UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS Honours Project Final Paper 2024

Title: Comparative Analysis of NEAT, PID, and Q-Learning for Effective Control of Non-Linear Systems.

Author: Matthew Fleischman

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*	0
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*	0
Total marks		**80**	**80**

Comparative Analysis of NEAT, PID, and Q-Learning for Effective Control of Non-Linear Systems.

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

Krupa Prag*
krupa.prag@uct.ac.za
University of Cape Town
South Africa

ABSTRACT

Many modern systems rely on non-linear dynamics. Data-Driven Control (DDC) combats the complexity of controlling non-linear systems. The emergence of computers, machine learning (ML), and big data has expanded the capabilities of DDC, creating numerous opportunities to explore new control methods. The existing literature on non-linear control lacks consideration of Neuroevolution of Augmenting Topologies (NEAT) as a DDC method.

This research implemented NEAT as a DDC method for two simulated non-linear systems: an inverted pendulum and a DC-DC inverting buck-boost converter (BBC). NEAT's performance was compared to that of a Proportional-Integral-Derivative (PID) controller and a Q-learning controller. Effective control was characterised as minimising stabilisation time and overshooting while providing robustness.

NEAT outperformed both PID and Q-Learning in controlling the inverted pendulum by providing shorter stabilisation times, no overshoot, and robustness to varying initial angles and observational noise. In the context of the BBC, NEAT lacked robustness and required long training times, but outperformed PID by providing shorter stabilisation times and less overshoot. NEAT proved to be an effective DDC method for the considered non-linear systems.

KEYWORDS

PID, NEAT, Q-Learning, Inverted Pendulum, Buck-Boost Converter, Data-Driven Control

1 INTRODUCTION

Many of humanity's technological advancements can be attributed to the design of effective control systems. A control system adjusts and regulates its environment's behaviour to achieve a desired state. If the controller adjusts its actions according to feedback received from the environment, it is known as a closed-loop control system [11]. Popular examples include thermostats, autonomous vehicles, and pacemakers.

1.1 History

One of the first automatic control devices, the water clock, was invented in Alexandria in the third century BC. A controlled stream of water was used to fill a reservoir at a constant rate, allowing the ancient Greeks to keep track of time. During the Industrial Revolution, engines, boilers, and furnaces needed to be controlled to operate efficiently; therefore, control devices such as pressure valves and temperature regulators were created. James Watt developed the flyball governor to regulate the speed of the steam engine, which is thought to be one of the first negative feedback devices [19]. In 1868, J.C. Maxwell formulated a mathematical model for Watt's flyball governor, effectively giving birth to control theory and model-based control [6].

1.2 Model-based and Model-free Control

Model-based control uses a mathematical model of a system to determine the necessary controlling actions. Model-based control works well for systems that rely on linear differential equations. However, when faced with non-linear systems, approximations of the system's dynamics are required. These approximations reduce the accuracy of the model and the performance of the control method [22]. The advancements of modern industries have led to the use of increasingly complex processes which rely on non-linear dynamics. This research considers two non-linear systems: an inverted pendulum and an inverting BBC, which are described in Sections 2.1 and 2.2, respectively.

Data-driven control (DDC) was developed to combat the complexity of modelling non-linear systems. DDC uses large volumes of experimentally obtained data to learn about a system's behaviour. Not only can DDC be used to enhance model-based control, but it also has led to the development of model-free control, which does not require a model of the system's dynamics [22].

1.3 Data-Driven Control Methods

The PID controller is one of the oldest and most popular examples of model-free DDC [13]. PID's popularity is owed to its simple nature and comprehensive literature. PID is highly effective for linear systems, but its performance for non-linear systems is limited. Further details on the functioning and performance of the PID controller are discussed in Section 2.3.

The emergence of computers, ML, and big data in recent decades has enhanced and broadened the capabilities of DDC. This has created numerous opportunities to explore new control methods. NEAT, the focus of this research, is a popular neuroevolution method developed by Stanley and Miikkulainen in 2002 [27]. NEAT is discussed in Section 2.4. Q-learning is one of the most popular reinforcement learning (RL) methods. RL is a type of ML. Q-learning is not the focus of this research, but it is used as a means of comparing NEAT to another ML method when investigating DDC.

1.4 Aims

The existing literature on non-linear control systems lacks consideration of NEAT as a DDC method. This research poses the following question: How does the performance of NEAT as a model-free DDC

*Supivisor

method compare to the performance of PID and Q-learning methods when controlling an inverted pendulum and a DC-DC inverting BBC?

The research question is addressed by developing and implementing both a NEAT controller and a PID controller. These controllers are applied to simulations of the specified non-linear systems. The performance of each control method is evaluated and compared to assess their effectiveness in controlling the systems. The metrics that define the quality of a controller's performance are stabilisation time, overshoot, and robustness. These metrics are further discussed in Section 3.5. This research does not design a Q-learning controller, but instead uses the results found in [18].

Additionally, the effects of observational noise on the controllers are discussed, and optimal fitness functions for the NEAT controllers are investigated.

2 BACKGROUND

This section first explains how the inverted pendulum and BBC systems function and behave. Next, the frameworks of PID and NEAT are given, and the literature on their performances as controllers for the inverted pendulum and BBC are discussed. Lastly, Q-learning is briefly described.

2.1 Inverted Pendulum

The inverted pendulum is one of the most popular benchmark applications for non-linear control. This research examines an inverted pendulum on a cart, which has the non-linear equations of motion given in Section 3.1. Figure 1 shows a diagram of this system.

