Measuring Dark Energy with Binary Black Holes

Teresa Symons
Embry-Riddle Aeronautical University,
Daytona Beach, FL 32114 and
Cardiff University, Cardiff, CF24 3AA, UK
(Dated: August 14, 2013)

Abstract: We explore the information about cosmology that can be gleaned by simulating gravitational wave signals from binary black hole coalescence events. Using a galaxy catalog, we select a certain preferential set of galaxies to use as injections for binary black hole merger events. After generating various parameters randomly, we set up our network of detectors. We calculate the antenna response patterns and signal-to-noise ratio for each signal and each detector, allowing us to determine which signals would be detectable by which detectors in the network. After making our mock detections, we discern which original galaxies the signals could possibly come from by adding a certain amount of noise to the original parameters and searching within an error box on the sky for possible galaxy matches. We then use information about those galaxies to calculate their redshift. Then, we calculate the Hubble constant for each galaxy and use a histogram to determine the true value. After some experimentation with different approximations for the Hubble constant, we arrived at the value of 69.56 km/s*Mpc, given a theoretical value of 70 km/s*Mpc.

I. INTRODUCTION

Gravitational waves are theorized to be ripples in space-time that are caused by moving large masses. There are multiple likely sources of these waves, including the gravitational collapse of stars such as those that produce supernovae or gamma-ray bursts, gravitational wave pulsars such as stars with non-symmetric deformations, merging binary systems, and the stochastic background, which is the random gravitational wave field throughout the universe. This project deals primarily with merging binary systems as a gravitational wave source. These can be the mergers of two neutron stars, a neutron star and a black hole, or two black holes. Specifically, we will be dealing with binary black hole coalescence events. Black holes can create gravitational waves during inspiral, the period of orbital decrease as the two black holes grow closer to one another, and also during ring down, the phase after merging where the single black hole is distorted in shape. Binary black hole mergers are theoretically the strongest source of gravitational waves [4].

Binary black holes are important to this project because changes in their masses are caused by the accretion of dark energy. The orbital changes of a binary system are induced by energy loss from gravitational radiation. Because binary black holes radiate gravitational waves with a power proportional to their masses, these gravitational waves carry information about dark energy [4]. Dark energy is a theoretical energy that permeates all of space and accelerates the expansion of the universe. Its counterpart, dark matter, is seen by its gravitational effects on visible matter. In fact, the total mass-energy of the universe is roughly 4.6% ordinary matter, with 24% dark matter and 71.4% dark energy [6].

Studying gravitational waves can help us to learn more about general relativity and cosmology. Cosmologists desire to learn more about the structure and evolution of the universe. One of the most important ways to study this is to discover the densities of the various types of matter in the universe. The average density of matter in the universe determines if the universe is closed and finite, or open and infinite. If the value of the average density is close to the critical density between the two, the universe is flat but still infinite. Likewise, the distribution of dark matter and energy affects if the universe is expanding or contracting, and how quickly it is doing either. Simulating gravitational wave signals not only allows us to develop algorithms that could be used to calculate the various cosmological parameters from real gravitational wave detections, but also gives us a tool to see how various networks of detectors can help us determine these crucial matter densities and learn more about the nature of the universe [6].

II. BACKGROUND

A. Project Overview

In this project, we seek to determine the cosmological parameters from simulated gravitational wave signals of binary black hole mergers. To accomplish this, we use a galaxy catalog based on the Millennium Simulation to select galaxies for the black hole coalescence events. We randomly generate various parameters and create a mock detector network. Then, we determine which of the signals would actually be detected, and add some randomly generated noise to those signals. We then use that noise to search for galaxies in the catalog that could be possible sources for the signals, and then use the redshifts of those galaxies to determine the cosmological parameters. The main MATLAB code written for this project is located in Appendix A. Supplementary, original functions are included in Appendices B through D.

B. Millennium Simulation

For this project, the galaxy catalog we used is derived from the Millennium Simulation. This simulation was performed by the Virgo Consortium for Cosmological Supercomputer Simulations and is the largest N-body simulation ever carried out. The simulation is used to study the evolution of the universe and structure formation caused by dark matter [5]. The galaxy catalog was created by Dr. Laura Nuttall, and contains over 40 million galaxies. Figure 1 was created by Dr. Nuttall and shows a three-dimensional mapping of all the galaxies in her catalog.

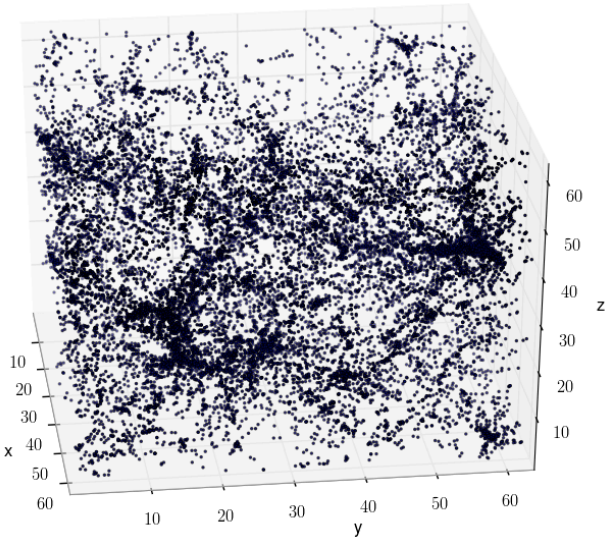

Figure 1: Map of galaxy catalog (Nuttall)

C. Network of Detectors

We looked at a specific network of detectors in order to detect the generated binary black hole coalescence events. Specifically, we used Advanced LIGO, including the Hanford, Washington and Livingston, Louisiana sites, Advanced VIRGO in Italy, LIGO India, KAGRA in Japan, and the Einstein Telescope that would be located somewhere in Europe. Not all of these detectors are up and running yet, but we used their power spectral densities to determine their potential detection power.

D. The Einstein Telescope

The Einstein Telescope (ET) is the most important detector in this project's network. As a third generation gravitational wave detector, it would be more sensitive than the advanced detectors by a factor of 10 [2]. This increase in sensitivity would give it a much high signal-to-noise ratio than any of the other network detectors, allowing for a high detection rate. It is suspected that the Einstein Telescope would allow us to better determine the cosmological parameters. One component of this project is to examine how well these parameters can be determined both with and without the Einstein Telescope included in the detector network. Figure 2 shows an artistic rendering of what the Einstein Telescope could look like.

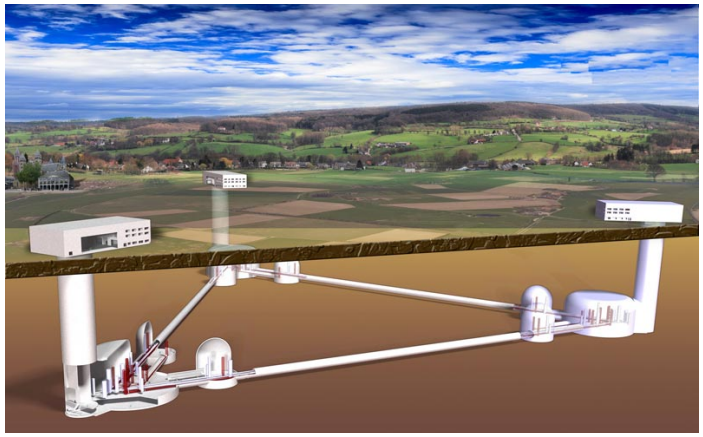

Figure 2: The Einstein Telescope (http://www.et-gw.eu/etimages)

E. Cosmological Parameters

This project seeks to determine the parameters that define the ΛCDM model of the universe, the Λ - Cold Dark Matter model. This is the currently accepted model that states the universe has a cosmological constant and is populated by cold dark matter. The particular parameter that we find in this project is the Hubble Constant, H_0, which tells us the current

expansion rate of the universe. Other important parameters that could one day be derived from this work are Ω_Λ, the dark energy density, Ω_M, the dark matter density, and w, the dark energy equation of state [3]. Equation 1 shows the relationship between these four parameters, where z is redshift and Ω_d is Ω_Λ.

$$H(z) = H_0[\Omega_M(1+z)^3 + \Omega_d(1+z)^{3(1+w)}]^{1/2} \text{ [2]} \qquad (1)$$

III. SIGNAL INJECTIONS

A. Galaxy Catalog

The galaxy catalog contains over 40 million galaxies and is complete to 750 Mpc. For each galaxy, the sky position (right ascension and declination) is given. Also given is the luminosity distance and an error on that value, a number indicating how spiral or elliptical the galaxy is, and the luminosity with an error on that value. It would be computationally impractical to use the entire catalog, so we have developed a method for randomly selecting which galaxies to use.

