UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS Honours Project Final Paper 2024

Title: **A Comparative Study of Control Methods for Non-Linear Binary Control Systems Using Data-Driven Techniques: Q-Learning**

Author: **Ariel J. Levy**

Project Abbreviation: **DDC**

Supervisor(s): **Krupa Prag**

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	20
Theoretical Analysis	*0*	*25*	10
Experiment Design and Execution	*0*	*20*	10
System Development and Implementation	*0*	*20*	-
Results, Findings and Conclusions	*10*	*20*	10
Aim Formulation and Background Work	*10*	*15*	10
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	-
Total marks		**80**	80

Comparing Data-Driven Control Methods for Non-Linear Binary Systems: Q-Learning

Ariel J. Levy
University of Cape Town
Cape Town, South Africa
LVYARI002@myuct.ac.za

Abstract

Data-driven control methods are steadily rising in prominence as solutions for tackling complex control problems in the data-rich environment of the Information Age. Machine learning-based approaches, such as reinforcement learning (RL) and neuroevolution (NE), are of particular interest due to their generalisability and adaptability. These methods aim to discover optimal control policies by relying on data, diverging from the manual tuning strategies employed by traditional controller architectures like Proportional-Integral-Derivative (PID). This research evaluates the performance of Q-Learning, a popular RL algorithm, against traditional PID control and another data-driven technique: Neuroevolution of Augmenting Topologies (NEAT). Applied to non-linear binary control systems, specifically the inverted pendulum (IP) and DC-DC buck converters (BCs), the results reveal that the performance of data-driven methods largely depends on the complexity of a system's state space; NEAT excels in complex systems, while Q-Learning performs better with simpler, less externally influenced state spaces. In contrast, though consistently adequate, PID control lags behind the data-driven approaches due to its lack of system knowledge and rigidity. In addition, experiments with Q-Learning reward functions showed that sharply decreasing unimodal rewards accelerate stabilisation, contributing to greater interpretability of model behaviour, a key aspect of explainable AI (XAI). These findings provide valuable insights for developing more effective and adaptable control systems, particularly for non-linear binary systems, in real-world applications.

1 INTRODUCTION

Control theory is a field that lies at the intersection of control engineering and applied mathematics; it deals with the development of algorithms that, given the current state of a system, serve to guide it towards a desired state. The field, while ever-present in today's technology-dependent society, has roots that stretch back centuries.

1.1 Historical Development

The First Industrial Revolution marked a profound shift in humanity's relationship with technology. Traditional artisanal craftsmanship yielded to the efficiency of mechanised production and, consequently, once rural, agrarian communities evolved into spawling urban societies [21]. The need to control and optimise the performance of these new, widely adopted machines – thereby maximising the throughput of factories – led to the conception of control theory, initially proposed by James C. Maxwell [1867].

Such a field was well-suited to aiding the development of novel technologies throughout the 19^{th} and into the 20^{th} century; with use found in burgeoning industries such as boiler control for steam generation, electric motor speed control, and temperature, pressure, and flow control in the process industries [7]. A large milestone in the progression of control theory came in 1911 when Elmer Sperry created the PID controller to automate ship steering for the US Navy [7]. PID controllers have since gained widespread adoption in industry, due to their versatility, robustness, and model-free nature.

The field continued to undergo major developments throughout the World Wars, particularly in the theoretical aspects of control theory, which had long been overshadowed by the focus on the theory's engineering applications. Analyses of control systems and their designs were performed by many independent groups, which ultimately resulted in the formalisation of modern control theory in the early 1950s [21].

1.2 Modern Advances

Since the 1960s, modern control theory – also referred to as model-based control (MBC) – has matured into a field with many successful applications, especially in the aerospace industry [13]. MBC theory works by using a model of the system being controlled (referred to as the plant) and a set of assumptions – that should, in theory, represent the true system – to predict and adapt the future behaviour of the controlled system. This methodology is flawed. If the model is inaccurate or the assumptions made about the nature of the system do not hold, then no meaningful conclusions can be drawn and the physical controller will not function well [13]. In practice, this disconnect between the theoretical and practical aspects of MBC presents an obstacle to the adoption of this controller architecture in certain industries where systems are: novel, highly complex, or uncertain.

MBC's reliance on the plant model being accurate, to function as intended, can be reduced by implementing an even more powerful control framework: data-driven control (DDC). DDC works by leveraging the abundance of information in the Digital Age to fine-tune controllers using learning techniques – irrespective of the framework's dependence on the system's mathematical model. This means that historical data can be used to account for a lacklustre system model, if one even exists, with the controller's performance improving autonomously as more input/output (I/O) data is collected [13, 21]; this means that DDC can support both model-based and model-free control methods – depending on the implementation chosen.

1.3 Data-Driven Control Frameworks

1.3.1 Classical Control. Classical control theory – which governs controllers such as PID – deals with managing a system so that its output follows a desired signal. It is believed that applying a data-driven approach to the concepts of classical control theory

could improve the performance of traditional controllers, in particular, this enables such controllers to adapt to varying system conditions (such as the introduction of noise), automatically tune parameters, and better handle non-linear systems. However, this control method has, thus far, failed to gain a foothold in industry due to the dominance of model-based frameworks [6].

1.3.2 Machine Learning and Control. As mentioned earlier, the concept of DDC is based on the principle of self-learning. Combined with the substantial volume of I/O data generated, this naturally directs one's attention to the field of machine learning (ML).

More specifically, the subfield of ML that aligns best with these factors is reinforcement learning; RL is a technique that allows a system to learn through trial and error – as humans do – to achieve the optimal output for a specific task. This enables a controller to teach itself how to control a system (optimally) in an environment, with no knowledge of the underlying dynamics of the system (i.e. it is model-free) [8].

Another distinct, but related, approach in the field of ML is NE. NE makes use of genetic algorithms (GAs) to evolve artificial neural networks (ANNs) [23, 28]. In the context of controllers, this allows an initial ANN – which governs the controller's actions – to evolve and adapt over generations subject to its environment.

This learning method can be improved by applying a technique known as NEAT. NEAT makes slight alterations to NE, which is said to result in increased efficiency. It should be noted that NEAT – and to a lesser extent NE – are not thoroughly researched learning methods.

1.4 Binary Control Systems

Binary control systems are systems that can controlled exclusively with two actions; for example "on/off" or "0/1". Despite their discrete nature, these systems can handle continuous input spaces and achieve precise control, by leveraging techniques such as pulse width modulation (PWM) – which emulates continuous control by adjusting the duration of the binary states to regulate a process [12].

This research aims to evaluate the performance of control methods in the context of two binary control systems: the IP, chosen due to its frequent use as a benchmark problem in the field, and DC-DC BCs, which are widely used in industrial settings and consumer appliances.

1.5 Aims

This research aims to provide a solution to the given problem statement through the careful assessment of clearly defined and measurable research questions.

1.5.1 Problem Statement. Existing literature on non-linear binary control systems lacks a thorough comparison of the effectiveness of established data-driven control methods like Q-Learning and NEAT.

