Comparing Control Methods for Non-Linear Binary Control Systems Using Data-Driven Techniques: A Proposal

Matthew Fleischman
University of Cape Town
Cape Town, South Africa
FLSMAT002@myuct.ac.za

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

Nicola Sartori
University of Cape Town
Cape Town, South Africa
SRTNIC002@myuct.ac.za

ABSTRACT

Robust 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, 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 aims to provide a thorough comparison between traditional PID control and data-driven methods, including Q-learning, Proximal Policy Optimisation (PPO), and Neuroevolution of Augmenting Topologies (NEAT). Each control method will be applied to non-linear binary control systems, specifically the inverted pendulum and DC-DC buck-boost converters (BBCs), to facilitate a comprehensive assessment of their capabilities. The contributions of this proposed research intend to provide a clear and informative comparison of control methods through several metrics that quantify the quality, efficiency, and reliability of their performance.

1 INTRODUCTION

Control theory is an area of study that focuses on the management, direction, and regulation of system behaviour. Over time, advancements in control techniques have driven technological progress, making the continuous development of effective control methods crucial to shaping humanity's technological future [10].

1.1 Model-Based Control

The model-based control (MBC) framework relies on a theoretical representation of the system to enable predictions and manipulate the system's behaviour. This representation is the combination of a mathematical model of the system's dynamics and a set of assumptions aiming to give an accurate description of the system; however, the efficacy of MBC suffers when such a model does not exist or when the system is poorly understood, complex, or non-linear (governed by a set non-linear differential equation). This is largely due to the resultant inaccuracies that would be present in the representations of such systems. [12, 22].

1.2 Data-Driven Control

Industrial systems have become increasingly complex, commonly exhibiting non-linear dynamics and generating large amounts of data. Through leveraging modern computational power, this data can be utilised via big data analysis and machine learning (ML). This application of ML and the shortcomings of MBC have inspired interest in developing an alternative paradigm to model-based control: data-driven control (DDC) [11]. DDC uses available historical data and input/output (I/O) data to learn, thereby improving the control quality of a system [12, 22]. DDC is a flexible controller design framework that can be used to enhance MBC designs or develop model-free controllers. Data-driven controllers can provide good performance for non-linear complex systems without relying on the accuracy of a system's mathematical model.

RL is a valuable ML technique for DDC designs. RL employs a reward mechanism and trial-and-error strategy to accomplish specific tasks. Within control theory, RL allows a controller to determine the best control policy without understanding the system dynamics. [6].

The RL techniques considered in this proposal are PPO and Q-learning. PPO is a popular gradient and policy-based RL technique. Q-learning is a popular off-policy RL technique.

Neuroevolution is another promising field of ML used in DDC design. Evolution-based methods, such as genetic algorithms, are applied to artificial neural networks (ANNs) to improve them iteratively over generations [24, 28]. This paper considers NEAT, a popular neuroevolution-based algorithm.

1.3 Binary Control Systems

There exists a lot of literature evaluating the performance of ML techniques in controller design but direct comparison between them is relatively sparse. This research compares the performance of the chosen data-driven techniques in controller design for non-linear binary control systems, to better understand each method's benefits and shortfalls.

Binary control systems can be controlled with two actions – for example "on or off", or "true or false". This research aims to evaluate and compare the performance of DDC methods applied to two binary control systems: the inverted pendulum, chosen due to its regular use as a benchmark problem in the field of control theory, and DC-DC BBCs, which have found widespread use in both industrial and consumer settings.

1.4 Aims

This section traces the progression of key project aspects: its title, problem statement, and research questions.

1.4.1 ***Project Title.*** Drawing from the above research questions the title for this project will be: *A Comparative Study of Control Methods for Non-Linear Binary Control Systems Using Data-Driven Techniques.*

1.4.2 ***Problem Statement***.
Existing literature on non-linear binary control systems lacks a thorough comparison of the effectiveness of established data-driven control methods.

1.4.3 ***Research Question***.

(1) How do DDC methods compare to traditional PID control with respect to control quality for non-linear binary systems?
(2) How do the DDC methods, Q-learning, PPO, and NEAT, compare to one another in terms of stabilisation time, training time, robustness, and steady-state error?

The metrics used to quantify the performance quality for a specific control method are detailed in Section 3.6.

2 BACKGROUND

This section provides a brief overview of the available literature surrounding the application of PID, Q-learning, PPO, and NEAT (as control methods) to the inverted pendulum and DC-DC BBC systems. Gaps in the literature are identified and discussed.

2.1 Inverted Pendulum

The inverted pendulum is considered a classic benchmark problem in the field of control theory due to its: simple structure, nonlinearity, and sensitivity to initial conditions [29].

