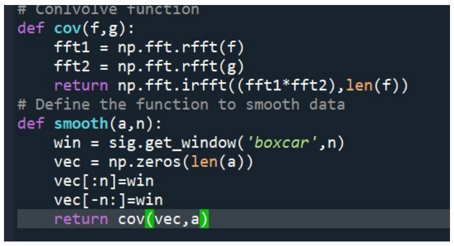
Here is the report of PHYS512 problemset 4 written by Jiaxing MA.

problem1

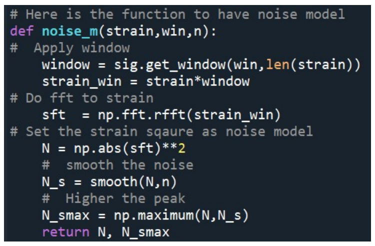
a

To have the noise model, first I set the square of the Fourier transform of strain as noise model, and then I apply window to the strain. I used hann window to the strain, because it have a extend flat period near the middle. Then I smooth the noise model by do convolution between the strain and the boxcar window, and I also try different flat length to find which is the best smooth parameter. At last, I make the peak higher by choose the maximum between the noise before and after smooth. Here is the code of convolution, smooth data and the noise model:



```python
# Conlvolve function
def cov(f,g):
    fft1 = np.fft.rfft(f)
    fft2 = np.fft.rfft(g)
    return np.fft.irfft((fft1*fft2),len(f))
# Define the function to smooth data
def smooth(a,n):
    win = sig.get_window('boxcar',n)
    vec = np.zeros(len(a))
    vec[:n]=win
    vec[-n:]=win
    return cov(vec,a)
```

(a)



```python
# Here is the function to have noise model
def noise_m(strain,win,n):
#  Apply window
    window = sig.get_window(win,len(strain))
    strain_win = strain*window
# Do fft to strain
    sft  = np.fft.rfft(strain_win)
# Set the strain sqaure as noise model
    N = np.abs(sft)**2
    #  smooth the noise
    N_s = smooth(N,n)
    #  Higher the peak
    N_smax = np.maximum(N,N_s)
    return N, N_smax
```

(b)

Figure 1: sample

Here is the noise model I got with different flat length of smooth. As we can see with n=6, the strain is not too flatten while the sharp peak is removed.

