LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note	**LIGO-T22xxxxx–**	2024/07/26

Interim Report for RL based Lock Acquisition

Anubhav Prakash

California Institute of Technology
LIGO Project, MS 18-34
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project, Room NW22-295
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
Route 10, Mile Marker 2
Richland, WA 99352
Phone (509) 372-8106
Fax (509) 372-8137
E-mail: info@ligo.caltech.edu

LIGO Livingston Observatory
19100 LIGO Lane
Livingston, LA 70754
Phone (225) 686-3100
Fax (225) 686-7189
E-mail: info@ligo.caltech.edu

http://www.ligo.caltech.edu/

Contents

1 Overview

This report presents a brief description of my work and how I spent my time in the first 3 weeks of my SURF project at 40m laboratory, Caltech.

The initial steps involved basic simulations of a Fabry-Perot cavity, followed by transitioning from kat script-based programming to using modern model constructor functions. This phase laid the groundwork for understanding the simulation environment and the necessary coding practices.

Subsequent tasks included performing mode scans, analyzing transmitted power variations, and simulating a high-finesse Fabry-Perot cavity with RF modulation. This analysis revealed the distinct peaks in transmitted power associated with different higher-order modes and their sidebands due to RF modulation.

Further studies involved using astigmatism to identify the peaks and understand the degeneracies when multiple modes resonate at the same cavity length. This led to the identification of peaks corresponding to specific mode orders and their sidebands.

I further found thermally robust and isolated carrier peaks by simulating thermal effects, basically by changing radii of curvature of both mirrors, and identifying cavity lengths that maintain isolated peaks across the range of radii of curvature. A significant portion of the report details a supervised learning exercise aimed at training a neural network on the MNIST dataset. This served as a precursor to integrating Neptune AI and Optuna for hyperparameter optimization in neural network training. The integration facilitated efficient hyperparameter tuning and experiment tracking, streamlining the model development process.

Future work includes extending the supervised learning approach to replicate Finesse simulation outputs, particularly for higher-order modes, and applying the learned techniques to the specific parameters of the 40m lab. The report concludes with an outline of upcoming tasks aimed at furthering the understanding and application of interferometric detection techniques and neural network training for optical simulations.

2 Learning Finesse and Higher-Order Mode Sweep Analysis

2.1 Finesse: A Python Package for Optical Interferometer Simulation

Finesse is a Python package utilized for simulating optical interferometers/systems incorporating optical elements. As described on the Finesse Documentation website, "It employs frequency-domain optical modeling to construct accurate quasi-static simulations of arbitrary interferometer configurations." It is grounded in an object-oriented design and furnishes an extensive set of utility functions to facilitate the handling of intricate simulation tasks, which in this instance involved simulating a Fabry-Perot cavity with a Gaussian laser beam carrying multiple Higher-Order Hermite-Gauss modes up to order 12. The learning process primarily entailed perusing the API documentation website: https://finesse.ifosim.org/docs/

latest/api/index.html and also studying the underlying physics behind Finesse from this website: https://finesse.ifosim.org/docs/latest/physics/index.html

2.2 Simulating Fabry-Perot Cavity and Incorporating Higher-Order Modes

I commenced with utilizing basic examples on Finesse that simulated a basic Fabry-Perot cavity, comprehending it by delving into the details. The example employed model.parse arguments to parse kat script. My initial objective was to reprogram the example without utilizing kat script, instead leveraging modern model constructor functions. Consequently, I immersed myself in the API documentation and identified corresponding functions to incorporate optical components, link them, define their parameters, and ultimately plot them.