The pendulum is unstable in the upright position and will fall over 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 [30]. To control the pendulum, a force is applied to the cart, accelerating the cart and applying a torque to the pendulum's arm. The aim of this control problem is to apply the correct forces to keep the pendulum in the upright position. The pendulum is regarded to be in the upright position when the angle (θ) is equal to 0. The inverted pendulum control problem is well-researched, easy to understand, and easy to simulate.

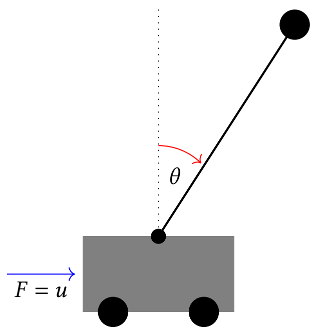

Figure 1: Inverted Pendulum on a Cart

2.2 BBC

The structure of a DC-DC inverting BBC is shown in Figure 2. A BBC 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 1. The BBC has two states: the buck state lowers the input voltage, and the boost state raises the input voltage. The relationship between the input and output voltages is non-linear due to the charging and discharging of the components within the circuit.

The BBC's output voltage, V_{out}, depends on the duty cycle, D, which is the ratio of a switch's on-time to the total switching period. The duty cycle is given to a pulse width modulator (PWM), which generates a pulse that alternates between on and off. The widths of the on and off signals are dependent on the duty cycle. This pulse is sent to the *MOSFET* switch of the BBC. When the *MOSFET* switch is on, the inductor stores energy and when the switch is off, the stored energy is transferred to the load [28].

The rate at which the PWM alternates between on and off is known as the switching frequency. An adequately high switching frequency is needed to enable a smooth output voltage. However, high switching frequencies cause an energy loss called switching loss. Switching loss reduces the efficiency of the circuit [26]. The switching frequency of the PWM must be chosen with consideration of switching loss.

Equation 1 gives an idealised relationship between V_{out}, V_{in}, and D. The BBC is in the boost state if $D > 0.5$. The BBC is in the buck state if $D < 0.5$ [9].

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

Due to the topology of the considered BBC, V_{out} is of the inverse sign to V_{in}, making this an inverting BBC. It should also be noted that when the BBC is used to raise or lower a voltage, V_{out} always starts from 0 before reaching its final voltage. When the *MOSFET* switch first closes, the inductor current starts from zero, resulting in V_{out} also starting from zero.

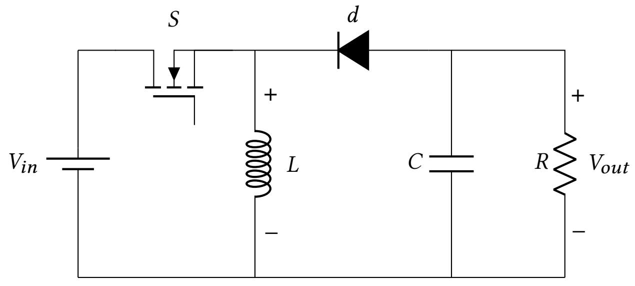

Figure 2: Inverting Buck-Boost Circuit

2.3 PID

Aspects of PID control have existed since the mid-1800s. In 1911, Elmer Sperry developed the first real PID-type controller, which was used to design automatic steering systems for the US Navy. A century later, PID controllers are found in 90-95% of all control loops [6].

Table 1: DC-DC Inverting BBC Components.

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

PID control consists of three contributions to the controlling mechanism. The proportional term adjusts the output proportionally to the error between the desired output and the measured output. A larger error means a larger corrective action. The integral term responds to the accumulated error over time, gradually eliminating any remaining deviation in the error. Finally, the derivative term reacts to the rate of change of error, helping stabilise the system by anticipating future deviation and adjusting the output accordingly [6]. Equation 2 shows the three contributions of PID control [4]. The gain coefficients for the proportional, integral, and derivative terms are represented by K_p, K_i, and K_d, respectively. The calculated error at time t is represented by $e(t)$.

$$u(t) = K_p e(t) + K_i \int_0^t e(t')dt' + K_d \frac{de(t)}{dt}. \tag{2}$$

PID controllers still show relevance in modern non-linear control systems. However, PID control is limited by uncertainties, non-linearities, and process dynamics. The development of new PID control methods has reached a stage where additional improvements are becoming increasingly marginal. [33].

2.3.1 ***PID for the Inverted Pendulum.*** A PID controller was implemented in [23] to stabilise an inverted pendulum on a moving cart. The objective of the cart was to travel a distance of 0.1m. Two PID controllers were used. A cart PID controller managed the cart position, and an angle PID controller managed the pendulum angle. The system was simulated for both perfect and noisy environments. For the ideal environment, the PID controllers stabilised the inverted pendulum quickly and smoothly. The PID controllers created minor oscillations in the noisy environment but performed well. These results testified to the robustness and effectiveness of PID control. The paper proposed using genetic algorithms to enhance the PID controllers.

A step-by-step guide on the design of PID controllers for inverted pendulum control is given in [31]. This paper demonstrates that a lot of time must be spent on understanding the behaviour of the system in order to create an effective PID controller. It should be noted, however, that the resultant PID controllers showed robustness and good dynamic performance.

2.3.2 ***PID for the BBC.*** A simulated boost converter was used to step up the input voltage to the desired voltage with an overshoot error of $2-3\%$. The PID-controlled boost converter took over 1000 seconds to produce the desired voltage, which was a very long time [1]. In another paper, a PID-controlled boost convert achieved 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.

2.4 NEAT

Neuroevolution is an ML technique that uses genetic algorithms and the principles of evolution to produce and refine generations of varying artificial neural networks (ANNs). During a generation, input data passes through an ANN, which returns an output. The quality of the output is evaluated according to a fitness function, and a fitness score is assigned to the ANN. The ANNs that achieve the highest fitness scores are selected to produce the next generation of ANNs. This process optimises the weights and nodes of the ANNs within the generation. 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 [10].

