DESIGN OF A DEEP REINFORCEMENT LEARNING BASED OPTIMAL PH CONTROLLER FOR NITRIFICATION BIOREACTORS IN AQUAPONICS SYSTEMS

Pin Chathushka Parami De Silva

(168658R)

Dissertation submitted in partial fulfillment of the requirements for the degree Master of Science in Industrial Automation

Department of Electrical Engineering

University of Moratuwa

Sri Lanka

May 2019

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date:

The above candidate has carried out research for the Masters dissertation under my supervision.

Name of the supervisor: Dr. A.G.B.P Jayesekera

Signature of the supervisor: Date:

To all the people who feed us….

ACKNOWLEDGEMENTS

This research would have been another dream without the support from the following people.

First of all, I wish to express my sincere thanks to my supervisor, Dr Buddhika Jayasekara, for guiding me in the research process and providing feedback and advice over the years.

Next, I wish to thank Dr. D.P. Chamdima and Prof. K.T.M.U. Hemapala for their valuable feedback. Their feedback was extremely valuable in asking the right research questions and steering the study in the correct direction.

I wish to thank Eng. B.S. Samarasiri for mentoring me through the years and introducing me to research and product development. I would also like to give my sincere thanks to Mr. Jayasiri Kumarasinghe for all those industrial outings and for the valuable and practical industrial experience. He has taken me all over the country to see various industries and introduced me to industrial instrumentation and automation. This exposure was significant in finding the currently studied research problem.

Finally, I would like to thank my wife, my mother, my father and my brother for all the love and support they have given me and persevering with me during the entire period of the research.

ABSTRACT

Recent advances in deep reinforcement learning has produced state of the art algorithms. These algorithms have better training stability, convergence and computational performance.

In this study a state of the art deep reinforcement learning algorithm is used to implement a self-learning, model free, non-linear controller to control pH of an aquaponic system. Aquaponics is a soil-less farming system where effluent water from a fish tank is used as nutrients for growing plants. Maintaining the pH of an aquaponic system provides the optimal condition for micro-organisms that convert the ammonia rich fish effluent to nitrates, which are easily absorbed by the plants. In order to optimize this conversion process known as nitrification, pH is maintained at optimal conditions within an intermediate setup known as the nitrification bioreactor.

The implementation of a deep reinforcement learning based controller is studied in detail and the performance of the deep reinforcement learning based pH controller is evaluated by comparing the performance of a classic PID based controller in an aquaponic system.

The results show that DRL based controllers are better suited for control of dynamic stochastic control pH process and is capable of learning complex plant models and tuning itself based on the learnt model. The outcomes of this research can be applied in the design of optimal controllers that learns purely from experience to optimize various industrial processes. This type of controllers is ideal in Industry 4.0 based applications.

Keywords: Deep Reinforcement Learning, Artificial Intelligence, Aquaponics, Nitrification, Process Control

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
AI	Artificial Intelligence
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
MDP	Markovian Decision Process
POMDP	Partially Observable Markovian Decision Process
SISO	Single Input Single Output
MIMO	Multiple Input Multiple Output
DP	Dynamic Programming
ADP	Asynchronous Dynamic Programming
GPI	Generalized Policy Iteration
RPi	Raspberry Pi
I2C	Inter- Integrated Circuit Protocol
ReLu	Rectified Linear Unit
IoT	Internet of Things
ANOVA	Analysis of Variance

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

Recent advances in Deep Reinforcement Learning (DRL) have produced state of the art learning algorithms. These algorithms are capable of performing searches in very large state spaces without instabilities and have human like performance in control. One such application of these algorithms is Alpha Go [1]. Alpha Go is a deep reinforcement learning based algorithm that was able to defeat a 9-dan champion Go player. Go is an ancient traditional Chinese board game, that is known to be much complex than chess, and widely agreed by the scientific community that the game requires human like intuition to win the game. This is a significant achievement in the field of Artificial Intelligence and was not expected to be achieved in the next decade [2]. The work carried out in creating the Go playing AI agent is expected to impact the industry and is currently under active research.

The current work in deep reinforcement learning is limited in scope and is mostly used in AI agents to play computer games and in simulation based research. However, these algorithms can be applied in solving complex industrial control and optimization problems. The application of state of art Deep Reinforcement Learning algorithms in industrial and process control systems have not been studied so far and forms the basis of this work. An early study carried out by Spielberg et al [3], provided a methodology to implement DRL algorithms in a simulated SISO & MIMO process control system and highlighted the requirement to extend the research to non-linear systems.

Hence this thesis reports on research aiming at applications of deep reinforcement learning on real world non-linear control systems. Since biochemical processes are highly non-linear with limited sensing and actuation, a biochemical process system was selected in order to evaluate the proposed DRL controller. Therefore, this research attempts to incorporate insights about application of deep reinforcement learning in designing an optimal pH controller for a nitrification bioreactor in an aquaponic system [4]. The use of advanced control schemes in bioreactor control have not been studied so far in aquaponics systems and is therefore studied in detail in the following work.

Aquaponics is an agricultural system that uses effluent from rearing fish tanks as nutrient fertilizer for growing plants in industrial, indoor and vertical farms [5]. In the aquaponics system shown in Figure 1.1, fish are reared in a tank from which effluent water is pumped into a soil-less grow bed system. This water flows through the grow bed system delivering nutrients to plants grown within the grow bed and recirculated back to the fish tank. The entire system forms a symbiotic system where the plants clean the fish effluent from the water and the fish provides the appropriate nutrients for the plants. This process is maintained by microorganism that convert the ammonia in fish waste in to nitrates that can be easily absorbed by the plants [6]. In order to facilitate the process of conversion of fish waste to fertilizer (nitrification), bioreactors are employed in aquaponics system and optimal conditions maintained within the bioreactor to facilitate the growth of nitrifying bacteria. Aeration of the water generally occurs naturally when water flows through the system and supplies oxygen for plant roots and fish. However, for complex aquaponics system aerators may be employed to supply additional oxygen required by the fish. Using this method, food production factories can be implemented to produce vegetables and fruits productively in smaller areas than conventional farms.

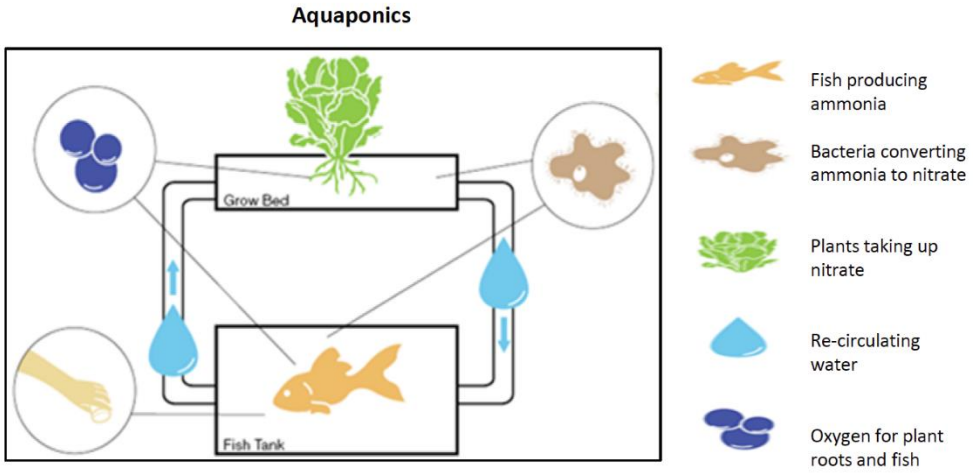

Figure 1.1 Operation of an Aquaponics System

1.1 Objectives

The objective of this study is to:

Evaluate the use of Deep Reinforcement Learning based optimal pH controller in a nitrification bioreactor of an aquaponics system.

As Deep Reinforcement Learning is a form of general purpose artificial intelligence, the following research questions are proposed to the context of this study.

RQ1: Can deep reinforcement learning algorithms be used to control the nonlinear bioreactor system without prior knowledge of the nitrification bioreactor?

Controlling without an explicit mathematical knowledge of the plant is preferred in the industry as complex systems can be setup with ease. This question attempts to isolate the algorithms that can learn the optimal control strategies without an explicit mathematical model of the system. In other words, what algorithms can learn the optimal control strategy based purely on experience?

RQ2: How can these deep reinforcement learning algorithms be used with partial observation of states?

Most system states in a nitrification bioreactor cannot be observed. This is due to non-availability of sensors (due to cost, accuracy or off-line measurements). Therefore, these control problems are partially observable. This research question attempts to find the deep reinforcement learning methods of optimal control with partial observability. In this case, we attempt to optimize the entire bioreactor by measuring the pH only, just as a human expert is capable of estimating the performance of the plant using the pH alone.

Figure 1.2 depicts the relationship between the objectives, research questions and various technological domains.

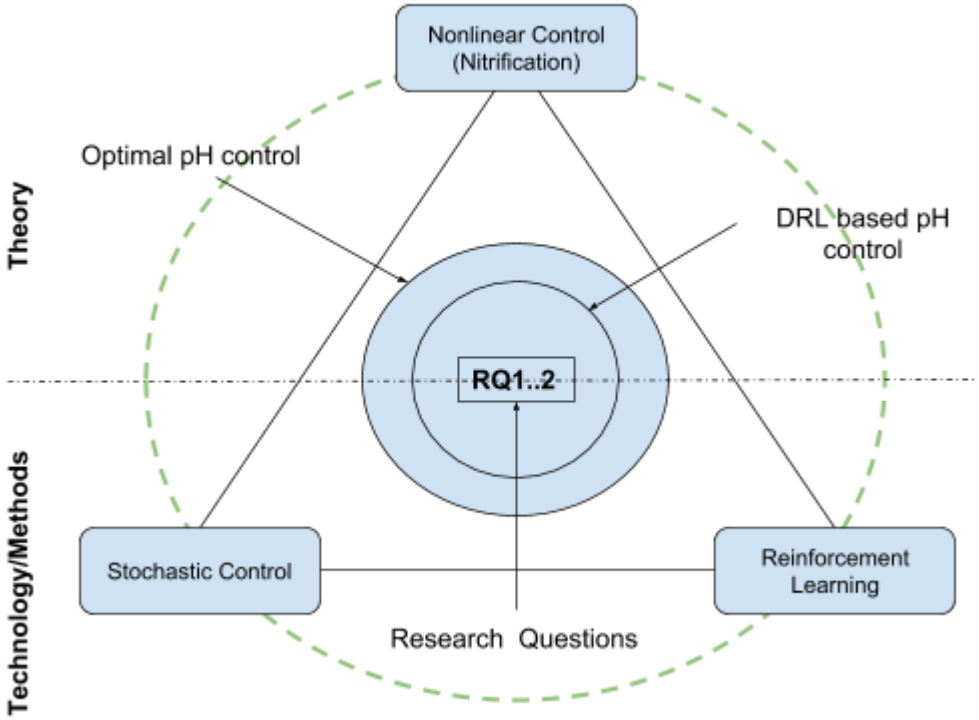

Figure 1.2 Relationship between the research problem, research questions and discipline

As shown in Figure 1.2, the research questions are bounded by the DRL based approach and by optimal pH control. The methodology and design of the controller in this study is determined by the three interacting disciplines of reinforcement learning, stochastic control and nonlinear control of nitrification.

1.2 Thesis Outline

This study is organized into five chapters.

- Chapter 1 introduces the research problem, the objectives of the study and the research questions.
- Chapter 2 presents detailed literature review and related works on nitrification process, pH control and fundamental theories of reinforcement learning and deep reinforcement learning. The first section of chapter 2 gives an overview on nitrification and mathematical models of nitrification used in bioreactors of aquaponic systems. Various linear and nonlinear methods of pH control are analyzed and the later part of chapter 2 introduces important deep reinforcement learning algorithms and caveats of using deep reinforcement learning algorithms when designing controllers for non-linear process control systems.
- Chapter 3 presents the design and development of the pH controller and the methodology to evaluate the performance of the implementation of the DRL controller. Here the controller is evaluated by comparing the performance of the DRL controller with respect to a classical PID controller. Two experimental setups were devised to characterize the performance of the controllers in the case of a deterministic process and an aquaponic system as a dynamic process.
- Chapter 4 presents the experimental results and the analysis of the results of each of the controller in both control scenarios. The details of the experimental setups, the experimental procedure and comparative analysis are presented in this chapter.
- Chapter 5 discusses the conclusion of the study and related further works.

1.3 Limitations of the Study

This sections explains the limitations which are important to the context of the current study.

Firstly, this study has established a boundary around its research problem by delimiting the scope of the investigation on optimal control of bioreactors and focusing specifically on two aspects: reinforcement learning and pH control. Although many other measurements can be taken for optimal bioreactor operation, this research has delimitation, focusing on a single important measurement to reach optimality.

CHAPTER 2 LITERATURE REVIEW

2.1 Introduction

This chapter presents important literature in the context of deep reinforcement learning based control. The complex nature of the pH control in aquaponics is also discussed before reviewing various theories and algorithms used in deep reinforcement learning.

2.2 Related works in Aquaponics and Limitations

Based on the literature review, there are no academic studies carried out in optimal pH control of Aquaponics systems. This is mainly due to aquaponics being developed by agricultural experts with little intervention from the engineering discipline. Previous work carried out by the author showed that aquaponics systems can be modeled using a waste water treatment process. Therefore, controllers used in waste water treatment are potential candidates for pH control in aquaponics systems. Most of the pH controllers used in waste water treatment are PID controllers (refer appendix A). These controllers require constant tuning and other adjustments to continuously monitor the process. This makes the entire process sub optimal.

The requirement for an optimal pH controller comes from our necessity to optimize biochemical processes such as nitrification [7]. Therefore, it is important to show the non-linearity of biochemical processes and how conventional linear control design techniques are obsolete. Nitrification is the process of converting ammonia to nitrates using bio-organisms. Nitrification is part of the natural nitrification cycle and is extensively studied in biology and agriculture. Nitrification has been used in waste-water treatment [8] where closed systems have been developed to enhance the nitrification process. Therefore, we shall discuss the process of nitrification in the context of waste-water treatment as applied to aquaponics systems.

Nitrification happens within a mineralization tank bioreactor where waste water with high concentrations of ammonia is fed in. The bioreactor hosts nitrification bacteria that convert ammonia into nitrates using an aerobic process. Aerobic processes consume oxygen and oxidize the ammonia into nitrates. As far as industrial processes are concerned, controllers have been used to maintain optimum conditions that promote growth of nitrifying bacteria within the mineralization tank. The nitrifying process causes the pH of the water to decrease, that is to become more acidic. However, the nitrification process itself depends on the pH of the environment. This relation is shown in Figure 2.1.

