Technical Note **LIGO-T2300145–v1** 2023/09/05

Noise analysis for the FROSTI adaptive optic

Sophia Arnold

1 Abstract

Uniform absorption and laser scattering into higher modes are the primary limitations on LIGO's laser power capabilities. A proposed solution to these problems is the Front Surface Type Irradiator (FROSTI), which is an annular ring heater designed to combat losses from thermal deformities on LIGO's test masses. In order to install FROSTI into LIGO's system, an in-depth intensity noise analysis must be performed on its heater components to ensure FROSTI will not introduce new sources of noise. The noise of FROSTI's heaters must fall below LIGO's pre-determinded threshold, which is roughly one hundred times smaller than the shot noise of LIGO's photodetectors. To measure this noise, we have designed a GUI, which takes in two voltage signals from a Red Pitaya, computes the Power Spectral Densities and Cross Spectral Density of these signals, and then averages them over the entire run-time of the GUI. This GUI is adaptable for wide-ranging applications outside of measuring the noise of FROSTI's heater elements, and has the potential for use in any system with electrical noise.

2 Introduction

The first prediction of gravitational waves was by Oliver Heaviside in 1893. Noting the symmetries between the behavior of the force of gravity and the force between electric charges, he proposed that gravity may also have wave-like behavior as to match its electrical field partner [1]. After Heaviside, the next major conjecture on gravitational waves came from Henri Poincaré, when he predicted that gravitational waves move at the speed of light [2]. Without experimental proof, these hypotheses had some traction within the physics community, but the first real proponent of gravitational waves came from Einstein's theory of General Relativity. Gravitational waves are what we would consider to be ripples in space-time, with detectable ones being byproducts of massive stellar events, like colliding black holes, or pulsars. Contrary to the behavior of electromagnetic waves, gravitational waves interact very weakly with matter, making them impossible to detect without the help of massive interstellar events [3]. While their weak interaction with matter makes detecting gravitational waves more of a challenge, experimentally exploiting this characteristic would give us key insight into the behavior of their astrophysical sources.

The first direct detection of gravitational waves was by aLIGO, the advanced Laser Interferometer Gravitational-wave Observatory in 2015. LIGO is effectively two massive Michelson interferometers, each which have two 4 km long perpendicular arms that form an L-shaped detector. An example of the basic premise of a Michelson interferometer is shown in Figure 1. The general premise of LIGO, is that in an extremely sterile environment, changes in the length of the arms will be a direct result of contortions and contractions that arise from very powerful gravitational radiation. These changes in length can be detected by minute phase differences in the laser beams when detected, allowing us to directly measure the effect of a gravitational wave's path through the detectors.

aLIGO is an advancement project for LIGO, which began in 2008, to upgrade the facilities of LIGO. In aLIGO and in future improvement initiatives to LIGO, such as LIGO A+ and LIGO A#, minimizing noise and loss from the laser is central to improving LIGO's detection.

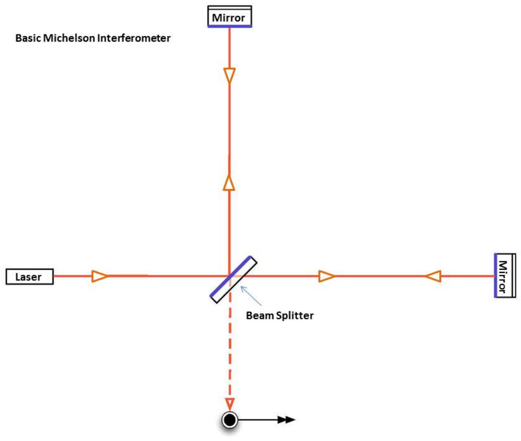


Figure 1: Diagram of basic Michelson Interferometer [4]

In advancing the precision and accuracy of LIGO's measurements, we are able to further understand both the behavior of gravitational waves and the behavior of the interstellar events. A diagram summarizing the aLIGO's improvements upon LIGO's initial design are shown below in Figure 2.