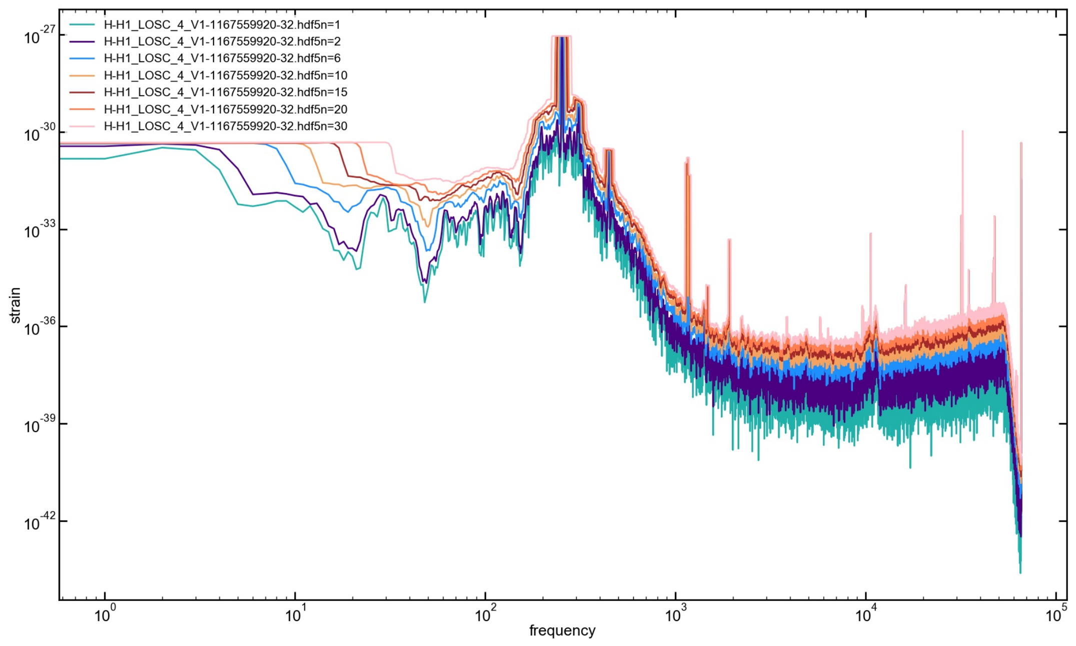


Figure 2: Result of smooth data with different n

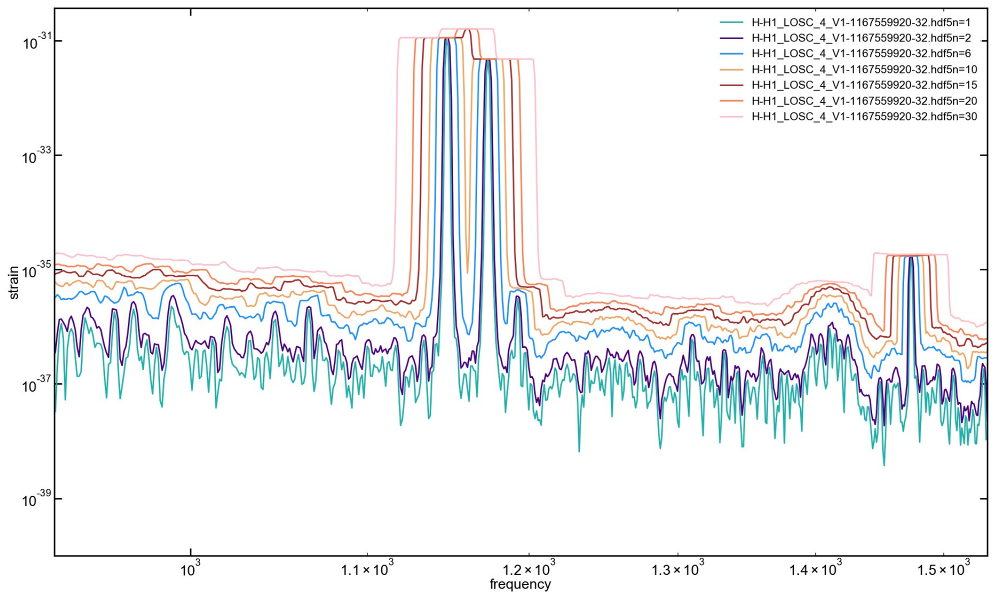


Figure 3: Result of smooth data with different n zooomed in

Here is the result I got by apply noise model to the strain of Livingston and Hanford separately.

