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

Technical Note	**LIGO-T2400293-v1-**	2024/10/15

Reinforcement Learning for Lock Acquisition

Kenny Moc

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 Introduction

In 1915, Einstein published his general theory of relativity [1]. He conjectured that gravity is the result of masses curving spacetime, and as a consequence, massive objects moving in spacetime produce gravitational waves. The propagation of these gravitational waves offers a new way to observe the universe As unlike electromagnetic radiation, gravitational waves interact very weakly with matter, and can travel through the universe almost unimpeded. It wasn't until 2015, a full century later, that the first direct detection of gravitational waves was achieved by the Laser Interferometer Gravitational-Wave Observatory (LIGO) [2].

LIGO uses modified Michelson interferometers to detect gravitational waves. The light in the interferometers are in antiphase, and without any disturbances, destructively interfere. The expansion and contraction of spacetime by gravitational waves causes microscopic changes in the length of the detectors, resulting in the light being out of antiphase. As a result, the light will no longer interfere, and this change in length can be measured by a power detector. However, this concept is only possible if the mirrors of the interferometer are perfectly still, but the mirrors are subject to seismic, thermal, and other sources of noise. Therefore, the crux of this project lies in the feedback control of these mirrors, to build robustness against noise.

2 Background

2.1 Interferometer Topology

In this report, we focus on the locking of the Power-Recycled Michelson Interferometer (PRMI) configuration of the 40m interferometer. This setup corresponds to the case where the End Test Mass X (ETMX), End Test Mass Y (ETMY), and the Signal Recycling Mirror (SRM) optics are intentionally misaligned, preventing these optics from forming resonant cavities. We define degrees of freedom that we are concrened with, these lengths are: the Michelson length (MICH), which is the difference in the lengths from the beam splitter (BS) to each input test mass (ITM); and the power recycling cavity length (PRCL), which is the average length between the power recycling mirror (PRM) and the ITMs.

The four main ports used to pick off signals and control the degrees of freedom (DOFs) of the interferometer are the reflection port (REFL), the pick-off port in the power recycling cavity (POP), and the anti-symmetric port (AS). The REFL port is located at the input side of the power recycling mirror (PRM), capturing light that is reflected from the interferometer before it enters the power recycling cavity. The POP port is located within the power recycling cavity, as a proxy for the circulating power in the cavity. Finally, the AS port is located at the dark fringe of the Michelson interferometer, collecting light that exits asymmetrically due to any imbalance in the arms.

Before the light enters the cavity, it is phase-modulated to enable the use of the Pound-Drever-Hall (PDH) technique [3]. This modulation introduces sidebands to the beam, which interact differently with the resonant cavities of the interferometer. The carrier frequency is mixed with the sidebands to generate error signals, which are used to control the degrees of freedom (DOFs) of the interferometer, such as the Michelson length (MICH) and the power

recycling cavity length (PRCL). These error signals, called the PDH error signals, are also observed in the detector.

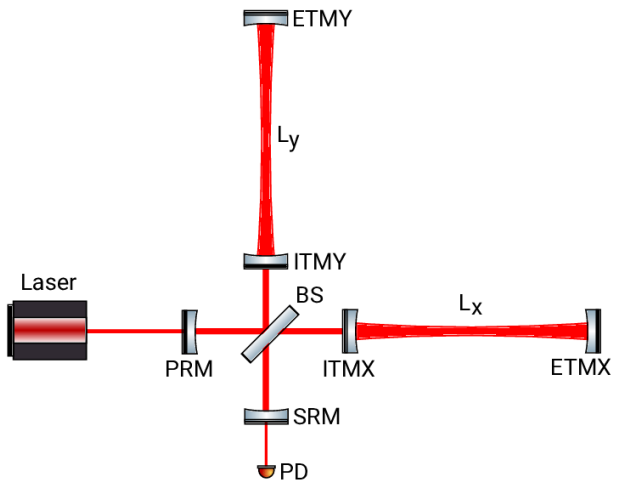

Figure 1: A simplified illustration of the Dual-Recycled Michelson Interferometer (DRMI). PRMI is the DRMI with ETMY, EMTX, and SRM are misaligned. Image courtesy of [4]

.