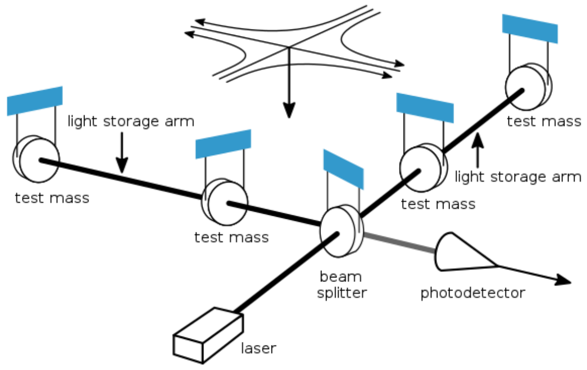


Figure 2: Diagram of aLIGO's updated interferometer [5]

A central goal of aLIGO and LIGO A+ is achieving higher laser power within LIGO's arms. Currently, LIGO's detection capabilities are limited at frequencies higher than 10^2 Hz by quantum shot noise, as shown in the purple line's relationship with the black line in Figure 3 [6]. By lowering the photon shot noise within the detector, the quantum noise also lessens. Photon shot noise drops with a factor of $\frac{1}{\sqrt{N}}$ where N is the number of photons incident on the test mass. Increasing laser power, increases this N, therefore reducing quantum noise and improving LIGO's sensitivity at higher frequencies [8].

Higher powered lasers introduce added potential for scattering and lensing from defects in the reflector's surface, which is already a major issue limiting the capabilities of aLIGO. A central source of this noise is from point absorbers and uniform coating absorption on the mirror sources scattering the beam that is central to LIGO's data collection [9]. To combat this noise, the FROSTI (Front Surface Type Irradiator) is being developed. FROSTI is an annular ring heater, which to applies a corrective heating pattern to the test mass to minimize scattering and power loss from the beam. The device has a diameter of 34 cm and

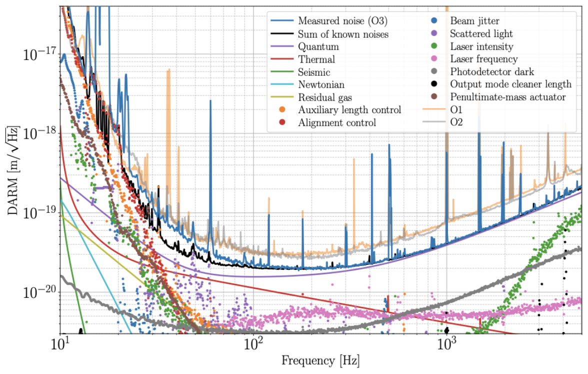


Figure 3: Noise Budget of LHO O3 [7]

will be placed 5 cm from the targeted test mass (Figure 4). There are eight heating elements which are placed along the circumference of the ring, which apply the incident radiation.

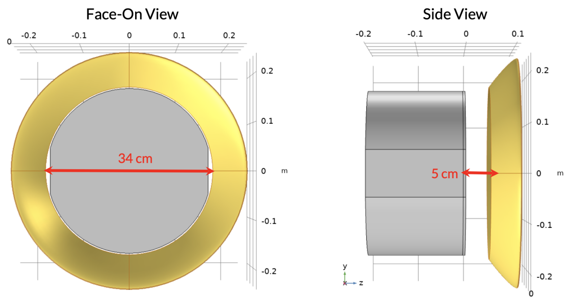


Figure 4: Model of the FroSTI prototype [9]

Theoretical models of the desired irradiance pattern have been produced, but experimental data has yet to be taken to confirm agreement with the expected profile. In experimentally testing FROSTI, it is necessary to perform an intensity noise analysis of FROSTI's heater elements. For a successful implementation of FROSTI onto LIGO's test masses, FROSTI's heater elements must introduce intensity noise that is less than 100-times below the noise floor of a photodector. In order to test for noise that is below this threshold, we must first design a system which can take extended periods of thermal radiation data and then analyze its received data to identify FROSTI's intensity noise performance. This SURF project focuses on creating a Python-based GUI interface that performs this analysis, and can now be applied to finding noise fluctuations from FROSTI's heater elements.

