Imitation Learning of Neural Network Control Systems

Fares Ahmad
Verimag
Grenoble, France
faresahmad9@outlook.com

Supervised by: Professor Thao Dang

I understand what plagiarism entails and I declare that this report is my own, original work.
Name, date and signature: Ahmad FARES on June 7,2024

Abstract

This study explores the integration of two PID controllers with complementary characteristics—one known for rapid responsiveness and the other for smooth operation—into a unified control system. The goal is to fully replace these controllers with a system that surpasses the performance of each individually. The process is guided by Signal Temporal Logic (STL), which directs the aggregation of data based on counter-examples and coverage measures. This approach is demonstrated through a case study on a water tank model, showing its effectiveness and potential advantages for industrial control systems.

1 Introduction

Control systems are essential in various industrial applications, utilizing mechanisms such as PID[1] (Proportional-Integral-Derivative) controllers to maintain desired system behaviors through continuous feedback. While PID controllers are renowned for their effectiveness and simplicity, their limitations become evident in complex or dynamically changing environments. These constraints have necessitated the exploration of more adaptive solutions, capable of addressing such challenges with greater flexibility and efficiency.

1.1 Context of Work

Recent advancements in control technology have highlighted the potential of neural networks as an alternative for traditional control methods. A notable example is found in the use of deep learning for controlling DC motors, which are important actuators[2] in various industrial operations. Research has explored the use of a Deep Belief Network (DBN) algorithm to learn from the operational data of a traditional PID controller. This approach was aimed at enhancing the control of motor speed, with simulations in MATLAB/Simulink indicating that the deep learning controller significantly outperformed the traditional PID controller, leading to improved performance and effectiveness[Cheon *et al.*, 2015].

Following this, more applications of neural network-based control systems have been explored, such as in the control of Segways, a two-wheeled, self-balancing personal transporter device. Traditionally managed via PID controllers, Segways require precise balance control, which was enhanced through the application of neural networks. The study demonstrated that neural networks could manage both the cart position and handlebar angle more efficiently than traditional PID controllers, resulting in better response accuracy [Ahmed and Saleh Alshandoli, 2020].

Both studies signify the growing trend towards integrating machine learning techniques into control systems, supporting the premise of this project, which seeks to combine the best features of different controllers into a superior neural network.

1.2 Brief Overview of the Project

The global aim of this project is to design a neural network (NN) controller capable of emulating the functionality of two distinct PID controllers. One is characterized by its rapid response,but with harsh overshoot, while the other operates slowly and smoothly. The objective is to integrate the advantageous traits of the two controllers; speed from the first and smoothness from the second into a neural network. The integration of PID control with neural networks represents a cooperative approach that combines the robustness of classical control theory with machine learning.

Motivation:

1. The primary motivation behind this stems from two key limitations of conventional controllers. Firstly, conventional controllers are often tied to their fixed operation domains and configurations and therefore can provide only limited performance. Secondly, conventional controllers can be expensive to deploy, (such as model predictive controllers, which require expensive online opti-

[1] A proportional–integral–derivative controller (PID controller or three-term controller) is a control loop mechanism employing feedback that is widely used in industrial control systems and a variety of other applications requiring continuously modulated control [From Wikipedia]

[2] a device that causes a machine to operate

mization).

2. Secondly, neural networks are known for their exceptional function approximation capabilities. They possess the ability to accurately approximate complex functions, making them well-suited for emulating the behavior of complex systems. Moreover, a well-trained neural network controller can deliver superior control performance and can be implemented on cost-effective, energy-efficient embedded platforms, presenting an alternative to traditional model-predictive controls and similar systems[Varshney *et al.*, 2019]

1.3 Expected Outcomes

The primary goal is to develop a neural network (NN) that not only imitates but also enhances the "good traits" of conventional PID controllers. We aim for the NN to eventually replace current controllers by integrating their best features while addressing common drawbacks such as high computational costs, slow response times, and significant energy consumption [Garcia *et al.*, 1989].

1.4 General Outline

Next in this paper, we will explore previous related work in the area of neural network and PID controllers in general. This is followed by the initial concepts and ideas, which establish the theoretical and practical foundation necessary for understanding the detailed Approach that has been implemented. Finally,the approach will be addressed with experimentation results.