2.2 Lock Acquisition

The optics of the 40m interferometer, like those at the LIGO sites, are suspended on pendulums. When not actively controlled, these optics shift due to seismic and thermal fluctuations. This leads to the cavities moving out of resonance. In order to use the interformeter as a precision instrument, one must lock the interferometer onto a resonance condition. This process is known as lock acquisition.

Lock acquisition involves multiple steps. In this report, we focus on length sensing and control of PRMI. The PRMI setup is particularly useful for fine-tuning the alignment of the Power Recycling Mirror (PRM) optic in preparation for locking more complex configurations. We can achieve locking so that either the carrier or sideband field is resonant within the Power Recycling Cavity (PRC). For our work with reinforcement learning, we emphasize maximizing POP, which acts as a proxy for intracavity buildup. Thus, we are locking onto the carrier field.

2.3 Proximal Policy Optimization

The reinforcement learning algorithm we use is Proximal Policy Optimization (PPO). This algorithm is an on-policy method, meaning it directly optimizes the current policy employed by the reinforcement learning (RL) agent. The agent operates according to its policy, explores the environment, and seeks to improve its policy based on the rewards gathered from its actions. On-policy algorithms tend to be more stable and reliable because they frequently sample more optimal actions, allowing for faster convergence.

In our lock acquisition problem, the stability and reliability of PPO are particularly valuable. The faster convergence enables us to prototype and refine our ideas more efficiently. In contrast, off-policy algorithms do not optimize the current policy directly. Instead, they allow the agent to learn from data generated by other policies, significantly enhancing sampling efficiency. This is especially beneficial for our lock acquisition problem, where using Finesse 3 can be computationally intensive. By improving sampling efficiency, we can reduce the frequency of calls to Finesse 3. However, off-policy methods often converge more slowly and can introduce instability, leading to high variance, divergence, and other undesirable effects. Therefore, our efforts have mostly focused on on-policy algorithms.

The main outline of PPO is as follows [5]:

- Interact with the environment to gather experience. Recording state, action, reward, next state, and probability of action under current policy.
- Calculate the advantage, A, which measures how much better a specific action is compared to the average action taken in a given state. Formally, it is defined as:

$$A(s,a) = Q(s,a) - V(s) \tag{1}$$

  where A represents the advantage, Q is the expected cumulative reward for taking action a in state s, and V denotes the expected cumulative reward for being in state s, independent of the action taken. The advantage function thus indicates how much more effective action a is in state s compared to the average action. In PPO, we estimate Q using our rewards gathered, while a critic neural network is used to estimate V.
- Update the policy. The defining feature of PPO is the assumption that updates to the policy should not stray too far from the previous policy, which increases stability. There are two primary ways to make sure updates are small: penalty and clipping. In our implementation, we utilize clipping. Therefore, we aim to maximize A, while clipping to keep our changes within a specific range. Consequently, our objective function[1] is expressed as:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}\left[\min\left(r(\theta)A(s,a), \begin{cases}(1-\epsilon)A(s,a) & \text{if } A \geq 0\\ (1+\epsilon)A(s,a) & \text{if } A < 0\end{cases}\right)\right] \tag{2}$$

  where θ parameterizes the policy π_θ, ϵ is a hyperparameter, and $r(\theta)$ is defined as:

$$r(\theta) = \frac{\pi_{\theta_{\text{new}}}(a \mid s)}{\pi_{\theta_{\text{old}}}(a \mid s)} \tag{3}$$

  is the probability ratio. It is calculated between the current policy being updated, and the policy used to collect the trajectories. It estimates the divergence between the policies.

