Machine learning-based control of living cells

Evaluation of Model-Free Reinforcement Learning for the in-silico control of a Genetic Toggle Switch

By

Pawan Gurung

Department of Engineering Mathematics
University of Bristol

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of MENG in engineering mathematics in the Faculty of Engineering.

August 18, 2023

Word count: ten thousand and four

Abstract

In this project, an evaluation of the performance of the Proximal Policy Optimisation (PPO) agent was conducted to demonstrate the feasibility of Model-Free Reinforcement Learning (MFRL) in Cybergenetics. The PPO was tested on a task requiring the regulation of a deterministic genetic toggle switch in an unstable state for an extended duration. The performance of PPO was evaluated against an untrained PPO, a Proportional-Integral-Derivative (PID) and relay controller. Integral Squared Error (ISE) was employed as a performance metric to assess the control capabilities of the controllers. The PID and relay controllers outperformed the trained PPO achieving a relatively lower mean ISE values. However, the observed improvement in performance from the untrained PPO to the trained PPO indicated that the agent had indeed learned a policy and could effectively control the GTS in its unstable state. Suggesting MFRL could be a viable option for future Cybergenetics applications but sub-optimal performance of the PPO agent suggests improvements can be made. Next steps include the further improvements of hyperparameter tuning, training process and the environment architecture which will allow for a more robust evaluation.

Dedication and acknowledgements

First and foremost, I would like to express my deepest appreciation to my supervisors, Dr. David Barton and Dr. Lucia Marucci, for their continuous guidance, support, and encouragement throughout the course of this project. Their expertise and vast knowledge and constructive feedback has been significant in helping me overcome the challenges I faced during my research. I am extremely grateful for their guidance and commitment to my development. I would also like to extend my appreciation to Andrew Shannon, whose assistance have been invaluable. His willingness to lend a helping hand and providing useful and unique insights allowed me to overcome many obstacles in this project. I am truly grateful for all the hours he has spent on me. I would also like to thank my family and friends for their continuous support. Lastly, I want to thank my girlfriend, Catalina Dica, whose support has been invaluable.

Author's declaration

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Synthetic biology aims to construct and engineer complex biological circuits and gene regulation networks (GRNs) to control cellular behaviour [3] [31]. However, the stochasticity and non-linearity of gene circuits and GRNs cause difficulty in predicting and controlling biological functions with precision and reliability [3]. A possible solution is Cybergenetics, an upcoming field of study merging the tools of synthetic biology and control theory [21] that aims to control complex stochastic non-linear systems. Reinforcement Learning (RL) offers alternative approaches to control; Model-Free Reinforcement Learning (MFRL) and Model-Based Reinforcement Learning (MBRL), each with unique strengths and limitations. MFRL algorithms learn the control policy from interacting with the system; without requiring an explicit model of the system's dynamics [34]. Conversely, MBRL involves learning a model of the system's dynamics, which is used to optimise the control policy [34]. Due to its adaptability to changing system dynamics without requiring model retaining, MFRL offers a more flexible and robust approach to controlling complex biological systems [34]. This project aims to evaluate MFRL and considers it as a potential tool for Cybergenetics control tasks.

1.2 Project Aims

1. This project aims to evaluate the feasibility of Model-Free Reinforcement Learning (MFRL) as a potential solution for controlling Cybergenetics control tasks.

1.3 Project Objectives

1. Identify a suitable model of a GRN that is currently relevant to the field, for the purpose of conducting control experiments.
2. Develop a custom gym environment of the chosen GRN, to train an agent to control the system.
3. Train the agent via an MFRL algorithm within the environment.
4. Implement a classical control techniques such as PID and relay to evaluate and contrast the performance of MFRL for control.
5. Assess the performance attained from the MFRL algorithm and compare it to PID and relay to determine the viability of MFRL as a useful tool for Cybergenetics.

1.4 Types of Control

One of the goals of Cybergenetics is to achieve accurate manipulation of GRNs to control cellular behaviour at various scales. This goal is achieved through various control paradigms that interact with the GRNs. An illustration of the paradigms is shown in figure 1.1.
Figure 1.1 illustrates external control, also referred to as in-silico, of GRNs, whereby the

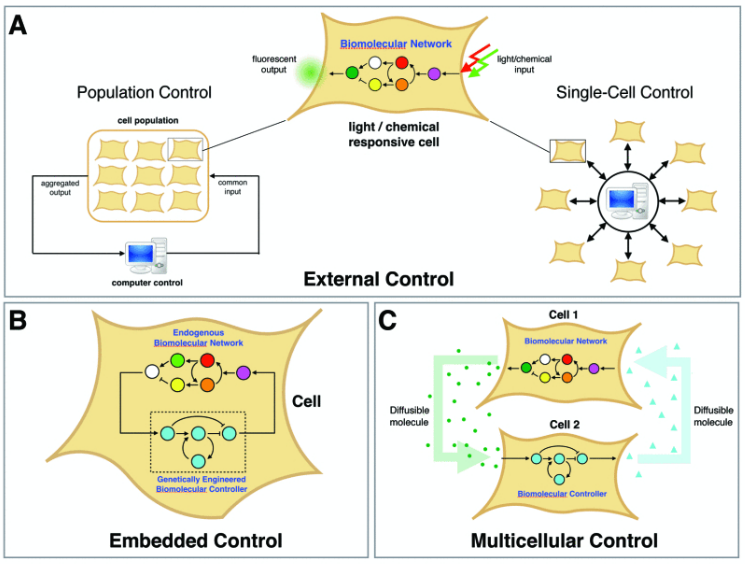


Figure 1.1: Different control methods of controlling GRNs [21]

GRN inside an individual or a population of cells are manipulated by external means such as microfluidics platforms that are controlled using more conventional techniques like PID control or RL. In-silico control provides a mechanism to manipulate GRNs within an environment without directly interfering with the cell's internal machinery, enabling precise control of cellular behaviour [21]. Embedded Controller, also reffered to as in-vivo, is a control paradigm that manipulates GRN's inside the living cells, meaning the controller is embedded in the cell and "are themselves bio-molecular species that are synthesised by the cell's machinery".This is illustrated in figure 1.1 [21]. In-vivo control requires direct intervention within the cell's internal machinery whereas in contrast to in-silico control, the GRN's can be manipulated through external means [21]. Figure 1.1C shows the multi-cellular control of GRNs; its characteristics are similar to that of the embedded (in-vivo) control - the controller species consist of bio-molecular species. However, the network and controller are contained in different cells and they interact with each other through diffusible molecules [21]. The external control approach is the primary focus of this project, particularly population control through microfluidic interaction, in which the average fluorescence of the cell population is monitored, and various mediums are used to regulate gene expression through interaction with the GRN.

1.5 Benefits of Cybergenetics

The potential of Cybergenetic is prodigious; its implications are extensive, having a significant impact on numerous industries and fields of science. For example:

- Regulation Processes: Regulation is essential in biology; cells utilise complex mechanisms to effectively regulate for survival and cellular homeostasis. Cybergenetics allows the testing of regulation hypotheses and proposes new ones, enabling a deeper understanding of biological regulation. For example, failure of regulatory mechanisms in the human body cause many human diseases. A genetically engineered control system can restore the regulatory mechanism by sensing the dysregulated physiological variables and responding to them by producing appropriate biological effectors so the control system can reinstate the variables in a controlled range. [22] [21]
- Medical Therapy: Cybergenetics offer a promising direction in bio-therapeutics involving genetically engineered 'smart' human cells to sense and evaluate a disease's state and then perform actions in response to the state in a closed-loop fashion, automatically managing the disease. An example is the treatment of type 1 diabetes in a diabetic mouse. The 'smart' cells detect the glucose level and activate insulin expression in response to the proportion of the glucose. [22]
- Biotechnology: Useful chemicals such as biofuels, vitamins, and antibiotics are a byproduct of modern biotechnology- genetically modifying or rewiring cells to synthesise these beneficial chemicals. Optimal production and yield of these products require regulation, a restrictive control of specific enzymes and metabolite levels in the metabolic pathway.[22]

Rapid advancements in cybergenetics may also have severe unfavourable effects. For instance, synthetic biology already makes it possible to produce deadly, highly contagious diseases using equipment that is readily available. The dangers posed by corrupt individuals abusing synthetic biological technologies will only grow with the addition of the capability to directly regulate genetic regulatory networks. Therefore, this work also increases the risk. [29]

Chapter 2

Background

2.1 Synthetic Biology

Throughout history, humans have developed an innate desire to control various systems. From the invention of the water clock [17] to modern-day classical and advanced control techniques, the pursuit of optimal performance, stability and efficiency has been the primary objective in developing and implementing control systems. One domain that demonstrates this pursuit is synthetic biology, an active area of research that seeks to construct and modify complex biological systems with the application of engineering to investigate biological phenomena and for various implementations [3]. The aim of synthetic biology is to produce, assemble and integrate the bio-molecular systems to control cellular behaviour to provide a greater understanding of natural cellular processes and for various uses.

2.1.1 Gene Regulatory Networks

GRNs are at the core of Synthetic Biology, defined as ”a network that has been inferred from gene expression data” by F.Emmert-Streib et al. [9] GRNs are a group of molecular species and the interactions between them that jointly regulate the gene-product abundance [20]. To model the networks, the law of mass action and Hill reaction kinetics can be applied to create systems of Ordinary Differential Equations that describe the rate of change of the concentration of the network's genes [20].

Biological systems, like GRNs, are highly complex. The lack of complete knowledge regarding the cellular environment causes difficulty in predicting and controlling the functions of biological processes with precision and reliability [3]. Accurate, robust control of the biological system is needed to gain a deeper understanding and will progress the field of Synthetic biology.

2.1.2 Genetic Toggle Switch

A genetic toggle switch (GTS) is a bi-stable GRN and is widely employed in complicated synthetic circuitry when bi-stability, memory, or binary signal processing is desired, as it is a fundamental topology in core natural gene regulation networks and one of the foundational results of synthetic biology. Therefore, the GTS was chosen as the subject of the control tests conducted in this project. [27] [12]

The first development of a synthetic toggle switch was proposed by Gardner et al., their construction of the toggle switch set a benchmark and foundation for other models of the toggle switch ([23], [24]), [6], [19]. The network is composed of two mutually repressing genes: *LacI* and $TetR$ inside *E.coli*. They stated that the behaviour of the toggle switch and the conditions for bi-stability are represented by the following dimensionless model represented by equations 2.1 and 2.2.

$$\frac{du}{dt} = \frac{\alpha_1}{1 + v_\beta} - u \tag{2.1}$$

$$\frac{dv}{dt} = \frac{\alpha_2}{1 + u_\gamma} - v \tag{2.2}$$

Where u and v represent the concentration of repressor one and repressor two, respectively, α_1 and α_2 represent the effective rate of synthesis of repressor one and repressor two, respectively, β represents the cooperativity of repression of promotor two, and γ represents the cooperativity of repression on promoter 1. The bi-stability behaviour is illustrated in figure 2.1; the nullclines ($du/dt = 0$ and $dv/dt = 0$) intersect at three points, two being in stable states and one being in an unstable state. The bi-stability behaviour, shown in figure 2.1, is possible when α_1 and α_2 are of equal strength (balanced) and when β and $\gamma > 1$. If the effective rate of synthesis of both repressors is not balanced, the nullclines will only produce one single stable, steady state as they will only intersect once, and this is shown in figure 2.2.

