Hints:

For problem 1(part 1 and 2):
You can directly load the data from the link(I hope this now works).
Or, you can download the data from LIGO server(event name - GW170817)
Run the following code for this.

```
!wget -nc
https://dcc.ligo.org/public/0146/P1700349/001/H-H1_LOSC_CLN_4_V1-118700704
0-2048.gwf
!wget -nc
https://dcc.ligo.org/public/0146/P1700349/001/L-L1_LOSC_CLN_4_V1-118700704
0-2048.gwf
```

Then run the following command to load the strain data.

```
%matplotlib inline
import pylab
from pycbc.filter import highpass
from pycbc.catalog import Merger
from pycbc.frame import read_frame

merger = Merger("GW170817")
strain, stilde = {}, {}
for ifo in ['H1', 'L1']:
   # We'll download the data and select 256 seconds that includes the
event time
   ts =
read_frame("{}-{}_LOSC_CLN_4_V1-1187007040-2048.gwf".format(ifo[0], ifo),
                   '{}:LOSC-STRAIN'.format(ifo),
                  start_time=merger.time - 224,
                  end_time=merger.time + 32,
                  check_integrity=False)

   # Read the detector data and remove low frequency content
   strain[ifo] = highpass(ts, 15)

   # Remove time corrupted by the high pass filter
   strain[ifo] = strain[ifo].crop(4, 4)

   # Also create a frequency domain version of the data
```

```
    stilde[ifo] = strain[ifo].to_frequencyseries()


#print (strain.delta_t)
pylab.plot(strain['H1'].sample_times, strain['H1'])
pylab.xlabel('Time (s)')
pylab.show()
```

In this way you should be able to load the data files.

Then you should be able to analyse the data similar to the GW150914 event( shown in demonstration). You can use the ‘TaylorT2’ approximation in order to generate the template waveforms for matched filtering. (As you may have noticed, the ‘SEOBNRv4_opt’ approximation doesn’t work when you choose the low mass system).

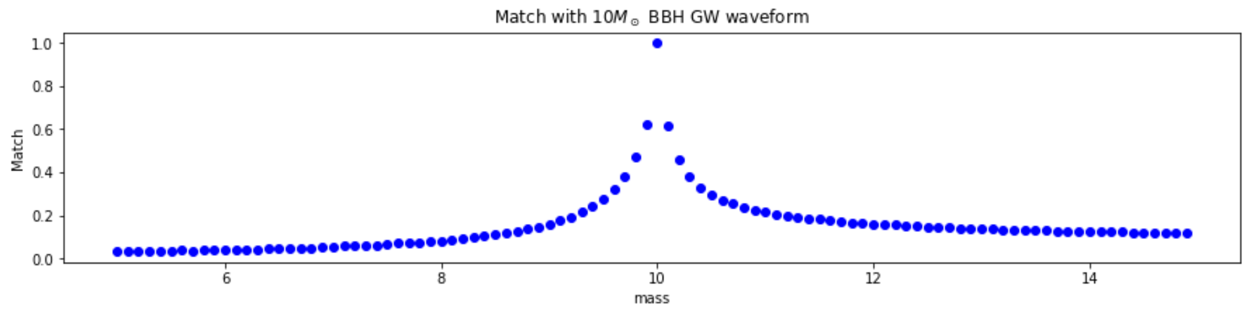
>>>For problem 1(part 3):

You first need to generate a waveform(h*) for m_1 = m_2 = 10
Then you generate another set of waveforms varying waveform(h_t) m1 and m2 between 5 t0 15.
Then calculate match between h* and h_t and plot the match verses mass.
Now in order to estimate the match you can use the following function.
[N.b. the last line of the code is estimating match between hp and sp. Here ‘m’ is the match value. ]

```
from pycbc.waveform import get_td_waveform
from pycbc.filter import match
from pycbc.psd import aLIGOZeroDetHighPower

f_low = 30
sample_rate = 4096

# Generate the two waveforms to compare
hp, hc = get_td_waveform(approximant="EOBNRv2",
                         mass1=10,
                         mass2=10,
                         f_lower=f_low,
                         delta_t=1.0/sample_rate)

sp, sc = get_td_waveform(approximant="TaylorT4",
                         mass1=10,
                         mass2=10,
                         f_lower=f_low,
                         delta_t=1.0/sample_rate)

# Resize the waveforms to the same length
tlen = max(len(sp), len(hp))
```

```
sp.resize(tlen)
hp.resize(tlen)

# Generate the aLIGO ZDHP PSD
delta_f = 1.0 / sp.duration
flen = tlen//2 + 1
psd = aLIGOZeroDetHighPower(flen, delta_f, f_low)

# Note: This takes a while the first time as an FFT plan is generated
# subsequent calls are much faster.
m, i = match(hp, sp, psd=psd, low_frequency_cutoff=f_low)
```

At the end you should find something similar to the following result.



