The Possibility of Earlier LIGO to Detect GW150914

Tony Lu

(Dated: July 8, 2016)

Abstract

The detection of gravitational waves was striking news. In some sense, it was quite coincidental that a detection was made almost immediately after the testing launch of the Advanced Laser Interferometer Gravitational Wave Observatory. I took the waveform signal of the event GW150914 to make injections on data from LIGO’s S6 run and recovered them. Overall, none of the signal-to-noise ratio values I recovered was high enough to claim adetection. In the later portion of S6 where noise was weaker, the signal could be identified as a candidate event. However, I also noticed that if the distance from the system is closer, LIGO at S6 would have the potential to make the detection.

I. THE MAIN QUESTIONS

One of the deciding factors that made the detection of elusive gravitational waves possible is the sensitivity of the equipment. Over LIGO’s S5 [1], S6 [2] and O1 [3] (which the scientists previously referred to as S7) runs, the sensitivity was improved significantly each time, with effective attenuation of noise.

At some degree, the detection seemed very much like a coincidence, for no detection in the past two decades was made until the one made within just one week after the testing launch of Advanced LIGO [4]. So here’s a question: was LIGO at any early stage sensitive enough to make this detection? Was the detection of GW150914 a coincidence?

II. BACKGROUND INFORMATION

More than a century ago, Albert Einstein’s General Theory of Relativity stated that we live in a four dimensional world—space-time (three dimensions for space and one for time). Masses cause distortions in space-time, which account for the phenomenon of gravity. The theory predicts that changing mass distribution will generally produce ripples in space-time, which is another prediction from him—gravitational waves. [5, 6] For decades, scientists have been struggling to detect gravitational waves from outer space. They upgraded their detectors over and over again, in order to catch the whispers of distant celestial bodies. Fortunately, on September 14, 2015, the LIGO detectors successfully recorded the signals from two colliding black holes, which was the first direct detection of gravitational waves ever, a remarkable one. [3] It is labeled GW150914.

This section aims to provide background information about LIGO and gravitational waves.

A. Introduction to Gravitational Waves

Einstein found that his linearized weak-field equations have wave solutions. [3] These transverse waves are ripples in the fabric of space-time, caused by some strong and catastrophic happenings in the universe.

According to Einstein’s calculations, gravitational waves are generated by objects accelerating with asymmetric motion. To be more precise, gravitational waves require changing

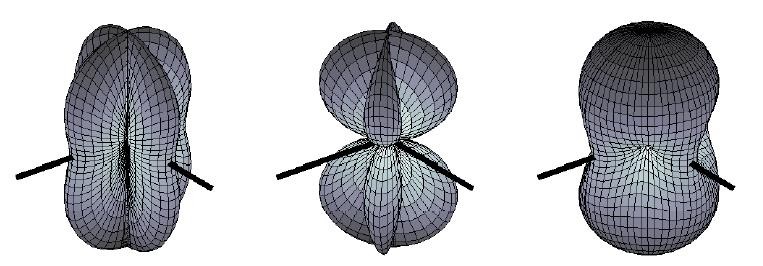


FIG. 1: The pattern on the left is for plus polarization, the middle pattern is for cross polarization, and the right-most one is for unpolarized waves. The black lines are the arms of LIGO detectors, which will be addressed in II B. Source: [8]

quadrupole mass distribution. Generally, the amount of gravitational waves given off is positively correlated to the system’s mass and its speed of motion. Changes in space-time produced by a moving mass are not felt in the distance at once, but they propagate at the speed of light. [7]

Gravitational waves are transverse waves, the same as electromagnetic waves, but they are waves of changes in tensors (quadrupole distortions of space-time), which result in expansions and contractions in lengths in certain directions. Unlike the horizontal and vertical polar- izations of electromagnetic waves, those of gravitational waves are “plus” and “cross”. However, gravitational waves do share lots of similarities with electromagnetic waves. They have frequencies and wavelengths, whose relationship is given by: λf = c, where λ is the wavelength, f is the frequency, and c is the speed of light. They are able to carry energy, momentum, and angular momentum away from the source. [7] The strain amplitude can be derived from Einstein’s quadrupole formula, and it’s inversely proportional to the distance from the mass center. [6] It can also be measured by ratio of the change in length to the original length h = ∆L/L.