2.1.1 *PID.*

2.1.2 Q-learning. Q-learning was successfully employed to stabilise a real-world inverted pendulum. 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].

2.1.3 PPO. In an experiment, PPO was shown to achieve comparable performance for transient and steady-state responses to PID for an inverted pendulum system. The PPO implementation achieved better overshoot and steady-state error performance than PID. However, PPO was not the overall best-performing method. This was attributed to how the reward function only concerned itself with the pendulum not falling and therefore did not reinforce other areas of performance – like speed to reach the desired setpoint. This highlights a shortfall of RL methods in that the reward function may fail to optimize all aspects of performance for the desired system [5].

2.1.4 NEAT. Multiple neuroevolution methods, including NEAT, FFNNs, and ERNNs, were compared in optimising the control of a real-world inverted pendulum [23]. One advantage of NEAT was noted: It did not require the topology to be decided before training. However, NEAT's performance was similar to FFNNs and surpassed ERNNs. NEAT did not do as well as expected. Another paper found that the controller's stability heavily relied on the sophistication of the fitness function used [2].

2.2 DC-DC BBCs

DC-DC BBCs have gained widespread use in various settings, such as renewable energy generation, electric cars, and small-scale electronic appliances [1, 19]. As a result, the precise and efficient control of voltage signal transitions is crucial to industry.

2.2.1 PID. A (simulated) boost converter was used to step up the source voltage to the correct reference voltage with an overshoot error of $2 - 3\%$; however, the converter took very long to reach the reference voltage (around 1000 s) [1]. Another paper used a closed-loop PID controller to control a DC-DC boost converter. The boost converter was simulated using MATLAB Simulink and measured an efficiency of over 90% [3]. In both papers, the PID controller enabled efficient DC-DC voltage conversion, however, the manual nature of the controller's tuning suggests that it lacks robustness in handling any potential irregularities – such as noise or disturbances.

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

2.2.3 PPO. The use of PPO in designing an optimal controller for a DC-DC BBC system was explored with a comparative experiment. The PPO implementation was found to have the fastest settling time, though it had poorer qualitative measurements regarding transient time in boost cases. Transient time was defined as the time between the end of the simulation duration and the first recorded data value that violated the error bound around the given reference value. This showed that the PPO implementation was not the most robust in terms of stability. The PPO controller was also tested in the presence of noise applied to the transmitted signal and found to be more robust than the PI controller. Lastly, it was found that the PPO method failed to achieve convergence across all experiments for buck cases [21].

2.2.4 NEAT. There is no available literature on the use of NEAT in DC-DC voltage conversion, therefore further research is required to understand NEAT's capabilities in this domain.

3 METHODOLOGY

Before a meaningful comparison can be made, each of the considered control methods needs to be well-understood in a general sense. Once this understanding is established, each control method can be contextualised within the framework of the two dynamic systems under consideration, which also require definition. Finally, the effectiveness of each of these control strategies can be assessed – in the context of a system – using relevant evaluation metrics.

3.1 PID

PID controllers attempt to manage a system so that its output follows a desired signal. This is achieved by taking an error signal, calculated by comparing the system's real-world and desired outputs and adjusting the system's input to move the real-world output closer to the desired output. They have three essential terms which regulate a process [6]. The Proportional (P) term adjusts the output proportionally to the error; a large error means strong corrective action. The Integral (I) term responds to accumulated error over time; this drives the system to the desired output as time increases even if it consistently falls short. Finally, the Derivative (D) term anticipates future error by considering its current rate of change; a greater or more rapid change results in a stronger signal-damping effect [4, 17].

Formulating this explanation mathematically, one finds that the controller output signal $u(t)$ – in terms of the error signal $e(t)$ and gain coefficients K– is [4],

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

Signal analysis is often carried out in the frequency domain – instead of in the time domain – achieved through the use of the Laplace transform [15]. The PID controller's transfer function $L(s)$, the Laplace transform of $u(t)$, illustrates the rationale behind the preference for frequency-based analysis through its formulation,

$$L(s) = K_p + \frac{K_i}{s} + K_d s. \tag{3.1.2}$$

While trial and error remain the most common approach for tuning PID gain coefficients, there exist optimisation techniques that enable automatic tuning of these parameters. One such procedure, the Sine Cosine Algorithm (SCA), iteratively adjusts candidate solutions using sine and cosine functions, exploring the search space efficiently and allowing candidates to converge toward optimal solutions [19].

Table 1: PID Parameters

Symbol	Description
K_p	Proportional term gain coefficient.
K_i	Integral term gain coefficient.
K_d	Derivative term gain coefficient.