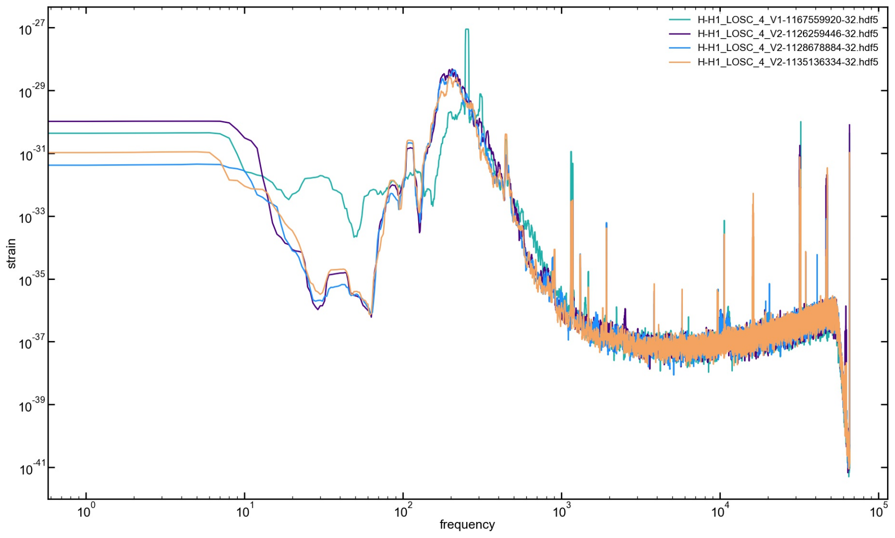


Figure 4: Result of Hanford

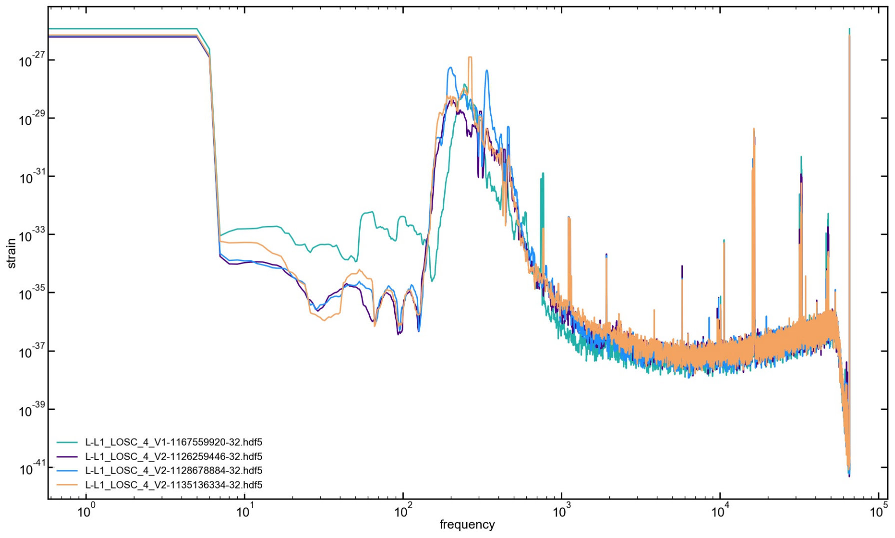
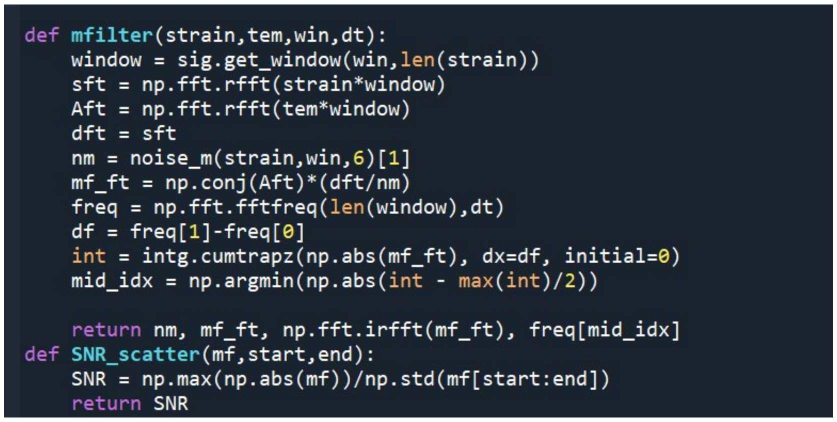


Figure 5: Result of Livingston

b

For the problem b, I apply the match filter to the correspond strain stored in the file BBH events. Here is the code I applied the match filter.



```python
def mfilter(strain,tem,win,dt):
    window = sig.get_window(win,len(strain))
    sft = np.fft.rfft(strain*window)
    Aft = np.fft.rfft(tem*window)
    dft = sft
    nm = noise_m(strain,win,6)[1]
    mf_ft = np.conj(Aft)*(dft/nm)
    freq = np.fft.fftfreq(len(window),dt)
    df = freq[1]-freq[0]
    int = intg.cumtrapz(np.abs(mf_ft), dx=df, initial=0)
    mid_idx = np.argmin(np.abs(int - max(int)/2))

    return nm, mf_ft, np.fft.irfft(mf_ft), freq[mid_idx]
def SNR_scatter(mf,start,end):
    SNR = np.max(np.abs(mf))/np.std(mf[start:end])
    return SNR
```

Figure 6: match filter code

The results will be shown in the end of the document with the results of other problem together.

c

I calculate the signal to noise ration by calculate the maximum of the match filter in time space and the noise around the peak. The results will be shown in the end of the document with the results of other problem together.