3 Objectives

1. Construct optical layout consisting of green laser and two photodetectors.
2. Compute the Power Spectral Densities (PSDs) of each channel along with the Cross Spectral Densities (CSDs) of the signals acquired from the optical setup using Python code of a finite data list.

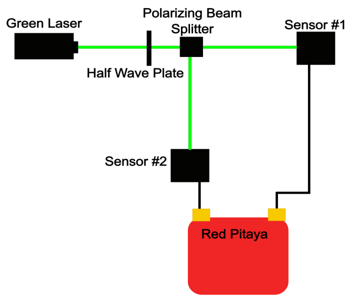
3. Apply this framework to create an interactive GUI for analyzing the intensity noise performance of FROSTI's heater elements among other electrical devices.

4 Methods

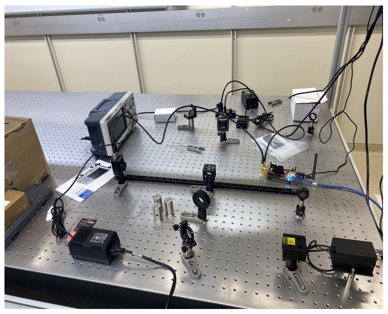
The first two weeks of this project had two primary objectives: first to construct the optical setup for the experiment and second to begin developing code which would dynamically generate the the Power Spectral Densities and the Cross Spectral Density (CSD) of the intensity noise of a green laser. The goal is to compute the PSD's and CSD of the raw data from the Red Pitaya as the PSD's are representations of the distribution of signal power from each of the individual channels, and the CSD represents the distribution of signal power for both channels.

4.1 Constructing Optical Setup

The first step in this project was to construct the optical table setup that noise intensity GUI would be collecting data from. The optical setup, as pictured in Figure 5(a), is a single green laser source, which is split into two beams by a polarizing beam splitter. Once the beam is split, the light is fed into two separate channels connected to a Red Pitaya, which we use to collect data from the signals. The largest challenge with this setup was achieving equal power sent to each detector. By feeding the beam into a half wave plate before the spilt, we are able to adjust the power that is sent along each arm. An image of the physical, in-lab setup is pictured in 5(b).



(a)



(b)

Figure 5: (a) Model of the optical setup for noise analysis of the green laser. (b) Image of the physical optical setup for intensity noise analysis of the green laser.

The construction of this optical table set-up took place over the first day of the SURF program. Throughout stages of the development of the intensity noise GUI, the laser system was switched out for a function generator to determine whether the GUI would produce values within the desired sensitivity. Yet, the overall setup with the Red Pitaya transmitting the desired data stream remained the same throughout the duration of this project.

4.2 GUI Development

A abstracted pipeline diagram of the data stream for the created noise intensity is pictured in 6. This system was modeled off of the method used in a Fermilab Holometer paper [10], and has been adapted for use in this project, specifically with the Red Pitaya acting as the signal receiver.

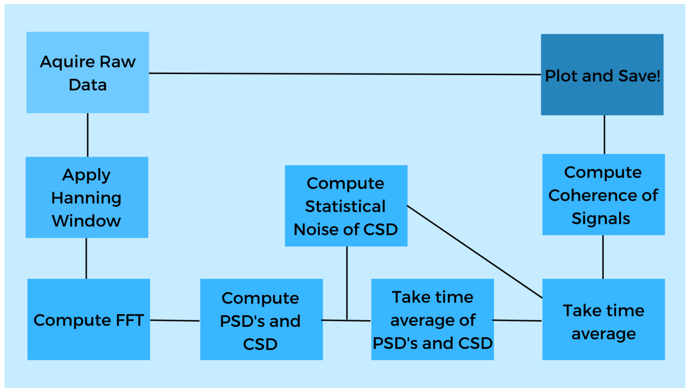


Figure 6: Data processing pipeline for intensity noise GUI.