1.5.2 Research Question. How does the performance of Q-Learning, a data-driven method, compare to traditional PID control and other data-driven methods such as NEAT, in non-linear binary systems, when evaluated in terms of stabilisation time, training time, robustness, and steady-state error?

2 BACKGROUND

To facilitate a meaningful comparison, control methods are evaluated within the context of specific dynamical systems, namely the IP and DC-DC BC systems.

2.1 Inverted Pendulum

The IP is considered a benchmark problem in the field of control theory as a result of its: simplicity, instability, and non-linearity.

2.1.1 PID. Two studies explored PID's application to the IP and found that it was able to successfully stabilise the system within a reasonable time frame ($\leq$ 6 s); which is unsurprising due to PID's longstanding position as one of the most reliable traditional controllers [30, 31]. The controller failed to effectively deal with noise, which is to be expected, as noise is a non-linear process.

2.1.2 Q-Learning. Q-Learning was successfully employed to stabilise a real-world IP. However, a critical failure occurred when the system exceeded the flange's position – a constraint that was likely not accounted for in the model's virtual training environment [25].

Another group's results were less promising, as the system was unable to stabilise for any meaningful amount of time, which was ultimately attributed to learning inefficiencies resulting from the Q-table's matrix representation [15].

2.1.3 NEAT. When NE-based techniques were applied to a real-world IP system, it was found that NEAT did not perform as well as other NE-based models. In ideal conditions, the NE models were able to successfully stabilise for long periods, however, when noise was introduced the NE models struggled to remain balanced [22].

2.2 DC-DC Voltage Converters

DC-DC voltage converters are essential components in industry and consumer electronics, where they facilitate voltage conversion, ensuring reliable performance and energy efficiency.

2.2.1 PID. The application of manually-tuned PID to a (simulated) boost converter was able to step up a source voltage of $90 - 110$ V to the reference voltage of 200 V with an overshoot error of less than 3%. However, this took around 1000 s to stabilise [1].

Another group used the Sine Cosine Algorithm (SCA) to automatically tune the PID's gain coefficients before it was applied to a BC system. This yielded significantly better results than the first paper mentioned. The SCA-tuned PID controller was able to step down the source voltage to the correct reference value (3 V) with five decimal places of precision within the order of 10^{-6} seconds; whereafter steady-state was maintained [18].

2.2.2 Q-Learning. Existing literature does not explore the application of Q-Learning to DC-DC buck converters as a control method. This further highlights the need to investigate the use of this control approach for this specific system.

2.2.3 NEAT. There is no available literature on the use of NEAT in DC-DC voltage conversion. However, other NE-based methods were used to successfully control a BC, which managed to stabilise in less than 130 ms with minimal overshoot [27].

3 METHODOLOGY

This section discusses the theory of the control methods employed and outlines the rationale behind the architectural and implementation decisions made regarding the core components of this research. Consequently, Q-Learning and, to a lesser extent, PID are described in detail, while NEAT is only briefly outlined. In addition, explanations of the systems under consideration and the respective metrics used to evaluate the performance of said control methods are included.

3.1 PID

3.1.1 Theory. PID controllers incorporate three key actions – proportional, integral, and derivative – to drive a system to a desired output. This is accomplished by taking an error signal, calculated by comparing the system's output and the desired output, and adjusting it using each of the actions independently to obtain a controlling signal [6]. The proportional (P) term generates an output proportional to the error; a large error means a strong corrective response. The integral (I) term responds to the accumulation of errors over time; this propels the system to the desired output as time increases. Finally, the derivative (D) term predicts future errors based on the current error's rate of change; a greater change results in a stronger signal-damping effect to prevent overshooting [4, 16].

Mathematically this explanation can be formulated, in terms of the control signal $u(t)$ and the error signal $e(t)$, as follows [4],

$$u(t) = K_p e(t) + K_i \int_0^t e(t')dt' + K_d \frac{de(t)}{dt}, \qquad (3.1.1)$$

where K is the gain coefficient of each of the terms. These gain coefficients are most commonly tuned through trial and error, however, other optimisation techniques do exist [18].

PID control is best suited to linear systems but can be applied successfully to non-linear systems. In such cases, linearisation approximations provide a sufficiently accurate representation of the true system.

3.1.2 Controller Architectures. For the IP system, a basic implementation of PID control was used, which associated the control signal (provided by the PID controller) with the output force to be applied to the cart.

Moreover, implementing PID for DC-DC BCs necessitated the introduction of additional measures to address the fact that the duty cycle, which ultimately controls the converter, is constrained within the range of 0 to 1. This issue is solved through the addition of saturation, which constrains the controller's output between two defined bounds [26]; these limits were set to be 0.1 and 0.9 to mitigate potential undesirable effects, such as excessive actuator wear, thereby ensuring that the controller can be applied to real-world scenarios safely. Anti-windup measures were also implemented to address overshoots resulting from the accumulation of the integral term during periods of controller saturation [4].

3.1.3 Tuning. Both PID controllers were tuned through MATLAB's PID Tuner application to optimise their performance [14]. Table 1 captures the values for the gain coefficients that were determined.

In contrast to the IP, the performance of PID for the BC proved extremely sensitive to the values of the gain coefficients. Consequently, any minor alterations to the values of these coefficients, including the introduction of the derivative term, resulted in sub-optimal performance.

Table 1: PID Gain Coefficient Values

Symbol	Inverted Pendulum	Buck Converter
K_p	44	0.45
K_i	80	43.5
K_d	6	0

3.2 Q-Learning

3.2.1 Theory. Q-Learning is a model-free RL method proposed by Christopher J. Watkins [1989]. It starts with an agent (the controller's logic) in a state (the information that defines the system at that moment in time). The agent then takes an action based on its strategy. Thereafter, it either receives a reward or a punishment. The agent learns from previous actions and optimises its strategy using this reward information. This process, referred to as an episode, iterates until an optimal strategy is found. The value of each state-action pair is stored in a Q-table, which is updated as the agent interacts with the environment and rewards/punishments are distributed [32].

Q-Learning often makes use of an ϵ-greedy strategy to select actions throughout its training process. Given the current state, it chooses the best action, most of the time, but occasionally explores the outcomes of taking random actions [15].

The pseudocode for a general application of the Q-Learning algorithm is given in Algorithm 1 [15, 33].

Algorithm 1 Implementation of Q-Learning

```
Intialise Q(s, a) ∀ s ∈ S and ∀ a ∈ A with zeros
for ep in episodes do
  Select random initial state s ← s_i
  while s ≠ s_goal do
    Using ϵ-greedy strategy select action a
    Define reward function r
    s' ← s_{i+1}
    Q(s, a) ← Q(s, a) + α[r + γ · max Q(s') − Q(s, a)]
    s ← s'
  end while
end for
```

The quality of a Q-Learning model, given a correct problem specification (i.e. a sufficient number of states and an appropriate number of actions), depends on three critical parameters described in Table 2.