You can see that the match is maximum(= 1), when you reach close to 10.

Problem 2 (part 1)

You have load the data using the link.
If you use numpy library, you can load the data following the code bellow

```
 >>> d = numpy.load('File_name')
 >>> time = d[:, 0]
>>> strain = d[:, 1]
```

By this way you can load the data from the downloaded fine. Then you can also convert this data into time series data.

```
>>>import pycbc.types
>>>dt = time[1] - time[0]
>>>data = pycbc.types.TimeSeries(d[:, 1], delta_t = dt)
```

After this you can use this data(same as before) and calculate the matched filter using template waveforms with mass ranges between 5 to 10. And see if your maximum SNR(signal to noise ratio) crosses 8. If not, this means you have only noise, no signal.
[Use the code from problem 1 for the calculation of matched filtering. ]

*****Part 2

You can whiten the data, which means you need to divide the by noise spectral density. As you will see that the data provided to you is just the noise, so you can use the data to estimate the psd ( this has been demonstrated in the event analysis of GW150914).

Once you find the power spectral density, you can easily whiten the data using the following code.

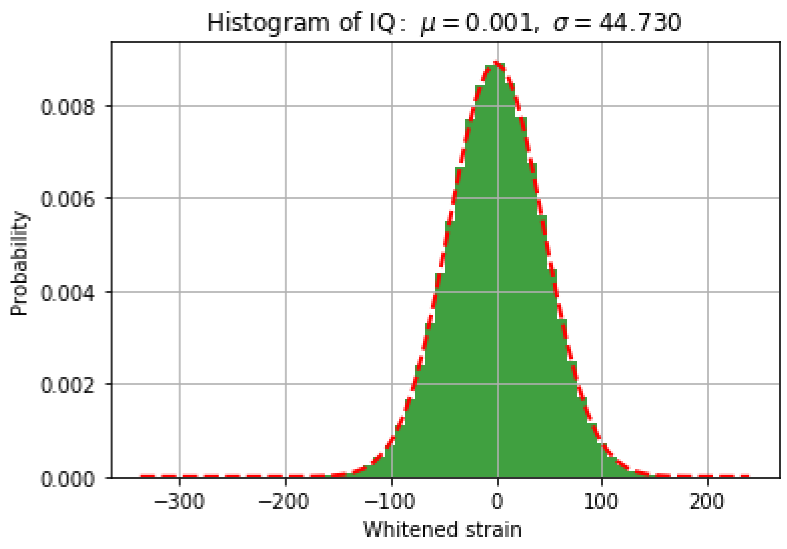
```
# We do it this way so that we can whiten both the template and the data
import pycbc.psd

white_data = (data.to_frequencyseries() / psds[ifo]**0.5).to_timeseries()
```

#psds - is the power spectral; density of your data)

Next, take this whitened data and make a histogram plot. The distribution you will see is Gaussian distribution, with  zero mean.

You should see something simpler to this.

(don’t forget to crop your data i.e. cut down the first 4 secs and last 4 secs, that is beasues artifacts appears at the beginning and the ending of your data).



You see that the noise follows Gaussian distribution.

****part 3

Now you have to test whether your noise is following the stationary assumption is not. The assumption is correct to a limit, not exactly true for real detectors. How to test for this has been explained in the assignment itself.

In order to calculate the optimal SNR you can use the following function-

```
 >>>>from pycbc.filter import sigmasq
 >>>>SNR = (pycbc.filter.sigmasq(hp, psd))**0.5
```

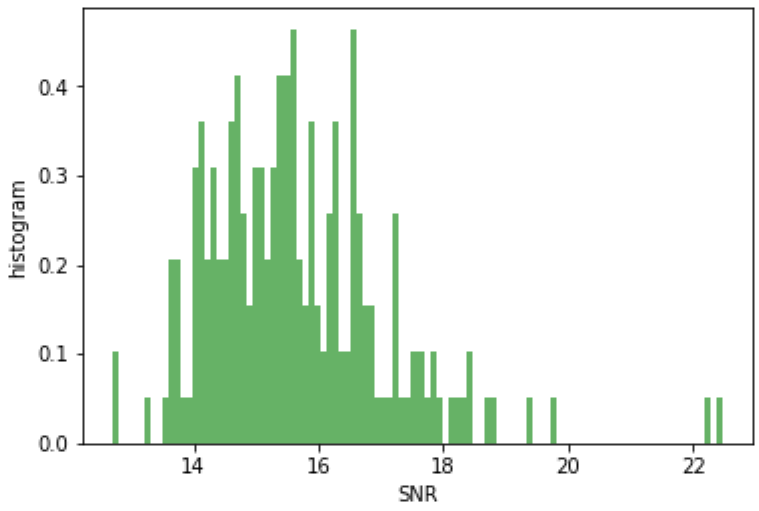
Or you can estimate this using the following estimator

```
 >>>>SNR = (np.sum((4*hp*np.conj(hp)*hp.delta_f/psd).data).real)**0.5
```

