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

Technical Note	**LIGO-T2300283–v1**	2023/09/22

LIGO SURF 2023 Final Report

Implementing Nonlinear Control in a Classical Experiment to Reduce Measurement Noise

Andrei C. Diaconu, Advait Mehla

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/

1 Abstract

A central problem in control theory is that most of the field focuses on linear controllers, even though most of the systems we are aiming to control are nonlinear in nature. To circumvent this issue, control theory aims to approximate the behavior of the nonlinear system around the desired mode of operation by a linear function. This unfortunately creates a theoretical limit on the performance specification of the linear as it tries to control a nonlinear system with a linear control law. We aim to show that this limitation can be overcome with a nonlinear controller based on Reinforcement Learning (RL) methods. As a proof of our concept, we aim to implement the RL-based controller in a purely classical experiment: temperature stabilization of a test mass. Moreover, we explore the possible implications of such a nonlinear controller in the field of quantum mechanics and non-classical experiments, where nonlinearities can be encountered even in the vicinity of the desired setpoint/mode of action of the system, exacerbating the need for a controller that can manage such nonlinearities.

2 Experimental Setup

2.1 Seismometer setup

As a proof of our concept, we chose to control the temperature of a seismometer at the LIGO 40m lab. Due to its mode of operation, the seismometer must be kept at a constant temperature in order to provide a reliable reading of movements of the earth. To this end, the seismometer is shielded by a large metal can from the fast temperature variations that might arise in the lab as a result of background processes such as winds. Effectively, the can acts as a low pass filter, since high temperature variations are being absorbed by the external can, and thus don't noticeably influence the temperature of the seismometer itself. However, low frequency variations (e.g., day-night temperature variations) would still influence the temperature of the seismometer as the can itself has enough time to thermalize with the environment, which in turn causes the seismometer to thermalize (and thus generate low frequency noise).

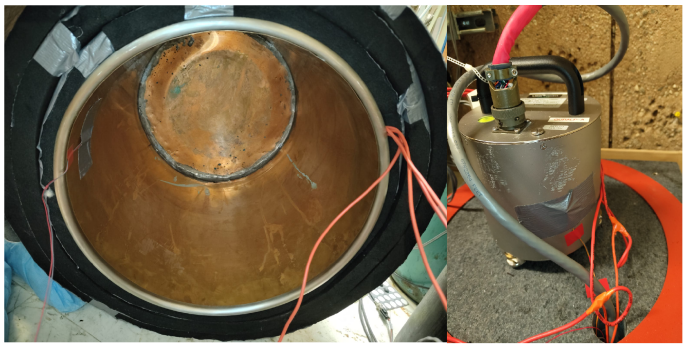


Figure 1: Seismometer setup. Left: the inside of the can, the foam is visible around the edge of the can; the red wires lead to the sensors and resistor used to measure the temperature and heat the can. Right: The seismometer device which rests inside the can.

Also visible in the picture are the three foam layers that surround the can which are used to further shield the can (and as a result the seismometer) from large temperature variations. The sensors and heater are also visible in the left picture.

The issue with the seismometer/can setup is that the masses are too large, which makes the testing of the controller take too long. As a result, we designed a toy example for our control problem: an aluminum puck wrapped in foam.

2.2 Puck setup

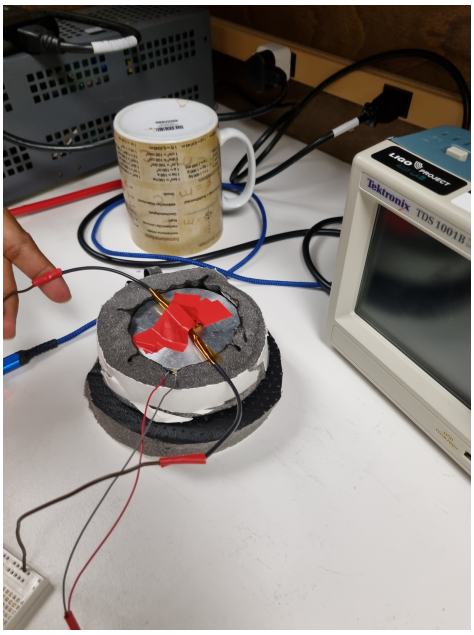


Figure 2: Puck setup. The heater resistor is visible on the top, and the temperature sensor is visible on the bottom edge of the puck.