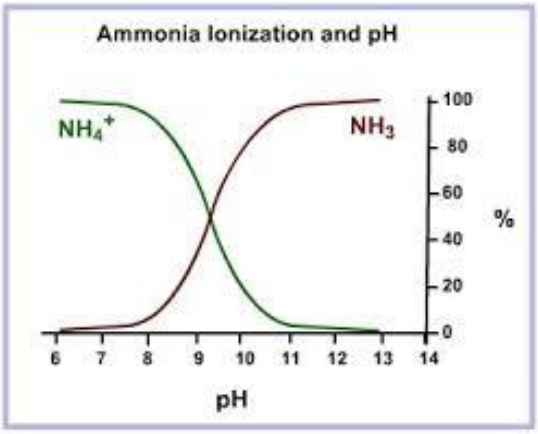

Figure 2.1. Ammonia ionization capability based on pH

Nitrification can be modeled mathematically and is widely used in the design of bioreactors, especially in water treatment plants. Active sludge modeling used in sewage treatment is used in designing nitrification bioreactors [9]. The modeling is carried out based on the mass balance equation shown in by Figure 2.2. Here a biomass, the amount of living organism in the system, feeds on a substrate to increase its biomass. If the substrate is increased and aerobic conditions are maintained, it would result in growth of the biomass. Similarly, if there is little amount of substrate and oxygen, the biomass will not feed on the substrate, resulting in decrease of biomass.

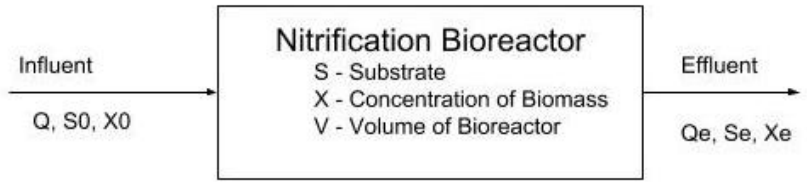

Figure 2.2. System Boundary used in Mass Balance Equation

An influent and effluent water flux travels across a system boundary as shown in Figure 2.3. Using the principle of conservation of mass, the mass balance equation of the system component can be derived for the specified system boundary. The quality of the water is determined by the amount of substrate and biomass flowing across the system boundary. A nitrification bioreactor of an aquaponic system can be described in the context of this boundary.

2.2.1 Biomass Balance Equation

The net biomass within the system boundary is responsible for enhancing the nitrification process within the system. The rate of biomass in the bioreactor is given by the net flow rate of biomass into the system and the growth of biomass within the reactor, which feeds on the substrate. Equation 2.1 models the growth of biomass within the system boundary. The rightmost part on the right hand side of the equation is known as the Monod's equation. Monod's equation models the growth of biomass with respect the amount of substrate or food it has to consume.

$$\frac{dX}{dt}V = QX_0 - Q_eX_e + V\left[\frac{\mu_m S}{K_s+S} - k_d\right]X \qquad \textbf{(2.1)}$$

Where

X - Concentration of biomass in the bioreactor (mg/l)

V - Volume of the bioreactor (m^3)

X_0- Initial biomass concentration in influent water (mg/l)

X_e- Concentration of biomass in effluent water (mg/l)

μ_m- Maximum concentration of biomass in effluent water (mg/l)

S - Amount of Substrate for biomass to consume (mg/l)

K_s- Half velocity constant used in Monod's equation (mg/l)

k_d- Death/decay coefficient of biomass

2.2.2 Substrate Balance Equation

Similar to the derivation of the balance equation for biomass within the bioreactor, we can model the amount of substrate using mass balance equations. In an aquaponics system the substrate is ammonia from fish effluent. The substrate equation given by equation 2.2 models how the biomass consumes the ammonia (substrate) within the aquaponic system.

$$\frac{dS}{dt}V = QS_0 - Q_eX_e + V\left[\frac{\mu_m}{Y}\frac{S}{K_s+S}\right]X \qquad \textbf{(2.2)}$$

The rate of change of substrate within the bioreactor is the net amount of substrate moving into the bioreactor plus the amount of substrate utilized by the biomass within the bioreactor as given by equation 2.2. Here Y represents the utilization factor of the maximum possible yield [9]. Now we can construct the total nitrification model using the previous two mass balance equations. Nitrification occurs by first utilizing ammonia to nitrite using the bacteria called Nitrosomonas. Next another type of bacteria known as Nitrobacter converts the nitrites into nitrates. Therefore, the nitrification process within the bioreactor can be modeled by the system of equations given by equations 2.3.

$$\frac{dX_{ns}}{dt}V = -Q_e X_{ns} + V\left[\frac{\mu_{m,ns}\, NH_3}{K_{NH_3} + NH_3} - k_d\right] X_{ns}$$

$$\frac{dX_{nb}}{dt}V = -Q_e X_{nb} + V\left[\frac{\mu_{m,nb}\, NO_2}{K_{NO_2} + NO_2} - k_d\right] X_{nb}$$

$$\frac{dNH_3}{dt}V = Q(NH_3)_0 - Q_e(NH_3)_e + V\left[\frac{\mu_{m,ns}}{Y_{ns}}\frac{NH_3}{K_{NH_3} + NH_3}\frac{DO}{K_{DO} + DO}[e^{0.098(T-15)}][1 - 0.833(7.2 - pH)]\right] X_{nb}$$

$$\frac{dNO_2}{dt}V = Q(NO_2)_0 - Q_e(NO_2)_e + V\left[\frac{\mu_{m,ns}}{Y_{ns}}\frac{NH_3}{K_{NH_3} + NH_3} X_{ns} - \frac{\mu_{m,ns}}{Y_{ns}}\frac{NO_2}{K_{NO_2} + NO_2}\, X_{nb}\right]$$

$$\frac{dNO_3}{dt}V = Q(NO_3)_0 - Q_e(NO_3)_e + V\left[\frac{\mu_{m,ns}}{Y_{ns}}\frac{NO_2}{K_{NO_2} + NO_2}\frac{DO}{K_{DO} + DO}[e^{0.098(T-15)}][1 - 0.833(7.2 - pH)]\right] X_{ns}$$

(2.3)

We can see from the above system of equations; the control problem is nonlinear. The coefficients have been indexed to represent influent, effluent water and the types of bacteria for each of the steps in nitrification, namely Nitrosomonas and Nitrobacter bacteria. It can be seen that nitrification depends on the temperature, pH and biomass within the bioreactor. The equations show the importance of pH control in biochemical process control [10] and are the subject of this study.

2.3 Related works in pH control

pH control is an important process variable extensively used in many industries. Therefore a wide range of pH control schemes have been developed for different process control applications [11]. Different types of controllers used in industrial pH control and in control of bioreactors are presented in the following text.

2.3.1 PID based pH controllers

Many industrial systems employ PID controllers in the pH control process. Proportional-Integral-Derivative (PID) control is the most common control algorithm used in industry (Refer Figure 2.3). PID controllers are widely popular due to its robust performance in a range of operational conditions and its installation simplicity. PID algorithm consists of three basic coefficients; proportional, integral and derivative which are varied to get optimal response.

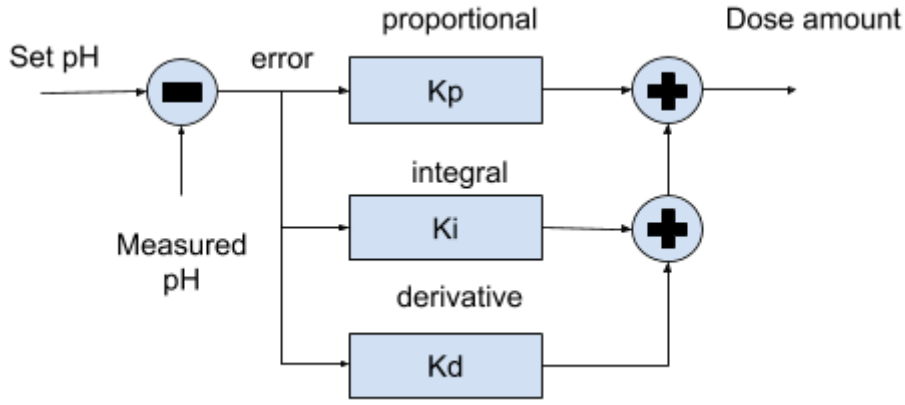

Figure 2.3 Block diagram of a typical PID controller used in process automation

The procedure followed to set the gain values of Kp, Ki & Kd to get the best response from the control system is known as tuning. Tuning sets the controller to its appropriate operating condition that the system operates without instability and delay.

PID controllers can be tuned using several standard methods [12].These are namely:

- Ziegler Nichols method
- Cohen Coon method
- Relay method
- PID tuning software

Other forms of adaptive tuning methods have also being developed in order to tune the parameters [13]. If the tuning is not performed correctly the system may operate slowly or even be unstable. Therefore, tuning is an important part of PID based controlling.

Limitations of PID control – PID controllers in general do not provide optimal control [14]. The major drawback with PID control is that its feedback system uses constant parameters and the controller has no knowledge of the process and only performs reactive control.

Another limitation of PID control is that in order to design a PID controller, a non-linear system needs to be linearized. This can cause approximation errors which can be neglected in many applications. However, in application such as pH control of bioreactors, these approximations can cause the system to perform sub optimally.

Furthermore, PID controllers under performs when the system is asymmetric. The pH control problem is an asymmetric system where a base is dosed into the system to raise the pH, but if the pH overshoots, we have no actuation to bring the pH down. In such case the PID controllers needs to over damped, and hence not optimal.

Solutions to these limitation are available, for example, techniques such as gain scheduling where a series of stored controller settings are used at different operating zones and adaptive techniques to automatically tune parameters by using fuzzy, neural networks or even using machine learning [15] . These are ad hoc fixes to inherent problem with PID controllers which cannot be ignored in certain applications such as pH control.

2.3.2 Fuzzy logic based pH control

Fuzzy control schemes have been widely used in pH process control applications [16]. Fuzzy logic is a branch of artificial intelligence that deals with modeling the controller with expert knowledge using fuzzy variables and membership functions that directly capture expert knowledge and expert actions. Fuzzy logic was introduced as an

alternative form of logic [17]. This alternative form of logic is many valued and is designed to accommodate partial truths into logical reasoning. The underlying concept in fuzzy control can be expressed in three main steps

- Fuzzification - This is the process of converting all inputs into membership values.
- Execution - The process of executing the rule base generally provided by an expert.
- Defuzzification - The process of generating the outputs from the membership functions.

Fuzzification is generally achieved with the use of membership functions. A membership function is a function that indicates the level of a certain truth. For example, we can define a membership function call warm that includes a range of temperatures and provide a value that closely fits the definition of warm. A temperature reading of 32 degrees Celsius can have a higher degree of membership to warm than a temperature reading of 25 degrees Celsius. Similarly, we can define several membership functions for different truths related to temperature. These membership functions are subjected to fuzzy operations such as min operation and max operation which are the equivalent fuzzy operations for logical AND & OR operation respectively. Different membership functions would be evaluated based on a rule base. The outputs from these fuzzy operations would then be used in defuzzification.

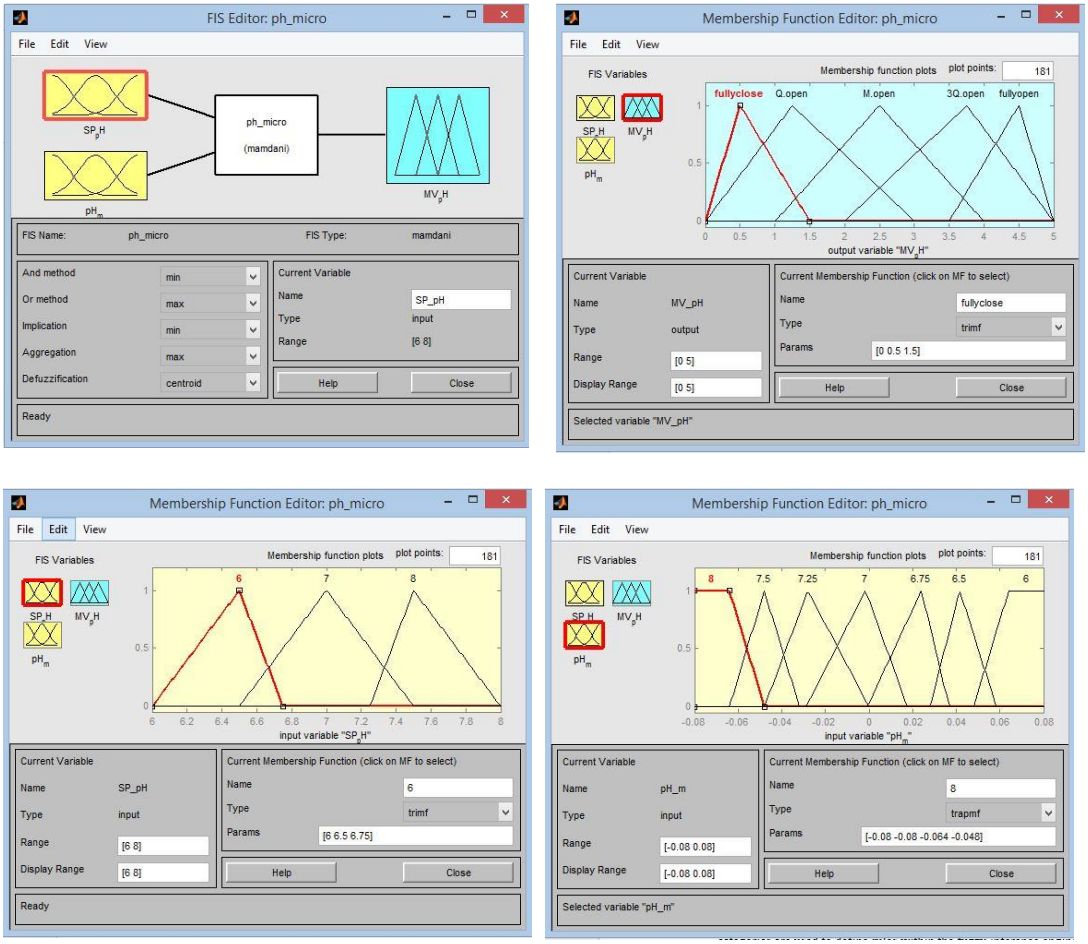

Figure 2.4 Fuzzy controller overview (top left), output membership function (top right) & input membership function of a fuzzy based pH controller designed

Defuzzification can be performed in many ways, but the most popular ones being the centre of gravity methods. These methods essentially convert the membership function into a value that can be used to control actuators. An implementation of a fuzzy controller for pH control is provided in Appendix I. The controller uses fixed membership functions based on expert recommendations and simulated in Simulink as show in Figure 2.4. The design of the controller was straight forward, however tuning the controller on site was cumbersome.

2.3.3 Adaptive Neuro Fuzzy Inference Systems

Fuzzy control provides a good method of representing knowledge in a structured manner. However, this knowledge needs to be explicitly written or embedded into the system. Thus fuzzy logic alone cannot mimic the natural learning observed in nature. In order to provide such capabilities fuzzy logic control has been successfully intertwined with neural networks to produce adaptive neuro-fuzzy inference systems, ANFIS [18]. Figure 2.5 shows a diagrammatic representation of an ANFIS system. It is important to note that rules are embedded into the neural network structure and usually represented as a Multilayer Perceptron. A noticeable limitation of using ANFIS is that the system should be a Takagi-Sugeno type inference system for the controller to be implemented as a neural network. ANFIS do not perform general learning, as the rules & membership function must be explicitly defined and carefully embedded into the neural structure, however it is capable of learning the degrees of membership and ranges occupied by each membership function.