Here ‘hp’ is the waveform in frequency domain.
For example, you can generate a frequency domain waveform -

```
>>>>from pycbc.waveform import get_fd_waveform
>>>>hp, hc = get_fd_waveform(approximant="TaylorF2",
                        mass1=3,
                        mass2=3,
                        delta_f=1.0/(200.0) ,
                        distance = 500,
                        f_lower=20.0, f_final = 2048.0)    #Luminosity distance = 500 Mpc
```

You need to estimate the psd from each chunks for data ( dividing your data into equal time intervals). Hence, you will get a list of SNR. Now if you want to make a histogram out of the estimated SNR, you should find something similar to the following diagram -



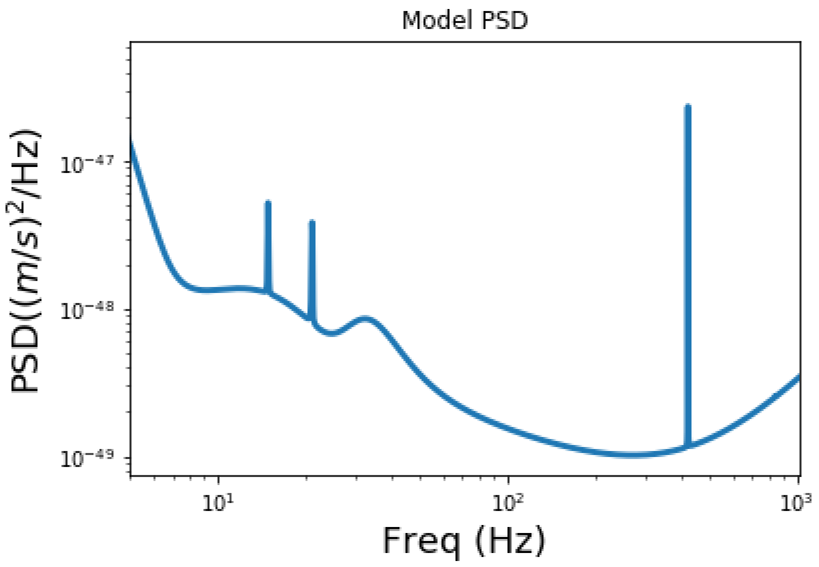
Now, you need to comment on your results.. What does this distribution tells you?

**Question** 3

**Part 1:** plot of horizon distance for advanced LIGO detector.
For this you need to consider first the modeled power spectral density( estimated after estimating noise budget of the detector). You can get the psd of the detector ‘aLIGO’ by calling the following function-

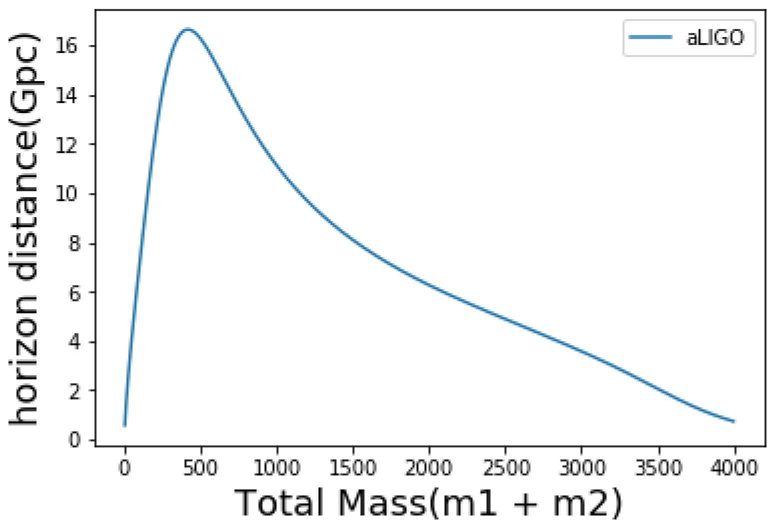
```
>>>>import pycbc.psd
>>>>import numpy as np
>>>>flow = 4.0   # set up the lower cut off frequency
>>>>delta_f = 1.0 / 16
>>>>flen = int(2048.0/ (delta_f)) + 1
>>>>psd = pycbc.psd.aLIGOZeroDetHighPower(flen, delta_f, flow)

>>>>psd.data[:int(flow/delta_f)] = np.inf  #set the value outside the frequency range to infinity
>>>>psd.data[-1] = np.inf
```

The modeled PSD for aLIGO detector should look like the following.