B. Weighted Random Selection

We developed a program to randomly select any desired number of galaxies to use as binary black hole coalescence events. For the purpose of this project, it is assumed that every selected galaxy has a coalescence event. The program gives preference to elliptical galaxies over spiral galaxies 60% of the time, although this percentage is variable. The reason for this is that elliptical galaxies are older and therefore have greater chances of having a merger event. This selection is accomplished by generating a random number and then checking if that number falls between 0 and 0.6 or above 0.6. If the number is below 0.6, an elliptical galaxy will be selected. Secondly, the program gives a weighted preference to brighter galaxies by first dividing each luminosity value by the sum of all values. These values are then sorted and the cumulative sum of each value and all previous values is calculated. The program then selects galaxies with smaller cumulative sums and greater brightness values. Typically, 1,000 galaxies were selected for each trial. As many as 10,000 were selected for longer runs, and as few as 100 were selected for short trials. One thousand was the optimal number to get enough detections to use for the remainder of the program. Figure 3 shows the sky positions of all selected galaxies in right ascension versus declination.

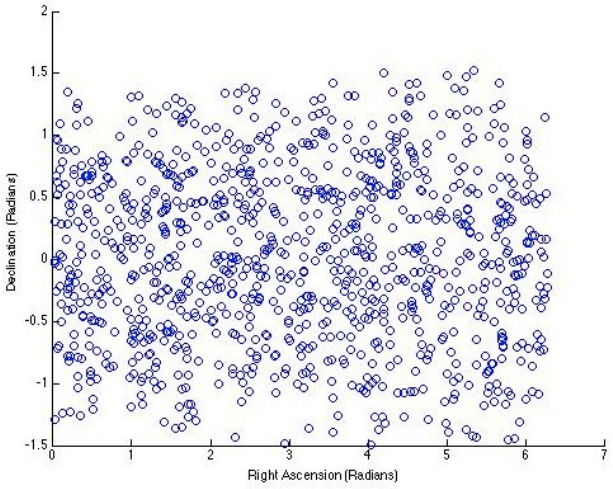

Figure 3: Sky positions of selected galaxies

The declination of the galaxies ranges from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$ radians. Right ascension ranges from 0 to 2π radians. Figures 4 and 5 illustrate this.

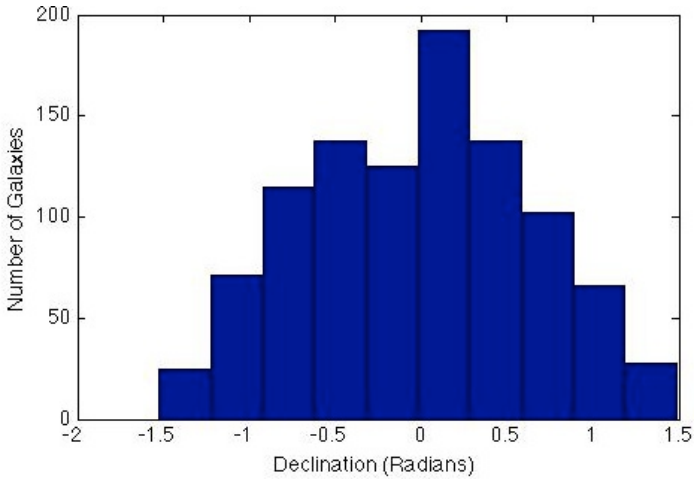

Figure 4: Distribution of declination

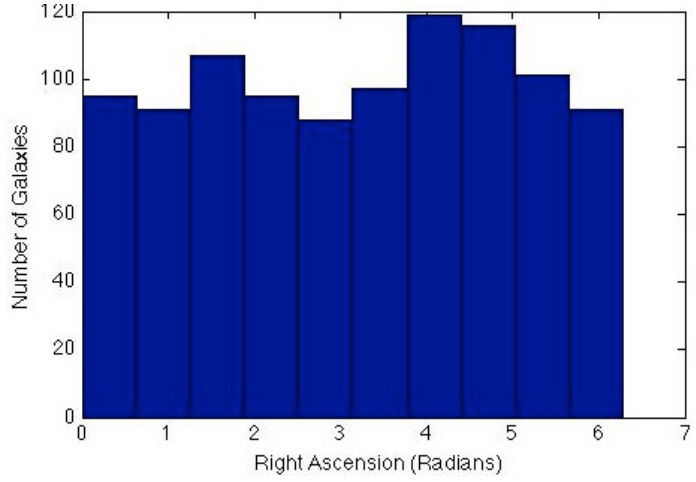

Figure 5: Distribution of right ascension

The values for luminosity distance were given in the catalog for each galaxy. They ranged from 0 to 750 Mpc. This is seen in Figure 6.

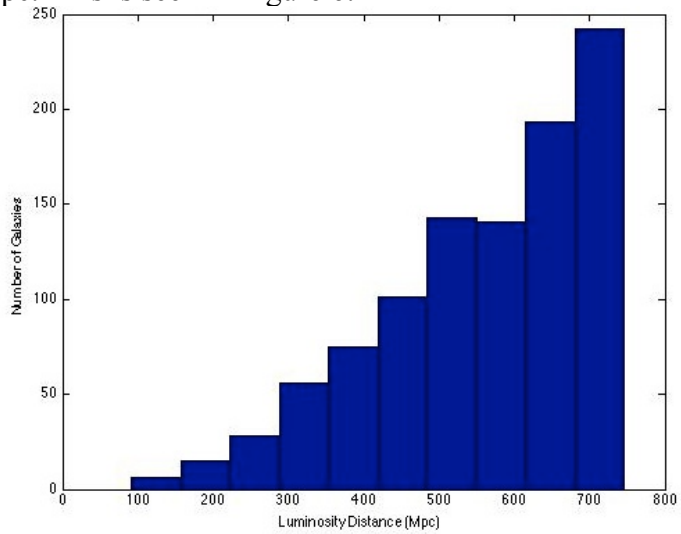

Figure 6: Distribution of distance

These are all of the parameters for each galaxy that were included in the catalog. Other parameters are needed to calculate the detectability of each signal, so those parameters were randomly generated.

C. Generating Parameters

The inclination angle, which is the angle between the line-of-sight to a merging binary from Earth and the angular momentum of the binary, is randomly generated and uniformly distributed in cosine. It ranges from 0 to π radians, as can be seen in Figure 7.

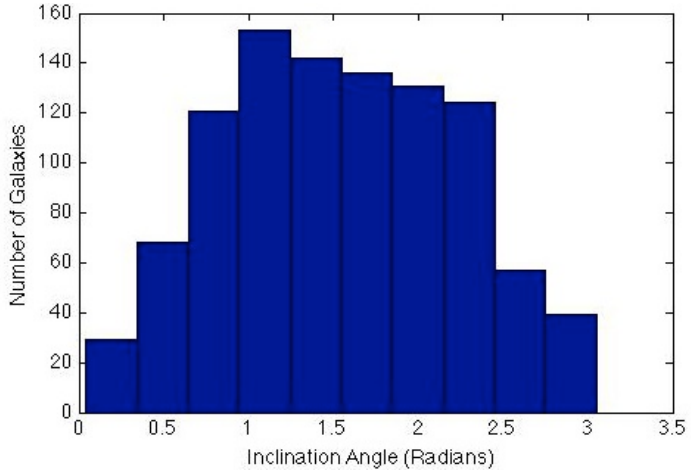

Figure 7: Distribution of inclination angle

The polarization angle is the orientation of the orbit of the binary on the sky with respect to the orientation of the detector. It is also randomly generated and uniformly distributed between 0 and $\frac{\pi}{2}$ radians, as is seen in Figure 8.

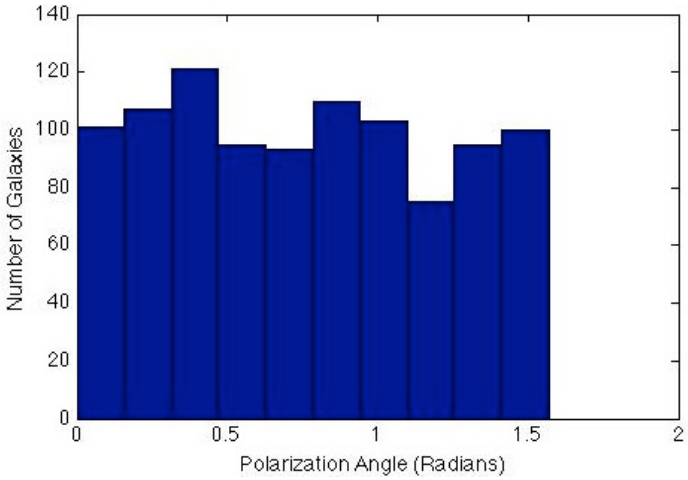

Figure 8: Distribution of polarization angle

IV. DETECTIONS

A. Detector Network

After creating our signals, we set up our detector network. The program is capable of using several combinations of the following detectors: Initial LIGO and VIRGO, Advanced LIGO and VIRGO, LIGO India, KAGRA, and ET. The various cases are: only initial LIGO and VIRGO; only advanced LIGO and VIRGO; advanced with LIGO India; only ET; advanced, India, and ET; and advanced, India, ET, and KAGRA. Typically, all detectors were used at once. Each detector in the network has a power spectral density function.

B. Making Detections

The masses of the black holes were generated randomly for each galaxy between three and 15 solar masses, which is what they would be for binary black hole systems. This is seen in Figures 9 and 10.