  To understand Equation 2, we can consider the two cases, if Advantage is positive or negative.

[1] Notation for the objective function may vary; in our report, we use L to represent the objective function, while other texts may use different symbols.

If $A \geq 0$:

$$L = \mathbb{E}\left(\min\left[r(\theta), (1+\epsilon)\right] A\right). \tag{4}$$

Because advantage is positive, the objective function increases if the action becomes more likely. Meaning, L increases if $\pi_\theta(a|s)$ increases. However, this is clipped by the *min* function. Once $\pi_\theta(a|s) > (1+\epsilon)\pi_\theta(a|s)$, it gets clipped, so that it doesn't go too far from the old policy.

Similarly, if $A < 0$:

$$L = \mathbb{E}\left(\max\left[r(\theta), (1-\epsilon)\right] A\right) \tag{5}$$

Because advantage is negative, the objective function will increase if the probability of the action becomes less likely. The *max* function in this case puts a limit to how much the objective function can increase. Once $\pi_\theta(a|s) < (1+\epsilon)\pi_\theta(a|s)$, it gets clipped and we hit a ceiling. Again, this means the new policy does not benefit by going too far away from the old policy.

Overall, we update the policy by maximizing this objective function. This is usually done with some gradient ascent.

- Finally, PPO updates the value function, V, estimator. In our case, our value function is estimated by a critic neural network. We do gradient descent to improve on it.

Proximal Policy Optimization (PPO) generally operates under the (weak) assumption of a Markovian decision process. However, this assumption does not hold in our specific problem. In this report, I will present two cases: the static case, where a Markov decision process is applicable, and the non-static case, where this assumption fails. It has been shown that even in simple problems, such as the single inverted pendulum, masking the velocity can lead PPO to not converge. This is because the system does not conform to the Markov decision process framework. Therefore, since our interferometer is not a Markov decision process, we need ways to derive velocity from the system, to make it one.

2.4 Convolutional Neural Networks

In our problem, we only observe power detector outputs, so we need a method to derive velocity from these instantaneous observations. To achieve this, we use a convolutional neural network (CNN) combined with frame stacking to extract velocity-related features from the environment. As we collect observations (such as power detector and PDH error signals), we stack these observations over time. This stacked sequence is then passed to a feature extractor, which is implemented using a CNN.

The feature extractor analyzes the stacked observations, creating a vector that incorporates information from neighboring time steps. This allows the reinforcement learning agent to infer velocity from a sequence of data points and incorporate it into its decision-making process. The feature extractor proceeds as follows:

- **Frame Stacking**: We first stack multiple sequential observations of the system, such as power detector (PD) outputs and PDH error signals, each instance observed is called a frame.

- **1D Convolution**: The feature extractor applies a 1D convolutional layer to convolve neighboring datapoints in the time series.
- **Activation Function**: To introduce non-linearities into the network, the convolution output passes through an activation function. In our case, we use the ReLU function.
- **Pooling**: After applying the activation function, we pool to downsample the number of features.
- **Flattening**: The pooled output is then flattened into a 1D vector, which serves as our final output vector.
- **Feeding to RL Agent**: Finally, we feed both the original observations (PD and PDH signals) and the extracted features $\mathbf{f}_t$ to the reinforcement learning agent as inputs.

