Localizing Acoustic Noise Sources Affecting the Sensitivity of LIGO Through Multilateration

Erin Thompson

Clemson University

2019 Louisiana State University REU

Mentors

Guillermo Valdes	Louisiana State University
Gabriela González	Louisiana State University

Contents

1. Abstract

The study of gravitational waves at the Laser Interferometer Gravitational Wave Observatory (LIGO) provides insight into the nature of the universe and events such as binary black hole mergers and neutron star collisions. These waves can alter the distance between the mirrors at LIGO on the scale so small that environmental noises can make the detection of gravitational waves incredibly difficult, if not impossible. Acoustic noise can cause vibrations in the beam tube and test masses, which results in unwanted waveforms that can overshadow a gravitational wave. It is therefore important to locate and mitigate noise sources. In order to determine what portion of the detector was most affected by an acoustic noise, we used multilateration to calculate the sound's origin and proximity to each arm. During this process, we successfully tested our code with thunder, a common source of noise in the Livingston detector. We determined that the method of multilateration and time difference of arrival by cross-correlation can be used to find other nearby noise sources at LIGO.

2. Introduction

Gravitational waves (GWs) are ripples in space-time, the stretching and compressing of space itself [1]. They were proposed by Einstein and detected by LIGO, an interferometer which can detect GWs from black hole mergers, neutron star collisions, and other violent events. However, the distortions measured here on Earth by LIGO that occur because of these waves are so small that extreme sensitivity is required in order to detect them. The change in the distance of LIGO's arms is on the scale of a fraction of a proton or 10^{-19} meters.

Any noise in the gravitational wave signal will impede the search for GWs as it decreases the sensitivity of LIGO. The Detector Characterization (DetChar) group locates and categorizes noise sources and, if possible, removes them from the data [2].

Acoustic noise can cause vibrations in the beam tube and test masses, which can overshadow the GW signal. However, some portions of the detector are affected by acoustic noise more than others and cause greater effects in the GW data. Manually injected acoustic noise is used to understand the behavior of the detector. For an injection, the scientists need to know the origin of the sound. Because thunder shakes the detector, is loud, and occurs frequently in Louisiana, it can be used as an injection. In this project, we focused on locating the origin of noise sources. Eventually, this work can be used to determine which portion of the detector was most affected by acoustic noise.

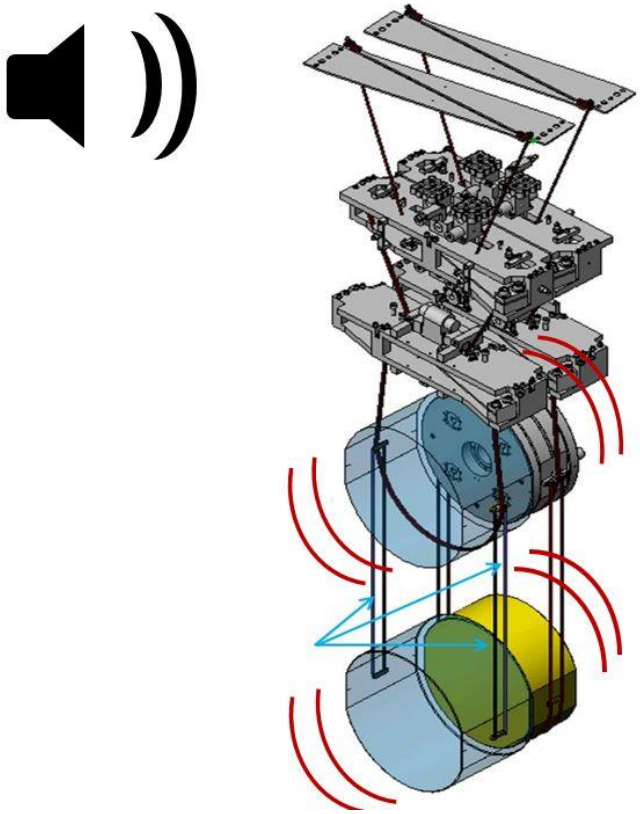

Figure 1: Audio noise causes vibrations in the exterior of the detector which eventually causes motion in the detector's mirrors [3]. Adapted from IGR, University of Glasgow.

On June 14, 2019, operators at LIGO Livingston used microphones to locate a noisy air conditioning (AC) unit. One objective of our project was to see if our method could have been used to locate the unit or other similar objects without actively walking around searching for them.

L1 PEM Link

anamaria.effler@LIGO.ORG - posted 16:45, Friday 14 June 2019 - last comment - 14:10, Tuesday 18 June 2019(46616)

70 Hz in DARM from LG AC unit

JoeH, Anamaria

Yesterday we slowly moved a microphone around the DC supply room trying to track down the intermittent 70 Hz noise previously mentioned. We ended up outside, and finally found that it comes from the white LG AC unit shown in the photo. Moving the mic a meter to the left or right of it reduces the peak in its spectrum. The unit has a compressor inside, best guess it's what's turning on and off. Also, the vibration doesn't have to transmit through the floor but through the pipes which are hard attached to the floor and to the walls inside the building. Hopefully we can figure out a way to get rid of it, it's worth a couple Mpc at least.

Figure 2: Record in aLIGO LLO Logbook records on 14 June, 2019, from Ana Maria about the AC unit that was causing noise around 70 Hz in DARM.

Figure 3: Image of the white LG AC unit referenced in Figure 2.