Cells remain in the two stable states even when exposed to stochastic perturbations. However, with the addition of diffusible inducer molecules, isopropyl-β-D-thiogalactopyranoside ($IPTG$) and anhydrotetracycline (aTc), the repression of LacI and TetR can be tuned, respectively. Hence it is possible to switch the circuit from one stable equilibrium state to the other. The unstable state, located on the separatrix - a curve separating the two basins of attraction, is formed when both genes are expressed at low comparable levels. Consequently, a toggle with an initial condition that lies above the separatrix will shift to state one, and a toggle with an initial condition that lies below the separatrix will shift to state two. [27] [12]

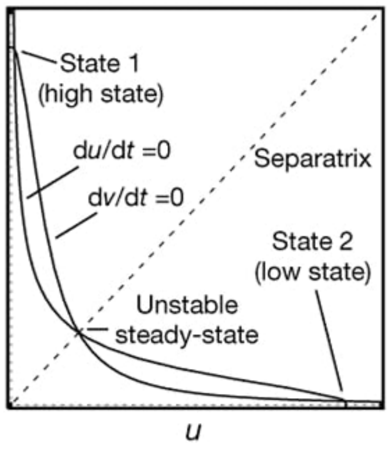


Figure 2.1: Bi-stable toggle system with balanced promoter strength

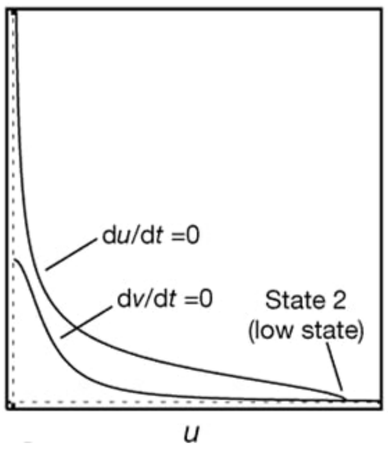


Figure 2.2: Mono-stable toggle system with imbalanced promoter strengths

2.2 Modelling the Genetic Toggle Switch

To understand the dynamics of the toggle switch and analyse the results of the control experiments, an in-silico model of the toggle switch must be developed. In this project, an improved model of Gardner's original Toggle Switch, proposed by J.Lugagne et al., is used. The authors developed two models of the Toggle Switch based on pseudo-reactions representing transcription, translation and degradation/dilution due to growth, represented by equations 2.3, 2.4 and 2.5 respectively. Depending on the context of the problem, either a deterministic or a stochastic model of the pseudo reactions can be employed. [27][12]

$$\emptyset^{f_L^m(TetR,aTc)} \rightarrow MRNA_L \qquad \emptyset^{f_T^m(LacI,IPTG)} \rightarrow MRNA_T \tag{2.3}$$

$$mRNA_L \overset{k_L^P}{\rightarrow} mRNA_L + LacI \qquad mRNA_T \overset{k_T^P}{\rightarrow} mRNA_T + TetR \tag{2.4}$$

$$mRNA_L \overset{g_L^m}{\rightarrow} \emptyset \qquad mRNA_T \overset{g_T^m}{\rightarrow} \emptyset \qquad LacI \overset{g_L^P}{\rightarrow} \emptyset \qquad TetR \overset{g_T^P}{\rightarrow} \emptyset \tag{2.5}$$

2.2.1 Deterministic Model

J.Lugagne developed a deterministic,a hill-type model of the GTS. To reduce the dimensionality of the model and the number of parameters involved, a hill-function was used to model the binding-unbinding event. With that assumption and using differential rate laws, the Ordinary Differential Equation (ODE) model is narrowed down to a 4-dimensional set of coupled ODES:

$$\frac{dmRNA_{TetR}}{dt} = k_T^{m0} + \frac{k_T^m}{1 + (\frac{LacI}{\theta_{LacI}} \times \frac{1}{1+(\frac{IPTG}{\theta_{IPTG}})^{\eta_{IPTG}}})^{\eta_{LacI}}} - g_T^m \times mRNA_{TetR} \tag{2.6}$$

$$\frac{dmRNA_{LacI}}{dt} = k_L^{m0} + \frac{k_L^m}{1 + (\frac{TetR}{\theta_{TetR}} \times \frac{1}{1+(\frac{aTC}{\theta_{aTC}})^{\eta_{aTC}}})^{\eta_{TetR}}} - g_L^m \times mRNA_{LacI} \tag{2.7}$$

$$\frac{dLacI}{dt} = k_L^P \times mRNA_{LacI} - g_L^P \times LacI \tag{2.8}$$

$$\frac{dTetR}{dt} = k_T^P \times mRNA_{TetR} - g_T^P \times TetR \tag{2.9}$$

Equations 2.6 and 2.7 represent the transcription rate, and equations 2.8 and 2.9 represent the translation rate. In the experiment, aTc and $IPTG$ are the control variables, so only the rest of the model parameters are fixed values. The parameters $k_{L/T}^{m0}$, $k_{L/T}^{m}$, $k_{L/T}^{P}$, $g_{L/T}^{m}$ and $g_{L/T}^{P}$ represent the leakage transcription, transcription, translation, mRNA degradation and protein degradation respectively. The deterministic model's parameter values are presented in table A.1 located in the appendix A.

Lugagne conducted experiments in which the inducer molecules were applied to the Toggle Switch using an external controller, as these molecules are not naturally produced within the system. The purpose of applying the inducers in this manner was to achieve a high degree of accuracy in the calibration data, and this led to their first inducer exchange model. In this model, only the relatively slow exchanges of $IPTG$ in and out of the cell were considered. [27]

$$aTc = u_{aTc} \tag{2.10}$$

$$\frac{dIPTG}{dt} = k_{IPTG}\ (u_{IPTG} - IPTG) \tag{2.11}$$

Where u_{aTc} and u_{IPTG} denote the concentration of the external inducers. Although the first inducer model obtained an excellent fit to the calibration data, a more intricate inducer exchange model was required to ensure consistency within the control experiments. This developed model incorporated an asymmetrical exchange of both aTc and $IPTG$ in and out of the cell, which was not accounted for in the first model. [27]

$$\frac{daTc}{dt} = \begin{cases} k_{aTc}^{in} * (u_{aTc} - aTc), if & u_{aTc} > aTc \\ k_{aTc}^{out} * (u_{aTc} - aTc), if & u_{aTc} \leq aTc \end{cases} \tag{2.12}$$

$$\frac{dIPTG}{dt} = \begin{cases} k_{IPTG}^{in} * (u_{IPTG} - IPTG), if & u_{IPTG} > IPTG \\ k_{IPTG}^{out} * (u_{IPTG} - IPTG), if & u_{IPTG} \leq IPTG \end{cases} \tag{2.13}$$

Lugagne stated that in the experiments where aTc and $IPTG$ molecules were changed infrequently, the asymmetrical exchange of the inducers was not observed, potentially due to the strong binding of effectors to repressors or adsorption effects. As a result, the dynamics of the system were not entirely understood until the development of the second inducer exchange model, which accounted for the asymmetrical exchange of the inducer molecules. The second inducer exchange model required manual modifications of certain model parameters. Despite these alterations, the behaviour of the system was not affected severely, and the location of equilibrium points in the state space was only marginally changed. [27]

2.3 Control Theory

Control Theory is the study and analysis of dynamic systems and methodologies to construct controllers [8]. The goal is to develop an algorithm that controls the system's input to drive the system to the desired state while ensuring a level of control to acquire a degree of optimality. There are two main approaches to control: an open-looped system and a closed-loop system. In an open-loop system, the output is not monitored or fed back to the system, and the control input follows a pre-determined rule or policy. Whereas for a closed loop system, the output is monitored, and the control inputs are adjusted to achieve the desired output. Each approach is practical in specific contexts. However, in general, open-loop methods struggle to handle external noise and disturbances while also being inaccurate due to no information being fed back to the system. The closed-loop system resolves these issues, providing superior performance.

Accurate control of a dynamical system is a complex issue for many reasons, and the choice of the control algorithm is one of the crucial factors. The performance of a controller is highly dependent on the control algorithm and the specific context for its use; this enhances the importance of choosing the correct and suitable method, particularly as there is a broad range of control algorithms. The aim of this project is to maintain a single cell around an unstable state while minimising disturbance; therefore, positive loop (feed-forward loop) methods can be disregarded.

2.4 Reinforcement Learning

Reinforcement learning (RL) is the learning framework for sequential decision-making; it relies on the assumption that an agent's environment is modelled as a Markov Decision Process (MDP) within which sequential decision optimisation is normalised [35]. In general terms, an agent iteratively interacts with the environment through actions and observes the resulting state and the reward associated with it. The agent learns either a value function or a policy function, or even both, with the purpose of obtaining a control policy. [35]

RL algorithms can be divided into two main categories: Model-Based Reinforcement Learning and Model-Free Reinforcement Learning. In MBRL algorithms, the agent utilises an exact or approximate model of the system dynamics to plan actions and maximise rewards [34]. MFRL algorithms operate in a different manner; the agent seeks to learn the consequences of their actions through experience via algorithms such as Policy gradient and Q-Learning. This is so adjustments to the policy can be made to optimise and maximise reward. [34]

2.4.1 Problems of RL learning methods

The primary objective in RL is to ensure the agent is trained optimally so it interacts with the environment to maximise the reward signal. This objective is not so necessarily easy as the challenge lies in designing an optimal policy that maximises the expected reward. There has been numerous learning methods to achieve this optimal policy, the two main methods notably being offline and on-policy learning.