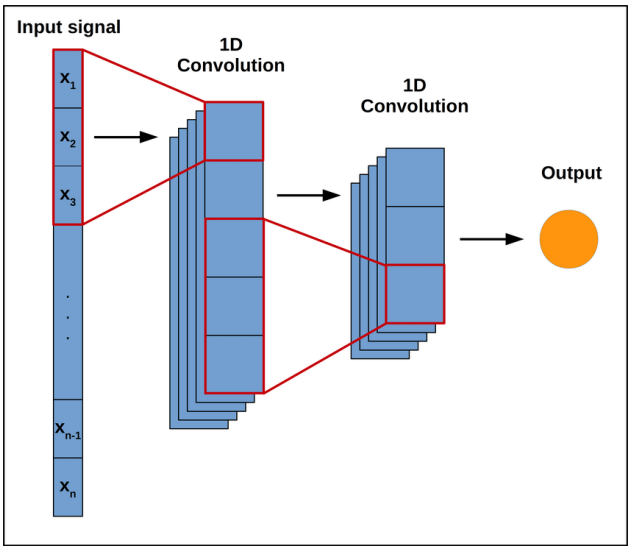

Figure 2: Depiction of a 1D convolution neural network architecture with two convolution layers. In our case, each $x_1, x_2..x_n$ is a vector of observations. The total number of x vectors, n, is the size of our frame stack. Image courtesy of [6]

.

2.5 FINESSE

FINESSE (Frequency domain INterferomEter Simulation SoftwarE) is a tool for simulating the optical behavior of laser interferometers. It numerically computes light field amplitudes within an optical system by using Hermite-Gaussian modes in the frequency domain. We use FINESSE to simulate the power detector outputs and PDH error signals.

In our work, we utilize the FINESSE 40m package [7], which is configured specifically for the 40-meter interferometer. This package is the standard FINESSE software configured with the parameters specific to the 40-meter interferometer. It provides the response of the interferometer to changes in its degrees of freedom, forming the basis for our environment. However, FINESSE is static by nature and does not emulate how the degrees of freedom

evolve over time. Therefore, while the package serves as a basis, we must develop additional code to model the changes in the degrees of freedom.

3 Approach

3.1 Time-Domain Simulation and FINESSE Integration

FINESSE provides the Power Detector (PD) and Pound-Drever-Hall (PDH) signals based on specific interferometer parameters, but it doesn't model how the interferometer dynamically responds to noise or external forces. To accurately simulate the interferometer's behavior, we need to account for how these noise forces affect the system over time. Additionally, we must model the impulse response to understand how the interferometer's degrees of freedom—MICH and PRCL—change under external forces.

To address these challenges, I developed a time-domain simulation of the Power-Recycled Michelson Interferometer (PRMI). The key steps involved were as follows:

- **Interferometer Modeling:** I modeled the interferometer's key optics, placing them on single pendulums. These optics were evolved over time using Runge-Kutta 4, which accounts for the forces acting on them. Each optic has a resonant frequency of 1 Hz. As the optics swing, the simulation computes the relative distances between them, which directly impact the interferometer's degrees of freedom, MICH and PRCL. Notably, only the Power Recycling Mirror and the Beamsplitter are simulated, as all other optics are kept stationary for simplicity.

- **Noise Forces:** To emulate the noise forces acting on the PRM and BS, I first locked the interferometer and then analyzed the control signals sent to maintain this locked state. These control signals, responsible for counteracting environmental noise, serve as proxies for the noise forces in the system. The signals obtained were in units of floating-point counts, which required calibration to actual forces. Additionally, these calibrated forces had to be converted from forces acting on the interferometer's degrees of freedom (MICH and PRCL) to forces acting on the individual optics (PRM and BS). The simulation also adds random phases to the inverse fourier transform of the power spectrum to generate unique noise series each run.

- **Maximum Force Calibration:** We also calibrated the maximum applicable force to correspond to the maximum force that can be exerted on the real optics.

- **FINESSE Integration:** At each point in the simulation, noise forces and applied forces cause the optics' positions to shift, which, in turn, affects the interferometer's degrees of freedom. These changes were fed back into FINESSE to update the PD and PDH signals accordingly. At each step, FINESSE computed the new optical signals based on the updated interferometer parameters.

Once we established the pendulum model, incorporated noise forces, and calibrated the maximum forces applicable to the mirrors, we created a basic simulation of the interferometer.