3.2.2 State Discretisation. The continuous state space of each system must be converted into a set of discrete states due to the discrete nature of the Q-Table.

Initial training on the pendulum was done on an 18-state Q-table, which effectively and rapidly stabilised small angular displacements (ranging from -0.2 to 0.2 rad). This was later expanded to a Q-table with 144 states, allowing for the stabilisation of larger angular displacements within the range $[-\pi/3, \pi/3]$ rad. This enhanced

Table 2: Q-Learning Parameters

Symbol	Description
r	A system-dependent function that specifies the reward to be received for an action.
γ	The discount factor for future rewards.
α	The learning rate of the algorithm.

state discretisation made use of both the pendulum's angle θ as well as its angular velocity $\dot{\theta}$, following the approach suggested by Barto et al. [1983] [3].

Training on the BC was conducted using a 24-state, voltage-based Q-table, which successfully stabilised output voltages below the chosen source voltage of 48 V. Various expansions to the Q-table – such as incorporating inductor current, reference voltage, and previous actions – were considered, however, the added complexity did not yield improved controller performance.

3.2.3 Action Space. In a similar vein to state discretisation, Q-Learning can only operate with a discrete set of actions. As such, when dealing with binary control systems, it is most natural to start by assigning two actions. For the systems being examined, these actions include: applying a force to the left or right of the pendulum's cart or setting the duty cycle of the voltage converter to 0 or 1.

Subsequently, the action space was expanded for each system, with four actions for the pendulum and five for the BC. The rationale was to assign two distinct sets of actions: an extreme action set for making larger adjustments and a moderate action set for finer corrections; thus allowing for more precise control. Notably, this intended approach was fully embraced by the trained Q-Learning controllers.

3.2.4 Action Selection Strategy. To improve the action selection process, an epsilon decay strategy was implemented instead of a standard epsilon-greedy approach. This approach enhances the exploration of the state space during the initial training phases, whilst allowing the algorithm to better optimise effective stabilisation strategies in later episodes; this is achieved by starting with a large initial epsilon which decays, after a certain episode threshold is exceeded, to a minimum value [9].

In this case, the method known as exponential epsilon decay was incorporated into the action selection mechanism. It can be implemented using Algorithm 2.

Algorithm 2 Exponential Epsilon Decay

if $\epsilon > \epsilon_{min}$ **and** $N_i > N_0$ **then**
  $\epsilon = \lambda^{N_i - N_0}$
end if

The parameters used in the algorithm were defined, across both systems, as specified in Table 3.

3.2.5 Reward Functions and Tradeoffs. In Q-Learning, the reward function should be designed to drive a system towards equilibrium.

Table 3: Epsilon Greedy Parameters

Symbol	Description	Parameter Values	
		Inverted Pendulum	**Buck Converter**
ϵ	Exploration Rate.	1.0	1.0
ϵ_{min}	Min. Exploration Rate.	0.05	0.05
λ	Decay Factor.	0.995	0.995
N_i	Current Episode.	≤ 1500	≤ 1250
N_0	Episode Threshold.	150	125

Inverted Pendulum. For the IP, the reward function was adapted from OpenAI [2022], which is known to stabilise an IP for any angular deviation of magnitude 0.2 radians or less [20]. This reward at time t is provided in Eqn. (3.2.1) as

$$R_t = -(\theta_t^2 + 0.1\dot{\theta}_t^2 + 0.001\tau^2). \tag{3.2.1}$$

This reward function delivered impressive performance, effectively maintaining stability within $\pm\pi/3$ radians – beyond its expected limits. However, it resulted in stabilisation that was slower than expected, especially for larger angles. To address this, an additional "stabilisation bonus" was added to the reward function to promote faster stabilisation; it does this by increasing the intended reward when the pendulum is: close to equilibrium and moving slowly. The impact of this bonus on the Q-Learning controller's performance is demonstrated in Figure 1.

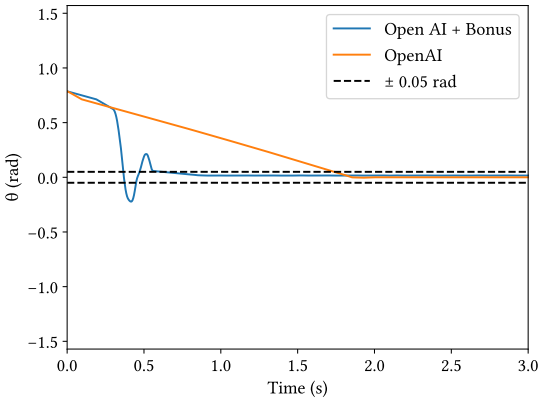

Figure 1: Stabilisation of inverted pendulum Q-Learning reward functions from an initial angle of $\pi/4$ rad.

Figure 1 illustrates the effectiveness of the "OpenAI + Bonus" reward function in stabilising the system more quickly. It stabilises in 0.85 s, compared to 1.81 s for the solely "OpenAI" reward function. This trend is consistent across all angles in the range $[-\pi/3, \pi/3]$, but becomes more pronounced as the initial angular displacement increases.

While the "OpenAI + Bonus" reward function stabilises faster, the "OpenAI" reward function provides a much smoother stabilisation response, gradually approaching equilibrium without abrupt changes in angle. This trade-off between stabilisation time and

smoothness is crucial in specific control scenarios, such as aircraft flight control; in such cases, sudden changes in parameters – though they might lead to quicker stabilisation – can be undesirable and uncomfortable for users. In the case of the IP, abrupt changes in angle do not present any inherent downsides, making the faster stabilisation achieved by the "OpenAI + Bonus" reward function preferable.

Furthermore, this modification to the reward function supports XAI by enhancing the transparency of the algorithm's behaviour through a clearly defined reward bonus that accelerates the system's stabilisation.

Buck Converter. For the BC, two distinct reward functions will be compared to advance the pursuit of XAI in this research.

The first reward function r_t, defined in Eqn. (3.2.2), is expressed in terms of the output voltage V_t and reference voltage V_r at time t [29].

$$r_t = \frac{1}{1 + (V_t - V_r)^2}. \tag{3.2.2}$$

The second proposed reward function ρ_t is novel and is provided in Eqn. (3.2.3).

$$\rho_t = \begin{cases} \left(\left|\frac{V_r}{4}\right| - |V_t - V_r|\right)^2 & \text{if } \frac{3}{4}V_r \le V_t \le \frac{5}{4}V_r \\ 0 & \text{otheriwse} \end{cases}. \tag{3.2.3}$$

The result of training a BC controller using these reward functions is shown in Figure 2.

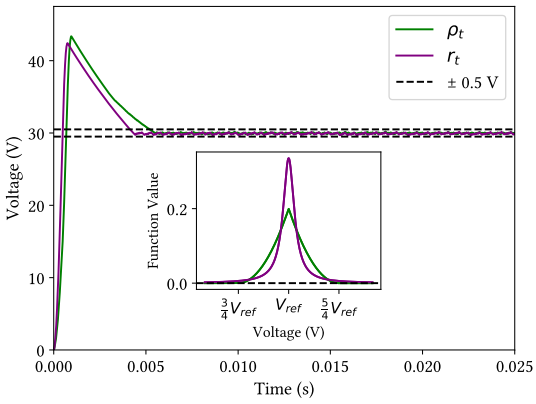

Figure 2: The main plot shows the stabilisation of a buck converter via Q-Learning using two different reward functions. The inset plot displays the area-normalised graphs of said reward functions.