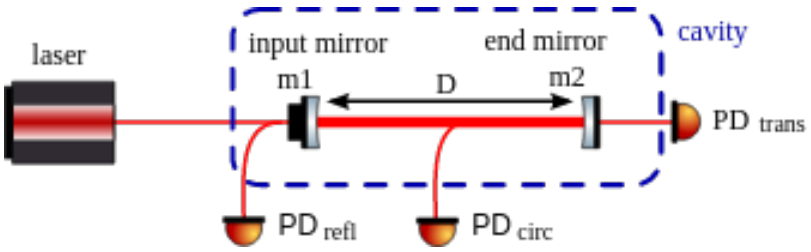

Figure 1: Image Source: Finesse documentation

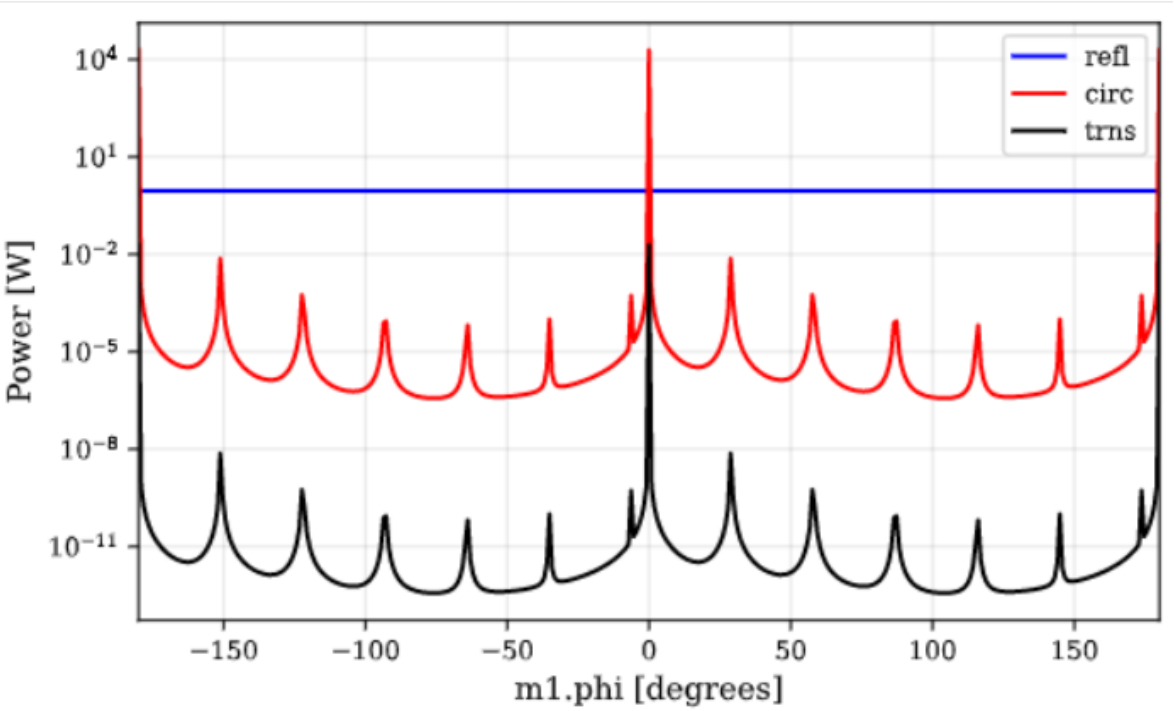

Figure 2: Mode sweep Scan

The second undertaking was to execute a mode scan up to maxtem 12. In the input laser beam, I incorporated Hermite-Gauss modes up to order 12. Now, the fundamental Gaussian mode (0,0) is resonant for a particular length of the Fabry-Perot cavity, but the other modes

resonate at a slightly different length value somewhere between L (cavity length) and $L+\lambda/2$, where λ is the wavelength of the monochromatic laser beam. Owing to this property of the HG modes, we obtain peaks in transmitted power at different values as we vary the ϕ value or the frequency offset of the cavity, which physically translates to minutely altering the cavity length of the Fabry-Perot cavity.

The third task involved simulating a high-finesse Fabry-Perot cavity with RF modulation to the input beam and then plotting the output/transmission power along with the frequency offset to the cavity with the y-axis in log scale to discern the small peaks as well. This generated plots with multiple peaks within the frequency spectral range (FSR, corresponding to the cavity length ranging between L and $L+\lambda/2$) corresponding to various maxtem values, and each of them having a sideband due to RF modulation.

It was further studied with added astigmatism to comprehend the degeneracies, and it was discovered that for a given maxtem value, all the peaks will resonate at the same exact cavity length (frequency offset). For instance, $(0,7)$, $(1,6)$, ..., $(7,0)$ all these peaks will be degenerate. Thus, for analysis up to maxtem 12, we will obtain 12 peaks in the FSR, which will be named as CR_X, with X corresponding to the maxtem value of the peak, and based on that, they will have upper sidebands (USB_X) and lower sidebands (LSB_X).