PID control is most effective in linear systems, but can also be applied successfully to non-linear systems when the nonlinearities are weak. In such cases, linearisation approximations offer a sufficiently accurate representation of the true system.

3.2 Q-learning

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 [31].

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 [14].

The pseudocode for a general application of the Q-learning algorithm is given in Algorithm 1 [14, 32].

Algorithm 1 Implementation of Q-learning

```
Intialise $Q(s, a)$ $\forall$ $s \in S$ and $\forall$ $a \in A$ with zeros
for ep in episodes do
    Select random initial state $s \leftarrow s_i$
    while $s \neq s_{goal}$ do
        Using $\epsilon$-greedy strategy select action $a$
        Define reward function $r$
        $s' \leftarrow s_{i+1}$
        $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \cdot \max Q(s') - Q(s, a)]$
        $s \leftarrow s'$
    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.

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.

3.3 PPO

PPO is a gradient-based, on-policy RL method. It aims to optimise a decision policy through multiple iterations of updates. It uses batches of experience gathered from interactions with the environment, discarding old batches as the policy is updated. Each experience is a tuple of a state, a corresponding action, and a reward, (s, a, r) at a particular time step t. PPO restricts the degree to which consecutive policies differ, reducing instability in the learning process and ensuring consistent progress. It does so by minimising a clipped surrogate objective function [5, 8, 16, 21, 26, 30].

PPO uses an Actor-Critic system implemented through training two neural networks. The actor-network optimises the policy and the critic-network optimises an approximation value function. The actor-network maps states to a Gaussian distribution over actions. That is, the actor returns the probability of taking action, a_t, from the set of possible actions for the current state, s_t, at time t. The critic is responsible for determining how valuable the actions returned by the actor-network are, relative to other available actions. It calculates the expected discounted long-term reward, known as value. Reward is the immediate benefit of taking the action and value is the accumulated benefits of multiple future rewards from the new state [5, 8, 16, 21, 26, 30].

The pseudocode for a general application of the PPO algorithm is given in Algorithm 2 [5, 16, 21, 27].

The surrogate objective function is given by [27],

$$L_{PPO} = \mathbb{E}[min\,(arg_1, arg_2)], \tag{3.3.1}$$

Algorithm 2 Implementation of PPO

