Gravitational Wave Signal Processing

Jason Glowney
University of Washington
Department of Applied Mathematics

Abstract

This report is an exploration on the application of signal processing to study the gravitational event GW170814, that was detected on August 14, 2017, representing a binary black hole coalescence. We use the techniques of Fourier Domain representation, amplitude spectral density, noise filtering, spectrograms and principle component analysis to help isolate the gravitational wave signals from background noises in the strain data.

Sec. I. - Introduction and Overview

In 1916, Albert Einstein first proposed the existence of gravitational waves upon finding that solutions to the linearized weak-field equations had wave characteristics. These waves encompassed spatial strains that traveled at the speed of light, resulting from time variations of the mass quadrupole of the source. It would take nearly a century for technology to advance in order to detect the incredibly small amplitudes of the these gravitational waves. It was on September 14, 2015, that the newly updated Laser Interferometer Gravitational Wave detectors in Hanford Washington (H1) and Livingston Louisiana (L1), would simultaneously observe a transient gravitational wave signal.

We fast forward to August 14, 2017, where the Advanced Virgo detector in Italy, and the two Advanced LIGO detectors in the United States, detected the gravitational wave signal of the coalescence of two stellar mass black holes, which was dubbed GW170814. Analysis of this event would reveal a merger of two black holes with approximately 30.5 and 25.3 solar masses at a distance of 540 Mpc from earth. [1]

Sec. II. - Theoretical Background

Strain

The advanced laser interferometers that listen for gravitational waves rely on strain as the means to do this. Each detector, in essence, acts as a modified Michelson interferometer to measure gravitational strain as differences in the length of its orthogonal arms. There are mirrors, that act as test masses, at the end of each 4 kilometer long arm for the LIGO detectors, that is monitored for length change. This difference can be expressed as

$$\Delta L(t) = \delta L_x - \delta L_y = h(t) \cdot L, \tag{1}$$

where *L* is the length of the orthogonal arms and *h(t)* is the gravitational-wave strain amplitude projected on the detector. [2]

Power and Amplitude Spectral Densities

We use spectral densities as a means to find the breakdown of variance in a signal with respect to frequency. [3] We let $\{X_k\}$ define a weakly stationary random process, that dissipates sufficiently fast to be described as

$$\sum_{k=-\infty}^{\infty} |\gamma(k)| < \infty. \tag{2}$$

If ***Eq.(2)*** above is satisfied, we can define the power spectral density as

$$h(\omega) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \gamma(k) e^{i\omega k} \qquad -\pi \leq \omega \leq \pi. \tag{3}$$

We have the reader note, that $h(\omega)$ and $\gamma(k)$ represent a Fourier transform pair, and if given one we can find the other. This pair provides complimentary information on $\{X_k\}$ in the lag of (k) and frequency (ω) domains. We can show the compliment transform as

$$\gamma(k) = \int_{-\pi}^{\pi} h(\omega) e^{i\omega k} d\omega \qquad k = 0 \pm 1, \pm 2, \dots \tag{4}$$

The amplitude spectral density is closely related to the above, and is simply calculated by taking the square root of the power spectral density

$$h(\omega) = \sqrt{\frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \gamma(k) e^{i\omega k}} \qquad -\pi \leq \omega \leq \pi. \tag{5}$$

The Spectrogram

The spectrogram is a visual form of the spectrum of frequencies of a sound or signal varying with time and are frequently used in the field of signal processing. The creation of a spectrogram in digital processing involves using the FFT, where sampled data in the time domain is broken into overlapping pieces, and then the Fourier transform is applied to each piece to calculate the magnitude of its frequency spectrum. [4]

The process of creating the spectrogram of a signal *s(t),* is essentially computing the squared magnitude of its short-time Fourier transform and we can state this as

$$spectrogram(t, \omega) = |STFT(t, \omega)|^2. \tag{6}$$

Principle Component Analysis (PCA)
We can apply the PCA onto a complex and seemingly random set of data, in efforts to simplify the system by way of finding lower dimensional representations. We find dynamics of the system, without governing equations, all the while, eliminating noise and redundant data. Data redundancies can be ascertained in the covariance matrix, where C_x is a square, symmetric $m \times m$ matrix, with a diagonal that represents the variance of particular measurements. The redundancies between measurements are manifest in large values for the off-diagonals, while smaller off-diagonal values are consistent with increased statistical independence between the measurements. We take a large data collection matrix X, where *n* is the number of columns in *X*, and signifies the number of data points in the particular measurement. We then express the covariant matrix C_x as

$$C_x = \frac{1}{n-1} XX^T. \tag{7}$$

With eigenvalue decomposition we can express XX^T as

$$XX^T = S\Lambda S^{-1}. \tag{8}$$

The above suggests that instead of working with *X*, we can instead considering working with a transformed variable in the principle component basis as

$$Y = S^T X. \tag{9}$$

With our new basis, we consider its covariance as

$$C_Y = \frac{1}{n-1} YY^T \rightarrow \frac{1}{n-1}(S^T X)(S^T X)^T \rightarrow \frac{1}{n-1} S^T (XX^T) S \rightarrow \frac{1}{n-1} S^T (XX^T) S \rightarrow \frac{1}{n-1}(S^T S)\Lambda(SS^T)$$

$$\therefore C_y = \frac{1}{n-1}\Lambda. \tag{10}$$

In the above basis, the principle components are the eigenvectors of XX^T and the j^{th} component of the diagonal C_y is the variance of *X* along x_j.

Using the SVD we can define a transformed variable

$$Y = U^* X, \tag{11}$$

Where *U* is the unitary matrix of the SVD: $X = U\Sigma V^*$. We can now compute the variance in *Y* as

$$C_Y = \frac{1}{n-1} YY^T \rightarrow \frac{1}{n-1}(U^* X)(U^* X)^T \rightarrow \frac{1}{n-1} U^* (XX^T) U \rightarrow \frac{1}{n-1}(U^* U)\Lambda(UU^*)$$

$$\therefore C_Y = \frac{1}{n-1}\Sigma^2. \qquad (12)$$

We can now see the link between the SVD and the eigenvalue method, where we have that $\Lambda = \Sigma^2$. [5]

Sec. III. - Algorithm Implementation and Development

For this report, we used strain data from the LIGO website [6] for the GW170814 event. The three strain files were in'hdf5' format, 32 seconds in length, with sampling rates of 4096 Hz, with the gravitation wave event centered at approximately 16 seconds into the dataset. Each respective strain data file was from one of the three individual advanced gravitational wave detectors.

Much of the tools for gravitational wave study on the LIGO.org site make use of the Python programming language. For this reason, we used Python for much of the data extraction and analysis of the strain data. We then used MATLAB on the whitened data to perform our PCA analysis.

We make use of the following to visualize and help interpret our data:

- Time Series Plots – To look at the pre-processed strain data along our GPS time interval.
- Amplitude Spectral Density Plots – Plotting our strain signals in the Fourier domain gives us information about the frequency contents of our data.
- Spectrograms – These enable us to visualize the power spectral density of our strain signal in a time series. We can look for higher powers over the signal, possibly indicating a gravitational wave event, and zoom in on the area to look for the 'chirp' of the event.
- Signal to Noise Plots – Allow us to visualize frequency spikes that rise above the noise of the signal at specific times.
- PCA Mode Plots – Help us to hone in on the dominant modes of the signal and further eliminate noise in the signal.

