Reinforcement Learning and Optimal Control for Additive Manufacturing

A dissertation submitted for the degree of
Doctor of Philosophy

By:

Eleni Zavrakli

Under the supervision of:

Prof. Andrew C. Parnell
Prof. Subhrakanti Dey

Hamilton Institute
Maynooth University
Maynooth, Co. Kildare, Ireland

November 2023

Declaration

I, Eleni Zavrakli, hereby declare that I have produced this manuscript myself, without the prohibited assistance of third parties and without making use of aids other than those specified.

This work was conducted from August 2019 to November 2023 under the supervision of Professor Andrew C. Parnell and Professor Subhrakanti Dey in the Hamilton Institute, Maynooth University. This project was also part of I-Form Advanced Manufacturing research centre.

Eleni Zavrakli

Eleni Zavrakli

Maynooth, Ireland

November 2023

Sponsor

This work has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) under Grant Number 16/RC/3872 and is co-funded under the European Regional Development Fund.

Publications

Some of the work this thesis have been published in peer reviewed journals, submitted for publication and is currently under review, or presented in conferences and published in conference proceedings.

Submitted (under peer review):

- Zavrakli, E., Parnell, A., Dickson, A. and Dey, S., 2023. Data-driven Linear Quadratic Tracking based Temperature Control of a Big Area Additive Manufacturing System. Under review in the *Journal of Intelligent Manufacturing*, arXiv preprint `https://doi.org/10.48550/arXiv.2307.07039`

- Zavrakli, E., Parnell, A. and Dey, S., 2023. Output Feedback Reinforcement Learning with Parameter Optimisation for Temperature Control in a Material Extrusion Additive Manufacturing system. Under review in *Control Engineering Practice*, arXiv preprint `https://doi.org/10.48550/arXiv.2310.03599`

Published in conference proceedings:

Zavrakli, E., Parnell, A., and Dey, S., Output Feedback Reinforcement Learning for Temperature Control in a Fused Deposition Modelling Additive Manufacturing System, *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT), Rome, Italy, 2023*, pp. 1483-1487, `doi:10.1109/CoDIT58514.2023.10284168`

Peer reviewed journal article:

Zavrakli, E., Parnell, A., Malone, D., Duffy, K. and Dey, S. 2023. Optimal age-specific vaccination control for COVID-19: An Irish case study. *PLoS ONE* 18(9): e0290974. `https://doi.org/10.1371/journal.pone.0290974`

Collaborations

Andrew C. Parnell: As my supervisor, Professor Parnell (Maynooth University) supervised and collaborated on the work of all chapters. This includes reviewing and editing all chapters.
Subhrakanti Dey: As my supervisor, Professor Dey (Uppsala University) supervised and collaborated on the work of all chapters. This includes reviewing and editing all chapters.
Andrew Dickson: Dr. Dickson (University College Dublin) contributed to the work in Chapter 3, by providing the Additive Manufacturing context.
David Malone: Professor Malone (Maynooth University) contributed to the work in Appendix A, by working on the development of the compartmental model used. He also reviewed the manuscript.
Ken Duffy: Professor Duffy (Northeastern University) contributed to the work in Appendix A, by working on the development of the compartmental model used. He also reviewed the manuscript.

Acknowledgements

"We all change when you think about it. We're all different people all through our lives, and that's okay. You gotta keep moving so long as you remember all the people that you used to be. I will not forget one line of this, not one day, I swear."
Doctor Who

To my supervisors, Andrew and Subhra, thank you for choosing me for this position and trusting me with this project. It has been a privilege to work with you. Your vast expertise and guidance throughout the PhD have been invaluable. I will forever be grateful for your kindness, patience, motivation and support.

To my wonderful family, thank you for the love and unwavering support, and for making me the person I am today. To my mother, thank you for being my rock and my biggest cheerleader. To my father, thank you for being my inspiration and my greatest supporter. To my brother, thank you for always being there, offering love and support. Thank you to my sister-in-law for offering me encouragement and comfort, and giving me the most amazing nephew in the world, who is always able to put a smile on my face.

To my lifelong friends, thank you for being my support system. In particular, thank you to Efi for being the sister I never had, growing up with me and learning from each other about life; Evita for being there for all the good and the bad, the endless laughs and the existential conversations; Dimitra for the adventures, the support, the comfort and bringing home a little closer; Areti for always being there through thick and thin; Giannis for the inspiration and camaraderie throughout our studies and after, always making me proud and being proud of me; Christos for all the memories, the funny, the sad and the preposterous; Christina for all the laughs, the study sessions and the endless hours of playing cards and talking about life. Everything I have achieved in life would not be possible without all of you. Thank you to Marianna for being a good friend, and making the pandemic and all the hard times away from home a lot easier. To my partner Jason, thank you for the love and support through this challenging period, I am so grateful for you.