4.2.1 Implementing Power Spectral Density and Cross Spectral Density Analysis: Starting With Finite Lists of Raw Data

After initial setup and data collection from the Red Pitaya setup, the focus of the project moved to performing the desired analysis of the raw data from the Red Pitaya using Python. The goal of the final GUI was to generate a dynamic plot based on Figure 7 as given in a Fermilab intensity noise analysis of a Holometer [10]. This GUI's job will be to receive the data, then generate and continuously update a visible plot of the Amplitude and Phase information from time-averaged CSD and PSD values.

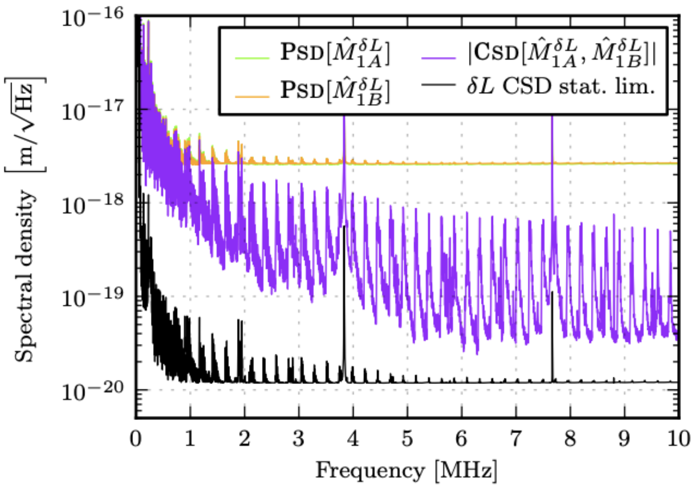


Figure 7: Plot from [10], which gives the thermal noise of one interferometer given by a CSD.

The beginning stages of GUI development were dedicated to analyzing finite sets of raw data, saved from the Red Pitaya. This allowed us to create a successful framework that analyzed raw data prior to adding the complexity needed for a dynamically updating GUI.

The finite data analysis code was built using framework outlined in [11]. Specifically, to perform the desired analysis of the raw time series from the Red Pitaya, the GUI makes use of two scipy.signal functions: scipy.signal.weltch, and scipy.signal.csd. These functions work by partitioning the data set, applying a Hanning window to partitioned sections of raw data given from the Red Pitaya, the computing the FFT of this section, and finally taking the averages of the result of the PSD's and CSD [11, 10]. The Hanning window is defined by the equation:

$$w_j = \frac{1}{2}[1 - \cos(\frac{2\pi * j}{N})] : j = 0...N-1$$

Where the windows are symmetric by:

$$w_j = w_{N-j}$$

To compute the PSD and the CSD, first we must calculate the window sums defined by as:

$$S_1 = \sum_{j=0}^{N-1} w_j$$

$$S_2 = \sum_{j=0}^{N-1} w_j^2$$

Then, these functions put the computed values along with the values calculated in the FFT algorithm, through the following equations to compute the Power Spectra (PS) and the Power Spectral Density (PSD) based off of the Discrete Fourier Transformed data (y_m) [11]:

$$\mathrm{PSD}\left[\widehat{\mathrm{M}}_1^{\mathrm{ex}}; f, t\right] = \mathrm{H}_1^{\mathrm{A}}(f,t) \cdot \overline{\mathrm{H}_1^{\mathrm{A}}(f,t)}$$

Figure 8: General equation for calculating power spectral density of signal, as described in [10].

To compute the Cross Spectral Density (CSD), we simply apply these same equations for the PSD to data which has been multiplied with a complex conjugate of the FFT of one of the two signal channels. This is shown in the following equation:

$$\mathrm{CSD}\left[\widehat{\mathrm{M}}_1^{\mathrm{ex}}, \widehat{\mathrm{M}}_2^{\mathrm{ex}}; f, t\right] = \mathrm{H}_1^{\mathrm{A}}(f,t) \cdot \overline{\mathrm{H}_2^{\mathrm{A}}(f,t)}$$

Figure 9: General equation for calculating cross spectral density of both signals intercepted from Red Pitaya, as described in [10].