Gravitational waves have various sources, but there are mainly four catagories: stochastic background, bursts from gravitational collapse, pulsars, and binary systems. Binary systems refer to those that consist of two bodies rotating aroung each other. Whether or not the two stars collide, gravitaional waves will be generated. The detection made on September 14,

2015 is the collision of two black holes, which is named compact binary coalescence—“chirp”. [3] More details are given by [7] in section 3.

B. Introduction to LIGO

Distortions in space-time is almost impossible to measure directly, for if you use a meter stick, the meter stick, itself, will expand and contract with changing space-time. Fortunately, lights remain the same speed despite any deformation of space-time, which sets the basis for the LIGO detectors. [9]

The detector of LIGO—Michelson Interferometer—was invented by Albert Abraham Michelson, and was first used in the famous Michelson-Morley experiment [10]. Its function is for optical interferometry. With a beam splitter, a light source is split into two arms, and each of the new beams is reflected back and combined again. The result of such combination leads to optical interference, either constructive or destructive depending on the changes in the length of the arms. Therefore, the photo detector could sense the variation in the arm lengths. [9]

The interferometer in LIGO consists of two vacuum arms of equal length—4 kilometers long, which form an L shape. There is a laser light source and a photodetector at the corner of the L, and a beam splitter at the corner. At each end and in the middle of the arms freely hanged four mirrors, which reflect the laser beam, and the two beams are combined into one at the beam splitter. Those mirrors in the middle are used to increase the distance that light beams travel, and as a result, the actual effective length is increased from 4km to 1120 km, greatly enhancing LIGO’s sensitivity. If the arm length changed, the two light beams that were originally in phase would be out of phase, creating destructive interference, which allows the photodetector to sense the infinitesimal change in the length of the arms. [9, 11]

Moreover, the two LIGOs in the US are located in Livingston Parish , Louisiana and Hanford , Washington. Both are L-shaped and of the same size–their arms are all 4km long. The two Ls are in different directions, and they are separated by 3030.3 kilometers on land, which means the planes these observatories are on have a dihedral angle of 27.3 degrees. [30] Therefore, it’s guaranteed that gravitational waves from any directions can be detected, given enough sensitivity.

A brief history of LIGO is given by a website by Caltech [12]. A more technically and

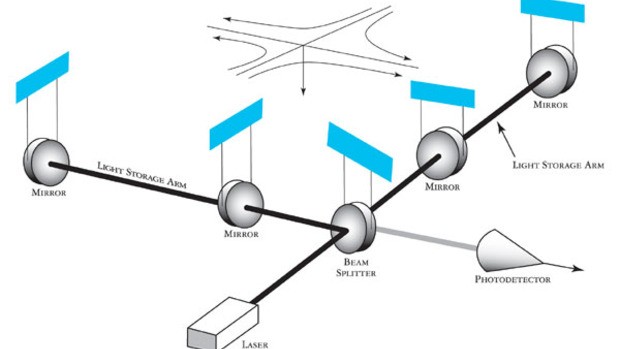


FIG. 2: Basic schematic of LIGO’s interferometers with an incoming gravitational wave depicted as arriving from directly above the detector. Source: [9]

quantatatively detailed description of LIGO is given by the paper on LIGO’s S6 run. [8]

C. Sensitivity of the Detectors

Similar to any real physics experiment, the LIGO detectors face various disturbances from the environment. Considering the small amplitudes of gravitational waves, we see that sensitivity become decisive. At the very beginning of the construction, the interference of the noises is so high that it was even impossible for the detectors to collect data clean enough to analyze. But the quality of data collected has been significantly improved over the several stages of LIGO. The datasets of its S5 (2005-2007), S6 (2009-2010), and O1 (2015 Sep-2016

Jan\*) are available online. [13] Among all these sensitivity-limiting factors are primarily: seismic noise (at relatively low frequencies), thermal noise (at mid frequencies), shot noise (at high frequencies), and the problems with the lasers and electronics. Some more specific details, including solutions, are given by [14–16].

The problems related to sensitivity are curious to investigate. My research is closely related to sensitivity.