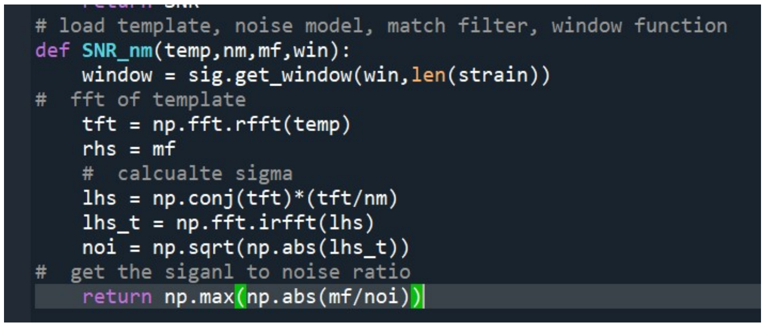
d

I have already calculated the scatter SNR in problem b. To calculate the analytical SNR from the noise model, I calculate the noise in frequency space and then compare the match filter signal to the noise in time space. Here is how I calculate the noise. As we can see from the result, the SNR of scatter and the analytical are different, I think the difference comes from the windowing of strain and the template. The results will be shown in the end of the document with the results of other problem together.

$$\sigma(f) = \sqrt{template(f)(template^{*}(f)/noisemodel)}$$

$$SNR = maximum(\frac{matchfilter(t)}{\sigma(t)})$$

Here is the code how I calculate the SNR in noise model.



```python
# load template, noise model, match filter, window function
def SNR_nm(temp,nm,mf,win):
    window = sig.get_window(win,len(strain))
#   fft of template
    tft = np.fft.rfft(temp)
    rhs = mf
    #  calcualte sigma
    lhs = np.conj(tft)*(tft/nm)
    lhs_t = np.fft.irfft(lhs)
    noi = np.sqrt(np.abs(lhs_t))
#   get the siganl to noise ratio
    return np.max(np.abs(mf/noi))
```

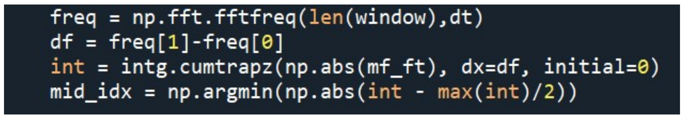
Figure 7: SNR code

e

The match filter in frequency space is the weight of the signal at the frequency, so I use the integration to calculate the mid frequency.

$$\int_0^f matchfilter(f')df' = 0.5 \int_0^\infty matchfilter(f')df'$$

Here is the code I used. The results will be shown in the end of the document with the results of other problem together.

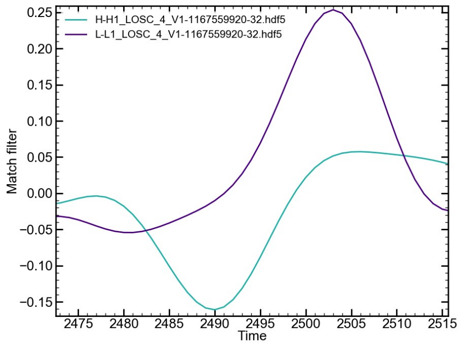


```
freq = np.fft.fftfreq(len(window),dt)
df = freq[1]-freq[0]
int = intg.cumtrapz(np.abs(mf_ft), dx=df, initial=0)
mid_idx = np.argmin(np.abs(int - max(int)/2))
```