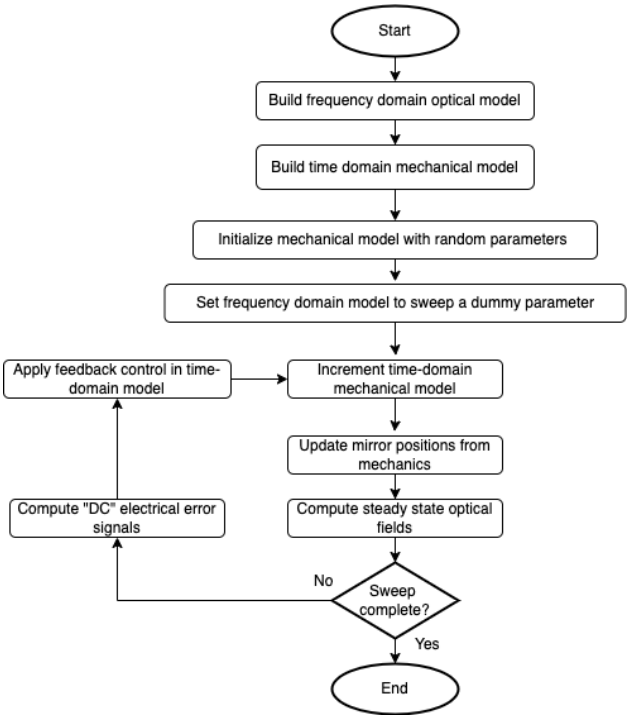

Figure 3: A flowchart depicting the logic of the time-domain simulation [8].

However, this model has its limitations. Specifically, it only simulates the free-swinging motion of well-aligned optics. In reality, we will get offsets and our beam becomes misaligned with the optics, but these are not included in the model. Therefore, this simulation is only valid for the well-aligned, uncontrolled, free-swinging state.

Beyond this, we needed to improve the simulation speed. Each time we adjust the interferometer, FINESSE performs a full beam trace, which involves matrix inversion and can be quite slow. Since our simulation doesn't introduce macroscopic changes at any step, performing a full beam trace every time is unnecessary. By introducing optimizations, I was able to achieve a **two-order-of-magnitude speedup** in the simulation. To achieve this, I did the following:

- **Custom FINESSE Action:** FINESSE actions are performed before or after scanning a variable. We create a custom FINESSE action to change our variables of interest directly, while scanning a dummy variable. By doing so, we bypass the time-consuming beam tracing and pre-checkups that are normally required. Whereas the *run()* method needs to perform a full beam trace each time.
- **Overwriting the *_requests()* Method:** To update degrees of freedom such as PRCL and MICH, we had to modify the action's internal *_requests()* method. Normally, FINESSE actions don't allow for direct changes to these degrees of freedom, so this rewrite was necessary.
- **Overwriting the *_do()* Method:** Similarly, we needed to modify the internal *_do()* method to ensure that PRCL and MICH could change during the simulation, even within a custom FINESSE action.

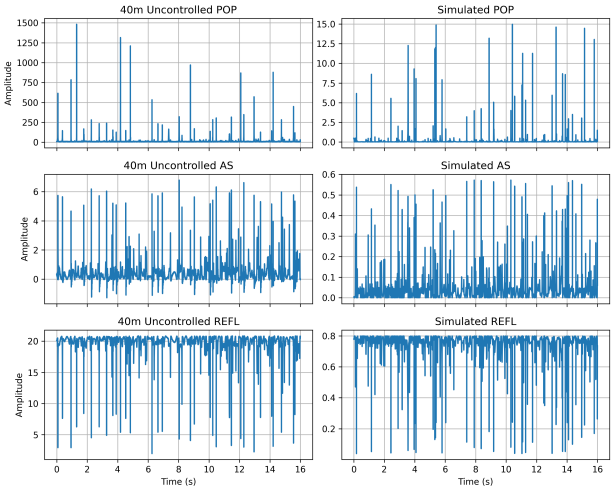

Figure 4: The simulated flashing of our PDs closely resemble real-world data measured at the 40m.

