Technical Note **LIGO-T2000291–v1** 2020/11/08

LIGO SURF Final Report

Ella Buehner Gattis, Mentors: *Rana Adhikari, Koji Arai, Tega Edo*

1 Abstract

Seismic movement is a source of noise in LIGO detector readings. Seismometers stationed around the detector monitor seismic activity. Temperature fluctuations add noise to the seismometer readings. The aim of this project was to design a control system to maintain a stable temperature within the seismometer enclosure, thus eliminating noise from the seismic data so that seismic noise can be removed from LIGO data. Control systems can be divided into two categories; linear and nonlinear control systems. Linear control systems obey the principle of superposition and are governed by linear differential equations. Nonlinear control systems do not obey the principle of superposition and are governed by nonlinear differential equations. The objective of this project was to design a nonlinear control system using a convolutional neural network controller to improve disturbance rejection and decrease overshoot/undershoot in comparison to a linear control system. The neural network was trained by reinforcement learning, using a custom OpenAI gym environment to train the network for temperature stabilization. The network was built with RLzoo's framework, which uses the tensorflow machine learning library. The end goal is then to evaluate the performance of the NN controller and compare it to PID control in terms of overall efficiency. Neural network control may later replace linear control across other stabilization problems in LIGO to further reduce noise.

2 Introduction

The Laser Interferometry Gravitational Wave Observatory (LIGO) aims to detect gravitational waves predicted by Einstein's theory of relativity in 1916, which predicted that gravitational waves would cause ripples in space-time, which can be measured as contractions and expansions in space [1]. LIGO consists of two L-shaped detectors, with arms 4km in length. The basic concept they employ to detect changes in length caused by ripples from gravitational waves is the same as that for a Michelson interferometer. Each arm of the detector is an exact integer number of wavelengths long. Monochromatic light travels down each detector arm and is reflected by the mirror at the end. Because the arm is an integer number of wavelengths long, the reflected beam is exactly in antiphase with the initial beam and no light returns to the detector. As such, any microscopic change in length of the LIGO detector arm will cause the transmitted and reflected beam to no longer be in antiphase, and the resulting change in length can then be measured by the detector.

However, gravitational waves are not the only source of movement in the detector arm length. Seismic activity also causes movement in the detector arms, introducing noise into the detector readings [2]. It is important to understand and measure noise sources in the detector and its readings for LIGO data to be useful. We aim to create a heat map of seismic activity in the area surrounding LIGO to better understand and account for seismic activity in detector readings.

Seismometers measure ground movement and are usually placed 2 meters below the surface, where temperature is steady with very few or negligible fluctuations. However, the seismometers around LIGO are placed above ground in order to measure seismic activity at the surface. Because of this, they are subject to above-ground temperature fluctuations, which affect the seismometer readings and introduce noise into the seismic data. In order to minimise the effect of temperature fluctuations on seismometer readings, the seismometer is placed in a steel enclosure surrounded by a resistive heater and insulating foam.

The focus of this project was to develop a feedback control system using an artificial neural network to ensure temperature stability within the seismometer enclosure. The network was built using the RLzoo framework for reinforcement learning algorithms, and trained using a custom environment built to function as part of the OpenAI gym.

3 Objectives

The aim of this project was to design a feedback system that will keep the temperature within the seismometer enclosure constant, to ensure there is as little noise in the seismic data due to temperature fluctuations as possible. The first step was to design a model of the system, and test the system response to different changes in temperature.

Once an accurate model of the system had been created, it was possible to test the response of the system with linear control, such as PID (proportional-integral-derivative) control. In linear control, the output signal is proportional to the input signal supplied, and the system is governed by linear differential equations, which limits the flexibility of control that can be implemented to only be linear. Real control systems are nonlinear, and are governed by nonlinear differential equations.

The main focus of my project is to test nonlinear feedback systems - in particular, a machine learning mechanism using a neural network - for control. In nonlinear control the output signal is not necessarily proportional to the input signal. The aim of using nonlinear control here is to further reduce noise from temperature fluctuation by minimising overshoot and improving the system response to disturbances, which is sometimes a problem with linear control methods such as PID [3], and to reduce the settling time.