2 Related work on the subject

For a better understanding of the topic, a skim through previous studies is essential. At first, this work extends from a research topic which presented a framework for efficiently training a neural network-based controller by imitation learning using a dataset aggregation approach. A key aspect of this framework is the strategic collection of data from an expert controller, defined by criteria specified using Signal Temporal Logic. Another key aspect is trying to falsify the NN by generating counterexamples through an algorithm, and in case of not finding counterexamples, providing evidence that the NN successfully imitates the controller with test coverage of at least ϵ [Dang *et al.*, 2023]. This methodology was applied in a flying robot case study involving a model predictive controller (MPC). The robot, using two thrusters[3] for 2-D navigation, was tasked with maintaining proximity to a predefined origin point during 15-second simulations. The performance evaluation of both the MPC and neural network-based controllers was guided by PSTL formulations (Explained later in section 3.2), focusing on overshoot, transient time, and stabilization.

The results showed that the neural networks improved overshoot metrics quickly, within just two iterations, yet they struggled to stabilize the robot near the origin.

[3] A propulsion device that combines a propeller with electric motor

3 Description of the Initial Concepts and Ideas Behind the Project

To achieve the operational level of this project, several foundational concepts must be thoroughly understood.

3.1 STL

Signal Temporal Logic (STL) extends LTL to handle real-time and real-valued constraints effectively, making it suitable for continuous and hybrid systems where precise timing and magnitude constraints are crucial. STL incorporates these elements using intervals and thresholds directly in the logic expressions.

The syntax of STL is enriched to express conditions over real-valued signals and time intervals. Key operators include:

- $\mathbf{U}[a, b]$: Until within a time interval $[a, b]$.
- $\mathbf{F}[a, b]$: Eventually within $[a, b]$, shorthand for true until a condition holds within the interval.
- $\mathbf{G}[a, b]$: Always within $[a, b]$, signifying that a condition holds throughout the interval.

STL predicates are relations over real-valued functions of time, such as:

$$f(x[t]) > 0$$

where f can be any real-valued function, and $x[t]$ denotes the signal at time t.

Semantics: The satisfaction of STL formulas is defined with respect to signals and their values over time. For example, the formula:

$$\mathbf{G}[0, T](x[t] > 5)$$

states that for all times t in the interval from 0 to T, the signal $x[t]$ must be greater than 5.

Robustness: STL also supports robustness metrics, which quantify how well a signal satisfies or violates a specification. This is particularly useful in applications where you need to measure the degree of compliance with the requirements, rather than just a binary true/false evaluation.

STL allows for detailed specification and checking of temporal properties that our neural network-based controller must satisfy.

3.2 Parametric Signal Temporal Logic (PSTL)

Parametric Signal Temporal Logic (PSTL) extends traditional STL by incorporating parameters within the logic formulas, allowing numeric constants to be represented as symbolic parameters.

Definition

Informally, a PSTL formula is a standard STL formula where numeric constants are replaced by symbolic parameters, which can be tuned or optimized based on specific system requirements or conditions. The general syntax for a PSTL formula is defined as follows:

$$\varphi := \mu(x[t]) > \pi \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \mathbf{U}_{[\tau_1, \tau_2]} \psi$$

where π represents a scale parameter, and τ_1, τ_2 are time parameters.

Illustrations
PSTL allows for the dynamic adaptation of conditions in a system's specification [Asarin *et al.*, 2012]. For example, a typical PSTL specification might state that "After τ seconds, the signal is never above π," which can be mathematically represented as:

$$\varphi := \mathbf{G}_{[\tau,\infty)}(x[t] < \pi)$$

This flexibility in defining τ and π allows for fine-tuning the system behavior based on observed performance or changing operational conditions.

Parameter Synthesis
The objective of parameter synthesis in PSTL is to find an optimal or tight valuation of parameters that satisfy the PSTL formula under specific conditions. The problem is formally defined as:

> Given a system S with a PSTL formula containing n symbolic parameters $\varphi(p_1, \ldots, p_n)$, find a valuation function v such that the system behavior at time t satisfies $\varphi(v(p_1), \ldots, v(p_n))$.

A valuation v is considered *tight* if small perturbations in parameters do not violate the formula, providing robustness in parameter choice.

Challenges in PSTL
While PSTL offers significant advantages in specifying and analyzing system behaviors, it also introduces challenges such as:

- Determining which parameter values to choose among potentially many solutions.
- Ensuring the tightness of these solutions, which often requires complex optimization techniques.

