SURF First Interim Project Report

Examining Change Points in LIGO Hanford Data over Observing Run 4 to Improve Detector Performance

Iain Morton
Advised by Dr. Ansel Neunzrt, Dr. Camilla Compton

DCC: T2400227
California Institute of Technology
LIGO Hanford Observatory

Contents

1 Narrow Spectral Artifacts

1.1 Background and Motivation

The detection of gravitational wave pulsar data at LIGO Hanford Observatory is typically subject to long-duration degredations, which leads to noise artifacts over the O4 data run. As such, certain frequencies may show spikes or "long-duration noises", which then impact long-duration searches. While these frequency values which produce these artifacts may be viewed manually, it is necessary to implement appropriate statistical modeling in python to determiine the impact of these artifacts over a search duration.

Investigation of the amplitude of the peak of the artifacts over this long-duration search may be implemented into python via well-known mathematical methods of change point detection in signal processing [?]. For the purposes of artifacts, we can categorize the data as a random process of input signals of $y = \{y_1, \dots , y_T\}$ with a cardinality of τ. Furthermore, as the incoming pulsar signals will experience abrupt changes, we can furthermore categorize these change-points as estimated indices $t_1^* < \dots < t_{K^*}^*$. For our purposes, we will need to estimate the number appropriate K^* segmentations.

Data segmentation is reliant upon identifying the correct signals to segment the data [?]. This is first accomplished with an appropriate sum of all the costs of segmentation, typically referred to as a cost function. For the purpose of incoming signals of gravitational waves, we can identify the most sufficient cost function as:

$$c(y_{a,b}) := \sum_{t=a+1}^{b} \|y_t - \tilde{y}_{a,b}\|_2^2 \tag{1}$$

Where $\{y_t\}_{t=a+1}^{b} (1 \leq a < b \leq T)$ is simplified to $y_{a,b}$ and $\tilde{y}_{a,b}$ refers to the empirical mean of the sub-signal.

Provided a given frequency with a sudden change in data, the data itself can be segmented into two separate regions amongst a signal $y_{a<b}$.If two points along The discrepancy between the two regions can be calculaed as:

$$d(y_{a,t}, y_{t,b}) := c(y_{a,b}) - c(y_{a,t}) - c(y_{t,b}), (1 \leq a < b \leq T) \tag{2}$$

Additionally, we can establish a criterion function, dependent on a possible segmentation and the signal itself. This is referred as $V(\tau, y)$, but often referred instead as $V(\tau)$ for simplicity. We can mathematically define the criterion as the summation of the chosen cost function appropriate for the data:

$$V(\tau, y) := \sum_{k=0}^{K} c(y_{t_k \cdot t_{k+1}}) \tag{3}$$

Because the incoming data does not have specified break point changes, it is hence necessary to introduce a penalty function. This would lead to minimal complexity of analyzing breakpoints. This can be optimized as:

$$\min_{\tau} V(\tau) + \text{pen}(\tau) \tag{4}$$

Similar to the cost function, the penalty function itself may be chosen. For simplicity and for the flexibility of the chosen smoothing parameter with the incoming data, our team found the following penalty function most suitable for Pulsar data:

$$\text{pen}(\tau) := \beta|\tau| \tag{5}$$

where $\beta > 0$ [**?**].

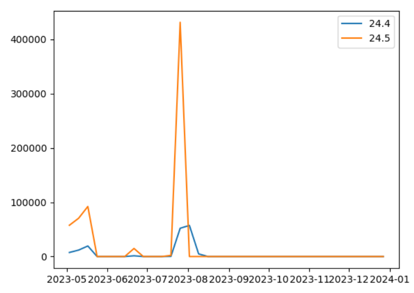
1.2 Ongoing Progress

Over the run of O4a, identification of provided frequencies of 24.4 hertz and 24.5 hertz have proven to show significant narrow spectral artifacts. These frequencies were chosen arbitrarily to test any such implementation of the necessary code to calculate the given discrepancy over the time period of O4a strictly for testing.