NEAT is a form of Neuroevolution. Unlike traditional Neuroevolution methods, NEAT enables the topology of the ANNs to mutate and self-optimise. This reduces the need for human intervention because the structure of the ANN does not need to be decided on in advance [21]. 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 characteristics 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" [27].

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 variables and parameters of the specific application.

2.4.1 ***NEAT for the Inverted Pendulum.*** In [25] NEAT was compared to other neuroevolution methods in its ability to control the inverted pendulum effectively. It was found that NEAT did not perform as well as the other methods. However, it was stated that NEAT's networks were not trained as well as they could have been, and further investigation into the use of different parameter values was suggested. An advantage of NEAT was that it did not require the topology of the ANNs defined before training. In [2] it was found that the performance controller stability relies on the sophistication of the fitness function used.

2.4.2 ***NEAT for the BBC.*** 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.

2.5 Q-Learning

Q-learning is one of the oldest and most popular RL methods. It was first described by Christopher J. Watkins in 1989 [32]. Q-learning uses an off-policy method, which means that two policies are used: the greedy policy that the agent evaluates and updates, and the ϵ-greedy policy used to perform actions on the system. This separation provides flexibility and allows the agent to learn from suboptimal actions and explore different strategies, mitigating the dilemma of making wrong choices. Like other RL methods, Q-learning uses a reward system to encourage actions that result in positive environmental changes. The expected cumulative reward is determined by the Q-value. Q-learning selects actions that greedily maximise the Q-value until an optimal strategy is found [17].

3 METHODOLOGY

This section first describes how the non-linear systems, the inverted pendulum and BBC, are implemented. Both of these systems are simulated using MATLAB Simulink [15]. Next, the PID and NEAT control methods are discussed. Each control method is implemented in Python, and separated into a main file and a controller file. The main file runs the simulation using the MATLAB.engine library. The controller file is what the simulation calls at each time step to receive the controlling action. Data is plotted using the Matplotlib library [14]. Lastly, the controller objectives and evaluation metrics for effective control are defined.

3.1 Inverted Pendulum Simulation

A pendulum balancing on a cart is modelled using the equations of motion derived in [24]. The angle, angular velocity, and angular acceleration of the pendulum are calculated using Equation 3.

$$\ddot{\theta} = \frac{(u\cos\theta - (M+m)g\sin\theta + ml(\cos\theta\sin\theta)\dot{\theta}^2}{ml\cos^2\theta - (M+m)l} \tag{3}$$

Table 2 shows the name and value associated with each symbol in Equation 3.

Table 2: Pendulum and Cart Parameters.

Symbol	Name	Value
θ	Pendulum Angle	Variable (rad)
u	Applied Force	Variable (N)
M	Cart Mass	$0.5kg$
m	Pendulum Mass	$0.2kg$
g	Gravitational Acceleration	$9.8ms^{-2}$
l	Pendulum Arm Length	$0.15m$

The initial angle of the pendulum is initialized at the start of the simulation. A fixed time step of 0.001 seconds is used. The variable values for the pendulum and cart are calculated at each time step. The system states, θ and $\dot{\theta}$, are sent to a MATLAB function block. The function block calls the Python controller and supplies θ and $\dot{\theta}$ as inputs. The Python controller returns a force, which is fed to the pendulum and cart simulation as parameter u.

The pendulum's angle is measured clockwise from the vertical, as shown in Figure 1. The angle of the pendulum is limited to $\pm\frac{\pi}{2}$ because if it were to go out of these bounds, the pendulum arm would be blocked by the top of the cart.

Noise is introduced to the system by adding a band-limited white noise signal to the pendulum's angle and angular velocity, before supplying them to the Python controller. This simulates observational noise for the controller. The noise added to the observation of the angle is approximately 0.05 radians, which is equal to the error bounds defined in 3.5. The noise added to the observation of the velocity is approximately $1ms^{-1}$.

3.2 BBC Simulation

An DC-DC inverting BBC is simulated using the MATLAB Simscape Electrical library. The circuit is modelled as shown in Figure 2. The parameters of the circuit are taken from [8] and are given in Table 3.

Table 3: BBC Parameters.

Symbol	Name	Value
V_{out}	Output Voltage	Variable (V)
V_{in}	Input Voltage	$48V$
L	Inductor	$220\mu H$
C	Capacitor	$8200\mu F$
R	Resistor	5.1Ω

Simscape Electrical's default MOSFET switch and diode are used. The signal supplied to the MOSFET switch is produced by a PMW generator. The PMW generator's switching frequency is set to $7.8kHz$. The fixed step size of the simulation is set to 5×10^{-6} seconds. The system state, V_{out}, is sent to the Python controller at each time step. The Python controller returns a duty cycle D ranging between 0.1 and 0.9 [28], which is fed to the PWM generator.

Observational noise is added to the system in the same way as described in Section 3.1. The noise added to the observation of V_{out} is approximately equal to the error bounds defined in Section 3.5.

3.3 PID Controller

The PID main file is responsible for setting initial system parameters, running the simulation, and collecting data. The PID controller file is responsible for calculating the PID terms using Equation 2, adding them together, and returning the controlling action. The coefficients of the terms, K_p, K_i, and K_d are determined using a MATLAB PID parameter turner. The pseudocode for the PID controller file is described in Algorithm 1.

Algorithm 1 Implementation of the PID controller file.

```
Receive input from the simulation.
Calculate the P, I, and D terms from the input.
Add the P, I, and D terms together and return the result.
```

3.4 NEAT Controller

The NEAT main file uses the NEAT-python library [20]. This library manages the creation, reproduction, and adaptation of the ANNs in each generation.