```
Randomly initialize $\theta$ and $\phi$
Initialise buffer $B$
for iteration = 1 to $N$ do
    for episode = 1 to $M$ do
        Generate a trajectory using policy $\pi_\theta$ in the environment until $H$ time steps.
        Store transitions $(a_t, s_t, r_t)$ in $B$
        Calculate $\hat{A_t}$ using any method of advantage estimation
    end for
    for epochs = 1 to $K$ do
        Randomly select a mini-batch $N_B$ from $B$
        Optimise the surrogate objective function w.r.t $\theta$, via stochastic gradient ascent
        Optimise the value function by minimising the critic loss function, via gradient descent method w.r.t $\phi$
        Update $\theta_{old} \leftarrow \theta$
        Update $\phi_{old} \leftarrow \phi$
    end for
    Clear $B$
end for
```

$$arg_1 = \frac{\pi_\theta \left(a_t|s_t\right)}{\pi_{\theta old} \left(a_t|s_t\right)} \hat{A_t}, \tag{3.3.2}$$

$$arg_2 = clip \left(\frac{\pi_\theta \left(a_t|s_t\right)}{\pi_{\theta old} \left(a_t|s_t\right)}, 1 - \epsilon, 1 + \epsilon \right) \hat{A_t}. \tag{3.3.3}$$

Eqn. (3.3.2) denotes a policy probability ratio between the current and previous policy, indicating the likelihood of a_t in s_t, for policy π_θ. The advantage function denoted by $\hat{A_t}$ measures whether an action is detrimental or beneficial relative to other available actions for a given state. Eqn. (3.3.3) clips the policy probability ratio between $1 - \epsilon$ and $1 + \epsilon$. Thus if the action taken is beneficial, $\hat{A_t} > 0$, and the action is now more probable, Eqn. (3.3.2) > 0, it limits the extent to which the policy is updated so that it does not move too far and potentially worsen. If the action is undesirable, $\hat{A_t} < 0$, and the action is more probable then it does not limit by how much it corrects the policy.

The critic loss function is given by taking the mean squared error of the action-value function and value function [21],

$$L_c(\phi) = \frac{1}{M} \sum_{t=1}^{M} \left(Q^\pi(s,a) - V(s|\phi)\right)^2 . \tag{3.3.4}$$

The action-value function and value function are given [21],

$$Q^\pi(s,a) = \sum_t \mathbb{E}_{\pi_\phi} \left[R(s_t, a_t)|s, a\right], \tag{3.3.5}$$

$$V^\pi(s) = \sum_t \mathbb{E}_{\pi_\phi} \left[R(s_t, a_t)|s\right]. \tag{3.3.6}$$

The action-value function estimates the expected return when taking action, a, from the initial state, s, and following the current policy. The value function provides a measure of how beneficial it is to be in a particular state.

Table 3: PPO Parameters

Symbol	Description
θ	The policy parameter for the actor-network.
π_θ	The current policy.
ϕ	The value function parameter for the critic-network.
a	An action.
s	A state.
r	The reward for an action-state pair.
ϵ	The hyperparameter for the clip function.
M	The size of the experience mini-batch.

3.4 NEAT

Neuroevolution is an ML technique that uses genetic algorithms and the principles of evolution to produce generations of varying ANNs. During a generation, the input data passes through each ANN, and those that achieve the highest fitness scores are selected to produce the next generation of ANNs. This process optimises the weights and nodes of the fixed-topology ANNs. Random mutations occur at the end of each generation to discover new ways of improving the fitness score. In the context of non-linear control, Neuroevolution can deal with large state spaces and can perform well with limited observations of the environment [9].

NEAT is a form of Neuroevolution created by Stanley and Miikkulainen in 2002 [28]. Unlike traditional Neuroevolution methods, NEAT enables the topology of the ANNs to mutate and self-optimise, reducing the need for human intervention – the structure of ANNs does not need to be decided on in advance [20]. The topology of an ANN can mutate in one of two ways: (1) A pair of previously unconnected nodes is connected with a random weight. (2) A new node is inserted between two previously connected nodes, the old connection is disabled, and new connections are made.

ANNs are crossed over at the end of a generation to produce offspring with favourable traits. Random topological mutations create differently-sized ANNs, introducing complexity in the crossing-over process. This problem is solved by using historical markings to match up genes (nodes and connections) of different ANNs that share the same origin. For more information on the implementation of ANN reproduction, and the use of speciation to protect structural innovation, see Section 3 of "Evolving Neural Networks through Augmenting Topologies" [28].

Defining a valuable fitness function is crucial in discovering an optimal ANN. The fitness function determines which ANNs are allowed to reproduce, mutate, and move on to the next generation. The mathematical form of the fitness function is dependent on the evaluation metrics of the specific application.

The pseudocode for a general application of the NEAT algorithm is given in Algorithm 3.

3.5 Applications

To facilitate a comparison of the considered control methods, they are evaluated in the context of their application to the control of the inverted pendulum and DC-DC BBC systems.

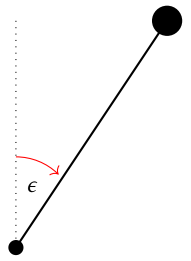
Algorithm 3 Implementation of NEAT