Figure 2 demonstrates that both reward functions enable Q-Learning to control the BC. Reward function r_t stabilises more quickly, and with less overshoot (42 V compared to 43 V). In contrast, ρ_t exhibits more gradual stabilisation. Both reward functions produce minimal steady-state error.

The inset plot provides insight into this behaviour. The steeper gradient of r_t imposes harsher penalties for deviations, quickly driving the system toward equilibrium. In comparison, the gentler slope of ρ_t does not sufficiently penalise oscillations, causing the Q-Learning algorithm to stabilise the system more gradually. This suggests that a unimodal reward function, like r_t, which decreases sharply in both directions from the reference point, is best practice for achieving rapid, low-error stabilisation.

By analysing how ρ_t and r_t affect Q-Learning's decision-making, one reveals how the algorithm adjusts its actions to meet its objectives. Comparing the responses to these reward structures, particularly in terms of stabilisation time, showcases how XAI can be leveraged to enhance algorithm performance – in this case – through more effective reward function design.

Given these considerations, r_t will be adopted as the reward function for the BC system. It not only facilitates faster stabilisation but also results in less overshoot.

3.2.6 Hyperparameter Tuning. After extensive manual tuning, the values for the learning rate α and discount factor γ presented in Table 4 were identified as providing the most reliable results.

Table 4: Q-Learning Hyperparameter Values

Symbol	Inverted Pendulum	Buck Converter
γ	0.99	0.995
α	0.1	0.01

3.3 NEAT

NEAT is a form of neuroevolution created by Stanley and Miikkulainen [2002], which is a type of GA that simulates the evolutionary process of natural selection observed in biological populations. It evolves neural networks by dynamically adjusting and optimising both their weights and topologies over time [23, 28].

This process enables ANNs to mutate and crossover with other ANNs across generations, resulting in offspring with desirable traits.

The quality of each ANN is assessed using a fitness function, which is specific to the system. This function ultimately determines which ANNs are able to reproduce, undergo mutation, and advance to the next generation [28].

3.4 Applications

To facilitate a comparison of the control methods under consideration, they are evaluated in the context of their application to the control of the IP and DC-DC BC systems.

3.4.1 Inverted Pendulum.

System Overview. The IP is a pendulum system in which the centre of mass lies directly above the pivot point; the system is intrinsically unstable and will fall over without assistance due to a gravitational torque. The pendulum's motion is confined to angular movement along the arc of a circle [31].

This system is usually placed on top of a cart – see Figure 3 – to allow control through the application of a force to the cart (either to the left or to the right).

Experimental Model. This research utilised an IP on a cart system – commonly known as the cartpole. The system was simulated using

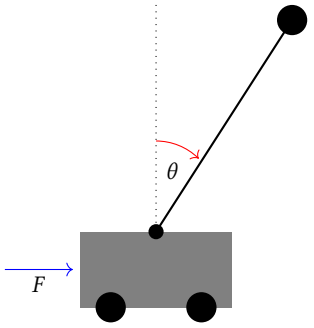

Figure 3: Inverted Pendulum on a Cart

MATLAB's Simulink toolbox [14], with fixed-step iteration applied to the differential equations of motion that govern the system [5].

The system was controlled by applying a corrective force, generated by a controller, to the cart. This force's magnitude and direction were determined, by said controller, using two key state variables: angle and angular velocity.

A stopping condition was also imposed on the simulation when the angle exceeded a magnitude of $\pi/2$ radians. This was done because the primary focus was on controlling angular displacements of $\pi/3$ radians (or less), furthermore, this condition significantly improved training efficiency and time.

Parameters. The parameters used in the simulation of the IP are provided in Table 5.

Table 5: Inverted Pendulum Parameters

Symbol	Description	Value
m	Pendulum Mass.	0.2 kg
M	Cart Mass.	0.5 kg
l	Length of the String.	0.15 m
g	Acceleration due to Gravity.	9.8 m/s^2
dt	Fixed Step Size.	0.001 s
t_{max}	Stop Time.	4.000 s

3.4.2 DC-DC Buck Converters.

System Overview. The DC-DC BC, shown in Figure 4, is an electronic circuit that converts direct current from one voltage level to another. The names and purposes of each component in the circuit are given in Table 6. The DC-DC BC has a single state, the buck state, which lowers the input voltage [1, 18]. The relationship between the input and output voltages under non-ideal, transient conditions is non-linear, making employing an effective control method for precise voltage conversion essential.

The average output voltage V_{out}, under ideal steady-state conditions, is given by Eqn. (3.4.1), where D is the duty cycle, defined as the ratio of the switch's on-time to the total switching period [24]. Eqn. (3.4.1) implies that V_{out} is always less than (or equal to) V_{in}, since $0 \leq D \leq 1$. The duty cycle will be calculated by a control method and subsequently used to regulate the BC through the *MOSFET* switch.

$$V_{out} = D \cdot V_{in}. \tag{3.4.1}$$

The topology used, as shown in Figure 4, ensures that the output voltage polarity matches that of the input, meaning V_{out} remains positive at all times [24]. In addition, V_{out} starts at zero volts due to the capacitor being uncharged and the lack of current flowing through the inductor.

Figure 4: DC-DC BC Circuit

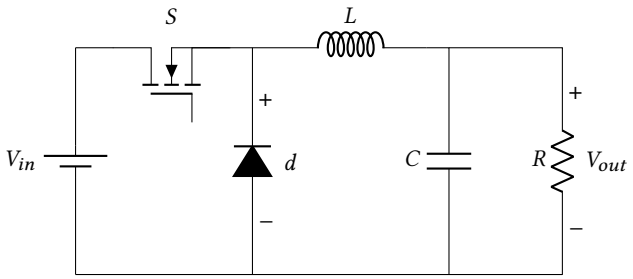

Table 6: DC-DC Buck Circuit Components

Symbol	Name	Purpose
S	*MOSFET* Switch	Dictates the state of the circuit.
L	Inductor	Supplies load during off periods.
d	Diode	Controls direction of the current.
C	Capacitor	Reduces the voltage ripple.
R	Resistor	Acts as the load.

Experimental Model. This research utilised a simple BC. The system was simulated using MATLAB's Simscape Electrical toolbox [14], with fixed-step iteration applied to the virtual circuit.

Stabilisation was tested by bucking from a source voltage of 48 V to 30 V.