When using the scipy.signal functions to compute the PSD's and CSD, there were two necessary parameters to be included alongside the raw data segments from the two channels.

These were: the sampling frequency, and the size of each partition used to compute the FFT. The sampling frequency is determined by the decimation used in the Red Pitaya, which will be further explained in a later section of this report. Then, the size of partitions for the FFT is determined by both the sampling frequency and the bandwidth range of frequencies we want to observe when everything is processed.

These functions are called for each 16k data segment given from the Red Pitaya, and then averaged in specialized system explained in the following section.

4.2.2 Continuous Data Stream Averaging

To achieve a measurement sensitivity of 100 times less than the sensitivity of our photodetectors, the GUI most perform a time average of its computed CSD and PSD's. To implement this, consideration needed to be taken of the data averages that were performed within the scipy.signal.weltch and scipy.signal.csd functions. Each time these functions are run on an 16k finite data list sent from the Red Pitaya, the 16k data list is split into eight segments, each of 2048 (N) elements in length, and then averaged together post processing. To account for this initial averaging, the function performs a weighted average on the existing total average with this new eight-segment average, and eventually saves and sends this new total average to be plotted.

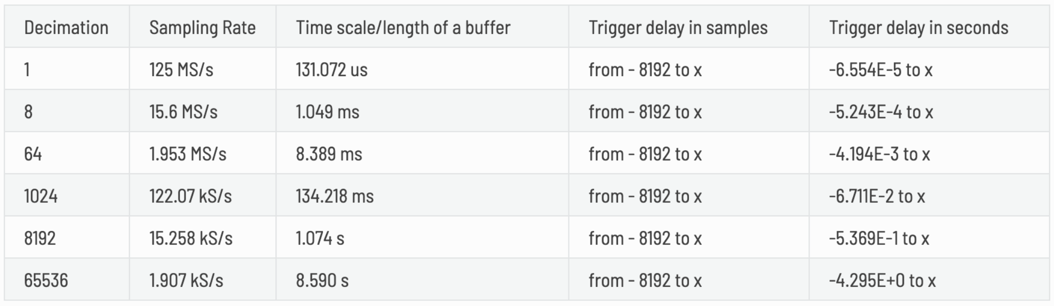
4.2.3 Red Pitaya Decimation

To implement the prior analysis into the final GUI, the decimation rate of the Red Pitaya needed to be decided upon. The decimation of a signal processor, in our case the Red Pitaya, is a way to reduce the sampling frequency of a signal to a lower one. Effectively, this allows for better performance for a signal processor looking to take large amounts of samples, without saving a massive amount of data. It functions by imposing a low-pass filter on the incoming data stream and by removing some of its samples [12]. For the noise intensity GUI to perform at the desired rate, while also reading out values within the desired frequency bandwidth of FROSTI's heater elements' predicted noise, we needed to perform a computation.

Following the Nyquist theorem, the maximal useful frequency is $f_{Ny} = f_s/2$ where f_s is the sampling frequency [11]. The goal for this GUI was to read frequencies up to 1kHz. As the Red Pitaya is able to give more accurate measurements when reading samples that have at or less than half the Nyquist frequency of a given decimation, we originally chose a decimation of 65536, with a sampling frequency of roughly 2kHz.

Once the decimation rate was decided upon, the Red Pitaya then needed to operate at it. This was a difficult feat to accomplish, both in communicating for the Red Pitaya to properly decimate within the Python package, and in getting the Red Pitaya to read out accurate data, and not misrepresented by signals that were more than half the Nyquist Frequency of the decimation rate.