My subsequent endeavor was to generate the complete plot with one RF modulation for a non-astigmatic, critically coupled cavity with given Radius of Curvature (RoC), Reflectivity (R), and Transmissivity (T) values for the mirror, and then label each peak with a vertical line (see image below) and identify what peak it is. I calculated the FSR range ($c/2L$) using the resonant condition of the cavity when the length is an integral multiple of $\lambda/2$, where λ was 1064 nm. Furthermore, the peak spacing between different maxtem mode peaks was calculated by the formula:

$$crf = \frac{c}{2L\pi}\cos^{-1}\sqrt{1-\frac{L}{\mathrm{RoC}}} \tag{1}$$

Thus, the mode peaks will be located at 0, crf, 2crf, ..., up to 12crf (as we are limiting ourselves to maxtem 12). And then, the locations of the sidebands of these peaks were calculated by simply adding and subtracting the modulation frequency (33MHz) from the CR_X peaks. Find the image here : https://caltech-my.sharepoint.com/:i:/g/personal/aprakas2_caltech_edu/EejQRwsh2MZHq8EYaq8rVZoB-wt7BMqSvfSK9MVNfaWVBQ?e=yyIJ4f

2.3 Finding Robust Cavity Length Values

The next objective was to ascertain the L_{cav} (length of the cavity) values for which we obtain carrier peaks at 0 offset frequency where there are no peaks for a given maxtem value within a range of 1MHz. This was accomplished by writing a simple loop that calculated the position of the peaks using the formulas and then checked if any of the peaks happened to lie in the region 0 to 1MHz or FSR − 1MHz to FSR, because the plot is periodic with period FSR.

The second part was to incorporate the thermal effects of the laser that cause local deformation due to absorption in real lenses and change the radius of curvature locally. Consequently, I endeavored to find robust L_{cav} values, robust in the sense that as the Radius of Curvature (RoC) is varying from 15m to 16m, for those robust L_{cav} values, the carrier peak at 0 must be isolated (no peaks within a 1MHz range) for all those RoC values.