D. Injections

In short, injections are used to test if the detectors were working well. There are two types of injections: hardware injections and software injections. The first type is made by manually changing the position of the mirrors in the interferometer. The second type is achieved by adding a signal directly into the data flow. Moreover, there were those called “blind injections”, when a few scientists know it was an injection but the majority were convinced that it really was a detection. One of the most famous events was back in 2010, which was reported on the LSC news site [17].

III. PREPARATION OF DATA FILES

The strain data of the GW150914 event I used is from the LOSC site. Specifically, it’s stored as a waveform.txt file from the “Download the Data” section of the LOSC tutorial [18]. This strain data was later used as the signal for the self-made injection.

As for the template for injection recovery, it’s different from the waveform. It is included in the zip file of another tutorial. [19] The complex template contains two waveforms, one of which is used as the real part and the other as the imaginary part. Moreover, as pointed out in the “Waveform Template” of this tutorial, the templates are not 100% the same as what the scientists actually used. Many subtleties are skipped, for example. But the quality of the templates online is good enough for my investigation. Note that this was different from the strain data for making the injection.

The background strain data of my injections were selected from a number of data files in the “H1” detector from LIGO’s S6 run, from GPS time 931035615 to 971622015. The S6 data archive is available on the LOSC site as well [20]. Because of the probable fluctuation in the power of the background noise over the entire S6 run, I chose 10 different and relatively evenly dispersed data files, each starting at GPS time 931127296 (0.226%), 934846464 (9.39%),

941707264 (26.3%), 941785088 (26.5%), 947154944 (39.7%), 952623104 (53.2%), 959344640 (69.8%), 963629056 (80.3%), 967442432 (89.7%), and 971407360 (99.5%), all lasting for 4096 seconds. The files will be referred to with numbers from 1 to 10, respectively.

The background quality is critical. A piece of low-quality data could result in a misleading outcome. Furthermore, that kind of data files would be vetoed by the scientists. As a result,

even if there really were an event, the segment it lies in would not be searched. And it becomes useless for me do investigate on such a data file. In general, I needed to assure that the background I was injecting on has the proper data quality flags. Because the event of GW150914 is a compact binary coalescence system [3] and the mass is big enough to be catagorized as “high mass” [21], the flag of “CBCHIGH CAT4” should be on. Besides, there should not be any other injections which would affect my recovery, so the flag of “HW” should be off. [22] The specific way to slice qualified data segments is given in the code in appendix section VIII.

The process of gathering the raw data for my analysis could be catagorized into steps—making and recovering the software injection. I made the injections by simply superposing the waveform on to the background signal. Five injections were made to each of the ten data files at randomly selected points. In addition to the original strain, I amplified the waveform signal by factors ranging from 2 to 30, with a serial interval of 2, and injected them at the same points as I did for the unamplified signal. In total, that was 800 injections. To recover the injection, I computed the signal-to-noise ratio (SNR) [23] of each injection using matched filter [24]. In addition, I did the recovery with no injections in the data file, which served as a check of the false alarms, the details of which will be addressed in section VIII. That was 850 SNR values in total. Signal processing was almost entirely the same as that in the tutorial on LOSC site [18]. The exact python code is given in the appendix section VIII with a more detailed description.

IV. DATA ANALYSIS

This section presents the data analyzing process.

A. Computing the RMS of Strain Data

In this case, the root mean squared (RMS) [25] of strain data functioned as a representation of the significance of the noise in each data file. It can be inferred from the RMSs that noise in the detectors is generally louder in earlier stages of S6 than later in later ones.

TABLE I: RMS calculated from each raw data file. File Number 1 2 3 4 5 6 7 8 9 10

RMS (×10−17 ) 15.0 7.68 10.3 8.84 8.72 9.27 5.33 2.64 4.91 3.30

B. The Mean SNRs

As stated in section III, there are in total 850 SNRs computed from the matched filter corresponding to 17 different amplifications, including an amplification of zero (recovering from uninjected strain data). The SNRs at the same amplification from the same file were calculated the average value. These mean values were further sorted into 10 series, each for a single file.

In fact, the SNRs that I—and also the scientists in LIGO—recovered were all time- varying functions. What I actually focused on was the local maximum of each function. As a result, the SNR value recovered from the raw strain data was a highest partial or accidental matching of the background noise to the template. However, it served as a decent indication of the minimum SNR value that could have been a false alarm in each data file. In other words, if the maximum SNR value is not sufficiently higher than that calculated from raw strain data, that SNR is probably a false alarm. As it turns out, what was recovered from raw data was quite close to that which was recovered with an original signal, which can be seen from the first two points of each series in the scatterplot figure 3.