Offline methods are a widely adopted approach to RL problems due to their data and sample efficiency. The method does not allow for any interaction with the environment, meaning the training or learning process is done from a fixed dataset of previously gathered data. There are pros and cons to this approach, the main pro being that if the interaction with the environment is costly or dangerous then offline method provide a safe and cost-efficient approach. However, the significant disadvantage to offline learning is the distributional shift between the training data and policy. More specifically, the training data is not collected from the current policy that is being learned on, meaning there is a shift in distribution of the data. This then leads to instability and slow convergence during training and results in a sub-optimal policy. [26] [11] [25]

One way to mitigate the issue of distributional shift that offline learning method faces is to update the policy using the data generated by the current policy during the training process. On-policy learning methods adopts this approach, this allows for the training data and the current policy to be aligned, which provides stability during training and policy updates. Furthermore, on-policy methods provides robustness, meaning it is capable to adapt to changes in the environment and learn more optimal policies as the agent is continuously collecting and utilising the new data gathered from the new policies during training. However, there is a major drawback to on-policy method, it is highly data and sample-inefficient. This is primarily due to the discard of the sample data used for training after each policy update. [26] [13]

2.4.2 Overview of RL Algorithms

In recent years, literature and previous studies have mitigated these drawbacks by proposing RL algorithms. In 2015, J. Schulman et al. proposed Trust Region Policy Optimization (TRPO), a novel policy gradient method to optimise control policies with guaranteed monotonic improvement [32]. Although the algorithm proved successful, in 2017, J. Schulman et al. proposed a new classification of policy gradient methods for RL, intending to resolve the issues of TRPO: not being compatible with architectures that include noise or parameter sharing [33]. The authors proposed the Proximal Policy Optimization (PPO) algorithm to remedy these drawbacks. Their results show that the PPO is capable of attaining the stability and reliability of TRPO and is much simpler to implement while also outperforming other online policy gradient methods (AC2 and ACER). To address the problems of hyperparameter sensitivity and weak convergence properties, Harnooja developed Soft Actor-Critic (SAC), an off-policy actor-critic deep RL algorithm based on the maximum entropy reinforcement learning framework [14]. In 2018, K. Chua et al. proposed probabilistic ensembles with trajectory sampling (PETS), a novel MBRL algorithm that matches the asymptotic performance of MFRL algorithms while still retaining the sample efficiency of MBRL algorithms [7]. Their results show that PETS is capable of attaining model-free asymptotic performance that's on par with MFRL algorithms, such as PPO and SAC, while retaining high sample efficiency. M. Janner et al. developed Model-Based Policy Optimisation (MBPO) in 2019 in an effort to lessen the impact of model inaccuracy, which negatively impacts model-based RL performance [18]. Similar to PETS, it has a much higher sample efficiency and comparable asymptotic performance to leading-edge model-free algorithms.

2.5 Proximal Policy Optimisation

PPO combines several characteristics, namely being a model-free algorithm in RL, a policy gradient method, and an on-policy learning method. The main goal of PPO and the basis it was built upon was to improve the stability and convergence of policy updates and to improve upon TRPO [33]. PPO achieves this stability and convergence of policy updates by using a clipped surrogate objective function that penalises the agent if the policy update is too large which then results in more stable updates. TRPO, like PPO, addresses the issues of instability of policy updates by ensuring the updates are within a trust region, controlling the size of the policy update, guaranteeing stability. However, TRPO suffers from several limitations. For example, the high computational cost due to the need for the second-order optimisation. Furthermore, the tuning of the hyperparameters and the implementation of the algorithm is difficult. PPO addresses these issues while still attaining the reliable performance of TRPO. [33] [32]

2.5.1 Clipped Surrogate Objective Function

The equation 2.14 represents the main objective function used in PPO. Its primary goal is to ensure that the policy updates remain close to the current policy during optimisation.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{2.14}$$

The objective function achieves this goal through the use of the min and clip functions. The *min* function selects the smaller value between the two values; by selecting the lower bound, PPO ensures a controlled policy update mitigating the risk of instability throughout the optimisation process. The *clip* function is the clipped probability ratio multiplied by the advantage function; it constrains the value of $r_t(\theta)$, where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}$ denotes the probability ratio between the updated policy and the old policy, within the range of [1 - ϵ, 1 + ϵ], where ϵ is a hyper-parameter. This guarantees that the policy updates will be appropriately scaled and remain close to the current policy. The term $r_t(\theta)\hat{A}_t$ denotes the default objective for normal policy gradients, which pushes the policy towards actions that yield a high positive advantage over the baseline. [33]
The advantage function ($\hat{A}_t$) is defined as the difference between the baseline estimate and the discounted sum of rewards. In summary, the advantage function indicates and evaluates whether the action taken by the agent was better or worse than expected. [33]

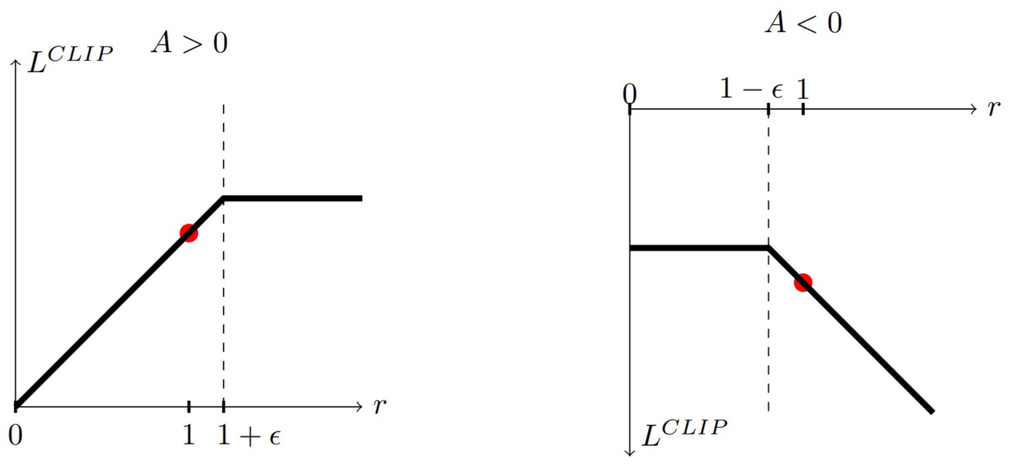


Figure 2.3: Plots depicting the positive and negative of the advantage function, while also illustrating a single time step of the surrogate function as a function of probability ratio r.

2.5.2 Loss Function

$$L^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \tag{2.15}$$

The equation 2.15 represents the loss function used in PPO. The loss function is a weighted sum of the Clipped Surrogate Objective Function, Value Function and Entropy Bonus. Each component serves a specific purpose in the training process. The Clipped Surrogate Objective Function, as discussed before, aims to optimise the policy by encouraging the agent to take actions that yield higher rewards while also preventing large policy updates. The Value Function estimates the state values; it aims to minimise the difference between the estimated state value and the actual returns to allow the value function to provide optimal guidance to the policy. The Entropy Bonus encourages exploration by adding an entropy component to the loss function, meaning it allows the agent to explore the environment. Finally, c_1 and c_2, represent the hyperparameters; they are positive constants that weigh the contributions of the Value Function and the Entropy Bonus in the loss function. [33]

PPO is a powerful RL algorithm that has been shown to address many of the challenges encountered in learning complex and intricate control systems. Its ease of implementation and the robustness and stability in the training process, make it an ideal candidate for controlling complex and non-linear systems . Consequently, PPO was a logical choice for the control system application examined in this project.

Control methods need to be proven using complex, challenging biological systems in order to show the potential of in-silico feedback control of cellular processes for synthetic biology and

Cybergenetics. Physical control theory's quintessential test case is stabilising a pendulum in its upwardly unstable equilibrium state. Similarly, achieving the unstable equilibrium state of a bi-stable gene regulatory network and maintaining it represent a significant step toward developing in-silico feedback control of gene regulatory networks [27]. Because bi-stable networks exhibit hysteresis, they pose specific control challenges and make for exemplary systems to show that RL can be a viable solution in controlling intricate biological systems. Therefore, in this project, the PPO algorithm is employed to control the GTS to investigate the potential of RL for addressing the unique challenges posed by complex and challenging biological systems in the field of Cybergenetics. Additionally, to evaluate the efficacy of the PPO algorithm, its performance will be compared to that of a PID controller.

Chapter 3

Methodology

3.1 Controlling the Genetic Toggle Switch

A comprehensive investigation of four control methodologies is conducted to control the GTS effectively. The first approach involves employing the PPO algorithm from the stable-baseline3 library to train an agent in a custom gym environment created using the OpenAI framework [28] [1]. The secondary approach involves the analysis of an untrained PPO agent on the custom gym environment, the purpose of this evaluation is to provide a baseline comparison to highlight the learning process of the PPO agent and the value of the RL approach to controlling complex biological systems. Other approach is a PID controller, a classical benchmark in control systems which provides as an essential point of comparison when evaluating the performance of the trained PPO agent. This allows for a better understanding of pros and cons of both techniques and determines whether the PPO agent offers any distinct advantages over PID. Lastly, a relay method is employed. It is known as a simple control strategy which alternates output between two states, usually on and off, which depends if the error is positive or negative. The simplicity of the relay method provides a baseline to assess the trained PPO agent's capabilities.

Test Cases

- Untrained/ Uncontrolled PPO
- Trained PPO
- PID Controller
- Relay Method

The bi-stability behaviour of the GTS is modulated through the incorporation of inducers, aTc and $IPTG$. These inducers serve as the primary control inputs for the experimental procedure. It is important to note that the diffusion of these molecules was not implemented into the system's design, as Lugagne stated that it negligibly affects the system's behaviour and the location of the equilibrium points. [27] [12]

3.2 Environment

The main objective of this investigation is to stabilise a single cell around the GTS's unstable equilibrium state for an extended duration. To simulate the behaviour of the GTS, a custom gym environment was implemented using the OpenAI gym library. It is an open source library for developing and testing RL algorithms, the ease of use makes it a popular choice for developers. The fundamental concepts of RL are illustrated in 3.1; the agent observes the environment,

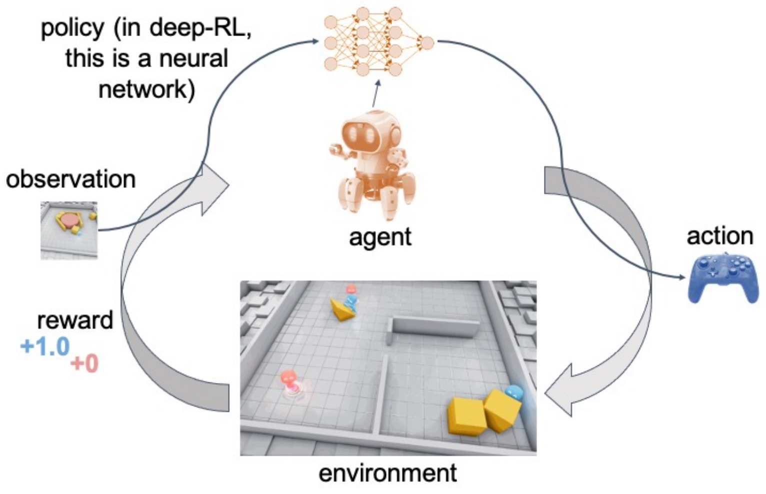


Figure 3.1: A visual illustration of an agent interacting and communicating with its environment. [2]

takes an action and receives a corresponding reward for that action. This cycle is iterated over time, allowing the agent to learn a policy through experience and optimisation gradually [34]. A custom gym environment is designed to capture these fundamental concepts of RL. The custom gym environment implemented in this project consisted of four main functions: The *init* function, *step* function, *reset* function and the *render* function. The *init* function initialises the environment with the appropriate environment parameters while also defining the action and observation space. The action space in this custom environment was a discrete space consisting of four possible actions (0, 1, 2, or 3), where each action corresponds to a combination of increasing or decreasing the values of the aTc and $IPTG$.