Filtering of our signals plays an important role in isolating gravitational waves from our signal data. We utilize the following in our analysis to help achieve whitening of our data:

- Blackman Windows - To eliminate excess noise that is present along the lower frequencies to avoid spectral leakage.
- Interpolation – To fill in intermediate data points of our power spectral density for improved resolution.
- Butterworth Filters - To further filter our signal data to the frequencies of expectation for celestial gravitational events. These are typically in the range of 25 to 350 Hz. We use a 4th order Butterworth filter for our strain data.

Python modules
We used the following Python modules:

- Numpy.py and scipy – for scientific computing
- H5py.py – for working with 'hdf5' files
- Readligo.py – LIGO module to aid in reading ligo data files
- Matplotlib – Python plotting module

MATLAB Commands
We used the following MATLAB command:

- *[U,S,V] = svd(MATRIX)* - To do our singular value decompositions on our data matrices.

Sec. IV. - Computational Results

We start with our simple strain vs time series below in ***Fig. 1*** and note that the amplitudes of the waves are extremely small being on the order of 10^{-18}, while have the reader note that the gravitational waves are actually three orders or magnitude lower at 10^{-21}! Visualizing this plot with the naked eye, we can see that detecting the GW170814 event within our unfiltered strain data is not possible. The peak of the GW170184 should be centered at time '0.0' and looking at the graph we see very different wave morphologies from each detector.

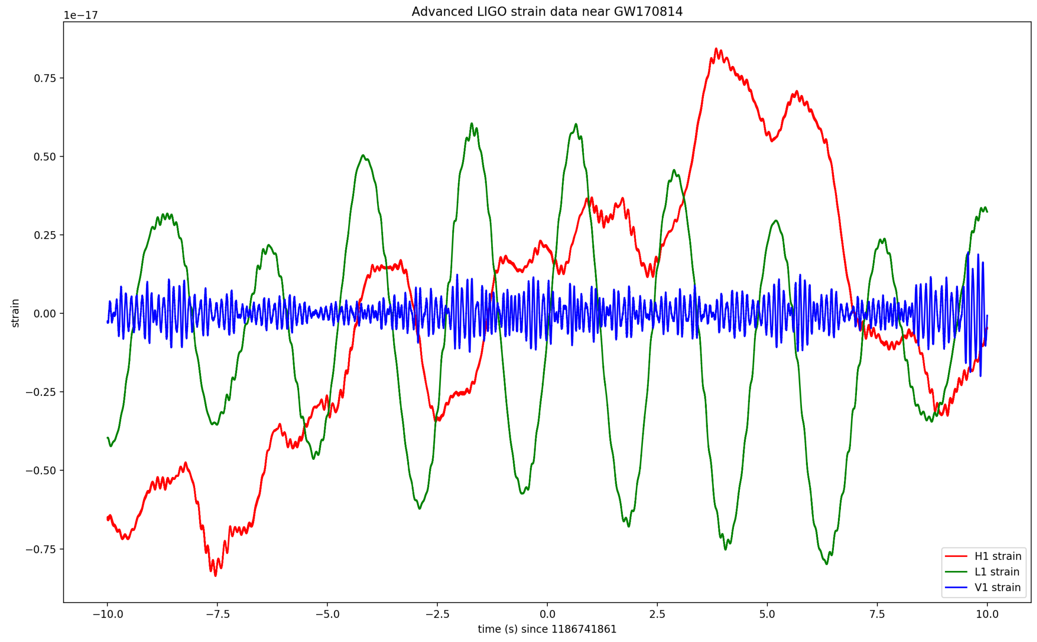

Fig. 1 – *GW170814 Strain vs Time Series for H1, L1 and V1 Detectors*

When we put our strain data into the Fourier domain, we start to get an idea of their frequency contributions. We plot in ***Fig. 2***, our amplitude spectral densities for all three detectors. By plotting the frequency along the x-axis in log form, we see that around 100 Hz there is a dip in the values for the ASD down to the order of 10^{-23} strain/$\sqrt{Hz}$. This is called the 'bucket' in LIGO parlance and represents the sensitivity for the detector for gravitational waves. We note that L1 and H1 are more sensitive than V1. The large amplitude spikes along multiples of 500 Hz, represent the internal harmonics of the lasers in the detectors. We also see large spikes for the Virgo detector around 50 Hz, which are due to localized seismic artifacts at the site. It is from about 50 Hz to 300 Hz where the GW170814 resides in its frequency spectrum. [7]

We continue in ***Fig. 3*** with our spectrograms for the L1, H1 and V1 detectors around the event. We see striking white bands along the 500 Hz, 1000 Hz, and 1500 Hz spectrums that are due to the harmonics of the lasers, most notably in H1 and L1 panes. It is the forms along the bottom of the spectrograms, that give us the hints of the gravitational waves of our GW170814 event. We see that for the V1 detector on the far right, we have more difficulty discerning these spectral bands.