Figure 8: Mid frequency code

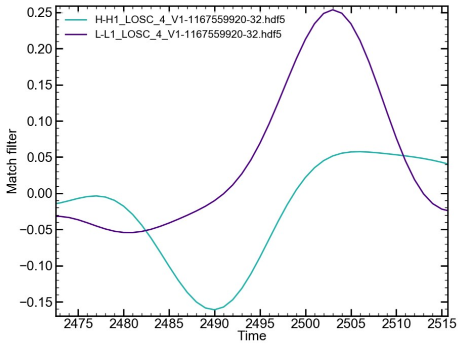
f

To calculate the precision of localize the arrival of gravitation wave, I calculate the σ by find the FWHM of the peak in the match filter time line, and use equation $\sigma = \frac{FWHM}{2.355}$ to calculate the σ Here is the peak. As we can see the FWHM is around 10*dt. We can extract dt from the data, so the FWHM is around 0.0024s. Times the speed of light, we can have the precision is around 720 km, so the $\sigma = 306km$. The position is calculated by assume the two detector is 2000 km apart, so it will be $\frac{306}{2000} = 15.3\%$



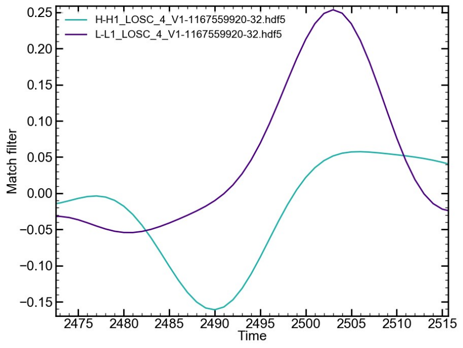
H-H1_LOSC_4_V1-1167559920-32.hdf5
L-L1_LOSC_4_V1-1167559920-32.hdf5
Match filter
Time

(a)



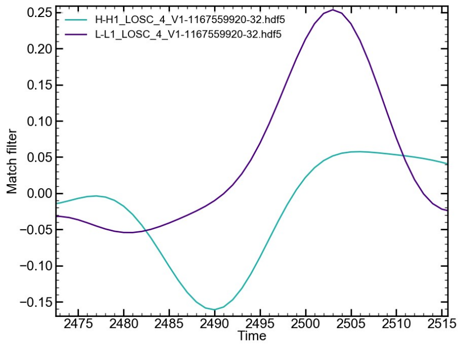
H-H1_LOSC_4_V1-1167559920-32.hdf5
L-L1_LOSC_4_V1-1167559920-32.hdf5
Match filter
Time

(b)



H-H1_LOSC_4_V1-1167559920-32.hdf5
L-L1_LOSC_4_V1-1167559920-32.hdf5
Match filter
Time

(c)



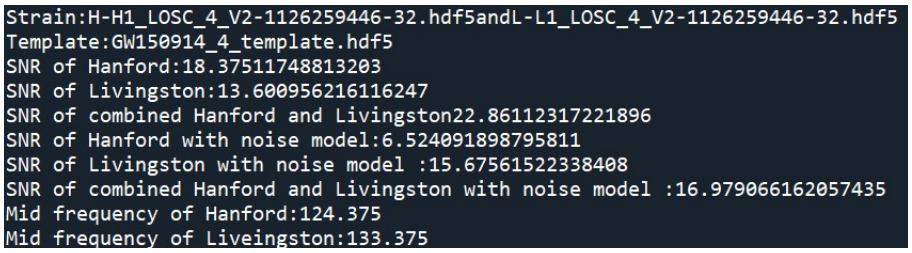
H-H1_LOSC_4_V1-1167559920-32.hdf5
L-L1_LOSC_4_V1-1167559920-32.hdf5
Match filter
Time

(d)

Results

Here are the results labeled by different templates

GW150914_4_template.hdf5



```
Strain:H-H1_LOSC_4_V2-1126259446-32.hdf5andL-L1_LOSC_4_V2-1126259446-32.hdf5
Template:GW150914_4_template.hdf5
SNR of Hanford:18.37511748813203
SNR of Livingston:13.600956216116247
SNR of combined Hanford and Livingston22.86112317221896
SNR of Hanford with noise model:6.524091898795811
SNR of Livingston with noise model :15.67561522338408
SNR of combined Hanford and Livingston with noise model :16.979066162057435
Mid frequency of Hanford:124.375
Mid frequency of Liveingston:133.375
```