As described in Section 2.4, a fitness function is used to evaluate the fitness of each genome. In this research, NEAT is configured to maximize fitness scores within the population. Various fitness functions for both non-linear system controllers are considered and tested. The fitness function that produces the most effective ANNs is used to collect the final results for each system.

Equations 4, 5, and 6 are the fitness functions considered for the inverted pendulum NEAT controller. Equations 5 and 4 provide a maximum fitness score of 1 when θ is at 0, and a minimum fitness score of 0 when θ is at $\pm\frac{\pi}{2}$. Equation 6 was taken from OpenAi Gym [7], and additionally considers the angular velocity and the controller's previous output force. Equation 6 allows a maximum fitness score of 0, and has an unbounded minimum fitness score.

$$f(\theta) = \frac{1}{N}\sum_{k}^{N} cos(\theta_k) \tag{4}$$

$$f(\theta) = \frac{1}{N}\sum_{k}^{N}(1 - |\frac{2}{\pi}\theta_k|)^2 \tag{5}$$

$$f(\theta, \dot{\theta}, f_{prev}) = -\frac{1}{N}\sum_{k}^{N}(\theta_k^2 + 0.1\dot{\theta}_k^2 + 0.001f_{k-1}^2) \tag{6}$$

Equations 7 and 8 are the fitness functions considered for the BBC NEAT controller. Equation 7, taken from [29], provides a maximum fitness score of 1 when the measured output voltage (V_k) is equal to the desired voltage (V_g), and a minimum fitness score of approximately 0 when V_k is far from V_g. The graphical shape of Equation 7 is similar to that of Equation 4. Equation 8 provides a maximum fitness score of 10 when V_k is equal to V_g, and a minimum fitness score of 0 when $V_k \geq (V_g + 10)$ or $V_k \leq (V_g - 10)$. The graphical shape of Equation 8 is similar to that of Equation 5.

$$f(V) = (1 + \frac{1}{N}\sum_{k}^{N}(V_k - V_g)^2)^{-1} \tag{7}$$

$$f(V) = \frac{1}{N}\sum_{k}^{N}\begin{cases}\frac{(10-|V_k-V_g|)^2}{10} & \text{if } V_g - 10 \leq V_k \leq V_g + 10\\ 0 & \text{otherwise}\end{cases} \tag{8}$$

The pseudocode for the NEAT main file is described in Algorithm 2.

Algorithm 2 Implementation of the NEAT main file.

```
Create a population of genomes.
while generation < max generations and highest fitness < fitness threshold do
    for genome in population do
        Send the genome's ANN to the controller file.
        Run the simulation.
        Evaluate the genome's fitness.
    end for
    Reproduce with and mutate the genomes that score the highest fitness scores.
    Advance to the next generation.
end while
Return the genome with the highest fitness score.
```

NEAT-python configures the ANNs according to the parameters specified in a configuration file. In this research, NEAT-python's default reproduction, speciation, stagnation, and general settings are used. A higher population size and number of generations is ideal for achieving optimal ANN training [16]. However, these values increase training time. The population sizes and number of generations used to train the ANNs for the inverted pendulum and BBC are given in Section 4.2.

Simulations are run to evaluate the performance of each genome. During a simulation, the NEAT controller file is called, the simulation's state variables are supplied to the genome's ANN, and the ANN's output is returned to the simulation as the controlling action. The pseudocode for the NEAT controller file is described in Algorithm 3.

Algorithm 3 Implementation of the NEAT controller file.

```
if start of simulation then
    Get the genome's ANN from the main file.
end if
Receive input from the simulation.
Supply inputs to the ANN and return the ANN's output.
```

3.5 Controller Objectives and Evaluation Metrics

The controllers aim to control the non-linear systems effectively. Effective control is defined as minimising stabilisation time and overshooting while providing robustness. Controlling the inverted pendulum is defined as balancing the pendulum in the upright position for 4 seconds. This research considers three initial angles of the inverted pendulum: 0.2, 0.6, and 1.0 radians. Controlling the BBC is defined as converting the input voltage into a desired voltage, and maintaining this voltage for 0.3 seconds. This research considers three voltage conversions: bucking to -30V, boosting to -80V, and boosting to -110V.

Stabilization time is the duration required for the system to reach a steady state within predefined error bounds. The error bounds of the inverted pendulum are defined as ±0.05 radians. The error bounds of the BBC are defined as ±4% of the desired voltage. Average controller stabilisation times are presented with an uncertainty value. The standard error of the mean (SEM) is taken as the uncertainty and is calculated as shown in Equation 9, where σ is the standard deviation, and N is the sample size [5].

$$SEM = \frac{\sigma}{\sqrt{N}} \tag{9}$$

Overshoot is a measure of how much the controller overcorrects for the perceived error. Overshoot for the inverted pendulum is the maximum angle that the pendulum reaches after passing through the desired angle of 0 radians. Overshoot for the BBC is the maximum voltage that the BBC reaches after passing through the desired voltage.

Robustness refers to the controller's ability to maintain stability and performance despite observational noise or variations in the system's parameters. The controllers should provide consistent performance for different initial angles in the inverted pendulum system.

4 RESULTS AND DISCUSSION

This section presents and discusses the PID and NEAT results for the inverted pendulum and BBC. A side-by-side comparison of PID, NEAT, and Q-learning's performances is made. All results were obtained using the Python code in the linked Github repository: Data-Driven-Control

4.1 PID Results

The results for the PID-controlled inverted pendulum and PID-controlled BBC are discussed in this subsection.