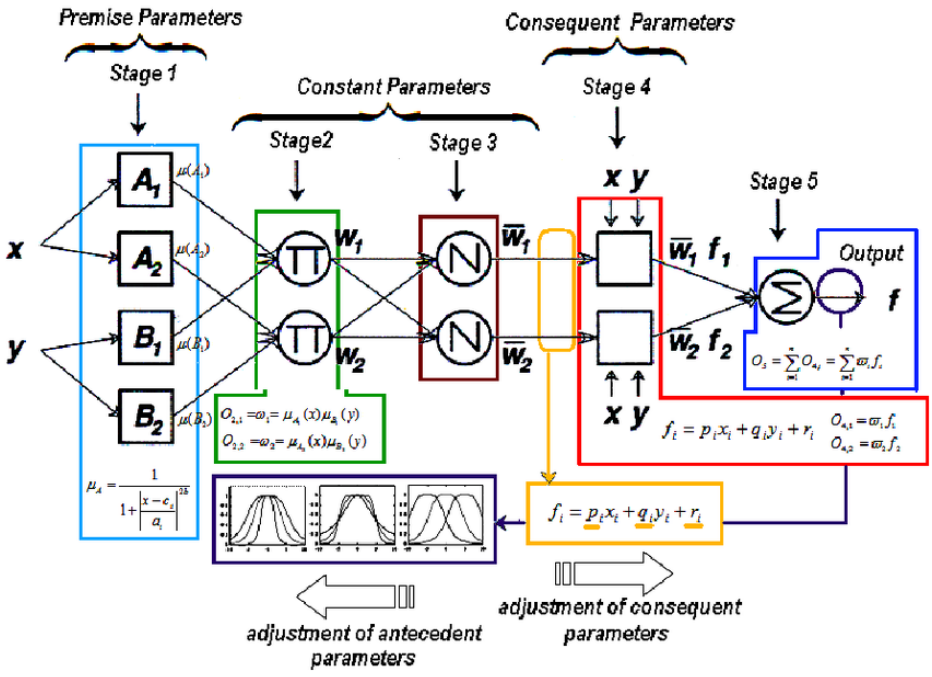

Figure 2.5 ANFIS architecture

2.3.4 pH controllers based on optimal control

In optimal control theory, a dynamic system is operated in a manner that reduces a cost heuristic. It determines what sequence of actions need to be taken to fully optimize the process in question. Optimal control techniques usually employ a heuristic known as a cost function as shown in equation 2.4 to evaluate the optimality of the controller. The objective of the controller is to generate a control law in a manner that reduces the overall cost or maximize the accumulated reward over time. The use of such control are essential in a variety of non-linear control engineering applications such as bioreactor controllers, space exploration vehicles and guidance systems.

$$\mathrm{J} = \varphi[x_0, t_0, x_f, t_f] + \int_{t_0}^{t_f} [\mathrm{x(t)}, \mathrm{u(t)}]\mathrm{dt} \quad \textbf{(2.4)}$$

Subjected to first order dynamics of states (state space) $\frac{dX}{dt} = F[x(t), u(t), t]$ *; path constraints* $B[x(t), u(t), t] \leq 0$*; and boundary conditions* $[x_0, t_0, x_f, t_f] = 0$ *where x(t) is the state, u(t) is the control, t is the independent variable time t,* t_0 *is the initial time and* t_f *is the terminal time. The term* φ *signifies the endpoint cost and the L signifies the Lagrangian, which gives the trajectory of the current solution.*

The optimal control solution can be derived using the Pontryagin's maximum principle or by solving the Hamilton-Jacobi-Bellman equation in continuous time applications [19]. The main problem with nonlinear optimal control is that the solution requires to solve the nonlinear Hamilton–Jacobi–Bellman (HJB) equation shown by equation 2.5. The cost Equation 2.4 is used with HJB to find the optimal control solution.

$$V(x,t) + \min_u \{\nabla V(x,t) \cdot F(x,u) + J(x,u)\} = 0 \quad \textbf{(2.5)}$$

Subjected to terminal condition $V(x,T) = D(x)$; where D(x) gives the economic value of the final state.

The HJB equation is a nonlinear partial differential equation (PDE) and cannot be solved using analytical methods. In order to solve this problem dynamic programming (DP) methods were developed. DP methods generally provided solutions backward-in-time which makes the numerical implementation expensive with higher dimension of nonlinear systems. Werbos [20] presented a technique called as approximate dynamic methods where approximate solution to equation 2.5 is given forward in time with the use of neural networks as a function approximate of the solution.

Solution to these equations can be used to optimize continuous, discrete and stochastic systems in order to generate an optimal control law. The application of optimal control with stochastic processes involving process noise can be analyzed using stochastic estimation and likelihood. Techniques such as Linear Quadratic Optimization (LQO) have sprung from the application of stochastic estimation and control to yield better controllers. Optimal controllers use Markovian Decision Processes (MDP) to solve the optimal control problem. MDP [21] is a theoretical formulation of a series of state transitions within the problem's state space and is a tuple of 4 {S',P,S,A}that completely describes the behavior of the system in where state transition are deterministic.

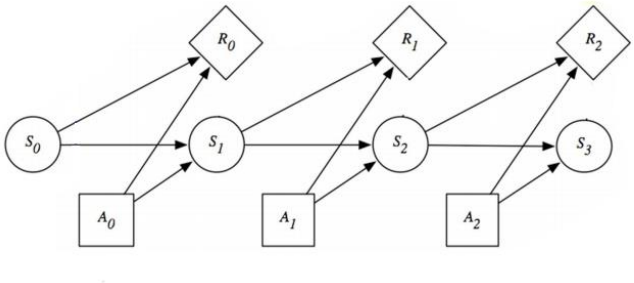

Figure 2.6 Graphical representation of a Markovian Decision Process

The control problem can usually be stated as a MDP, as shown in Fig. 2.6, and then solved using optimal control. In other words, MDP is a generalization of the state space used in deriving the HJB equation and its cost function. Generally, the states of the MDP are assumed to be fully observable and a given state can be directly identified. But in many real world applications the states of system are not fully observable. Such a system is known as a Partially Observable Markovian Decision Process (POMDP). The POMDP is defined by a tuple of 5 {S', P, S, A, O}. POMDP have less knowledge of the plant and therefore estimates the unobserved states of the plant. Figure 2.7 shows a graphical representation of a POMDP. MDPs & POMDPs are important analytical tools in the design of estimators and controllers for optimal control problems. Most systems have unobserved system states due to constraints in sensing or due to noisy sensory readings. Furthermore, the amount of actuation/controllability in real world systems is quite limited thereby making POMDPs well suited for modeling real world problems.

In an ideal case the principle of optimality assumes that the system is completely observable for each state. However as most real problems are not completely observable, we have to compensate in order for the principle of optimality to hold. In such cases the use of belief is incorporated in the design [22]. Beliefs are learnt truths based on a sequence of information and are enforced using belief functions. Belief functions are approximate functions that can be used to determine the certain truths about the system. Recurrent Neural Networks (RNN) are ideal for representing belief functions and are used in deep reinforcement learning based algorithms.

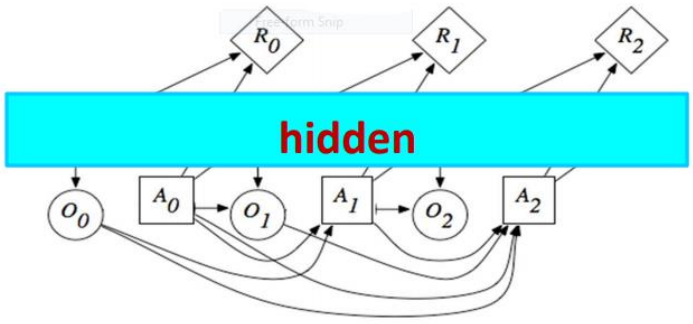

Figure 2.7 Graphical representation of a Partially Observable Markovian Decision Process

2.4 Deep Reinforcement Learning Techniques

Advances in deep reinforcement learning has shown that human like control can be achieved using DRL algorithms. Previously, DRL algorithms require large servers for training the AI agents, but state of the art algorithms such as Asynchronous Advantage Actor Critic (A3C) have proved that these algorithms can be implemented with lesser computation power. The interaction between the RL agent and the environment in a typical reinforcement learning problem is shown in Figure 2.8

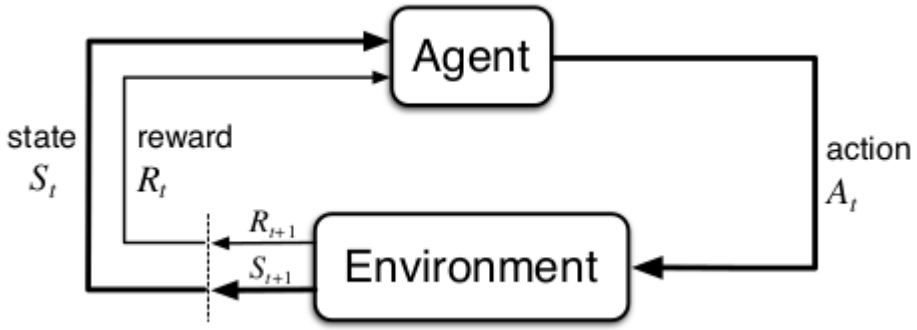

Figure 2.8 Agent environment interaction in a reinforcement learning problem

2.4.1 Dynamic Programming in the Context of Reinforcement Learning

Dynamic programming is the method of optimizing a given policy (program) of a temporal or sequential component of the problem. Dynamic programming methods are used to solve complex problems, especially optimization problems, by breaking the main problem into sub problems and combining them in order to solve the entire problem. Dynamic programming provides a generalized solution for problems that are optimal in substructure and when the sub problems are overlapping each other. This allows to break down the main problem into smaller sub problems and solve each of these sub problems for optimality. The overlapping nature of the sub problems allows us to merge the sub solutions to construct the solution for the main problem.

A Markovian Decision Processes (MDP) can satisfy both properties. These two properties are evident from the Bellman's equation (equation 2.6) which is the solution to the Markovian Decision Process. The recursive part of the Bellman's equation provides a recursive sub structuring of the optimization problem and the use of a value function allows caching and reusing in a manner that satisfy the condition of overlapping sub problems.

Dynamic programming methods require an explicit model [21][23]. Once the model of a system is fully known, optimality can be achieved using the Bellman's Equation. This is in contrast to the reinforcement learning problem where the model of the system is not known and it is up to the algorithm to learn the model based on its interaction with the environment. However, dynamic programming methods form the basis for reinforcement learning methods with modification to make model free reinforcement learning possible. Important concepts in dynamic programming techniques that are fundamentally important to the reinforcement learning problem are presented in this section.

Policy Evaluation: - Policy evaluation is also known as prediction, where the state-value function is determined. In policy evaluation we input a MDP and a policy π and get a value function for that particular policy V. The value function is generally defined for a particular policy in which the agent interacts with. The value function is used to determine how good the current policy is. We start off with an arbitrary initialized value function and update using the Bellman's equation as given in equation 2.6.

$$V_{k+1}(s) = E[R_{t+1} + V_k(S_t)|S_t = s] \quad \textbf{(2.6)}$$

As can be seen from the above equation, policy evaluation uses synchronous updates to update the value function. At each k+1 iteration and for all states the value function is updated using the next possible states.

Policy Improvement - Let's assume that we take a particular action not recommended by the current following policy. If this action produces a value function that is greater the current value function, then we can switch from the current policy to another. The policy improvement can be represented by the following equation. Let π and π' be a pair of deterministic policies where, for all s ∈ S

$$q\big(s, \pi'(s)\big) \geq V(s) \tag{2.7}$$

Then we can conclude that the new policy π' must be better than or equal to current policy π. In other words, the expected return of the new policy from all states s ∈ S must be greater than or equal to expected return from current policy. This condition is given in equation 2.8.

$$V'(s) \geq V(s) \tag{2.8}$$

Policy Iteration - Policy Iteration is a two-step process, where the current policy is evaluated and the resulting value function is used to improve the policy [24]. Once a policy, π, has been improved using v to yield a better policy, π', we can then compute v' and improve it again to yield an even better policy, π''. We can thus obtain a sequence of monotonically improving policies and value functions. This process of policy iteration always converges to the optimal policy. Policy Iteration is also known as the control problem, where the optimal control strategy is determined. The following diagram summarizes the policy iteration technique.

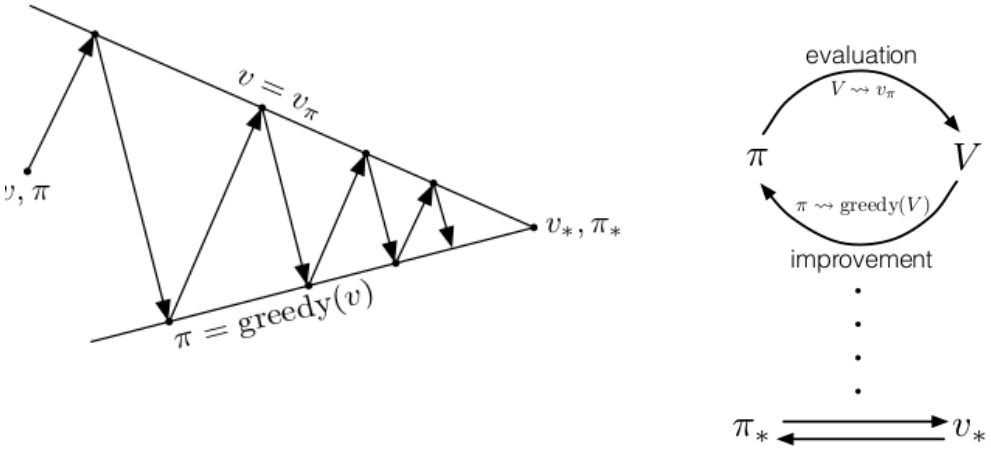

Figure 2.9 Pictorial representation of policy evaluation & improvement (right) and value iteration (left)

Value Iteration - Value iteration is much like Policy evaluation, but it requires the maximum to be taken over all actions. Value iteration effectively combines one sweep of policy evaluation and one sweep of policy improvements. An important intuition used to explain value iteration is that in value iteration, the recursive backups are performed from the goal state towards the current state that is being evaluated. Value iteration uses the Bellman's optimality equation for determining the value function. Intermediate value function may not indicate any policy.

$$V_{k+1}(s) = \max_a E[R_{t+1} + V_k(S_t)|S_t = s,\ A_t = a] \quad \textbf{(2.9)}$$

Generalized Policy Iteration – Generalized Policy Iteration [25] refers to the technique of policy evaluation interacting with policy improvement and vice versa. In generalized policy iteration the value function is made consistent with the current policy and the policy made consistent or greedily exploited with the respect to the current value function [26]. In policy iteration, several policy evaluations follow a policy improvement stage, but with value iteration, only a single policy evaluation is performed between policy improvements.

Almost all reinforcement learning methods use the concept of a generalized policy iteration. These reinforcement algorithms utilize an identifiable policy and value function, where the policy is improved with respect to the value function and the value function improved to become the value function of the policy. A major drawback of dynamic programming methods is that they require operations over the entire state space of the MDP. This can be very expensive when the state space is large. Asynchronous dynamic programming methods uses iterative methods where entire sweeps of the state space are not required. These algorithms update values of the states asynchronously using the most recent observations. These algorithms do not update all states at once but may update some states several times before updating the other states. This makes implementation of the algorithm computationally efficient. However, in order for the algorithm to be stable and to converge properly all states should be updated during some point.