The final objective is to test the efficacy of a linear PID control system and compare it to that of nonlinear neural network temperature control. Our overall aim is to design a feedback system that can achieve and maintain temperature stability to within the sensitivity of the temperature sensor being used.

4 Methods

The first step of this project was to model the seismometer and stainless steel enclosure as a system in Python, using the heat equation to model temperature changes as a function of applied power. This modelling was done in the Laplace domain, and also in the time domain to model nonlinearities.

Once the first step of modelling the system was done, it was be possible to also model the effect of PID for temperature control, and to evaluate different methods for tuning the PID parameters.

The next step was to set up an environment in which to train a neural network for temperature control.

4.1 Modelling the system

The physical system consists of a seismometer in a stainless steel enclosure wrapped in resistive heating mesh with a resistance of 30 Ohms, covered with an insulating foam. The temperature sensor is an AD590 temperature sensor. Figure 1 shows the physical setup of the enclosure.

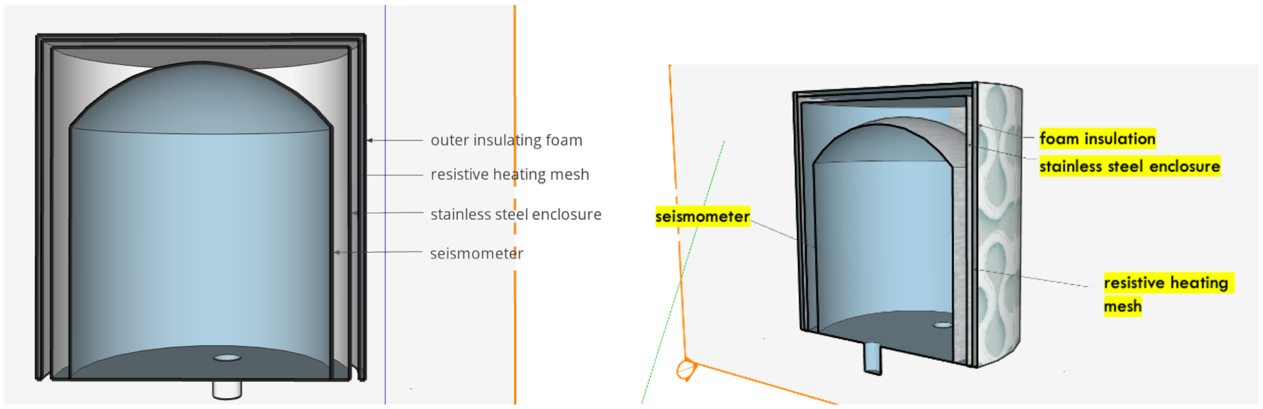


Figure 1: Diagram of seismometer enclosure setup

A pulse-width modulation (PWM) circuit is used to control the power supplied to the heating system using a MOSFET to control current flow to the resistive heater.

The physical system can then be modelled by using the heat equation:

$$P_{in} = M_{ss}C_{ss}\frac{dT_{ss}}{dt} + P_{loss} \tag{1}$$

where P_{in} is the input power supplied as heat, M_{ss} is the mass of stainless steel, C_{ss} its heat capacity and T_{ss} its temperature at a given time t. P_{loss} is the power lost through conduction and radiation.

The power lost by conduction will be

$$P_{\text{conductive loss}} = \frac{k_{foam}A_{foam}}{t_{foam}}(T_{ss} - T_{amb}) \tag{2}$$

where k_{foam} is the thermal conductivity of the insulating foam, A_{foam} is the outward facing surface area of foam, andt_{foam} the thickness. The difference in temperature between the stainless steel enclosure and the ambient temperature of its surroundings is given by $(T_{ss} - T_{amb})$. Then, the power lost by radiation is given according to the Stefan-Boltzmann law by

$$P_{\text{radiative loss}} = \sigma A_{foam}(T_{ss}^4 - T_{amb}^4) \tag{3}$$

This system was modelled in Python.

To model the nonlinearities such as radiative losses and other noise, it was necessary to model the system in the time domain rather than in the Laplace domain. Neural network training is also done in the time domain, which means it can also account for radiative losses.

4.2 Linear control- PID

In PID control, there are 3 parameters to be tuned, according to the PID equation:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t) \tag{4}$$

There are a number of different methods with which to tune the PID parameters, which have been investigated and evaluated in Rushabha's project.

