Problem Set #3

Observational Techniques of Modern Astrophysics — PHYS 641

Fall 2018

Bridget Andersen

November 2, 2018

The code script for this problem is located in "./pset3_main.py". We also have an additional script called "./pset3_plots.py" that completes analysis for a single event and outputs many useful plots that we present in this report.

In this problem set, we use what we've learned in class about Fourier transforms and matched filtering to complete a very simple version of the LIGO analysis.

1 Matched Filtering

In this section, we review the basic concepts of matched filtering. Matched filtering is the process of correlating a known template signal with an unknown dataset in order to discover instances of the templat signal within the dataset. This method naturally extends to LIGO data analysis, with the gravitational wave strain signature as the template (see Figure 1) and strain measurements from the detectors as the the unknown dataset.

The framework for matched filtering can be derived from fitting a 1-parameter template to the data at many positions within the data (shifting the template. The "fitting" we are talking about is accomplished my minimizing the χ^2 maximum likelihood function. This is given by:

$$\chi^2 = (d - Am)^T N^{-1} (d - Am) \tag{1}$$

where d is the input data, A is the model or template that we are searching for, and m is the one parameter being fit (for example, the amplitude). To minimize this, we simply take the derivative and set it equal to zero:

$$\begin{aligned}
\nabla\chi^2 = \frac{\partial\chi^2}{\partial m} &= \frac{\partial}{\partial m}\left[(d - Am)^T N^{-1} (d - Am)\right] \\
&= (-A)^T N^{-1} (d - Am) + (d - Am)^T N^{-1} (-A) \\
&= -2A^T N^{-1} (d - Am) = 0 \\
\implies A^T N^{-1} (d - Am) &= 0 \\
A^T N^{-1} d - A^T N^{-1} Am &= 0 \\
A^T N^{-1} Am &= A^T N^{-1} d
\end{aligned}$$

Then the best fit parameter is given by:

$$m = \left(A^T N^{-1} d\right)\left(A^T N^{-1} A\right)^{-1} \tag{2}$$

If A is a 1D template like it is in the LIGO analysis (n by 1), then this finally becomes:

$$m = \frac{A^T N^{-1} d}{A^T N^{-1} A} \tag{3}$$

Now if we want to fit at several positions within the dataset, we can imagine shifting the template over the data and completing this fit at each shift. If the noise in the data is stationary (does not systematically change in time), then the denominator in this expression does not change with shifted template, and it only needs to be calculated once. For the numerator, we can take $N^{-1/2}d$ and dot it with $A^T N^{-1/2}$ for each shifted template. Note that we split N^{-1} into $N^{-1/2}$'s for the purposes of pre-whitening (see section 3). We are essentially calculating:

$$\texttt{numerator} = \sum_t \left[N^{-1/2} d\right](t) \left[A^T N^{-1/2}\right](t - \tau) \tag{4}$$

which is just the correlation of the data with the template for a given delay τ. By the convolution theorem, the convolution between two functions is given by:

$$f * g = \mathcal{F}^{-1}\left(\mathcal{F}(f) \cdot \mathcal{F}(g)\right) \tag{5}$$

where $\mathcal{F}$ is the Fourier transform. Therefore, our equation for the matched filter becomes:

$$\boxed{m = \frac{\mathcal{F}^{-1}\left(\mathcal{F}\left(A^T N^{-1/2}\right) \cdot \mathcal{F}\left(N^{-1/2} d\right)\right)}{\left(N^{-1/2} A\right)^T \left(N^{-1/2} A\right)}} \tag{6}$$

This gives the best-fit amplitude of the input template across the input dataset. When the dataset has a structure in it that looks like the template, this amplitude value will increase.