Consequently, the SNRs recovered from the unamplified injection are largely unreliable, so I came up with another way to figure those out, which was to use regression analysis to extrapolate the SNRs of the injected signal at original amplitude.

C. Computing the Linear Regression

Due to the nature of SNR, the amplification (signal power) should be positively correlated to the output. From figure 3 it can be seen that the y and x values are not only positively correlated, but they have a nice linear relationship. Therefore, I chose linear regression [26] as the model.

Before computing each linear regression, I excluded the data points at the first four amplifications, which are not clean enough, from figure 3. The reason for this

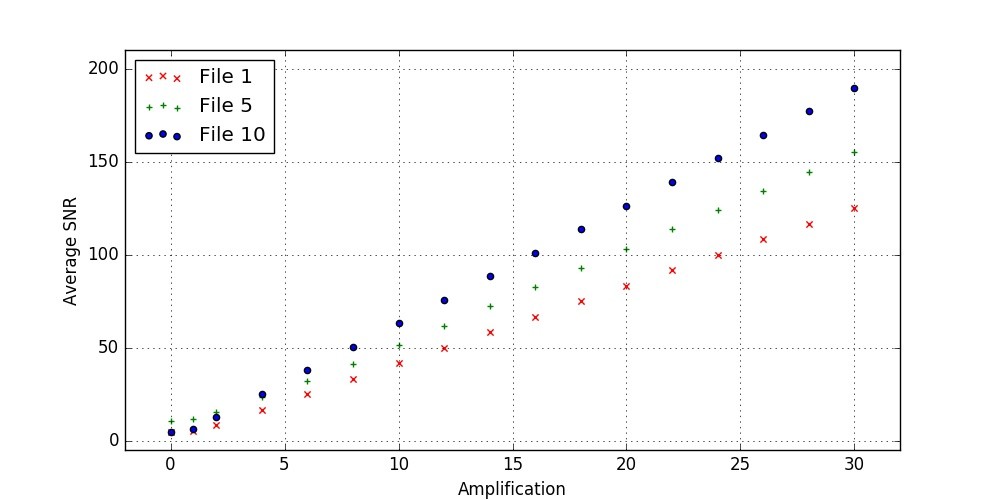


FIG. 3: A scatterplot—mean SNR plotted against the amplification of injected signal—generated by matplotlib.pyplot. The plots are the average maximum SNR values computed from each file at various amplifications of signals. There are three out of the ten series plotted. They represent the early, middle, and late stage of S6. One can see from table I that file 1 contains the most noise, the noise in file 5 is among the middle, and file 10’s noise is the second least.

kind of unclearness is probably the relatively high ratio of the strength of the background noise to that of the signal.

In general, the statistics in table II make good sense. The fourth column of figures (R2)

indicates the fitness of my model: the closer the value is to one, the better the statisitcs fit the model. [27] That these values are so close to one means my extrapolations would be quite accurate. In addition, the interceptions are also very close to zero. Because SNR is the relative power of the signal over the background noise, it should theoretically be zero if there’s no signal present. In reality, however, the SNR is never zero because of noise’s partial matching with the template. Now that the interceptions are close to zero, my extrapolations became more credible.

Finally, the predicted SNRs of the GW150914 signal in different stages all LIGO’s over S6 run were found. Over all, none of those SNRs would not be significant enough to claim a detection. In the early stages where noises were high, the signal would be overwhelmed by the noise, while in the late stage where noises had been reduced, the maximum SNR is

TABLE II: These are the statistics calculated from the mean-SNR series in section IV B by linear regression. Since the model of linear regression in this case is simply SN R = Amplitude × Slope + Interception, I was able to extrapolate the mean SNR at amplification equals one, which gave me the values in the fifth column.