2.4.2 Summary of Dynamic Programming Methods

Table 1 gives a summary of different dynamic programming schemes used with reinforcement learning [27]. Dynamic programming problems can be divided into estimation or control problems. A controller could be designed using both these techniques but the extent to which policy iteration and value iteration is used differs from one another. Therefore, it is up to the designer to select the correct method to better suit the problem. However, both these techniques are incorporated into GPI algorithms making them general purpose algorithms.

Table 1: Summary of Dynamic Programming Methods

Problem	Bellman Equation	Algorithm
Estimation	Bellman Expectation Equation	Iterative Policy Evaluation
Control	Bellman Expectation Equation + Greedy Policy Improvement	Policy Iteration
Control	Bellman Optimality Equation	Value Iteration

2.4.3 Monte Carlo Learning

It was shown previously, that solving the Bellman's equation provides the optimal solution for a finite Markovian Decision Process. Direct solution of the Bellman's equation is however difficult. In such cases the solution can be derived stochastically. Interactions with the environment can be used to generate experience and the expectations of these experience can be used to estimate the value function. This approach in solving complex and algorithmic problems falls under the general category of Monte Carlo algorithms [28].

The Bellman's equation can be solved elegantly by approximating the dynamic programming problem using Monte Carlo methods. Learning from experience allows the RL agent to operate without prior knowledge of the system [29]. Monte Carlo methods solves the reinforcement learning problem based on averaging sample rewards. In general, Monte Carlo algorithms allow us to solve the dynamic programming problem without explicitly knowing the state transition probability matrix of a finite MDP. Algorithm 1 of appendix E shows the Monte Carlo method to update the value function. Experience gathered in episodes is averaged in order to estimate the state-value function. For example, the value function is the average reward observed at each state experienced in all of the episodes. Each state maintains a list (memoization) of observed rewards. In each episode the observed reward is appended

to the respective state's reward list. The resulting value function for that state is the average of the rewards it has seen throughout its experience.

Similarly, we can estimate a action value function instead of a value function. The difference is that the heuristic of how good the current operating point depends on both the state and action. The estimation problem is to determine and learn the heuristic value function that determines or how good the current action value function or state value function is.

In Monte Carlo control, the same approach will be used to determine the control strategy or the policy to achieve the optimal control strategy. In Monte Carlo control, the policy and the value function is approximated by a function approximate. Just as with dynamic programming methods, the value function is improved and updated based on the current policy. This policy is then improved based on the updated value function, as suggested by the right side diagram of Figure 2.9. This is also known as generalized policy iteration but within the context of reinforcement learning. Algorithm 2 (appendix E) shows how Monte Carlo algorithms are used to determine action value function.

In Monte Carlo control, an arbitrary policy and an arbitrary value function is initialized. Next an episode is generated using the current policy until it reaches a terminal state. The state value function is then updated using accumulated rewards. We define a policy where we chose an action that maximizes the state-action value function q. This is the generalized policy iteration concept, where the value function is used to evaluate the policy and then improve the policy.

Based on the studies carried out, we can use Monte Carlo along with approximate DPs to successfully solve complex POMDP control problem. An aspect of Monte-Carlo methods is that the estimates for each state are independent. Monte Carlo methods do not bootstrap and hence a particular state do not make estimates based on the estimate of any other state. The use of extensive bootstrapping is the main idea on TD learning and will be dealt in the following section.

2.4.4 Temporal Difference Learning

Temporal Difference (TD) combines ideas from Monte-Carlo and Dynamic Programming. Like Monte-Carlo methods it is capable of learning from experience and like dynamic programming methods, it updates estimates in-part based on other learned estimates. The use of estimates to update estimates is usually known as boot-strapping. The advantage of TD is that it is inherently implemented as an incremental on-policy algorithm [24]. The main distinguishing feature of TD is that it uses bootstrapping where an estimated value is used to determine another estimate. The simplest TD method makes the update immediately to transition to $S_{t+1}$and receiving R_{t+1} as shown by equation 2.10.

$$V(t) \leftarrow V(S_t) + \alpha\big[R_{t+1} + \gamma[V(S_{t+1}) - V(S_t)]\big] \quad \textbf{(2.10)}$$

This is in contrast to Monte Carlo methods that update its value function at the end of each episode, when the final goal state reward has been achieved as given in equation 2.11.

$$V(t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \quad \textbf{(2.11)}$$

The TD estimation and TD control methods are presented by algorithms 3 & 4 respectively in appendix E. The pseudo code in algorithm 3 shows how TD estimation is carried out and how the value function is updated for each step within an episode. TD control is based on the generalized policy iteration but uses TD methods to update the value function and the policy as given by algorithm 4. The update rule for online TD control is given by equation 2.12. This algorithm is known as the SARSA algorithm. This is because the quintuple consisting of, current state, current action, current reward, next state and next action are used to estimate and update the action value function.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big[R_{t+1} + \gamma[Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]\big] \quad \textbf{(2.12)}$$

The off policy algorithm given by algorithm 5 is one of the early breakthroughs in reinforcement learning, widely known as Q learning. Here the action-value function, Q, directly learns the optimal action-value function, Q_{max}, even though the current policy that is executed may not be the policy that is approximated by the action-value function Q. This relation is shown in equation 2.13.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \left[\max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \right] \quad \textbf{(2.13)}$$

2.4.5 Policy Gradient Methods

Policy gradient methods do not employ a value function that estimates how good a state is. Policy search methods directly model the policy and update the policy model in order to maximize the expected reward from executing the policy and are successfully implemented using gradient and gradient free methods. The policy network outputs the means and standard deviation of a probability distribution. These are then used to generate the actual actions in continuous space. In the case of discrete actions, the policy outputs a probability for each discrete action and the action with the highest probability will be selected as the next action.

Policies can be improved using gradients, but in order to compute the expected return, the average has to be taken over entire plausible trajectories under the current policy. However, such averaging requires either deterministic approximations or approximations made from stochastic sampling. A deterministic estimate can only be derived from a model-based setting and is not applicable for model free learning. Monte-Carlo methods are therefore used in model-free settings as it provides stochastic alternatives to estimate the expected return. However, there is a caveat in using gradient learning methods with Monte Carlo methods as the gradients cannot pass through sampled values of a stochastic function. In order to circumvent this problem, an estimator is used to estimate the gradient using a maximum-likelihood-ratio estimator or commonly used in RL terminology as REINFORCE rule [30].

Policy search methods tries to find a policy directly using gradient based or gradient-free methods. Most gradient free methods use evolutionary algorithms to learn a policy directly whilst back propagation still remains as the most popular gradient based learning approach. The use of evolutionary algorithms with large populations or agents with large number of parameters is quite expensive but it is capable of optimizing arbitrary, non-differentiable models and allows for more exploration in parameter space.

Direct policy searching using neural network with large parameters is difficult and it can converge to local maxima. A particular solution to this problem is to use guided policy search (GPS), where an optimized controller is used to teach the neural network in a supervised manner and then combined with importance sampling [31]. This prevents the policy from being stuck in a local maxima and biases the search towards a good optimum.

2.4.6 Actor Critic Methods

Methods that blend the use of value functions and policy gradients are generally known as actor-critic methods [32]. The actor and critic emulates the policy and the value function respectively. This when used with policy gradients reduces the high variance in policy gradients. The actor "Policy" learns by feedback from the critic "Value Function". In doing so it reduces the problem of high variance in policy gradients. These methods use policy gradients. Actor-critic methods are capable of combining policy gradient methods with learned value functions effectively and are generally trained based on rewards and TD errors [33]. Actor Critic methods can converge quickly as it is sensitive to improvements in policy gradients methods as well as improvements in value function methods.

2.4.7 Curse of Dimensionality

The curse of dimensionality refers to phenomena that occur when analyzing data in high dimensional space. As far as optimization is concerned, the objective function must compute for each combination of values. This can be problematic with higher dimensional data. Neural networks provide an ingenious solution to overcome the problems of high dimensionality as it can represent complex objective function and can be evaluated with little computational power [34].

2.4.8 The Deathly Triads

The danger of divergence and instability occurs in reinforcement learning due to the combination of all of the following three elements, known as the deadly triads.

- **Function approximation** – Estimating the state space using function approximate such as neural networks, non-liner function, tables etc. It also enables to model the control with features rather than fully observable states of the control problem.
- **Bootstrapping** – Updating the value function and policy based on existing estimates rather than waiting for rewards based on actions.
- **Off-policy training** – Training the value function based on a different policy that the policy that is being currently followed.

It is important to note that the combination of all of the above makes the learning process and if one of the above is excluded then the process is unstable. Function approximation is critical in solving POMDPs and when the number of states is very large. Therefore, it cannot be avoided. Bootstrapping can be avoided at the cost of memory and computational costs. In most applications, on-policy learning will suffice. Therefore, as far as control problems are involved we can use on-policy techniques instead to avoid instability.

2.4.9 Activation Functions for neural networks

Activation functions are mathematical models of neurons firing. Different mathematical models have been developed based on neurons of different parts of the brain [35]. Therefore, proper selection of activation functions must be carried out when designing neural architectures for different requirements. A comparison of commonly used activation functions are shown in Figure 2.10.

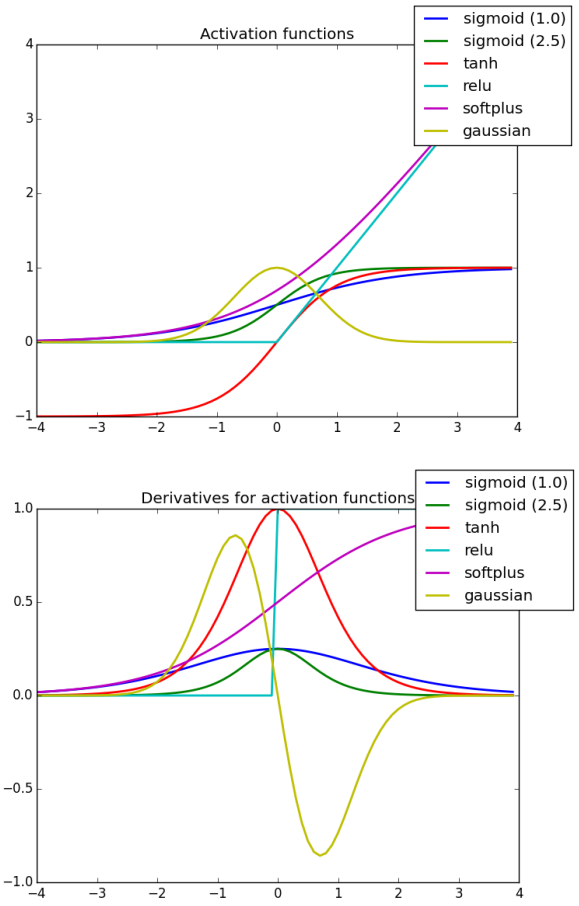

Figure 2.10 Comparison of activation functions

- **Sigmoid** - Sigmoid is a popular activation function used in neural network designs. The sigmoid function is particularly useful in classification and regression problems and is extensively used in supervised learning

applications. A family of sigmoid functions can be generated and selected in order to streamline activation.

- **tanH -** tanH is also a s-shaped activation function like the sigmoid function, but with a larger range. The main distinguishing feature of tanH is that negative inputs will be strongly mapped negative. This is also used in classification problems. The function is monotonic, but its differentiation is non-monotonic.

- **Rectified Linear Unit (ReLu) Activation** - This is a newer but simple activation function derived by studying biological activation of visual cortex neurons. RNNS are better approximated and trained without instabilities using the ReLU activation function. Both the ReLu function and its derivative are monotonic.

- **Softplus/Softmax** - Softmax is a normalized exponential function and is a generalization of the sigmoid function. The output of the softmax function is capable of representing the probability distribution of K possible outcomes. In most cases, softmax function is used in the classification problems and usually trained with cross entropy. In reinforcement learning the softmax function is used in selecting the next action based on the probability distribution of the action space.

2.5 Summary

It was seen from the literature, that currently no academic work has been performed in automated control of pH in aquaponic system using reinforcement learning. However, pH controlling schemes have been studied in water treatment processes that can be applied in aquaponic system. The literature also pointed out that PID controllers have limitations when controlling highly non-linear processes such as pH control in aquaponics. Optimal control techniques have been used in designing pH controllers, but they lack in the performance and stability promised by the state art deep reinforcement learning algorithm. Therefore, deep reinforcement learning concepts were extensively studied. Based on the literature reviewed, the research gap addressed by this study is to design a controller using state of the state art deep learning algorithm to control the pH of an aquaponic system. Table 2 shows individual gaps in research identified to construct the research problem of this study.

Table 2: Summary of literature review and identified research gaps

Key Points	**Findings**	**Gaps**
Application of Deep Reinforcement Learning in linear process control applications.	Developed an artificial intelligence based approach to linear process control using deep reinforcement learning.	The use of deep reinforcement learning in the design of controllers for nonlinear process controlling is yet to be studied.
Human-level control through deep reinforcement learning	Demonstrated a single architecture can successfully learn control policies in a range of different environments with only very minimal prior knowledge	Current work in deep reinforcement learning is confined to simulations and computer games. Applications in industrial processes are to be done.

Limitations of PID control	PI-PD controller, corresponding to PI control of a plant transfer function changed by the PD feedback can pro- duce improved control in several situations	Tuning & PID architectures s for non-linear process control is not optimal.
Asynchronous Methods for Deep Reinforcement Learning	Asynchronous versions of four standard reinforcement learning algorithms	Useful algorithms to be used in industrial control applications, currently implemented only in computer simulations & games
Predicting Periodicity with Temporal Difference Learning	A TD agent using complex discount rates can identify periodic patterns in the return	Can periodic process be automatically identified in an industrial process?
Continuous control with deep reinforcement learning	Continuous control using deep reinforcement learning.	Study performed in simulated environments
Natural Value Approximators: Learning when to Trust Past Estimates.	A method that learns how to combine a direct value estimate with ones projected from past estimate	Is there an improvement in performance when using A3C algorithm with NVAs in industrial control?
Comparison of Linear and Nonlinear Adaptive Control of a pH-Process	Nonlinear adaptive controller works better than the linear one for tracking as well as for regulation purposes when infrequent disturbances in process feed occur	Deep Reinforcement learning based controller have not been considered in the comparison

CHAPTER 3 METHODOLOGY AND CONTROLLER DESIGN

3.1 Introduction

This chapter presents an overview of the methodology of this study, including the design of the software, hardware and the experimental setups to study the behavior of the DRL based pH controller.

3.2 Methodology

This research methodology is based on several refinements to the context of the study as a response to new observations and due to comments brought about during the progress reviews.

In order to study the performance of the controller, two experimental setups were designed. One setup was to evaluate the performance of the controllers for deterministic static conditions and the other to evaluate the performance in the stochastic dynamic system using the aquaponics system.

In the static case the controller was studied for its transient characteristic under a step response, whilst in the dynamic case the controller would be tested for its steady state performance. A digital PID controller was also implemented to validate and compare the results of the DRL controller.