To correct for the Red Pitaya's incorrect decimation of the signal, we needed to manually halt the data stream within the raw data collection part of the code. This halts the data stream pipeline for the entire buffer duration of the decimation rate, as was specified in Red Pitaya



Decimation	Sampling Rate	Time scale/length of a buffer	Trigger delay in samples	Trigger delay in seconds
1	125 MS/s	131.072 us	from - 8192 to x	-6.554E-5 to x
8	15.6 MS/s	1.049 ms	from - 8192 to x	-5.243E-4 to x
64	1.953 MS/s	8.389 ms	from - 8192 to x	-4.194E-3 to x
1024	122.07 kS/s	134.218 ms	from - 8192 to x	-6.711E-2 to x
8192	15.258 kS/s	1.074 s	from - 8192 to x	-5.369E-1 to x
65536	1.907 kS/s	8.590 s	from - 8192 to x	-4.295E+0 to x

Figure 10: Suggested decimations and their sampling frequencies from the Red Pitaya documentation [12].

documentation [12]. Without manually halting the pipeline, the Red Pitaya would sample too quickly, taking in an excessive amount of data without decimating. This issue was first found in the calculated PSD's and CSD of the incoming data stream giving peaks that were significantly larger than the input signal. Using a function generator, it was quickly noticed that the GUI was producing values less than expected by slight, but noticeable margins. Once the manual halt of the data pipeline was implemented, these issues disappeared and the GUI was then able to output peaks that were within the statistical limit of the desired outcome.

After the above problem with incorrect power-peak frequencies was resolved, it was discovered that incorrect voltage readings, which we had thought to be a by-product of the failing decimation, were an independent problem needing to be solved. While using a decimation of 65536, the expected Nyquist frequency for our data is about 2kHz. While this implies that we are free to sample frequencies up to 1kHz, we learned that the Red Pitaya is unhappy sampling signals at or close to the Nyquist frequency. Because of this, and the problems it had the potential to cause for analyzing the intensity noise of FROSTI's heater elements, we decided to switch to a lower decimation which would give a larger frequency range to sample in. The decided decimation then became 8192, and the voltage-reading issues quickly disappeared.

4.2.4 Splitting CSD into Amplitude and Phase Components

As the CSD value is a complex number, we care about both its amplitude and its phase. Both the phase and the amplitude of the CSD value is necessary in order to understand its full, complex behavior, and in order to determine whether the GUI is performing as we wish.

In order to separate the amplitude component of the CSD, we use the function numpy.abs, a way of taking the magnitude of a complex-valued list of data. This then plots alongside the two PSD's, and the statistical noise of the CSD amplitude, which is computed using the following equation:

$$\text{Statistical Noise} = \frac{\sqrt{PSD_0 PSD_1}}{0.947n}$$

where n is the number of averages taken of the list of data over time.

The phase is found through the following equation:

$$\text{Phase} = \arctan 2(\frac{real(PSD_0)}{img(PSD_1)})$$

Once the phase value has been computed, it is then plotted on an axes below the CSD amplitude plot.

4.2.5 Including CSD Amplitude and Phase Information Plots: Final GUI Interface

In the final GUI interface, banners with the amplitude peak values of the signals' voltages, the peaks of each PSD and of the CSD, and the value of the phase component of the CSD at the peak frequency are includes in addition to plots. Within the code, there are the capabilities to plot the raw data from the channels as well as the coherence of the channels. An image of the final interface of the GUI is shown in Figure 11.