The system was controlled by applying the appropriate duty cycle, generated by a controller, to the *MOSFET* switch. However, the raw duty cycle (which lies between 0 and 1) could not be applied directly to the switch. Instead, it was first be processed by a PWM generator. This generator operates at a specific frequency and converts the duty cycle into a square wave signal. The duty cycle of the PWM output signal, manifested as a percentage of the total period, dictates the duration of the high state within each cycle and was ultimately used to control the switch [10]. The duty cycle's magnitude was determined, by said controller, using a single key state variable: the voltage.

Unfortunately, controlling the BC proved highly sensitive to the simulation step size; the largest functional step size was $dt = 5 \cdot 10^{-6}$ s. The primary drawback was that this significantly extended the training time for the data-driven models, although their performance was unaffected.

Table 7: Buck Converter Parameters

Symbol	Description	Value
V_{in}	Source Voltage.	48 V
I	Inductance.	10^{-4} H
C	Capacitance.	10^{-3} F
R	Resistance.	10 Ω
f	Switching Frequency.	10^{3} Hz
dt	Fixed Step Size.	$5 \cdot 10^{-6}$ s
t_{max}	Stop Time.	0.6 s

Parameters. The parameters used in the simulation of the DC-DC BC are provided in Table 7.

3.5 Evaluation Metrics

Evaluation metrics quantify the quality of control systems. As such, four metrics – chosen for their applicability to various systems, encompassing scope, and insight into system performance – are used as measures of system quality throughout this project. They are defined as follows:

- **Stabilisation Time** – A measure (in seconds) of how long a system takes to reach and maintain equilibrium (within a defined margin of error).
  - A shorter stabilisation time indicates faster controller response time and better overall performance.
- **Training Time** – A measure (in simulations run) of how quickly a self-learning model converges.
  - Short training times are desirable for practical deployment, while longer training times may lead to resource inefficiency.
- **Robustness** – A measure of how a controller's performance changes when presented with irregularities or variations in system parameters.
  - A robust controller maintains stable performance despite external disturbances.
- **Steady-State Error** – How stable the system remains (as a range) after reaching equilibrium over a designated period.
  - A low steady-state error implies accurate tracking of the desired output, while a high error may lead to unwanted deviations.

The definitions of the terms "equilibrium" and "convergence" are further clarified within the contexts of the systems and methods under consideration:

Equilibrium.

- Inverted Pendulum – Equilibrium is achieved when the pendulum remains vertical. Here the angle, θ, to the vertical is equal to zero (±0.05 rad)
- DC-DC BC – Equilibrium is achieved when the output voltage equals the reference voltage (±0.5 V).

In this experiment, the definition of equilibrium is relaxed to include a small margin of error. Furthermore, the system must maintain the equilibrium state for a defined time ($\geq$ 1 s) to handle cases where equilibrium is momentarily achieved, but the output signal subsequently overshoots, undershoots, or deviates from the desired trajectory.

Convergence.

- Q-Learning – When the cumulative reward per episode stabilises for at least 100 consecutive episodes.

4 RESULTS, ANALYSIS, AND DISCUSSION

Following the details of the control methods provided in the Methodology section, this section presents and analyses the results of the experimental research, focusing on the performance of these methods according to defined evaluation metrics.

For the data-driven methods, results were averaged across ten trained models for the IP and five trained models for the DC-DC BCs, reflecting the differences in their respective training durations. Results for NEAT were provided by Matthew Fleischman (2024) [11].

Furthermore, a comprehensive overview of the results can be found in Appendix A.

4.1 Stabilisation and Steady-State

A controller should aim to stabilise control systems quickly, in a manner that minimises deviations whilst in steady-state, thereby guaranteeing sustained stability.

4.1.1 Inverted Pendulum. In the case of this system, all the considered control methods were able to successfully stabilise the IP for starting angles θ_0 within the range $[-\pi/3, \pi/3]$.

The mean stabilisation time, $\bar{t}_s$, was calculated by averaging the stabilisation times across sampled angles within the considered range. Meanwhile, its associated uncertainty, represented by the standard deviation (SD), σ_{t_s}, was selected to illustrate the variability in stabilisation times across the angular range [2].

Once stabilisation was achieved, all methods managed to maintain a steady-state error well within the equilibrium bound of 0.00 ± 0.05 rad.

The stabilisation times of the control methods for the IP are presented in Figure 5.

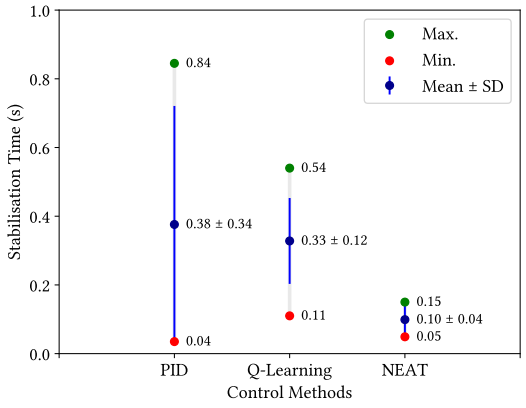

Figure 5: Stabilisation times of control methods, for the inverted pendulum system, averaged over the range $\theta \in [-\pi/3, \pi/3]$ rad.

Figure 5 shows that NEAT is the top-performing control method in terms of stabilisation time, although it has a slightly longer minimum stabilisation time than PID. This can be attributed to NEAT's ability to evolve both its structure and parameters, as well as its exploration capabilities. NEAT's evolution of ANNs enables it to adapt to complex environments and discover sophisticated state representations. This offers a distinct advantage over methods like Q-Learning, which is limited by its finite, manually-designed Q-table, and PID, which lacks state representation altogether. In addition, NEAT's process of selecting and evolving the best-performing offspring enables it to identify optimal strategies, suited to the environment, more efficiently than Q-Learning's functional, yet primitive, epsilon-greedy approach and PID – which has no inherent exploration capabilities.

The disparity between PID's minimum and maximum stabilisation times, coupled with the notable skew towards the minimum, suggests that PID's average stabilisation times (and SD) are inflated by outliers resulting from large initial angular displacements; since a larger stabilisation time results from larger angular displacements. In contrast, Q-Learning's stabilisation times appear consistent and are more evenly distributed. Thus, one can conclude that Q-Learning outperforms PID, in terms of stabilisation time, from larger initial angular displacements only.

It should also be noted that despite being trained to stabilise for only 4 seconds, both data-driven methods were able to remain stable for over 30 seconds when implemented as controllers.

4.1.2 Buck Converter. For the BC, all the methods evaluated successfully stabilised the system from the 48 V source voltage to the desired reference voltage of 30 V. It is also important to note that the NEAT implementation was carried out on a buck-boost converter (BBC), however only its bucking capabilities were utilised.

The mean stabilisation time, $\bar{t}_s$, and the SD, σ_{t_s}, were calculated by averaging these metrics across all (three) fully-trained models.

Once stabilisation was achieved, all methods were able to maintain a steady-state error within the equilibrium range of 0.00 ± 0.5 V and could do so indefinitely.

The stabilisation times of the control methods for the BC are presented in Figure 6.

Comparing Figure 6 (BC) to Figure 5 (IP) one sees that the BC data is more concentrated. This difference arises because the IP stabilisation times were averaged over a range of angles, with larger angles significantly extending stabilisation times, whereas the BC's times were calculated using a single reference voltage.