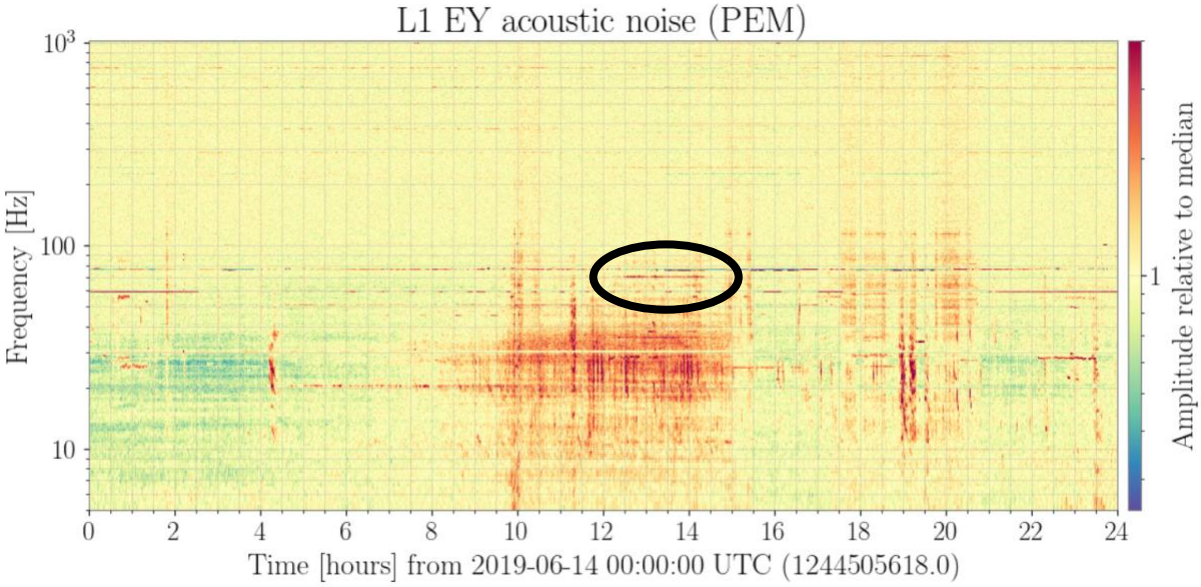

Figure 4: Acoustic noise plot from LLO Summary Pages for PEM sensors for the Y-end station microphone. The area around 70 Hz where the AC unit was likely producing noise is marked.

To calculate the position of the acoustic noise, I implemented multilateration which uses time difference of arrival (TDOA) calculated through the cross-correlation of three microphone time series. The TDOA is not the time of arrival. It is how long it took before the noise was recorded in a microphone compared to the reference mic. The TDOA for the reference mic will always be zero seconds. The equations of position which provide the x and y coordinates for the noise origin were

$$d_i = v * \tau_{0i} = r_i - r_0$$

$$r_i^2 = (x - x_i)^2 + (y - y_i)^2 = (v * \tau_{0i} + r_0)^2$$

In these equations, d_i is the distance between a reference microphone and microphone *i*. The velocity of sound in air is *v*, the TDOA between the reference mic and mic *i* is τ_{0i}. The distance between the sound's origin and mic *i* is r_i and r_0 is the same but for the reference mic. The (x, y) coordinates of the microphone are x_i and y_i. Finally, *x* and *y* are the sound's origin. In this coordinate system, the beam splitter is located at (0, 0).

3. Methods

The primary technique we utilized was multilateration based on TDOA. We worked with the Physical and Environmental Monitors (PEM) at LIGO Livingston, specifically the microphones located at the X-end station, Y-end station, and near the beam splitter in the central station. To find the TDOA, we used the cross correlation of the microphone signals. This work was completed using Python computer script.

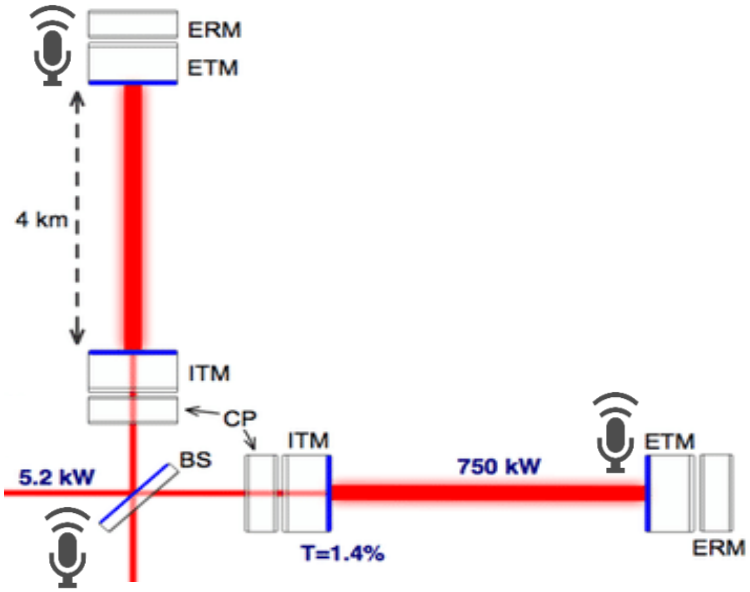

Figure 5: Advanced LIGO schematic with primary microphones marked [3].