```
Initialise population of $N$ simple neural networks (genomes)
Define fitness function $F$
while iteration number < max iterations do
    for genome in population do
        Assess the fitness of the genome using $F$
    end for
    Sort the population by fitness scores
    if max fitness in population > threshold then
        Break
    end if
    Remove the bottom $x$% of the population
    for genome in remaining population do
        $r \leftarrow$ random value, $r' \leftarrow$ random value
        if $r$ > mutation requirement then
            Mutate the genome's weights or topology
        end if
        if $r'$ > crossover requirement then
            Crossover the genome with another genome
        end if
    end for
    Advance to next generation
end while
return best genome
```

3.5.1 *The Inverted Pendulum.*

The inverted pendulum is a pendulum system in which the centre of mass lies directly above the pivot point – see Fig. 1; the system is intrinsically unstable and will fall over without additional assistance due to a gravitational torque. With only one degree of freedom, the pendulum's motion is confined to angular movement along the arc of a circle [29]. It is a binary control problem because the controller has two states: apply a torque clockwise (CW) or counterclockwise (CCW). The control methods must determine the correct sequence of torque applications to keep the pendulum upright.

Figure 1: Simple Inverted Pendulum



The motion of an idealised pendulum (i.e. no damping due to air resistance) is simulated using Eqn. (3.5.1). The equilibrium position of the pendulum is $\theta = \pi$; angular displacements from this position are considered to be by an angle $\epsilon = \pi - \theta$.

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta = 0, \tag{3.5.1}$$

The relation between torque applied (by the controller) for a time Δt and the resultant angle change of the pendulum is given by Eqn. (3.5.2) – derived in Appendix A. Noise is injected into the system by creating random torque events.

$$\Delta\theta = \omega_0 \Delta t + \frac{1}{2}\frac{\tau}{I}(\Delta t)^2. \tag{3.5.2}$$

Table 4: Pendulum Parameters

Symbol	Description
θ	The principal angle of the pendulum, which increases CCW from the negative y-axis.
ϵ	The angle of the pendulum from equilibrium.
g	Acceleration due to gravity.
L	Length of the pendulum arm.
$\Delta\theta$	The change in angle due to torque.
ω_0	Initial angular velocity.
I	Moment of inertia of the pendulum.
τ	The torque applied.

3.5.2 *DC-DC BBC.*

The second considered application is a DC-DC BBC, shown in Fig. 2. It 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 5. The DC-DC BBC has two states: The buck state lowers the input voltage, and the boost state raises the input voltage [1, 19]. The relationship between the input and output voltages is non-linear, therefore, an effective control method is required to achieve precise voltage conversion.

The output voltage, V_{out}, is given by Eqn. (3.5.3), where D is the duty cycle – the ratio of the switch's on time to the total switching period. Eqn. (3.5.3) implies that V_{out} is greater than V_{in} if $D < 0.5$, and V_{out} is less than V_{in} if $D > 0.5$ [7]. Therefore, the DC-DC BBC is controlled via the *MOSFET* switch. Noise is injected into the system through random switching events.

$$V_{out} = V_{in} \cdot \frac{D}{1-D}. \tag{3.5.3}$$

Figure 2: DC-DC Buck-Boost Circuit

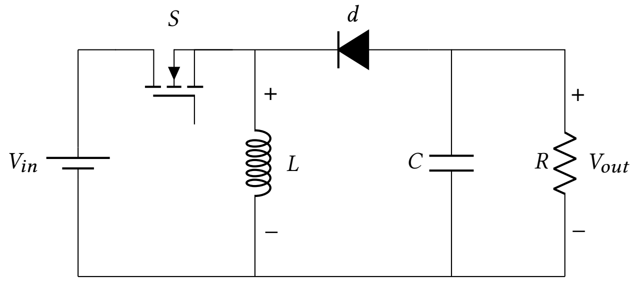


Table 5: DC-DC Buck-Boost Circuit Components

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.

3.5.3 ***Resources Required.*** The control systems are simulated in Python with models designed in MATLAB [13]. The inverted pendulum simulation uses MATLAB Simulink, and the buck-boost circuit simulation uses MATLAB Simscape Electrical. The MATLAB licenses are supplied by the University of Cape Town. Python is used because of its ability to integrate with MATLAB, ease of use, and versatility; Simulink models can be run in Python using the *matlab.engine* package. Each control method – detailed in Sections 3.1 - 3.4 – is implemented in Python and applied to both systems separately. The performance of each of the control methods is evaluated according to the metrics discussed in Section 3.6. The NEAT control method makes use of the *NEAT-python* library [18].