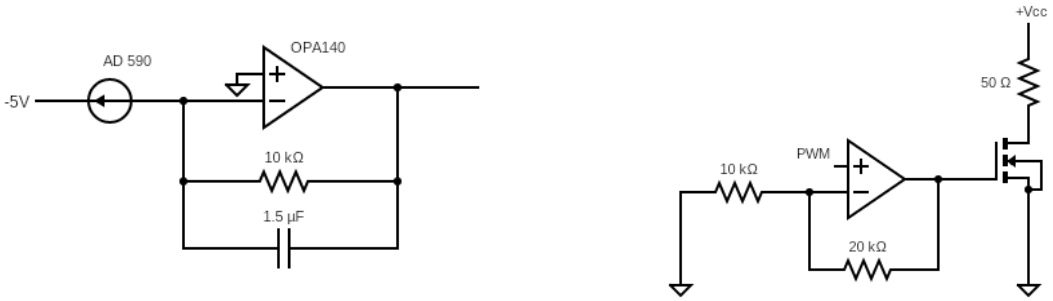
The puck weighs 298 grams and we supply a maximum heating power of 2W, which enables us to heat the puck by 8°C in less than half an hour. This property enabled us to test new controllers in a timely manner, as opposed to how long it would've taken on the seismometer. In this setup, both the sensor circuit and the heater circuit were connected to a RaspberryPi (not displayed in figure) which also acted as the controller for our system.

2.3 Hardware

2.3.1 Sensor

For sensing temperatures, we use a circuit based on the AD590 temperature transducer manufactured by Analog Devices. This sensor produces a current that varies linearly with the temperature, with a dependence of 1 $\mu A/K$.

We use this sensor with a transimpedance amplifier (Figure 3a) to convert the current signal to a voltage signal that can be read out by an ADC. We use a Raspberry Pi with the WaveShare AD/DA board, which is equipped with an ADS1256 ADC chip. This provides



(a) Temperature Sensor

(b) Heater

8 input channels with a range of 0 - 5V. The designed circuit outputs a voltage of 2.98 V at a temperature of 298 K, and has a feedback capacitor to act as a low pass filter of roughly 10 Hz to eliminate high frequency noise. OPA 140 was chosen to minimize the noise contribution of the TIA, as it is a JFET low input noise amplifier.

We estimated the amplitude spectral density of the noise source using LTSpice, and compared it to the spectral density of fluctuations of the environment itself. The design goal was to achieve noise that is at least 2-3 orders of magnitude below the ambient fluctuations, and this was successful (Figure 7).

2.3.2 Heater

The heater circuit is a power MOSFET (IRF 630) operating as a switch, as seen in Figure 3b. The heating power is controlled by pulse-width modulation (PWM). The Raspberry Pi is capable of supplying a PWM signal between 0 - 3.3 V, and this voltage is doubled by a non-inverting amplifier in order to exceed the V_{DS} required to turn the MOSFET on. This voltage is delivered to the gate of the MOSFET, which only operates in the fully-on or fully-off states depending on the input being on or off. The IRF 630 was chosen for its low R_{DS} value of 0.4Ω, leading to low heat dissipation even with several amperes of drain-source current.

This circuit was tested with PWM frequencies up to 1 kHz and performed well, with little to no heating at a few hundred milliamps of current. A small heatsink was required for currents of a few amperes.