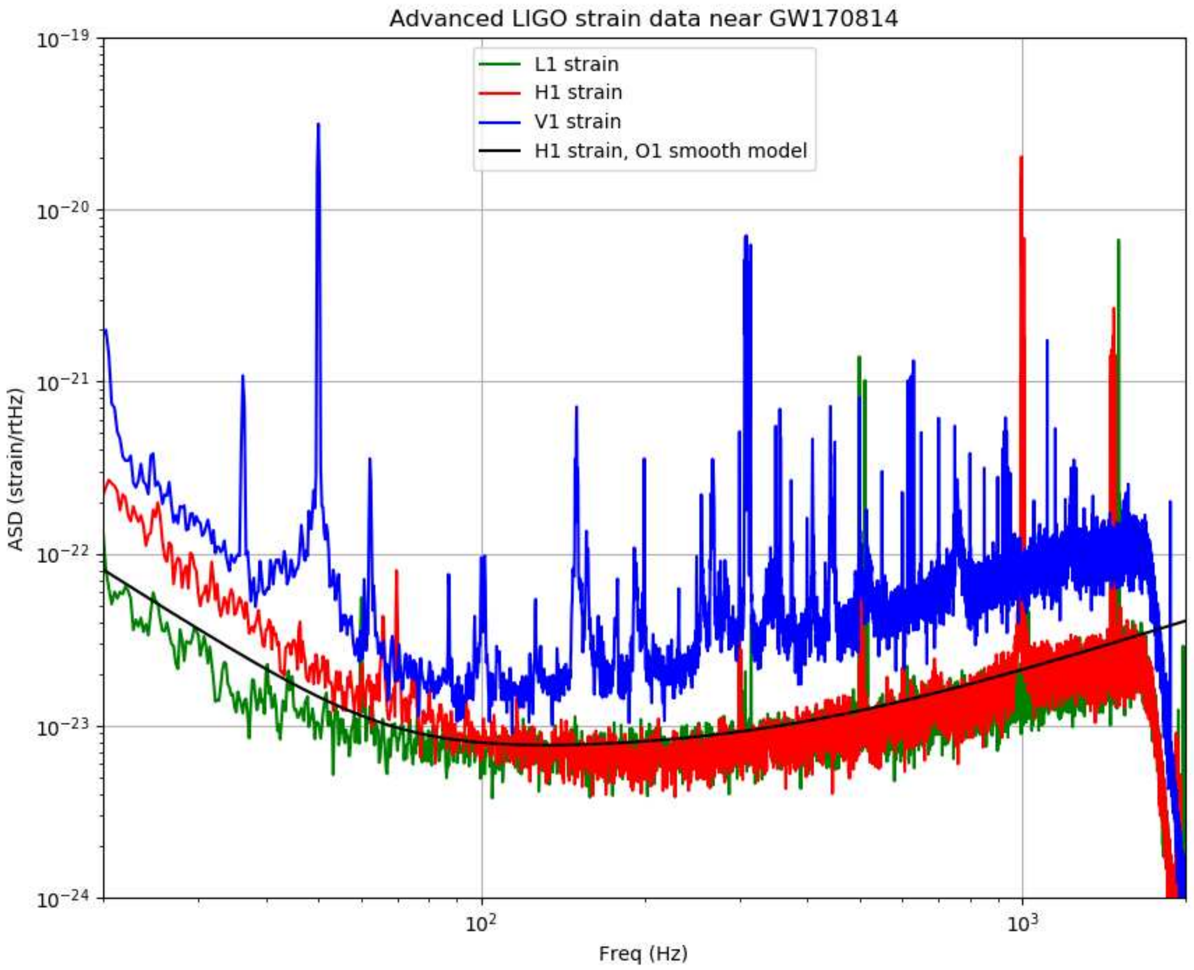

Fig. 2 – *Amplitude Spectral Density Plot around the GW170814 Event*

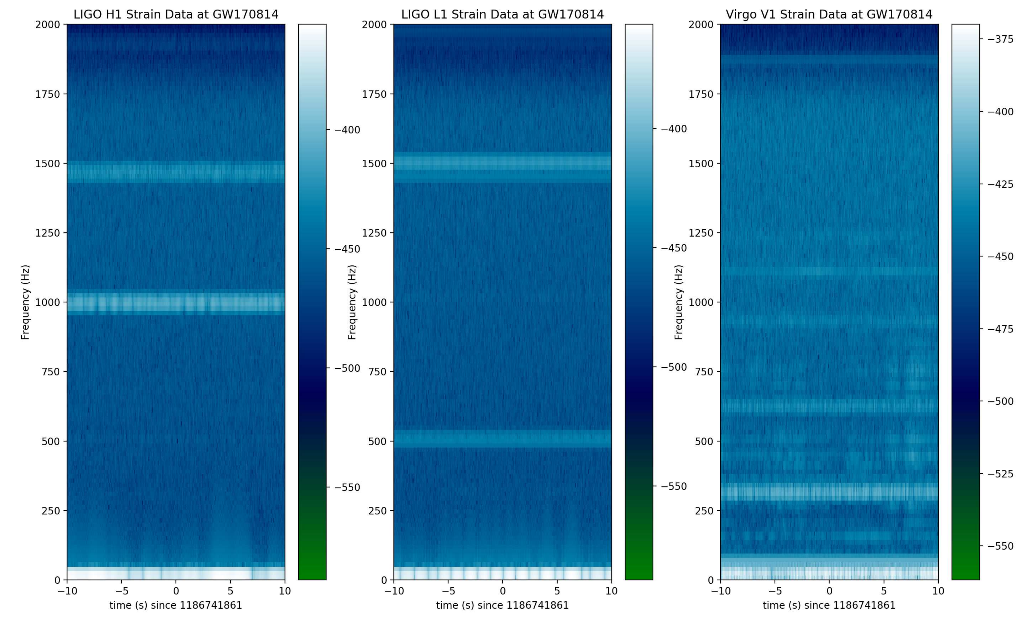

Fig. 3 – *GW170814 Spectrograms for H1, L1 and V1 detectors*

In ***Fig. 4*** below, we zoom in around the centered event at time '0.0' and we can see 'chirps' (white spectral intensity bands), in our H1 and L1 spectrograms, representing the detection of our GW170814 event. Once again, in our V1 spectrogram, we have difficulties with seeing our 'chirp' potentially from the decreased sensitivity of the detector.

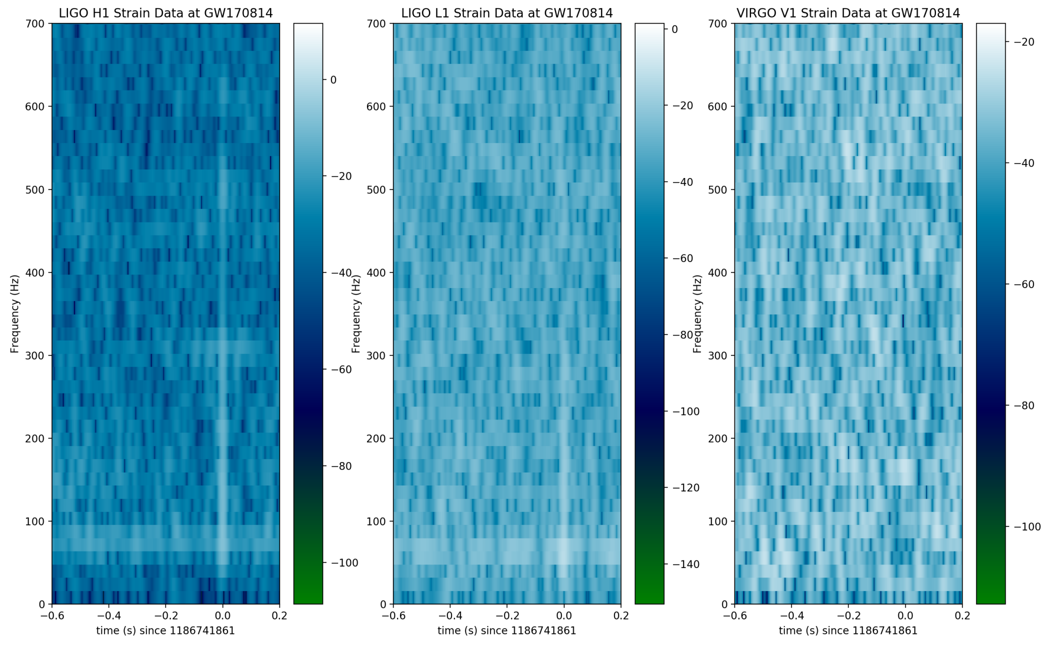

Fig. 4 – *'Chirps' for H1, L1 and V1 detectors. Signal is easily seen for H1 and L1 but not for V1.*

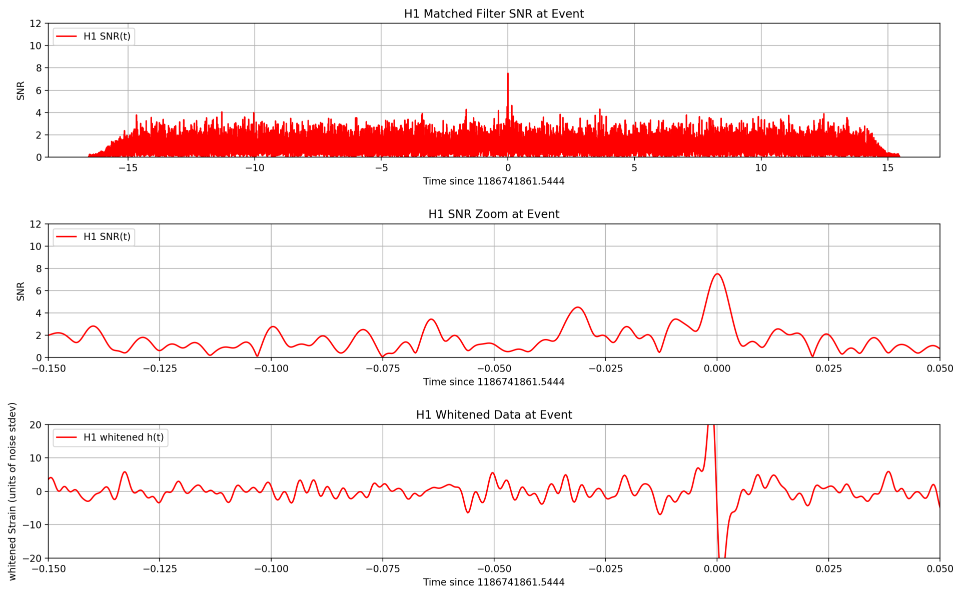

Fig. 5 – GW170814 - *H1 Signal to Noise (top), H1 SNR Zoom at event (middle) and H1 Whitened Signal at event (bottom).*

We plot in ***Figs. 5-7***, our signal to noise ratios for the whole 32 seconds of strain data, a zoomed in SNR at the center of the event and whitened strain data in for each of the gravitational wave detectors. For H1 and L1, we see a very large spike in amplitude for our signal to noise ratios center at time zero, with the L1 detector showing the overall highest amplitude of all the detectors. We see very evident spikes in strain levels as well for both H1 and L1.