File Number Slope Interception R2 Extrapolated Mean SNR

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 4.167 | -0.073 | 0.999999999511 | 4.094 |
| 2 | 4.371 | 0.093 | 0.999999985140 | 4.464 |
| 3 | 4.171 | -0.692 | 0.999999992354 | 3.479 |
| 4 | 5.151 | -0.487 | 0.999999995775 | 4.664 |
| 5 | 5.155 | 0.134 | 0.999959583093 | 5.290 |
| 6 | 6.164 | -0.042 | 0.999999998879 | 6.122 |
| 7 | 6.279 | 0.206 | 0.999999998734 | 6.485 |
| 8 | 5.953 | 0.178 | 0.999999998084 | 6.131 |
| 9 | 6.032 | -0.036 | 0.999999997650 | 5.996 |
| 10 | 6.328 | -0.201 | 0.999999999774 | 6.128 |

still not significant enough. One can see from figure 4 that the SNR function of background noise it constantly around 4 with some spikes up to 6, regardless of that maximum around

3500 due to hardware injection. In addition, other SNR functions look very similar.

V. SIDE EVIDENCE

In fact, the subsequent successful detections of both a gravitational waves event (GW151226 [28]) and a candidate event (LVT151012 [29]) are also evidence that the detection of GW150914 was not a coincidence. We can infer from these detections that the newest LIGO had finally been sensitive enough to make frequent detections. It appears that more detections will come in the near future.

VI. CONCLUSION

I found that the GW150914 event would not be loud enough for LIGO at S6 run to detect. The signal could, at best, be identified as a candidate event at the later stages

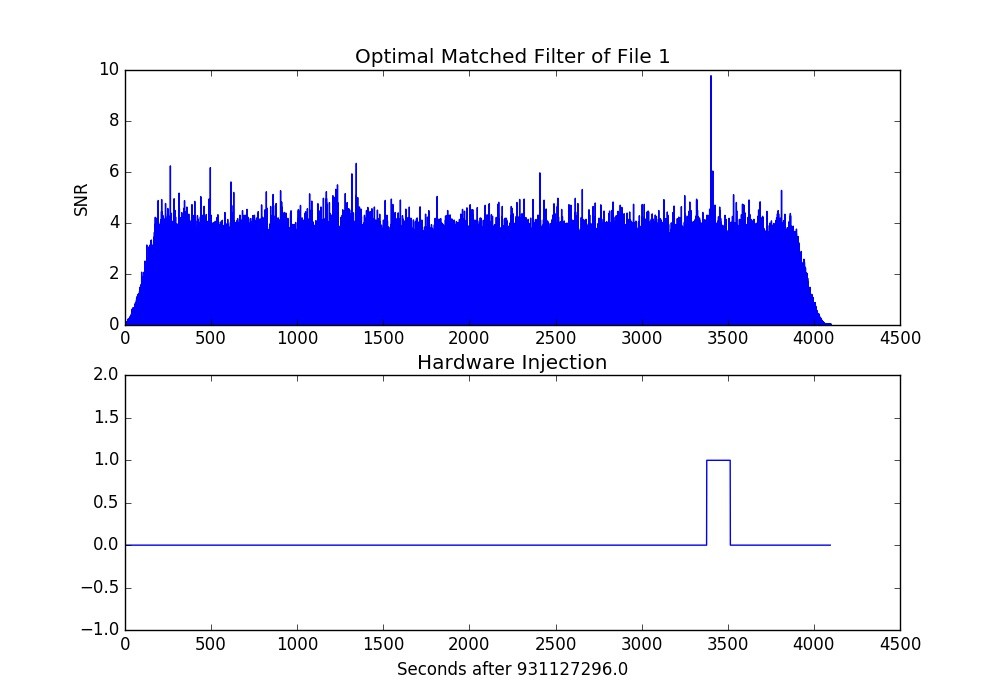


FIG. 4: The upper image is the SNR value over the full span of file one. The lower image flags a hardware injection where there is a bulge in the graph. The highest spike in the upper graph is due to an injection indicated by the lower graph.

where the sensitivity was relatively better, but it’s significance would be far from enough to claim a real detection. In short, LIGO in the S6 run was not sensitive enough to make this detection. Besides, the other two subsequent detections have made it more likely that detecting GW150914 was just a beginning triggered by increased sensitivity. Therefore, the detection of the GW150914 event was not a mere coincidence. In contrast, the successful detection was (and future detections will be) made possible by persisting in upgrading the detectors.

VII. EXTENSIONS

To be added...