4.3 Nonlinear control using a Neural Network

The role of the artificial neural network is to function as a nonlinear controller. The neural network will take in measurements of the seismometer temperature and ambient temperature and regulate the duty cycle of the PWM circuit, thus controlling the power to the heating circuit to maintain temperature stability. Nonlinear control may be more favourable and outperform linear control in terms of reducing settling time, rejecting disturbances, and minimising overshoot [5].

A neural network consists of multiple individual neurons arranged in layers to form a dense network. Each neuron receives inputs multiplied by a weight. The output of the neuron is then the sum of the input values multiplied by their weights. This is illustrated in the figure below. The connections between neurons and inputs are initially randomly assigned, and are adjusted in the process of training.

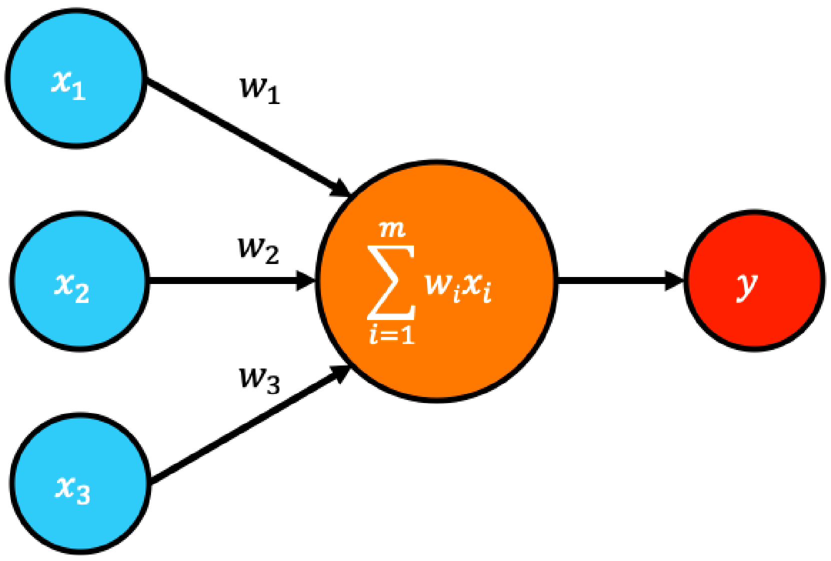


Figure 2: Diagram of the inputs and output of a single neuron

4.3.1 Software

I am developing the neural network for temperature control in Python using TensorFlow, an existing open-source machine learning library distributed by Google. I am using the RLzoo framework for reinforcement learning algorithms to build the network. The OpenAI gym forms the basis for the learning environment of the neural network.

4.3.2 Supervised Learning

Supervised learning uses a given set of labelled data to learn a function that maps the input onto the output without changing the input space. The neural network then learns to adjust its weights and biases for a given set of input variables to get the desired output. This means supervised learning needs a labelled set of data for input and corresponding output to train. The training process for supervised learning is illustrated in Figure 3.

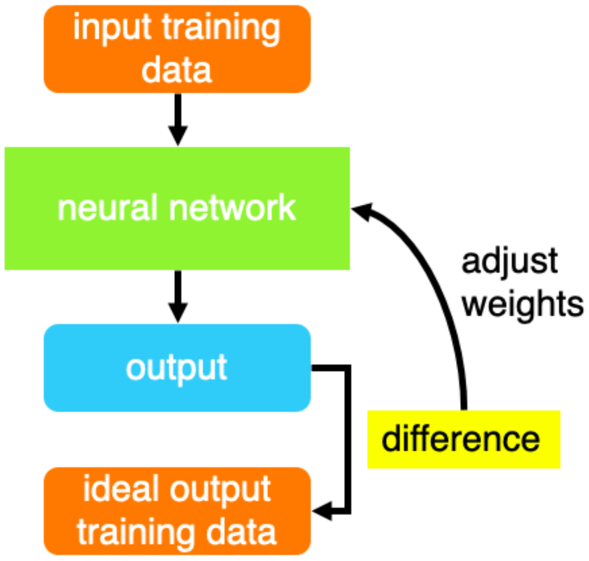


Figure 3: Diagram of supervised learning training for a neural network

4.3.3 Reinforcement Learning

Reinforcement Learning, in contrast to Supervised Learning, doesn't need a labelled data set, and uses trial and error with game-like scenarios operating in discrete time steps to find optimum control. RL interacts directly with its environment to create its own data, meaning it requires environments such as those in the openAI gym for training. The network interacts with the environment by taking an action which moves the environment into a different state, for which a reward is defined. The reward measures how "good" or "bad" a particular action is by evaluating how effective it is in bringing the environment into the desired final state. The network uses the feedback from the reward to learn. This process is illustrated in Figure 4.