2.4 Thermal Modelling

Both the seismometer and puck setup have similar heat transfer dynamics with the environment, with a mass being heated directly using ohmic heating, tightly wrapped by one or multiple layers of thermal insulation to isolate it from high frequency ambient noise. In principle, the major heat transfer processes involved can be summarized as below -

1. Conduction across the dimensions of the mass itself
2. Conduction across the thermal foam

3. Convection, conduction and radiation from the foam surface to the environment

We attempted to accurately model or reasonably rule out components out of the above, and arrived at a couple of models. One of them involved a system of two coupled differential equations, which incorporated the actual temperature gradient across the foam and convection from the foam surface. However, in practice, we were unable to fit this to experimental data, likely due to the number of free parameters.

Instead, we used a simpler model which neglected the effect of the temperature gradient entirely, and assumed that all the foam did was to reduce the convective heat transfer coefficient [2] of the system. This coefficient can be experimentally obtained. This gives a single differential equation that was in line with experimental results. The equation is given below -

$$mcdT = \eta H dt - hS(T - T_{env})dt$$
$$\frac{dT}{dt} = \frac{\eta H}{mc} - \frac{hS(T - T_{env})}{mc}$$

Here, m, c and S are known constants - the mass of the object (puck or cylinder), its specific heat, and the total surface area of the foam exposed to the environment. H is the heating rate, and η is an efficiency factor which is the fraction of heat that is actually delivered to the mass. As the heater is wrapped tightly by insulation, some heat is inadvertently lost to it. h is the convective heat transfer coefficient discussed earlier, and must be determined experimentally. T and T_{env} are the temperature of the mass and environment respectively.

Therefore, our model has two free parameters that must be fitted for - h and η. We also kept T_{env} as a constant free parameter as a way of confirming the actual temperature through the model.

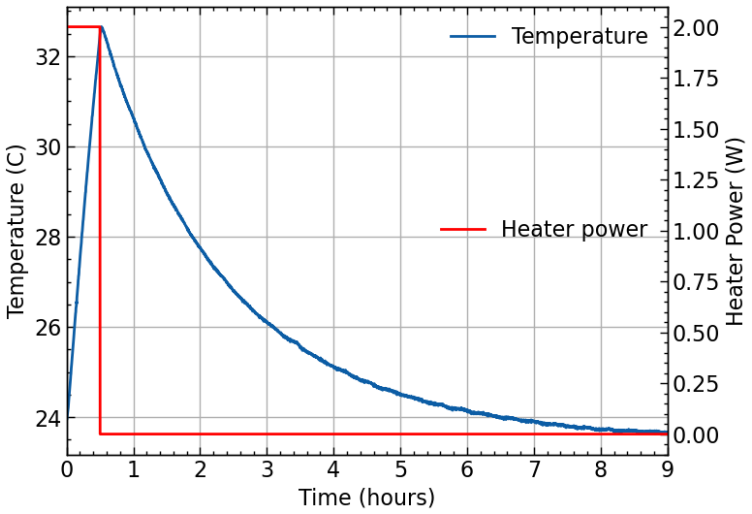
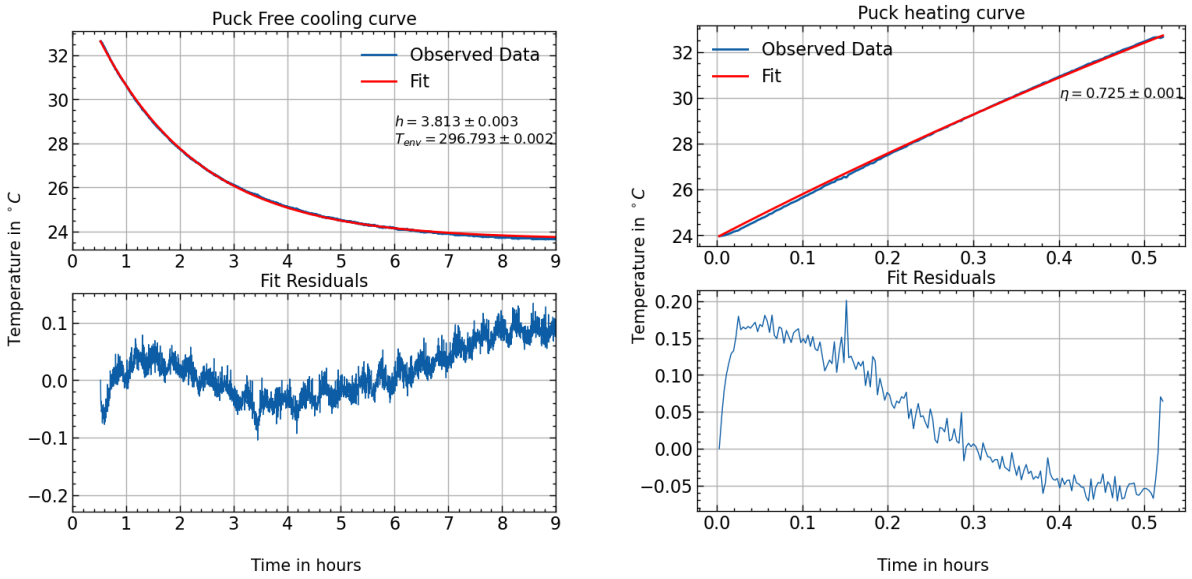