Figure 9

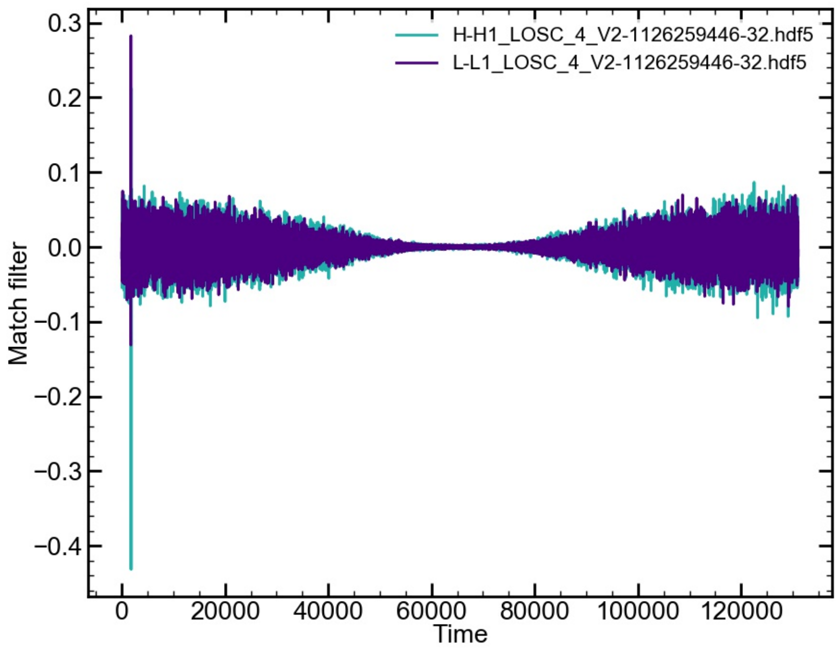
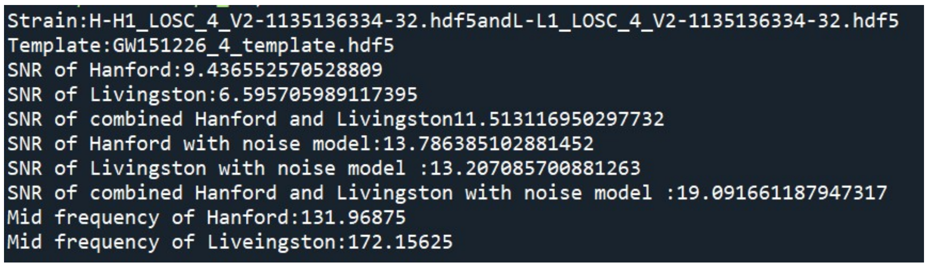


Figure 10

GW151226_4_template.hdf5



```
Strain:H-H1_LOSC_4_V2-1135136334-32.hdf5andL-L1_LOSC_4_V2-1135136334-32.hdf5
Template:GW151226_4_template.hdf5
SNR of Hanford:9.436552570528809
SNR of Livingston:6.595705989117395
SNR of combined Hanford and Livingston11.513116950297732
SNR of Hanford with noise model:13.786385102881452
SNR of Livingston with noise model :13.207085700881263
SNR of combined Hanford and Livingston with noise model :19.091661187947317
Mid frequency of Hanford:131.96875
Mid frequency of Liveingston:172.15625
```