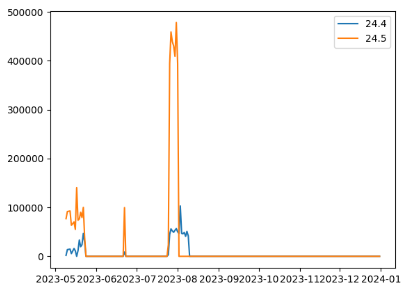
Since beginning the program, code has since been developed to first take all 2024 data with a ".timeaverage.npz" file, whih correlates to the specific time in which a narrow spectral artifact has been observed. After choosing both 24.4 and 24.5 hertz, the line heights (amplitudes) of these weeks were stored in arrays and plotted over the duration of O4a times. These were initialliy plotted utilizing matplotlib, shown as figure1a.

Due to the imprecision of the periods in which the artifact may be interpreted, it was hence necessary to change the scope of the line frequencies over individual days over O4a instead of on a weekly basis. The following figure displays the display of the artifact line height again at 24.4 and 24.5 hertz, but instead on a daily basis. This is demonstrated in figure 1b.

The next steps will be to implement both the criterion functions and the penalty functions into the python code in terms of the array of data pulled from specific files. This is likely to be accomplished via calculating the criterion of each respective element in the chosen arrays, then plotted with respect to the number of chosen change points. Following this, the length of the possible segmentations τ will be calculated, and from there will be added to each value of $\min_\tau V(\tau)$. This will be hence plotted and recorded. Figure 3 below is the criterion is plotted, which mimics our expectations of the consistent lowering of the criterion function with respect to the changepoints.



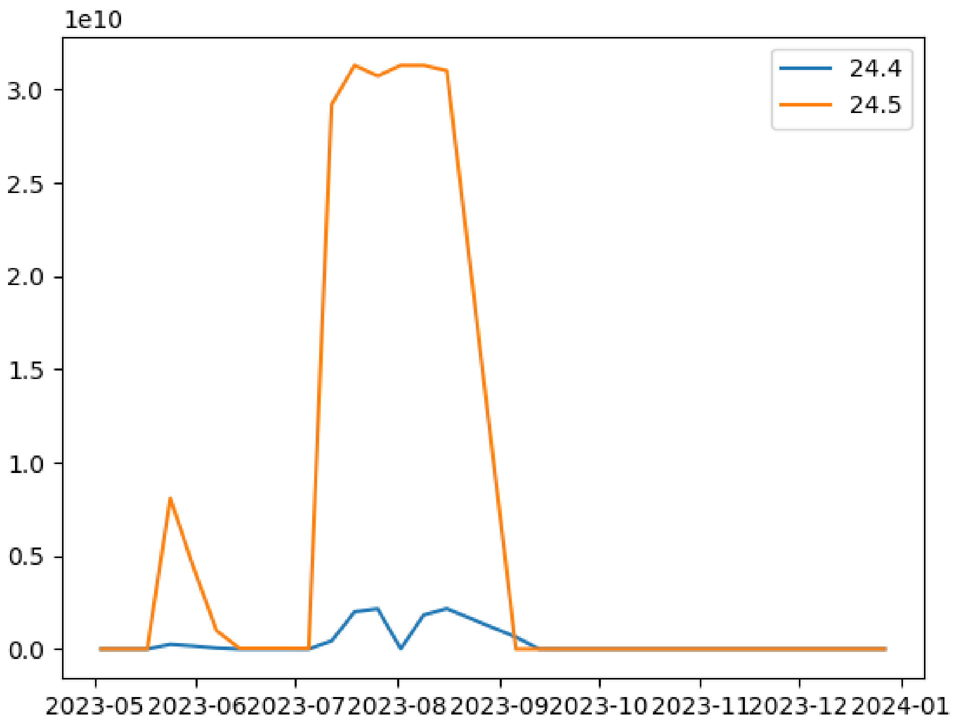
(a). 1a



(b). 1b

Figure 1: Line heights versus time for O4a, for a) weekly and b) daily

The calculation of the discrepancies was then recorded, in which the cost function of the incoming pulsar data was stored into array values, from which the discrepancies were calculated and the index t_k was chosen as the average between two signals. The discrepancy curve was then plotted, reliant on the length of an arbitrary length of the half-window necessary for data segmentation. The result is Figure 3, still over a daily basis:



(a). Discrepancy of line heights versus time for daily basis

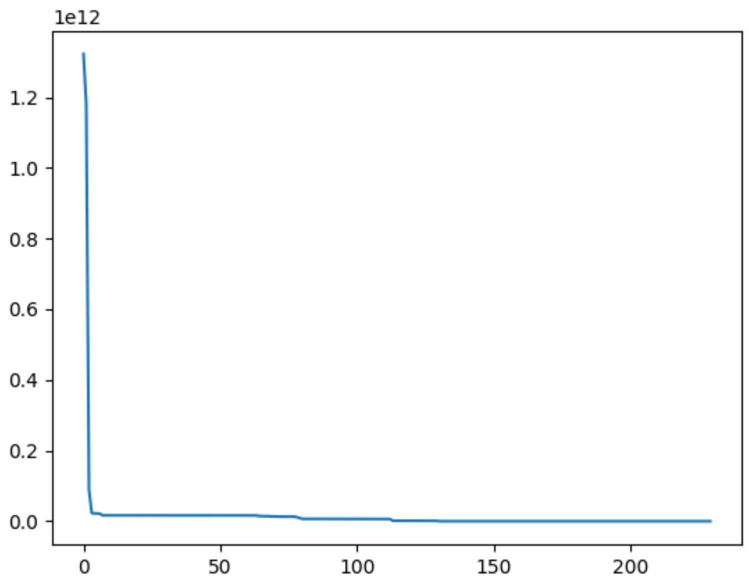


Figure 3: Plotted criterion function (no penalty function)

1.3 Challenges and Potential Solutions

An initital challenge of the project was developing the necessary code to properly retrieve the Pulsar data from the ".timeaverage.npz" files, which was promptly resolved and all data was stored as an array. After carefully implementing appropriate functions for the cost function and discrepancy, an ongoing challenge is to store each element in the array to then calculate the criterion and plot this with respect to each segmented value.

Furthermore, an additional challenge is the typical method of supposing an appropriate smoothing parameter β for the penalty function. While the penalty function was chosen for its simplicity, it also assumes an established model with known parameters. This is certainly not the case for incoming frequency data from pulsars, in which the data is sporadic. As such, a common difficulty will be determining a sufficient smoothing parameter which would not violate the statistical modeling of the data itself. One such potential approach is to potentially collaborate with separate researchers at other institutions for the purpose of determining the best fit for the data itself, which may lead to an approximate data model.

2 Lockloss and Glitches

2.1 Background and Motivation

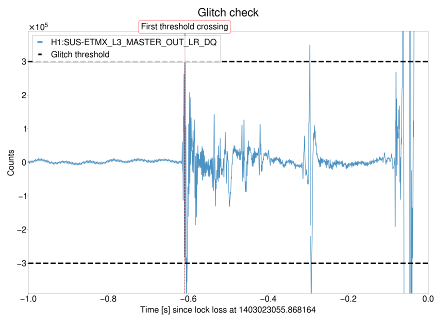
LIGO's interferometers are well-known for their precision due to the 4-km arm length. As such, gravitational wave detection rely strictly on optical cavities "locked" to a fixed length with feedback control/loop. However, due to the sheer necessity of precision, data cannot be collected when the interferometers are misaligned [**?**]. This is commonly referred to as a "lockloss".

Locklosses may occur due to a variety of technical and/or environmental factors. Events such as earthquakes and high winds may cause lockloss, costing several hours of lost data [**?**]. Ongoing efforts to computationally analyze the variety of causes and external phenomena are being utilized, one such tool being known as the "locklost" tool, coded using python scripts. The locklost tool's primary role is to search for past locklosses and depict plots for the precise times in which the detectors lose lock. The locklost tool itself relies on a variety of plugins for searching and appropriately plotting when the interferometer losses lock, as well as detecting technical and enviornmental factors in the surrounding area.

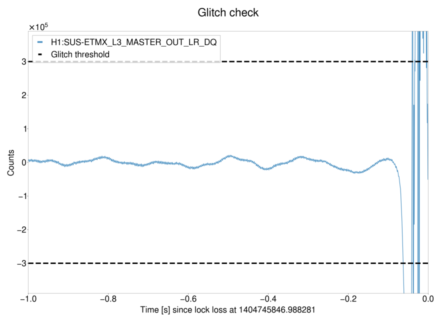
An interesting phenomena occuring approximately 1 second to 100 milliseconds prior to locklosses are detection of "glitches" within certain detection windows. Gravitational wave signals from the differential x and y arm length is held constant using the ETMX suspension. The lowest level of this suspension is dubbed "`ETMX_L3'`", where glitches can be seen. The origin of these glitches in this channel are unknown. Nonetheless, they must be differentiated from a lockloss itself. As stated, if these glitches occur, they are typically between the 1 second to 100 milliseconds prior to the lockloss themselves, thus requiring explicit computational detection.