- **Dummy Variable for Iteration:** FINESSE actions are typically called while scanning a variable. We introduced a dummy variable to scan over, triggering our custom action at each post-step.

By applying these optimizations, we successfully created a time-domain simulation of the PRMI that operates at approximately 1/16th of real-time. That is, it takes 16 seconds to generate one second of real-life data.

3.2 Static Simulation

We first worked with a static simulation of the PRMI as a toy model. In this scenario, there are no mechanical pendulums involved; the agent interacts directly with FINESSE as a frequency-domain simulation. Additionally, noise forces are absent, and the agent can instantaneously adjust the interferometer's degrees of freedom without needing to account for any applied forces.

This static model serves as a proof of concept, demonstrating that it is indeed possible for an agent to use the PDH and PD signals to guide the interferometer into a locked state.

4 Results

4.1 Static Locking

The agent is capable of bringing the static PRMI into lock, albeit not consistently, but with enough frequency to demonstrate proof of concept. In this case, we utilize multi-agent reinforcement learning, where each agent is responsible for controlling a specific degree of freedom. The agents are rewarded based on if they can maintain the interferometer within a few nanometers of the resonance condition.

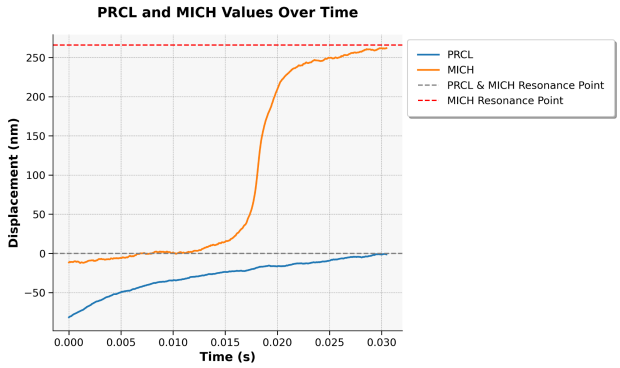

Figure 5: One episode where the agent is able to bring the static interferometer to lock.

Since this scenario involves no velocities or dynamic parameters, it can be modeled as a Markov Decision Process, where each agent only requires the current observation to make decisions. Consequently, we do not implement a CNN-based feature extractor in this instance.

Furthermore, we exploit the static nature of the problem by training each agent independently. During testing, the agents can be called consecutively, allowing them to work together to achieve the locking condition.

4.2 Linear Controllers

The time-domain simulation has also proven useful for testing linear controllers. In one demonstration, I implemented a PID controller to achieve lock. The controller is activated when the POP signal flashes above approximately 30% of its maximum value, indicating that the system is entering the linear region of the error signal. Once inside this region, the controller works to acquire lock as the pendulum swings through.

In these tests, I allowed the PRM to swing freely while keeping the BS and ITMs stationary. The controller successfully stabilized the PRCL cavity starting from a random initial position.

This simulation allows the opportunity to optimize the gains of our linear controllers—something that was previously unachievable. Now, we can rigorously test and fine-tune the controller gains for improved performance.

4.3 Reinforcement Learning for Lock Acquisition

Presently, reinforcement learning for lock acquisition of the time-dependent case remains a work in progress. The agent has not yet successfully locked the interferometer, as it struggles to converge and fails to learn from the environment. To address this, we plan to further explore reward shaping and curriculum learning. However, as of now, the agent has yet to lock.