4.1.1 ***Inverted Pendulum.*** The MATLAB PID parameter turner found the optimal PID coefficients for the inverted pendulum to be: $K_p = 44$, $K_i = 80$, and $K_d = 6$.

Figure 3 shows the angles over time of the PID-controlled inverted pendulum without noise applied. The PID controller produced a slight overshoot for an initial angle of 0.6 radians. This overshoot did not exceed the error bounds. However, the initial angle of 1.0 radians caused the PID controller to overshoot the error bounds, which resulted in a much longer stabilisation time compared to the other initial angles, as shown in Table 4. This overshoot implies that the PID controller was more sensitive to large initial angles, and is unable to stabilise them as effectively as smaller initial angles.

Noise caused a greater increase in stabilisation times for larger initial angles, as shown in Table 4. The largest noise-related increase in stabilisation time was 12.2% for the initial angle of 1.0 radians. The PID controller performed better than expected when exposed to noise.

In the previously mentioned results, the output force of the PID controller was unconstrained and often went up to $3000N$. The stabilisation time of a PID controller with its output forces constrained to be within $\pm 30N$ is given in Table 4 and is idicated by an astrix. The constrained PID controller produced a much longer stabilisation time (0.994 ± 0.000 seconds) than the unconstrained controller (0.047 ± 0.000 seconds). When any level of noise was applied to the constrained controller, the controller could not stabilize the pendulum. This result is important to consider for the real-world design of inverted pendulum controllers because actuators that supply a maximum force of $3000N$ are more expensive than actuators that supply a maximum force of $30N$.

Table 4: Stabilisation Times of PID-controlled Inverted Pendulum.

Initial Angle (rad)	Time (s), No Noise	Time (s), Noise
0.2	0.016 ± 0.000	0.016 ± 0.001
0.6	0.047 ± 0.000	0.062 ± 0.009
0.6^*	0.994 ± 0.000	*Failed*
1.0	0.777 ± 0.000	0.872 ± 0.038

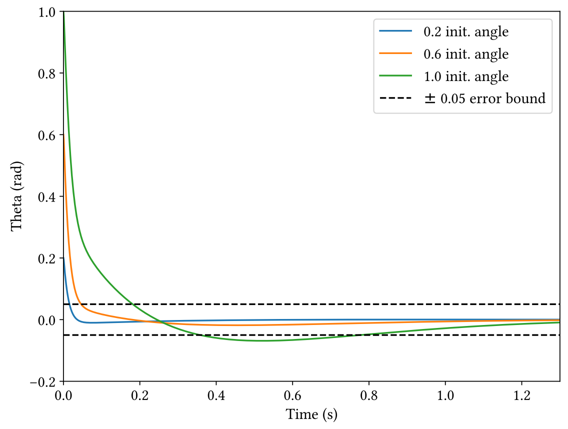

Figure 3: PID-controlled Inverted Pendulum Without Noise for Different Initial Angles.

4.1.2 ***BBC.*** The MATLAB PID parameter turner found the optimal PID coefficients for the BBC to be: $K_p = 1.3$, $K_i = 0$, and $K_d = 0.001$. The performance of the BBC PID controller was sensitive to changes in the coefficients.

Figure 4 shows the -30V buck, -80V boost, and -110V boost performed by the PID controller without noise applied. The -30V buck produced the largest overshoot and the longest stabilisation time of the three voltage conversions, which was unexpected. The overshoot produced by the PID controller was inversely proportional to the magnitude of the output voltage.

The PID controller was unable to buck or boost to any of the desired voltages when noise was applied, as shown in Table 5. Therefore, the PID controller lacks robustness in handling observational noise in the BBC.

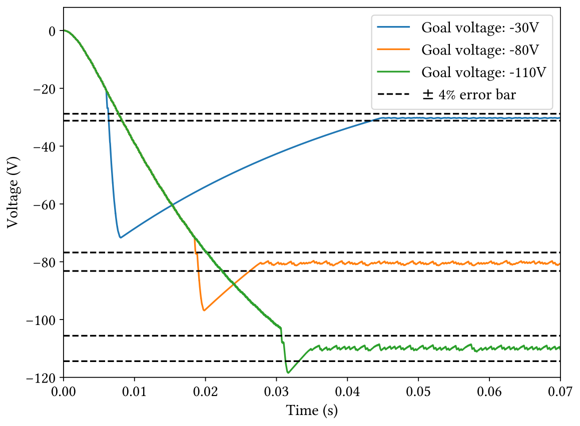

Figure 4: PID-controlled BBC Without Noise for Different Desired Voltages.

Table 5: Stabilisation Times of PID-controlled BBC.

Goal Voltage (V)	Time (s), No Noise	Time (s), Noise
−30	0.043 ± 0.000	*Failed*
−80	0.026 ± 0.000	*Failed*
−110	0.033 ± 0.000	*Failed*

4.2 NEAT Results

The results for the NEAT-controlled inverted pendulum and NEAT-controlled BBC are discussed in this subsection. For both non-linear systems, training the NEAT controllers with noise produced lower fitness scores than training without noise. For this reason, all ANNs were trained without noise, and noise was only applied to the NEAT controllers when collecting results.

4.2.1 ***Inverted pendulum.*** A population of 70 ANNs was trained over 25 generations for a training session. This took approximately 15 minutes. Of the activation functions made available to the ANNs, the highest performing ANNs primarily used the tanh activation function. Occasionally, some nodes in the hidden layers use the identity and clamped activation functions. The highest-performing ANNs used two input nodes: angle and angular velocity. NEAT always discarded the previous controller action as an input. Training with random initial angles prevented overfitting and produced more effective ANNs than training with a constant initial angle.