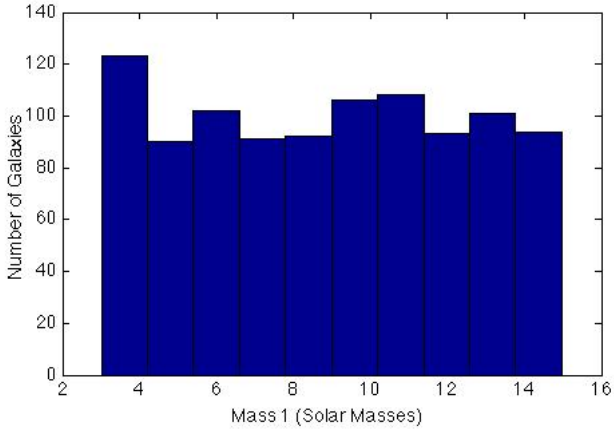

Figure 9: Distribution of Mass 1

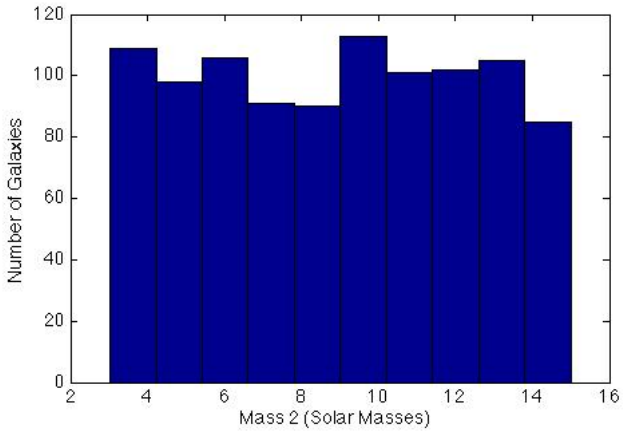

Figure 10: Distribution of Mass 2

In order to detect the mock signals from each galaxy, first the F_+ and F_x antenna responses were calculated for each detector and galaxy combination. Next, the signal-to-noise ratio (SNR) was calculated for each detector and galaxy combination using Equation 2, where G is the gravitational constant, C is the speed of light, M is the chirp mass (a composite of both masses), d_L is the luminosity distance, and ι is the inclination angle.

$$SNR = \frac{G^{\frac{5}{6}}}{C^{\frac{3}{2}}}\left(\frac{5}{24\pi^{\frac{4}{3}}}\right)^{\frac{1}{2}}\frac{M^{\frac{5}{6}}}{d_L}[F_+^2(1+(\cos\iota)^2)^2 + 4F_x^2(\cos\iota)^2]^{\frac{1}{2}}[\int_0^\infty df\,\frac{1}{f^{\frac{7}{3}}S(f)}]^{\frac{1}{2}} \quad (2)$$

The equation also uses the calculated antenna responses and the power spectral density function for each detector. After the SNR has been calculated for each galaxy and each detector, we figure out which galaxies would actually be detectable by our network. We search for galaxies that have SNR greater than five in at least two detectors, and then galaxies that have a quadrature sum SNR greater than 12. This allows us to realistically determine which galaxies would actually be detected in a real scenario. The typical detection rate using all detectors is about 99.5%, which is expected. This high detection rate is greatly helped by including ET, because ET typically produces much higher SNR values than the other detectors.

C. Adding Noise

After we have made our detections, we add noise to the parameters for only the detected galaxies. The purpose of this is to produce an error box that we can then use to search the catalog for galaxy candidates for each signal. For most of the parameters, the noise has Gaussian distribution with various widths for different parameters. The widths are as follows: 0.01 for the total mass, 0.1 for the mass ratio, 5 degrees for right ascension and declination, 3 for luminosity distance, 50 degrees for the inclination angle, and 25 degrees for the polarization angle. All of these widths are also divided by the system SNR, which is calculated by squaring all of the SNR values for each detector for one galaxy, and then taking the square root of that value. From this point forward, only the system SNR is used and not individual SNR values for particular detectors. Noise is also added to the system SNR with a chi-square distribution. Error values are randomly generated using the methods and widths described above, and then randomly added to or subtracted from all of the original values.

V. LOCATING SIGNALS

A. Reconstructing Error Box

After adding in the noise, we use it to reconstruct an error box on the sky and then search in that box for each signal to attempt to figure out which galaxy in the catalog the signal came from. All galaxies in each signal's error box are included as possible sources. For some signals there are few or no galaxies found, and for some there are hundreds.

The error box is formed by taking just one detected signal and its distance, right ascension, and declination. Then, we search for any galaxies that have a distance value of the signal distance plus or minus its noise value. The same is done for right ascension and declination. This gives us three lists of galaxies that match the signal's distance and position components. By finding galaxies present in all three lists we develop a master list of potential source galaxies for each signal. Figure 11 illustrates the error box concept.

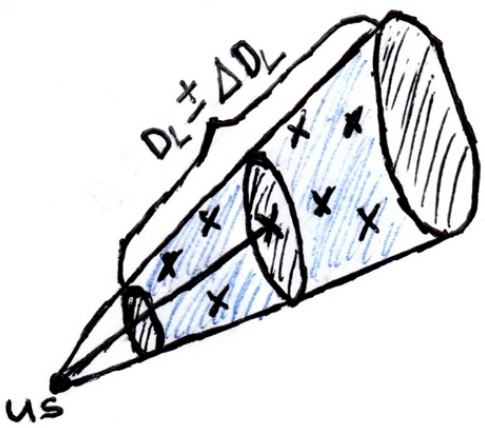

Figure 11: Error box

The mark in the center of the diagram is the actual galaxy that the signal came from, with a line to the point, Earth, that represents the real distance. The blue box includes the distance plus the noise and the distance minus the noise, and so includes the surrounding galaxies that will be found in the error box search.

B. Finding Hubble Parameter

After generating a list of potential sources for each signal, we can proceed with calculating the cosmological constants. From this point forward, we no longer use the list of signals we detected, but only the list of potential source galaxies. In a realistic scenario, gravitational waves would be detected and then the source galaxies for the signals would be determined, so we use only that data.

First, we calculated the redshift for each galaxy using the luminosity distance from the catalog, the desired dark matter percentage, 0.24, and a theoretical value of H_0, 70. Then, using the redshift value for each galaxy, the luminosity distance from the catalog, and the speed of light, we calculated the Hubble parameter using the approximation where redshift is much less than 1. The formula used is seen in Equation 3.

$$H_0 = \frac{z*C}{d_L} \; for \; z \ll 1 \; [2] \qquad (3)$$

Then we created a histogram of all the H_0 values for all of the source galaxies, with the hope that it would peak at the true value.

VI. RESULTS

A. First Results and Issues

Our first histograms all seemed to peak around 62.5 km/s*Mpc. This was slightly disappointing given that our target value was 70 km/s*Mpc. In addition, our histograms were all one-sided, with only a right side. Figure 12 shows an example of this result.

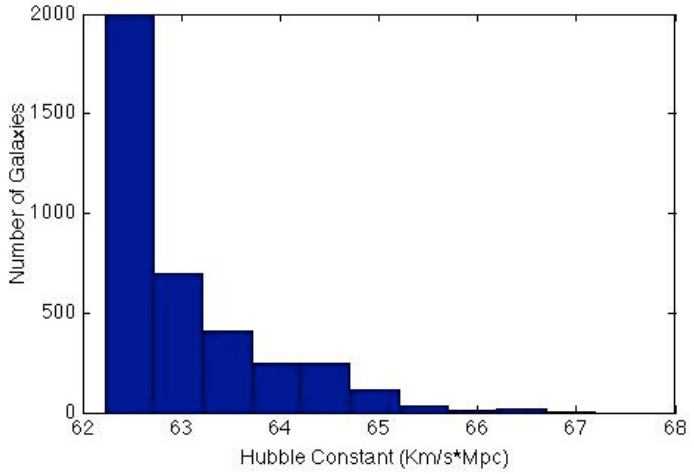

Figure 12: One-sided histogram for Hubble Parameter

We attempted multiple changes to try to get a more accurate value.

B. Changing Sigma Values

First, we tried decreasing the noise errors by a factor of 10. This proved to have no effect on the histogram's shape or the value of the Hubble constant. Next, we tried increasing the sigma values for the error box, essentially doubling it in size and then tripling it. This drastically increased the number of potential source galaxies, on the supposition that adding more values of the Hubble constant would help the real value to stand out more accurately. Figures 13 and 14 show these results.