No external datasets are used. The training of data-driven control methods and the assessment of performance for all control methods rely exclusively on data collected from applying each control method to the simulated systems described in Sections 3.5.1 and 3.5.2.

3.6 Evaluation Metrics

Four key 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:

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

Equilibrium and convergence are further clarified within their respective contexts. These clarifications are provided below.

Equilibrium:

- Inverted Pendulum – equilibrium is achieved when the pendulum remains in a vertical position. Here the angle θ to the vertical is equal to zero, the torque due to gravity is equal to zero and the potential energy is at its maximum.
- DC-DC BBC – equilibrium is achieved when the output voltage equals the reference voltage.

In this experiment, the definition of equilibrium is relaxed to include a small margin of error. Additionally, the system must maintain the equilibrium state for a set amount of time to handle cases where equilibrium is momentarily achieved, but the system ultimately overshoots or undershoots.

Convergence:

- Q-learning – When the Q-values in the Q-table stabilise. This can be determined by comparing consecutive Q-values, checking if the difference between them remains below a certain threshold for a set period.
- PPO – When the termination criteria are met. This is when the loss of the value function and surrogate objective function stabilise and the agent's performance in the training environment fails to improve.
- NEAT – When the max fitness score of the current population of nodes is greater than a given fitness threshold.

4 ETHICAL, PROFESSIONAL, AND LEGAL CONSIDERATIONS

Several aspects must be thoughtfully considered when embarking on a project. Developers must ensure that their work aligns with ethical standards. Since this project entails working with virtual models, the only moral considerations are those of team members (such as fair and equal work allocation).

Additionally, maintaining professionalism throughout the project's lifecycle – whether collaborating with team members or interacting with stakeholders – is crucial. Clear communication, regular updates, and a commitment to high-quality work contribute to a successful and professional work environment.

Legal compliance cannot be overlooked. Adherence to intellectual property (IP) rights and licensing agreements is paramount. These complexities can be navigated responsibly by documenting code and models, sharing them openly, and obtaining permissions for the use of IP when necessary.

5 ANTICIPATED OUTCOMES

The success of this project depends on delivering a thorough comparison of the control methods under investigation, specifically in terms of key metrics.

5.1 Research-Based

The results of this research should show (1) a comparison of the performance of data-driven control methods with traditional PID control and (2) a comparison of the performance of PPO, Q-learning, and NEAT.

Using the literature review as a basis, it is believed that all the data-driven methods should outperform PID control – especially in the robustness metric due to the non-linear nature of the systems. Among data-driven control methods, PPO shows the most promise, consistently performing well in literature when used to control non-linear binary control systems. This is not the case for the other methods. The effectiveness of Q-learning and NEAT has been found to vary depending on the implementation quality, making it challenging to predict their performance based on existing literature alone.

5.2 Wider Impact

This research primarily aims to provide a means of comparison between distinct data-driven control methods. This could, for example, simplify the control method selection process for a yet-to-be-designed system, allowing a designer to refer to a table of practical evaluation metrics and choose the method that performs best in the metrics relevant to that control system's requirements.

Although the results of this research are directly linked to the two non-linear binary control systems under investigation, if the performance across both systems remains relatively consistent for each method, one could argue that the findings could apply to all non-linear binary control systems – pending further research. This would make the findings of this project applicable to many other useful systems, such as programmable logic controllers (PLCs), and binary sensors and actuators. Furthermore, industries, such as aerospace, electric utility, and chemical processing would also greatly benefit from this research.