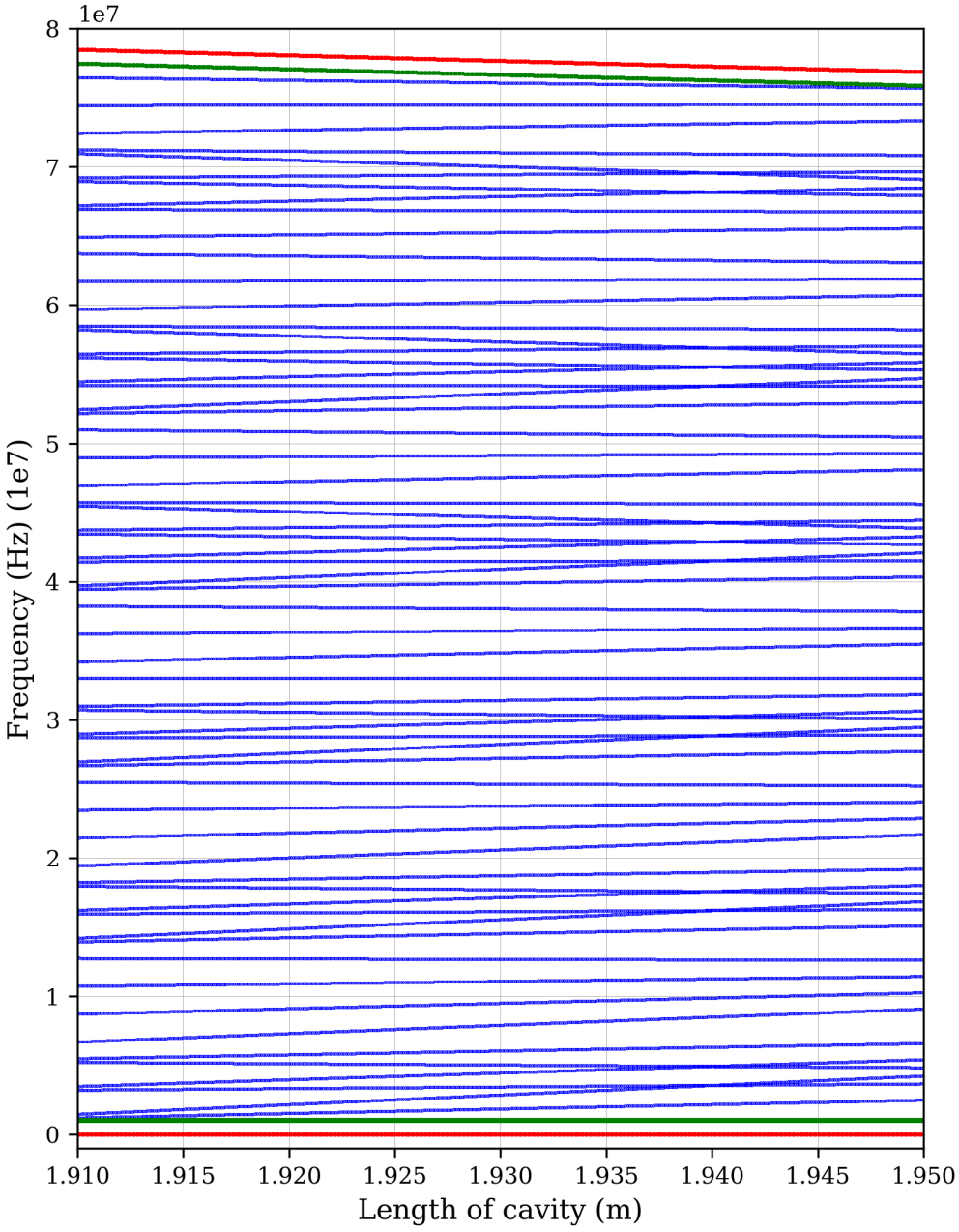

Figure 3: The space between the green and red line(both top and bottom) is where no blue line should reach

This was again calculated by a similar approach using the following code:

```
1
2  fmod1 = 33e6
3  c = 299792458
4
5  def find_good_Lcav(RC, mxtm, Lcav_min, Lcav_max,
       resolution_step):
6      good_Lcav=[]
7      maxtm=mxtm
8      Roc=RC
9      for m in np.arange(Lcav_min, Lcav_max, resolution_step):
10         n=0
11         peak_pnts=[]
12         FSR=c/2/m
13         crf = FSR*np.arccos(1-m/Roc)/np.pi
14         for j in range(0,maxtm+1):
15             peak_pnts.append(np.mod(j*crf,FSR))
16             peak_pnts.append(np.mod(j*crf-fmod1,FSR))
17             peak_pnts.append(np.mod(j*crf+fmod1,FSR))
18         for k in peak_pnts:
19             if (k<=1e6 or k>=FSR-1e6) and k!=0:
20                 n=1
21                 break
22         if n==0 and (1-m/Roc <=0.95 and 1-m/Roc>=0.05):#
               condition to
23         include statbility of the cavity
24             good_Lcav.append(m)
25
26     return good_Lcav
27
28 def modified_intersection(list1, list2, resolution_step):
29 #use the same resolution_step as in function
30 find_good_Lcav
31     intersection=[]
32     for k in list1:
33         for j in list2:
34             if abs(k-j)<resolution_step/10:
35                 intersection.append(k)
36     return intersection
37
38
39 total_Lcav=[]
40
41 #Here below I start scanning for all values of Roc from 15 to
       16 m.
42 for m in range(0,101):
```

```
    r=15+m/100
    total_Lcav.append(find_good_Lcav(r, 12, 0.5, 2, 0.01))

l=total_Lcav[0]
for m in total_Lcav:
    l=modified_intersection(l, m, 0.01)

print(l)
```

```
"""
From the above cell, the output I get is [0.8300000000000003,
0.8400000000000003, 1.3200000000000007, 1.3300000000000007,
1.3400000000000007, 1.3500000000000008, 1.3600000000000008,
1.3700000000000008]

Thus, I ascertained that if the RoC is between 15 to 16,
we are secure if Lcav is within the range (1.32, 1.37).
"""
```

The complete code can be found here on our GitHub repository. Note that I employed an unconventional approach to finding intersections between two lists because of the floating-point precision in Python. Some generated numbers are like 1.32000000003, and thus, if there are two lists, and both are supposed to carry the number 1.32 within them, but if I take their intersection, I will not obtain 1.32 in the output because one list stores it as 1.3200000002, and the other stores it as 1.320000000001 (as you can observe in the output above in the markdown part), and they are deemed unequal when compared with basic intersection functions. Consequently, I defined a function to find intersections for me in such a way that if the difference between two points is 0.1 times the resolution of my sampling, then the two numbers must be considered equal, and this process works exceptionally well to generate proper intersections.