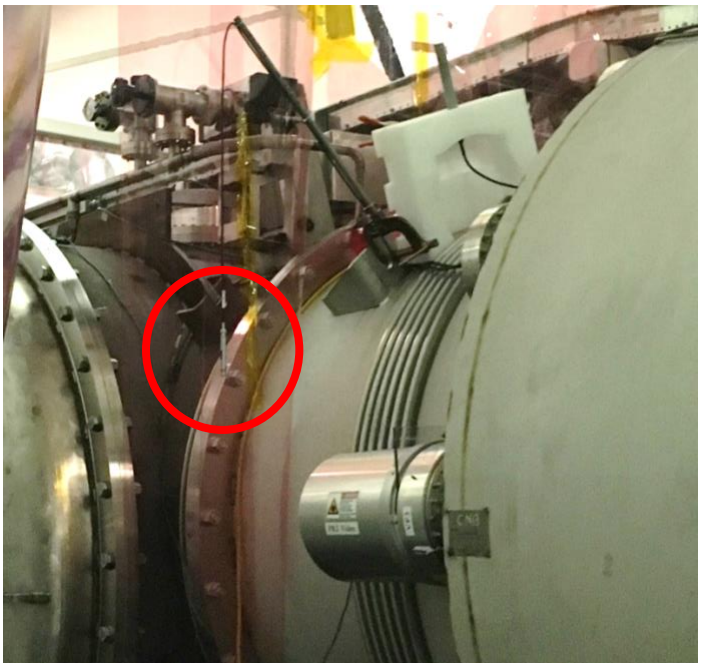

Figure 6: Microphone located near the laser table in the central station. While this microphone was not one I used, it serves an example of what the mics looked like.

For cross-correlation, my code selected the point with the largest correlation between a mic *i* and the reference microphone. The reference microphone was the one in the central station for most tests; it changed for the shorter TDOA test in the Y-end station. The maximum of the correlation was converted into a time offset between the two microphone time series.

When the TDOA was found using cross-correlation, I was left with a series of equations for the sound's origin. I solved this series of equations two ways. The first method used the Moore–Penrose inverse of the matrix containing the equations. This solution was not always as accurate as the second. The second way was to use the fsolve Python Scipy library function, but this one required a guess in order to properly solve the series of equations. The guess was the result provided from the matrix solution.

Testing was done through three main methods. First, I injected a Gaussian pulse into the various microphones at differences in time determined in advance by the distance from a random origin point for the simulated signal to each microphone. I altered the amplitude and frequency of these pulses. In some cases, I injected multiple pulses at the same time to replicate competing noise. I also tested this during calm and moderately noisy times. The accuracy of the code was appraised by the percent difference between the actual and returned coordinates.

After testing with a Gaussian pulse, we validated our code with historical lightning data[1] from O3. I entered in a day and picked one of the lightning strikes near the detector during that time. I

[1] Lightning data from LightningMaps.org

converted the time into a GPS time and then entered that into the code. This test was, unfortunately, more qualitative than quantitative. I had to judge the accuracy of my code based on its approximate position since the lightning database did not include latitude and longitude coordinates. That being said, the website also contains warnings about errors in its positioning and probably should not be considered absolutely perfect.

The third testing method was where I simulated a case with shorter TDOAs. Because the noisy AC unit was in the Y-end station only, I created a case where the three microphones were all located in the Y-end station. At that time, only two mics were present in that station and the position for one was missing. I created the position for that mic and created a third mic that I placed near the other two.

4. Results

4a. Gaussian pulse injections

The Gaussian pulse injection tests demonstrated that the code was able to locate a sound's origin; the accuracy of the position is determined by the TDOA. The pulses were injected during both calm and moderately noisy times to test the robustness of my code. I also injected multiple pulses at once to determine the accuracy of my code when there were competing noises.

From my tests, I determined that if the TDOA is off on the scale of seconds, then the location will be wrong. Still, an error of less than 1 second will place the TDOA in the general location of the noise origin. The accuracy of each test was determined by how close to the actual position the returned value was based on the percent error.

In the beginning, I was running the simulation in 3D space. I was also trying to calculate the z-coordinate of the noise. However, this proved to be unachievable as all of the microphones were placed at about the same z-coordinate. There was not a major difference in their heights and therefore the TDOA from $\hat{z}$ would be miniscule. The error that resulted from injections when I was considering 3D space was extremely large as to not even be worth considering. From those tests, I determined that it was best to remain in 2D space.

During the initial phase of these tests, I also discovered that the method I was previously using to find the location at that time was not forgiving to minor changes in the TDOA. In one case, an offset of 0.1 seconds produced a percent error of several million. Because that was utterly unacceptable, I had to change the method through which I was solving the equations, which is how I arrived at the final code. The percent error is now much, much smaller. In the process of determining which methods to use, I had a code that ran five methods at once and picked the one which returned the point closest to the actual location.

When injecting the pulses, I found that they would not be added unless the time offset was an integer or half integer number. For example, one test required that the pulse be inserted into the X-end station 3.43 seconds after the pulse in the central station mic time series. It refused to be injected at this time, so I had to round the time to 3.5 seconds. Each time, I rounded to the nearest integer or half integer, which caused slight, predictable errors in the TDOA that I used to

evaluate the accuracy of my code. I found that the code was still able to place the origin of the storm near the actual point.