Figure 6 shows that Q-Learning is the top-performing control method for the BC in terms of stabilisation time. Without external forces, like gravity, pushing the system away from equilibrium, the relationship between the system's state space and dynamics becomes simplified. This allows Q-Learning, whose performance relies heavily on effective state discretisation, to explore and learn optimal control strategies within a manageable state space. In contrast, NEAT may require more training to converge or be stuck in a local extremum, while PID lacks the adaptability of data-driven methods due to its fixed gain coefficients.

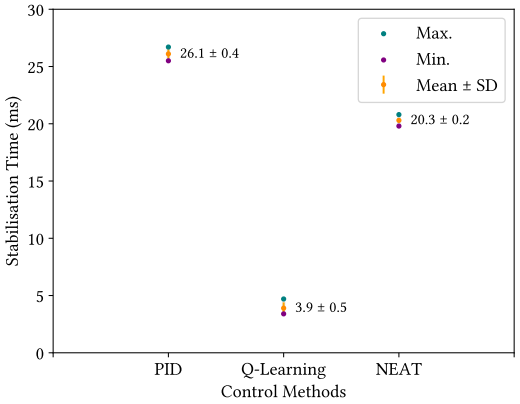

Figure 6: Stabilisation times of control methods applied to a buck converter with a source voltage of 48 V, stabilising to 30 V.

4.2 Training

Training involves optimising a model, over time, to improve performance. Key factors that gauge the success of this process include training time, which depends on the method used and the complexity of the model, and convergence, the point at which further training yields diminishing returns.

4.2.1 Training Time. When deploying control methods, training time can significantly impact a method's practicality; methods with long training durations may result in delays and impose higher computational requirements. All data-driven methods require some degree of training, whether that be populating Q-Learning's Q-Table or optimising the weights and structure of NEAT's ANNs. In contrast, PID controllers do not require direct training; instead, time is needed to tune the controller's gain coefficients.

The training time for data-driven methods will use simulations (equivalent to Q-Learning's episodes or NEAT's generations) as the measurand. This avoids distortions that could arise from using alternative measurands, such as elapsed time in seconds, which can be influenced by factors like: computational power, degree of multiprocessing, and use of multitasking.

The training times of each method are presented in Table 8.

Table 8: Training Time (in simulations) of Data-Driven Control Methods.

Control Method	Inverted Pendulum	Buck Converter
Q-Learning	1500	1250
NEAT	1750	810

The efficiency of a method, in terms of training time for a given system, cannot be evaluated in isolation from its performance. As such, NEAT's significantly faster stabilisation time outweighs Q-Learning's marginally shorter training period for the IP, making NEAT the preferred approach. For the BC, while Q-Learning outperforms NEAT, its nearly twice as long training time raises two

questions: does the additional performance justify the extra training time, and could extending NEAT's training period improve its performance, potentially making it more viable than Q-Learning? The answer to the first question is generally yes, while the second remains to be seen and is subject to further testing. For now, Q-Learning is the recommended method for the BC.

4.2.2 Convergence. The convergence of the Q-Learning algorithm used to stabilise an IP is presented in Figure 7.

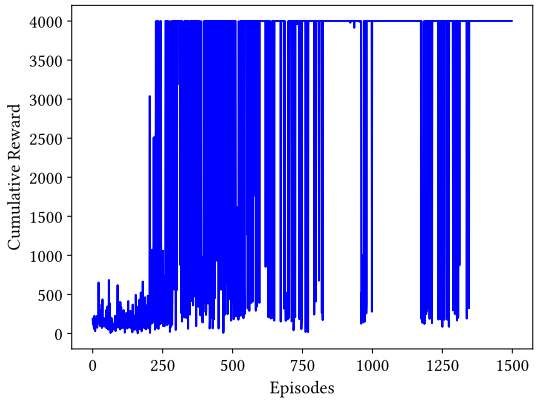

Figure 7: Convergence of Q-Learning over 1500 training episodes for the inverted pendulum system.

Figure 7 shows that Q-Learning initially requires 200 episodes to develop a functional control strategy, followed by 600 episodes spent refining its strategy and exploring alternative methods. Beyond this, improvements are incremental, with occasional failures attributed to epsilon-greedy action selection. The final 150 training episodes achieve the maximum cumulative reward, indicating convergence.

Similarly, the convergence of Q-Learning applied to the BC system is illustrated in Figure 8.

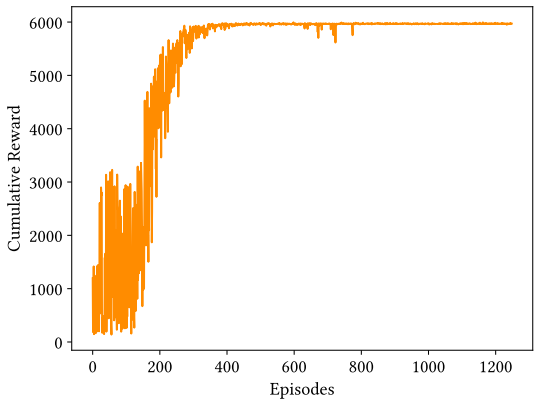

Figure 8: Convergence of Q-Learning over 1250 training episodes for the buck converter system.

Figure 8 shows that the convergence of the Q-Learning implementation for the BC is significantly smoother than that of the IP, likely due to the lower learning rate associated with the BC system. Initial growth is slow, but steady, taking 400 episodes to reach the maximum cumulative reward. Afterwards, occasional refinements are made to the controller's strategy, with convergence achieved around the 800^{th} episode and minimal progress made thereafter.

4.3 Robustness

Robustness refers to a control method's ability to maintain effective performance in spite of irregularities or disturbances.

4.3.1 Mitigating Overfitting. To prevent overfitting during the training of data-driven methods for control of the IP system, the models were trained to stabilise the pendulum across the entire range of angles: $[-\pi/3, \pi/3]$ rad. Random angle sampling was employed to ensure that, throughout an extensive training process of 1000 episodes or more, a diverse range of initial angles was explored.

This allowed Q-Learning to effectively learn how to stabilise the pendulum, including from angles not explicitly encountered during training, rather than merely memorising a set of actions to stabilise it from a single position.

Similarly, the BC, which only needed to be trained to stabilise to 30 V, was instead trained across a range of reference voltages $V \in [V_r - \delta, V_r + \delta]$ to develop a more general stabilisation strategy and avoid overfitting; in this case, $\delta = 0.05V_r$ produced satisfactory results.

The effectiveness of this adjustment to the BC's training regimen is demonstrated by its ability to stabilise signals using a generalised strategy, even for reference voltages in the range $V_r \in [15, 35]$ V, despite not being specifically trained to do so. However, stabilising to reference voltages very close to or far from the (48 V) source voltage likely requires more specialised training.