Figure 4: Plot summarising the experiment

2.4.1 Experimental Validation

In order to validate this model, we conducted experiments with the foam-wrapped puck where heating at a constant rate of 2W was turned on for 30 minutes, and then the puck was subsequently allowed to cool freely. The cooling curve accurately yields values of h and T_{env}, while the heating section gives us η. The data obtained from this is shown in Figure 4.

This data was then separated to heating and cooling sections, and both were fitted to the model independently. In the free cooling curve, H and η were, of course, fixed to zero. In the heating curve, h and T_{env} values obtained from the cooling section were used as fixed parameters and only η was inferred.



(a) Cooling curve

(b) Heating Curve

Figure 5: Fits to the experimental data with residuals

Figure 5 summarizes results from the fitting of this data. We obtain the values $h = 3.813$ and $\eta = 0.725$ for the puck setup, and these were repeatable across experiments for the setup. This model was also successful in fitting data from an experiment demonstrating I control on the puck temperature, as shown in Figure 9.

2.5 The Control Problem

We aimed to maintain the temperature of the puck at a constant temperature of 45^oC, while the environment temperature was kept at around 25^oC. However, the temperature ion the lab is constantly varying as measured here: As you can see in the data, the day-night temperature fluctuations are clearly visible, with there being a difference of around one degree between day and night. Consequently, one can notice in the spectral density the peaks corresponding to a frequency of $1/24h$ (red) and $1/12h$ (green).

We also estimated the noise put out by the AD590 sensors and their attached circuits. Below is a comparison between the lower bound of the total noise in the temperature readings and the environment temperature noise.

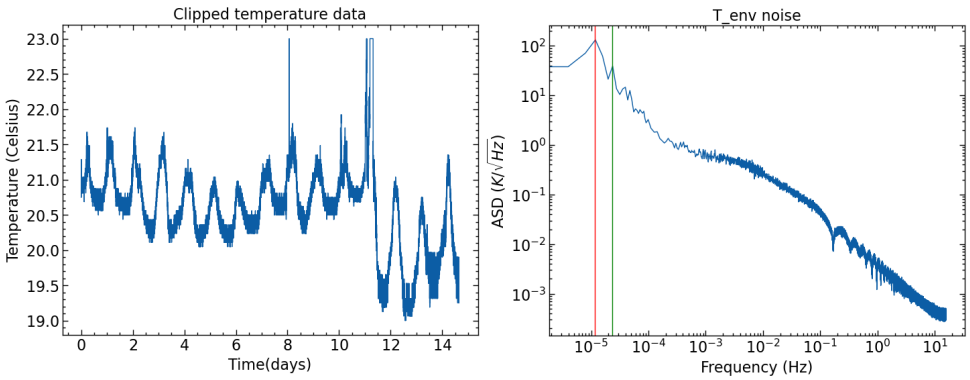


Figure 6: Environment temperature in the lab. The raw temperature data is displayed on the left, and the spectrum of the environment is shown on the right-hand-side. In the raw data, the readings of 23^oC are glitches, and in the spectrum figure the red line corresponds to $1/24h$ frequency and the $1/12h$ frequency.