The outline of the controller design is given below:

1. Comprehensive literature review & background study
2. Formulating the problem as a stochastic decisions process
3. Definition of a input to the pH controllers
4. Definition of outputs/action of the pH controller
5. Define the reward function heuristic

6. Determine model free control or model based control
7. Determine whether the actions are continuous or discrete
8. Determine the type of function approximation
9. Determine the learning mechanism
10. Selecting the tool chain for controller development
11. Selecting the inputs sensors and output actuators
12. Design Experimental setups for the following cases:
    a. In static system case – In a deterministic setup
    b. In dynamic system case – In a stochastic setup
13. Evaluating the performance of the controllers with respect to a digital PID controller.

3.3 Development of the Deep Reinforcement Learning based controller

The deep reinforcement learning approach uses deep neural networks in order to approximate the value function and the policy. Based on the studies carried out, recurrent neural network architecture is used to represent the value function. This is due to two main reasons, the first being that the entire process is treated as a partially observable Markovian decision (POMDPs) process. It was noted in literature that in such cases the concept of beliefs should be incorporated into the MDP so that Bellman's solution to the optimal problem can be effectively solved. A belief function is capable of estimating unknown states with a series of direct or indirect measurement related to system. A relatively straightforward method of employing beliefs into the reinforcement learning problem is to represent the approximation function using a recurrent neural network. A recurrent neural network is particularly good at handling time series data and inherently capable of learning dependence between samples. Therefore a recurrent neural network approximates the value function of the current state based on the previous sequence of observed state transitions, thereby giving a broader analysis of the current state.

3.3.1 Specification of Inputs and Outputs to the controller

The controller shall take in the measured pH value of the system as the input only. The output is determined by the DRL controller and it directly actuates a peristaltic dosing pump. The neural architectures are configured for this input and output. The use of pH alone allows to introduce partial information of the state space and to learn beliefs from a sequence of input pH measurements. Figure 3.1 shows the block diagram of the controller.

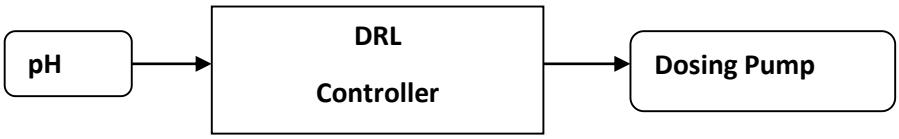

Figure 3.1 Relationship between the input and output of the DRL controller

The overall system is a negative feedback system. However, the DRL controller is designed to learn and model the system using experience. The experience is collected in the form of pH values and dosed amounts corresponding to observations and actions in the context of MDP and reinforcement learning.

The system of equation given by Equation 2.3 showed that pH, biomass and temperature are important parameters of nitrification in an aquaponics system. Practically in aquaponics system, pH is the more contributing factor, therefore, this study attempts to use the pH to observe the system and use the DRL controller to learn control policies to optimize the system with minimal amount of data.

3.3.2 Design of Critic

The critic for the proposed deep reinforcement learning controller will be based on a Recurrent Neural Network (RNN) with 10 steps. The recurrent neural network was necessary in order to mitigate the instabilities in learning due to control of a partially observable state space by introducing a belief into the learning process. This allows to learn a better value function that not only determines the current state based purely on observation but based on current and previous 9 observations. All activations for the critic are based on the Rectified Linear Unit (ReLu). This is known to be good in training recurrent neural networks and also because the critic attempts to estimate a function rather than perform a classification function in which case the softmax or sigmoid function is better suited. Furthermore, a basic RNN cell was used in creating the RNN network in contrast to using other types of cells such as Long Short Term Memory (LSTM) & Differential Neural Computer (DNC). Figure 3.2 shows the diagrammatic overview of the RNN used to model the critic that approximates the value function of the state space.

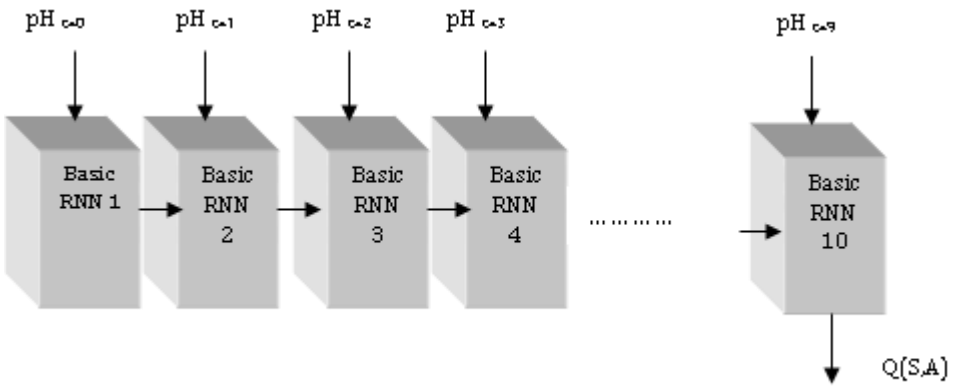

Figure 3.2 Recurrent Neural Network that approximates the critic

3.3.3 Design of Policy Network

The policy network produces a probability distribution of the next action based on current observations. The output of the policy network will then be used to select the next action given the current state in order to maximize the total reward. We can also consider a RNN based policy network, but we shall use a feed forward network as we can assume that the critic is capable of producing a good estimate of the current state. This also can speed up the execution of the reinforcement learning algorithm. Hence, the policy will be a feed forward neural network with two hidden layers and activated using softmax activation layers.

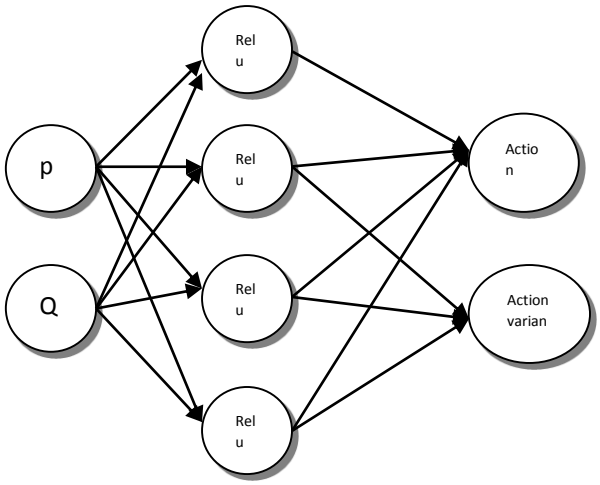

Figure 3.3 Neural network that approximates the actor/policy network

The policy network uses the current pH and critic as inputs to the multi-layer neural feed forward network. The two outputs from the policy network is combined using the normal distribution to produce the actual dosing value as shown in Figure 3.3.

3.3.4 Determination of Learning Rate

The learning rate plays an important role in training the neural network. Small learning rates makes learning slow to converge and thereby require more training steps to achieve the desired accuracy. In contrary, large training steps may cause oscillatory behavior in training, as seen in hill-climbing problems. Therefore, it is important to select an appropriate learning rate to achieve productive results. In our proposed system we use a learning rate of 0.01 based on previous training experience.

3.3.5 Learning and Gradient Descent based update

The update mechanism for the policy network and the critic network is performed differently. Policy gradient methods discussed in section 2.4.5 are used to update the parameterized neural policy network. The policy network updates the policy parameters in the direction suggested by the critic. This gives rise to the following update rule

$$d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi\left(a_i \middle| s_i; \theta'\right)\left(R - V\left(s_i; \theta'_v\right)\right) \quad \textbf{(3.1)}$$

The parameter of the critic is updating based on the action value function as shown below

$$d\theta_v \leftarrow d\theta_v + \ \partial(R - V(s_i; \theta'_v))^2 \ / \ \partial\theta'_v \quad \textbf{(3.2)}$$

Therefore, these updates will be used to update the policy and critic networks within the proposed DRL controller.

3.3.6 Selection of deep reinforcement algorithm

In order to represent the problem as a general purpose learning problem, the controller will be modeled as a model-free reinforcement learning based controller in continuous action space. Model-free control provides a general purpose learning technique for designing controllers without an explicit mathematical model of the system. Therefore, controllers can be designed purely using experience without supervisory training or an explicit mathematical model. In order to train the two neural structures, we shall employ a gradient based approach in contrast to a gradient-free method such as genetic algorithms. The recursive form of Bellman's equation would be used to update the neural network. A range of model free algorithms are compared in table 3.

Table 3: Comparison of different DRL algorithms

Algorithm	**Monte Carlo**	**Q-Learning**	**SARSA**	**DQN**	**A3C**
Policy	Off-Policy	Off-Policy	On-Policy	Off-Policy	Off-Policy
State Space	Discrete	Discrete	Discrete	Discrete	Continuous
Action Space	Discrete	Discrete	Discrete	Continuous	Continuous
Operator	Sample means	Q-Value	Q-Learning	Q-Learning	Advantage

Based on the above comparison, the A3C algorithm would be used in the design. The critic will approximate the value function and the actor shall produce actions that are continuous in time. This can be achieved using a feed forward neural structure that produces the mean and variance for each output action. The actual actuation signal is constructed using a probability distribution, function such as the Gaussian function, with the generated mean and variance. Actor-Critic methods provide stable and effective learning and have been shown to work well in a range of applications. Therefore, the controller would use the A3C actor-critic algorithm shown by Algorithm 6.

Algorithm 6 Asynchronous advantage actor-critic - pseudo code for each actor-learner thread.

//Assume global shared parameter vectors θ and θ_v and global shared counter
//Assume thread specific parameter vectors θ' and θ'_v
Initialize the thread step counter
repeat
Reset gradients: $d\theta \leftarrow 0 \; and \; d\theta_v \leftarrow 0$
Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
$t_{start} = t$
Get state s_t
repeat
Perform a_t according to policy $\pi(a_t|s_t; \theta')$
Receive reward r_t and new state s_{t+1}
$t \leftarrow t + 1$
T$\leftarrow T + 1$
Until terminal s_t ort $t - t_{start} == t_{max}$
$R = \begin{cases} 0 & for\ terminal\ s_t \\ V(s_t, \theta'_v) & for\ non-terminal\ s_t \end{cases}$
for $i \in \{t-1, \dots, t_{start}\}$ **do**
$R \leftarrow r_i + \gamma R$
Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i; \theta')(R - V(s_i; \theta'_v))$
Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial(R - V(s_i; \theta'_v))^2 \,/\, \partial\theta'_v$
end for
Perform asynchronous update of θ using $d\theta$ and θ_v using $d\theta_v$
until T> T_{max}

3.3.7 Designing the Reward Function

The reward function will be based on how close the current measured value is to the set point and the measured pH value. The set value generally needs to be kept at 7.2 based on studies performed on Nitrification. Initially, the reward function will be designed based on this information as shown by equation 3.3. However, different reward signals can be designed based on the different parameters. For example, the reward signal can be obtained with the use of an additional Oxidation Reduction Potential (ORP) probe that gives a vague indication of nitrification.

$$r = \begin{cases} 100 & if\ |7.2 - pH| \leq 0.01 \\ -100 & if\ |7.2 - pH| > 0.01 \end{cases} \quad \textbf{(3.3)}$$

3.3.8 Overall Architecture of the DRL controller

The overall system architecture including, software, hardware is represented in the block diagram shown in Figure 3.4.

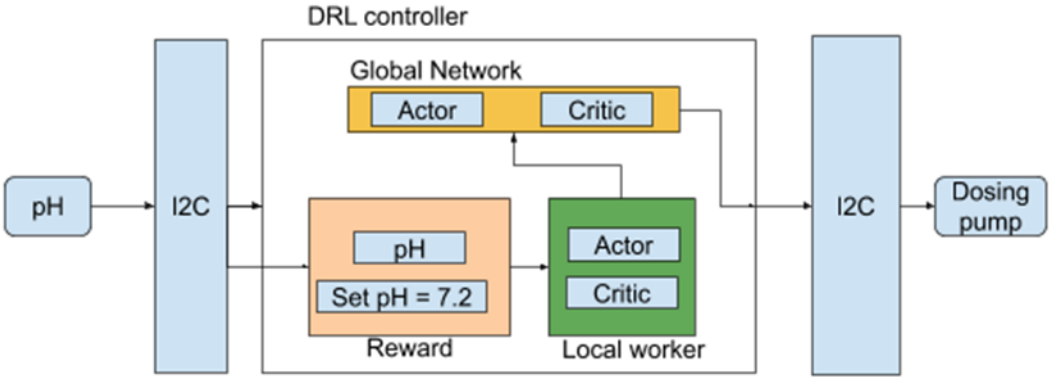

Figure 3.4 Overall system architecture of DRL controller and its peripherals

pH measurements are consumed by the DRL controller and are used to construct the reward signal based on the error. In this scenario the pH is set at 7.2 and the error is determined. This error is used with equation 3.3 to generate the reward signal. Elements of the A3C algorithm are represented by the global & local networks and each of these networks contains an actor and a critic network. Architecture of these networks were explained in detail in section 3.3.2 and 3.3.4.

3.3.9 Results based on empirical work

The UML diagram shown in Figure 3.5, presents the object oriented implementation of the controller. The ACNet class represents the controller containing the actor and critic neural nets. Each network is initialized during instantiation of the ACNet neural class within the constructor. All parameters required to configure each neural net is defined as static class variables or local/global variables. The ACNet class implements

several methods to construct concrete implementation of the computation graph of the neural networks.

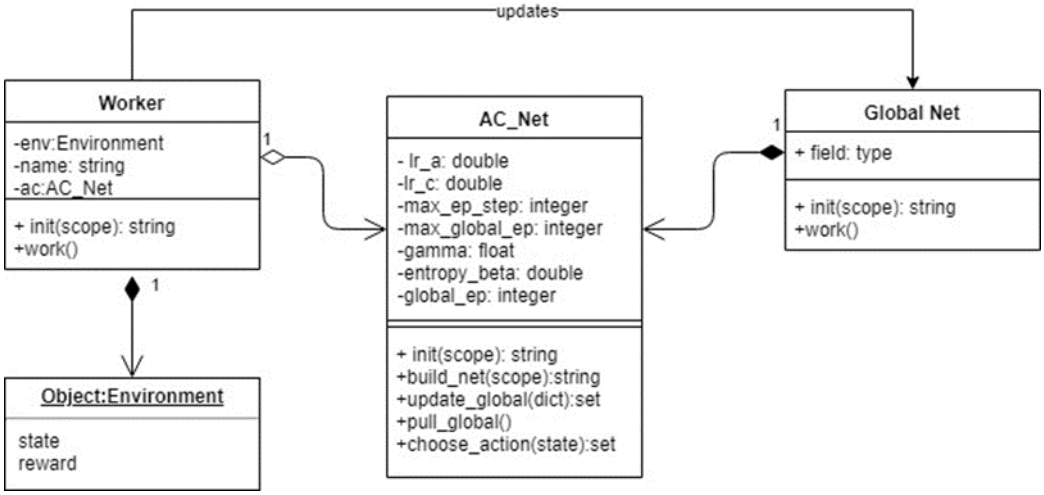

Figure 3.5 UML diagram of the DRL controller implementing the A3C algorithm

A visual representation of the controller implementation using tensorboard is shown in Figure 3.6. Tensorboard is a utility provided by google to visualize computational graphs used in designing neural architectures and in deep learning research. This neural architecture is purely based on the standard reinforcement learning architecture, where losses are optimally minimized. Python implementation of the controller is produced in Appendix II. The interaction between several worker networks with the global network is shown in Figure 3.6. (Four worker nodes each designated as W_0 to W3 and the Global node consists of two neural networks, an actor network and a critic network). When the A3C algorithm executes, the neural parameters of the global networks are asynchronously updated by the worker networks after several epochs.