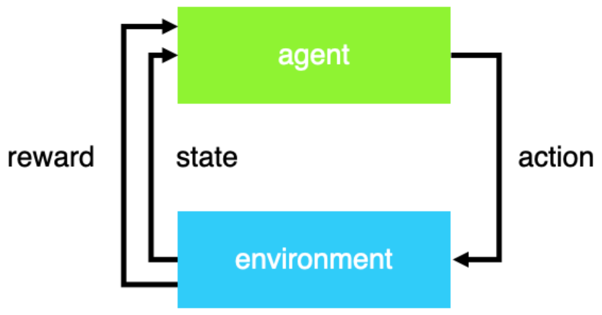


Figure 4: Diagram of reinforcement learning training for a neural network

A Reinforcement Learning model takes much longer to train than Supervised Learning, so I am using SSH to use a LIGO computer to run the programs and train the network more quickly.

Because this project is focused on control, the neural network has to interact with the input space (the temperature of the enclosure) to adjust its parameters to optimise the output (power supplied to the heating circuit). This means that this network will use Reinforcement Learning for training. There are a number of different algorithms for reinforcement learning. In this case, the input (temperature of the enclosure) and the output (power supplied to the heater) are both continuous variables, so we need a Reinforcement Learning algorithm that has a continuous input and output space. For this reason, we decided on actor critic as the most suitable training algorithm.

4.3.4 Actor Critic training

Actor critic networks consist of two networks which train in parallel; the actor chooses which action to take, and the critic evaluates how good or bad the action is. The actor uses policy based learning, wherein the input is the observation of the current state, and the output is the action that the actor takes. The role of the policy network (the ”actor”) is to map states to actions. The value function (the ”critic”) finds the expected reward for a particular state. The value function takes as input the observation of the current state and the action taken, and produces as output a Q-value which defines the reward associated with that state to quantify how good a particular action was.

Each step in a training episode produces the next state and its associated reward, so as more steps are taken both the actor and the critic learn and improve in their roles. By merging the concepts of policy-based and value-based algorithms, actor critic algorithms can reduce the total training time and outperform each of the separate networks individually.

The RLzoo framework provides a number of actor critic training algorithms, of which we decided Soft Actor Critic (SAC) was the most suitable. Soft Actor Critic algorithms aim to maximise entropy, and thus the randomness of actions taken. This means that the network will adapt to new situations better, and is less sensitive to hyperparameters, thus reducing the time needed for hyperparameter tuning [6].

5 Progress

The first step was to design a model of the system in Jupyter notebooks - this is the foundation for all subsequent things. This has been done by Rushabha in the weeks before I began my project. His project also developed PID tuning methods and modelled the system response to changes in temperature.

The first task in my project was to set up a Supervised Learning convolutional neural network to learn the behaviour of a simple electronic system - fitting the frequency response of an LC filter. This was done using the Keras Sequential model and a Conv1D layer. This formed a useful basis for better understanding signal processing and the principles of convolutional neural networks.

The learning set used was a large sample of noisy signals - random noise generated by the numpy.randn function - together with a filtered signal, which had been denoised using a digital impulse response filter applied using second order sections. The filter produced the smooth signal by convolving the impulse response (purple signal on the figure below) with the noisy input signal. The job of the network then was to take in a sample of noisy signal and produce a denoised signal, thus imitating the output of the filter. The noisy input, impulse response, and filtered signal are shown in Figure 5.