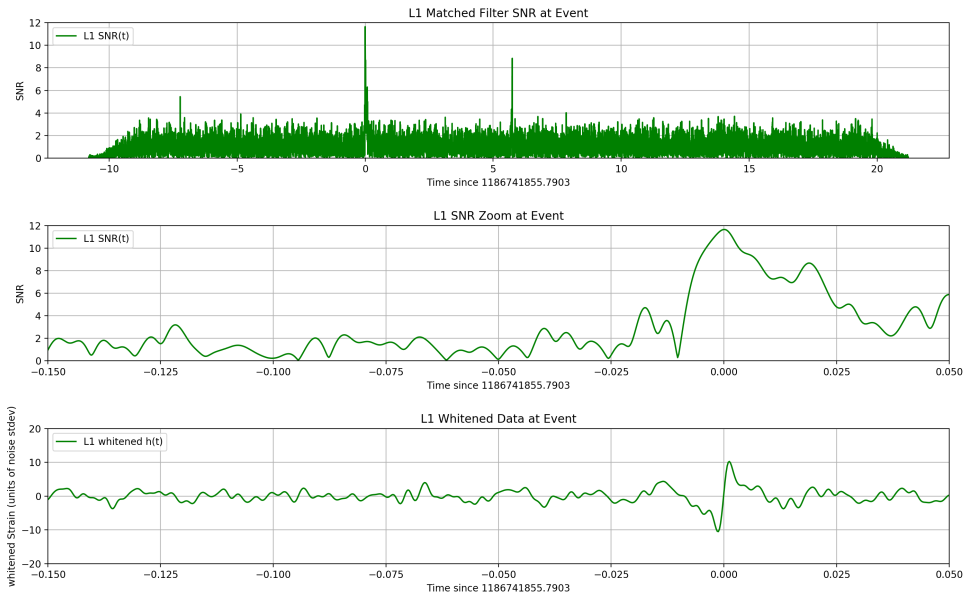

Fig. 6 – GW170814 - *L1 Signal to Noise (top), L1 SNR Zoom at event (middle) and L1 Whitened Signal at event (bottom).*

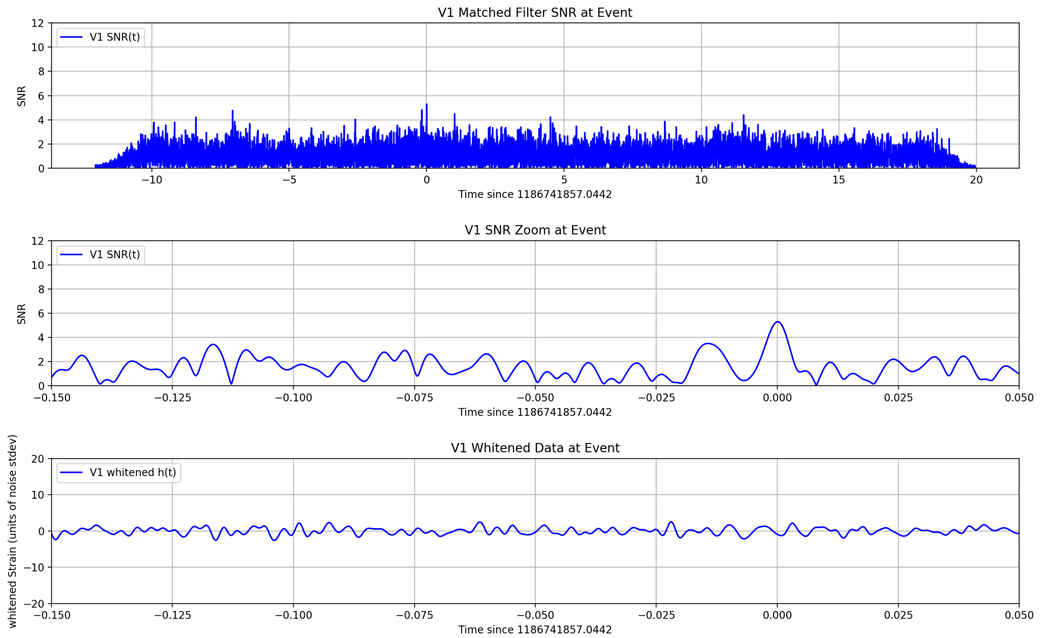

Fig. 7 – GW170814 - *V1 Signal to Noise (top), V1 SNR Zoom at event (middle) and V1 Whitened Signal at event (bottom).*

For our V1 signal to noise ratios, we see our lowest amplitudes and at that same time have more difficulty in seeing an obvious strain pattern in our whitened data.

We move on, to take our whitened signal data and utilize principle component analysis to try and isolate their dominant modes. We plot in ***Fig. 8***, our diagonal variances' contributions,