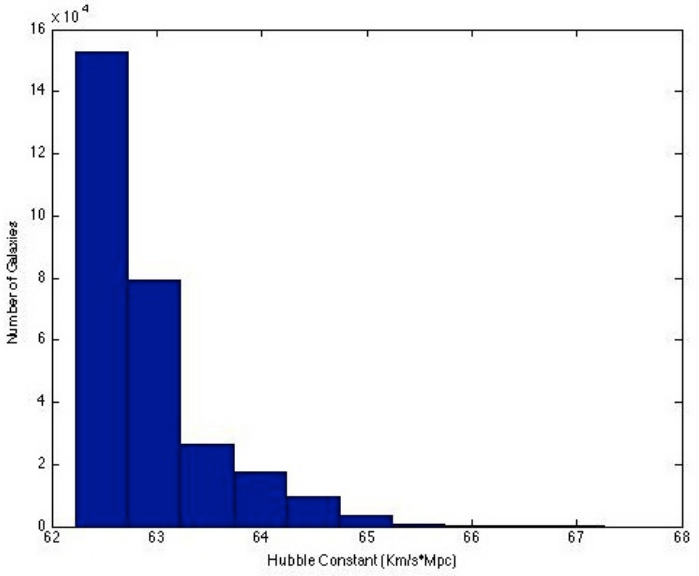

Figure 13: Doubled error box

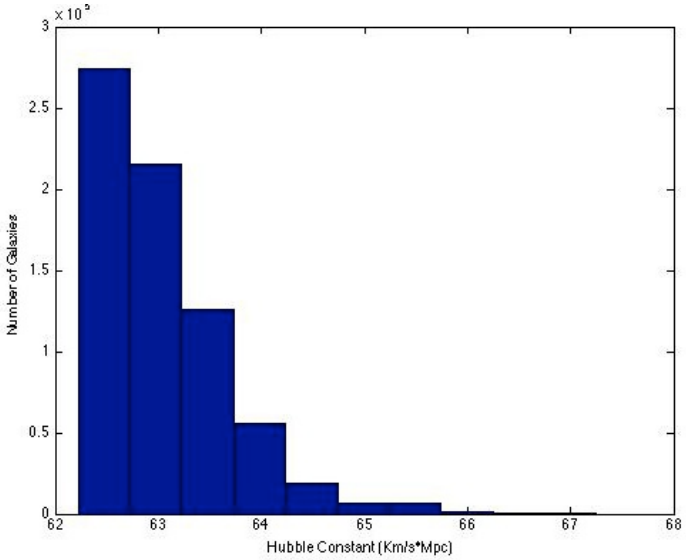

Figure 14: Tripled error box

These histograms continue to show a value for the Hubble constant of 62.5 km/s*Mpc. The only difference is that the number of galaxies increases vastly. In addition, the histograms are still one-sided. In order to verify that there was no issue with the way the program was selecting or finding galaxies, we calculated the redshift and Hubble constant for all 40 million galaxies in the catalog and found that the histogram for those Hubble parameter values peaked in the same place, 62.5 km/s*Mpc. The average Hubble constant value for the whole catalog was consistent with the original histograms, verifying that the issue was with the way the Hubble constant was being calculated and not with the galaxy selection or some other component of the program.

C. Removing ET

We also tried removing ET from the detector network to see if this would have an effect on the Hubble constant calculation. Only using Advanced LIGO and VIRGO, LIGO India, and KAGRA produced a 94.4% detection rate, as opposed to the previous 99.5% detection rate with ET. This was expected due to ET's vastly superior detection power. There was ultimately no change in the Hubble constant, demonstrating that the detection rate has no effect on this calculation. Figure 15 shows that the value is still 62.5 km/s*Mpc.

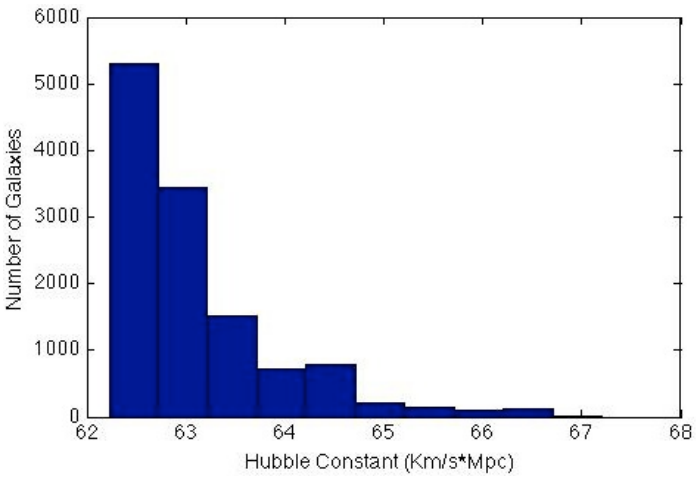

Figure 15: Hubble Constant without ET

This also matches the value for the entire galaxy catalog, showing that the value is not dependent on any detector set but either on the catalog itself or the way in which it is calculated.

D. Changing Approximation

We then switched to a different, more precise method of approximating the Hubble constant. The assumption that redshift is much less than 1 is not actually that accurate, given that our redshift values were typically on the order of 0.1. The new approximation is given in Equation 4, where z is redshift and c is the speed of light. d_L is the luminosity distance, but instead of using the distance for the source galaxy from the catalog, we used the distance with the noise estimate figured in for the corresponding signal galaxy.

$$H_0 = \frac{(1+z)c}{d_L} \int_0^z \frac{dz'}{h(z)'} \text{ [2]} \qquad (4)$$

In this equation, h(z) is given by Equation 5. This is a simplified form of Equation 3, where the dark matter and energy densities are included. The values used were 0.24 and 0.76.

$$h(z) = [\Omega_M(1+z)^3 + \Omega_d]^{1/2} \text{ [2]} \qquad (5)$$

The result of using this new approximation was a highly improved Hubble constant value. The new value is 69.56 km/s*Mpc, which is extremely close to the expected value of 70 km/s*Mpc. However, the histogram is still one-sided. Figure 16 shows the new result.

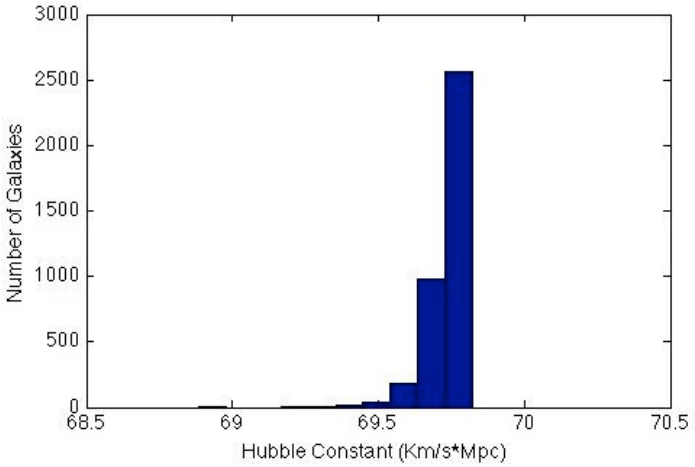

Figure 16: New Hubble constant

These results are much more in line with what we expected, and show a big improvement on the initial results. We would still like to investigate why the histogram is one-sided.

VII. CONCLUSIONS

This project has been moderately successful in determining the Hubble constant from theoretical black hole coalescence signals. At the time when gravitational wave detections are made, this work will be useful in making that calculation. We have learned that we can accurately determine the Hubble constant from such observations with some degree of error. This project has a vast potential future. The next steps would include determining the other cosmological parameters, including the dark matter and energy densities and the dark energy equation of state. In this way, binary black hole mergers could give us information about the nature of dark matter and energy. This project is similar to the ET mock data challenge, which involves using actual injections of mock ET data and detecting the signals through an analysis pipeline. This project is essentially a mock of the mock data challenge in that it does not use mock ET data and simulates the detection process. However, this project can offer similar benefits in helping to determine ET's capabilities and detection power. This project may be limited by the galaxy catalog, which is only complete to 750 Mpc. A larger, more complete catalog with higher redshifts may yield different results. Another next step for this project could involve repeating the process with a new catalog, and then going on to calculate the other parameters.

VIII. REFERENCES

[1] Amaro-Seoane, P. et al. (2011). Einstein telescope design study: vision document. Retrieved from http://www.et-gw.eu/etdsdocument

[2] Arun, K. G. et al. (2007). Higher signal harmonics, LISA's angular resolution, and dark energy. *Physical Review D, 76*(10), doi: 10.1103/PhysRevD.76.104016

[3] Planck Collaboration. (2013). Planck 2013 results. xvi. cosmological parameters. Retrieved from http://arxiv.org/pdf/1303.5076v1.pdf

[4] Sathyaprakash, B .S., & Schutz, B. (2009). Physics, astrophysics and cosmology with gravitational waves. *Living Reviews in Relativity,* 12. Retrieved from http://www.livingreviews.org/lrr-2009-2

[5] The Virgo Consortium. (2009). The millennium simulation. Retrieved from http://www.virgo.dur.ac.uk/index.php?subject=millennium

[6] WMAP Science Team. (2011). Cosmology: the study of the universe. Retrieved from http://map.gsfc.nasa.gov/universe/