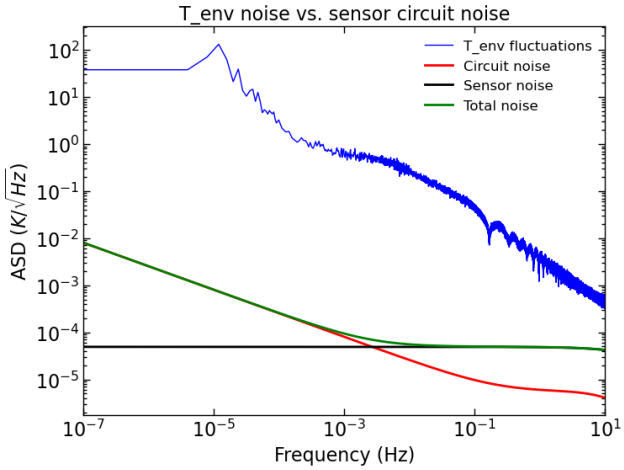


Figure 7: Comparison between the noise spectra of the temperature sensor and the environment temperature fluctuations.

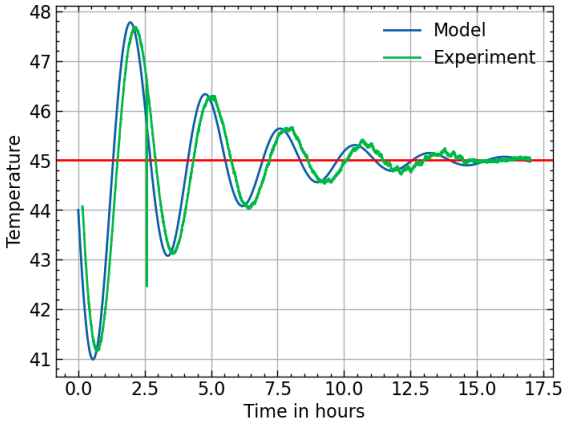
3 The Linear Controllers

3.1 Integral controller for the puck

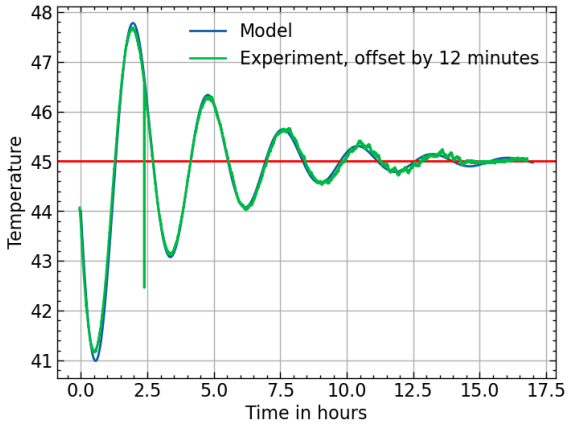
Using the heater and temperature sensor circuits outlined in the previous report, we implemented an integral controller for the puck. The controller had no proportional or derivative components, just an integrator with constant gain. Based on our step response fit, we managed to compute an estimate for the response of the integral response. Here is the comparison between the measured and predicted responses:

As it can be seen in Figure 8a, there seems to be a delay between the measured and predicted temperature responses. In fact, one can make the two curves match almost perfectly by

shifting the measured temperature by a 12 minutes, visible in Figure 8b.



(a) Modelled and measured responses



(b) Measured response shifted by 12 mins

This behaviour was also successfully recovered by fitting the model to the data, by supplying the model with an interpolated version of the heating values recorded during the experiment, further validating the model.