Now, you can estimate horizon distance for aLIGO by estimating the distance at which the value of SNR will be 8. You need to generate waveform and estimate the SNR as the previous problem. (you can choose an arbitrary distance (like 1000Mpc) while generating the waveform). Then estimate the SNR(ρ). As we know the distance(D) is inversely proportional to SNR(ρ), i.e. D $\propto$ 1/ρ. So, from this relation you can set up the distance for which the SNR is 8, and this distance will be horizon distance. You need to estimate the horizon distance for different masses( generating waveforms from different mass systems, assuming mass 1 = mass 2).

The horizon plot should like the following -



Part 2

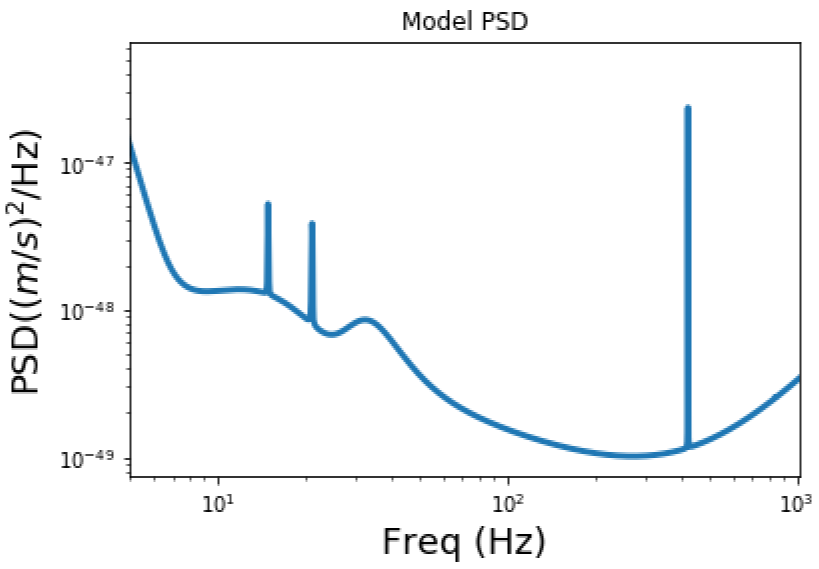
You need to explain why you see such behaviour of the curve.

## Part 3

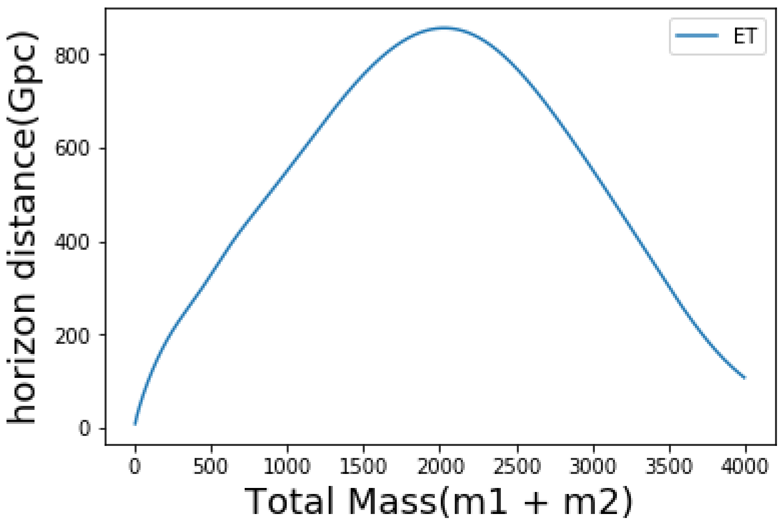
You need to repeat the same analysis at part 1. But this time you need to repeat for Einstein telescope(ET), which means you need to use ET's modeled PSD curve.

```
>>>>import pycbc.psd
>>>>import numpy as np
>>>>flow = 4.0   # set up the lower cut off frequency
>>>>delta_f = 1.0 / 16
>>>>flen = int(2048.0/ (delta_f)) + 1
>>>>psd = pycbc.psd.EinsteinTelescopeP1600143(flen, delta_f, flow)

>>>>psd.data[:int(flow/delta_f)] = np.inf  #set the value outside the frequency range to infinity
>>>>psd.data[-1] = np.inf
```

The PSD should look like this.



Now, you can estimate the horizon distance plot and the plot should look like the following -


horizon distance(Gpc)
800
600
400
200
0
0
500
1000
1500
2000
2500
3000
3500
4000
Total Mass(m1 + m2)
ET