Appendix A – Main Program

```
%This code created by Teresa Symons 2013
%Requires galaxy catalogue to run

clc
clearvars -except galaxy spiral elliptical galred hub
close all
%**For this code to work, you must first load the milliennium full
%catalogue, spiral, elliptical, galred, and hub**

% select configuration of detectors:
```

```
% initial = HLV
% enhanced = HLV
% advanced = HLV
% advanced/india = HLVI
% et = E
% et and advanced = HLVIE
% et/advanced/kagra = HLVIKE

config = 'et/advanced/kagra';

%load appropriate galaxy catalog
%you MUST run this if you have not loaded a catalogue yet
%load('fake_millennium_full.mat')

%split galaxies by spiral and elliptical
%this doesn't need to be run again, but you MUST load 'spiral' and
%'elliptical' for this code to work
% spiral =
[galaxy.dec(galaxy.type==1),galaxy.ra(galaxy.type==1),galaxy.type(galaxy.type==1),galaxy.Lfr
ac(galaxy.type==1),galaxy.Lfracerr(galaxy.type==1),galaxy.d(galaxy.type==1),galaxy.derr(gala
xy.type==1)];
% elliptical = [galaxy.dec(galaxy.type==-1),galaxy.ra(galaxy.type==-
1),galaxy.type(galaxy.type==-1),galaxy.Lfrac(galaxy.type==-1),galaxy.Lfracerr(galaxy.type==-
1),galaxy.d(galaxy.type==-1),galaxy.derr(galaxy.type==-1)];

%This is just a test to see the horizon distance of all the detectors
% det = 'LHO';
% r = horizdist(det);
% hdist(1,1) = r;
% det = 'LLO';
% r = horizdist(det);
% hdist(1,2) = r;
% det = 'Virgo';
% r = horizdist(det);
% hdist(1,3) = r;
% det = 'aLIGO';
% r = horizdist(det);
% hdist(1,4) = r;
% det = 'aVirgo';
% r = horizdist(det);
% hdist(1,5) = r;
% det = 'ETB';
% r = horizdist(det);
% hdist(1,6) = r;
% det = 'KAGRA';
% r = horizdist(det);
% hdist(1,7) = r;
% varargin{1} = 'LLO';
% hdist(1,8) = r;

%This defines the amount of galaxies you want to pick from the catalogue
newamt = 1000;

%Here and below is old code no longer needed
%restrict amount of sources by distance
% ind = find(galaxy.d<=max(r));
% amt = length(ind);
% newind = round(1 + (amt-1).*rand(newamt,1));

%import ra, dec, and distance for distance limit desired
% for num = 1:length(newind)
%     theta(num,:) = galaxy.ra(ind(newind((num))))/24*2*pi;
%     phi(num,:) = galaxy.dec(ind(newind((num))))*pi/180;
%     dist(num,:) = galaxy.d(ind(newind((num))));
```

```
%     disterr(num,:) = galaxy.derr(ind(newind((num))));
% end

%random numbers between 1 and length of elliptica/spiral for indices
% newinde = round(1 + (length(elliptical)-1).*rand(newamt,1));
% newinds = round(1 + (length(spiral)-1).*rand(newamt,1));

%This is preparation to chose brighter galaxies over dimmer ones
%create new column with lfrac divided by sum of lfrac
% new = sum(elliptical(:,4));
% elliptical(:,8) = elliptical(:,4)./new;
% new2 = sum(spiral(:,4));
% spiral(:,8) = spiral(:,4)./new2;

%sort in descending order
% [elliptical(:,8),elliptical(:,9)] = sort(elliptical(:,8),'descend');
% [spiral(:,8),spiral(:,9)] = sort(spiral(:,8),'descend');

%find cumsum of result
% elliptical(:,10) = cumsum(elliptical(:,8));
% spiral(:,10) = cumsum(spiral(:,8));

%This is the percentage of elliptical galaxies desired
pctellip = 60;
%random selection weighted by distance and shape
for i = 1:newamt
    choice = rand;
    if choice < (pctellip/100)
        [c index] = min(abs(elliptical(:,10)-choice));
        newind = elliptical(index,9);
        theta(i,:) = elliptical(newind,2)/24*2*pi;
        theta2(i,:) = elliptical(newind,2)*360/24;
        phi2(i,:) = elliptical(newind,1);
        phi(i,:) = elliptical(newind,1)*pi/180;
        dist(i,:) = elliptical(newind,6);
        type(i,:) = 'e';
    elseif choice > (pctellip/100)
        [c index] = min(abs(spiral(:,10)-choice));
        newind = spiral(index,9);
        theta(i,:) = spiral(newind,2)/24*2*pi;
        theta2(i,:) = spiral(newind,2)*360/24;
        phi2(i,:) = spiral(newind,1);
        phi(i,:) = spiral(newind,1)*pi/180;
        dist(i,:) = spiral(newind,6);
        type(i,:) = 's';
    end
end

%Old code no longer needed
%import ra, dec, and distance for distance limit desired
%         for num = 1:length(newind)
%             theta(num,:) = galaxy.ra(newind((num)))/24*2*pi;
%             theta2(num,:) = galaxy.ra(newind((num)));
%             phi(num,:) = galaxy.dec(newind((num)))*pi/180;
%             dist(num,:) = galaxy.d(newind((num)));
%             disterr(num,:) = galaxy.derr(newind((num)));
%         end

% djit = -1 + (1-(-1)).*rand(newamt,1);
% dist = dist + djit;

%angles for compute antenna response
computetheta = phi*(-1)+(pi/2);
computephi = theta-pi;
```

```
%generate random sky position (theta and phi)
% theta = galaxy.dec(1:10000)*pi/180*(-1)+(pi/2);
% phi = galaxy.ra(1:10000)/24*2*pi-pi;
% skypos = [phi,theta];

%generate random polarization angle in degrees
pol = (pi/2)*rand(newamt,1);

%make parameter matrix
% param = [skypos,pol];

%add distance to earth in units of mpc to parameter matrix
% param(:,4) = galaxy.d(1:10000);

%add inclination angle in degrees to parameter matrix
inc = acos(2*rand(newamt,1)-1);

%generate phic
phic = 2*pi*rand(newamt,1);

% generate tc
lambda = 12; %avg coalescence in months
tc = poissrnd(lambda,newamt,1);
%avg number of coalescences per year
avg = sum(tc) /(newamt*12);
rate = avg/((4/3)*pi*max(dist)^3);

%define frequency
f0 = 70;

%define lower and upper boundaries of integral
intlow = 10;
intup = 4000;

%define mass one and mass two
lowerlim = 3;
upperlim = 15;
mass1 = (lowerlim + (upperlim-lowerlim).*rand(newamt,1));
mass2 = (lowerlim + (upperlim-lowerlim).*rand(newamt,1));
m1 = mass1.*1.9891e30;
m2 = mass2.*1.9891e30;

%empty arrays for detectors and integrals
det1 = [];
det2 = [];
det3 = [];
det4 = [];
det5 = [];
det6 = [];

q1 = [];
q2 = [];
q3 = [];
q4 = [];
q5 = [];
q6 = [];

%switch for configuration
switch config

    case 'initial'
        detnum = 3;
        det = 'Virgo';
```

```
        %         for z = 1:newamt
        %             r(z,1) = horizdist(det,mass1(z,1),mass2(z,1));
        %         end
        det1 = 'LHO';
        det2 = 'LLO';
        det3 = 'V';
        %ligo
        fun1 = @(x) 1./( (x).^(7/3).* 9.*(1e-46).*( (4.49.*(x./f0)).^(-56) +
0.16.*(x./f0).^(-4.52) + 0.52 + 0.32.*(x./f0).^2 ));

        %virgo
        fun2 = @(x) 1./( (x).^(7/3).* (3.2e-46 .* (7.8.*(x./500)).^-5 + 2.8.*(x./500).^-1 +
0.63 + (x./500).^2));

        %integrate
        q1 = integral(fun1,intlow,intup);
        q2 = integral(fun1,intlow,intup);
        q3 = integral(fun1,intlow,intup);

    case 'enchanced'
        detnum = 3;

    case 'advanced'
        detnum = 3;
        det = 'aVirgo';
        %         for z = 1:newamt
        %             r(z,1) = horizdist(det,mass1(z,1),mass2(z,1));
        %         end
        det1 = 'LHO';
        det2 = 'LLO';
        det3 = 'V';

        %aligo
        fun1 = @(x)1./( (x).^(7/3).*1.35e-50 .* x .* ( 60000.*(x./10).^(-30) +
5.*(x./50).^(-6) ...
            + 1.07.*(x./100).^(-3.25) + 3.7.*(x./200).^(-1.25) ...
            + 0.9.*(x./300).^(-0.08) + 0.85.*(x./1000).^(0.8) ...
            + 0.53.*(x./2000).^(3) ));

        %avirgo
        fun2 = @(x)1./( (x).^(7/3).* ( 1.259e-24 .* (0.07.*exp(-0.142-
1.437.*(log10(x./300))+0.407.*(log10(x./300)).^2)+3.10.*exp(-0.466-1.043.*(log10(x./300))-
0.548.*(log10(x./300)).^2)+0.40.*exp(-0.304+2.896.*(log10(x./300))-
0.293.*(log10(x./300)).^2)+0.09.*exp(1.466+3.722.*(log10(x./300))-
0.984.*(log10(x./300)).^2))).^2);

        q1 = integral(fun1,intlow,intup);
        q2 = integral(fun1,intlow,intup);
        q3 = integral(fun2,intlow,intup);

    case 'advanced/india'
        detnum = 4;
        det = 'aVirgo';
        %         for z = 1:newamt
        %             r(z,1) = horizdist(det,mass1(z,1),mass2(z,1));
        %         end
        det1 = 'LHO';
        det2 = 'LLO';
        det3 = 'V';
        det4 = 'LIO';

        %aligo
        fun1 = @(x)1./( (x).^(7/3).*1.35e-50 .* x .* ( 60000.*(x./10).^(-30) +
5.*(x./50).^(-6) ...
```

```
            + 1.07.*(x./100).^(-3.25) + 3.7.*(x./200).^(-1.25) ...
            + 0.9.*(x./300).^(-0.08) + 0.85.*(x./1000).^(0.8) ...
            + 0.53.*(x./2000).^(3) ));

        %avirgo
        fun2 = @(x)1./( (x).^(7/3).* ( 1.259e-24 .* (0.07.*exp(-0.142-
1.437.*(log10(x./300))+0.407.*(log10(x./300)).^2)+3.10.*exp(-0.466-1.043.*(log10(x./300))-
0.548.*(log10(x./300)).^2)+0.40.*exp(-0.304+2.896.*(log10(x./300))-
0.293.*(log10(x./300)).^2)+0.09.*exp(1.466+3.722.*(log10(x./300))-
0.984.*(log10(x./300)).^2))).^2);

        q1 = integral(fun1,intlow,intup);
        q2 = integral(fun1,intlow,intup);
        q3 = integral(fun2,intlow,intup);
        q4 = integral(fun1,intlow,intup);

    case 'et'
        detnum = 1;
        det = 'ETB';
        %         for z = 1:newamt
        %             r(z,1) = horizdist(det,mass1(z,1),mass2(z,1));
        %         end
        det1 = 'ET1';

        p1 = -4.05049;
        p2 = -0.687263;
        a0 = 185.623;
        b0 = 232.565;
        b1 = 31.1777;
        b2 = -64.7178;
        b3 = 52.242;
        b4 = -42.1643;
        b5 = 10.1708;
        b6 = 11.5317;
        c1 = 13.5793;
        c2 = -36.4586;
        c3 = 18.5625;
        c4 = 27.4297;
        S0 = 1.449936*10^-52;

        fun = @(x)1./( (x).^(7/3).*S0.*((x./200).^p1 + a0.*(x./200).^p2 + b0 .* ...
            ( 1. + b1.*(x./200) + b2.*(x./200).^2 + b3.*(x./200).^3 + b4.*(x./200).^4 +
b5.*(x./200).^5 + b6.*(x./200).^6 ) ...
            ./ (1. + c1.*(x./200) + c2.*(x./200).^2 + c3.*(x./200).^3 + c4.*(x./200).^4) ));

        q1 = integral(fun,intlow,intup);

    case 'et and advanced'
        detnum = 5;
        det = 'aVirgo';
        %         for z = 1:newamt
        %             r(z,1) = horizdist(det,mass1(z,1),mass2(z,1));
        %         end
        det1 = 'LHO';
        det2 = 'LLO';
        det3 = 'V';
        det4 = 'LIO';
        det5 = 'ET1';

        %aligo
        fun1 = @(x)1./( (x).^(7/3).*1.35e-50 .* x .* ( 60000.*(x./10).^(-30) +
5.*(x./50).^(-6) ...
            + 1.07.*(x./100).^(-3.25) + 3.7.*(x./200).^(-1.25) ...
            + 0.9.*(x./300).^(-0.08) + 0.85.*(x./1000).^(0.8) ...
```

```
            + 0.53.*(x./2000).^(3) ));

        %avirgo
        fun2 = @(x)1./( (x).^(7/3).* ( 1.259e-24 .* (0.07.*exp(-0.142-
1.437.*(log10(x./300))+0.407.*(log10(x./300)).^2)+3.10.*exp(-0.466-1.043.*(log10(x./300))-
0.548.*(log10(x./300)).^2)+0.40.*exp(-0.304+2.896.*(log10(x./300))-
0.293.*(log10(x./300)).^2)+0.09.*exp(1.466+3.722.*(log10(x./300))-
0.984.*(log10(x./300)).^2))).^2);

        %et
        p1 = -4.05049;
        p2 = -0.687263;
        a0 = 185.623;
        b0 = 232.565;
        b1 = 31.1777;
        b2 = -64.7178;
        b3 = 52.242;
        b4 = -42.1643;
        b5 = 10.1708;
        b6 = 11.5317;
        c1 = 13.5793;
        c2 = -36.4586;
        c3 = 18.5625;
        c4 = 27.4297;
        S0 = 1.449936*10^-52;

        fun3 = @(x)1./( (x).^(7/3).*S0.*((x./200).^p1 + a0.*(x./200).^p2 + b0 .* ...
            ( 1. + b1.*(x./200) + b2.*(x./200).^2 + b3.*(x./200).^3 + b4.*(x./200).^4 +
b5.*(x./200).^5 + b6.*(x./200).^6 ) ...
            ./ (1. + c1.*(x./200) + c2.*(x./200).^2 + c3.*(x./200).^3 + c4.*(x./200).^4) ));

        q1 = integral(fun1,intlow,intup);
        q2 = integral(fun1,intlow,intup);
        q3 = integral(fun2,intlow,intup);
        q4 = integral(fun1,intlow,intup);
        q5 = integral(fun3,intlow,intup);

    case 'et/advanced/kagra'
        detnum = 6;
        det = 'aVirgo';
        %         for z = 1:newamt
        %             r(z,1) = horizdist(det,mass1(z,1),mass2(z,1));
        %         end
        det1 = 'LHO';
        det2 = 'LLO';
        det3 = 'V';
        det4 = 'LIO';
        det5 = 'K';
        det6 = 'ET1';

        %aligo
        fun1 = @(x)1./( (x).^(7/3).*1.35e-50 .* x .* ( 60000.*(x./10).^(-30) +