3.3 Breach Tool

Breach is a MATLAB/C++ toolbox designed for time series analysis and simulation-based analysis of dynamical, Cyber-Physical Systems (CPS), and hybrid systems [Donzé, 2010]. This tool plays a core role in our study, including checking formal requirements on data and dynamical system behaviors using STL formulae, conducting extensive testing, and ultimately falsifying an STL requirement using various optimization algorithms. Consequently, the Breach tool is essential to our testing of neural networks, since it allows testing the system with a neural network controller in closed loop.

Moreover, Breach facilitates the computation and of trajectories in hybrid systems, providing a set of simulation-based techniques for analyzing deterministic models. Its primary feature is its ability to compute and analyze the properties of large sets of trajectories efficiently. This is achieved through an efficient numerical solver for ordinary differential equations, which also provides sensitivity analysis with respect to parameter variations.It also, supports the evaluation of STL formulas, allowing for detailed specification and monitoring of system properties over time.

Property-Driven Parameter Synthesis
Breach supports property-driven parameter synthesis using STL formulas. This involves specifying properties as STL formulas and using Breach to find parameter values that satisfy these properties. This featiure is useful for designing systems that need to meet temporal and logical specifications.

Applications
Breach has been effectively applied in various fields, including:[Donzé, 2010]

- **Embedded Systems Design:** For verifying and validating system behaviors under different operational scenarios.
- **Systems Biology:** For analyzing complex biological models where parameters are uncertain or vary over time.

Breach is a powerful tool for the analysis and verification of hybrid dynamical systems. Its integration into our project enables detailed and robust testing of neural network controllers, ensuring they meet the desired specifications.

3.4 STL-Imitation-Algorithm

This code is associated with the research paper [Dang *et al.*, 2023], where the behaviours of a flying robot, controlled by a (computationally expensive) model prodictive controller (MPC), are simulated in order to generate data for the imitation learning algorithm that trains the Neural Network to imitate the MPC controller. The imitation learning algorithm makes use of formal STL specification and validation techniques. Below, the training and validation/falsification algorithms used.

Algorithm 1 Dataset aggregation-based training algorithm

$\mathcal{N}_0 \leftarrow \emptyset, \mathcal{D}_0 \leftarrow \emptyset, k \leftarrow 1$
repeat
  $(\mathcal{D}_k, \text{Status}) \leftarrow \text{getNewData}(\mathcal{N}_{k-1}, \mathcal{D}_{k-1})$
  if $\mathcal{D}_k \neq \mathcal{D}_{k-1}$ **then**
    $\mathcal{N}_k \leftarrow \text{Train}(\mathcal{D}_k)$
    $k \leftarrow k + 1$
  end if
until $\mathcal{D}_k = \mathcal{D}_{k-1}$ or $k > k_{\max}$
return $\mathcal{N}_k$, Status

Algorithm 1 is the main training algorithm. The dataset aggregation algorithm determines whether it should generate new data for retraining or stop. It stops after a maximum number of iterations, denoted by $k_{\max}$, or as soon as the procedure responsible for providing new training samples fails to do so. The training procedure is considered successful if it terminates with no new data found, and the status returned by GetNewData (Algorithm 2) is "No counter-example found", as we will see in the sequel.

Algorithm 2 provides Algorithm 1 with the data. On the first iteration, with no initial data available, the procedure begins by sampling initial state through the `getInitSamples()` function. These samples are then

Algorithm 2 New data acquisition procedure

procedure GETNEWDATA(D, N)
  if $D = \emptyset$ **then**
    InitSamples ← GETINITSAMPLES
    InitTraces ← SIMNOMINAL(InitSamples)
    D_{new} ← GRIDFILTER(InitTraces)
    status ← "New data available for training."
  else
    CexTraces ← FALSIFY(N)
    if CexTraces $\neq \emptyset$ **then**
      (D_{new}, **status**) ← FIXANDMERGE(D, CexTraces)
    else
      $D_{\text{new}} \leftarrow D$
      status ← "No counter-example found."
    end if
  end if
  return (D_{new}, **status**)
end procedure

processed by a function `simNominal` to simulate nominal behaviours of the system with the expert controller, which are further refined through a `gridFilter` to obtain new data (D_{new}) which is significant for the quality of the learning process. The data refinement is needed to discard some data that does not bring relevant additional information. The status is then updated to indicate that new data is available for training.