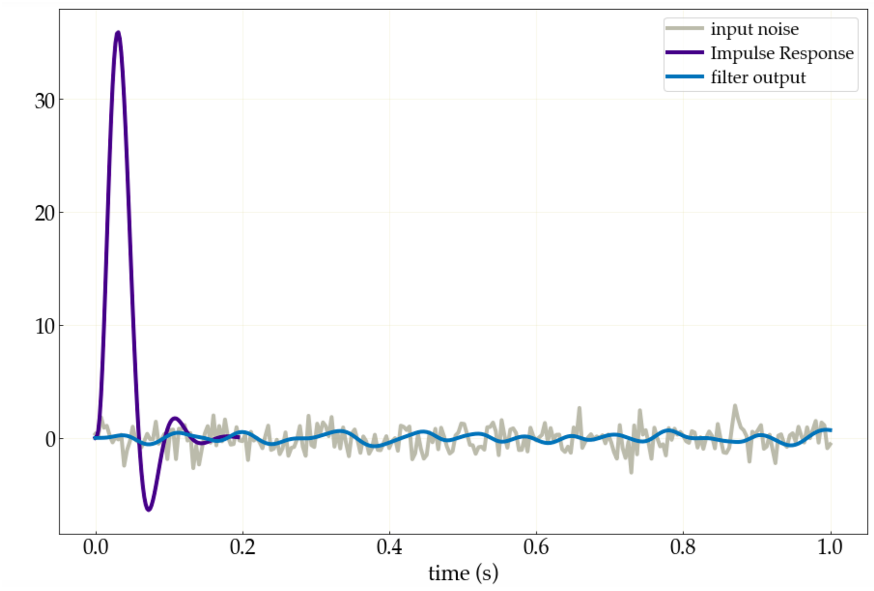


Figure 5: Graph of time series data of input, output, and impulse response of digital filter

Since the filter was applied by convolving the impulse response with the noisy signal, the network also needed to use convolution. Convolutional neural networks (CNNs) employ the same concept as convolution of signals, where the element-wise product of the convolutional filter and inputs is taken to produce the output. This process is illustrated in the figure below. At least one of the layers in a CNN uses a convolutional filter, which allows the network to search for patterns within the data.

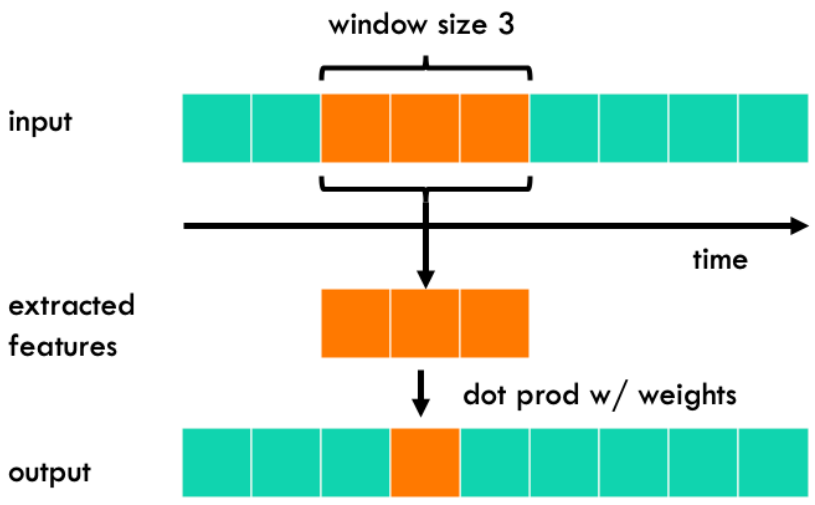
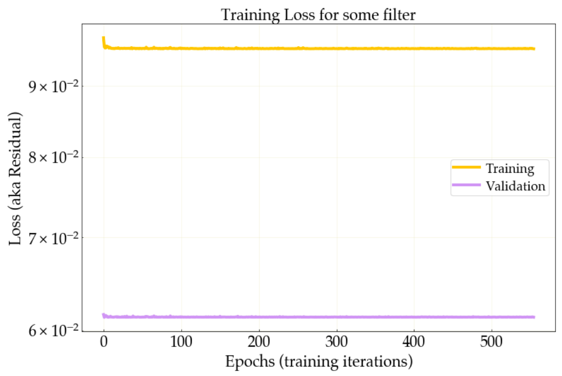
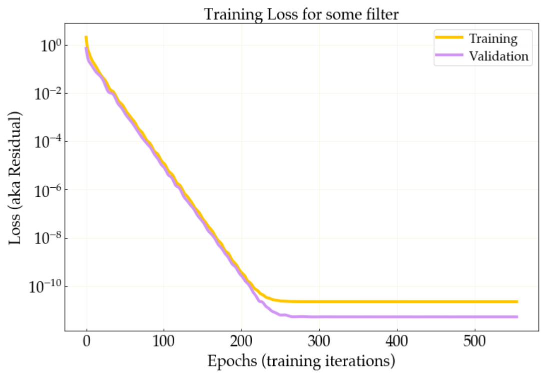