4.3.2 Exposure to Noise. In real-world applications, controllers can often encounter noise, which can disrupt system performance, causing deviations from desired behaviour; the ability to effectively respond to noise is, thus, a critical factor in ensuring reliable operation.

Noise was introduced to a system by sampling normally distributed white noise and applying it to the state variables utilised by the controllers; the normal distribution's SD determines the noise's intensity. This intensity is a function of both the noise power – a tunable parameter – and the simulation's sample time, with uncertainties propagated according to the product of powers formula [19]. By adjusting the noise intensity, the threshold noise – the minimum noise level that caused the controller to fail in stabilising angles within the considered range – was identified for each control method. This metric, shown in Table 9, will be used to assess each method's resilience to noise.

Data-driven methods, like Q-Learning and NEAT, show greater noise resilience compared to PID, largely due to their training process, which prepares them to handle sudden changes in parameters. Interestingly, while NEAT outperforms Q-Learning in noise resilience for the BC, it is less robust to changes in reference voltage; this suggests that some degree of overfitting may enhance noise resilience.

Table 9: Noise Threshold of Control Methods.

Control Method	Inverted Pendulum (rad)	Buck Converter (V)
PID	0.08 ± 0.01	0.37 ± 0.01
Q-Learning	0.14 ± 0.02	0.28 ± 0.01
NEAT	0.10 ± 0.02	0.89 ± 0.03

Counterintuitively, when the data-driven methods were trained on noisy input data, they exhibited reduced noise resilience. This is likely due to noise-induced variations during training hindering the methods' ability to develop a general control strategy, ultimately preventing convergence.

4.3.3 Performance under Constraints. The data-driven control methods, particularly Q-Learning, were able to effectively stabilise the IP using a maximum controlling force of 30 N. This is ideal for real-world applications, where actuators delivering smaller forces are more cost-effective than those providing larger forces. In contrast, when a PID controller operated under these constraints – with the same maximum output force – its performance significantly deteriorated, as shown in Figure 9.

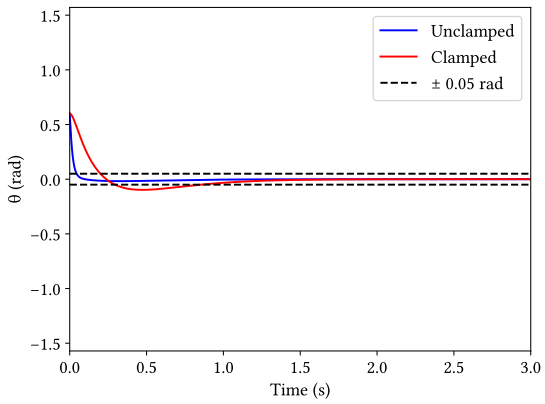

Figure 9: Comparison between PID and clamped PID control for the inverted pendulum from an initial angle of 0.6 rad.

A comparison of the responses in 9 shows that the unclamped controller achieved stabilisation significantly faster, reaching steady-state in 0.09 s, compared to 1.03 s; furthermore, the clamped controller failed to regulate the process in the presence of any amount of noise. The rapid stabilisation of the unclamped controller necessitated a force of 3000 N: two orders of magnitude greater than what was required for the clamped controller. This enormous force requirement would translate to a significantly greater implementation cost in practical applications.

The ability of data-driven methods to operate effectively with limited resources, in both ideal and noisy environments, highlights their robustness in comparison to PID.

5 CONCLUSIONS

This research aimed to evaluate the performance of Q-Learning, compared to PID control and NEAT, in non-linear binary systems, focusing on key metrics such as stabilisation time, training time, robustness, and steady-state error. The findings address the central research question by highlighting the strengths and weaknesses of each control method, offering insights into their practical applications and future development.

5.1 Summary of Findings

The effectiveness of data-driven methods, as evidenced by the results, largely depends on the complexity of a system's state space, particularly how the controller's response is influenced by the current state and, potentially, the environment. NEAT performs better in more complex conditions, particularly when provided sufficient training to converge. On the other hand, Q-Learning, with its reliance on effective state discretisation, is more adept in simpler settings where the state space is less dependent on external factors. Meanwhile, PID control offers consistent, "good-enough" performance but generally lags behind data-driven methods. This is due to PID's lack of system knowledge and inherent rigidity, whereas data-driven methods demonstrate superior robustness and adaptability by learning intricate system behaviours through training.

Experimentation with reward functions for Q-Learning showed that carefully adjusting these functions can encourage desired behaviours, such as faster stabilisation, and that sharply decreasing unimodal reward functions promote rapid and direct stabilisation through focused feedback. These insights contribute to XAI by improving the interpretability of model behaviour, thereby increasing transparency and trust in ML decision-making processes.

5.2 Practical Applications

Both data-driven, state-based methods are well-suited for real-world controllers managing non-linear binary control systems, with NEAT preferred for dynamically complex environments and Q-Learning for systems with minimal dependence on external factors. Their resource efficiency, robustness, and reliable performance make them viable options for deployment across a range of industrial applications, most notably in electronics.

5.3 Future Work

Regarding potential extensions of this work, further training of NEAT on the BC could help determine whether its performance is due to a lack of convergence or if it inherently performs worse than Q-Learning. Additionally, implementing and testing these controllers on real-world systems would be valuable to identify any considerations that may have been overlooked in the virtual training environment; this approach is supported by the findings of Mohammad Safeea and Pedro Neto (2024), who observed system failures due to such overlooked factors [25].

It remains to be seen whether data-driven methods ultimately represent the next logical evolution of the field of control theory. However, the findings of this research indicate significant promise in this direction.

Acknowledgments

Firstly, I would like to express my gratitude to my supervisor, Prof. K. Prag, for her invaluable guidance and feedback throughout this research project. I would also like to thank my team members, M. Fleischman and N. Sartori. I am grateful to Matt for his extensive work with NEAT and the countless hours he spent learning and configuring MATLAB models, and to Nicola for his insights into RL algorithmic design.

Supplementary Material

The source code and models used in this research can be accessed, openly and freely, at https://github.com/AJ-Levy/Data-Driven-Control.

References

[1] Mirza F. Adnan, Mohammad A. Oninda, Mirza M. Nishat, and Nafiul Islam. 2017. Design and Simulation of a DC-DC Boost Converter with PID Controller for Enhanced Performance. *International Journal of Engineering Research & Technology* 06 (2017). https://doi.org/10.17577/IJERTV6IS090029

[2] Douglas G Altman and J Martin Bland. 2005. Standard deviations and standard errors. *BMJ* 331, 7521 (Oct. 2005), 903. https://doi.org/10.1136/bmj.331.7521.903

[3] Charles W. Anderson Andrew G. Barto, Richard S. Sutton. 1983. Neuronlike Adaptive Elements That Solve Difficult Learning Control Problems. *IEEE Transactions on Systems, Man, and Cybernetics* (1983), 834–846.

[4] Karl J. Astrom and Tore Hagglund. 1995. *PID Controllers Theory, Design and Tuning* (second ed.). Instrument Society of America, 59–92.