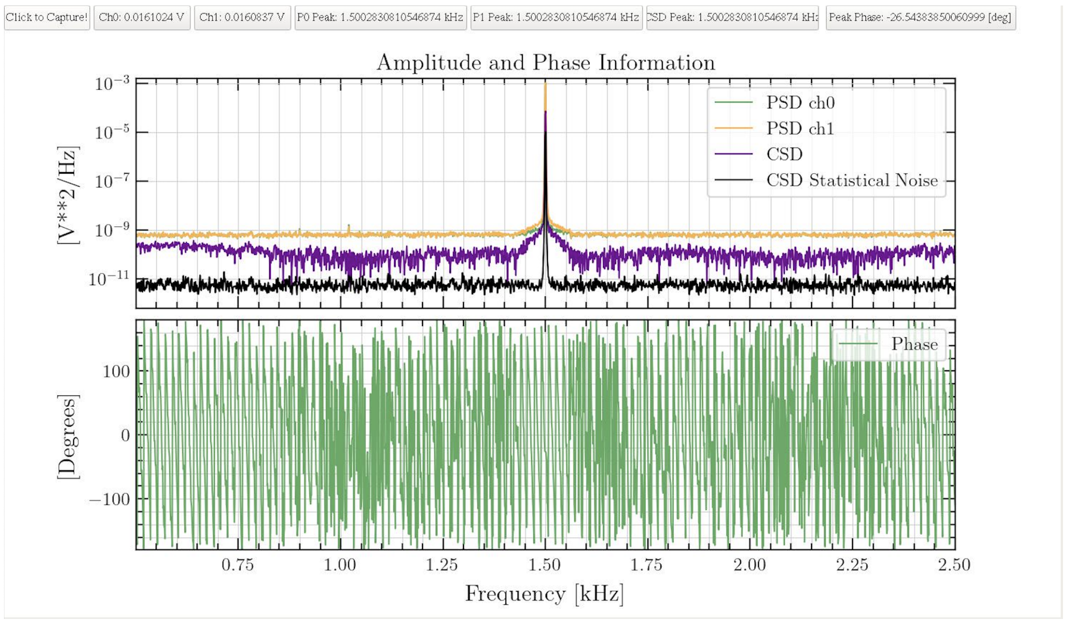


Figure 11: Final interface of intensity noise GUI. Additional capabilities for GUI to plot raw data from channels and coherence of channels.

4.3 Function Generator Tests

To determine whether the GUI performed as intended, a series of varying signal, function generator tests were done. They ranged from inputting various frequency and voltage values to confirm the GUI's performance across its entire intended range. As mentioned in the Red Pitaya Decimation section, this is how various issues affecting the GUI's performance were found. Look to that section for a more in-depth analysis of the issues that arose and their solutions.

Each test run on the GUI confirmed its perfomance. Both Figures 11 and 12 exhibited the desired performance. Figure 12 shows the GUI's performance when a signal of white noise was sent into the GUI from the fucntion generator, and then averaged over time. The harmonics shown are from the electrical signals in the Richardson lab from the electrical outlets in the room. The bump in CSD values just below 1kHz is the measurement the GUI was designed for: general ranges of high noise areas. Figure 11 depicts a specific frequency sent in at 1.5kHz, with a band of white noise superposed with the signal. The goal for the GUI is for noise bands to stay high in the CSD and over time while the rest of the band drops, showing specific areas of high noise. Figure 11 shows this as there is a bulge around a high-noise area, with the rest of the CSD curve slowly falling down.