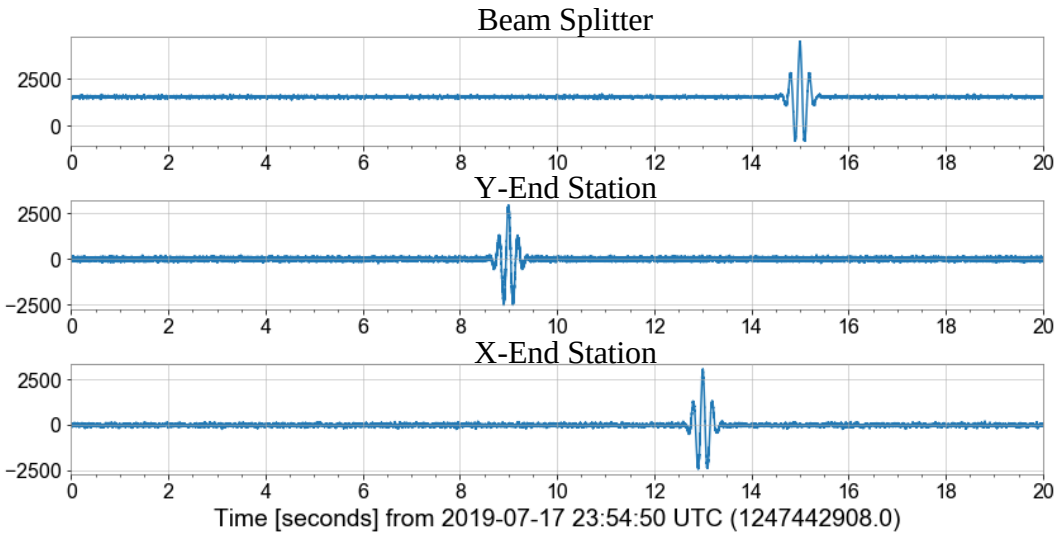

Figure 7: An example of one test with a Gaussian pulse injection of amplitude 2920 counts and frequency of 5 Hz at GPS time 1247442908 with a duration of 20 seconds. The pulses were injected during a calm time. The randomly generated origin of the storm was (2707, 4025).

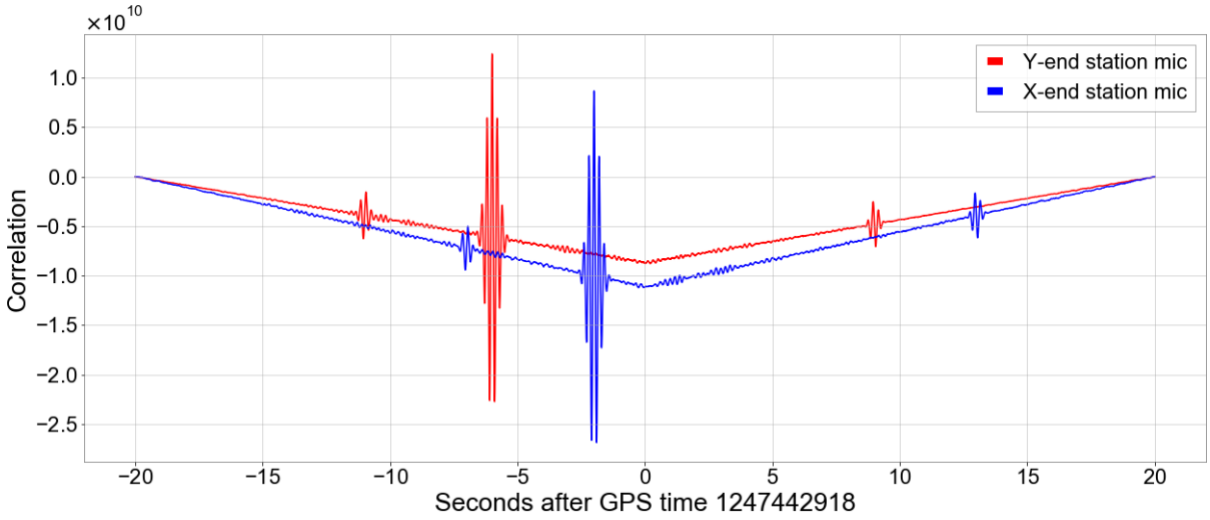

Figure 8: The cross-correlation of the time series in Figure 7. The Y-end station mic and X-end station mic time series were compared to the one from the central station mic near the beam splitter.

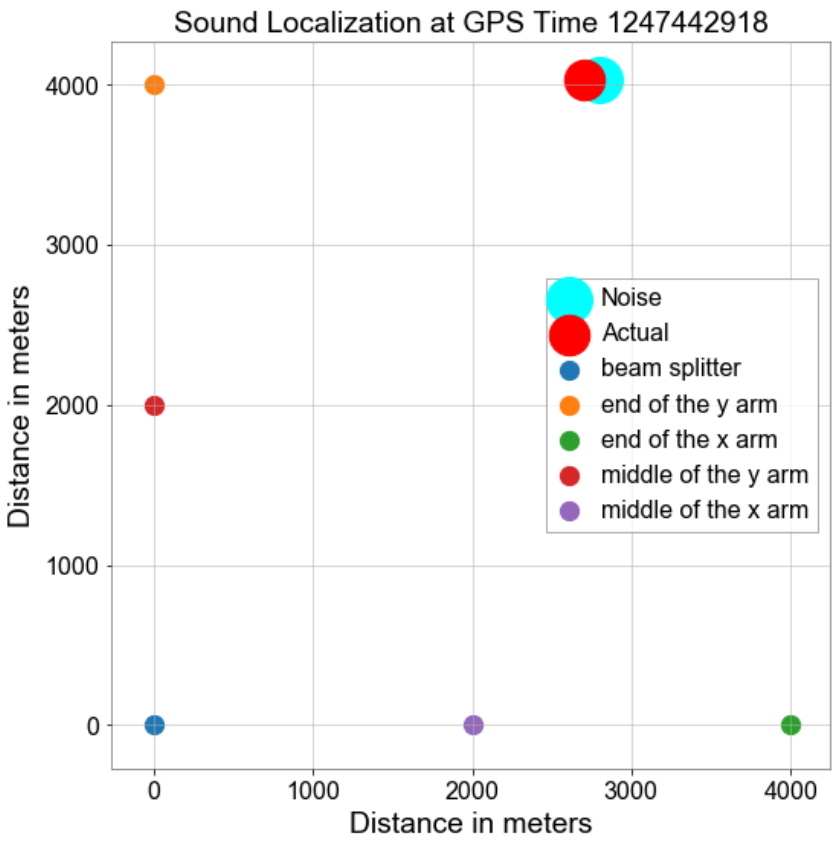

Figure 9: A diagram showing the calculated and actual origin of the noise based on the simulated audio injection in Figure 7 with a small, intentional error in the TDOA.