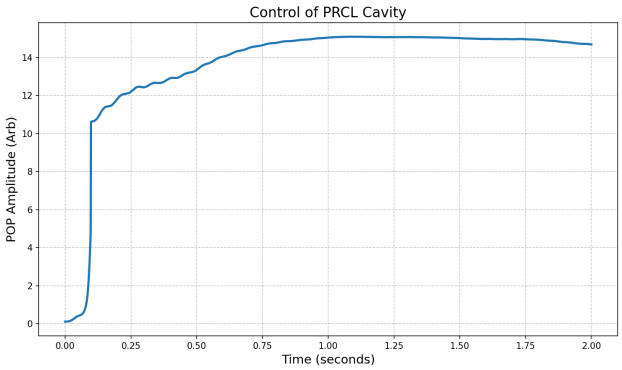

Figure 6: Power at POP With PID Controller. This demonstrates lock acquisition of the PRCL activity with all other optics held still.

5 Discussion

Over the course of the summer, we have developed a time-domain simulation of the PRMI, demonstrated locking of the static interferometer, and began investigations in locking the time-domain simulation. This simulation also opens up new opportunities for optimizing linear controller gains, allowing us to fine-tune our current control systems.

Despite this progress, we have yet to lock the interferometer using reinforcement learning. Key difficulties include expensive sampling, large state-action space, complicated dynamics, non-markovian processes, and sparse rewards. These combined have made it difficult for the agent to converge.

Looking ahead, we are experimenting with reward shaping, curriculum learning, and other RL algorithms in trying to overcome these challenges. Concurrently, we are attempting to improve our current linear controllers using the time-domain simulation.

6 Other Efforts

During the summer, I also worked on continuously monitoring the Unity Gain Frequency (UGF) of various control loops at the 40-meter interferometer. My efforts involved programming a Moku:GO to send band-limited Gaussian noise through one channel while simultaneously collecting the same noise after it passed through the control loop. I then analyzed the power spectral densities to identify the frequencies at which unity gain occurs. This approach enables continuous monitoring of control loops at the 40m without the need for sweeping sine waves, which could inadvertently excite a resonance condition and disrupt the interferometer's lock.

7 Acknowledgements

This work was made possible by the supervision of my mentor, Rana Adhikari.

I am also grateful for the mentorship and fruitful discussion provided by R. Bhatt, J. C. Sanches , F. Carcoba, A. Prakash, A. Goodwin-Jones, and K. Arai.

Lastly, I must thank the National Science Foundation (NSF), the LIGO Scientific Collaboration, and the California Institute of Technology Summer Undergraduate Research Fellowship (SURF) program for funding this research.

References

[1] A. Einstein, *The Foundation of the General Theory of Relativity*, Annalen der Physik, vol. 49, pp. 769-822 (1916).

[2] B. P. Abbott et al., *Observation of Gravitational Waves from a Binary Black Hole Merger*, Physical Review Letters, vol. 116, 061102 (2016).

[3] R. W. P. Drever, J. L. Hall, F. V. Kowalski, J. Hough, G. M. Ford, A. J. Munley, and H. Ward, *Laser Phase and Frequency Stabilization Using an Optical Resonator*, Applied Physics B, vol. 31, no. 2, pp. 97-105 (1983). DOI: 10.1007/BF00702605.

[4] H. Grote and Y. Stadnik, *Novel Signatures of Dark Matter in Laser-Interferometric Gravitational-Wave Detectors*, arXiv:1906.06193 (2019). https://doi.org/10.48550/arXiv.1906.06193

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal Policy Optimization Algorithms*, arXiv:1707.06347. https://arxiv.org/abs/1707.06347

[6] R. Pérez and A. Fernández, *Simple 1D Convolutional Neural Network (CNN) Architecture with Two Convolutional Layers*, ResearchGate. https://www.researchgate.net/figure/Simple-1D-convolutional-neural-network-CNN-architecture-with-two-convolutional-layer fig1_344229502

[7] Finesse 40m Repository, https://git.ligo.org/finesse/finesse-40m.

[8] Finesse Documentation, *Time Domain Simulation Examples*. https://finesse.docs.ligo.org/finesse-40m/examples/time_domain.html