6 PROJECT PLAN

This section outlines critical components that will guide this project from inception to completion.

6.1 Risks

There are several risks that this project faces. These risks are detailed – along with management strategies – in the risk assessment diagram in Appendix B.

6.2 Timeline, Milestones and Deliverables

The Gantt chart in Appendix C shows the different tasks, their due dates, and their work allocations throughout the project's life cycle. Milestones and their due dates are given in Table 6.

Table 6: Due Dates of Milestones

Milestone	Date
Simulation Models Complete	18 May
Control Methods Working	7 June
Optimisation and Testing of Methods Done	17 June
Data Collected	10 July

Deliverables, presented in Table 7, outline the tangible results that will be produced throughout this project.

Table 7: Due Dates of Deliverables

Deliverable	Date
Progress Presentation	22 July
Draft Report Hand-in	23 August
Final Report Hand-in	30 August
Code Submission	9 September
Final Presentation	16 September
Poster Due	27 September
Website Due	4 October

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] Awjin Ahn and Caleb Cochrane. 2014. Using NEAT to Stabilize an Inverted Pendulum. (2014).

[3] R Arulmurugan and N Suthanthira Vanitha. 2012. Optimal design of DC to DC boost converter with closed loop control PID mechanism for high voltage photovoltaic application. *International Journal of Power Electronics and Drive Systems* 2, 4 (2012), 434.

[4] Karl J. Astrom and Tore Hagglund. 1995. *PID Controllers Theory, Design and Tuning* (second ed.). Instrument Society of America, 59–80.

[5] Ahmad Ataka, Andreas Sandiwan, Hilton Tnunay, Dzuhri Radityo Utomo, and Adha Imam Cahyadi. 2023. Inverted Pendulum Control: A Comparative Study from Conventional Control to Reinforcement Learning. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi* 12, 3 (Aug. 2023), 197–204. https://doi.org/10.22146/jnteti.v12i3.7065

[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] Minh Quan Duong, Van Tan Nguyen, Gabriela Nicoleta Sava, Mircea Scripcariu, and Marco Mussetta. 2017. Design and simulation of PI-type control for the Buck Boost converter. In *2017 International Conference on ENERGY and ENVIRONMENT (CIEM)*. 79–82. https://doi.org/10.1109/CIEM.2017.8120769

[8] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, L. Rudolph, and Aleksander Madry. 2020. Implementation Matters in Deep RL: A Case Study on PPO and TRPO. In *International Conference on Learning Representations*. https://api.semanticscholar.org/CorpusID:213580904

[9] Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. 2006. Efficient non-linear control through neuroevolution. In *European Conference on Machine Learning*. Springer, 654–662.

[10] Graham Clifford Goodwin, Stefan F Graebe, Mario E Salgado, et al. 2001. *Control system design*. Vol. 240. Prentice Hall Upper Saddle River.

[11] Zhongsheng Hou, Ronghu Chi, and Huijun Gao. 2016. An overview of dynamic-linearization-based data-driven control and applications. *IEEE Transactions on Industrial Electronics* 64, 5 (2016), 4076–4090.

[12] 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

[13] The MathWorks Inc. 2024. *MATLAB version: R2024a*. Natick, Massachusetts, United States. https://www.mathworks.com

[14] 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

[15] Reto B. Keller. 2023. *Time-Domain and Frequency-Domain*. Springer International Publishing, 305–306. https://doi.org/10.1007/978-3-031-14186-7_5

[16] Matthias Lehmann. 2024. The Definitive Guide to Policy Gradients in Deep Reinforcement Learning: Theory, Algorithms and Implementations. arXiv:2401.13662 [cs.LG]

[17] 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

[18] Alan McIntyre, Matt Kallada, Cesar G. Miguel, Carolina Feher de Silva, and Marcio Lobo Netto. 2024. *neat-python version: 0.92*. https://neat-python.readthedocs.io/en/latest/

[19] 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

[20] Paolo Pagliuca, Nicola Milano, and Stefano Nolfi. 2018. Maximizing adaptive power in neuroevolution. *PloS one* 13, 7 (2018), e0198788.

[21] Krupa Prag, Matthew Woolway, and Turgay Celik. 2021. Data-Driven Model Predictive Control of DC-to-DC Buck-Boost Converter. *IEEE Access* 9 (01 2021),

101902–101915. https://doi.org/10.1109/ACCESS.2021.3098169

