

Name (please print legibly!) _____

Gravitational Waves: Assignment 5

Gravitational-wave Data Analysis

1. In class, we saw that it is not necessary for a signal to have an amplitude h_0 greater than the noise floor amplitude n_0 for us to be able to detect it. Instead, it is sufficient to have $h_0 > (\tau_0/T)^{1/2} n_0$ where τ_0 is the typical characteristic time of the signal and T is the observation time. Consider the following astrophysical sources. How many orders of magnitude weaker than the noise amplitude can signal amplitude be for us to begin to detect it?
 - (a) Millisecond pulsar with a period of 1 ms with data collected for 1 year
 - (b) Binary object with period of 10^{-2} s with data collected for 100 seconds
2. In this exercise, we will find an inspiral signal buried in detector noise. To get started, download the data and template files from the following locations:
https://www.nikhef.nl/~caudills/assignments/strain_data.dat
https://www.nikhef.nl/~caudills/assignments/template_data.dat
The data contains a signal hidden in the first 12 seconds of data. Both the data and the template have a sampling rate of 4096 Hz.
 - (a) Separately plot the data and the template. Can you see the inspiral signal in the data? Also note the structure of the beginning and end of the template. It has been tapered at the beginning and end so that it can be more easily Fourier transformed.
 - (b) As you should have seen above, the signal amplitude is much smaller than the noise amplitude. But the signal does have more power than the noise at some frequencies. Make an amplitude spectral density (ASD) plot of the template and the noise on the same figure. To get the ASD for the noise, use data from the last 4 seconds since the signal is present somewhere in the first 12 seconds and we don't want it to contaminate our measurement. Note in Python, you can use `matplotlib.pyplot.psd(data, NFFT=fs, Fs=fs)` where `fs` is the the sampling rate. This function will return a power spectral density (PSD) and the frequencies corresponding to each point in the PSD. To get the ASD, simply take the square root of the PSD. You will want to plot these on log-log scales.
 - (c) The noise has been high-pass filtered with a 20 Hz cut-off. What effect does this have on the noise ASD and what can we infer from the ASD below 20 Hz?
 - (d) The noise ASD starts to fall-off around 2048 Hz. What is special about this frequency and what can we infer from the ASD above 2048 Hz?
 - (e) **Method 1: Bandpass Filter.** This method is very simple. We know that the signal probably has frequency evolution between 80 to 250 Hz so we can look for a signal in this frequency range. We can make a bandpass filter using Python's `scipy.signal.butter(N=4, Wn=[80/(fs/2), 250/(fs/2)])` where `N=4` is the order of the filter and `Wn` is an array giving the frequencies of importance as a fraction of the Nyquist frequency. To apply the filter to the data, we can use `scipy.signal.lfilter(B, A, data)` where `B,A` are the coefficients of the bandpass filter that you made and `data` contains the hidden signal.

- i. Plot the band-passed data versus time. Do you see any evidence for the signal? At what time?
- (f) **Method 2: Time Domain Cross Correlation** This method looks for the similarity between two time series. We will take the cross-correlation of the template with the data. We can use Python's `numpy.correlate(data, template)` for this.
 - i. Plot the cross-correlation of the template with the data. Note that the time axis of the cross-correlation represents the off-set between the data start time and the filter start time. There should be a peak when the start of the template matches the data.
 - ii. Plot the cross-correlation of the template with the band-passed data. Now the peak will occur when the signal is the loudest, near the end of the template. This is closest to what we actually do in LIGO and Virgo, when we report the coalescence time when the binary crosses the inner-most stable circular orbit, near the end of the template.
- (g) **Method 3: Optimal Matched Filter** The cross correlation in the time domain does not allow us to weight frequency bins. We would like to be able to do this because the gravitational wave data has more noise in some frequency bins than others. Thus, we can perform a cross correlation in the frequency domain and then weight each frequency bin by the inverse of the noise power spectrum.
 - i. Take the Fourier Transform of the data and the template. For the data, we can use Python's `numpy.fft.fft(data)`. We can use this for the template too but we need the template and data to be the same length. Thus, we should zero pad the template before we take the FFT. I suggest to look at `numpy.zeros()` and `numpy.append()` for achieving this.
 - ii. We need a measurement of the PSD again, with no signal in it. Take the PSD of the last 4 seconds of data. You can achieve this again with `matplotlib.pyplot.psd()`. The numpy FFT function returns an array with a particular convention for the order of the frequency bins. You need to interpolate this PSD to estimate the values at each FFT frequency. The needed frequencies can be obtained with `numpy.fft.fftfreq(data.size)*fs`. Then you can use `numpy.interp()` to determine the interpolated PSD values.
 - iii. Multiply the Fourier space template conjugate (`template_fft.conjugate()`) with the Fourier space data and divide by the noise power in each frequency bin to obtain the matched filter output. Then we can take the Inverse Fourier Transform (IFFT) of the filter output to put it back in the time domain. This can be obtained with `numpy.fft.ifft()`.
 - iv. Finally, we can normalize the matched filter output so that we expect a value of 1 at times of just noise. Then the peak of the matched filter output will tell us the signal-to-noise ratio (SNR) of the signal. To do this, we need to compute the variance of the template using `2 * (template_fft * template_fft.conjugate()) / psd.sum() * df`. The SNR is then the IFFT of the filter output divided by the square root of the variance of the template.
 - v. Plot SNR versus time.