2.2 Ongoing Progress

The first step towards this area of research was becoming knowledgeable of how locklosses are monitored via the extensive usage of channels. It was important to note that locklosses themselves are due to a variety of both instrumental and environmental factors, thus thorough insight regarding how locklosses may be related to a variety of both internal and external phenomena



(a). 2a



(b). 2b

Figure 4: Glitch threshold check for lockloss events 1403023055 and 1404745847, respectively

was important to consider and read documentation over. The following step was then to become familiar with the git repository and how to edit python code via my own home device which could be then accessed by the computing grid and ran. Setting up personal homepages and creating appropriate directories took approximately half a week, which was then followed by a few days of further testing and administrative approval.

Following this, the first step was developing a plugin to supplement detection that would tag any glitches as seen on `ETMX_L3` within the 1 second to 100 millisecond time window. Establishing a hard threshold was determined to be the most optimal path to do so, specifically due to the complicatiions of alternative methods of comparing computed deviation prior to the glitch. Patterns indicated that glitches commonly hit at least approximately 300×10^3 counts. However, complications arose when registering appropriate windows for the plugin to tag glitches instead of locklosses. As such, the method of identifying the glitches transitioned to calculating the standard deviation of an incoming signal prior to a glitch to the standard deviation of the glitch itself. While there were initial challeneges (will be discussed thoroughly in *Challenges and Potential Solutions*), there was a success in analyzing a particular frame of lockloss events via the glitch plugin. This is shown in figures 2a and 2b. The GPS times were tested dependent on chosen dates, with the examples given being lockloss event 1403023055 (June 21st, 2024, Figure 2a) and event 1404745847 (July 11th, 2024, Figure 2b). The former event in June registered and was tagged as a glitch, and the latter event was not registering at all despite significant seismic activity, thus proving the validity of the plugin.

The next step will be to search a wide variety of lockloss data for glitches, which would prove immensely useful for analyzing many of the same lockloss events which could be devised to be caused via similar ways, i.e., comparing glitches in various locklosses which are known to be due seismic activity.

Another potential path would be to add a variety of plugins for channel saturation nearing locklosses and comparing a variety of channels to the main IFO channel nearing 50 milliseconds. However, this project will still need to be finalized and approved by staff.

2.3 Challenges and Potential Solutions

The first initial challenges were minor, in which being familiarized with the git repository was a temporary hurdle in modifying and testing code on both my home directory and while being connected to the computing grid. This has since been resolved, and code can now be easily modified locally on a text editor and then pulled into the home directory.

The next challenge were glitch detections themselves. Initially, we devised ways in glitches might be computationally detected. Two methods were proposed:

1. Establishing an appropriate threshold that runs above and below the zero mark to check for glitches in the incoming signal.
2. Comparing standard deviation values from prior the glitch to the glitch itself, which will then tag the glitch.

For simplicity, establishing a hard threshold was decided as the simpler and more efficient path, with approximations of the threshold itself being afforded to fit the particular glitch. Nonetheless, issues arose to decide the particular threshold for any such case. Furthermore, ongoing problems persist in that the lockloss itself is tagged as a "glitch" in the proposed time window. Solutions involved determining the event identification number of the lockloss, which would be subtracted from the refined GPS time in which the lockloss occured. Doing so would ensure that the lockloss event itself is not mis-tagged as a glitch, and then isolate our actual glitch event for appropriate tagging. However, further complications arose when deciding if the various windows were sufficient. To ease this complication, the standard deviation callculations are currently being pursued and tested. As such, the next solution to consider would be using this plugin to test several events for glitches simultaneously.

3 References

References

[1] C. Truong, L. Oudre, N. Vayatis, Selective review of offline change point detection methods, Signal Processing University Paris, 2019 pp. 1-11

[2] A. Buikema, et. al, Sensitivity and performance of the Advanced LIGO detectors in the third observing run, Physical Review D, American Physical Society, 2020, pp. 1

[3] R.Abbot, et al., GWTC-3: Compact Binary Coalescences Observed by LIGO and Virgo during the Second Part of the Third Observing Run, LIGO Scientific Collaboration, Virgo Collaboration, and KAGRA Collaboration, American Physical Society, 2023, pp. 1