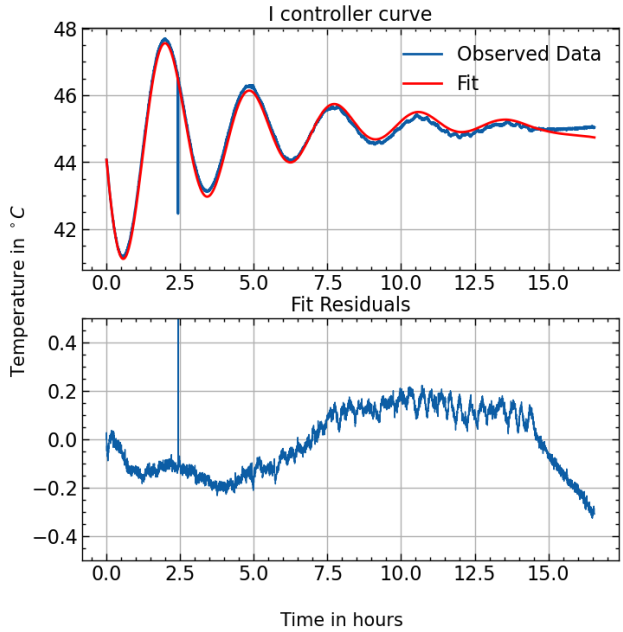


Figure 9: The model successfully fitted to I control data

3.2 The Optimal Linear Controller

Before implementing an optimal linear controller, we noted that in the frequency domain (see fig. 7) the sensor noise and the measured environment temperature fluctuations become comparable at higher frequencies. Moreover, if one accounts for the fact that the noise estimate of the sensor is just a lower bound, and that the insulation (foam and/or can) acts

as a low-pass filter, than it becomes clear that the noise introduced by the sensor is likely to be higher than the actual temperature noise at the puck. Hence, in order to avoid introducing the high frequency noise through feedback, we designed the optimal controller to make use of an estimator. The estimator can be implemented using the RaspberryPi to internally simulate the plant, and then the simulation may be constantly with the aid of Kalman filtering: by comparing the noisy temperature measurement with the expected simulated temperature, the Kalman filter effectively makes a small correction to the simulated plant, in order to keep the simulation as close as possible to the actual state of the physical system. This idea is still under development, but then one could employ their preferred form of optimal controller on top of this estimator to create an optimal linear controller.

4 Reinforcement Learning for nonlinear controller

4.1 Agents

Moving on, we implemented the first attempts at a nonlinear controller using Reinforcement Learning. We initially wanted to use the [4] library of RL agents, but it turned out that this library is not well maintained and there are numerous bugs in the code that prevented us from advancing in our endeavor. Therefore, we decided to use the TensorFlow Agents [3] library as a means of quickly deploying a reinforcement learning model. TF agents comes with an assortment of agents, but the ones that we were most interested in were:

1. **DQN**. This is one of the simplest modern RL agents: it has a single neural network (q-net) that aims to predict the reward of an action given the current and/or past states of the system. The main advantage with this architecture is that there are numerous resources online for it and training times are relatively short due to its simplicity. The disadvantages are that there is evidence that the algorithm can be unreliable during training and only works on discrete environments.

2. **PPO**. This agent is a popular choice usually for continuous environments, as it builds upon older, robust, but slower algorithms such as TRPO which have been shown to converge on most tasks (regardless of the complexity). The main idea behind PPO is to have 2 networks: one for the policy (i.e., the function that gives an action based on the current state) and another for the critic (the function that approximates the value of each action). The main advantage of this algorithm are its robustness which has been developed on in numerous papers, but the disadvantage is that it is a more complex agent which requires relatively longer training times.

4.2 Environment

Before we could implement any of our agents, we needed to define an environment, similar to the ones from OpenAI's Gym [1]. An environment is a python class which must contain the state of the system/game that we are trying to control/play. Essentially, the environment's responsibility is to respond to an action from the agent with a "timestep": a data structure which contains the next state of the system (following the application of the given action),

the reward for that action, the discount for the reward, and whether the episode has ended. In the usual setting of RL, the environment is usually the game (e.g., an Atari game) which has its own internal rules of advancing based on the actions of the player. In our case, the environment contains the simulation of the system that we are trying to control: it takes in the applied heating and outputs the temperature of the system after 10 seconds of continuous application of that inputted heating. The great advantage of this framework is that it is entirely modular: one can always adjust the equation governing the temperature or the reward function, while leaving all of the rest of the environment code intact. We used the heat equation fitted from the above step response and a reward function which gives as a reward the negative squared error between the set reference temperature and the current temperature of the system.