Action Space

- Action 1 - Increase the concentration of both aTc and $IPTG$
- Action 2 - Increase the concentration of aTc but decrease $IPTG$

- Action 3 - Decrease the concentration of aTc but increase $IPTG$
- Action 4 - Decrease the concentration of both aTc and $IPTG$

aTc is increased and decreased by an increment of 10 $ngmL^{-1}$ and $IPTG$ is increased and decreased by an increment of 0.1 mM. The observation space is a box space containing four continuous values that represents the current state of the system, which includes the values of $mRNA$ of $LacI$ and $TetR$, and also the values of $LacI$ and $TetR$. To account for potential large values of the state, the observation space is continuous and unbounded. The *step* function is responsible for executing a single time step in the environment; it takes an action from the agent which it chooses from the action space which is defined in the *init* function, updates the current environment state and calculates the reward. Specifically the reward function is based on the Euclidean distance between the current state of the system and the target set-point.

$$Reward = -\sqrt{((LacI_{target} - LacI(t))^2 + ((TetR_{target} - TetR(t))^2} \tag{3.1}$$

The agent is rewarded the negative of the euclidean distance such that the agent receives a higher reward the closer it is to the target set-point (unstable state), and a lower reward if it is further away from the set-point. The *reset* function prepares for a new episode as it initialises the environment's state and initial conditions. To elaborate further, at the start of an episode, the initial state of the system is randomised. Specifically, $mRNA$ of $LacI$ and $TetR$ and the values of $LacI$ and $TetR$ are randomised to values between 0 and 1000. This randomisation of the initial conditions of the state is to ensure the agent encounters a diverse range of states and situations during the training process which is essential for its ability to learn and adapt to different scenarios. The render function illustrates the cell's path trajectory in the $LacI$ and $TetR$ space in a 2D plot.
After defining the environment, the agent is trained on the environment using the stable-baseline3 library PPO algorithm. The agent is trained on a 1000 episodes and the length of an episode is 1000 minutes (steps). PPO updates the parameters of the policy network based on the observations, actions and rewards collected during the training episodes.

3.3 PPO Hyperparameters

Hyperparameters are adjustable parameters that are manually set by the user. These hyperparameters govern the behaviour of the algorithm and can have a significant impact on the agent's learning process and its performance as they determine how the learning algorithm updates the model's parameters during training. Therefore, choosing the appropriate hyperparameters for PPO is critical for achieving the optimal performance of the agent [10]. Poor choices of the hyperparameters can lead to slow learning or even convergence to sub-optimal solutions. [33] [10]

The significance of hyperparameters in PPO cannot be understated, as they play a crucial role in understanding its behaviour and tuning its performance [10]. These hyperparameters are essential in controlling various aspects of the learning process, such as the step size in the optimisation, the importance of future rewards, the exploration and exploitation trade-off and the degree of policy change in each update [33]. The key hyperparameters of PPO encompass:

1. Learning rate: This hyperparameter is responsible for the step size in the optimisation process. A smaller learning rate might lead to slower convergence, while a larger learning rate can cause instability in training. [30]

2. Discount factor: The discount factor determines the importance of future rewards. A value closer to 1 gives more weight to future rewards, while a value closer to 0 will cause the agent to focus more on immediate rewards. [33]

3. Generalised Advantage Estimation parameter (lambda): This parameter determines the balance of the bias-variance trade-off in the advantage estimation. A value closer to 1 will result in higher variance and lower bias, while a value closer to 0 will result in lower variance and higher bias. [33]

4. PPO clipping parameter (ϵ): The clipping parameter is responsible for the degree of policy change allowed in each update. A smaller clipping parameter restricts the policy update more tightly, preventing drastic changes that might lead to instability in training. A larger clipping parameter allows for more significant policy updates, which can potentially speed up learning but also introduce instability. [33]

5. Number of epochs: Epochs determine the number of iterations used for updating the policy using the collected trajectories. Increasing the number of epochs can lead to better convergence but also requires more computation time per update.[33]

With time constraints and limited computational resources, hyperparameter tuning was ruled out. Therefore, the choice was to employ the default values for the hyperparamters from the stable-baseline3 library. The default hyperparameters are shown in 3.1.
However, these default values have demonstrated their effectiveness across a range of tasks and settings, offering a reasonable starting point for experimentation. Although a hyperparameter tuning process might yield a better agent performance, the use of default values served as a pragmatic approach within the constraints of the project.[33]

3.4 PID Tuning

In this project, a dual PID controller was appointed to regulate the levels of $LacI$ and $TetR$. More precisely, one controller adjusts the concentration of aTc to minimise the deviation

Hyperparameter	**Default Value**
Learning Rate	0.0003
Number of Steps	2048
Batch Size	64
Number of Epochs	10
Discount Factor	0.99
Generalised Advantage Estimator	0.95
Clipping Parameter	0.2
Coefficient of Entropy Bonus	0
Coefficient of Value Function	0.5
Maximum Gradient Norm	0.5

Table 3.1: Default hyperparameter values of PPO in Stable Baselines 3.[33]

between the target set-point of $LacI$ and the value of $LacI$, while the other controller adjusts the concentration of $IPTG$ to minimise the deviation between the set-point of $TetR$ and the $TetR$ value. It should be emphasised that the performance of the PID controller heavily depends on its gain parameters, namely the proportional gain (K_C), integral gain (K_I) and derivative gain (K_D). Therefore, tuning a PID controller parameters is an essential step to achieve an optimal performance and there are various strategies that can be employed, such as manual tuning, the Zielger - Nichlos method, optimisers, the Cohen-Coon method and many more. Manual tuning was the preferred choice for the dual PID controllers tuning process due to the complex and non-linear nature of the GTS. This unpredictable behaviour can cause difficulties when applying conventional tuning methods such as the Ziegler - Nichlos and the Cohen-Coon method [4] [37]. The dual PID controller employed in this project is denoted by the equations 3.2 and 3.3. Where $e_L(t) = LacI_{Target} - LacI(t)$ and $e_T(t) = TetR_{Target} - TetR(t)$, u^0_{aTc} and u^0_{IPTG} represents the initial values of aTc and $IPTG$ respectively.

$$u_{aTc} = u^0_{aTc} + K^L_C e_L(t) + K^L_I \int_0^t e_L(\tau)d\tau + K^L_D \frac{de_L(t)}{dt} \tag{3.2}$$

$$u_{IPTG} = u^0_{IPTG} + K^T_C e_T(t) + K^T_I \int_0^t e_T(\tau)d\tau + K^T_D \frac{de_T(t)}{dt} \tag{3.3}$$

Manually tuning the gain parameters of the PID controller involves an iterative adjustment of the proportional, integral and derivative terms. The tuning process starts by setting the integral and derivative term to zero and increasing the proportional term until the system produces a stable oscillation about the set-point. Next, the integral term is introduced and is gradually increased to minimise the steady-state error and eliminate potential offset errors. Then the derivative term is added and adjusted to provide damping and enhance system stability. From this process, the final gain parameters used in the PID controllers are presented in the table 3.2. [5]

Gain Parameters	**Value**
K_C^L	1.9
K_I^L	0.02
K_D^L	0.01
K_C^T	0.06
K_I^T	0.0001
K_D^T	0.0001

Table 3.2: Gain parameters of PID controller obtained from manual tuning

3.5 Relay

A dual relay controller was employed in this project due to its ease of implementation. The method works by switching the output between two states depending on the error, which is the difference between the observed value and the set-point. In the context of the GTS, the switch between two states refer to switching between a certain value of $aTc/IPTG$. A range of $aTc/IPTG$ values were investigated to determine the optimal performance, resulting that the value of aTc should switch between 20 and 0, whereas for $IPTG$, it should switch between 0.25 and 0.

$$u_{aTc}(t) = \begin{cases} u_{aTc}^{max}, if \quad e_L(t) > 0 \\ u_{aTc}^{min}, if \quad e_L(t) \leq 0 \end{cases} \tag{3.4}$$

$$u_{IPTG}(t) = \begin{cases} u_{IPTG}^{max}, if \quad e_T(t) > 0 \\ u_{IPTG}^{min}, if \quad e_T(t) \leq 0 \end{cases} \tag{3.5}$$

where $e_L(t) = LacI_{Target} - LacI(t)$ and $e_T(t) = TetR_{Target} - TetR(t)$. This switching between states causes an oscillatory behaviour around the desired set-point.

3.6 Performance Metric

Each of the test cases must be evaluated in order to extract meaningful insights on their capability of controlling the GTS. In this project, the Integral Squared Error (ISE) was used, averaged across over 5 replicate trials conducted by each method. More specifically two tests were conducted, one test with arbitrary initial conditions across the five trials and the other test with the same initial conditions across the five trials. The mean and the standard deviation of the ISE values across these trials were computed and plotted. The variation of the initial conditions in the trials provides a way to evaluate the controller's ability to adapt and perform under unique situations. ISE is defined as the sum of the squared error.

$$ISE_{LacI} = \int_{t0}^{T} (LacI_{target} - LacI(t))^2 dt \tag{3.6}$$

$$ISE_{TetR} = \int_{t0}^{T} (TetR_{target} - TetR(t))^2 dt \tag{3.7}$$

$$ISE_{total} = ISE_{LacI} + ISE_{TetR} \tag{3.8}$$

Considering that each of the control systems involves two control inputs, $LacI$ and $TetR$, it is vital to assess the ISE for each input to evaluate the performance of the control strategy. Therefore, the total ISE of the control system is the summation of the $LacI$ and $TetR$ ISEs. ISE was chosen due to its ability to factor in both the magnitude and the duration of the error. Furthermore, the squared error places a greater emphasises on large errors, imposing a significant penalty if the deviation from the set-point is substantial. This characteristic is beneficial in the context of this project as the goal is to minimise the large errors.

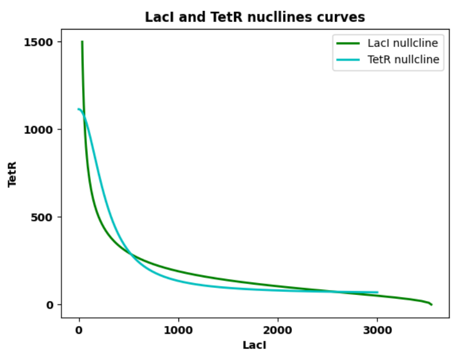
Chapter 4

Model Validation

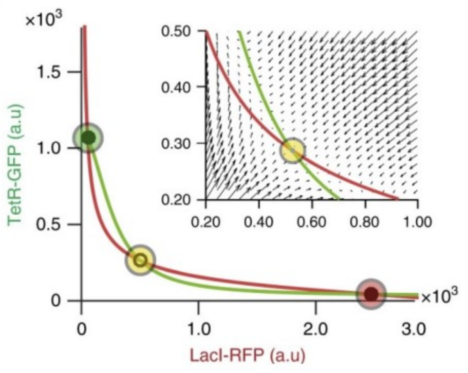
This chapter focuses on the task of model validation. It is an essential step in the process of verifying a mathematical model's ability to capture the behaviour of the biological system. Specifically, the chapter aims to validate the GTS model proposed in this project against the observations made by Lugagne et al. Analytical methods such as nullcline plots and stable state plots will be employed, providing clear insights into how the systems behave under certain conditions.