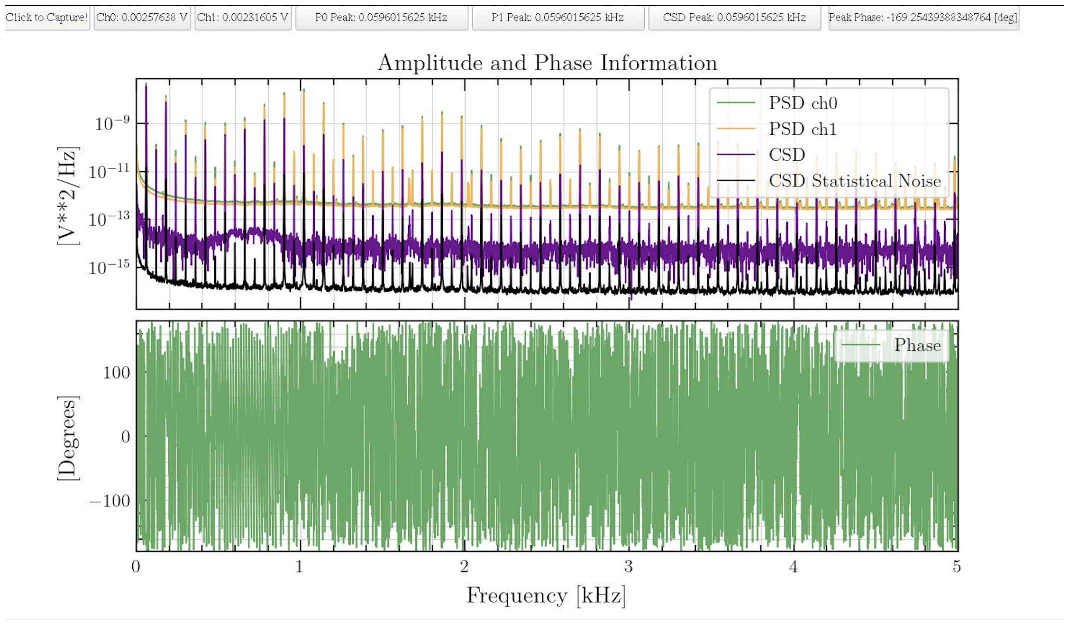


Figure 12: Test of GUI over an hour of data collection with white noise fed into both channels. Voltage was driven at 5mV peak to peak, with white noise fed in.

All of these tests confirmed the performance and accuracy of the GUI to a great degree.

5 Summary and Next Steps

In the coming months, this GUI will be used to test the heater elements of FROSTI, first using the Red Pitaya interface, and then being adapted for use through a LIGO Real Time System that is currently being built for use in the Richardson lab.

While this GUI was designed specifically to test the intensity noise values of FROSTI's heater elements, it has the potential for many other applications. This is a universal code which can be updated and used to find electrical noise signatures in any electrical system.

References

[1] *A Gravitational and Electromagnetic Analogy*. The Electrician, (1893).

[2] Poincar´e, Henri, *Sur la dynamique d' l'´electron.* Comptes Rendus of the French Academy of Sciences 140, 1504-1508 (1905).

[3] Rana X. Adhikarli, *Gravitational radiation detection with laser interferometry.* American Physical Society (APS), LIGO P1200121, v3 (2014).

[4] https://www.ligo.caltech.edu/system/media_files/binaries/237/original/Basic_michelson_labeled.jpg?1435862648

[5] Phoebe Zyla, *A Calibrated Blackbody Source for Testing Next-Generation Wavefront Actuators.* LIGO T2200206 v6 (2022).

[6] Brooks, A. F., Vajente, G., Yamamoto, H., Abbott, R., Adams, C., Adhikari, R. X., Ananyeva, A., Appert, S., Arai, K., Areeda, J. S., Asali, Y., Aston, S.M., Austin, C., Baer, A. M., Ball, M., Ballmer, S. W., Banagiri, S., Barker, D., Barsotti, L., ... Zweizig, J., *Point Absorbers in Advanced LIGO.* Applied Optics 13, (2021).

[7] Buikema, A., et al. *Sensitivity and Performance of the Advanced LIGO Detectors in the Third Observing Run.* LIGO P2000122-v4 (2020).

[8] Cassidy Nicks, *Developing an In-Air IR Test Facility for Next-Generation Wavefront Control.* LIGO T2200205, v6 (2022).

[9] Jonathan Richardson, *Active Wavefront Control for Megawatt Arm Power* LVK Meeting, LIGO G2200399, v1 (2022).

[10] Aaron Chou, *The Holometer: an instrument to probe Planckian quantum geometry* Classical and Quantum Gravity, IOP Publishing, v34 (2017)

[11] Heinzel, G., Rüdiger, A., & Schilling, R., *Spectrum and spectral density estimation by the Discrete Fourier transform (DFT), including a comprehensive list of window functions and some new at-top windows.* (2002) https://hdl.handle.net/11858/00-001M-0000-0013-557A-5

[12] https://redpitaya.com/documentation/