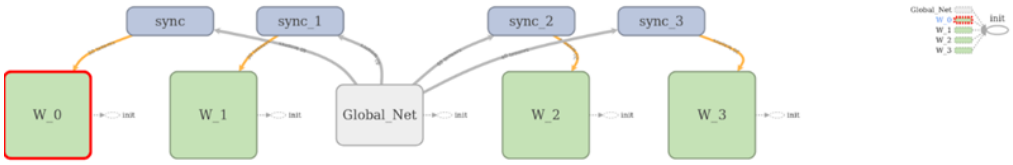

Figure 3.6 Visual Representation of the implemented DRL controller using Tensorboard

A detailed view of a worker node is presented in Figure 3.7. The actor and critic networks can be seen here. Observed readings and predictions from the neural network structures are used to generate TD errors. These errors are then used to form the actor losses, critic loses. The output from the actor is first bounded within an interval. This is then used with the Gaussian function to generate the actual dosing value.

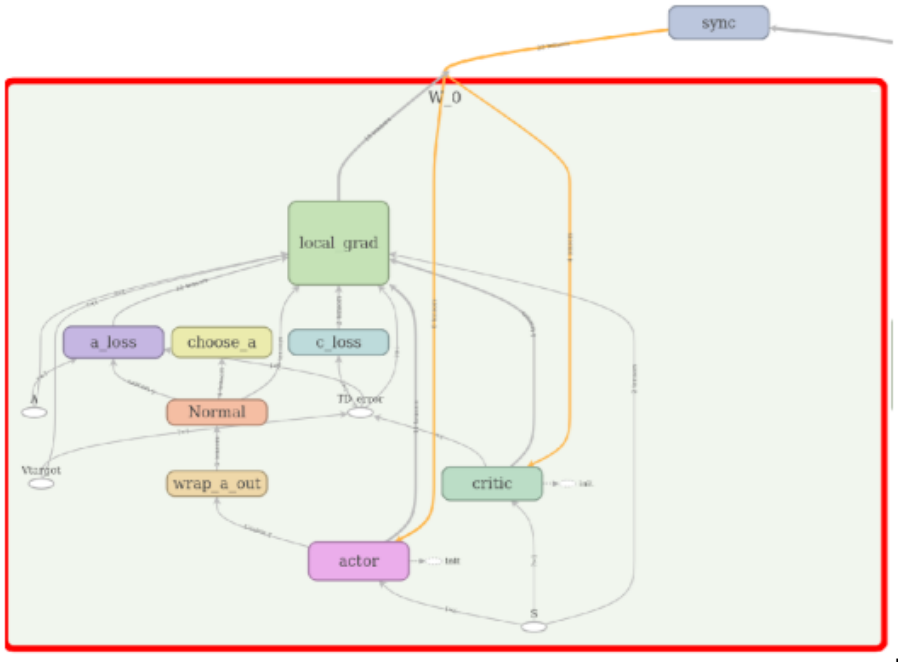

Figure 3.7 Internal networks of the DRL controller represented using Tensorboard

The learning process of the controller is shown in Figure 3.8. It can be seen that the actor and the critic components of the controller minimizes its entropy losses and converges towards zero. This shows that controller successfully learns and the critic is capable of accurately mapping the current state of the system within the learnt state space.

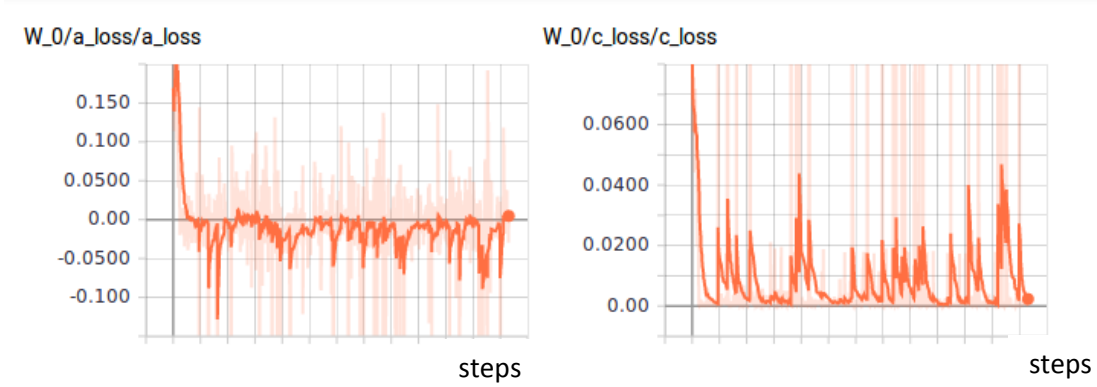

Figure 3.8 Training losses of the actor network and critic network plotted at each training steps.

The process of learning was observed using the total reward accumulated during an episode over several training epochs. The objective of reinforcement learning is to maximize total number of rewards over the duration of an episode. Figure 3.9 shows the variation of the total reward with each time step of an episode during an initial phase of controller learning. It can be seen that the reward signal produces large negative numbers. It is important to note that the design of the reward function has a considerable effect on the reward obtained using the controller. Therefore, it is important to consider vital parameters when designing the reward signal of the system.

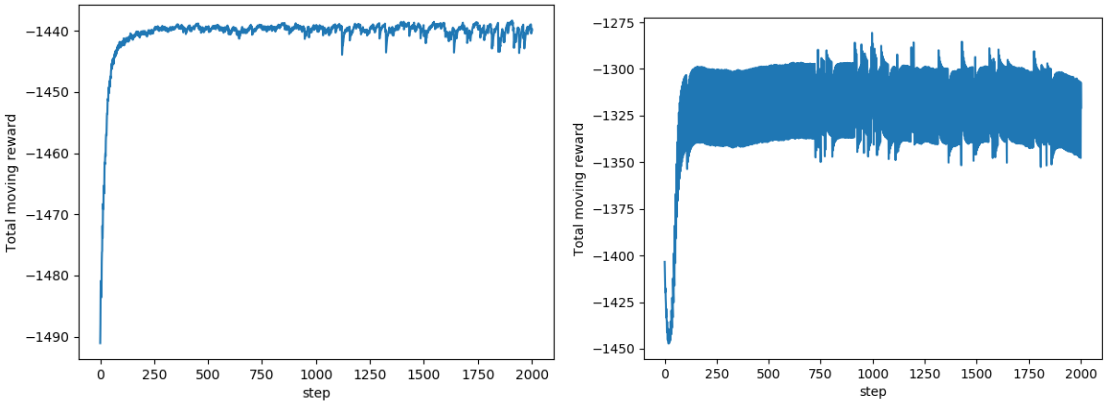

Figure 3.9 Total moving reward generated at two different epochs.

3.3.10 Tool chains and Development tools

As the controller is based on deep reinforcement learning, the software stack plays a major role. There are several software frameworks and the following table compare different software framework for developing deep reinforcement learning algorithms.

Table 4: Comparison of different software framework for implementing DRL controller

Software Library	Method	Production Level	Platform
Tensorflow	Computational graph	Yes	Single/Multi
PyTorch	Computational graph	No	Single
Torch	Computational graph	Yes	Single
Matlab	Matrix based	Yes	Single/Multi
Numpy	Array -based	No	Single

Based on the comparison given in table 4, Tensorflow was selected as the framework of choice for the implementation of Deep RL based controller. Tensorflow is an open source computational framework by Google. It has inherent advantages in machine learning, parallel computation and provide unparalleled networking features thereby

making it easy to run tensorflow code in production. Tensorflow operates by defining a computational graph and then executing this graph using a Session. Any computation within the solution is restructured in a manner that is presentable as a graph. This graph is an object, which a session can use to get output/s based on the given input/s.

3.4 Development of the PID controller

A digital PID controller is required to compare the performance of the controller. The controller takes in the pH readings from the Atlas sensor and finds the error between the reading and the set point of pH 7.2. The controller gains are important parameters of the PID controller. Therefore, in order to establish the required PID gain values a Matlab/Simulink model was used. This model is shown in Figure 3.10.

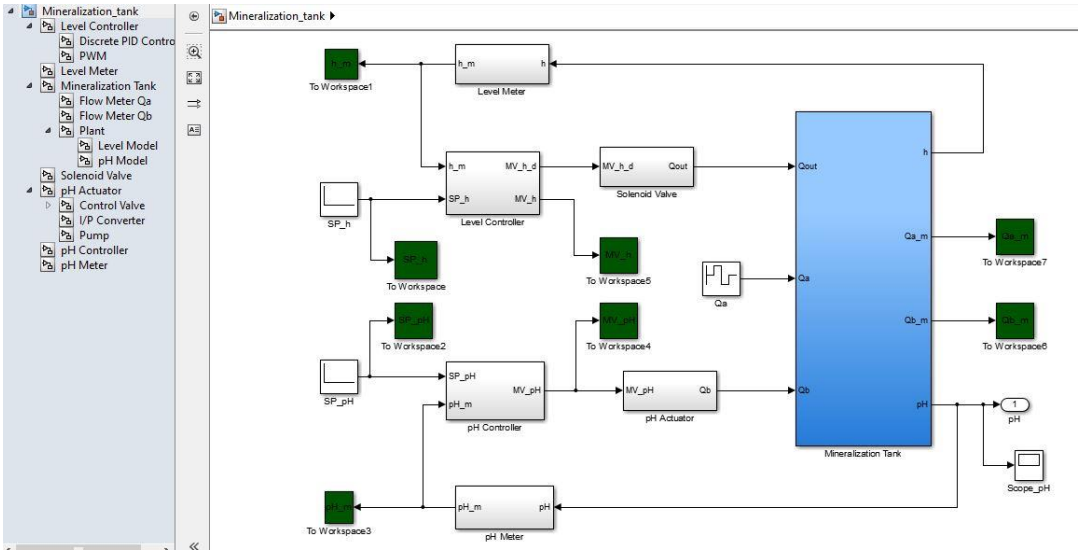

Figure 3.10 Simulink model of a nitrification bioreactor in an aquaponics system

The gains shown in table 5 were obtained from the Matlab/Simulink model based on works carried out by the author in Appendix I. Figure 3.10 shows the results on modeling carried out using Simulink/Matlab from the study presented in Appendix I. These values are fed in to the digital PID controller implementation presented in the code listing in Appendix IV.

Table 5 Gain values obtained from Simulink model

Proportional gain	**1.2**
Integral gain	0.5
Differential gain	0.2

3.5 Hardware Design

The pH is generally acquired from a pH probe. The probe is an amperometric device that measures a small current generated from the probe. The pH measures the concentration of H+ and OH- ions within a solution and converts them to an electric signal using an ion sensitive membrane transducer. In this study off-the-shall hardware would be used in the controller. Table 6 compares different pH measuring systems.

Table 6 Rise times of DRL & PID controller in static system

	pH Sensor			**Dosing Pump**			
	Interface	**Cost**	**Accuracy**	**Range**	**Interface**	**Cost**	**Accuracy**
Atlas Scientific	UART/ I2C	Average	+/– 0.002	.001 – 14.000	UART/ I2C	Average	+/- 1%
Endress Hauser	4-20mA	Expensive	+/-0.01	.01 – 14	Modbus TCP/RTU	Expensive	+/-0.5%
Hanna	Manual	Low	+/-0.05	.1– 14	-	Average	+/-5%
Horiba	Serial	Expensive	+/– 0.001	.001 – 14.000	Modbus TCP/RTU	Expensive	+/-0.5%

Based on the above criteria, the Atlas Scientific pH probe and dosing pump was selected in our design. The main actuator for the pH controller is the dosing pump. The dosing pump is based on the peristaltic pump by Atlas Scientific. Atlas scientific provides laboratory grade OEM devices that is fully supported with an application programming interface to implement our own hardware & software. This drastically reduce hardware development time.

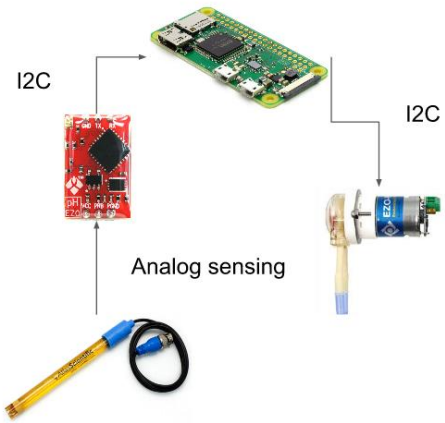

Figure 3.11 Hardware Interfacing

The chosen hardware for implementing the DRL controller is the raspberry pi single board computer. Raspberry Pi is low cost computational/educational platform that is capable of running the linux OS. This makes the raspberry Pi a powerful computer ideal for prototyping robotic controllers, IoT gateways and even nodes of a computer cluster. Therefore, the Raspberry Pi is ideal for prototyping the proposed controller in embedded computing infrastructure. The raspberry pi model that is used in prototyping the proposed deep reinforcement learning based controller is the raspberry pi zero w. The raspberry pi zero has inbuilt wifi and bluetooth connectivity, enabling easy hardware interface to the Internet. This is an important feature that helps in implementing the deep RL based controller design.

The Atlas pH probe takes pH readings from water that flows through the nitrification bioreactor of the aquaponics system. The probe connects to the Atlas pH sensor as shown in Figure 3.11**.** The sensor converts the readings to a digital signal and sends it to the raspberry Pi using an I2C interface. The control signals generated from the DRL controller is sent to the dosing pump also using the I2C interface.

CHAPTER 4 EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Introduction

In this chapter details on experimental results and their analysis are presented for a deterministic/ static system and a dynamic system setup. The static system is a normal titration experiment and the dynamic system is the aquaponics system.

4.2 Evaluating controller performance under a static deterministic system

The experimental setup for the static deterministic case is shown in Figure.4.1. The pH controller controls the dosing based on the input pH. The DRL controller designed in this study is programmed in to the raspberry, which acts as the pH controller. In this setup, the performance under acid-base titration is recorded and analyzed.

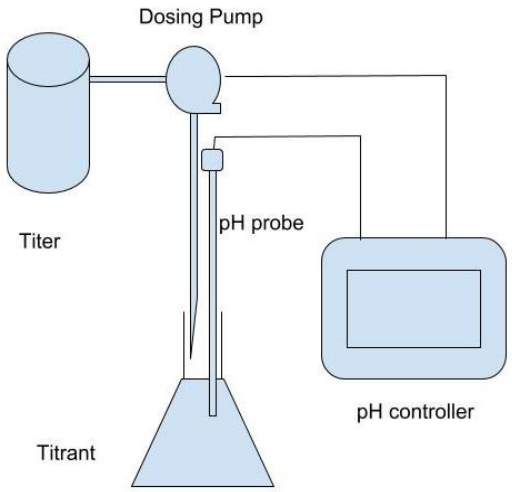

Figure 4.1 Setup to study the performance in a static system

4.2.1 Hardware Setup

Figure 4.2 shows the assembled setup with the pH probe, dosing pump and the raspberry pi.