Figure 11

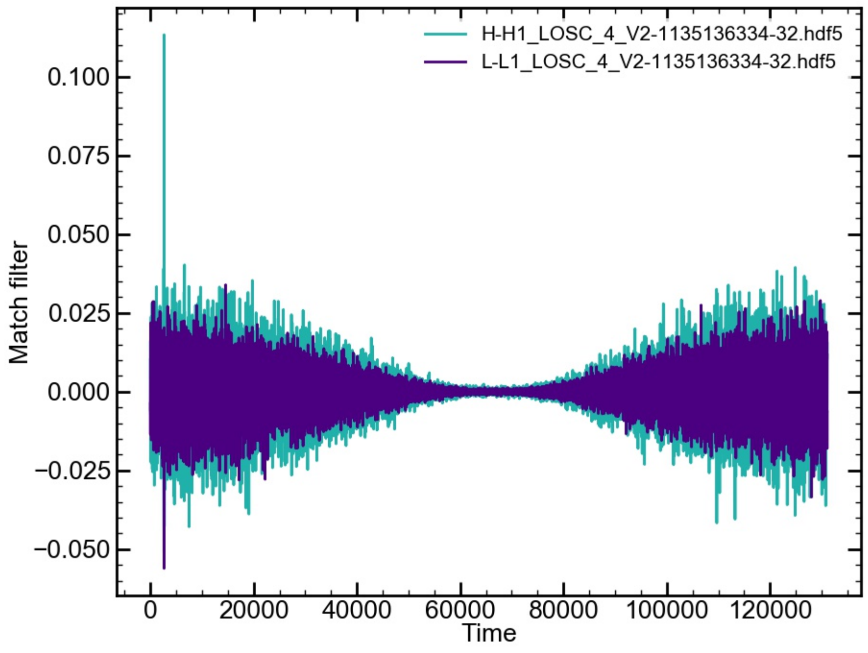
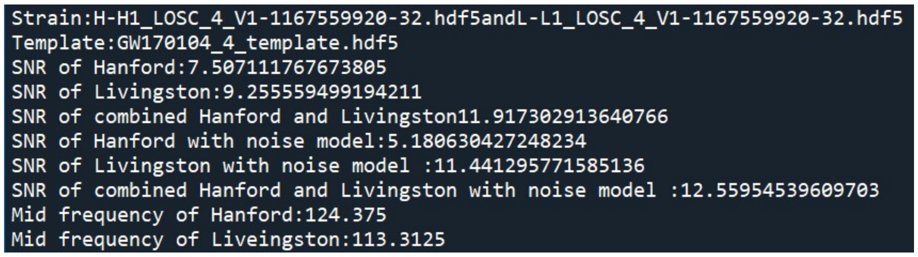


Figure 12

GW170104__4__template.hdf5



```
Strain:H-H1_LOSC_4_V1-1167559920-32.hdf5andL-L1_LOSC_4_V1-1167559920-32.hdf5
Template:GW170104_4_template.hdf5
SNR of Hanford:7.507111767673805
SNR of Livingston:9.255559499194211
SNR of combined Hanford and Livingston11.917302913640766
SNR of Hanford with noise model:5.180630427248234
SNR of Livingston with noise model :11.441295771585136
SNR of combined Hanford and Livingston with noise model :12.55954539609703
Mid frequency of Hanford:124.375
Mid frequency of Liveingston:113.3125
```

Figure 13

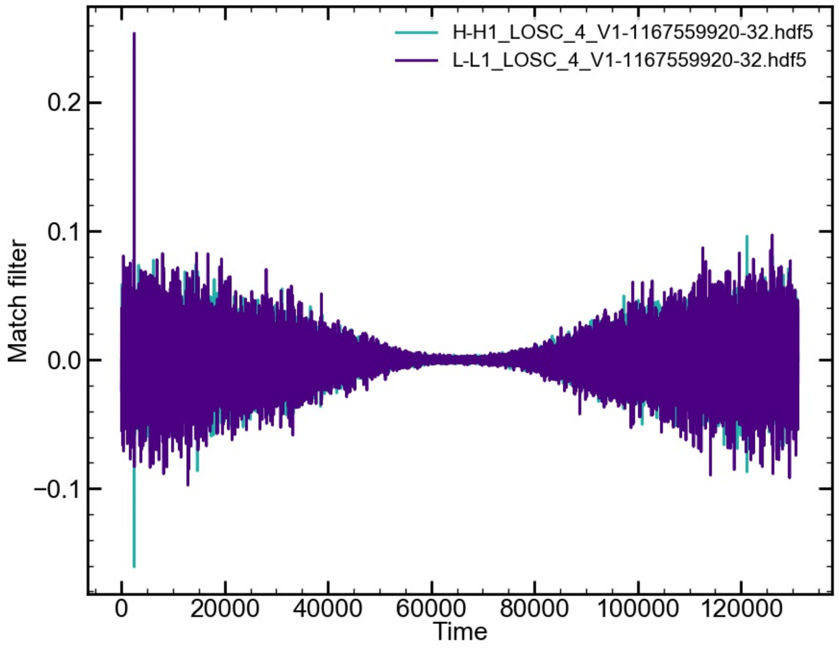
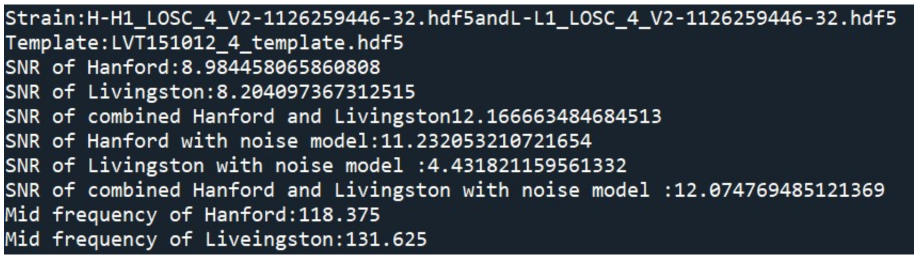