4.1 Nullclines

One way to ensure the GTS being simulated in the custom gym environment is valid, is to reproduce the nullcline plots presented in Lugagne's paper. By definition a nullcline plot is a set of points in the phase space of the system where the rate of change of a particular variable is zero [36] [38]. Nullcline plots are one way to access important insights into the behaviour of the system, allowing it to visualise many features such as limit cycles and stable points [38]. In the context of GTS, as mentioned earlier, the GTS is represented as a 4-dimensional ODE system, where it outputs the values of $mRNA$ of $LacI$ and $TetR$, and the value of $LacI$ and $TetR$. Since, the goal of this project is to control the expression level of $LacI$ and $TetR$, only their nullclines of them are focused upon in this project, and this was also the case for Lugagne. Figures 4.1a and 4.1b represent the nullcline plots produced by the GTS model and the nullcline plot produced from Lugagne's paper respectively. Both nullcline plots were developed under reference conditions $aTc = 20\ \frac{ng}{ml}$, $IPTG = 0.25\ mM$ and the numerical values of the constants presented in table A.1 . The intersection of the nullclines for $LacI$ and $TetR$, represent two stable equilibrium points and one unstable equilibrium point. The points are illustrated in Lugagne's nullcline plots where the green ($TetR$) and red ($LacI$) circle represents the stable equilibrium state and the yellow circle represents the unstable equilibrium state. From both of the plots, it is evident that the stable and unstable states are formed in the same location in the state space.

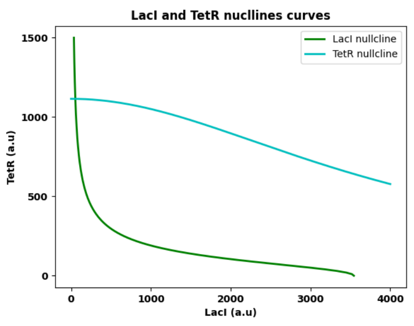


(a) GTS model Nullcline plot.

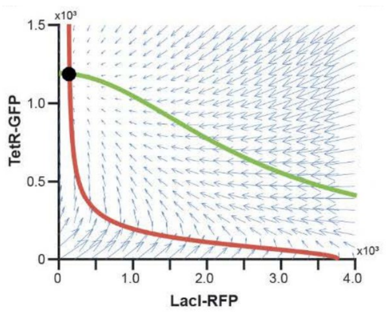


(b) Lugagne's Nullcline plot.[27]

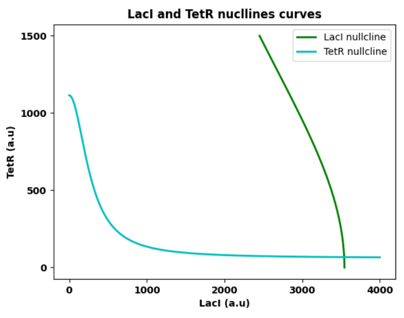
Figure 4.1: Nullcline plots under the reference conditions $aTc = 20\ \frac{ng}{ml}$, $IPTG = 0.25\ mM$ and the numerical values of the constants presented in table A.1.



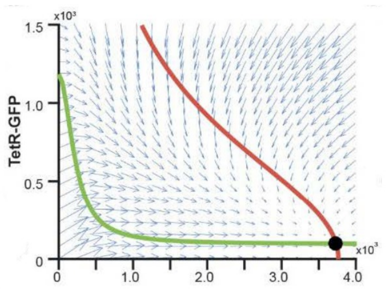
(a) GTS model Nullcline plot under the reference conditions $aTc = 20\ \frac{ng}{ml}$, $IPTG = 1.0\ mM$.



(b) Lugagne's Nullcline plot under the reference conditions $aTc = 20\ \frac{ng}{ml}$, $IPTG = 1.0\ mM$. [27]



(c) GTS model Nullcline plot under the reference conditions $aTc = 100\ \frac{ng}{ml}$, $IPTG = 0.25\ mM$.



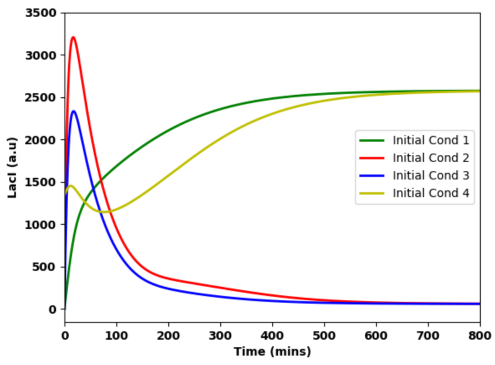
(d) Lugagne's Nullcline plot under the reference conditions $aTc = 100\ \frac{ng}{ml}$, $IPTG = 0.25\ mM$.[27]

Figure 4.2: Nullcline plots of the mono-stable behaviour of the GTS. The numerical values of the constants presented in table A.1.

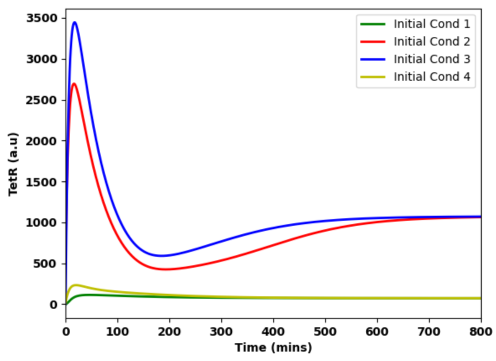
Figure 4.2 exhibits four nullcline plots, where 4.2a and 4.2b represents the system under the reference conditions $aTc = 20\ \frac{ng}{ml}$, $IPTG = 1.0\ mM$ and figure 4.2c and 4.2d represents the system under the reference conditions $aTc = 100\ \frac{ng}{ml}$, $IPTG = 0.25\ mM$, all nullcline plots are consistent with the numerical values of the constants presented in table A.1. The four nullcline plots, exhibits a mono-stable behaviour, producing only one stable equilibrium state, which is represented by the nullcline intersection. In figure 4.2a and 4.2b, it is demonstrably evident that the nullcline intersection (stable equilibrium point) for the GTS model's and Lugagne's plots are the same. This is also the case for figure 4.2c and 4.2d, proving that the GTS model is yielding results that are consistent with the results generated by Lugagne's model.

4.2 Time Course Plots

Time course plots are a graphical representation of a variable's behaviour over time, where the x-axis represents the time and the y-axis represents the variable. This then visualises the dynamics of the system allowing for a clear understanding of the trends and patterns within the system over time. In the context of GTS, the variables are the concentration of the genes $LacI$ and $TetR$. Time course plots were constructed to supplement the nullcline plots, the nullcline intersections represent stable points so by plotting the behaviour of $LacI$ and $TetR$ over time converging to the stable points, it will validate the accuracy of the nullcline plots.



(a) Stabilisation of $LacI$ concentration over time under various initial conditions..



(b) Stabilisation of $TetR$ concentration over time under various initial conditions.

Figure 4.3: Time course plots of $LacI$ and $TetR$ stabilising to their respective stable states with four arbitrary initial conditions under the reference conditions $aTc = 20\ \frac{ng}{ml}$, $IPTG = 0.25\ mM$ and the numerical values of the constants presented in table A.1 .

Figure 4.3a and 4.3b illustrates the trajectory of $LacI$ and $TetR$ converging to stable points. The time course plots were generated under the reference conditions $aTc = 20\ \frac{ng}{ml}$, $IPTG = 0.25\ mM$ and the numerical values of the constants presented in table A.1. Referring to the figure 4.1, the stable points are represented by the co-ordinates [56, 1072] and [2576, 73] ($LacI$,

$TetR$) in the phase space. In figure 4.3a and 4.3b, it shows four trajectories with four arbitrary initial conditions (which is $mRNALacI$, $mRNATetR$, $LacI$ and $TetR$), this demonstrates that the stable point the system converges to depends on the initial condition. In figure 4.3a, "initial Cond 1"-[60.11025325,0,0,0] and "initial Cond 2"-[550, 360, 404, 530] illustrates the system converging to a stable $LacI$ value of 2576. Then with "initial Cond 3" - [440, 540, 0, 1] and "initial Cond 4" - [34, 33, 1344, 8], the system converges to a stable $LacI$ value of 56. In figure 4.3b, "initial Cond 1" and "initial Cond 2" converges to a stable $TetR$ value of 73, where "initial Cond 3" and "initial Cond 4" converges to a stable $TetR$ value of 1072, it is important to note that the arbitrary initial conditions used in figure 4.3b are the same as the ones used in 4.3a. The results of the time course plots are consistent with the nullcline intersections shown in figure 4.1a and 4.1b.

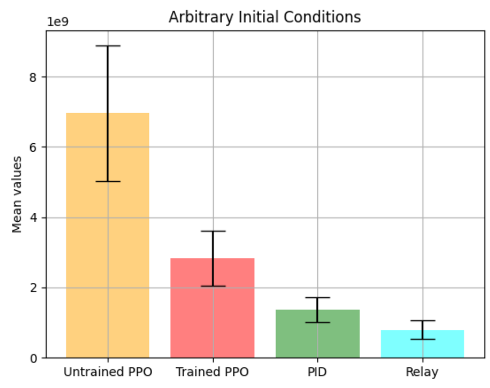
Chapter 5

Results

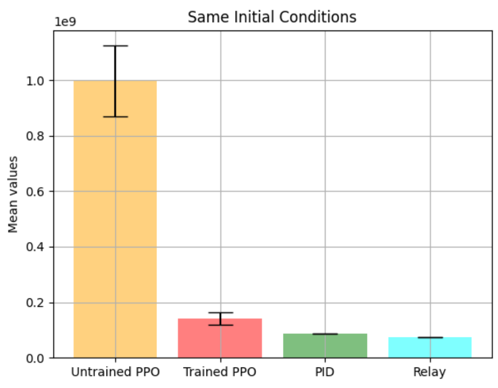
In this chapter, the results of the experiments comparing the performance of the four control methods are presented. It is important to note that each of the experiments in this project were applied to a single cell via in-silico simulation. First, the ISE of each control methods are presented to showcase their overall performance in regulating the GTS. To elucidate the factors responsible for the observed ISE plots, a detailed analysis of each experiment is conducted. Lastly, the trajectories of the single cell in the *LacI-TetR* phase space are plotted. Additionally, the Euclidean distance between the system's state (single cell) and the target set-point is analysed over time, this provides insights into the ability of each controller to maintain the system near the desired set-point. The plots of the control variables aTc and $IPTG$ are not presented in this chapter as it will not provide useful insights and also the primary goal of this chapter is to evaluate the performance of the controllers. However, the plots of the control variables will be plotted and are available in the A.