4b. Historical Lightning Data

As mentioned before, for the actual storm data, we used a database of lightning strikes and positions. When I entered in the time for a lightning strike, I occasionally had to add an offset to make sure that the thunderclap arrived in the detector microphones within the duration I was searching. The lightning might have been from one cloud to another and therefore it would have taken the thunderclap longer to reach the microphones. The thunderclap had to register in a second microphone after being recorded in another within about 12 seconds. The number 12 came from the distance between the end stations.

Ericka Florio and I compiled a list of around 40 lightning strikes to use for testing. We went to the days when a storm occurred and randomly chose lightning strikes to use. When my code failed one of these tests, it was for one of two reasons.

The first reason my code failed was competing noise. When 2 or more lightning strikes occurred at about the same time, they registered in the mics over the same duration. When that happened, the cross-correlation function produced the wrong TDOA and the resultant position was sometimes halfway between the actual origins. For example, the cross-correlation between the

mics near the beam splitter and Y-end station might have found the TDOA from a thunderclap which originated near the Y-end station. At the same time, the cross-correlation between the mics near the beam splitter and X-end station might have found the TDOA for a thunderclap near the X-end station. As a result, the code might produce a location near the Y-mid station.

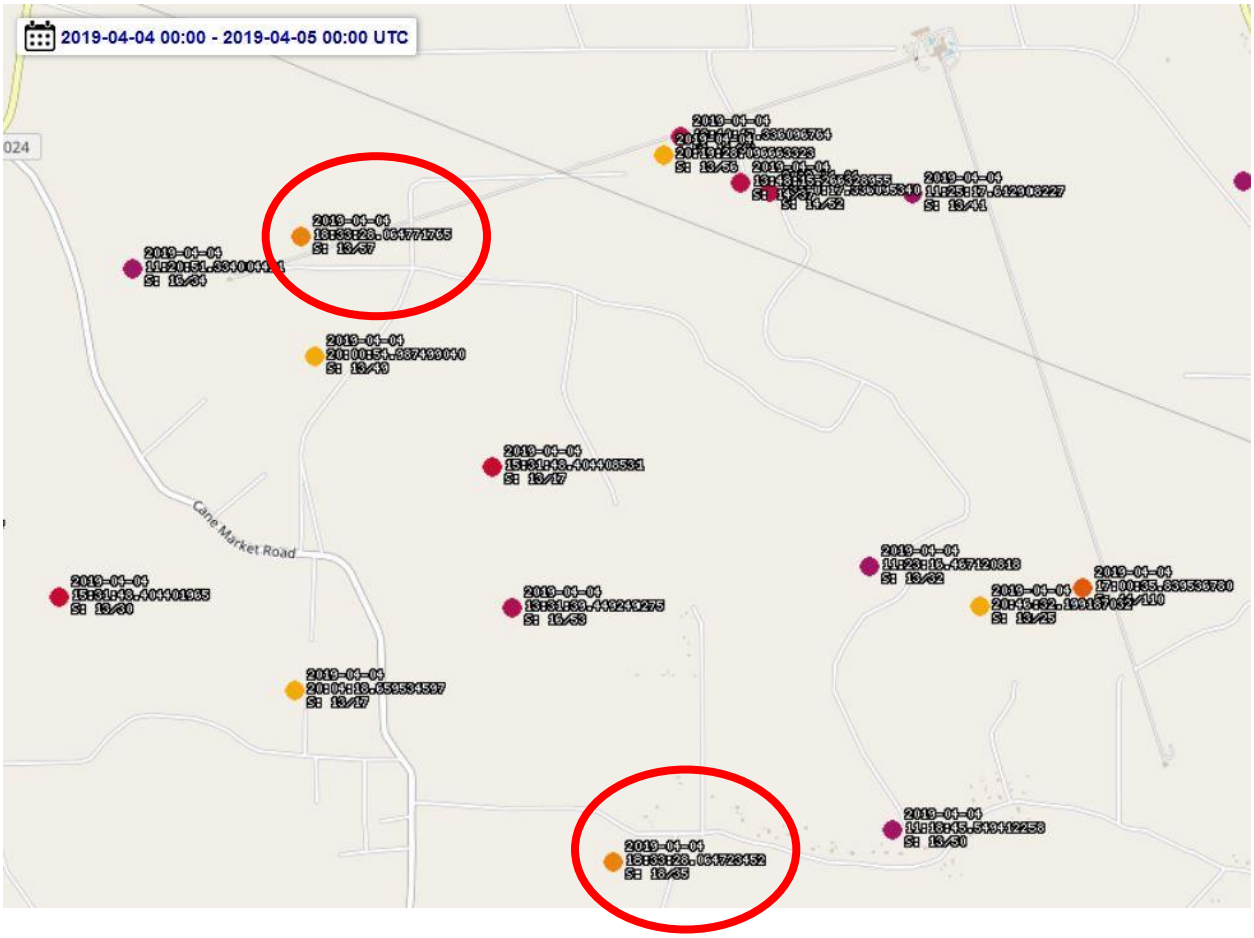

Figure 10: Image from LightningMaps.org of LIGO and the surrounding area on 4 April 2019. One of the tests was for the lightning strike at 18:33 UTC, GPS time 1238438026, near the X-end station. However, another strike occurred at the same time at a y-coordinate level with the Y-end station.

The second reason my code failed was because the thunderclap did not register in all 3 microphones. When this was the case, Ericka's code provided that there was no effect on DARM, the GW signal. For those situations, it would be pointless to locate the storm anyway since it was inconsequential.