On later iterations, the procedure tests if the learned neural network controller satisfies the requirements by searching for counter-examples via the `falsify` function. If counter-examples are found, they are integrated with the existing data D using the **fixAndMerge** (Algorithm 3), updating D_{new} with the merged data and a status indicating that new data is acquired and to be treated by the training Algorithm 1. If no counter-examples are found, D_{new} remains the same as D, and the status is updated to reflect that no counter-examples were found, indicating a successful data validation without the need for updates. This algorithm enhances a training

Algorithm 3 Augmenting the training data from bad traces

procedure FIXANDMERGE($\mathcal{D}$, CexTraces)
  NeighSamples ← gridFilter(CexTraces) Cover($\mathcal{D}$)
  if NeighSamples $= \emptyset$ **then**
    status ← "Counter-examples do not add new data."
  else
    FixedTraces ← simNominal(NeighSamples)
    $\mathcal{D}_{\text{new}}$ ← gridFilter($\mathcal{D} \cup$ FixedTraces)
    status ← "New data available for training."
  end if
  return $\mathcal{D}_{\text{new}}$, status
end procedure

dataset by incorporating information from bad data traces corresponding to the unsatisfactory behaviours of the system with the current neural network controller. It starts by applying a `gridFilter` to the counter-example traces and checking if they overlap with the existing covered dataset (`Cover`(D)). If the counter-examples do not contain samples that are not covered, it concludes that the counter-examples do not provide any new information, and the dataset remains unchanged. However, if new samples are identified, the procedure generates fixed traces from them by simulating the system with the expert controller via the function `simNominal`. These fixed traces are then merged with the original dataset D using again the function `gridFilter`, resulting in an updated dataset (D_{new}) that now includes these corrected traces. The status is updated accordingly to reflect whether new data has been added for training.

3.5 Prototype: Water Tank

A water tank prototype is employed, due to its simplicity in simulation compared to the previously used flying robot, allowing for quicker results, more extensive testing, and ideas for improvements. It is important to note that PID controllers are not universally applicable across different physical systems to be controlled; they require specific fine-tuning to be compatible with the intended physical system. The existence of the water tank model, equipped with its adjustable PID controllers, facilitates a comprehensive understanding of the impact of each type of controller. This setup ensures a full understanding of each controller's interaction with the system. Water tank systems are commonly used to illustrate both traditional and advanced control strategies. For example, robust control techniques and MIMO PID controllers have been employed to stabilize the level and temperature of water tanks.[Rojas–Moreno and Parra–Quispe, 2023].

4 Testing on 1 Controller

A crucial step before addressing the main problem is to thoroughly understand the behavior of each controller individually. This involves documenting and analysing the results to provide a solid comparison at the end, allowing us to determine if we have met the research objectives.

First, each controller is tested to ensure it is working correctly. Once verified, the algorithm mentioned in Section 3.4 is applied to it. Each step is carefully tracked as outlined above.

4.1 Simulation Algorithm

Algorithm 4 Simulation of Breach Water Tank

procedure SIM_BREACH_WATERTANK(control_fn, t, p)
  Initialize parameters and initial state
  for each time step k **do**
    Get current state
    Compute control input using the provided function
    Update plant state using ODE solver
    Store new state values
  end for
  Finalize the last control input
end procedure

Algorithm 4, simulates the behavior of a water tank system controlled by a specified controller. It starts by initializing the

necessary parameters and the initial state of the water tank system. An output signal array is then used to store the state values of the system at each time step.

The main simulation loop iterates over each time step in the simulation horizon. For each time step, the algorithm retrieves the current state, computes the control input (Detailed in next section 4.2) using the provided control function (control_fn) produced by the controller, updates the plant state using an Ordinary Differential Equation (ODE) solver, and stores the new state values in the output signal array.

4.2 Computation of Control Signal (u)

The control signal u is computed using a discrete-time PID control law, which adjusts the system's response based on the error between the current height of the water tank and the reference height.