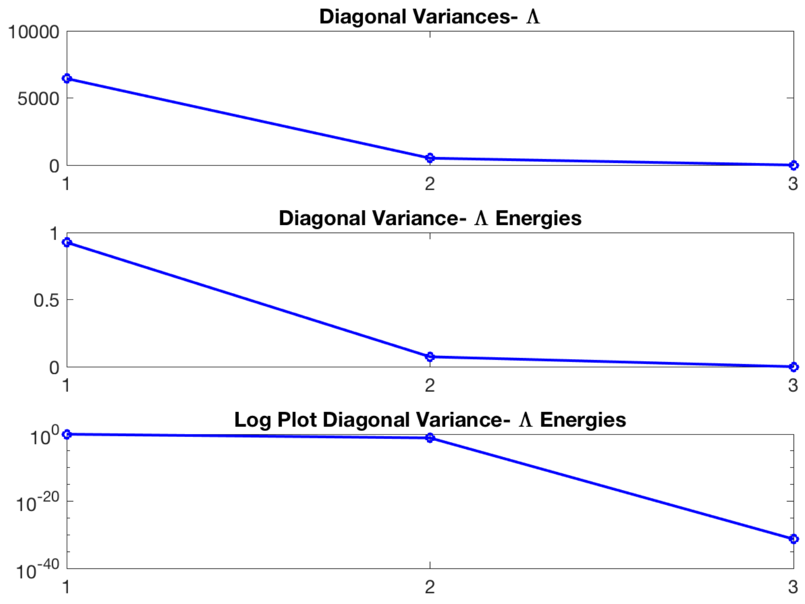

Fig. 8 – *Variances of Whitened GW170814 Signal*

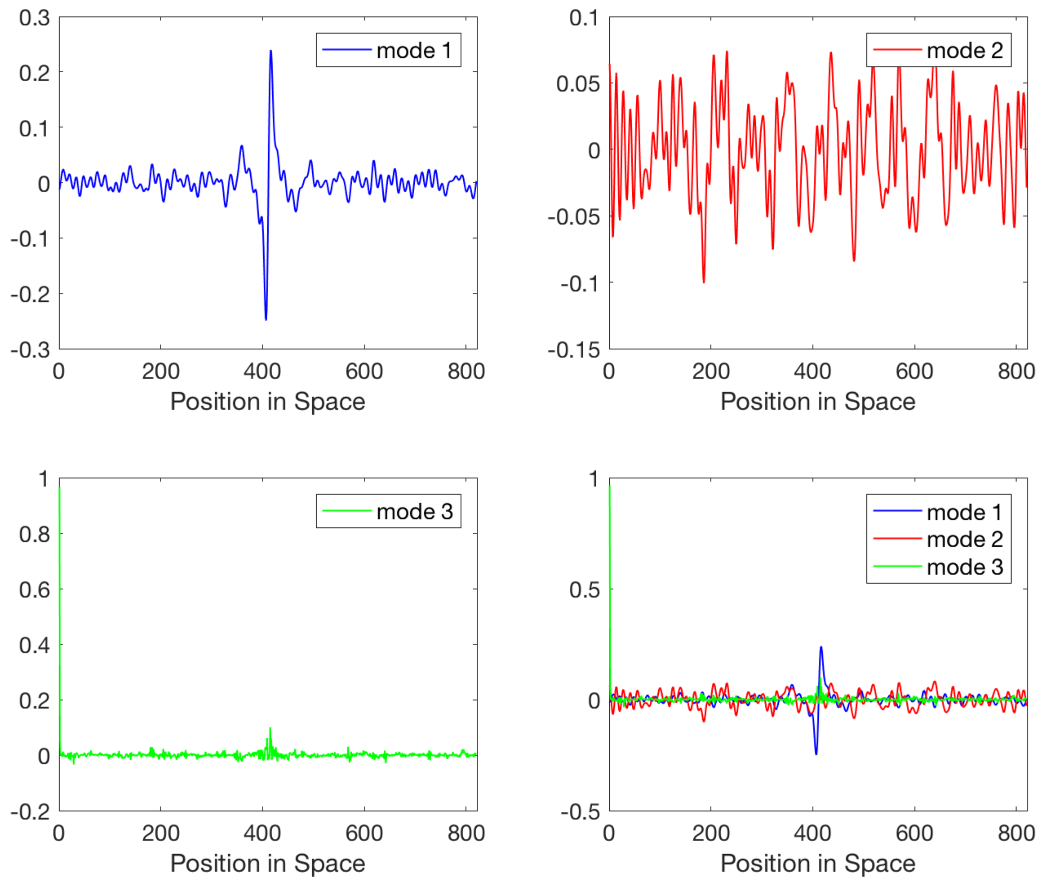

Fig. 9 – *GW170814 PCA Modes*

seeing that by variance two, we have nearly 100% of the energy of the system. We plot above in ***Fig. 9***, the three modes of the system and all modes together. We see a very distinguishable waveform in the first PCA mode, that shows the gravitational wave event nicely. We see by mode 3, we have basically background noise, but there is a remnant of the gravitational wave spike that still remains in the center of the plot.

We finish below with our principle component analysis waveform plot, reconstructed with our first 2 PCA modes. We see front and center, our gravitational wave event on the plot, as well as increasing in amplitude waveforms, that give us evidence of the increasing frequency of the binary black hole merger over time for its inspiral. The mode reconstruction shows the amplitude spike just before the coalescence being several times larger than preceding initial wave amplitudes.

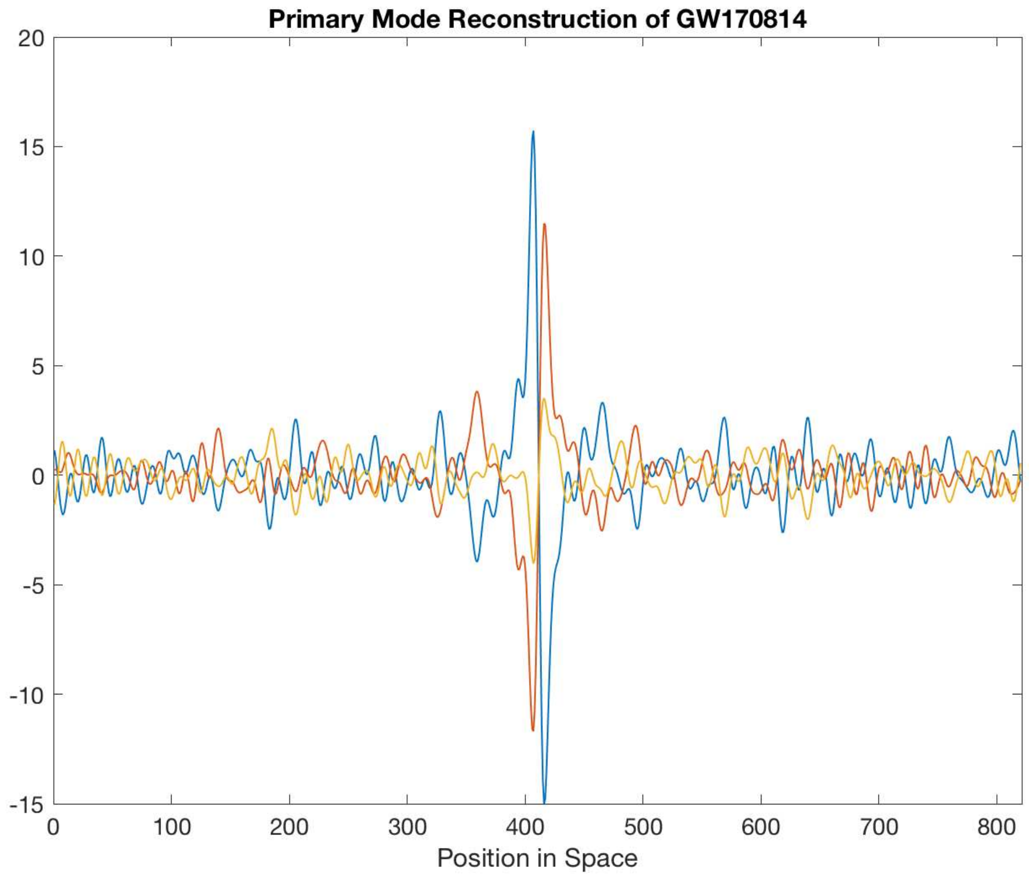

Fig. 10 – *GW170814 PCA Modes 1 and 2 Reconstruction*

Sec V. - Summary and Conclusions

This report demonstrates the application of signal processing to help extract meaningful gravitational wave strain data from the advanced laser interferometers used for their detection. We were easily able to isolate the GW170184 event for the signal from the LIGO detectors in Hanford and Livingston (H1, L1), but had more difficulty in doing so for the Virgo detector in Italy. This difficulty is perhaps related to decreased sensitivity for the V1 detector, evidenced on our ASD plot secondary to having only a 3km arm, and the position and orientation of the detector. We were none the less, able to appreciate spikes for V1 in the signal to noise ratios at the time of the event, that did coincide with the signals from H1 and L1. We were able to successfully employ principle component analysis to both further isolate our gravitational wave event and to decrease additionally the signal noise around the event.