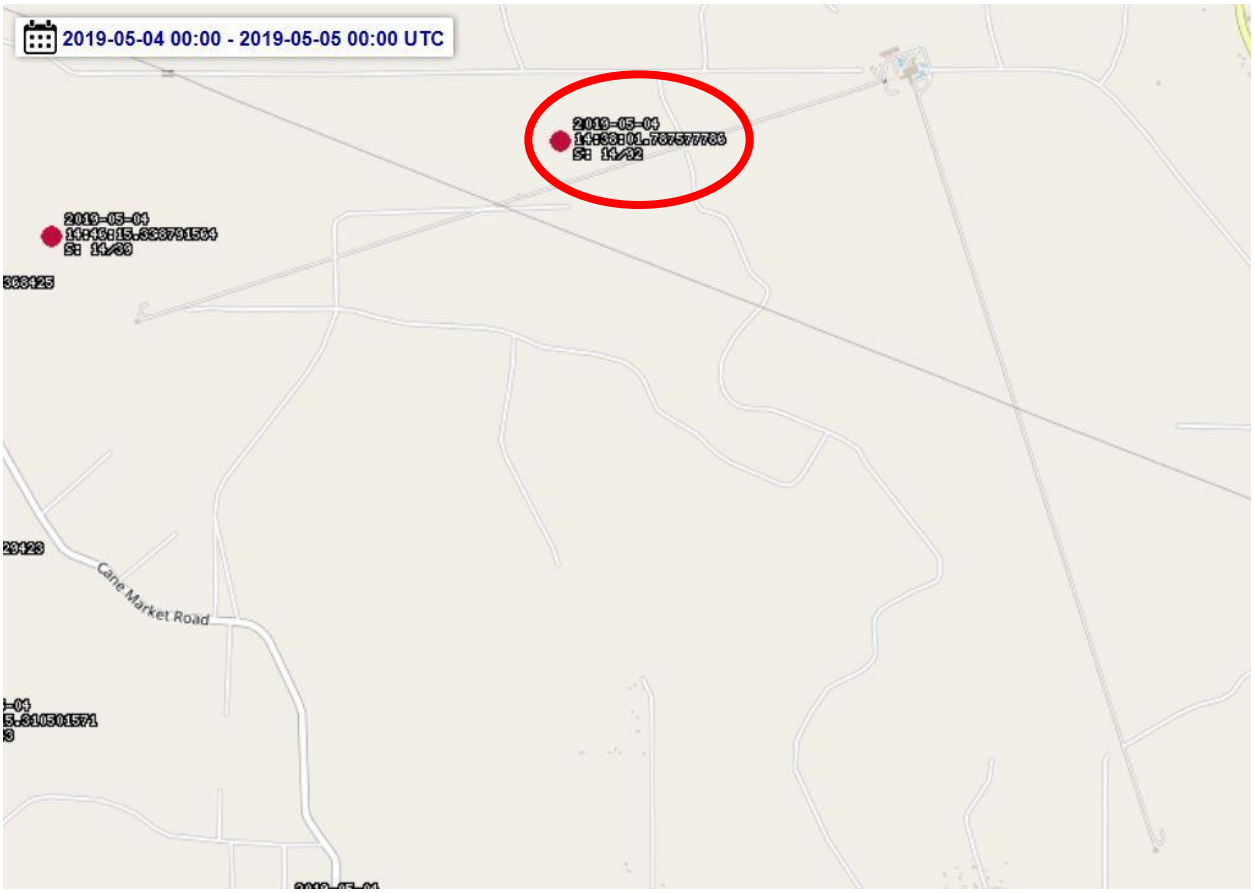

Figure 11: A thunderclap that we used for testing on 4 May 2019 at GPS time 1241012899. Image from LightningMaps.org.

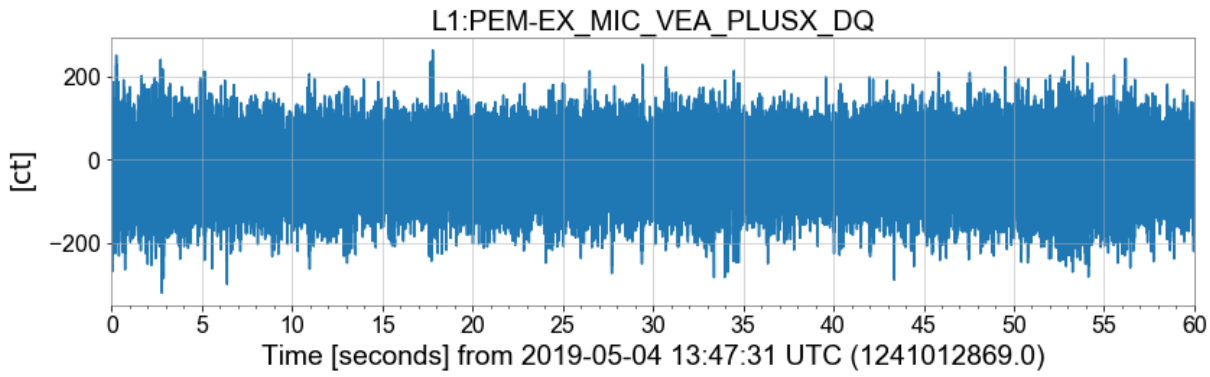

Figure 12: The time series for the thunderclap in Figure 11 with a duration of 60 seconds. While it appeared on the lightning database, it was not loud enough to be recorded in the X-end station mic.

Even with a few failures, my code was able to find thunderclaps. Figure 10 shows a chart containing an example of a few GPS times when my code was able to locate storms. Each one of these was near a specific portion of the detector.

GPS Times for Storms by Location		
Beam Splitter	**Y-End Station**	**X-End Station**
1244750354	1236680804	1241016393
1244825133	1244824858	1241528630
1244825054	1244824921	1242309344
1245535836	1245723804	1244824230

Figure 13: Chart of GPS times for thunderclaps found to have originated from specific portions of the Livingston detector, specifically from near the beam splitter, near the Y-end station, and near the X-end station. This list was generated to be used in future study.