5.1 Performance Assessment

Figure 5.1 presents two ISE plots, with figure 5.1a showing the ISE error plots for arbitrary initial conditions and 5.1b showing the ISE plots for the same initial conditions. From both of the figures, it is evident that the untrained PPO was the worst performing controller followed by the trained PPO. The PID controller and the relay controller outperformed trained PPO by a relatively large margin. Furthermore, the low values of standard deviation observed for the Relay and PID controllers indicate their robustness in the given control scenarios. There may be several factors for the success of Relay controller in regulating the GTS compared to the limitations observed in the PPO and PID controllers. The following sections aims to examine these contributing factors in greater detail.



(a) Tested over 5 trials with arbitrary initial conditions each time.

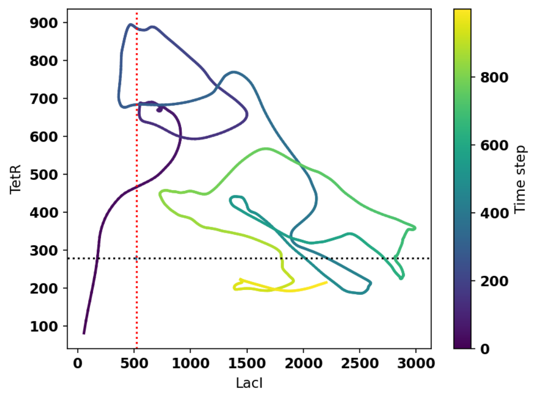


(b) Tested over 5 trials with the same initial conditions each time.

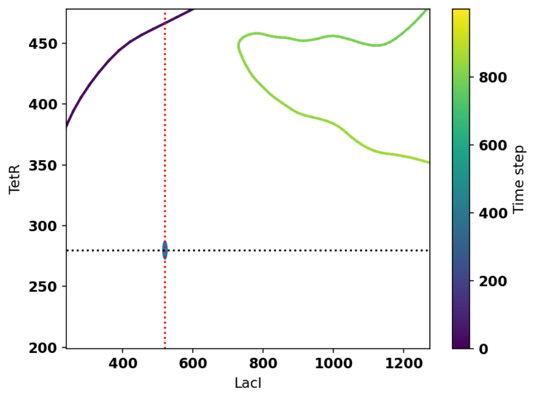
Figure 5.1: Mean performance of the four controllers tested for 5 trials, measured by Integral Squared Error (ISE). Error bars showing one standard deviation.

5.2 Untrained PPO

Figure 5.2a presents an illustration of the cell's trajectory during an episode. The intersection of the red vertical dotted line and black horizontal dotted line represents the unstable state. It is evident to see the untrained PPO fails to converge at this unstable state, figure 5.2b provides a more detailed depiction. This was expected as the untrained PPO agent does not obtain any policy and instead chooses random actions, Not considering the rewards or the exploration/ exploitation trade-off. However, as mentioned before, the purpose of the untrained PPO agent was to provide a baseline comparison to the rest of the controllers.



(a) Cell path in the *LacI-TetR* space.



(b) A zoomed in look of the cell path.

Figure 5.2: Trajectory of a single cell within the *LacI-TetR* phase space for an untrained PPO agent. Each simulation (length of an episode) spans for 1000 minutes (steps).

Figure 5.3 represents the Euclidean distance of the system's state and the target set-point over time. The overall trend is very stochastic and does not follow a clear pattern, and this corresponds to the agent's behaviour. Since the agent has not learnt any policy, it's behaviour is random.

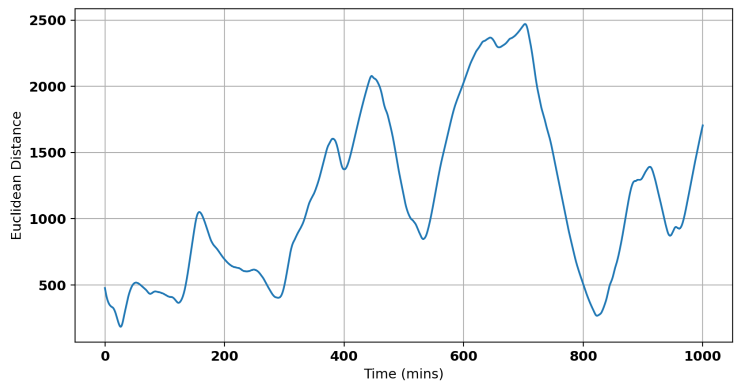
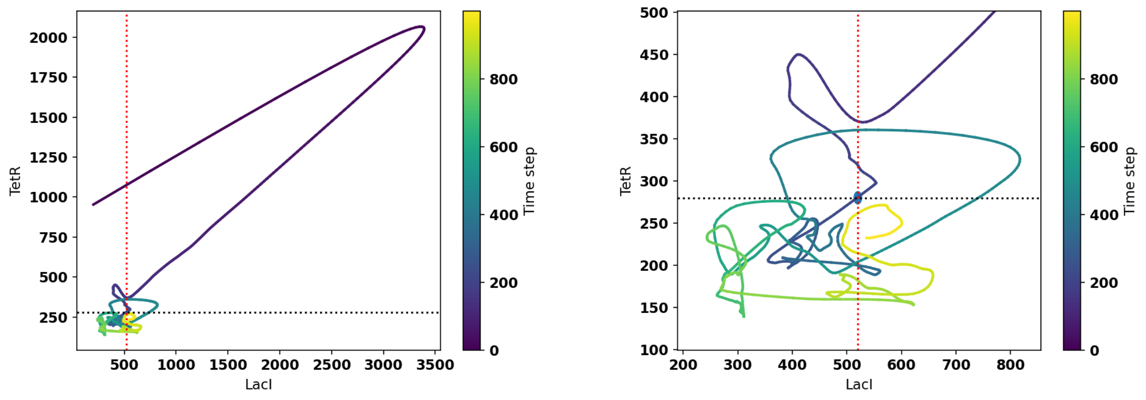


Figure 5.3: The euclidean distance of the state at time t and the target set-point (unstable state) over time. Note that this figure represents the final episode of the agent's 1000-episode run.

5.3 Trained PPO

The trained PPO underwent a learning process of 1000 episodes with each episode being 1000 minutes (steps) and with the default hyperparameter values from the stable-baseline3 library. In comparison to the Proportional-Integral-Derivative (PID) controller and relay-based control strategies, the PPO agent demonstrated relatively poorer performance. However, this does not imply that the PPO agent performed poorly in an absolute sense, but rather that its performance was sub-optimal when compared to the other methods. Figure 5.4a depicts the trajectory of the controlled cell during the 1,000th episode, highlighting the agent's acquisition of a near-optimal policy. Although the cell approaches the unstable state, it exhibits stochastic behavior in the vicinity of the stable state. A more detailed examination of the cell's path close to the unstable state is presented in Figure 5.4b. The observed behavior is reflected in the mean ISE value for the trained PPO agent, which indicates that the cell does not fully converge to the unstable state. Several factors could contribute to this outcome, which will be further explored and discussed. For instance, the choice of hyperparameters, the network architecture, or the exploration-exploitation balance during training may impact the agent's performance. Additionally, it is important to consider whether the PPO algorithm is inherently suitable for this specific control task, as some reinforcement learning techniques may be better suited to certain problems than others.



(a) Cell path in the *LacI*-*TetR* space. (b) A zoomed in look of the cell path

Figure 5.4: Trajectory of a single cell within the *LacI*-*TetR* phase space for the trained PPO agent. Each simulation (length of an episode) spans for 1000 minutes (steps).

Based on figure 5.5, it appears that the trained PPO agent performed fairly well in controlling the genetic toggle switch system about its unstable state. The graph shows that the distance between the system's state and the unstable set point (i.e., the ED) decreases over time. While the system does not converge to zero, the fact that the ED remains close to the set point indicates that the PPO agent is able to maintain the system in a stable state. A comparison between the untrained and trained PPO agents clearly indicates that the trained PPO has acquired a policy that is effective in regulating the deterministic GTS system. Overall, figure 5.5 suggests that the trained PPO agent is a promising solution for controlling complex biological systems, although further evaluation and comparison with other control algorithms may be necessary to determine its efficacy in a broader context.

5.4 PID

The dual PID controller yielded a favourable performance, as evidenced by figures 5.6a and 5.6b. The single cell converges to the unstable state which is illustrated by the intersection of the dotted red line and the dotted black line. Figures 5.6a and 5.6b indicate the PID controller exhibits superior control over a deterministic GTS, compared to the trained PPO agent.
This could be due to effective tuning of the gain parameters, which ensures a robust and stable control mechanism. Additionally, PID control is deterministic, it directly responds to the error making it a potentially more suitable control strategy for deterministic GTS.
Figure 5.7 illustrates the Euclidean distance over time for the PID controller. The general trend of the graph demonstrates a decline over time, which aligns with the desired outcome. A minor

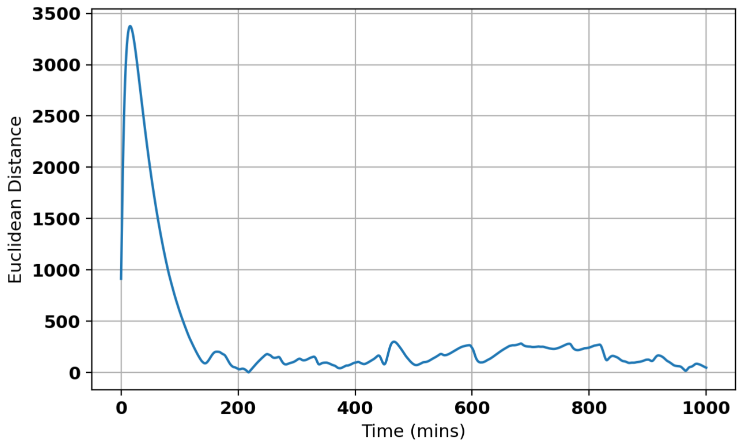
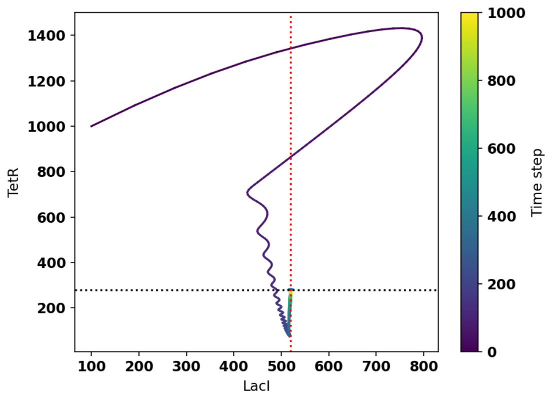
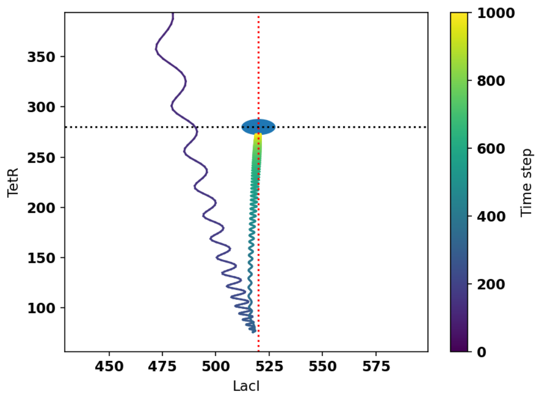


Figure 5.5: The euclidean distance of the state at time t and the target set-point (unstable state) over time of the trained PPO agent.