[1] LIGO Scientific Collaboration, The s5 data release, URL https://losc.ligo.org/S5/. [2] LIGO Scientific Collaboration, The s6 data release, URL https://losc.ligo.org/S6/. [3] B. P. Abbott et al., Phys. Rev. Lett. (2016).

[4] LIGO CalTech, Ligo timeline, URL [https://www.ligo.caltech.edu/page/](http://www.ligo.caltech.edu/page/)

timeline.

[5] A. Einstein, Sitzungsber. K. Preuss. Akad. Wiss 1, 688 (1916). [6] A. Einstein, Sitzungsber. K. Preuss. Akad. Wiss 1, 154 (1918).

[7] B. Sathyaprakash and B. F. Schutz, Living Rev. Relativity 12 (2009), URL http://www. livingreviews.org/lrr-2009-2.

[8] B. P. Abbott et al., Rept. Prog. Phys. 72 (2009).

[9] LIGO Caltech, What is an interferometer?, URL [https://www.ligo.caltech.edu/](http://www.ligo.caltech.edu/)

page/what-is-interferometer.

[10] A. Michelson and E. Morley, American Journal of Science pp. 333–345 (1887).

[11] MIT LIGO, A comprehensive overview of advanced ligo (2015), URL https://www. advancedligo.mit.edu/summary.html.

[12] K. Thorne and R. Weiss, A brief history of ligo (2016), URL [https://www.caltech.edu/](http://www.caltech.edu/)

content/brief-history-ligo.

[13] LIGO Scientific Collaboration, Timeline, URL https://losc.ligo.org/timeline/. [14] M. Pitkin, S. Reid, S. Rowan, and J. Hough, Living Rev. Relativity 14 (2011).

[15] Y. Levin, Phys. Rev. D 57, 659 (1998).

[16] D. M. Macleod, S. Fairhurst, B. Hughey, A. P. Lundgren, L. Pekowsky, J. Rollins, and J. R.

Smith, Class. Quant. Grav. 29 (2012).

[17] LIGO Scientific Collaboration, Blind injection: Stress-tests ligo and virgo’s search for gravi- tational waves, URL <http://www.ligo.org/news/blind-injection.php>.

[18] LIGO Scientific Collaboration, Signal processing with gw150914 open data (2016), URL

https://losc.ligo.org/s/events/GW150914/GW150914\_tutorial.html.

[19] LIGO Scientific Collaboration, Binary black hole signals in ligo open data (2016), URL https://losc.ligo.org/s/events/GW150914/LOSC\_Event\_tutorial\_ GW150914.html.

[20] LIGO Scientific Collaboration, S6 data archive, URL https://losc.ligo.org/

archive/links/S6/H1/931035615/971622015/simple/. [21] J. Aasi, Phys. Rev. D 87, 022002 (2013).

[22] LIGO Scientific Collaboration, URL https://losc.ligo.org/archive/dataset/S6/. [23] D. H. Johnson.

[24] B. Allen, W. G. Anderson, P. R. Brady, D. A. Brown, and J. D. E. Creighton, Phys. Rev. D

85, 122006 (2012).

[25] E. W. Weisstein, Root-mean-square, URL [http://mathworld.wolfram.com/ Root-Mean-Square.html.](http://mathworld.wolfram.com/)

[26] E. W. Weisstein, Linear regression, URL [http://mathworld.wolfram.com/ LinearRegression.html.](http://mathworld.wolfram.com/)

[27] E. W. Weisstein, Correlation coefficient, URL [http://mathworld.wolfram.com/ CorrelationCoefficient.html.](http://mathworld.wolfram.com/)

[28] LIGO, Data release for event gw151226, URL https://losc.ligo.org/events/ GW151226/.

[29] LIGO, Data release for candidate event lvt151012, URL https://losc.ligo.org/

events/LVT151012/. [30] By Google Earth

VIII. APPENDIX—PYTHON SCRIPT

This is the main Python script I used for my research. Please be noted that an automatic line break would cause an absence of line break operator “\” or a line without the comment notation “#”.

What I exactly did was that I first did the recovery on uninjected data. Then I superposed the unamplified data at five random points and did the same recovery. Then, instead of starting from raw data, I just superposed another signal with an amplification equal to the difference from the current one to the next one, and I did the recovery for the next

amplification. For example, if I was now at an amplification of 10 and the next one is 12, I

just superposed five signals amplified by a factor of 2 to the five according points.