4c. Shorter TDOA

For the third type of testing in the Y-end station, I created another microphone and came up with coordinates for the Y-end station electronics station microphone. The latter did not have coordinates recorded in the PEM schematic.

I did the same runs as before, with a few changes. I hard coded in the temperature since we were considering inside the building. I also removed all microphones except for the two I created and the Y-end station mic. I did not use the cross-correlation function to get the TDOA. There were not enough time-series. Instead, I calculated the TDOA and used that. I did test it with small, added errors in TDOA.

Because the microphones are so close, a small change in the TDOA will cause a larger change in the position. On that scale, a shift of 20 meters or so is more impactful than when considering LIGO as a whole.

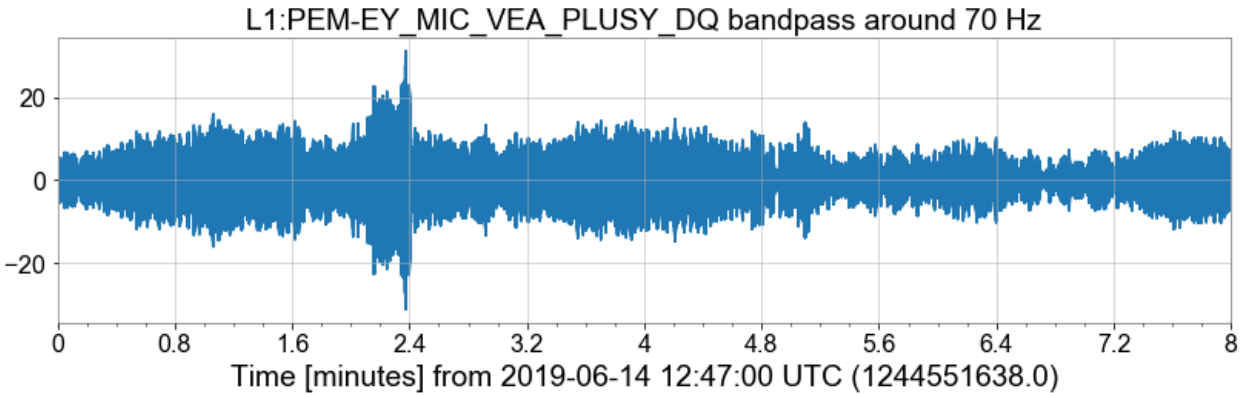

Figure 14: The time series for the Y-end station microphone with a band-pass filter applied around 70 Hz. This time was when the noise from an air condition unit was present. This audio was not recorded in the other end station or in the central station.

End Station Noise Localization	
Microphone	**TDOA**
Y-end Station	0
Y-end station electronics bay	0.0704 seconds
Another mic	-0.0026 seconds

Figure 15: Given a randomly generated origin for the noise near the Y-end station, this chart shows the TDOA for the three mics. The Y-end station mic was used arbitrarily as a reference for the TDOA of the other two. A negative offset means that the sound arrived at that mic prior to arriving at the Y-end station mic.

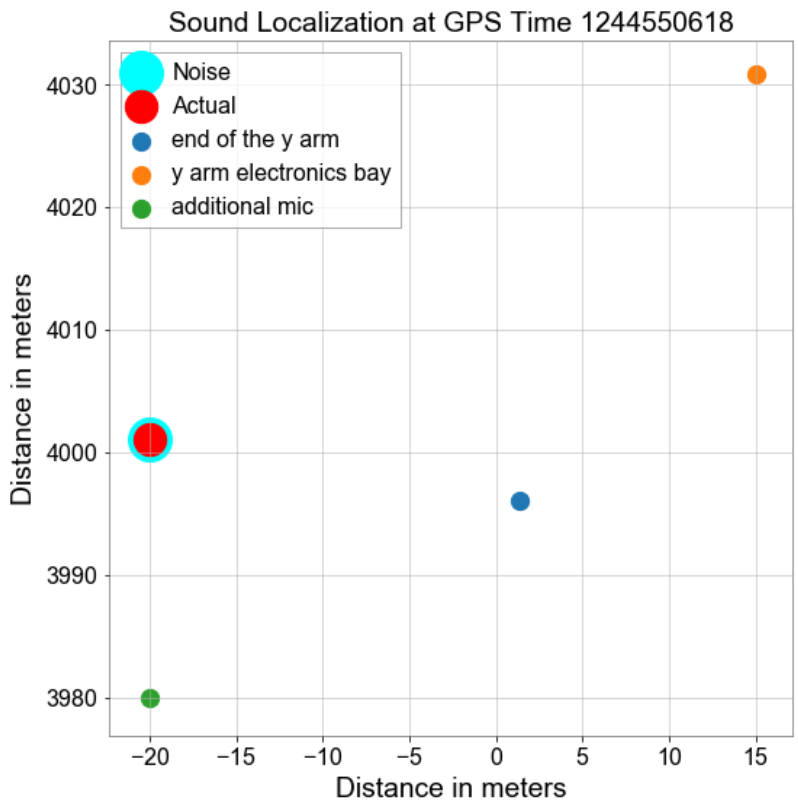

Figure 16: Position of the simulated noise source in figure 15.

5. Code Explanation

5a. Setup

Currently, the code looks on either side of the GPS time entered such that overall it is considering the duration entered with the GPS time in the center.

Lines 38-66 have the definitions for several functions that I used. The function descriptions should be in the comments above each.

Lines 72-87 are getting the speed of sound. I access the thermometer file and then find the average temperature around a 4 second interval at the GPS time entered.

5b. Adding/ Using Another Microphone

Using or adding another microphone is fairly simple. Go to the section of code between lines 90 and 125.

The variable total_mics_stored holds the number of microphones the code is keeping track of. Each microphone needs a position in (x, y) coordinates centered around the beam splitter. The variable num_mics is how many microphones are being used for the guess solution method of the code.