References

[1] B.P. Abbott et al. (2016). "Observation of Gravitational Waves from a Black Hole Merger". Physical Review Letters

[2] https://en.wikipedia.org/wiki/Deformation_(mechanics)

[3] https://en.wikipedia.org/wiki/Spectral_density

[4] https://en.wikipedia.org/wiki/Spectrogram

[5] N. Kutz (2013). "Data-Driven Modeling & Scientific Computation". Oxford University Press.

[6] https://losc.ligo.org/events/GW170814/

[7] B.P. Abbott et al. (2017). "GW170814: A Three-Detector Observation of Gravitational Waves from a Binary Black Hole Coalescence". Physical Review Letters

Appendix A

Python modules
We used the following Python modules:

- Numpy.py and scipy – for scientific computing
    - Scipy.interpolate.interp1d to interpolate ASD data
    - Scipy.signal.butter for butterworth filtering
    - Numpy.fft for Fourier Transforms
    - Numpy.blackman for Blackman filtering

- H5py.py – for working with 'hdf5' files

- Readligo.py – LIGO module to aid in reading LIGO data files
    - Readligo.loaddata to extract strain, time and channel data

- Matplotlib – Python plotting module
    - Matlabplotlib.mlab.psd for power spectral densities

MATLAB Commands
We used the following MATLAB command:

- *[U,S,V] = svd(MATRIX)* - To do our singular value decompositions on our data matrices.