Figure 14

LVT151012__4__template.hdf5



```
Strain:H-H1_LOSC_4_V2-1126259446-32.hdf5andL-L1_LOSC_4_V2-1126259446-32.hdf5
Template:LVT151012_4_template.hdf5
SNR of Hanford:8.984458065860808
SNR of Livingston:8.204097367312515
SNR of combined Hanford and Livingston12.166663484684513
SNR of Hanford with noise model:11.232053210721654
SNR of Livingston with noise model :4.431821159561332
SNR of combined Hanford and Livingston with noise model :12.074769485121369
Mid frequency of Hanford:118.375
Mid frequency of Liveingston:131.625
```

Figure 15

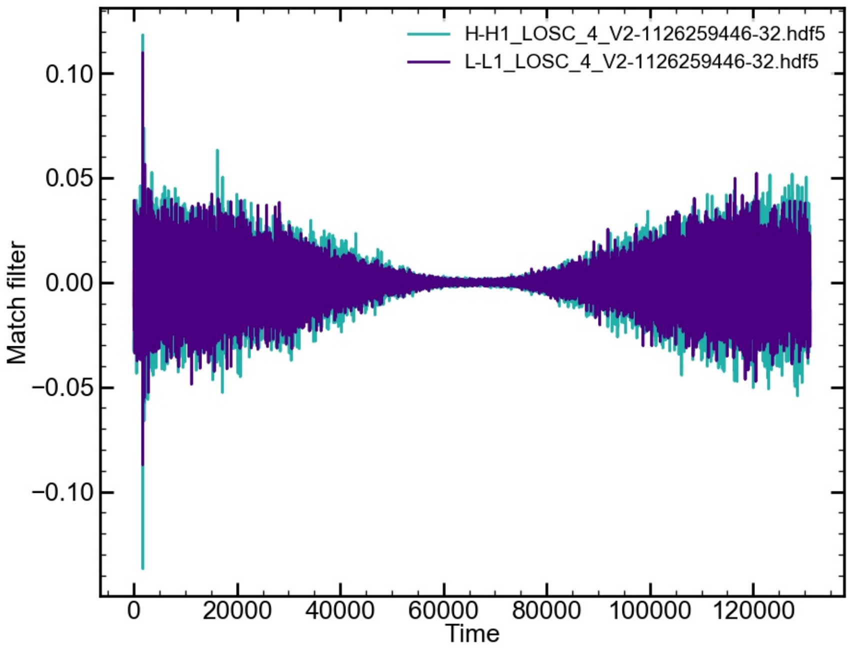

H-H1_LOSC_4_V2-1126259446-32.hdf5
L-L1_LOSC_4_V2-1126259446-32.hdf5
Match filter
Time
0.10
0.05
0.00
−0.05
−0.10
0
20000
40000
60000
80000
100000
120000

Figure 16