(a) Cell path in the *LacI*-*TetR* space.



(b) A zoomed in look of the cell path

Figure 5.6: Trajectory of a single cell within the *LacI*-*TetR* phase space for the dual PID controller. Each simulation (length of an episode) spans for 1000 minutes (steps).

increase in the Euclidean distance is observed between time intervals 170 (minutes) and 330 (minutes), this could potentially be due to the delayed convergence of TetR to its set-point compared to the faster convergence of LacI to its target set-point. In conclusion, the observed trend in the graph corresponds to the objective of minimizing the Euclidean distance as much as possible, which serves as an indicator that the single cell approaches and remains around the unstable state.

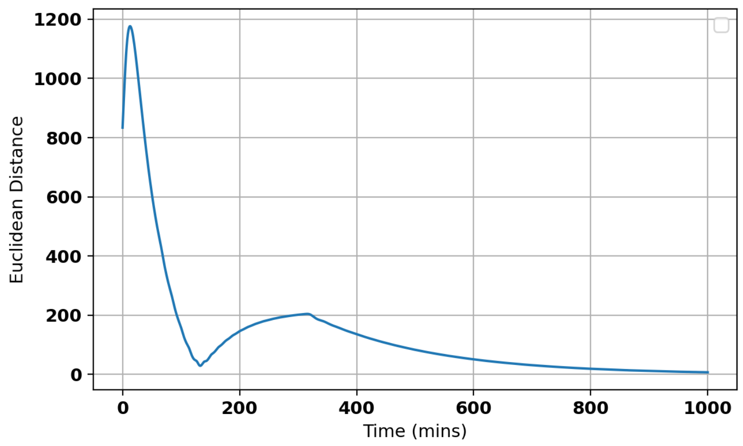
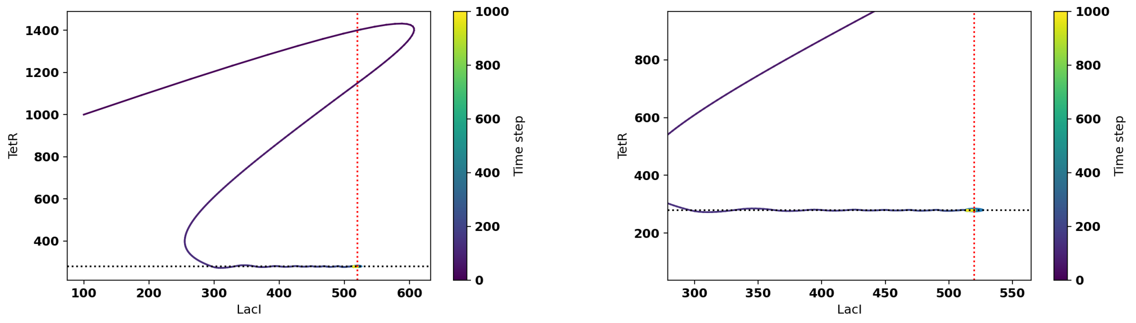


Figure 5.7: The euclidean distance of the state at time t and the target set-point (unstable state) over time of the dual PID controller.

5.5 Relay

The relay controller demonstrated superior performance in comparison to the other controllers evaluated. It attained the minimum mean and standard deviation values across both trials, indicating that the relay controller exhibits robustness and is apt for regulating a deterministic GTS. In Figure 5.8a, the cell's trajectory exhibits rapid convergence to the unstable state, surpassing the rates exhibited by alternative controllers. This swift convergence behavior is further illustrated in Figure 5.8b, reinforcing the efficacy of the relay controller for this application. The fast response could be due to the on-off control mechanism which reduces the complexity in implementation.



(a) Cell path in the *LacI*-*TetR* space.

(b) A zoomed in look of the cell path

Figure 5.8: Trajectory of a single cell within the *LacI*-*TetR* phase space for the relay controller. Each simulation (length of an episode) spans for 1000 minutes (steps).

Figure 5.9 illustrates detailed insights into the controller's performance, demonstrating exceptional efficacy as the graph exhibits a rapid convergence to zero distance within approximately 250 minutes, followed by a sustained period of stability at that level.

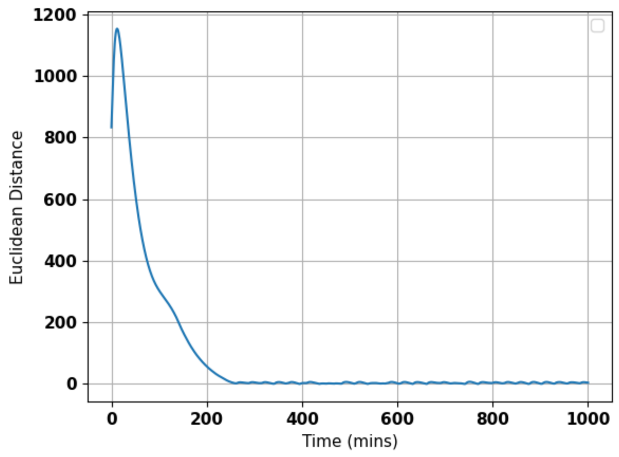


Figure 5.9: The euclidean distance of the state at time t and the target set-point (unstable state) over time of the relay controller.

Chapter 6

Discussion and Further Work

The trained PPO agent showed satisfactory performance, but its performance was notably inferior to that of the PID and relay-based control strategies. One possible explanation for the sub-optimal performance of the PPO agent is the lack of appropriate hyperparameter tuning. While the default hyperparameter values provided by the stable-baseline3 library are often effective in many applications, they may not be optimal for all tasks, such as controlling the GTS in this study. The default hyperparameters failed to achieve the most optimal performance, although it is worth noting that there may be other factors that contribute to the sub-optimal performance of the PPO agent. PPO is a model-free reinforcement learning algorithm that heavily relies on selecting optimal hyperparameters. The selection of hyperparameters such as the clip ratio, learning rate, batch size, and entropy coefficient all impact the exploration/exploitation trade-off, the agent must balance the exploration of new actions with the exploitation of the current optimal policy. To improve the performance of the PPO agent, it is crucial to conduct hyperparameter tuning to select the optimal values of these hyperparameters. This can be accomplished through various techniques, including grid search or Bayesian optimisation, to identify the hyperparameter settings that lead to the best agent performance.

Another reasoning for the sub-optimal performance of the trained PPO agent could be due to the length of training process, more precisely the length of each episodes and the number of episodes during training . The length of episode affects the agent's ability to explore and learn from it interacting with the environment, if the length of the episode is not sufficient for the agent to explore and learn, then this can cause the agent to learn a sub-optimal policy or it can even result to the agent not learning any policy. Similarly, the number of episodes used in the training process can affect the agent's performance, insufficient number of episodes can lead to the agent learning a sub-optimal policy or if the number of episodes are too high then it can lead to over-fitting to the training data. In the experiments, the training process consisted of 1000 episodes with each episode consisting of 1000 steps (minutes in simulation time). These values were chosen due to the limited computational resources and the time constraints, so an investigation into the length and the number of episodes that achieved the optimal performance of the trained PPO agent was not conducted. So it is important to note that a comprehensive investigation into a range of episode lengths and numbers is necessary to determine the best values for controlling the GTS system. Choosing the values of the length and number of episode that yield the optimal performance may not be appropriate in certain circumstances. Many users may want to balance performance with data efficiency as they might lack computational resources so selecting the length of episodes or the number of episodes is entirely up to the user.

The environment's architecture is critical in an agent's performance, it provides the agent the necessary observations and actions to learn an optimal policy. It is crucial that the environment accurately represents the system's dynamics and give the agent appropriate observations and actions, because if the environment's architecture does not provide the agent relevant observations and actions then this can lead to a sub-optimal performance and policy. In the custom gym environment, a mathematical model of the GTS was implemented with a discrete action space. From openAI gym library, it states the action space can be classified into two categories: discrete and continuous. The choice of discrete action space was employed in this project simply due to interpretability of discrete actions. However, a continuous action space may have produced better results in terms of the agent's performance, this is due to the continuous action space providing a wider range of actions with greater precision. Additionally, the reward signal is crucial to the agent's performance as it's purpose is to tell the agent how to behave. In the experiments, the reward function was designed to penalise the agent based on the distance between itself and the target set-point. While this reward function yielded satisfactory performance, a different or a better-designed reward function could've produced better agent performance. An in-depth investigation into different versions of the action and observation spaces, as well as the reward system, was not conducted due to time and computational resource constraints. Therefore, to achieve an optimal performance of the PPO agent, the architecture of the environment is crucial.

The PID controller showcased superior performance when compared to the trained PPO agent, but was outperformed by the relay method. Achieving the second-lowest mean ISE for both tests, the PID controller shows promise in controlling complex biological systems. However, the PID controller could have yielded even better performance as the manual tuning method utilised in this project might not have produced the optimal values for the gain parameters. Although manual tuning offers a simple strategy to optimise the PID controller, there are far better options that can obtain values of the gain parameters that yield optimal performance. Therefore, the PID controller might have the potential to surpass the performance of the relay method. The relay controller exhibited the best performance among all the controllers tested in the experiments. This could be due to the simplicity of the method, allowing for efficient and effective responses to the system changes. The excellent performance of both PID and the relay controller could potentially suggest that a RL approach may not be necessary for controlling the deterministic GTS. However, it is important to note the GTS model used in this project was deterministic. Although they provide valuable insights into the dynamics of the system, they do not always account for the full complexity of biological systems, as stochastic effects are not considered. Therefore, if the PID and the relay controller was tasked to control a highly stochastic system, it might exhibit poor performance.