To my friends/colleagues at the Hamilton Institute, thank you for all the stimulating conversations, fun times and for being such a supportive community. The International Lunches, the Christmas Parties, the Board-game Evenings, and travelling together are all moments that I will cherish forever. It is a blessing to be surrounded by so many talented and intelligent people, but what is even more rare is being in the presence of so many fundamentally good people. Special thanks to YC for being my other half, and showing me support and friendship in and out of the office. Thank you to Amin, Niloufar and Nahia for being the best office mates anyone could ask for, always lending a listening ear and a friendly hug when nothing was going right. To my good friends Orla, Fergal, Blake, and Akash, thank you for all the laughs, the cries, the pointless and the deep conversations. Thank you to Leah, Estevão, Alessandra, Alan and Hochan for being the first people to make Ireland feel like home. Thank you to Rosemary, Kate, Gráinne and Janice for all the help and kindness. To the I-Form team, thank you for creating such a stimulating environment where people from so many different backgrounds come together and inspire each other.

Last, but certainly not least, I owe the biggest “Thank you” to all my friends at the Hamilton Institute who rushed to my aid when I needed access to more computational power, while in an incredible time crunch. YC, Fergal, Dara, Anna, Shauna, Nathan, Jack, Conor, Nahia, Blake, Niamh, Paddy, Abeer, Darshana, Victor, Aoife, Cormac, Emma, Alessandra, Daire, Sam, Kevin, Jonny, Chang, Niloufar, I am so grateful for your help! Without all of you, this thesis would not have been possible.

Summary

Additive Manufacturing (AM), commonly referred to as 3D printing, is the technology associated with manufacturing objects through computer-aided design, by building them layer-by-layer. The flexibility of AM allows for creating custom objects with very complicated geometries, utilising a wide variety of different materials. Despite its great potential, AM still faces some challenges that hinder its wider adoption. These challenges are centred around the lack of appropriate modelling, monitoring and closed-loop control in AM processes, given the large number of factors that influence the performance of an AM system. Temperature has proven to be one of the most significant aspects, making its effective control of crucial importance.

In this thesis, we address the problem of temperature control in AM. As a case study, we use the temperatures in the extruder of a Big Area Additive Manufacturing (BAAM) system. We first solve the problem in a model-based manner, assuming access to the full system state, a linear process model and a quadratic performance index. We then develop an equivalent data-driven controller that learns optimal behaviours directly from process data. Next, we present a solution accounting for no direct access to the system state. We incorporate a state estimation step and compare the data-driven algorithm trained on input-output data with a model-based, observer-based controller. In both cases, we can achieve parity with the model-based algorithms, while learning directly from data, with no knowledge of the system's dynamics.

Finally, we remove all assumptions about the process dynamics and performance index form. We solve the temperature control problem using Deep Reinforcement Learning techniques, training Deep Neural Networks to learn about the process itself as well as the optimal course of action needed to achieve the desired behaviour.

Our findings can be extended to different temperature systems in AM, or to control completely different aspects of AM, making the idea of an intelligent, self-correcting AM system a real possibility.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Background

In this thesis we focus on the design of closed-loop control algorithms for applications in Additive Manufacturing systems. Additive Manufacturing (AM), more commonly known as 3D printing, is the technology associated with creating 3D objects by building them in a layer-by-layer fashion [1]. There are many different AM technologies, involving a wide array of materials and used for different applications [2–4]. In general, AM has revolutionised the field of Manufacturing due to its ability to build very complicated structures through computer aided design. Contrary to traditional manufacturing methods, AM allows for production without the need of dedicated tooling to remove material from a work-piece, which also implies a significant reduction in the material needed. This cost-efficiency associated with AM is a significant factor driving the desire for wider adoption of these technologies. Modern applications range from automotive and aerospace, to bio-medicine, prosthetics and robotics. In addition to such industrial-scale applications, the appeal of 3D printing also comes from its ability to bring manufacturing to the hands of the consumer. In recent years, 3D printers are becoming more common in households, small businesses and even schools and academic institutions.

As AM technologies become more widespread, there is an evident need to ensure the accuracy, safety and repeatability of the process. This need can be directly translated as a need for efficient closed-loop controllers, an area that is currently under-developed within the AM space [5]. Given the numerous facets of AM processes that require monitoring and control, the active research focus lies in the creation of monitoring systems [6, 7] and efficient feedback controllers [8, 9]. Some of the most noteworthy aspects needing to be controlled are the temperature in various steps of the process, the printing speed, the geometry design, the material viscosity and flow, the layer thickness and many others. Some control problems are specific to a particular AM technology, while others may be universally applicable.