Often we want to know if our signal detection is statistically significant by plotting the signal to noise of the detection. To do this, we need to determine the noise on the fit amplitude. We can determine this by subtracting the best fit model from the data and examining the variance of what is left, like so:

$$\langle m m^T \rangle = \langle \left(A^T N^{-1} A\right)^{-1} A^T N^{-1} d \cdot d^T N^{-1} A \left(A^T N^{-1} A\right)^{-1} \rangle$$

where we have used the fact that both $N^T = N$ and so $\left(A^T N^{-1} A\right)^T = A^T N^{-1} A$. Now, since we have subtracted the best fit model off the data, we have that $d^2 = \sigma^2$ and so $d^T d = N$. Then we have:

$$\begin{aligned}
\langle m m^T \rangle &= \langle \left(A^T N^{-1} A\right)^{-1} A^T N^{-1} \cdot \cancel{N} \cdot \cancel{N^{-1}} A \left(A^T N^{-1} A\right)^{-1} \rangle \\
&= \langle \cancel{\left(A^T N^{-1} A\right)^{-1}} \cdot \cancel{A^T N^{-1} A} \cdot \left(A^T N^{-1} A\right)^{-1} \rangle = \langle \left(A^T N^{-1} A\right)^{-1} \rangle \\
\implies \sigma_m &= \sqrt{\langle m m^T \rangle} = \sqrt{\left(A^T N^{-1} A\right)^{-1}} = \left(\left(N^{-1/2} A\right)^T \left(N^{-1/2} A\right)\right)^{-1/2}
\end{aligned}$$

Thus, the signal to noise of the matched filter detection is given by:

$$\text{SNR} = \frac{\mathcal{F}^{-1}\left(\mathcal{F}\left(A^T N^{-1/2}\right) \cdot \mathcal{F}\left(N^{-1/2} d\right)\right)}{\left(N^{-1/2} A\right)^T \left(N^{-1/2} A\right)} \cdot \frac{1}{\left(\left(N^{-1/2} A\right)^T \left(N^{-1/2} A\right)\right)^{-1/2}}$$

$$\implies \boxed{\text{SNR} = \frac{\mathcal{F}^{-1}\left(\mathcal{F}\left(A^T N^{-1/2}\right) \cdot \mathcal{F}\left(N^{-1/2} d\right)\right)}{\sqrt{\left(N^{-1/2} A\right)^T \left(N^{-1/2} A\right)}}}$$

We use this formula in our code to derive the signal to noise of our LIGO detections (see Figure 7).

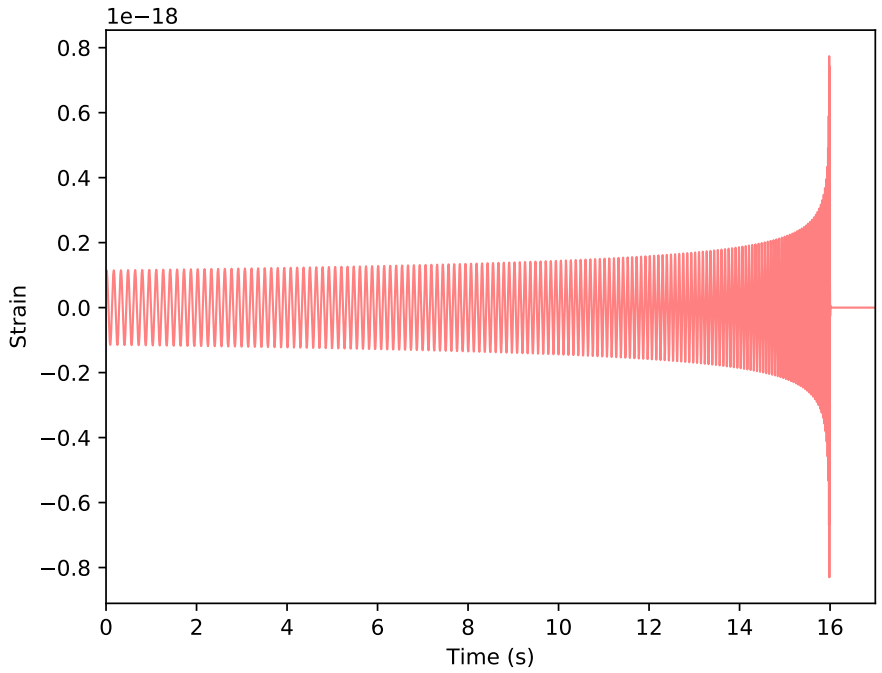

Figure 1: A plot of the given template for the GW150914 event.