The state vector y is defined as:

$$y = [H, H_p, H_{pp}, \text{ref}, \text{ref}_p, \text{ref}_{pp}, u_p]$$

where:

- $y(1) = H$: Current height of the water tank
- $y(2) = H_p$: First derivative of the height (velocity)
- $y(3) = H_{pp}$: Second derivative of the height (acceleration)
- $y(4) = \text{ref}$: Reference height
- $y(5) = \text{ref}_p$: First derivative of the reference height
- $y(6) = \text{ref}_{pp}$: Second derivative of the reference height
- $y(7) = u_p$: Previous control input

The errors are defined as:

$$e = y(1) - y(4)$$

$$ep = y(2) - y(5)$$

$$epp = y(3) - y(6)$$

The control signal u is then computed as:

$$u = y(7) + a \cdot e + b \cdot ep + c \cdot epp$$

where the coefficients a, b, and c are given by:

$$a = Kp + \frac{Ki \cdot Ts}{2} + \frac{Kd}{Ts}$$

$$b = -Kp + \frac{Ki \cdot Ts}{2} - \frac{2 \cdot Kd}{Ts}$$

$$c = \frac{Kd}{Ts}$$

To change the behavior of the controller(fast/slow/...), the proportional (Kp), integral (Ki), and derivative (Kd) are adjusted. These parameters influence the responsiveness and stability of the control system.

4.3 Results of Simulation

To determine if a controller is working as expected, we monitor the height of the water tank, "H", and ensure it follows the reference signal, which is a step function. A controller is considered correct if "H" closely tracks the reference signal with minimal error, acceptable rise time, and appropriate settling time.

The results of the simulations for Controller 1 and Controller 2 are presented in Figure 1 and Figure 2, respectively. Both graphs display the reference signal alongside the actual height of the water tank.

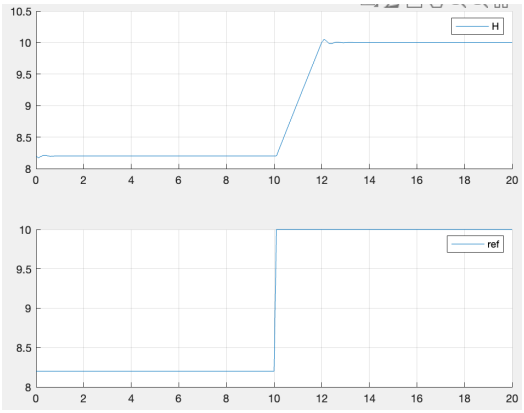

Figure 1: Controller Fast: Height of Water Tank (H) vs. Reference

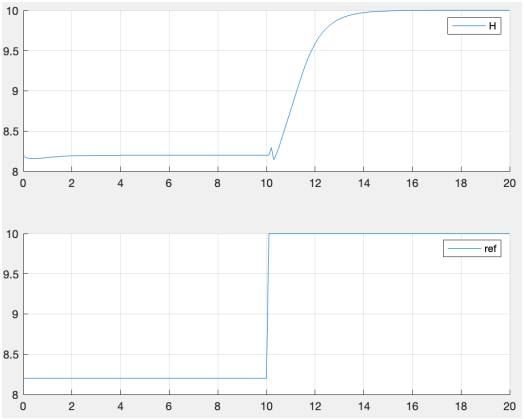

Figure 2: Controller Slow: Height of Water Tank (H) vs. Reference

Figure 1 demonstrates that Controller 1 responds rapidly to changes in the reference signal, showing a faster rise time and quicker adjustments. In contrast, Figure 2 illustrates that Controller 2 has a slower response, with a more gradual rise time and smoother adjustments to the reference signal.

After thoroughly testing both controllers, it is evident that they are functioning as expected and are eligible for further use. We will apply the algorithm mentioned in Section 3.4 to train a neural network that imitates each controller's behavior individually.

5 Proposed Approach

The main objective of this approach is to combine two distinct controllers into a neural network that can effectively emulate their behaviors. To achieve this, an instance of the water tank system model is created with two control functions. The rest of the work is handled within the simulation function, where data is produced to train the neural net.

The modified simulation algorithm (Algorithm 5) incorporates both controllers by applying them at each time step and then selecting the better outcome based on performance evaluation.