# Programmer: Tony Lu Zhehao

# From: 2AP2 International Cirriculum Center,

# Shenzhen Foreign Languages School, Shenzhen, China

# This script is to make and recover injections on LIGO’s raw data files.

# It generally randomly pick 5 points in each data file to work on.

# The recovery algorithm is adapted from the tutorial of GW150914 on LOSC.

# It has been very slightly altered, but 99% of it is the same.

# URL: https://losc.ligo.org/s/events/GW150914/LOSC\_Event\_tutorial\_GW150914.html

# NOTICE:

# Highly recommend backing up the files, because the injections will not be

# easily removed.

# The output is the average SNR of each injection.

**import** h5py

**import** numpy as np

**import** matplotlib.pyplot as plt

**import** matplotlib.mlab as mlab

**import** scipy.signal as sig

**import** readligo as rl

**from** random **import** randint

#-- Sampling rate equals to 4096 Hz

fs = 4096

#-- Read in the waveform

temp\_time, temp\_strain = np.genfromtxt(’GW150914\_4\_NR\_waveform.txt’).transpose()

#-- Read in the template

f\_template = h5py.File(’GW150914\_4\_template.hdf5’, ’r’)

template\_p, template\_c = f\_template[’template’].value

template = template\_p + template\_c \* 1.j

# to remove effects at the beginning and end of the data stretch, window the data

# https://en.wikipedia.org/wiki/Window\_function#Tukey\_window

**try**: dwindow2 = sig.tukey(template.size, alpha=1./8) # Tukey window preferred, but requires

recent scipy version

**except**: dwindow2 = sig.blackman(template.size) # Blackman window OK if Tukey is not

available

# the length and sampling rate of the template MUST match that of the data.

datafreq = np.fft.fftfreq(template.size)\*fs df = np.**abs**(datafreq[1] - datafreq[0])

# prepare the template fft.

template\_fft = np.fft.fft(template\*dwindow2) / fs

# -- To calculate the PSD of the data, choose an overlap and a window (common to all detectors)

# that minimizes \enquote{spectral leakage} https://en.wikipedia.org/wiki/Spectral\_leakage

NFFT = 4\*fs

psd\_window = np.blackman(NFFT)

# and a 50% overlap:

NOVL = NFFT/2

#-- Set the amplifications of signal to be injected

A = [0,1,2] + **range**(4,31,2)

#-- Define the function to segment the segments into suitable segments

**def** slice\_filter(dq, hw):

#-- Cut the whole segment in to small ones of 60s, with spacing of 1s

i = 0

**while** i < **len**(dq):

**if** dq[i].stop - dq[i].start < 32\*fs:

**del** dq[i]

**continue**

dq.insert(i, **slice**(dq[i].start, dq[i].start + 32\*fs))

dq[i+1] = **slice**(dq[i].stop + 1\*fs, dq[i+1].stop)

i += 1

#-- Delete the segments with injection activated

i = 0

j = 0

**if len**(hw) == 0:

**return** dq

**while** i < **len**(dq) **and** j < **len**(hw):

#-- dq is 100% before hw

**if** dq[i].stop < hw[j].start:

i += 1

#-- dq is 100% after hw

**elif** dq[i].start > hw[j].stop:

j += 1

#-- dq and hw overlap

**else**:

**del** dq[i]

**return** dq

#-- Prepare the files to work on

fileName = []

fileName.append(’H-H1\_LOSC\_4\_V1-931127296-4096’)

fileName.append(’H-H1\_LOSC\_4\_V1-934846464-4096’)

fileName.append(’H-H1\_LOSC\_4\_V1-941707264-4096’)

fileName.append(’H-H1\_LOSC\_4\_V1-941785088-4096’)

fileName.append(’H-H1\_LOSC\_4\_V1-947154944-4096’)

fileName.append(’H-H1\_LOSC\_4\_V1-952623104-4096’)

fileName.append(’H-H1\_LOSC\_4\_V1-959344640-4096’)

fileName.append(’H-H1\_LOSC\_4\_V1-963629056-4096’)

fileName.append(’H-H1\_LOSC\_4\_V1-967442432-4096’)

fileName.append(’H-H1\_LOSC\_4\_V1-971407360-4096’)