Currently, there are no noise sources implemented in the code (environment's temperature is fixed, and the Measurement noise is neglected), however, as outlined above, one can always change the governing equation to include any noise sources, time delays, or any other elements deemed of importance for the system. Moreover, since TensorFlow 2 infers the input sizes of the layers by the size of the given input, one can always easily add more data to the environment's observation that might be useful for the agent to reach a better performance: time of day, temperature forecast, last few measured timesteps and their corresponding actions (which is particularly useful if there is a delay).

4.3 Results and Further Work

Below you can see an early result of the PPO agent on this environment:

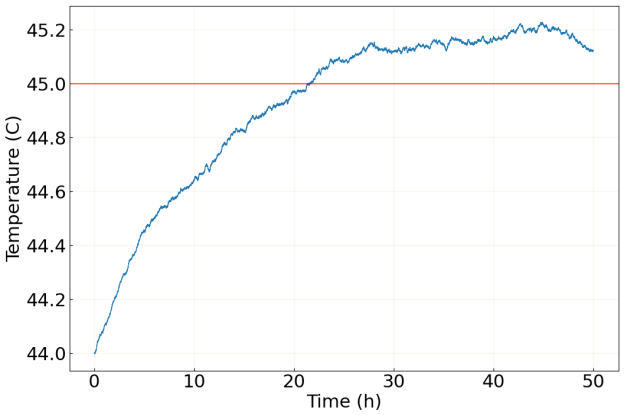


Figure 10: Early iteration of a PPO agent running on the puck environment.

So far the training has been very slow due to inefficiencies in the data collection for training. With careful tuning of the system, the training times have been reduced from days to hours. Moving forward we aim to come up with a code that reliably trains a model on the given environment and expand that model to also include additional data such as previous timesteps, time of day, weather forecast, etc.

5 Implications of Nonlinear Controls for Non-classical Systems

After the completion of these tasks, we are interested in exploring the implications of the nonlinear controllers for non-classical systems. The problem in non-classical experiments is that nonlinearities are not as easily mitigated by just keeping the system close to the desired mode of operation. Hence, it is expected that linear controllers are even less effective for such experiments, emphasizing the need for a nonlinear controller. One such experiment would be quantum state preparation: in order to measure the quantized energy levels of an object, one needs to prepare the state of the said object to be a superposition of two quantum states. This problem is deeply nonlinear, not least because of quantum effects such as the Uncertainty Principle. Therefore, there is an indication that a nonlinear controller that itself is being governed by quantum effects (such as the equivalent of a neural network using quantum gate logic) could be more effective at tasks that involve quantum effects. This topic will be further explored in the coming months.

6 Acknowledgements

I would like to thank my mentor, Prof. Rana Adhikari, the LIGO 40m lab, and the LIGO SURF program for giving me the opportunity to work on this project and their constant guidance and support through the summer.

This work was supported by the National Science Foundation Research Experience for Undergraduates (NSF REU) program, the LIGO Laboratory Summer Undergraduate Research Fellowship program (NSF LIGO), and the California Institute of Technology Student-Faculty Programs.

References

[1] Greg Brockman et al. *OpenAI Gym*. 2016. arXiv: 1606.01540 [cs.LG].

[2] Y.A. Cengel. *Heat Transfer: A Practical Approach. 2nd Edition,* 2020.

[3] Sergio Guadarrama et al. *TF-Agents: A library for Reinforcement Learning in TensorFlow*. https://github.com/tensorflow/agents. [Online; accessed 25-June-2019]. 2018. URL: https://github.com/tensorflow/agents.

[4] Matthew W. Hoffman et al. "Acme: A Research Framework for Distributed Reinforcement Learning". In: *arXiv preprint arXiv:2006.00979* (2020). URL: https://arxiv.org/abs/2006.00979.