Windowing The Data and Noise

We can compute all of the Fourier transforms above using the discrete Fourier transform given by:

$$F(k) = \sum_x f(x)\, e^{-2\pi i k x/N} \tag{7}$$

where N is the number of data points in your input dataset. Essentially, the DFT uses a sum of the $\exp(-2\pi i k x/N)$ basis functions to reconstruct your signal. Since the basis functions are periodic and defined over infinity, the DFT will also assume that your input dataset $f(x)$ is periodic over infinity, and that the finite dataset that you input represents one period. Essentially, the DFT will act like the input signal is your finite signal repeated over and over again. Thus, if the two endpoints of your input signal are not the same, then the DFT will see a sharp discontinuity at every period transition. These discontinuities appear in the DFT as frequency components that are not present in the original finite signal, also called ringing artifacts.

Since our input LIGO strain data does not necessarily have the same values at the start and the end, we need to take these ringing artifacts into account in our analysis. One wasy to avoid ringing artifacts is through a method called windowing. Windowing consists of multiplying the input data by a a finite length window with an amplitude that varies smoothly and gradually toward zero at the edges. This makes the endpoints of the waveform meet and, therefore, results in a continuous waveform without sharp transitions.

In our analysis, we window our initial strain data using a Tukey function (note that we got the code to produce this function from this link). In Figure 2, we show the data before and after windowing, as well as the general shape of the Tukey function. We chose the Tukey function because it has a nice, flat plateau in the middle which preserves as much of the original dat amplitudes as possible, while still gradually tapering at the edges. We complete this windowing of our strain data before using it in further analysis.

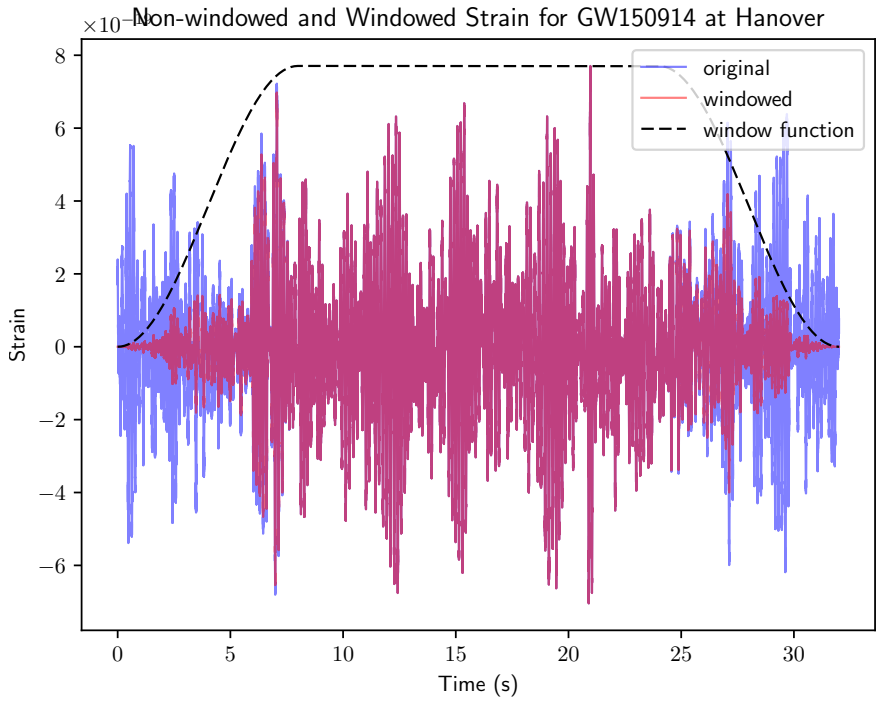

Figure 2: A plot of the windowed and non-windowed strain for the GW150914 event at the Hanover detector.