Appendix B

B1
Python Code

```
# Jason Glowney AMATH 582 Gravitational Wave Signal Processing Project
# From GW 170814 (losc.ligo.org)
eventname = 'GW170814'

make_plots = 1
plottype = "png"


# Python Modules Import
import numpy as np
from scipy import signal
from scipy.interpolate import interp1d
from scipy.signal import butter, filtfilt, iirdesign, zpk2tf, freqz
import h5py
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import readligo as rl

# 4096 Sample Rate Files for Event GW170814
fn_H1 = 'H1.hdf5'                                # File name for H1 data
fn_L1 = 'L1.hdf5'                                                  # File name for L1 data
fn_V1 = 'V1.hdf5'                                # File name for V1 data
fn_template = 'GW170814_4_template.hdf5'         # File name for template waveform

#Read Stain, Time, Channels
strain_H1, time_H1, channel_dict_H1 = rl.loaddata(fn_H1,'H1')
strain_L1, time_L1, channel_dict_L1 = rl.loaddata(fn_L1,'L1')
strain_V1, time_V1, channel_dict_V1 = rl.loaddata(fn_V1,'V1')

time = time_H1
dt = time[1] - time[0]        #-- Time between samples
fs = int(1.0 / dt)            #-- Sampling frequency
tevent = 1186741861             # Set approximate event GPS time
fband = [25, 360.0]           # frequency band for bandpassing signal


# Plot 5 seconds around the event GW170814
deltat = 10
indxt = np.where((time >= tevent-deltat) & (time < tevent+deltat))
print(tevent)

if make_plots:
   plt.figure()
   plt.plot(time[indxt]-tevent,strain_H1[indxt],'r',label='H1 strain')
   plt.plot(time[indxt]-tevent,strain_L1[indxt],'g',label='L1 strain')
   plt.plot(time[indxt]-tevent,strain_V1[indxt],'b',label='V1 strain')
   plt.xlabel('time (s) since '+str(tevent))
   plt.ylabel('strain')
   plt.legend(loc='lower right')
   plt.title('Advanced LIGO strain data near '+eventname)
   plt.savefig(eventname+'_strain.'+plottype)
   plt.show()

#Amplitude Spectral Density (ASD) Plot
make_psds = 1
if make_psds:
   # number of sample for the fast fourier transform:
   NFFT = 4*fs
   Pxx_H1, freqs = mlab.psd(strain_H1, Fs = fs, NFFT = NFFT)
   Pxx_L1, freqs = mlab.psd(strain_L1, Fs = fs, NFFT = NFFT)
   Pxx_V1, freqs = mlab.psd(strain_V1, Fs = fs, NFFT = NFFT)

   # We will use interpolations of the ASDs computed above for whitening:
   psd_H1 = interp1d(freqs, Pxx_H1)
   psd_L1 = interp1d(freqs, Pxx_L1)
   psd_V1 = interp1d(freqs, Pxx_V1)

   #PSD for H1
```

```
    Pxx = (1.e-22*(18./(0.1+freqs))**2)**2+0.7e-23**2+((freqs/2000.)*4.e-23)**2
    psd_smooth = interp1d(freqs, Pxx)

if make_plots:
    # plot the ASDs, with the template overlaid:
    f_min = 20.
    f_max = 2000.
    plt.figure(figsize=(10,8))
    plt.loglog(freqs, np.sqrt(Pxx_L1),'g',label='L1 strain')
    plt.loglog(freqs, np.sqrt(Pxx_H1),'r',label='H1 strain')
    plt.loglog(freqs, np.sqrt(Pxx_V1),'b',label='V1 strain')
    plt.loglog(freqs, np.sqrt(Pxx),'k',label='H1 strain, O1 smooth model')
    plt.axis([f_min, f_max, 1e-24, 1e-19])
    plt.grid('on')
    plt.ylabel('ASD (strain/rtHz)')
    plt.xlabel('Freq (Hz)')
    plt.legend(loc='upper center')
    plt.title('Advanced LIGO strain data near '+eventname)
    plt.savefig(eventname+'_ASDs.'+plottype)
    plt.show()

#Whitening

# function to whiten data
def whiten(strain, interp_psd, dt):
    Nt = len(strain)
    freqs = np.fft.rfftfreq(Nt, dt)
    freqs1 = np.linspace(0,2048.,Nt/2+1)

    # whitening: transform to freq domain, divide by asd, then transform back,
    # taking care to get normalization right.
    hf = np.fft.rfft(strain)
    norm = 1./np.sqrt(1./(dt*2))
    white_hf = hf / np.sqrt(interp_psd(freqs)) * norm
    white_ht = np.fft.irfft(white_hf, n=Nt)
    return white_ht

whiten_data = 1
if whiten_data:
    # now whiten the data from H1,L1 and V1,(use H1 PSD):
    strain_H1_whiten = whiten(strain_H1,psd_H1,dt)
    strain_L1_whiten = whiten(strain_L1,psd_L1,dt)
    strain_V1_whiten = whiten(strain_V1,psd_V1,dt)

    # Filtering out high-frequency noise with Butterworth filtering:
    bb, ab = butter(4, [fband[0]*2./fs, fband[1]*2./fs], btype='band')
    normalization = np.sqrt((fband[1]-fband[0])/(fs/2))
    strain_H1_whitenbp = filtfilt(bb, ab, strain_H1_whiten) / normalization
    strain_L1_whitenbp = filtfilt(bb, ab, strain_L1_whiten) / normalization
    strain_V1_whitenbp = filtfilt(bb, ab, strain_V1_whiten) / normalization


# Spectrograms
```

```
if make_plots:
    #Strain Time Series Index
    indxt = np.where((time >= tevent-deltat) & (time < tevent+deltat))
    #FFT Time Interval and Overlap
    NFFT = int(fs/16)
    NOVL = int(NFFT*15./16)
    #Blackman Window to Limit Leak
    window = np.blackman(NFFT)

    #spec_cmap='viridis'
    spec_cmap='ocean'

    # Plot the H1 spectrogram:
    plt.figure(figsize=(10,6))
    plt.subplot(1,3,1)
    spec_H1, freqs, bins, im = plt.specgram(strain_H1[indxt], NFFT=NFFT, Fs=fs, window=window,
                                noverlap=NOVL, cmap=spec_cmap, xextent=[-deltat,deltat])
    plt.xlabel('time (s) since '+str(tevent))
    plt.ylabel('Frequency (Hz)')
    plt.colorbar()
    plt.axis([-deltat, deltat, 0, 2000])
    plt.title('LIGO H1 Strain Data at '+eventname)
    plt.savefig(eventname+'_H1_spectrogram.'+plottype)

    # Plot the L1 spectrogram:
    plt.subplot(1,3,2)
    spec_H1, freqs, bins, im = plt.specgram(strain_L1[indxt], NFFT=NFFT, Fs=fs, window=window,
                                noverlap=NOVL, cmap=spec_cmap, xextent=[-deltat,deltat])
    plt.xlabel('time (s) since '+str(tevent))
    plt.ylabel('Frequency (Hz)')
    plt.colorbar()
    plt.axis([-deltat, deltat, 0, 2000])
    plt.title('LIGO L1 Strain Data at '+eventname)
    plt.savefig(eventname+'_L1_spectrogram.'+plottype)

    # Plot the V1 spectrogram:
    plt.subplot(1,3,3)
    spec_H1, freqs, bins, im = plt.specgram(strain_V1[indxt], NFFT=NFFT, Fs=fs, window=window,
                                noverlap=NOVL, cmap=spec_cmap, xextent=[-deltat,deltat])
    plt.xlabel('time (s) since '+str(tevent))
    plt.ylabel('Frequency (Hz)')
    plt.colorbar()
    plt.axis([-deltat, deltat, 0, 2000])
    plt.title('Virgo V1 Strain Data at '+eventname)
    plt.savefig(eventname+'_L1_spectrogram.'+plottype)
    plt.show()


# Closeup around event to look for chirp

if make_plots:
    # FFT time interva and overlap
    NFFT = int(fs/16.0)
    NOVL = int(NFFT*15/16.0)
    window = np.blackman(NFFT)


    # Plot the H1 whitened spectrogram around the signal
    plt.figure(figsize=(10,6))
    plt.subplot(1,3,1)
    spec_H1, freqs, bins, im = plt.specgram(strain_H1_whiten[indxt], NFFT=NFFT, Fs=fs, window=window,
                                    noverlap=NOVL, cmap=spec_cmap, xextent=[-deltat,deltat])
    plt.xlabel('time (s) since '+str(tevent))
    plt.ylabel('Frequency (Hz)')
    plt.colorbar()
    plt.axis([-0.6, 0.2, 0, 700])
    plt.title('LIGO H1 Strain Data at '+eventname)
    plt.savefig(eventname+'_H1_spectrogram_whitened.'+plottype)

    # Plot the L1 whitened spectrogram around the signal
    plt.subplot(1,3,2)
    spec_H1, freqs, bins, im = plt.specgram(strain_L1_whiten[indxt], NFFT=NFFT, Fs=fs, window=window,
                                    noverlap=NOVL, cmap=spec_cmap, xextent=[-deltat,deltat])
    plt.xlabel('time (s) since '+str(tevent))
    plt.ylabel('Frequency (Hz)')
    plt.colorbar()
    plt.axis([-0.6, 0.2, 0, 700])
    plt.title('LIGO L1 Strain Data at '+eventname)
    plt.savefig(eventname+'_L1_spectrogram_whitened.'+plottype)

    # Plot the V1 whitened spectrogram around the signal
    plt.subplot(1,3,3)
    spec_H1, freqs, bins, im = plt.specgram(strain_V1_whiten[indxt], NFFT=NFFT, Fs=fs, window=window,
                                    noverlap=NOVL, cmap=spec_cmap, xextent=[-deltat,deltat])
    plt.xlabel('time (s) since '+str(tevent))
    plt.ylabel('Frequency (Hz)')
    plt.colorbar()
    plt.axis([-0.6, 0.2, 0, 700])
    plt.title('VIRGO V1 Strain Data at '+eventname)
    plt.savefig(eventname+'_V1_spectrogram_whitened.'+plottype)
    plt.show()

#Write H1,L1,V1 Whiten to files
np.savetxt('H1_white', strain_H1_whiten)
np.savetxt('L1_white', strain_L1_whiten)
np.savetxt('V1_white', strain_V1_whiten)


# read in the template
f_template = h5py.File(fn_template, "r")

# extract metadata from the template file:
template_p, template_c = f_template["template"][...]
template_offset = 16.
```

```
# PSD with blackman window to limit spectral leaking
NFFT = 4*fs
psd_window = np.blackman(NFFT)
NOVL = NFFT/2

# define the complex template, common to both detectors:
template = (template_p + template_c*1.j)
datafreq = np.fft.fftfreq(template.size)*fs
df = np.abs(datafreq[1] - datafreq[0])
dwindow = signal.tukey(template.size, alpha=1./8)
template_fft = np.fft.fft(template*dwindow) / fs

# loop over the detectors
dets = ['H1', 'L1', 'V1']
for det in dets:

    if det is 'L1':   data = strain_L1.copy()
    elif det is 'H1': data = strain_H1.copy()
    else:             data = strain_V1.copy()

    # -- Calculate the PSD of the data.  Also use an overlap, and window:
    data_psd, freqs = mlab.psd(data, Fs = fs, NFFT = NFFT, window=psd_window, noverlap=NOVL)

    # Take the Fourier Transform (FFT) of the data and the template (with dwindow)
    data_fft = np.fft.fft(data*dwindow) / fs

    # -- Interpolate to get the PSD values at the needed frequencies
    power_vec = np.interp(np.abs(datafreq), freqs, data_psd)

    # -- Calculate the matched filter output in the time domain:
    optimal = data_fft * template_fft.conjugate() / power_vec
    optimal_time = 2*np.fft.ifft(optimal)*fs

    # -- Normalize the matched filter output:
    sigmasq = 1*(template_fft * template_fft.conjugate() / power_vec).sum() * df
    sigma = np.sqrt(np.abs(sigmasq))
    SNR_complex = optimal_time/sigma

    # shift the SNR vector by the template length so that the peak is at the END of the template
    peaksample = int(data.size / 2)  # location of peak in the template
    SNR_complex = np.roll(SNR_complex,peaksample)
    SNR = abs(SNR_complex)

    # find the time and SNR value at maximum:
    indmax = np.argmax(SNR)
    timemax = time[indmax]
    SNRmax = SNR[indmax]

    if make_plots:

        # plotting changes for the detectors:
        if det is 'L1':
            pcolor='g'
            strain_whitenbp = strain_L1_whitenbp
        elif det is 'H1':
            pcolor='r'
            strain_whitenbp = strain_H1_whitenbp
        else:
            pcolor='b'
            strain_whitenbp = strain_V1_whitenbp

        # -- Plot the result
        plt.figure(figsize=(10,8))
        plt.subplot(3,1,1)
        plt.subplots_adjust(left=None, bottom=None, right=None, top=None,
                wspace=0.5, hspace=0.3)
        plt.plot(time-timemax, SNR, pcolor,label=det+' SNR(t)')
        plt.grid('on')
        plt.ylabel('SNR')
        plt.ylim([0,12])
        plt.xlabel('Time since {0:.4f}'.format(timemax))
        plt.legend(loc='upper left')
        plt.title(det+' Matched Filter SNR at Event')


        # zoom in
        plt.subplot(3,1,2)
        plt.subplots_adjust(left=None, bottom=None, right=None, top=None,
                wspace=0.1, hspace=0.5)
        plt.plot(time-timemax, SNR, pcolor,label=det+' SNR(t)')
        plt.grid('on')
        plt.ylabel('SNR')
        plt.ylim([0,12])
        plt.xlim([-0.15,0.05])
        plt.grid('on')
        plt.xlabel('Time since {0:.4f}'.format(timemax))
        plt.legend(loc='upper left')
        plt.title(det+' SNR Zoom at Event')
        plt.savefig(eventname+"_"+det+"_SNR."+plottype)
        #plt.show()

                                #Plot Whitened data
        plt.subplot(3,1,3)
        plt.subplots_adjust(left=None, bottom=None, right=None, top=None,
                wspace=0.1, hspace=0.5)
        plt.plot(time-tevent,strain_whitenbp,pcolor,label=det+' whitened h(t)')
        plt.ylim([-20,20])
        plt.xlim([-0.15,0.05])
        plt.grid('on')
        plt.xlabel('Time since {0:.4f}'.format(timemax))
        plt.ylabel('whitened Strain (units of noise stdev)')
        plt.legend(loc='upper left')
        plt.title(det+' Whitened Data at Event')
        plt.show()
```

```
#Write H1,L1,V1 Whiten to files
np.savetxt('H1_whitenbp', strain_H1_whitenbp)
np.savetxt('L1_whitenbp', strain_L1_whitenbp)
np.savetxt('V1_whitenbp', strain_V1_whitenbp)
```