[5] Ramashis Banerjee and Arnab Pal. 2018. Stabilization Of Inverted Pendulum On Cart Based On Pole Placement and LQR. In *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*. IEEE. https://doi.org/10.1109/iccsdet.2018.8821196

[6] Alexandre S. Bazanella, Lucíola Campestrini, and Diego Eckhard. 2023. The data-driven approach to classical control theory. *Annual Reviews in Control* 56 (2023), 100906. https://doi.org/10.1016/j.arcontrol.2023.100906

[7] Stuart Bennett. 1996. A brief history of automatic control. *IEEE Control Systems* 16, 3 (1996), 17–25. https://doi.org/10.1109/37.506394

[8] Alain Bensoussan, Yiqun Li, Dinh P. Nguyen, Minh-Binh Tran, Sheung C. Yam, and Xiang Zhou. 2020. Machine Learning and Control Theory. (2020). arXiv:2006.05604 [cs.LG]

[9] Caleb M. Bowyer. 2022. Strategies for Decaying Epsilon in Epsilon-Greedy. (2022).

[10] Shawn Dietrich. 2022. Understanding the Basics of Pulse Width Modulation (PWM). *Control Automation* (2022).

[11] Matthew Fleischman. 2024. Comparative Analysis of NEAT, PID, and RL Methods for Effective Control of Non-Linear Systems. *UCT* (2024).

[12] William Holderbaum. 2012. Control of Binary Input Systems. *IOSR Journal of Engineering* 02, 12 (Dec. 2012), 01–15. https://doi.org/10.9790/3021-021210115

[13] Zhong-Sheng Hou and Zhuo Wang. 2013. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences* 235 (2013), 3–35. https://doi.org/10.1016/j.ins.2012.07.014

[14] The MathWorks Inc. 2024. *MATLAB version: 9.13.0 (R2022b)*. Natick, Massachusetts, United States. https://www.mathworks.com

[15] Sardor Israilov, Jesús Fu, Sánchez-Rodríguez, Franco Fusco, and Guillaume Allibert. 2023. Reinforcement learning approach to control an inverted pendulum: A general framework for educational purposes. *PLOS ONE* 18, 2 (2023), e0280071. https://doi.org/10.1371/journal.pone.0280071

[16] Jiyan Li. 2023. Analysis And Comparison of Different Tuning Method of PID Control in Robot Manipulator. *Highlights in Science, Engineering and Technology* 71 (11 2023), 28–36. https://doi.org/10.54097/hset.v71i.12373

[17] James C. Maxwell. 1867. On Governors. *Proceedings of the Royal Society of London* 16 (1867), 270–283. http://www.jstor.org/stable/112510

[18] Norsyahidatul F. Nanyan, Mohd A. Ahmad, and Baran Hekimoğlu. 2024. Optimal PID controller for the DC-DC buck converter using the improved sine cosine algorithm. *Results in Control and Optimization* 14 (2024), 100352. https://doi.org/10.1016/j.rico.2023.100352

[19] UCT Department of Physics. 2020. Introduction to Measurement in the Physics Laboratory. (2020), 28, 60.

[20] OpenAI. 2022. Pendulum. *OpenAI Gym* (2022).

[21] Krupa Prag, Matthew Woolway, and Celik Turgay. 2022. Toward Data-Driven Optimal Control: A Systematic Review of the Landscape. *IEEE Access* 10 (2022), 32190–32212. https://doi.org/10.1109/access.2022.3160709

[22] Christiaan J. Pretorius, Mathys C. du Plessis, and John W. Gonsalves. 2017. Neuroevolution of Inverted Pendulum Control: A Comparative Study of Simulation Techniques. *Journal of Intelligent & Robotic Systems* 86, 3–4 (2017), 419–445. https://doi.org/10.1007/s10846-017-0465-1

[23] Risto Miikkulainen. 2010. *Neuroevolution*. Springer US, Boston, MA, 716–720. https://doi.org/10.1007/978-0-387-30164-8_589

[24] Everett Rogers. 1999. Understanding Buck Power Stages in Switchmode Power Supplies. *Texas Instruments* (1999), 1–7.

[25] Mohammad Safeea and Pedro Neto. 2024. A Q-learning approach to the continuous control problem of robot inverted pendulum balancing. *Intelligent Systems with Applications* 21 (2024), 200313. https://doi.org/10.1016/j.iswa.2023.200313

[26] Corrado Santoro. [n. d.]. Handling System Limits PID Control with Saturation.

[27] Fredy H. Sarmiento, Diego F. Molano, and Mariela C. Ortiz. 2012. Optimization of a neural architecture for the direct control of a Boost converter. *Tecnura* 16 (2012), 41–49. https://www.redalyc.org/pdf/2570/257024143004.pdf

[28] Kenneth O. Stanley and Risto Miikkulainen. 2002. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* 10, 2 (2002), 99–127. https://doi.org/10.1162/106365602320169811

[29] K Sundareswaran, Kiran Kuruvinashetti, B Hariprasad, P Sankar, PS Nayak, and V Vigneshkumar. 2014. Optimization of dual input buck converter control through genetic algorithm. *IFAC Proceedings Volumes* 47, 1 (2014), 142–146.

[30] Elisa S. Varghese, Anju K. Vincent, and V. Bagyaveereswaran. 2017. Optimal control of inverted pendulum system using PID controller, LQR and MPC. *IOP Conference Series: Materials Science and Engineering* 263, 5 (2017), 052007. https://doi.org/10.1088/1757-899X/263/5/052007

[31] Jia-Jun Wang. 2011. Simulation studies of inverted pendulum based on PID controllers. *Simulation Modelling Practice and Theory* 19, 1 (2011), 440–449. https://doi.org/10.1016/j.simpat.2010.08.003

[32] Christopher J. Watkins. 1989. *Learning From Delayed Rewards*. Ph. D. Dissertation. University of Cambridge. https://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf

[33] Christopher J. Watkins and Peter Dayan. 1992. Technical Note: Q-Learning. *Machine Learning* 8, 3/4 (1992), 279–292. https://doi.org/10.1023/a:1022676722315

A Overview of Results

Table 10: Overview of Results for all Control Methods.

System	Evaluation Metric	PID	Q-Learning	NEAT [11]
Inverted Pendulum	Stabilisation Time (s)	0.38 ± 0.34	0.33 ± 0.12	0.10 ± 0.04
	Steady-State Error (rad)	0.00 ± 0.05	0.00 ± 0.05	0.00 ± 0.05
	Training Time (sims)	–	1500	1750
	Noise Threshold (rad)	0.08 ± 0.01	0.14 ± 0.02	0.10 ± 0.02
Buck Converter [48 V]	Stabilisation Time (ms) [30 V]	26.1 ± 0.4	3.9 ± 0.5	20.3 ± 0.2
	Steady-State Error (V)	0.0 ± 0.5	0.0 ± 0.5	0.0 ± 0.5
	Training Time (sims)	–	810	1250
	Noise Threshold (V)	0.37 ± 0.01	0.28 ± 0.01	0.89 ± 0.03