2 Estimating The Noise

To complete the matched filtering described in section 1, we need to obtain an estimate of our noise model (N in the equations above). Previously, in problem 3 of Problem Set #4, we were able to recover an original given noise matrix by averaging over many noise realizations. In our situation, we do not have the original noise matrix, but we can use the same sort of process to derive estimates of the noise matrix for both the Hanford and Livingston detectors by averaging over the noise realizations for each event.

Assuming our data is stationary, then if we take the Fourier transform of our data $\mathcal{F}_k$:

$$\langle \mathcal{F}_k, \mathcal{F}_{k'} \rangle = \sigma_k^2 \delta_{k,k'} \tag{8}$$

Essentially, the noise becomes uncorrelated in Fourier space, so that the actual noise matrix becomes diagonal. Since the noise matrix is diagonal in Fourier space, we don't have to create an entire ginormous matrix in our analysis, we can just keep track of the diagonal of the matrix in a 1D array. Thus, we can get a very rough estimate of the noise matrix by just taking the power spectrum of the strain data for a given event and detector (see Figure 3 for examples of these power spectra for the GW150914 event at the Hanover and Livingston detectors). Since the actual LIGO signal for each event is very small, it won't significantly bias this noise estimate. We can then average these power spectra over each event to obtain noise models for both the Hanford and Livingston detectors.

However, if you look at the noise power spectra in Figure 3, we can see that they are noisey! So we have noisey noise! To reduce the amount of noise in our noise models, we smooth the models by convolving them with a smoothing filter kernel. We tried smoothing using both a boxcar kernel and a Gaussian kernel. Each kernel has slightly different effects, especially around peaks in the noise. Figure 4 shows the noise smoothed by both a Gaussian and a boxcar function, zoomed into

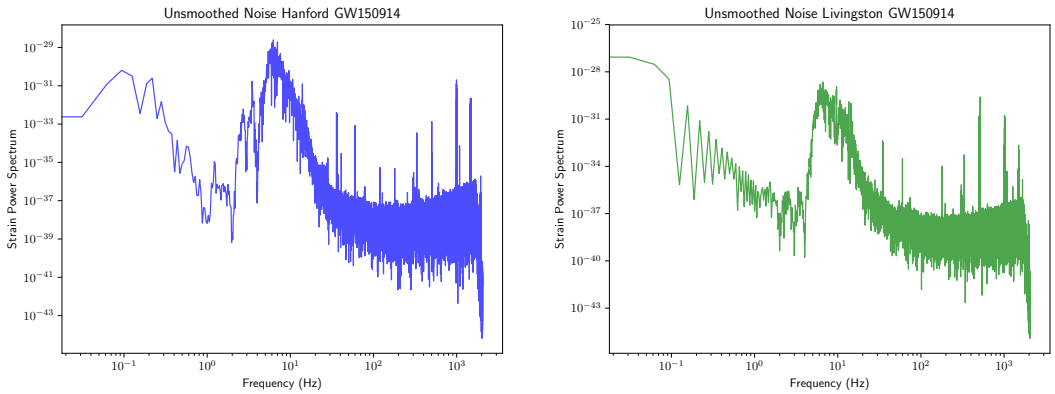

Figure 3: Plots of the unsmoothed estimated noise for just the GW150914 event at the Hanover and Livingston detectors.