Algorithm 5 Simulation of Breach Water Tank with Dual Controllers

```
procedure SIM_BREACH_WATERTANK(control_fn1, control_fn2, t, p)
    Initialize parameters and initial state
    for each time step k do
        Get current state
        Compute control inputs using both controllers
        for each controller 1 and 2 do
            Apply control input
            Update plant state using ODE solver
            Store new state values temporarily
            Evaluate performance
            if new state is better then
                Update best distance
                Update best state
            end if
        end for
        Update the state with the best result from this step
    end for
    The corresponding control input is stored in the data to be used for training.
end procedure
```

5.1 Simulation Algorithm with Dual Controllers

In the modified simulation function, we follow these steps:

- Parameter Initialization
- Main Simulation Loop: For each time step:
  - The current state is retrieved.
  - Both control inputs are computed using the two provided control functions.
  - Each control input is applied to the system to generate a new state.
  - The performance of each new state is evaluated by comparing the height of the water tank (H) to the reference value (ref).
  - The state that results in the smallest difference from the reference value is selected as the best state for that time step.
- Finalization: The corresponding control input is stored in the data to be used for training.

This approach avoids the need for external functions to handle repetitive tasks. Instead, it evaluates both control inputs within each iteration of the simulation loop. By directly comparing the outcomes, we can dynamically choose the better control action at each step.

6 Initial Experimentation

The goal of the initial experimentation is to ensure that the combined controller is a correct controller. We conducted simulations to compare the height of the water tank, H, against the reference height.

The results showed that the combined controller matched the fast controller's behavior, with H closely following the reference. This indicates that our combined controller is functioning correctly.

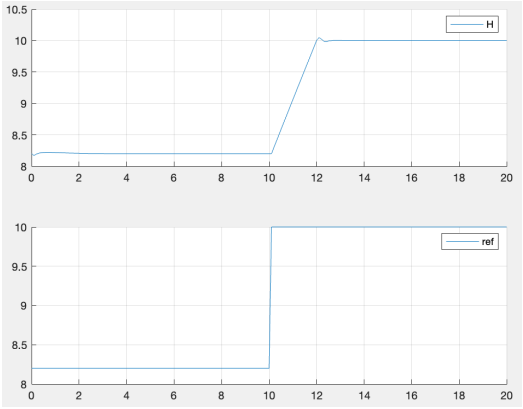

Figure 3: Combined Controller: Height of Water Tank (H) vs. Reference

7 Conclusion

In this study, we explored the integration of two PID controllers with distinct characteristics into a neural network-based control system. By thoroughly testing each controller individually and documenting their performance. The proposed approach involved simulating the water tank system with dual controllers, dynamically selecting the best control action at each time step.

The results demonstrated the feasibility of combining the strengths of both controllers to achieve improved control performance. However, further experimentation and observation are needed to refine and validate the controller during training. The initial results provide a promising foundation for developing a neural network-based control system that integrates the best features of both the fast and slow controllers.

Future work will focus on optimizing the neural network training process and exploring additional applications for this integrated control system.

References

[Ahmed and Saleh Alshandoli, 2020] Abdussalam Ali Ahmed and A.F. Saleh Alshandoli. On replacing a pid

controller with neural network controller for segway. In *2020 International Conference on Electrical Engineering (ICEE)*, pages 1–4, 2020.

[Asarin *et al.*, 2012] Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. Parametric identification of temporal properties. *Unknown Journal*, Unknown Volume(Unknown Number):Unknown Pages, 2012.

[Cheon *et al.*, 2015] Kangbeom Cheon, Jaehoon Kim, Moussa Hamadache, and Dongik Lee. On replacing pid controller with deep learning controller for dc motor system. *Journal of Automation and Control Engineering Vol*, 3(6):1–5, 2015.

[Dang *et al.*, 2023] Thao Dang, Alexandre Donzé, Inzemamul Haque, Nikolaos Kekatos, and Indranil Saha. Counter-example guided imitation learning of feedback controllers from temporal logic specifications. In *62nd IEEE Conference on Decision and Control (CDC)*, Singapore, Singapore, Dec 2023. hal-04295795.

[Donzé, 2010] Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification, 22nd International Conference*, pages 167–170. Springer, 2010.

[Garcia *et al.*, 1989] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[Rojas–Moreno and Parra–Quispe, 2023] Arturo Rojas–Moreno and Arturo Parra–Quispe. Design and implementation of a water tank control system employing a mimo pid controller. *Faculty of Electrical and Electronic Engineering, National University of Engineering Lima*, 2023. Lima 25, Peru.

[Varshney *et al.*, 2019] P. Varshney, G. Nagar, and I. Saha. Deepcontrol: Energy-efficient control of a quadrotor using a deep neural network. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 43–50, 2019.