To add a microphone, change total_mics_stored and then enterer the full channel name as the next spot in the mics array. Enter a nickname in the mic_nearest array. Add it to the pos array as well. For example, if this new microphone is number 6, enter its x-coordinate as pos[6][0] and y-coordinate as pos[6][1]. Please note that the numbering for each array starts at 0, not 1. The position needs to be as accurate as possible and is in meters.

To change which microphones are in use, edit the in_use array. This is a list that needs to have at least three values. It designates which of the microphones are being used for the TDOA. The first one listed will be used as a reference for the cross-correlation of the others. The reason why you cannot use all of them is because some of the microphones do not have data that goes back as far as the others. The mid station mics did not exist during some times when the end station mics did. The same will probably be true for any microphones you add. If you are not sure if data exists at a certain time, use the verbose parameter, as mentioned in Section 6.

5c. Calculations

Once it has which microphones to use, the program will gather the data and begin cross-correlation. The cross-correlation is calculated on lines 160-163. These lines call the getDeltaT function which is included on lines 54-59.

Lines 170-182 check for warnings surrounding the TDOA. See Section 7 for more information on that.

Lines 184-202 are for the first solution method, the one which used matrices; this is used as a guess in the fsolve method on lines 207-218. While the matrix method can use more than three microphones, the fsolve method can only use three.

Lines 235-242 calculates which microphone the noise was nearest to.

5d. Output

In section 6, I will discuss the optional parameters. One of which allows the user to print information in PDF and TXT documents. The PDF document contains the time series for the microphones in use and the plot of the noise relative to the microphones at LIGO. The time series plots are on lines 142-157. Lines 220-233 create the plot of the noise's origin.

The TXT document contains extra information such as the names of the microphones in use, the TDOA, any warnings, which station the noise was nearest to, and how far from the beam splitter the noise was.

In addition to those two documents which can be turned off if desired, the code will always print information into a CSV file. This can be found on lines 253-284. If the CSV, named "localization_output" does not already exist, it will create a new one. If the file does exist, it will simply append to the existing file without deleting any information. The CSV stores the GPS time, x-coordinate, y-coordinate, duration, TDOA for all three microphones, and the name of the microphone nearest to the noise's origin.

6. How to Run

The name of my code is “localization.py”.

To get into the environment where the packages required for this code are installed, use the following command in terminal:

```
$ source activate /home/guillermo.valdes/.conda/envs/hht-py37
```

Exit the environment once done with:

```
$ conda deactivate
```

My code is already compiled, but if it had not been, use:

```
$ chmod +x localization.py
```

Run my code with the following command once in the environment:

```
$ ./localization.py -gps=<GPS time>
```

Other parameters can also be added as part of the run command. This code has five parameters, all but one of which are optional. The user must enter the GPS time for the noise. There is no default GPS time; this is required. The optional parameters are found in Figure 17. A dash must be included before each parameter identifier.

Optional Parameters			
Terminal identifier	**Name**	**Description**	**Default**
-vb	Verbose	This should be 1 if you want it to print info while acquiring data.	0
-d	Duration	The length of time in seconds over which the sound arrived at all mics.	60
-f	Filename	The name of the output file without any extension.	localization_output
-i	Info	This should be 1 if you want it to print all information into a PDF and TXT file.	0

Figure 17: A table containing the optional parameters to run my code.

7. Warnings

If you are printing the information out, look at the .txt file to see any warnings. There are two possible warnings. The first is “Cross-correlation function warning: Change in time is too large.” This means that the cross-correlation did not accurately find the TDOA as it was a larger value than was possible.

The maximum value is calculated using the largest distance between two stations and the speed of sound in air at the temperature from that day. I also added 1 second just to make sure it was not too sensitive. Generally, the TDOA should not be larger than about 12 seconds.

The second warning is “Cross-correlation function warning: The correlation between mics <name of mic 2> and <name of mic 3> does not match what was found for each one and <name of mic 1> separately. There might not be a major event at this time or there could be multiple.” This one was most likely caused because of competing noise.

To determine the second warning, I calculated the cross-correlation between the second and third microphones. This is not what I use for the TDOA calculations. To get the TDOA it uses in later functions, it calculates the cross-correlation between microphone 1 and 2 and between microphone 1 and 3. When I have the TDOA between mics 2 and 3, I then calculate the actual TDOA between the two by subtracting the TDOA for mics 3 and 1 from the TDOA for mics 2 and 1. If the difference between the calculated and actual TDOA is larger than about 2 seconds, an error probably occurred.

8. Troubleshooting

To get around the issue of competing noise, change the duration if possible. If you can cut out the other thunderclap or noise and narrow it down to one you need, it will increase the accuracy of the code.

If you answer does not make sense, check the time series and see if there is a major noise at that time which is heard in all microphones. If a noise is restricted to one station, the cross-correlation will not function properly.

9. Acknowledgments

We thank the National Science Foundation for supporting this work through the REU Site in Physics & Astronomy (NSF Grant #1852356) at Louisiana State University.

Special thanks to the LIGO Laboratory and LIGO Scientific Collaboration.

10. References

1. We B.P. Abbott et al. (LIGO Scientific and Virgo Collaboration), GWTC-1: A Gravitational-Wave Transient Catalog of Compact Binary Mergers Observed by LIGO and Virgo during the First and Second Observing Runs. Nov 30, 2018.

2. B. P. Abbott et al. (LIGO Scientific and Virgo Collaboration), Characterization of transient noise in Advanced LIGO relevant to gravitational wave signal GW150914. Class. Quant. Grav., 33(13):134001, 2016.

3. J. Aasi et al. (LIGO Scientific and Virgo Collaboration), Advanced LIGO. Class. Quant. Grav., 32:074001, 2015.