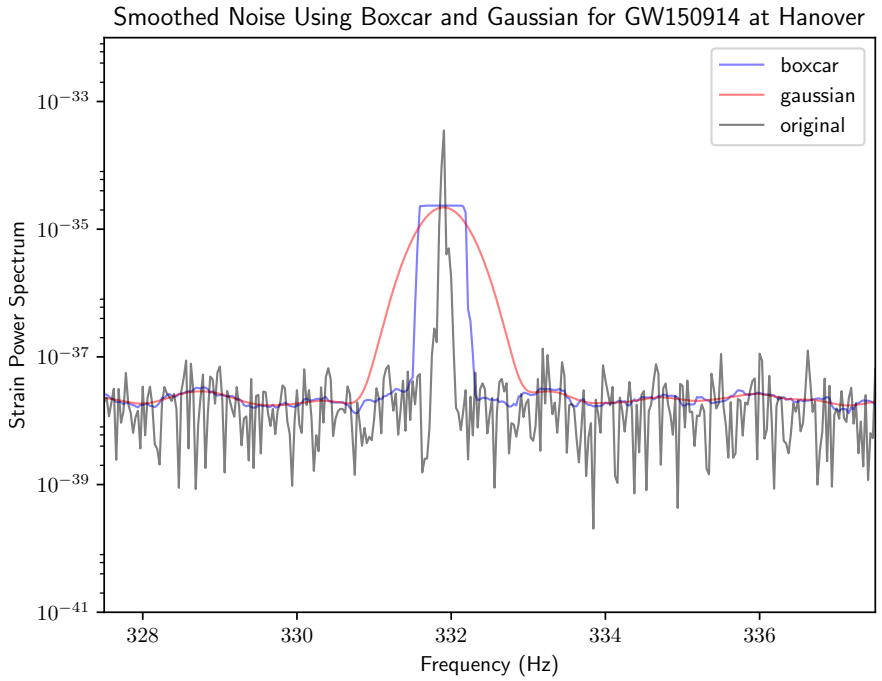

Figure 4: A plot of the GW150914 noise at the Hanover detector smoothed by both a Gaussian and a boxcar function, zoomed into a particular peak.

a particular peak to demonstrate these effects. Since the boxcar seems to broaden spectral features less than the Gaussian, we ultimately use the boxcar smoothing in our analysis.

You may also notice that there are odd turn-overs in the power spectra shown in Figure 3 at about 10 Hz and 1700 Hz. Since these turn-overs are likely unphysical, we set the noise beyond those frequencies to zero for each model.

The resulting averaged, smoothed, and excised noise models are shown in Figure 5. These are the final noise models that we use in our matched filtering.

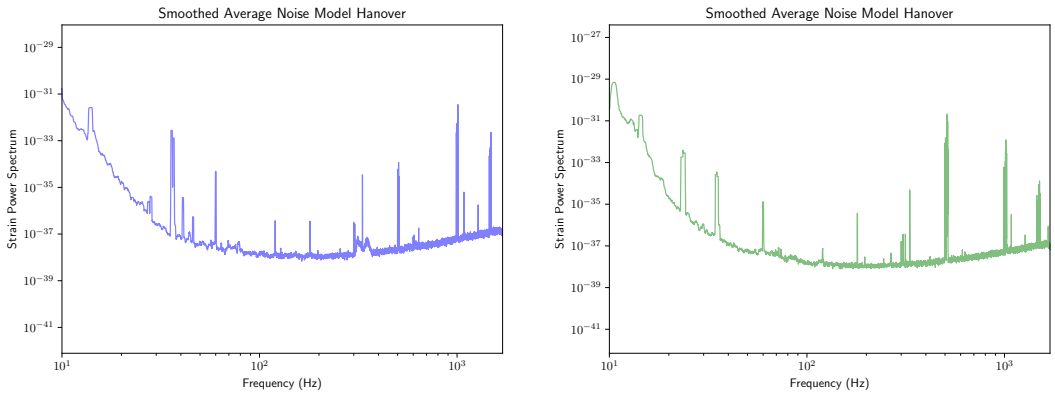

Figure 5: Plots of the smoothed noise averaged over all events for both the Hanover and Livingston detectors.

3 Prewhitening

Note that in section 1, we split the numerator into $N^{-1/2}d$ and $A^T N^{-1/2}$. This was done to facilitate pre-whitening. Pre-whitening is an operation that makes a dataset behave more like white noise, where white noise is a random signal with equal intensity at different frequencies. We can show that this is true by the following:

$$\chi^2 = n^T N^{-1} n = n^T N^{-1/2} N^{-1/2} n = \left(N^{-1/2} n\right)^T \mathbb{1} \left(N^{-1/2} n\right)$$

Where here we have essentially changed bases $n \to \tilde{n} = N^{-1/2} n$. Therefore, since the noise matrix in the new basis is the identity $\tilde{N}^{-1} = \mathbb{1}$, the new noise $\tilde{n}$ must be Gaussian with a standard deviation of 1, or white noise.