Five populations were trained to create 5 ANNs, each capable of controlling the inverted pendulum. The angle over time for each ANN-controlled inverted pendulum was averaged for each of the three initial angles. These results are shown in Figure 5. No overshoot occurred during stabilisation. All average stabilisation times had an uncertainy of around 15%. Table 6 compares the average stabilisation times with and without noise applied. Noise had no impact on stabilisation times when considering the associated uncertainties. Therefore, NEAT is a robust control method when handling observational noise in the inverted pendulum.

The maximum and mean fitness scores achieved by each of the 5 populations were averaged and are shown in Figure 6. Each population produced an ANN that could control the inverted pendulum within the 1st generation, which was unexpected. NEAT converged to a solution by the 11th generation, after which each population achieved a maximum score above 98% of the possible maximum score. The average population fitness varied within the range of 0.2 and 0.4. This variation was a result of NEAT discarding 80% of the genomes in the previous generation and trying new ANNs, of which some perform well, and some perform poorly.

NEAT's ability to produce effective ANNs for the inverted pendulum was sensitive to the fitness function used. Both Equations 5 and 4 allowed for effective ANNs to be produced, but Equation 5 provided faster fitness score convergence. Equation 5 was the highest-performing fitness function because it supplied an exponentially higher reward as θ approached 0. The stabilisation times of the ANNs produced by using Equation 6 were 5 times longer than the stabilisation times of the ANNs produced by using Equation 5.

4.2.2 ***BBC.*** A single ANN could not be trained to perform all three desired voltage conversions, and could only handle slight deviations in the desired voltage. This reduced the robustness of NEAT as a control method for handling different desired voltages. As a consequence, different ANNs were trained for each desired voltage conversion.

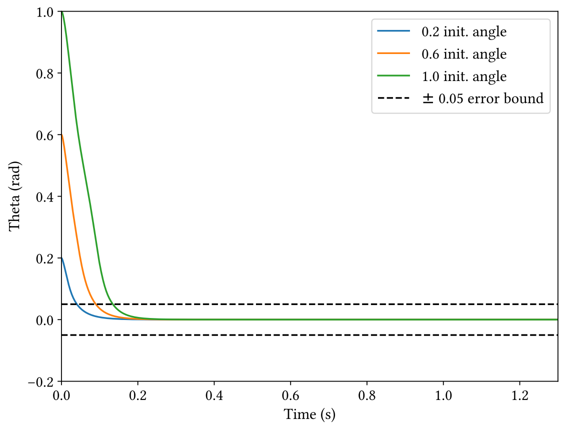

Figure 5: NEAT-controlled Inverted Pendulum Without Noise for Different Initial Angles.

Table 6: Average Stabilisation Times of NEAT-controlled Inverted Pendulum.

Initial Angle (rad)	Time (s), No Noise	Time (s), Noise
0.2	0.043 ± 0.007	0.043 ± 0.006
0.6	0.085 ± 0.012	0.083 ± 0.012
1.0	0.123 ± 0.017	0.122 ± 0.017

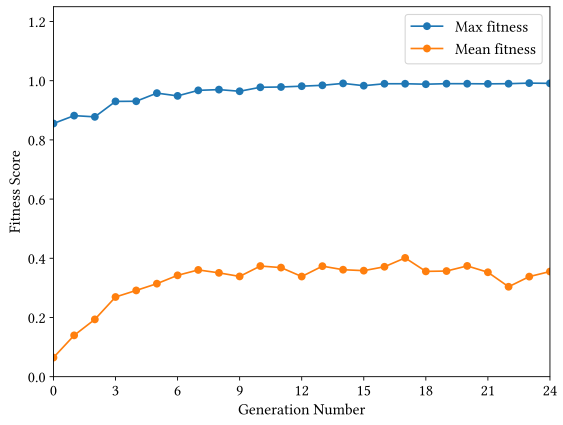

Figure 6: The Max and Average Fitness Scores of the NEAT Populations Over 25 Generations for the Inverted Pendulum.

A population of 50 genomes was trained over 15 generations for a training session. This took approximately 35 minutes. The highest-performing ANNs used the sigmoid activation function and three input nodes: voltage, error, and the derivative of the error. The error was calculated as the difference between the measured voltage and the desired voltage. These input nodes agree with those used in [12].

Three populations were trained for each of the 3 desired voltages to create a total of 9 ANNs. The voltage over time for each ANN-controlled BBC was averaged for their respective desired voltage. These results are shown in Figure 7. All ANNs stabilised the output voltage to their respective desired voltages when no noise was applied. The -80V boost produced the most overshoot, the -30V buck produced a slight overshoot, and the -110V boost produced no overshoot.

Table 6 compares the average stabilisation times with and without noise applied. All ANNs trained for the -80V boost and -110V boost were able to stabilise the output voltage with noise applied. In these cases, noise did not affect stabilisation times. However, 2 of the 3 ANNs that were trained for the -30V buck were unable to stabilise the output voltage when noise was applied. This was due to overfitting. One of the ANNs trained for the -30V buck was unaffected by noise. NEAT lacked reliability in producing ANNs that could handle noise for the -30V buck.

The maximum and mean fitness scores achieved by each of the 9 populations were averaged and are shown in Figure 8. A large increase in maximum scores occurred during the first 3 generations. NEAT often converged to a solution by the 11th generation, after which the average of the maximum scores was 77% of the possible maximum score. However, populations that were trained for the same target voltages showed different maximum scores after 15 generations. Therefore, training the populations for more generations could help achieve a higher average of maximum scores and reduce variance in convergence.

Equation 7 enabled the production of ANNs that could stabilise the -30V buck, but not the -80V or the -100V boosts. It should be noted that Equation 7 was taken from a paper that used Equation 7 to train a genetic algorithm for a buck converter [29]. Equation 8 enabled the production of ANNs for that could stabilise all desired voltages. Like Equation 5 for the inverted pendulum, Equation 8 was effective because it supplied an exponentially higher reward as the output voltage approached the desired voltage.