B2
MATLAB Code

```
%Jason Glowney
%Amath 582 LIGO Project SVD/PCA Analysis

cd('/Users/jasonglowney/Desktop/LIGO')
H1=load('H1_whitenbp');
L1=load('L1_whitenbp');
V1=load('V1_whitenbp');

M=[H1 L1 V1];
figure(7)
plot(M)

A=[H1 L1 V1];
[u,s,v]=svd(A,'econ');

figure(1)
subplot(1,2,1)
plot(diag(s),'-ro','LineWidth',2)
title('Singular Values Trial 1','fontsize',22,'fontweight','bold')
ylabel('\sigma_j','fontsize',24,'fontweight','bold')
xlabel('mode j','fontsize',22,'fontweight','bold')
legend({'a'},'FontSize',20,'Location','southwest')
xt1 = get(gca, 'XTick');
yt1 = get(gca, 'YTick');
set(gca, 'FontSize', 14);
set(gcf,'color','w');
subplot(1,2,2)
semilogy(diag(s)/sum(diag(s)),'-bo','LineWidth',2)
title('Energy of \sigma_j Trial 1','fontsize',22,'fontweight','bold')
ylabel('Weighting - \sigma_j (weight * 100% =
Percentage)','fontsize',20,'fontweight','bold')
xlabel('mode j','fontsize',22,'fontweight','bold')
legend({'b'},'FontSize',20,'Location','southwest')
xt3 = get(gca, 'XTick');
yt3 = get(gca, 'YTick');
set(gca, 'FontSize', 14);
set(gcf,'color','w');

HX=H1((15.9*4096):(16.1*4096),:);LX=L1((15.9*4096):(16.1*4096),:);VX=V1((15.9
*4096):(16.1*4096),:);
X=[HX LX VX];
[s,n]=size(X);
mn=mean(X,2);
X=X-repmat(mn,1,n);
[u2,s2,v2]=svd(X/sqrt(n-1));
```

```
lambda=diag(s2).^2;
Y2=u2'*X;

figure(2)
subplot(3,1,1)
plot(lambda,'-bo','LineWidth',2) %diagonal variances
title('Diagonal Variances- \Lambda','fontsize',22,'fontweight','bold')
set(gca,'Fontsize',12),set(gca,'XTick',1:6)
xt1 = get(gca, 'XTick');
yt1 = get(gca, 'YTick');
set(gca, 'FontSize', 14);
set(gcf,'color','w');

subplot(3,1,2)
plot(lambda/sum(lambda),'-bo','LineWidth',2)
title('Diagonal Variance- \Lambda
Energies','fontsize',22,'fontweight','bold')
set(gca,'Fontsize',12),set(gca,'XTick',1:6)
xt1 = get(gca, 'XTick');
yt1 = get(gca, 'YTick');
set(gca, 'FontSize', 14);
set(gcf,'color','w');

subplot(3,1,3)
semilogy(lambda/sum(lambda),'-bo','LineWidth',2)
title('Log Plot Diagonal Variance- \Lambda
Energies','fontsize',22,'fontweight','bold')
set(gca,'Fontsize',12),set(gca,'XTick',1:6)
xt1 = get(gca, 'XTick');
yt1 = get(gca, 'YTick');
set(gca, 'FontSize', 14);
set(gcf,'color','w');

figure(3)
subplot(2,1,1)
plot(u2(:,1:3),'LineWidth',1),xlim([0 821]),xlabel('Position in
Space'),%set(gca,'XTick',1:6)
legend({'mode 1','mode 2','Mode 3'},'FontSize',14,'Location','northeast')
set(gca, 'FontSize', 14);
set(gcf,'color','w');
subplot(2,1,2)
plot(v2(:,1:3),'LineWidth',1),xlabel('Position in Time'),%xlim([0 mt-10])
title('Trial 1','fontsize',50)
legend({'mode 1','mode 2','mode 3'},'FontSize',14,'Location','northeast')
set(gca, 'FontSize', 14);
set(gcf,'color','w');

figure(4)
plot(Y2(1:3,:),'LineWidth',1)
title('Principal Components Data Projection -
GW170814','fontsize',44,'fontweight','bold')
legend({'PCA mode 1','PCA mode 2','PCA mode
3'},'FontSize',16,'Location','northeast')
xlabel('time','fontsize',20),%xlim([0 821])
xt1 = get(gca, 'XTick');
yt1 = get(gca, 'YTick');
set(gca, 'FontSize', 14);
set(gcf,'color','w');

figure(5)
subplot(2,2,1)
plot(u2(:,1),'b','LineWidth',1),xlabel('Position in Time'),xlim([0 821])
%suptitle('PCA Modes of GW170814','fontsize',30,'fontweight','bold')
legend({'mode 1'},'FontSize',14,'Location','northeast')
set(gca, 'FontSize', 14);
set(gcf,'color','w');
subplot(2,2,2)
plot(u2(:,2),'r','LineWidth',1),xlabel('Position in Time'),xlim([0 821])
legend({'mode 2'},'FontSize',14,'Location','northeast')
set(gca, 'FontSize', 14);
set(gcf,'color','w');
subplot(2,2,3)
plot(u2(:,3),'g','LineWidth',1),xlabel('Position in Time'),xlim([0 821])
legend({'mode 3'},'FontSize',14,'Location','northeast')
set(gca, 'FontSize', 14);
set(gcf,'color','w');
subplot(2,2,4)
plot(u2(:,1:3),'LineWidth',1),xlim([0 821]),xlabel('Position in 
Time'),%set(gca,'XTick',1:6)
legend({'mode 1','mode 2','mode 3'},'FontSize',14,'Location','northeast')
set(gca, 'FontSize', 14);
set(gcf,'color','w');

%PCA Modes of GW Waveforms
NR=u2(:,1:2)*s2(1:2,1:2)*v2(:,1:2)';
NRR=u2*s2*v2';
figure(6)
plot(NR,'LineWidth',[1]),xlabel('Position in Space'),xlim([0 821])
title('Primary Mode Reconstruction of 
GW170814','fontsize',30,'fontweight','bold')
set(gca, 'FontSize', 14);
set(gcf,'color','w');
```