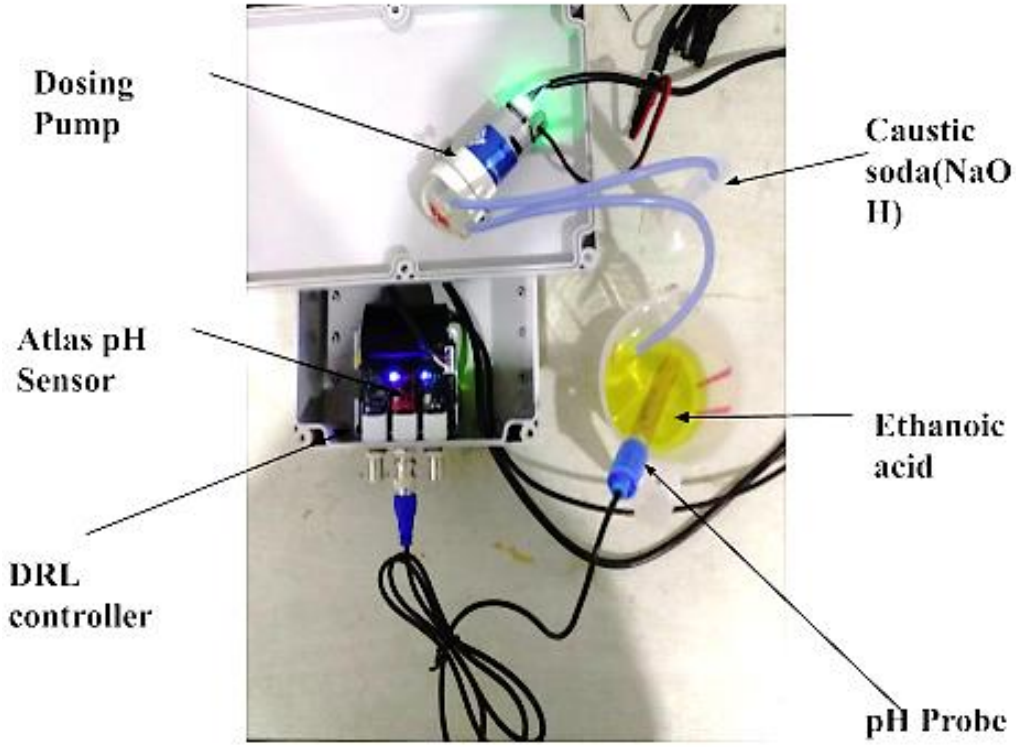

Figure 4.2 Setup to study the performance in a static

4.2.2 Experimental procedure

In the static process, a base would be dosed into an acid and the output characteristic observed. In this case, caustic soda (NaOH) of normality 0.1 mol/L will be used to neutralize concentrated vinegar (ethanoic acid).

4.2.3 Results

The transient response of the controller for a set pH of 7.2 is given in given in Figure 4.3

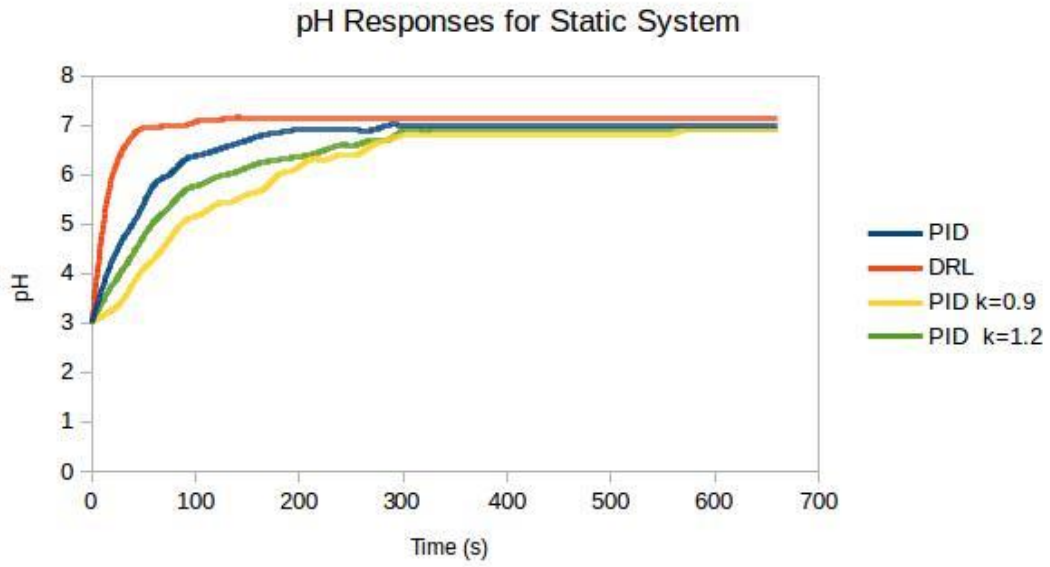

Figure 4.3 Transient responses of the controllers in the static

The proposed controller is promising as it is capable of achieving the set point faster than the digital PID controller. The controller is capable of predicting the systems next state using the learnt model, whereas in PID controllers no such prediction is made. Therefore, the ability to predict makes the DRL based controller respond quickly to the error, process noises and external disturbances. This results in significantly smaller rise times for a step input. Based on Figure 4.3, the rise time of the deep reinforcement learning based controller is around 50 min whilst the rise time of the of the PID controller is about 150 min in static system. The PID parameters were based on table 5 given in section 3.4. Two more titration curves were obtained for two different variations of the proportional gains and as shown in Figure 4.3. These gains are deviations of proportional gains obtained from section 3.4. Further improvement of the rise time was limited due to limitation in actuator dosing, only a maximum of +10 mL/min could be dosed by the Atlas Scientific peristaltic dosing pump.

4.2.4 Analysis

In order to test whether there is a significant difference between the DRL based controller and the PID, the rise time and settling times of both DRL and PID controller transient responses were collected. The PID controllers were tested out with a range of different gain values, introducing extra variance into the system. Sampled rise time from these step responses is shown in table 7.

Table 7: Rise times of DRL & PID controller in static system

	Rise Times (s)	
	DRL	**PID**
	51	96
	61	109
	42	155
	50	93
	52	151
	56	95
	58	86
	64	130
	54	119
	49	170
	55	138
Mean	53.8	122
Variance	36.8	827.4

The data in table 8, shows that there is very little skew in the observed readings. Therefore, we can say that the probability distributions of the rise time of both the controllers are symmetric and can be modeled by the normal distribution. Therefore, we can say that the rise time of the DRL control can be modeled with a Normal~ (53.8, 36.7) distribution and the rise time of the PID controller can be modeled with a Normal~ (122, 827.4). The large variance in the PID controller is because the PID gains can be set to different values to obtain different rise times. Therefore, we can proceed to test for Analysis of variance.

Table 8: Rise times of DRL & PID controller in static system

	DRL Control	PID Control
Mean	53.82	122
Standard error	1.83	8.67
Median	54	119
First Quartile	50.5	95.5
Third Quartile	57	144.5
Variance	36.76	827.4
Standard Deviation	6.06	28.76
Kurtosis	0.45	-1.31
Skewness	-0.18	0.33
Range	22	84
Minimum	42	86
Maximum	64	170
Sum	592	1342
Count	11	11

The data was subjected to a 1-way ANOVA test to determine if there is a significant difference between the performances of the two controllers. The following hypothesis was used in the ANOVA test with alpha = 0.05

H0. The means of observations for RL controller is same as PID controller.

H2. The means of observation for RL controller is different to PID controller.

Table 9: Results of ANOVA test on the static case results

ANOVA-Single Factor						
Alpha	0.05					
Groups	Count	Sum	Mean	Variance		
DRL Controller	11	593	54	34		
PID Controller	11	1342	122	832		
Source of Variation	SS	df	MS	F	P-value	F critical
Between Groups	25525	1	25525	58.95	2.19	4.35
Within Groups	8660	20	433			
Total	34185	21				

Based on table 9, the null hypothesis is rejected, where the performance of the two controllers are not the same. So we can firmly propose that the Deep RL based controller is different to the PID controller with higher confidence. However, ANOVA does not say which is better than the other. The mean of Deep RL is smaller than the PID controller and the two controllers are different, meaning that the Deep RL controller is faster.

The mean steady state values for the two controllers are given in table 10. It can be seen that both the controllers have similar final values.

Table 10: Comparison of steady state value of the two controllers in the static case

	DRL Control	**PID Control**
Mean Steady State Value	7.202136313	7.67035

4.3 Evaluating controller performance under a dynamic stochastic system

The aquaponic system represents the dynamic system that is used to evaluate the performance controller as shown in Figure 4.4. Effluent waste from the fish tank is pumped to a bioreactor before it goes to the grow bed and then returned to the fish tank. The pH controlling process happens within the bioreactor under the influence of the proposed DRL based controller.

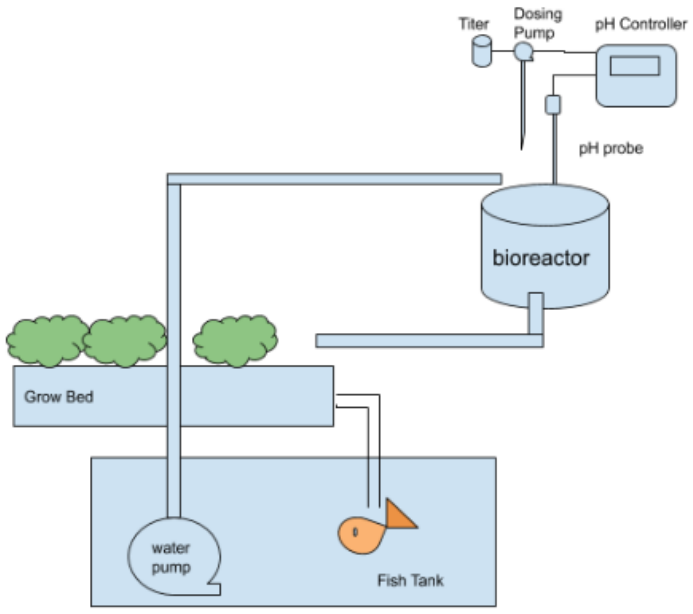

Figure 4.4 Setup used to study the performance in a dynamic

4.3.1 Hardware Setup

The aquaponics setup with the bioreactor and the DRL controller is shown in Figure.4.5. The experimental procedure for the above setup is given in the next section

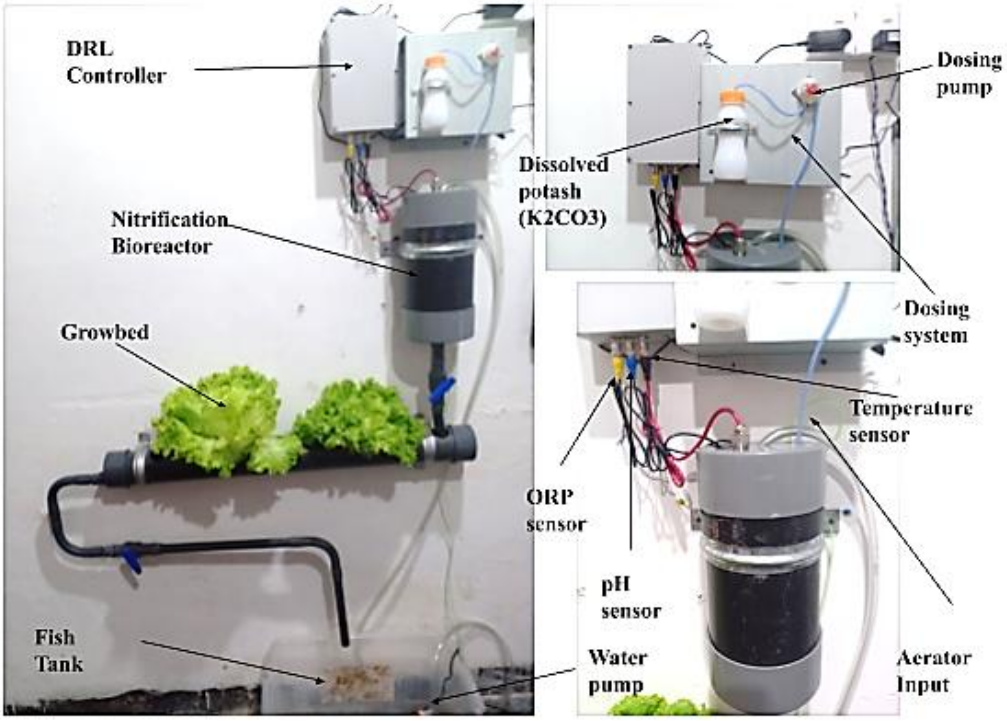

Figure 4.5 The aquaponics system used to determine the response of the DRL controller dynamic stochastic conditions

.

4.3.2 Experimental procedure

In the dynamic case, which is the harder control problem, the current state of the system depends on the previous inputs of the system. In order to evaluate the performance of the controller in the dynamic case, the following setup would be used. A dynamic system shown below is a simple aquaponics system. Aquaponics is a closed loop system for growing plants, where water from a fish tank is used as nutrients to plants through the recirculation of water using a water pump. Nitrification occurs within the system in the grow bed using microorganisms. Nitrification causes the pH to decrease and this pH in turn causes nitrification process to decrease. This pH decrease is also indicative of the amount of microorganism available to actually perform the conversion from ammonia to nitrates. Clearly the current pH value has an impact on the performance of the system in the future, therefore in this setup, we want the controller to maintain a pH of 7.2 continuously and taking the necessary actions to counter system changes due to process noise and disturbances. The system ran for 45 days and the pH was observed over the entire duration. The performance of the two controllers was evaluated based on the steady state pH value.

4.3.3 Results

The controller was tested for its steady state behavior under a dynamic system. Due to the stochastic nature of the control process is better handled by the proposed Deep RL based controller. The stochastic nature of the process is effectively learnt by the neural networks, especially due to the recurrent neural network. The recurrent network in particular is capable of accurately predicting the behavior of the dynamic system as it learns the state space through gathered experience. With limited experience or limited exploration of state space, the controller slightly under performs but with experience, the system shows superior results. The pH values of the aquaponic system monitored over a 45-day period is shown in Figure 4.6.