It is crucial to consider the practical implications and challenges of applying these controllers in real-world Cybergenetics experiments. Firstly, although the PID controller is a widely adopted and well-established technique, tuning of the controller requires substantial time investment. Especially manual tuning, while it proved to be effective in this project by producing excellent performances, in real-world experiments, manual tuning is not a scalable or efficient method to optimise the PID controller. This is due to the iterative process of tuning being very costly and time consuming [16]. Furthermore, real-world Cybergenetics experiments will often involve systems with high non-linearity, complexity and stochastic behaviour. The PID proved to be effective in a deterministic model of the GTS but might not yield the same performance when subjected to highly stochastic systems. The PID controller is not the only method with limitations in real-world applications, the relay controller also suffers. The discontinuous nature of the relay controller's actions can induce oscillations or fluctuations in the controlled variable which can potentially compromise the stability of the system in control. Specifically, the abrupt and sudden changes of the controller's action may have negative impacts, rather than providing beneficial effects [15]. It is noting that although the tuning of the relay controller is much simpler than of the PID controller, it still requires the same iterative process of the PID controller, which can be costly or even detrimental to the system in real-world experiments [15] [16]. The PPO also has limitations when applied to real-world applications. In this project, the trained PPO agent underwent a learning process of 1000 episodes, with each episode consisting of 1000 steps (minutes in simulation time). The agent achieved a satisfactory performance with these specific values however the complex and stochastic nature of real-world environments presents a significant challenged in training the PPO agents. The PPO agent may require extensive training to develop an optimal policy that can account for the non-linearity and stochastic nature of real-world systems and environments. As mentioned before, higher performance may be attainable with a more extensive training process, this can cause substantial perplexities as longer training processes can be costly and time-consuming.

In summary, the trained PPO agent demonstrated satisfactory performance as evidenced in figures 5.4a and 5.4b. The learning process of the trained PPO agent can be further supported by the contrasting performance of the untrained PPO agent. Despite exhibiting a commendable learning capability, the trained PPO agent was outperformed by the PID and relay controllers. As previously discussed, several factors could contribute to the sub-optimal performance of the trained PPO agent. Further research could focus on exploring the mentioned factors, all while considering the experimental application, as the ultimate goal of this project was to evaluate the feasibility of MFRL in Cybergenetics.

Chapter 7

Conclusion

In this study, a performance evaluation of PPO was conducted to demonstrate the feasibility of MFRL in Cybergenetics. The PPO agent was assessed on a task requiring the regulation of a deterministic GTS in its unstable state for an extended duration. The performance of the PPO agent was compared against an untrained PPO agent, a PID controller and a relay controller. A performance metric of ISE was employed to analyse the performance of the four controllers. The PID and the relay controller outperformed the trained PPO, where the PID and the relay controller obtained lower mean ISE values than the trained PPO across both tests. However, the observed improvement in performance from the untrained PPO to the trained PPO indicated that the trained PPO agent had indeed learned a policy and could effectively control the GTS in its unstable state. This suggests that a MFRL approach could be feasible in Cybergenetics, but further improvements must be explored before implementing it in real-world. Potential improvements such as hyperparameter tuning, extending the training duration and refining the environment architecture will allow for a more robust evaluation of PPO so it can demonstrate the feasibility of MFRL in Cybergenetics. Moreover, future research should also take into account of the practicality of experimental applications, where factors such as cost and time are of a significant importance. In conclusion the PPO agent demonstrated control capabilities that was comparable to those of the PID and the relay controllers, implying that MFRL may be a viable tool in Cybergenetics applications. However, the findings of this study is not conclusive, and further research will help to confirm the feasibility of MFRL for future use within Cybergenetics.

Appendix A

Appendix A

Transcription rates $(mRNA * min^{-1})$	K_L^{m0}	3.20e-2	*plac* regulation by $TetR$	θLacI	31.94
	K_T^{m0}	1.19e-1		ηLacI	2.00
	K_L^m	8.30		θ_{IPTG}	9.06e-2
	K_T^m	2.06		η_{IPTG}	2.00
Translation rates $(a.u. * mRNA^{-1} * min^{-1})$	K_{pL}	9.726e-1	*ptet* regulation by $LacI$	θ_{TetR}	30.00
	K_T^p	1.170		ηTetR	2.00
Degradtion rates $(mins^{-1})$	g_L^m	1.386e-1		θ_{aTc}	11.65
	g_T^m	1.386e-1		η_{aTc}	2.00
	g_L^p	1.65e-2	IPTG exchange rate (min^{-1})	k_{IPTG}^{in}	2.75e-2
	g_T^p	1.65e-2		k_{IPTG}^{out}	1.11e-1

Table A.1: Parameters for the GTS

Bibliography

[1] *Baselines3 docs - reliable reinforcement learning implementations¶*.

[2] *An introduction to reinforcement learning with openai gym, rllib, and google colab*.

[3] E. Andrianantoandro, S. Basu, D. K. Karig, and R. Weiss, *Synthetic biology: new engineering rules for an emerging discipline*, Molecular systems biology, 2 (2006), pp. 2006–0028.

[4] K. Astrom and T. Hagglund, *Pid controllers: Theory, design, and tuning, instrument society of america: Research triangle park, nc, 1995*, Google Scholar There is no corresponding record for this reference, (1954).

[5] K. J. Åström, T. Hägglund, and K. J. Astrom, *Advanced PID control*, vol. 461, ISA-The Instrumentation, Systems, and Automation Society Research Triangle Park, 2006.

[6] S. M. Brancato, F. De Lellis, D. Salzano, G. Russo, and M. di Bernardo, *External control of a genetic toggle switch via reinforcement learning*, arXiv preprint arXiv:2204.04972, (2022).

[7] K. Chua, R. Calandra, R. McAllister, and S. Levine, *Deep reinforcement learning in a handful of trials using probabilistic dynamics models*, Advances in neural information processing systems, 31 (2018).

[8] D. Del Vecchio, A. J. Dy, and Y. Qian, *Control theory meets synthetic biology*, Journal of The Royal Society Interface, 13 (2016), p. 20160380.

[9] F. Emmert-Streib, M. Dehmer, and B. Haibe-Kains, *Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks*, Frontiers in cell and developmental biology, 2 (2014), p. 38.

[10] M. Feurer and F. Hutter, *Hyperparameter optimization*, Automated machine learning: Methods, systems, challenges, (2019), pp. 3–33.

[11] S. Fujimoto, D. Meger, and D. Precup, *Off-policy deep reinforcement learning without exploration*, in International conference on machine learning, PMLR, 2019, pp. 2052–2062.

[12] T. S. Gardner, C. R. Cantor, and J. J. Collins, *Construction of a genetic toggle switch in escherichia coli*, Nature, 403 (2000), pp. 339–342.

[13] S. S. Gu, T. Lillicrap, R. E. Turner, Z. Ghahramani, B. Schölkopf, and S. Levine, *Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning*, Advances in neural information processing systems, 30 (2017).

[14] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*, in International conference on machine learning, PMLR, 2018, pp. 1861–1870.

[15] C. Hang, K. Astrom, and Q. Wang, *Relay feedback auto-tuning of process controllers—a tutorial review*, Journal of process control, 12 (2002), pp. 143–162.

[16] C. C. Hang, K. J. Åström, and W. K. Ho, *Refinements of the ziegler–nichols tuning formula*, in IEE Proceedings D (Control Theory and Applications), vol. 138, IET, 1991, pp. 111–118.

[17] A. Ibarguen, *A look at timekeeping in ancient times - montres publiques - the vintage watch magazine*, Apr 2022.

[18] M. Janner, J. Fu, M. Zhang, and S. Levine, *When to trust your model: Model-based policy optimization*, Advances in Neural Information Processing Systems, 32 (2019).

[19] J. Jaruszewicz-Błońska and T. Lipniacki, *Genetic toggle switch controlled by bacterial growth rate*, BMC systems biology, 11 (2017), pp. 1–11.

[20] G. Karlebach and R. Shamir, *Modelling and analysis of gene regulatory networks*, Nature reviews Molecular cell biology, 9 (2008), pp. 770–780.

[21] M. Khammash, M. Di Bernardo, and D. Di Bernardo, *Cybergenetics: Theory and methods for genetic control system*, in 2019 IEEE 58th Conference on Decision and Control (CDC), IEEE, 2019, pp. 916–926.

[22] M. H. Khammash, *Cybergenetics: Theory and applications of genetic control systems*, Proceedings of the IEEE, 110 (2022), pp. 631–658.

[23] H. Kobayashi, M. Kaern, M. Araki, K. Chung, T. S. Gardner, C. R. Cantor, and J. J. Collins, *Programmable cells: interfacing natural and engineered gene networks*, Proceedings of the National Academy of Sciences, 101 (2004), pp. 8414–8419.

[24] J. W. Kotula, S. J. Kerns, L. A. Shaket, L. Siraj, J. J. Collins, J. C. Way, and P. A. Silver, *Programmable bacteria detect and record an environmental signal in the mammalian gut*, Proceedings of the National Academy of Sciences, 111 (2014), pp. 4838–4843.

[25] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, *Stabilizing off-policy q-learning via bootstrapping error reduction*, Advances in Neural Information Processing Systems, 32 (2019).

[26] S. Levine, A. Kumar, G. Tucker, and J. Fu, *Offline reinforcement learning: Tutorial, review, and perspectives on open problems*, arXiv preprint arXiv:2005.01643, (2020).

[27] J.-B. Lugagne, S. Sosa Carrillo, M. Kirch, A. Köhler, G. Batt, and P. Hersen, *Balancing a genetic toggle switch by real-time feedback control and periodic forcing*, Nature communications, 8 (2017), pp. 1–8.

[28] Openai, *Openai/gym: A toolkit for developing and comparing reinforcement learning algorithms.*

[29] T. Ord, *The precipice: Existential risk and the future of humanity*, Hachette Books, 2020.

[30] S. Ruder, *An overview of gradient descent optimization algorithms*, arXiv preprint arXiv:1609.04747, (2016).

[31] I. Ruolo, S. Napolitano, D. Salzano, M. di Bernardo, and D. di Bernardo, *Control engineering meets synthetic biology: Foundations and applications*, Current Opinion in Systems Biology, 28 (2021), p. 100397.

[32] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, *Trust region policy optimization*, in International conference on machine learning, PMLR, 2015, pp. 1889–1897.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, arXiv preprint arXiv:1707.06347, (2017).

[34] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[35] C. Szepesvári, *Algorithms for reinforcement learning*, Synthesis lectures on artificial intelligence and machine learning, 4 (2010), pp. 1–103.

[36] R. Thomas, *Nullclines and nullcline intersections*, International Journal of Bifurcation and Chaos, 16 (2006), pp. 3023–3033.

[37] H. Wu, W. Su, and Z. Liu, *Pid controllers: Design and tuning methods*, in 2014 9th IEEE Conference on industrial electronics and applications, IEEE, 2014, pp. 808–813.

[38] M. Zabihi, S. Kiranyaz, V. Jäntti, T. Lipping, and M. Gabbouj, *Patient-specific seizure detection using nonlinear dynamics and nullclines*, IEEE journal of biomedical and health informatics, 24 (2019), pp. 543–555.