The above output is then plotted as an animation, where each frame corresponds to an RoC value between 15 to 16 m. It is noteworthy that my intention was not to find a specific single L_{cav} value but a small continuous range of L_{cav} values for which the carrier peak is isolated as well as robust, and this condition was upheld up to maxtem 12. I discovered the safe region of L_{cav} values to be the open set (1.32, 1.37) meters. See the animations below:

Link of Animation 1

Note that the above L_{cav} value range is secure only up to maxtem 12. If we include modes with TEM $(m + n)$ equal to 13, we obtain a lower sideband peak entering the zone of a 1MHz range, which can be observed in the following animation, which is up to maxtem 15. You will discern two peaks, LSB_{13} and CR_{15}, entering the region within the 1MHz range, marked by green vertical lines.

Link for animation 2

2.4 Interesting Thing to Notice

One thing you would notice in the second animation is that the peak value of the peaks oscillates. After reading the theoretical derivation of transmitted power for the maxtem 0 case, I gained an understanding of the origin of this oscillation. The output power expression for the CR_0 peak is given as (for critical coupling):

$$P_{\text{trans}} = \frac{T}{1 + R^2 - 2R\cos(2kL)} \tag{2}$$

Assuming that there will be similar expressions for power transmitted when higher modes are resonant. The main part to notice is that the length at which the modes oscillate is changing with a change in RoC, and the change rate is faster if the mode order is higher. Thus, for a faster change in L, the power expression will oscillate faster due to the presence of $\cos(2kL)$ in the denominator, where L corresponds to the resonating cavity length for the mode. This can be clearly observed in the second animation, where the peak corresponding to CR_0, USB_0, and LSB_0 are not oscillating because their frequency offset value (which can be bijectively mapped to the length of the cavity in the FSR range) does not change, but peaks corresponding to CR_{15}, LSB_{15}, and USB_{15} oscillate the fastest because their L value (or frequency offset value) is changing faster than all the other peaks.

3 Learning the Physics Behind the Interferometric Detection Techniques

3.1 Learning from the 40m Lab Tour

Thanks to Jan Carlo (alias JC), who provided me with a very informative and descriptive lab tour and patiently answered all my questions, I learned about the setup of the 40m lab, right from the optical bench that generates the infrared laser beam and makes the beam go through pre-mode cleaners with feedback loops, and then to the initial mode cleaner (the large one) and then to the 40m interferometer. He explained in detail how the mirrors are suspended and actuated to actively control their motion and showed me the actual mirrors (or lenses, I should say) through the view doors. He utilized the posters within the 40m labs to further elucidate me about power recycling and how it works. We did not delve into absolute experimental detail, but he furnished me with sufficient information to build an intuitive physical understanding of the system.

3.2 Reading Article by Andreas Freise: Interferometer Techniques for Gravitational-Wave Detection

- I primarily learned about the linear algebraic approach to solve for the amplitudes after an EM wave interacts with an optical component (particularly mirrors and beam splitters) and complex systems built from it (Cavities and simple interferometers).
- I also learned about the frequency domain analysis of the electric field signals that carry multiple frequencies, modulation of various kinds, etc.

- I learned about how we readout the signals and construct complex numbers corresponding to the amplitude of various frequency elements of the electric fields using I and Q phase of the demodulated (and low-pass filtered) output.
- I learned about basic interferometers and their power outputs for different channels (Transmission, circulation, and reflection). I gained knowledge of the mathematics behind the observed signal for various conditions.

4 Supervised Learning Exercise

Before jumping into Reinforcement learning, it was suggested to be beneficial to try supervised learning exercise. I was asked to do the basic supervised learning exercise which was to train a neural network to use the MNIST dataset to train to detect handwritten digits. The code used can be found on the github repository using this link.

The exercise involved training my neural network and then using it to build an interactive app with the help of YouTube tutorials to interactively see the model predicting the digits that you draw on your screen using your mouse.

The model was not very proficient in predicting the digits, which prompted me to make a detour and learn profoundly about the basics of neural networks and a little bit of the mathematics it utilized to make a prediction. I primarily learned this from the following Youtube Playlist by Grant Sanderson.