5.*(x./50).^(-6) ...
            + 1.07.*(x./100).^(-3.25) + 3.7.*(x./200).^(-1.25) ...
            + 0.9.*(x./300).^(-0.08) + 0.85.*(x./1000).^(0.8) ...
            + 0.53.*(x./2000).^(3) ));

        %avirgo
        fun2 = @(x)1./( (x).^(7/3).* ( 1.259e-24 .* (0.07.*exp(-0.142-
1.437.*(log10(x./300))+0.407.*(log10(x./300)).^2)+3.10.*exp(-0.466-1.043.*(log10(x./300))-
0.548.*(log10(x./300)).^2)+0.40.*exp(-0.304+2.896.*(log10(x./300))-
0.293.*(log10(x./300)).^2)+0.09.*exp(1.466+3.722.*(log10(x./300))-
0.984.*(log10(x./300)).^2))).^2);
```

```
        %kagra
        fun3 = @(x)1./( (x).^(7/3).* ( 6.499e-25 .* (0.72e-9.*exp(-1.43-
9.88.*(log10(x./100))-0.23.*(log10(x./100)).^2)+1.17.*exp(0.14-3.10.*(log10(x./100))-
0.26.*(log10(x./100)).^2)+1.70.*exp(0.14+1.09.*(log10(x./100))-
0.013.*(log10(x./100)).^2)+1.25.*exp(0.071+2.83.*(log10(x./100))-
4.91.*(log10(x./100)).^2))).^2);

        %et
        p1 = -4.05049;
        p2 = -0.687263;
        a0 = 185.623;
        b0 = 232.565;
        b1 = 31.1777;
        b2 = -64.7178;
        b3 = 52.242;
        b4 = -42.1643;
        b5 = 10.1708;
        b6 = 11.5317;
        c1 = 13.5793;
        c2 = -36.4586;
        c3 = 18.5625;
        c4 = 27.4297;
        S0 = 1.449936*10^-52;

        fun4 = @(x)1./( (x).^(7/3).*S0.*((x./200).^p1 + a0.*(x./200).^p2 + b0 .* ...
            ( 1. + b1.*(x./200) + b2.*(x./200).^2 + b3.*(x./200).^3 + b4.*(x./200).^4 +
b5.*(x./200).^5 + b6.*(x./200).^6 ) ...
            ./ (1. + c1.*(x./200) + c2.*(x./200).^2 + c3.*(x./200).^3 + c4.*(x./200).^4) ));

        q1 = integral(fun1,intlow,intup);
        q2 = integral(fun1,intlow,intup);
        q3 = integral(fun2,intlow,intup);
        q4 = integral(fun1,intlow,intup);
        q5 = integral(fun3,intlow,intup);
        q6 = integral(fun4,intlow,intup);
end

%add fplus and fcross
[Fp,Fc] = antenna(detnum,det1,det2,det3,det4,det5,det6,pol,computephi,computetheta,newamt);

%solve for chirp mass
chirp = ((m1.*m2).^(3/5))./((m1+m2).^(1/5));

%find SNR for all detectors
[signoise] = SNR(detnum,chirp,inc,dist,Fp,Fc,newamt,q1,q2,q3,q4,q5,q6);

%calculate last column of trigger matrix
endtrig = 10+detnum;

%find SNR >= 5 in two detectors
if detnum > 1
    [search,junk] = find(signoise>=5);
    search = unique(search);
    for k = 1:length(search)
        search2 = find(signoise(search(k),:)>=5);
        if length(search2)>=2
            triggers(k,1) = phi2(search(k),1);
            triggers(k,2) = theta2(search(k),1);
            triggers(k,3) = dist(search(k),1);
            triggers(k,4) = chirp(search(k),1);
            triggers(k,5) = mass1(search(k),1);
            triggers(k,6) = mass2(search(k),1);
            triggers(k,7) = inc(search(k),1);
            triggers(k,8) = pol(search(k),1);
```

```
            triggers(k,9) = Fp(search(k),1);
            triggers(k,10) = Fc(search(k),1);
            triggers(k,11:endtrig) = signoise(search(k),:);
        end
    end

    %clear out zeros
    [rows,col]=find(triggers == 0);
    u = unique(rows);
    triggers(u,:) = [];

    %create final matrix
    final = [];

    [row,col] = size(triggers);
    %find SNR quadrature sum >= 12
    for a = 1:row
        if sqrt(triggers(a,11)^2 + triggers(a,12)^2 + triggers(a,13)^2+ triggers(a,14)^2+
triggers(a,15)^2+ triggers(a,16)^2) >= 12
            final(a,1:col) = triggers(a,1:col);
        end
    end

    %clear out zeros
    [rows,col]=find(final == 0);
    u = unique(rows);
    final(u,:) = [];

    %find new parameters of found signals for all detectors
    %create ini file
    %use 'jitterval2' for errors decreased by factor of 10
    fileName = 'jitterval';
    %     inifile(fileName,'new');

    %read values from ini file
    [keys,sections,subsections] = inifile(fileName,'readall');

    %find system snr
    [r,c] = size(final);
    totalsnr = sqrt( (final(:,11).^2) + (final(:,12).^2) + (final(:,13).^2) +
(final(:,14).^2) + (final(:,15).^2)+(final(:,16).^2) );

    %Here we add the noise to all parameters
    %total mass
    %get current values
    totmass = final(:,5) + final(:,6);
    %get factor that snr is divided by
    fact = str2num(keys{1,4});
    %calculate array of jitter values using randn for gaussian
    totmassjit =  -(fact./totalsnr) + ((fact./totalsnr)-(-(fact./totalsnr))).*randn(r,1);
    %jitter old values
    totmass = totmass + totmassjit;

    %mass ratio
    massrat = final(:,5)./final(:,6);
    fact = str2num(keys{2,4});
    ratmassjit =  -(fact./totalsnr) + ((fact./totalsnr)-(-(fact./totalsnr))).*randn(r,1);
    massrat = massrat + ratmassjit;

    %right ascension
    ra = final(:,2);
    fact = str2num(keys{3,4});
    rajit =  -(fact./totalsnr) + ((fact./totalsnr)-(-(fact./totalsnr))).*randn(r,1);
    ra = (ra + rajit)*24/360;
```

```
    %declination
    dec = final(:,1);
    fact = str2num(keys{3,4});
    decjit =  -(fact./totalsnr) + ((fact./totalsnr)-(-(fact./totalsnr))).*randn(r,1);
    dec = dec + decjit;

    %distance
    newdist = final(:,3);
    fact = str2num(keys{4,4});
    distjit =  -(fact./totalsnr) + ((fact./totalsnr)-(-(fact./totalsnr))).*randn(r,1);
    newdist = newdist + distjit;

    %inclination
    newinc = final(:,7)*180/pi;
    fact = str2num(keys{5,4});
    incjit =  -(fact./totalsnr) + ((fact./totalsnr)-(-(fact./totalsnr))).*randn(r,1);
    newinc = newinc + incjit;

    %polarization
    newpol = final(:,8)*180/pi;
    fact = str2num(keys{6,4});
    poljit =  -(fact./totalsnr) + ((fact./totalsnr)-(-(fact./totalsnr))).*randn(r,1);
    newpol = newpol + poljit;

    %snr
    rnd = -1 + (1-(-1)).*rand(r,1);
    %create random set of numbers that are either -1 or 1
    find(rnd<0);
    rnd(ans) = -1;
    find(rnd>0);
    rnd(ans) = 1;
    %use chi square function of snr at 2 dof to get jitter
    %values
    snrjit = (chi2pdf(totalsnr,2)).*rnd;
    %jitter snr values
    totalsnr = totalsnr + snrjit;

    %error box

    %find indices of distance, ra, and dec that are within correct ranges
    for i = 1:r
        distind{:,i} = find(galaxy.d < ((newdist(i))+(abs(distjit(i)))) & galaxy.d >
((newdist(i))-(abs(distjit(i)))));
        raind{:,i} = find(galaxy.ra < ((ra(i))+(abs(rajit(i)))) & galaxy.ra > ((ra(i))-
(abs(rajit(i)))));
        decind{:,i} = find(galaxy.dec < ((dec(i))+(abs(decjit(i)))) & galaxy.dec >
((dec(i))-(abs(decjit(i)))));
    end

    %find matching indices of ra and dec, then also distance
    for j = 1:r
        comang{1,j} = intersect(raind{j},decind{j});
        galind{1,j} = intersect(comang{1,j},distind{j});
    end

    %create matrix of found galaxies
    gal = [];
    for i = 1:r
        newgal = galind{i};
        newgal(:,2) = newdist(i,1);
        gal = [gal;newgal];

    end
```

```
    gallength = length(gal);

    %first col is galaxy index, second col is jittered distance of
    %corresponding signal, third col is actual distance, fourth col is
    %redshift, fifth col is old hubble value, sixth col is new (better) hubble value
    gal(:,3) = galaxy.d(gal(:,1));
    gal(:,4) = galred(gal(:,1));
    gal(:,5) = hub(gal(:,1));
    gal(:,6) = hubble(gal,gallength);

    %plot histogram of hubble values
    hist(gal(:,6));

    %calculate effective distance
    effectdist = newdist./(((1/4).*(final(:,9).^2).*(1+ (cos(newinc).*cos(newinc))).^2 +
((final(:,10)).^2).*(cos(newinc).*cos(newinc))).^(1/2));

    %plot all signals, signals that pass first test, and signals that pass
    %second
    %     scatter(computephi(:,1),computetheta(:,1),'g','.')
    %     hold
    %     scatter(triggers(:,1),triggers(:,2),'r','.')
    %     scatter(final(:,1),final(:,2),'b','.')

    %plot ra and dec, vary color by distance
    %     scatter(final(:,1),final(:,2), 50, final(:,3),'.');
    %     figure(2);
    %     scatter(final(:,1),final(:,2), 50, final(:,4),'.');

    %plot 6 patches of sky position for different bins of distances
    %      for x = 125:125:750
    %         found = [];
    %         found = find(dist < x & dist > x-125);
    %         figure(x/125);
    %         plot(theta(found),phi(found),'o');
    %     end

    %     data = [];
    %     %plot number of signals versus distance in bins
    %     for x = min(dist):50:max(dist)
    %         found = [];
    %         found = find(dist < x & dist > x-50);
    %         datanew = [log(x),log(length(found))];
    %         data = [data;datanew];
    %     end
    %
    %     TF = isinf(data);
    %     [r,c] = find(TF ==1);
    %     data(r,:) = [];
    %
    %     plot(data(:,1),data(:,2),'o');
    %     hold on
    %     lsline
    %     title('Log Plot of Number Density');
    %     xlabel('Distance');
    %     ylabel('Number of Galaxies');
    %     p = polyfit(data(:,1),data(:,2),1);
    %     yfit = polyval(p,data(:,1));
    %     yresid = data(:,2) - yfit;
    %     SSresid = sum(yresid.^2);
    %     SStotal = (length(data(:,2))-1) * var(data(:,2));
    %     rsq = 1 - SSresid/SStotal;
    %     h = chi2gof(data(:,1));
```

```
    %     e = std(data(:,1))*ones(size(data(:,1)));
    %     eb = errorbar(data(:,1),data(:,2),e);
    %     set(eb(1),'Linestyle','none','Marker','s','MarkerEdgeColor','k');
    %
    %     %plot sky position with different colors for different distance bins
    %     data2 = [];
    %     figure(2);
    %     for x = 1:6
    %         found = [];
    %         found = find(dist < x*125 & dist > x*125-125);
    %         if x ==1
    %             scatter(theta2(found),sin(phi(found)),50,'r','o');
    %             title('Sky Positions of Various Sets of Distances');
    %             xlabel('Right Ascension (hrs)');
    %             ylabel('Sine of Declination (rad)');
    %             legend('Distance 0-125');
    %             hold on
    %         elseif x ==2
    %             scatter(theta2(found),sin(phi(found)),50,'g','o');
    %             legend('Distance 126-250');
    %         elseif x ==3
    %             scatter(theta2(found),sin(phi(found)),50,'b','o');
    %             legend('Distance 251-375');
    %         elseif x ==4
    %             scatter(theta2(found),sin(phi(found)),50,'c','o');
    %             legend('Distance 376-500');
    %         elseif x ==5
    %             scatter(theta2(found),sin(phi(found)),50,'m','o');
    %             legend('Distance 501-625');
    %         elseif x ==6
    %             scatter(theta2(found),sin(phi(found)),50,'k','o');
    %             legend('Distance 0-125','Distance 126-250','Distance 251-375','Distance
376-500','Distance 501-625','Distance 626-750');
    %         end
    %
    %     end
    %
    %
end



%optional test for timing error
% %detector list
% detectors = {'V';'L';'H'};
%
% %compute delay
% delay = computeTimeShifts(detectors,param(:,1:2));
%
% %bandwidths
% bandwidthV = 188.8889;
% bandwidthL = 94.9164;
%
% %find timing error for virgo
% SNR=8;
% dtV=(2*pi*bandwidthV*SNR)^-1;
% dtV = dtV*1.2; %-- adds a 20% timing error
% %timingErrorV = dtV * ones(size(detectors));
% % ytime = (2*rand(3,1)-1).*0.2;
% % timingErrorV = dtV+dtV.*ytime;
%
% %find timing error for ligo
% dtL=(2*pi*bandwidthL*SNR)^-1;
% dtL = dtL*1.2; %-- adds a 20% timing error
```

```
% % timingErrorL = dtL * ones(size(detectors));
% % timingErrorL = timingErrorL+timingErrorL.*ytime;
%
% %find final timing errors for all detectors
% alltimingErrorV = repmat(dtV,10000,1)./param(:,14);
% alltimingErrorL = repmat(dtL,10000,1)./param(:,13);
% alltimingErrorH = repmat(dtL,10000,1)./param(:,12);
%
% %combine timing errors into one matrix
% timingerrors = [alltimingErrorV,alltimingErrorL,alltimingErrorH]; %VLH
%
% %find arrival times
% arrivaltimes = delay + randn(10000,3) .* timingerrors; %VLH
%
% %test to see if times match
% pass = find(timecoincidence(arrivaltimes,detectors,timingerrors,3));
```