Table 7: Average Stabilisation Times of NEAT-controlled BBC.

Goal Voltage (V)	Time (s), No Noise	Time (s), Noise
−30	0.009 ± 0.002	*Failed*
−80	0.015 ± 0.001	0.015 ± 0.001
−110	0.017 ± 0.000	0.017 ± 0.000

4.3 Comparison of Control Methods

The performances of NEAT, PID, and Q-learning are compared to one another for the inverted pendulum and BBC in this subsection.

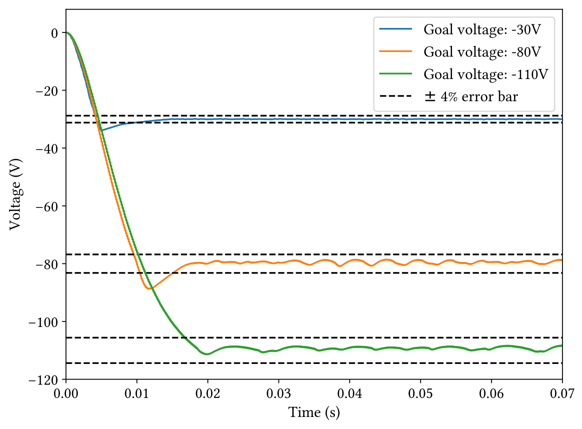

Figure 7: NEAT-controlled BBC Without Noise for Different Desired Voltages.

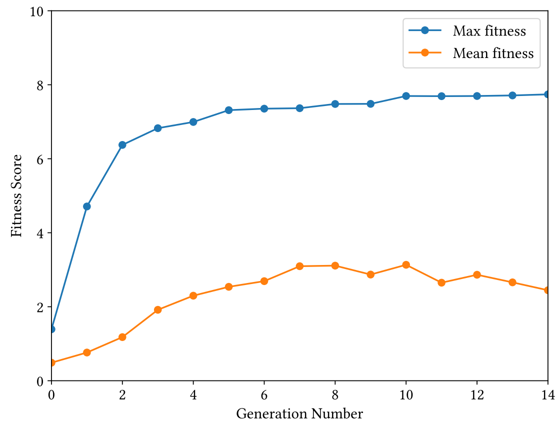

Figure 8: The Max and Average Fitness Scores of the NEAT Populations Over 15 Generations for the BBC.

A benefit of PID was that it did not require time to train its controller, unlike NEAT and Q-learning. The training time of Q-learning should not be compared to the training time of NEAT because a minimum required training time was not investigated for either of these methods.

4.3.1 ***Inverted Pendulum.*** Figure 9 compares the max, min, and mean values achieved by three different control methods without noise applied. PID produced the shortest minimum stabilisation time, and NEAT produced the shortest mean, and max stabilisation times. The largest overshoot was produced by Q-learning, followed by PID, and lastly by NEAT. Figure 9 shows that NEAT produced the lowest stabilisation time uncertainty. Therefore, NEAT was more robust to initial angle variations than the other methods. PID produced the highest uncertainty of the methods, making it the least robust to initial angle variations. NEAT was less affected by noise than PID when comparing the results in Table 6 to those in Table 4. Therefore, NEAT was more robust to noise when compared to the PID controller.

Q-learning could only produce discrete output forces, which reduced its performance. PID only had the pendulum's angle as an input to its controller, whereas NEAT and Q-learning also had the angular velocity as an input. The use of fewer inputs was a limiting factor in PID's performance.

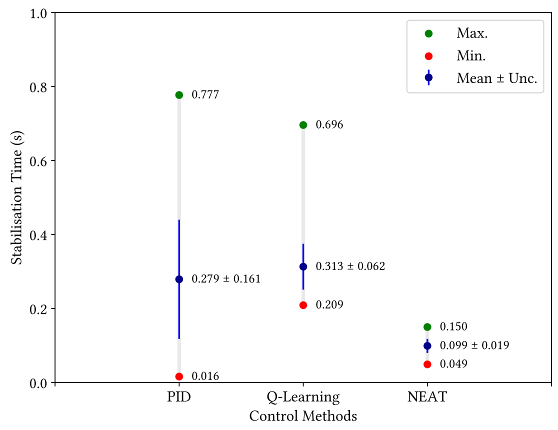

Figure 9: The Max, Min, and Mean Stabilisation Times for the PID, Q-Learning, and NEAT-controlled Inverted Pendulums Without Noise.

4.3.2 ***BBC.*** Q-learning was unable to stabilise the output voltage of the BBC to any of the 3 desired voltages. The Q-learning results discussed in the section were obtained from a simulation of a buck converter, not a BBC. As a consequence, the NEAT and PID results could only be compared to a -30V buck performed by Q-learning.

Figure 10 compares the voltages over time produced by the 3 control methods for the -30V buck without noise applied. Q-learning produced the shortest stabilisation time of 0.004 ± 0.001 seconds, followed by NEAT with a stabilisation time of 0.009 ± 0.002 seconds, and lastly by PID with a stabilisation time of 0.043 ± 0.000 seconds. The difference in stabilisation time between PID and NEAT was much larger than between NEAT and Q-learning. PID produced the largest overshoot and NEAT produced the smallest overshoot. Among the three methods, Q-learning was the most robust in handling noise. PID could not manage any level of noise. NEAT could produce ANNs that effectively handled noise, but it could not do so reliably.