Figure 6: Diagram illustrating 1D convolution process

In order for the convolutional layer to work it was necessary to specify the input shape of the data correctly, which took some time to figure out. The figure below illustrates the training loss during learning for the neural network in the two scenarios for differently specifying the input shape. It is clear to see that in the first case (a), the neural network was not learning at all. This is because the way that the input shape had been specified meant that the network only read in one data point at a time, making it impossible to learn a pattern. The Keras Conv1D input layer requires the input shape to be specified in the format (batch size, number of steps, channels), as was done in the second case (b). Once this was correctly defined, it was possible for the neural network to learn to imitate the output of the filter.



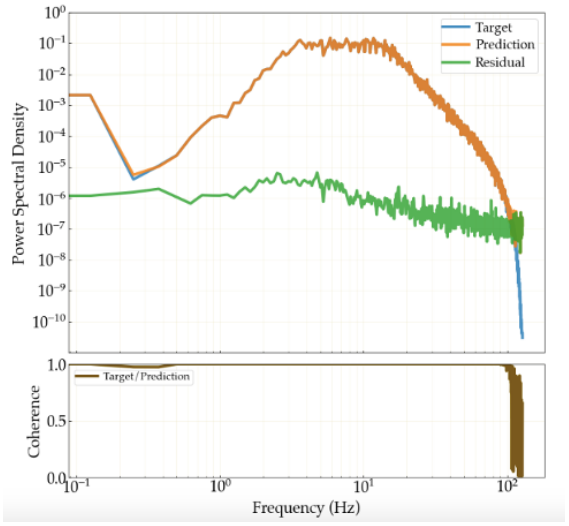
(a) Input shape specified in format (shape of data, 1, 1)



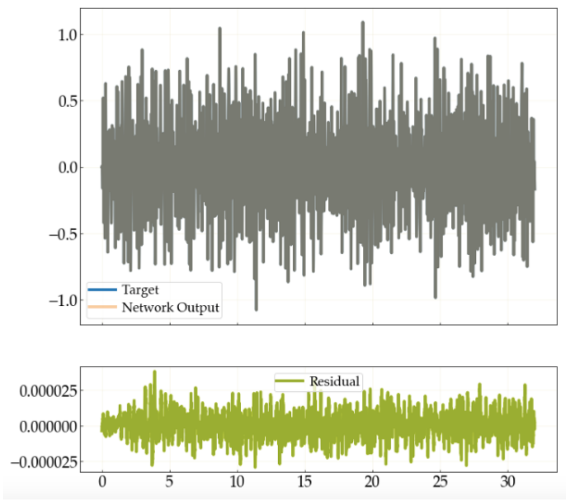
(b) Input shape specified in format (1, shape of data, 1)

Figure 7: Training loss for neural network over 500 epochs

The figures below show the output of the neural network in testing with new input data, as well as the residual between the target signal and the signal produced by the NN filter. Figure 8 (a) shows that the neural network is able to imitate the digital filter very accurately up to high frequencies (10^2 Hz), at which point the residual grows rapidly as the NN does not approximate the filter well. Figure 8 (b) shows the target signal in blue and NN output in orange- the graph appears grey because the two signals overlap almost completely, showing the network can imitate the response of the digital filter very well.



(a) Graph of power spectral density of input signal, neural network output, and residual



(b) Graph of target and neural network output, and residual

Figure 8: Signal and residual produced by neural network

The next step was to set up the neural network for temperature control, to keep the true temperature fluctuation of the seismometer enclosure close to zero.