Appendix B – Antenna Pattern Function

```
function [Fp,Fc] =
antenna(detnum,det1,det2,det3,det4,det5,det6,pol,computephi,computetheta,newamt)

%this function calculates the antenna patterns for up to six detectors
%by Teresa Symons 2013

%detector 1
if detnum == 1 || detnum == 2|| detnum == 3|| detnum == 4|| detnum == 5|| detnum == 6
    for i = 1:newamt
        [Fp1(i,:), Fc1(i,:), Fb] =
ComputeAntennaResponse(computephi(i,1),computetheta(i,1),pol(i,1),det1);
    end
    Fp = Fp1;
    Fc = Fc1;
end

%detector 2
if detnum ==2|| detnum == 3|| detnum == 4|| detnum == 5|| detnum == 6
    for i = 1:newamt
        [Fp2(i,:), Fc2(i,:), Fb] =
ComputeAntennaResponse(computephi(i,1),computetheta(i,1),pol(i,1),det2);
    end
    Fp = [Fp1,Fp2];
    Fc = [Fc1,Fc2];
end

%detector 3
if detnum == 3|| detnum == 4|| detnum == 5|| detnum == 6
    for i = 1:newamt
        [Fp3(i,:), Fc3(i,:), Fb] =
ComputeAntennaResponse(computephi(i,1),computetheta(i,1),pol(i,1),det3);
    end
    Fp = [Fp1,Fp2,Fp3];
    Fc = [Fc1,Fc2,Fc3];
end

%detector 4
if detnum == 4|| detnum == 5|| detnum == 6
    for i = 1:newamt
        [Fp4(i,:), Fc4(i,:), Fb] =
ComputeAntennaResponse(computephi(i,1),computetheta(i,1),pol(i,1),det4);
    end
    Fp = [Fp1,Fp2,Fp3,Fp4];
    Fc = [Fc1,Fc2,Fc3,Fc4];
```

```
end

%detector 5
if detnum == 5|| detnum == 6
    for i = 1:newamt
        [Fp5(i,:), Fc5(i,:), Fb] =
ComputeAntennaResponse(computephi(i,1),computetheta(i,1),pol(i,1),det5);
    end
    Fp = [Fp1,Fp2,Fp3,Fp4,Fp5];
    Fc = [Fc1,Fc2,Fc3,Fc4,Fc5];
end

%detector 6
if detnum == 6
    for i = 1:newamt
        [Fp6(i,:), Fc6(i,:), Fb] =
ComputeAntennaResponse(computephi(i,1),computetheta(i,1),pol(i,1),det6);
    end
    Fp = [Fp1,Fp2,Fp3,Fp4,Fp5,Fp6];
    Fc = [Fc1,Fc2,Fc3,Fc4,Fc5,Fc6];
end

end
```