#===========================#

#========Main Part==========#

#===========================#

**for** name **in** fileName:

#-- Prepare the document to record the table (SNR vs Ampli.)

f = **open**(name + ’.txt’, ’w’)

#-- Load data from the original file rawFile = h5py.File(name + ’.hdf5’, ’r’) strain\_raw = rawFile[’strain/Strain’].value

CBCHIGH\_CAT4 = (rawFile[’quality/simple/DQmask’].value >> 4) & 1

HW\_CBC = (rawFile[’quality/injections/Injmask’].value >> 0) & 1

GPSstart = rawFile[’meta/GPSstart’].value

rawFile.close()

#-- Load the file again which will be injected dataFile = h5py.File(name + ’.hdf5’, ’r+’) strain = dataFile[’strain/Strain’]

dqInj = dataFile[’quality/injections/Injmask’]

#-- Getting the suitable segement lists

segList = rl.dq\_channel\_to\_seglist(CBCHIGH\_CAT4)

segList\_HW = rl.dq\_channel\_to\_seglist(HW\_CBC)

segList = slice\_filter(segList, segList\_HW)

#-- Pick 10 (or if not more than 10, all) segments randomly

inj\_num = []

**if len**(segList) <= 5:

inj\_num = **range**(0, **len**(segList))

#-- If less than 5, throw a caution

**print** ’Caution: there are less than five suitable spots in %s!’ % name

**print** ’{0} only!’.**format**(**len**(segList))

**else**:

**while len**(inj\_num) < 5:

i = randint(0, **len**(segList)-1)

**if** i **in** inj\_num:

**continue**

inj\_num.append(i)

inj\_num.sort()

List = []

**for** i **in** inj\_num:

List.append(segList[i])

#-- Make and recover injections at each amplitude

**for** k **in range**(0, **len**(A)):

f.write(’{0} ’.**format**(A[k]))

#-- Amplify the template with the difference to next amplification

**if** k == 0:

temp = temp\_strain \* (A[k])

**else**:

temp = temp\_strain \* (A[k] - A[k-1])

#-- Make the injection to every piece

SNRsum = 0

**for** seg **in** List:

#-- Injection starts at the middle

inj\_sample = seg.start + 16\*fs

#-- Superpose the waveform to the signal

**for** j **in range**(0, temp.size):

strain[inj\_sample + j] += temp[j]

#-- Injection Recovery --#

#using the segments in the above section

data = strain[seg]

# to remove effects at the beginning and end of the data stretch, window the data

# https://en.wikipedia.org/wiki/Window\_function#Tukey\_window

**try**: dwindow = sig.tukey(data.size, alpha=1./8) # Tukey window preferred, but

requires recent scipy version

**except**: dwindow = sig.blackman(data.size) # Blackman window OK if Tukey is

not available

#-- Calculate the PSD of the data. Also use an overlap, and window:

# This is where the only change is made, I replaced the PSD of the 32-s segment with

# that of the whole file, because when the file is small the PSD will be greatly

affected by the

# signal in the file, whereas we only want the PSD of the background noise. So taking

the PSD

# of a much larger interval can help eliminating the effect.

data\_psd, freqs = mlab.psd(strain\_raw, 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:

# Multiply the Fourier Space template and data, and divide by the noise power in each

frequency bin.

# Taking the Inverse Fourier Transform (IFFT) of the filter output puts it back in

the time domain,

# so the result will be plotted as a function of time off-set between the template

and the data:

optimal = data\_fft \* template\_fft.conjugate() / power\_vec optimal\_time = 2\*np.fft.ifft(optimal)\*fs

#-- Normalize the matched filter output:

# Normalize the matched filter output so that we expect a value of 1 at times of just

noise.

# Then, the peak of the matched filter output will tell us the signal-to-noise ratio

(SNR) of the signal.

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)

SNRmax = SNR[indmax]

SNRsum += SNRmax

SNR\_avr = SNRsum / **len**(inj\_num) f.write(’{0}\n’.**format**(SNR\_avr)) dataFile.flush()

**print** ’%s at amplification = {0} finished.’.**format**(A[k]) % name dataFile.close()

f.close()