When you take the power spectrum of the strain it actually looks more like pink noise, which has a frequency spectrum that is inversely proportional to the frequency of the signal. However, once you prewhiten the data by multiplying it by $N^{-1/2}$, the power spectrum becomes whitened (see Figure 3) You can see that the power spectra for both become flatter, as expected. Pre-whitening both the data and the template makes our matched filter more accurate since the template will be less likely to identify noise features as actual signal.

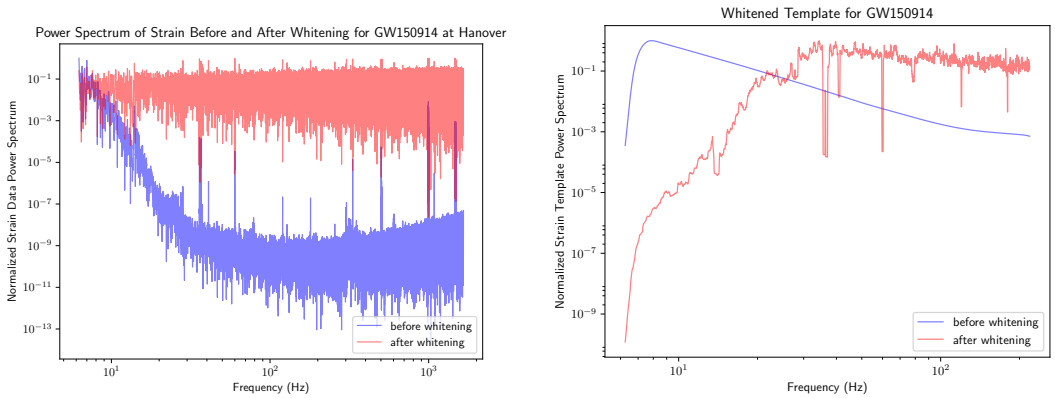

Figure 6: A plot of the power spectrum of the GW150914 strain and template at the Hanover detector before and after prewhitening.

4 Results

In **pset3_main.py**, we implement all that we have described above to obtain SNR's for the LIGO detections in each of the example datasets. To run **pset3_main.py** and get the results, you first need to set the variable `data_loc` to the location of all of the example datasets and the `BBH_events_v3.json` file. **pset3_main.py** also requires that the `simple_read_ligo.py` script is in the same directory. The script reports the SNR of each event detection at each detector, as well as the combined SNR from both detectors. It also reports the frequency from each event where half the weight comes from above that frequency and half below.

The resulting plots of SNR versus time for each event at each detector are shown on the next page in Figure 7. The output of the script is given by:

```
Event  GW150914
SNR for Hanford: 22.387049631
SNR for Livingston: 15.6928764759
Combined SNR: 27.339465307
Half-weight frequency for Hanford: 99.15625 Hz
Half-weight frequency for Livingston: 117.40625 Hz

Event  LVT151012
SNR for Hanford: 7.28374726657
SNR for Livingston: 7.13100191438
Combined SNR: 10.1933391264
Half-weight frequency for Hanford: 83.6875 Hz
Half-weight frequency for Livingston: 98.84375 Hz

Event  GW151226
SNR for Hanford: 8.95906654632
SNR for Livingston: 6.99326603951
Combined SNR: 11.3653263605
Half-weight frequency for Hanford: 83.9375 Hz
Half-weight frequency for Livingston: 98.03125 Hz

Event  GW170104
SNR for Hanford: 9.05397903101
SNR for Livingston: 7.21682976154
Combined SNR: 11.5783059254
Half-weight frequency for Hanford: 91.0625 Hz
Half-weight frequency for Livingston: 107.0 Hz
```