Appendix C – SNR Function

```
function [signoise] = SNR(detnum,chirp,inc,dist,Fp,Fc,newamt,q1,q2,q3,q4,q5,q6)

%this function calculates the SNR for up to six detectors
%by Teresa Symons 2013

%detector 1
if detnum == 1 || detnum == 2|| detnum == 3|| detnum == 4|| detnum == 5|| detnum == 6
    for j = 1:newamt
        signoise(j,1) = (((6.67384e-
11^(5/6))/(299792458^(3/2)))*((5/(24*pi^(4/3)))^(1/2))*((chirp(j,1)^(5/6))/(dist(j,1)*100000
0*3.08567758e16))*(((Fp(j,1)^2)*((1+(cos(inc(j,1))*cos(inc(j,1))))^2)+4*(((Fc(j,1))^2)*(cos
(inc(j,1))*cos(inc(j,1))))^(1/2)*((q1)^(1/2)));
    end
%     sjit =  -1 + (1-(-1)).*randn(newamt,1);
%     signoise(:,1) = signoise(:,1) + sjit;
end

%detector 2
if detnum ==2|| detnum == 3|| detnum == 4|| detnum == 5|| detnum == 6
    for j = 1:newamt
        signoise(j,2) = (((6.67384e-
11^(5/6))/(299792458^(3/2)))*((5/(24*pi^(4/3)))^(1/2))*((chirp(j,1)^(5/6))/(dist(j,1)*100000
0*3.08567758e16))*(((Fp(j,2)^2)*((1+(cos(inc(j,1))*cos(inc(j,1))))^2)+4*(((Fc(j,2))^2)*(cos
(inc(j,1))*cos(inc(j,1))))^(1/2)*((q2)^(1/2)));
    end
%     sjit =  -1 + (1-(-1)).*randn(newamt,1);
%     signoise(:,2) = signoise(:,2) + sjit;
end

%detector 3
if detnum == 3|| detnum == 4|| detnum == 5|| detnum == 6
    for j = 1:newamt
        signoise(j,3) = (((6.67384e-
11^(5/6))/(299792458^(3/2)))*((5/(24*pi^(4/3)))^(1/2))*((chirp(j,1)^(5/6))/(dist(j,1)*100000
0*3.08567758e16))*(((Fp(j,3)^2)*((1+(cos(inc(j,1))*cos(inc(j,1))))^2)+4*(((Fc(j,3))^2)*(cos
(inc(j,1))*cos(inc(j,1))))^(1/2)*((q3)^(1/2)));
    end
%     sjit =  -1 + (1-(-1)).*randn(newamt,1);
```

```
%     signoise(:,3) = signoise(:,3) + sjit;
end

%detector 4
if detnum == 4|| detnum == 5|| detnum == 6
    for j = 1:newamt
        signoise(j,4) = (((6.67384e-
11^(5/6))/(299792458^(3/2)))*((5/(24*pi^(4/3)))^(1/2))*((chirp(j,1)^(5/6))/(dist(j,1)*100000
0*3.08567758e16))*(((Fp(j,4)^2)*((1+(cos(inc(j,1))*cos(inc(j,1)))))^2)+4*(((Fc(j,4))^2)*(cos
(inc(j,1))*cos(inc(j,1)))))^(1/2)*((q4)^(1/2)));
    end
%     sjit =  -1 + (1-(-1)).*randn(newamt,1);
%     signoise(:,4) = signoise(:,4) + sjit;
end

%detector 5
if detnum == 5|| detnum == 6
    for j = 1:newamt
        signoise(j,5) = (((6.67384e-
11^(5/6))/(299792458^(3/2)))*((5/(24*pi^(4/3)))^(1/2))*((chirp(j,1)^(5/6))/(dist(j,1)*100000
0*3.08567758e16))*(((Fp(j,5)^2)*((1+(cos(inc(j,1))*cos(inc(j,1)))))^2)+4*(((Fc(j,5))^2)*(cos
(inc(j,1))*cos(inc(j,1)))))^(1/2)*((q5)^(1/2)));
    end
%     sjit =  -1 + (1-(-1)).*randn(newamt,1);
%     signoise(:,5) = signoise(:,5) + sjit;
end

%detector 6
if detnum == 6
    for j = 1:newamt
        signoise(j,6) = (((6.67384e-
11^(5/6))/(299792458^(3/2)))*((5/(24*pi^(4/3)))^(1/2))*((chirp(j,1)^(5/6))/(dist(j,1)*100000
0*3.08567758e16))*(((Fp(j,6)^2)*((1+(cos(inc(j,1))*cos(inc(j,1)))))^2)+4*(((Fc(j,6))^2)*(cos
(inc(j,1))*cos(inc(j,1)))))^(1/2)*((q6)^(1/2)));
    end
%     sjit =  -1 + (1-(-1)).*randn(newamt,1);
%     signoise(:,6) = signoise(:,6) + sjit;
end

end
```

Appendix D – Hubble Constant Function

```
function H0 = hubble(gal,gallength);

%This function calculates the Hubble parameter using the redshift and the
%distance
%by Teresa Symons 2013

fun = @(x)1./((0.24*(1+x).^3 + 0.76).^(1/2));
H0 = [];
%for each found galaxy integrate function from 0 to z
for i = 1:gallength
        hubint = integral(fun,0,gal(i,4));
        %hub value is c * 1+z / d*int
        newhub = ((299792.458 * (1+gal(i,4)))/ gal(i,2))*hubint;
        H0 = [H0;newhub];
end
```