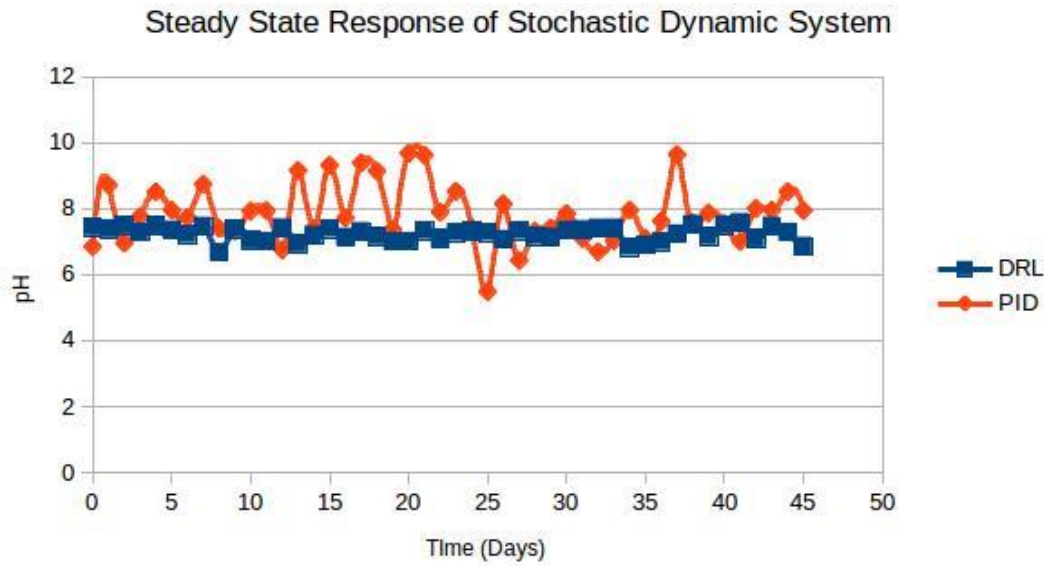

Figure 4.6 Steady state response of the aquaponics system. This setup is a stochastic system and the pH should be maintained at a set point of 7.2 for extended durations.

4.3.4 Analysis

In the dynamic case the mean steady state value was evaluated and compared for a set point of pH = 7.2

Table 11: Comparison of steady state value of the two controllers in the dynamic case

	DRL	PID
Mean Steady State Value	7.202136313	7.7180351952

Based on table 11 we can say that the DRL based controller is much better at maintaining the pH for long periods of time. In this study the controllers were observed for 45 days in an aquaponics system. The data was logged by the raspberry pi based controller hourly. All readings taken during the day were averaged. Average readings for the 45 days were used to generate the output characteristics of the controller. Table 11 gives the mean value for all 45 days and is summarized for both controllers.

CHAPTER 5 CONCLUSIONS

5.1 Conclusion on Objectives

This research set out to investigate the application of state of the art deep reinforcement algorithms (DRL) in the optimal control of real world nonlinear industrial applications, with focus on optimal pH control of a nitrification bioreactor in an aquaponics environment. The requirement for this research emerged from the widespread adoption of Artificial Intelligence in industrial applications in order to meet Industry 4.0 standards. **In this research study, a Deep Reinforcement Learning based controller was implemented for the optimal control of pH in a nitrification bioreactor of an aquaponics system. The controller used the Asynchronous Advantage Actor Critic algorithm as its main driving DRL algorithm. The performance of this algorithm was compared with a standard digital PID control algorithm. The studied DRL algorithm proved to be superior to the PID control scheme when operating on a deterministic system and on a dynamic system. Therefore, we can conclude that DRL algorithms can be used to effectively control highly non-linear process control systems.**

One of the major aspects of this study was that in classical pH control, sampling time requires to be long for the system to come to equilibrium. In the case of the DRL based controller, the system dynamics are slowly learnt with experience and is capable of taking drastic actions without causing instabilities within the process control system. The PID controller is concerned only with the current observed reading and the current error, it has no measure of the system states and its transition dynamics, it just feeds back weighted sums of the observed states. This highlights a limitation of using PID controllers in controlling non-linear systems. Therefore, we can conclude that deep RL based controllers can produce fine grained control with respect to PID controllers.

5.2 Conclusion on Research Questions

In this study, model-free reinforcement learning algorithms were identified and compared with respect to one another. Important aspects of these algorithms are its policy and its method of updating its parameters. The controller designed is a deep reinforcement learning algorithm that is capable of learning the system without any prior knowledge about the system or an explicit mathematical model. Therefore we can provide a positive answer to the first research question. Furthermore, in this study minimalistic use of sensors were enforced in controller design to simulate the partial observation scenario. We incorporated beliefs into the system to compensate for the limited observations in data through the use of recurrent neural network architectures in the controller design. Information & beliefs was extracted from a sequence of previous time varying data to estimate the current state without fully observing the system. The recurrent neural network provides a powerful function approximate that can estimate past history of the system and thereby aid in deriving the appropriate beliefs in obtaining an optimized controller under partial observation.

5.3 Further works

The work carried out in this thesis can be extended to discrete non-linear systems in the industry. The following text elaborates on the application of DRL algorithms in some important industrial processes.

5.3.1 Internet of Things use case

The deep reinforcement learning controller resides within the learning servers of an IoT cluster, where the algorithm resides and performs the DRL learning steps. This flow is shown in the learning cluster of Figure 5.1. The optimized controllers are then fed back into the endpoints. It is important to note that the controller is actually residing in the endpoint and only the learning and parameter optimization process performed using the learning cluster. This particular implementation was chosen to comply the implementation of the system using Industry 4.0 standards and complete justification of it is beyond the scope of this research study.

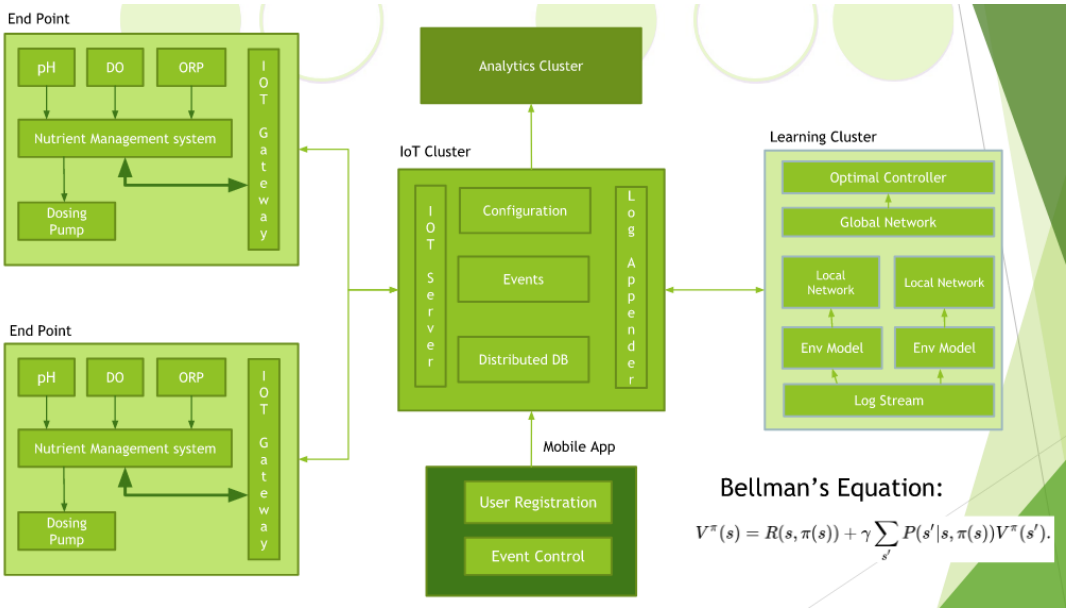

Figure 5.1 Implementation of the DRL controller in IoT/Industry 4.0 based applications

5.3.2 DRL controllers in SCADA systems

An example of implementing the Deep Reinforcement Learning based controllers in an Industrial SCADA system is shown in Figure 5.2.

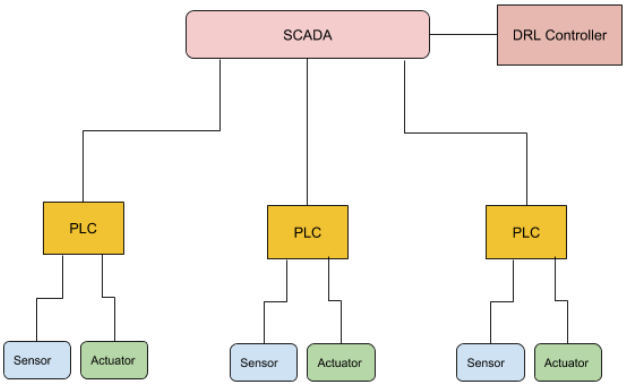

Figure 5.2 Implementation of the DRL controller in a SCADA scenario

The Deep RL controller consumes the data from sensors and actuators through the networking provided by the SCADA system. The DRL controller is hosted in a server from which the SCADA data is accessed. The DRL server then creates the policy and critic networks required by the DRL algorithm. These policies then can be enforced via the SCADA network. The following use case shows a potential application of a SCADA based DRL controller architecture.

Waste Management Optimization - The SCADA based DRL can be effectively used in applications of energy optimization and optimal pollution control in waste management incinerators. The DRL can be effectively combined with the SCADA system to produce optimal control sequences to minimize energy costs and to improve gas emissions from the incinerator. The application of state of the art deep reinforcement algorithms in a variety of non-linear industrial applications was shown in this study. These controllers, powered by artificial intelligence algorithms, paves the way for the next industrial revolution, Industry 4.0. Therefore, further studies in this domain is encouraged in order to for AI to really impact the modern world.

REFERENCES

[1] J. X. Chen, "The Evolution of Computing: AlphaGo," *Computing in Science & Engineering*, vol. 18, no. 4, p. 4–7, 2016.

[2] V. Mnih et al, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529–533, 2015.

[3] S. P. K. Spielberg, R. B. Gopaluni and P. D. Loewen, "Deep reinforcement learning approaches for process control," in *Proceedings of the 6th International Symposium on Advanced control of Industrial Processes (AdCONIP)*, Taipei, 2017.

[4] P. M. Jacob, "A Comparative Analysis on Smart Farming Techniques using Internet of Things (IoT)," *HELIX*, vol. 8, no. 2, p. 3294–3302, 2018.

[5] N. A. Savidov, E. Hutchings and J. E. Rakocy, "Fish and Plant Production in a Recirculating Aquaponic System: A New Approach to Sustainable Agriculture in Canada," *Acta Horticulturae*, no. 742, pp. 209-221, 2007.

[6] C. Somerville, M. Cohen, E. Pantanella, A. Stankus and A. Lovatelli, "Small-scale aquaponic," FAO, Rome, 2014.

[7] P. A. Pearce and D. M. Foster, "Optimizing Nitrification on Biological Filters," *Water and Environment Journal*, vol. 13, no. 6, p. 406–412, 1999.

[8] S. Okabe, Y. Aoi, H. Satoh and Y. Suwa , "Nitrification in Wastewater Treatment," in *Nitrification*, Washington, DC, ASM Press, 2011, pp. 405-433.

[9] J. Makinia, Mathematical Modelling and Computer Simulation of Activated Sludge Systems, London: IWA Publishing, 2010.

[10] J. Skadsen, "Effectiveness of High pH in Controlling Nitrification," *American Water Works Association*, vol. 94, no. 7, p. 73–83, 2002.

[11] D. R. James and R. W. Lumry, "Recent Developments in Control of pH and Similar Variables," in *Methods of Biochemical Analysis*, New York, John Wiley & Sons, 2006, p. 137–206.

[12] A. O'Dwyer, Handbook of PI and PID Controller Tuning Rules, London: Imperial College Press, 2009.

[13] M. M. Zirkohi , "Optimal Pid Controller Design Using Adaptive Vurpso Algorithm," *Open Engineering*, vol. 5, no. 1, p. 179–185, 2015.

[14] D. P. Atherton and S. Majhi, "Limitations of PID controllers," in *Proceedings of the American Control Conference*, San Diego, 1999.

[15] H. P. H. ANH and N. T. Nam, "A new approach of the online tuning gain scheduling nonlinear PID controller using neural network," 19 April 2011. [Online]. Available: http://www.intechopen.com/books/pid-controlimplementation-and-tuning/a-new-approach-of-the-online-tuning-gain-scheduling-nonlinear-pid-controllerusing-neural-network. [Accessed 18 03 2018].

[16] B. KELKAR and B. POSTLETH WAITE , "FUZZY-MODEL BASED pH CONTROL," in *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, Orlando, 1994.

[17] D. Shaghaghi, H. MonirVaghefi and A. Fatehi, "Generalized predictive control of pH neutralization process based on fuzzy inverse model.," in *13th Iranian Conference on Fuzzy System*, Tehran, 2013.

[18] D. Nauck and R. Kruse, Neuro-Fuzzy Methods in Fuzzy Rule Generation, Boston: Springer, 1999.

[19] D. Kalise, K. Kunisch and Z. Rao, Hamilton-Jacobi-Bellman Equations: Numerical Methods and Applications in Optimal Control, Berlin: De Gruyter, 2018.

[20] P. J. Werbos, "Using ADP to Understand and Replicate Brain Intelligence: the Next Level Design," in *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, Honolulu, 2007.

[21] J. L. Speyer and W. H. Chung, Stochastic Processes, Estimation, and Control, Philadelphia: Society for Industrial and Applied Mathematics, 2008.

[22] R. P. Srivastava and T. J. Mock, "Introduction to Belief Functions," in *Belief Functions in Business Decisions*, Heidelberg, Physica, 2002, p. 1–16.

[23] J. S. Kennedy, "Introduction to Dynamic Programming," in *Dynamic Programming*, Dordrecht, Springer, 1986, p. 27–49.

[24] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, London: Bradford Books, 2018.

[25] D. Liu, Q. Wei and P. Yan, "Generalized Policy Iteration Adaptive Dynamic Programming for Discrete-Time Nonlinear Systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 12, p. 1577–1591, 2015.

[26] Q. Wei and D. Liu, "Optimal learning control for discrete-time nonlinear systems using generalized policy iteration based adaptive dynamic

programming," in *Proceeding of the 11th World Congress on Intelligent Control and Automation*, Shenyang, 2014.

[27] A. Jeerige, D. Bein and A. Verma, "Comparison of Deep Reinforcement Learning Approaches for Intelligent Game Playing," in *IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, 2019.

[28] S. Tigani, M. Ouzzif and A. Hasbi, "Monte Carlo simulation based algorithm design for automatic learning machine performance analysis," in *2014 International Conference on Next Generation Networks and Services (NGNS)*, Casablanca, 2014.

[29] O. Sonmez and A. T. Cemgil, "Sequential Monte Carlo samplers for model-based reinforcement learning," in *2013 21st Signal Processing and Communications Applications Conference (SIU)*, Haspolat, 2013.

[30] M. Reidmiller, J. Peters and S. Schaal, "Evaluation of Policy Gradient Methods and Variants on the Cart-Pole Benchmark," Honolulu, 2007.

[31] H. S. Jakab and L. Csato, "Reinforcement learning with guided policy search using Gaussian processes," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, Brisbane, 2012.

[32] H. Lin, Q. Wei and D. Liu, "Online identifier-actor-critic algorithm for optimal control of nonlinear systems," *Optimal Control Applications and Methods,* vol. 38, no. 3, p. 317–335, 2017.

[33] S. Parisi, V. Tangkaratt, J. Peters and M. E. Khan, "TD-regularized actor-critic methods," 19 December 2018. [Online]. Available: https://arxiv.org/abs/1812.08288. [Accessed 8 January 2019].

[34] U. Doraszelski and K. L. Judd, "Avoiding the curse of dimensionality in dynamic stochastic games," *Quantitative Economics,* vol. 3, no. 1, p. 53–93, 2012.

[35] K. Hara and K. Nakayamma, "Comparison of activation functions in multilayer neural network for pattern classification," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, Orlando, 1994.