To do this, I have adapted one of the classic control examples from the OpenAI gym, namely the problem of stabilising a pendulum, and substituted in the differential equations for our seismometer enclosure system. Since these are both stabilization problems, the training environment for seismometer temperature control is similar to that for the pendulum problem. First, I set up a more simple scenario with which to train the neural network. I did not include an ambient temperature fluctuation, but instead set the start temperature as a random point between 15°and 25°Celsius, and trained the network to try to converge on the temperature set point of 35°Celsius. Once the network has been trained to do this, it will be possible to build the complexity of the learning environment by including ambient temperature fluctuation and noise until it resembles the real system.

It is important for the neural network to recognise the overall pattern in ambient temperature fluctuation (which generally occurs over a 24 hour cycle), so the network must incorporate a convolution layer. The ambient temperature data has been collected using the nds2 channel to receive temperature data in one-week chunks. This will also form part of the learning environment. The neural network should then be able to recognise the patterns of temperature fluctuation and account for them ahead of time, thus minimising overshoot and increasing the accuracy of the temperature control. The network will also be capable of recognising the trend in temperature change of the seismometer and account for this in its control of the heater, in the style of feedforward control.

I am using the RLzoo framework for reinforcement learning to build the network using soft actor critic. Soft actor critic aims to maximise the entropy of the policy, thus injecting maximal noise into learning, to improve the response of the network to unexpected environmental disturbances, increase its sample efficiency, and reduce the need for hyperparameter tuning [4]. First, I am trying to optimise a network in the pendulum gym environment to understand the layer structure and syntax. Once this is done, I will import my custom gym environment for temperature control in order to use the RLzoo framework to build our temperature control network.

After using the RLzoo framework with my custom gym environment to train the network, it will be possible to export the model and test it within our simulation of the system to evaluate the performance for temperature control. We will have to evaluate the performance using the same cost function used to evaluate the the PID controller in order to accurately compare the two. In order to do this, I needed to create and implement a custom loss function based on the cost function of the PID controller. This calculated the integral squared error (ISE), given by equation (5), where e is the error, the peak overshoot of the system response, and the settling time. The goal of the PID controller and the neural network then was to minimise this cost function.

$$ISE = \int_0^t e^2 dt \tag{5}$$

Once simulations of temperature control using both PID and neural network control are running it will be possible to compare the efficacy of the two. The overall target control is to hold the temperature of the seismometer constant to within the accuracy of the temperature sensor of the enclosure. At this point, the effect of temperature fluctuations on the seismometer readings should be negligible, producing the maximum accuracy for results with this setup.

6 Conclusions

I have not yet been able to export the ANN model trained for temperature control in order to compare it to PID control. As things stand, the final reward of a training or testing run is around -1200. It is unclear to me whether this is because the network is not training properly, and some changes are required in my custom gym environment, or because I have defined the reward in a way that is not suitable.

7 Future Work

References

[1] Albert Einstein. *The Foundation of the General Theory of Relativity*, Annalen der Physik, volume 49, 10.1002/andp.200590044, 1916

[2] D M Macleod, et al. *Reducing the effect of seismic noise in LIGO searches by targeted veto generation*, Classical and Quantum Gravity, vol. 29, no. 5, 10.1088/0264-9381/29/5/055006, 2012,

[3] Yuvraj V. Parkale. *Comparison of ANN Controller and PID Controller for Industrial Water Bath Temperature Control System using MATLAB Environment*, International Journal of Computer Applications, volume 53, 10.5120/8390-1967, 2012

[4] Daniel Seita. *Soft Actor Critic-Deep Reinforcement Learning with Real-World Robots.*, The Berkeley Artificial Intelligence Research Blog, bair.berkeley.edu/blog/2018/12/14/sac/

[5] Jinchuan Zheng, et al. *Nonlinear Pid Control Of Linear Plants For Improved Disturbance Rejection.*, IFAC Proceedings Volumes, vol. 38, no. 1, pp. 281–286., 10.3182/20050703-6-cz-1902.01257. 2005

[6] Tuomas Haarnoja, et al. *Soft Actor-Critic Algorithms and Applications.*, arXiv 1812.05905 2018

8 Acknowledgements

I would like to thank my supervisors Rana Adhikari, Koji Arai, and Tega Edo for all their help and patience throughout. I would also like to thank Caltech's SURF team and the NSF for making this possible.