[22] Krupa Prag, Matthew Woolway, and Turgay Celik. 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

[23] 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 (2017), 419–445.

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

[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] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. (07 2017).

[27] Asad Shahid, Dario Piga, Francesco Braghin, and Loris Roveda. 2022. Continuous Control Actions Learning and Adaptation for Robotic Manipulation through Reinforcement Learning. *Autonomous Robots* 46 (03 2022). https://doi.org/10.1007/s10514-022-10034-z

[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] 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

[30] Zhe Wang and Tianzhen Hong. 2020. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy* 269 (2020), 115036. https://doi.org/10.1016/j.apenergy.2020.115036

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

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

Appendix A DERIVATION OF EQN. (3.5.2)

The equation that provides the change in angle due to torque can be derived starting with Newton's Second Law for Rotational Motion:

$$\tau = I\frac{d^2\theta}{dt^2}, \tag{A.0.1}$$

which relates the torque applied τ to the moment of inertia I and angular acceleration $\frac{d^2\theta}{dt^2}$. Obtaining the solution requires one to solve a second-order differential equation in time t:

$$\int \frac{d^2\theta}{dt^2}dt = \int \frac{\tau}{I}dt. \tag{A.0.2}$$

Integrating with respect to (w.r.t) time t yields the following:

$$\frac{d\theta}{dt} = \omega_0 + \frac{\tau}{I}t, \tag{A.0.3}$$

where the integration constant is the initial angular velocity ω_0. Another integration w.r.t time t is then applied:

$$\int \frac{d\theta}{dt}dt = \int \left(\omega_0 + \frac{\tau}{I}t\right) dt. \tag{A.0.4}$$

Giving solution:

$$\theta_f = \omega_0 t + \frac{1}{2}\frac{\tau}{I}(t)^2 + \theta_i, \tag{A.0.5}$$

where the integration constant is the initial angle θ_i. However, the final solution only needs to be applied to find the change in angle $\Delta\theta$. Thus, $\Delta\theta = \theta_f - \theta_i$ is computed using Eqn. (A.0.6), producing the final solution:

$$\Delta\theta = \omega_0\Delta t + \frac{1}{2}\frac{\tau}{I}(\Delta t)^2, \tag{A.0.6}$$

where $t \longrightarrow \Delta t$ to clarify that this is the duration for which the torque is applied.

Appendix B RISK DIAGRAM

Table 8: Risk Assessment Diagram

Risk Condition	Consquence	Prob.	Impact	Mitigation	Monitoring	Management
Team member becomes seriously ill and takes a leave of absence.	Smaller team, more work per team member.	low	high	Members should 1. watch their overall health and avoid risky behaviours, 2. monitor their schedule to finish their work quickly.	Communicate regularly about how members are doing/feeling.	1. Split the work between the remaining team members. 2. Ask for the project scope to be adjusted.
Team falls behind schedule or does not meet deadlines.	Quality of research might be reduced or may have to reduce scope.	medium	high	1. Team members should start in good time on assigned parts. 2. Include some slack time in estimated completion times for deliverables.	Team should meet and communicate their progress regularly.	1. Ask for critical aspects that must be present in the final results. 2. Ask for an extension.
Delays due to dependence on external resources (Simulink)	Time is wasted learning to design and export the models	medium	medium	1. Practice designing basic models early. 2. Test APIs/libraries (for exporting) in a sandboxed environment beforehand.	1. Show progress in designing models regularly, for feedback and assistance where necessary. 2. Run integration tests.	1. Use premade (already working) models as a last resort. 2. Find an alternative framework that allows one to create viable simulation models.
Hyperparameter Tuning	Poorly tuned hyperparameters lead to suboptimal performance.	high	medium	Conduct regular retuning of parameters.	Use validation metrics (e.g. accuracy).	Use version control to retrieve the best-performing models.
Overfitting or Underfitting	Results in inadequate model generalisation to unseen data.	medium	high	Consider using simpler methods that strike a balance.	1. Regularly assess model performance. 2. Monitor training metrics.	1. Adjust model complexity based on observations. 2. Tune hyperparameters accordingly.

Appendix C GANTT CHART

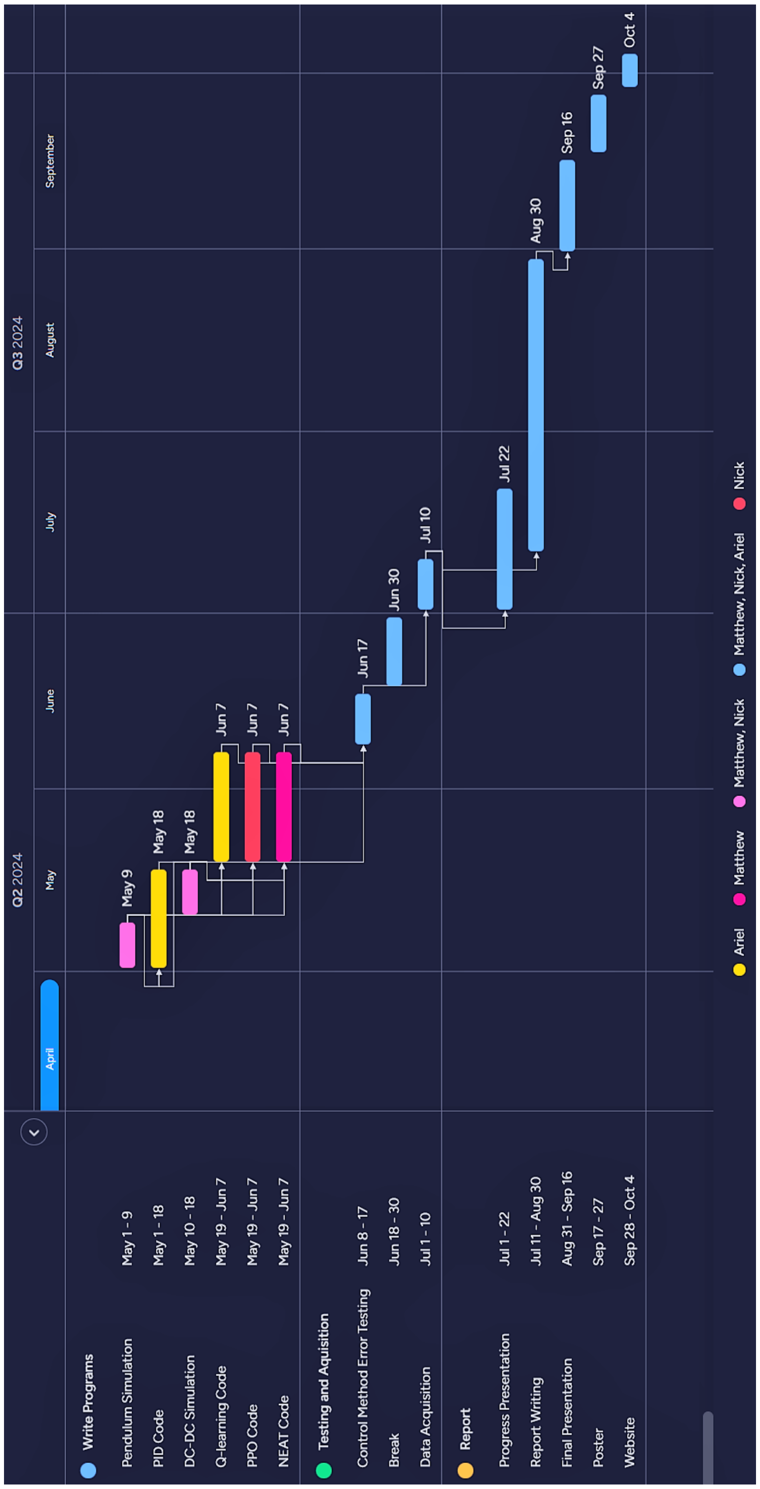


Figure 3: Gantt Chart of Project Timeline