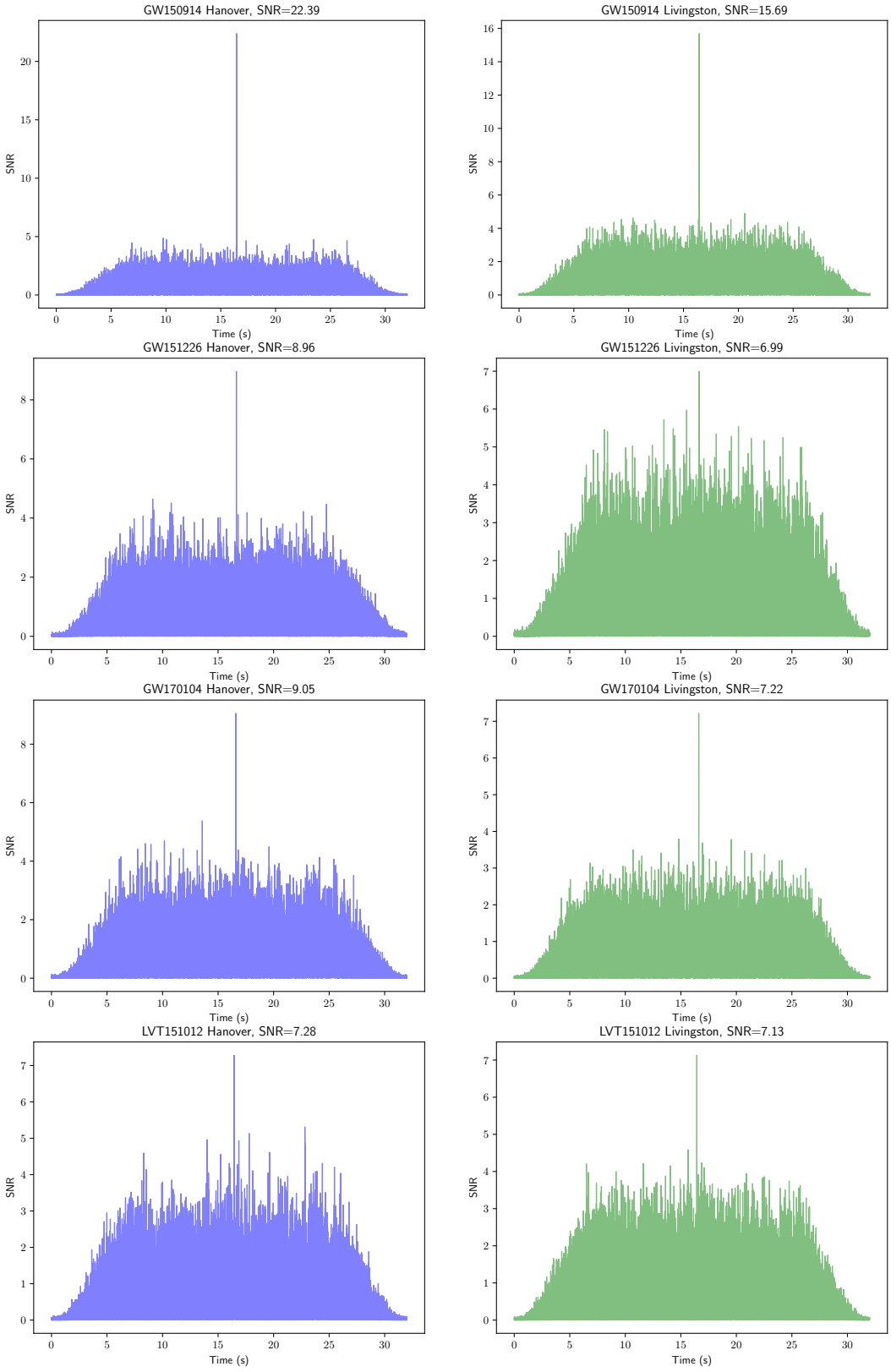

Figure 7: Signal to noise versus time plots at the Hanover and Livingston detectors for each of the events.