I tried other combinations of hyperparameters and training algorithms, but I did not make any improvement in the accuracy of prediction.

I left this problem partially unsolved to proceed with Neptune AI and Optuna integration.

5 Neptune-Optuna Integration

5.1 Introduction

Optuna is an open-source hyperparameter optimization framework that scans over all the hyperparameter values available while training a neural network model and helps us automatically find the best combination of hyperparameters for the model.

On the other hand, Neptune AI is a cloud-based platform that provides tools and infrastructure to build, deploy, and manage machine learning models and experiments.

Integrating Optuna with Neptune AI allows us to leverage Optuna's efficient hyperparameter optimization capabilities directly from within the Neptune AI platform. This integration streamlines the model development process by enabling users to configure and launch Optuna optimization studies seamlessly within Neptune AI's collaborative workspace. The integration also ensures that all hyperparameter values, metrics, and model artifacts generated during the optimization process are automatically tracked and logged in Neptune AI,

facilitating analysis, reproducibility, and collaboration among team members.

5.2 What did I do?

You can find the code used here: GitHub Repository.

In the provided code, I have demonstrated the integration of Optuna and Neptune AI for hyperparameter optimization and experiment tracking while training a convolutional neural network (CNN) model on the MNIST dataset for handwritten digit recognition. Here's a brief explanation of how the integration is achieved:

- First, I import the necessary libraries, including Optuna for hyperparameter optimization, Neptune AI for experiment tracking, and Keras for building and training the neural network model. I also import the `neptune.integrations.optuna` module, which facilitates the integration between Optuna and Neptune AI.
- Next, I initialize a Neptune AI run using `neptune.init_run()`, which creates a new experiment in my Neptune AI project. Then I also create a `NeptuneCallback` instance from `neptune.integrations.optuna`, which will handle the logging and tracking of hyperparameters and trial results during the Optuna optimization process.
- I then define an objective function `objective(trial)` for Optuna, where I specify the hyperparameters to be tuned, such as the number of filters in convolutional layers, the number of units in dense layers, the dropout rate, and the learning rate. Within this function, I build and train a CNN model using the specified hyperparameters on the MNIST dataset.
- After defining the objective function, I create an Optuna study using `optuna.create_study()` and optimize the objective function with `study.optimize(objective, n_trials=10, callbacks=[neptune_callback], n_jobs=4)`. This line initiates the hyperparameter optimization process, where Optuna will run 10 trials (training the model with different hyperparameter combinations) in parallel using 4 jobs (CPU cores). The `neptune_callback` is passed as a callback to log the hyperparameters and trial results to Neptune AI.
- Once the optimization is complete, I retrieve the best hyperparameters from the study using `study.best_params` and log them to Neptune AI using `run["best_params"] = best_params`. I then build a new CNN model using the best hyperparameters and train it on the MNIST dataset, logging the training metrics to Neptune AI using the `NeptuneLogger` callback.
- Finally, I evaluate the trained model on the test set, log the final test loss and accuracy to Neptune AI using `run["test/loss"]` and `run["test/accuracy"]`, save the trained model to disk, and stop the Neptune AI run using `run.stop()`.

Throughout the process, Neptune AI tracks and logs all the hyperparameters, trial results, training metrics, and final model performance, allowing us to visualize and analyze the optimization process and model performance within the Neptune AI platform.

6 Work Right Ahead

1. Utilize my learning from the Supervised Learning (SL) exercise to train a neural network to learn and replicate the output of a Fabry-Perot cavity as generated by Finesse simulation for maxtem 0. Further, upgrade the code to train it for higher-order modes and train it particularly for the parameters of the 40m lab.

2. Use the Neptune and Optuna integration to track and log the runs and find the best parameters to accomplish the task.

3. Generate data using Finesse simulations to generate data to train the models.

4. Read the article by Andreas Freise further and complete the sections regarding length sensing in interferometers and its feedback control systems, along with theoretical derivations and understanding of cavity eigenmodes and beam shapes, and all (beyond the plane wave approximation).

7 References

1. Finesse Python API Documentation

2. Finesse Documentation

3. Bond, C., Brown, D., Freise, A. et al. Interferometer techniques for gravitational-wave detection. Living Rev Relativ 19, 3 (2016).

4. CaltechExperimentalGravity/RL-Cavity-LockAcquisition