The robustness of Q-learning to different voltage conversions cannot be considered, as it was unable to control the BBC. A single PID controller was able to perform the 3 desired voltage conversions. In contrast, NEAT required the production of different ANNs to perform the 3 desired voltage conversions. Therefore, PID was more robust than NEAT in the context of handling various voltage conversions.

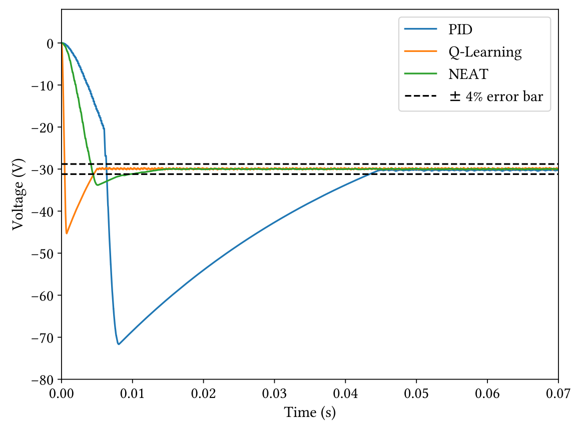

Figure 10: PID, Q-Learning, and NEAT-controlled BBCs Without Noise.

5 CONCLUSIONS

This research investigated the effectiveness of NEAT as a model-free DDC method for the inverted pendulum and DC-DC inverting BBC. This was done by comparing a NEAT controller's performance to that of PID and Q-learning controllers. Effective control was defined as minimising stabilisation time and overshooting while providing robustness.

NEAT outperformed both PID and Q-Learning in controlling the inverted pendulum by providing shorter stabilisation times, no overshoot, and robustness to varying initial angles and observational noise. NEAT outperformed PID in controlling the BBC by providing shorter stabilisation times and less overshoot. Q-learning was unable to control the BBC. As a consequence, Q-learning's performance in controlling a buck converter was considered. NEAT produced less overshoot than Q-learning for a -30V buck, but Q-learning produced a shorter stabilisation time.

In the context of BBC control, NEAT lacked robustness and required long training times. The BBC simulation had large computer memory requirements, which limited the number of generations that NEAT was able to train on. Future research should explore faster methods of training NEAT and extend its training across more generations. Additionally, NEAT's potential to produce a single ANN capable of performing all voltage conversions should be investigated, as this would enhance NEAT's robustness for BBC control.

PID's performance for the BBC was found to be sensitive to the chosen parameters. NEAT's performance for both non-linear systems was found to be sensitive to the fitness functions used. The highest-performing fitness functions provided an exponentially high reward for minimising the system error.

REFERENCES

[1] Mirza F. Adnan, Mohammad A. Oninda, Mirza M. Nishat, and Nafiul Islam. 2017. D. *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] Mohini P Barde and Prajakt J Barde. 2012. What to use to express the variability of data: Standard deviation or standard error of mean? *Perspectives in clinical research* 3, 3 (2012), 113–116.

[6] Rakesh P Borase, DK Maghade, SY Sondkar, and SN Pawar. 2021. A review of PID control, tuning methods and applications. *International Journal of Dynamics and Control* 9 (2021), 818–827.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[8] Farah Shabila Dinniyah, Wahidin Wahab, and Muhammad Alif. 2017. Simulation of buck-boost converter for solar panels using PID controller. *Energy Procedia* 115 (2017), 102–113.

[9] 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

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

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

[12] Mojtaba Hajihosseini, Milad Andalibi, Meysam Gheisarnejad, Hamed Farsizadeh, and Mohammad-Hassan Khooban. 2020. DC/DC power converter control-based deep machine learning techniques: Real-time implementation. *IEEE Transactions on Power Electronics* 35, 10 (2020), 9971–9977.

[13] 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.

[14] J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55

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

[16] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. 2017. Population based training of neural networks. *arXiv preprint arXiv:1711.09846* (2017).

[17] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, and Jong Wook Kim. 2019. Q-Learning Algorithms: A Comprehensive Classification and Applications. *IEEE Access* 7 (2019), 133653–133667. https://doi.org/10.1109/ACCESS.2019.2941229

[18] Ariel J. Levy. 2024. Comparing Data-Driven Control Methods for Non-Linear Binary Systems: Q-Learning. *UCT* (2024).

[19] Frank L Lewis. 1992. *Applied optimal control and estimation*. Prentice Hall PTR.

[20] 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/

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

[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.

[23] Lal Bahadur Prasad, Barjeev Tyagi, and Hari Om Gupta. 2014. Optimal control of nonlinear inverted pendulum system using PID controller and LQR: performance analysis without and with disturbance input. *International Journal of Automation and Computing* 11 (2014), 661–670.

[24] Lal Bahadur Prasad, Barjeev Tyagi, and Hari Om Gupta. 2014. Optimal control of nonlinear inverted pendulum system using PID controller and LQR: performance analysis without and with disturbance input. *International Journal of Automation and Computing* 11 (2014), 661–670.

[25] 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.

[26] On Semiconductor. [n. d.]. Effects of high switching frequency on buck regulators. *On Semiconductor. Available at http://www. onsemi. com/pub_link/Collateral/TND388-D. PDF* ([n. d.]).

[27] 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

[28] Frank De Stasi. 2022. Working With Inverting Buck-Boost Converters. *Texas Instruments* (2022).

[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] 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

[31] Jia-Jun Wang. 2011. Simulation studies of inverted pendulum based on PID controllers. *Simulation Modelling Practice and Theory* 19, 1 (2011), 440–449.

[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] K.J. Åström and T. Hägglund. 2001. The future of PID control. *Control Engineering Practice* 9, 11 (2001), 1163–1175. https://doi.org/10.1016/S0967-0661(01)00062-4 PID Control.