Control Theory [10,11] is the field dedicated to studying and controlling the behaviour of dynamical systems. Traditionally, control design is based on a mathematical model of the process, meaning that the system dynamics are known. Classical approaches involve studying the system through the frequency domain, expressing the system dynamics using a transfer function. Modern control theory commonly refers to the study of systems in the time domain, usually through expressing differential (in the continuous time case) or difference (in the discrete time case) equations in state space form. The most popular feedback controller is the proportional-integral-derivative (PID) controller [12] whose role is to reduce the error between the system's current and desired output, by tuning the three PID gains. Another equally popular and more versatile control design is the state feedback controller where, as the name suggests, the control action is informed by the current state.

A concept that goes hand-in-hand with state feedback controllers is the state observer, which is used to estimate the system state when it is not directly available, but the system output and a state space model of its dynamics are known. Tuning of PID controllers, state feedback controllers and state observers can be done in a number of ways, with the most popular being pole placement where the design parameters are chosen so that the resulting system has the desired poles. While these techniques provide consistently good results, they offer no guarantee of optimality.

Optimal Control theory [10, 13] is the sub-field that deals with optimisation in the context of control, where the main tool is a performance index, usually representing the process cost to be minimised. The most popular optimal controller for linear systems is the Linear Quadratic Regulator (LQR) [14] where the performance index is a quadratic function, allowing for convexity and hence good convergence of the optimisation. Depending on the problem at hand the LQR can be replaced by the Linear Quadratic Tracker (LQT) when the optimisation goal is tracking a specific reference signal, or the Linear Quadratic Gaussian (LQG) in the presence of Gaussian noise. Two of the most popular approaches to solving optimal control problems are through Pontryagin's maximum principle and the calculus of variations [15] and Dynamic Programming and the Bellman Optimality condition [13, 16]. These algorithms are more flexible and widely applicable for a range of systems, without the assumption of linearity in the system's dynamics.

In the context of AM, there has been a number of control theory applications in order to regulate various aspects of the manufacturing process. For the most part, these applications involve PID controllers. In material extrusion systems, an important part of the process that needs to be controlled is the melt temperature. In works such as [17] and [18] it is controlled using PID controllers, based on a transfer function model of the process. In [19] PID controllers were used to control the volumetric flow in a polymer extruder, addressing separately the control of the pressure and the temperature. The work in [20] presents the design of a PID controller for the material melt viscosity. In [21] the control of the cooling rate in laser-based AM is addressed through PID controllers and in [22] they are used to control the laser power. In [23] the meltpool in a Robotized Laser-based Direct Metal Addition process is controlled by means of a PID controller. Many other control applications involve non-linear, physics-based models of the process, as linearity is not always a realistic assumption, despite its simplicity and convenience for control design. Control of the layer height for Laser Metal Deposition is studied in [24] while the width and height of the molten puddle is controlled in [25] using a model of its geometry. Controlling the meltpool in Selective Laser Melting is addressed in [26], in [27] for Material Deposition and in [28] for Directed Energy Deposition.

In many physical systems, AM included, accurate models of the system are a very rare occurrence, making the modelling of AM processes a very active research field [29]. As a result, some attempts have been made at designing feedback controllers whose parameters can be tuned through information learned from process data. In [30] a PID controller for the melt temperature is tuned using a neural network. A controller for the extrusion force for Freeze-form Extrusion Fabrication is designed in [31] using a dynamic model of the process with parameters tuned using Recursive Least Squares. Iterative learning controllers are used in [32] and [33] to control the final part geometry and the meltpool characteristics respectively.

In general, lack of access to exact models of physical systems, is an overarching issue in Control Theory applications. This gave rise to the area of data-driven control [34–37], a very active research field of its own. The rise and popularisation of Machine Learning (ML) [38–40] has lead control engineers to use ML algorithms as a tool to designing intelligent control algorithms, that can be trained from process data. There is a particular category of ML algorithms that is very closely linked to Optimal Control theory and more specifically Dynamic Programming (DP) [13, 16]. Reinforcement Learning (RL) [41, 42] refers to the family of ML algorithms that learn optimal strategies through sequential decision making, within intricate and unpredictable settings by maximizing rewards (or minimizing costs). The usual setup of a reinforcement learning problem is that of an agent that acts in a certain environment, its actions affecting its state and resulting in some reward that it receives from the environment. This means that there is no supervision of the process, only a reward signal. Additionally, the actions that the agent takes directly influence its next states and hence the rewards it receives. The goal of reinforcement learning algorithms is to maximise the reward the agent receives from the environment. In the control theory literature, such decision making problems are solved using DP techniques, usually assuming knowledge of an underlying model of the process, although model-free approaches have been heavily studied, giving rise to Approximate Dynamic Programming [43]. In this setting the agent is referred to as the controller, the actions as the controls and usually instead of receiving a reward from the environment, the controller is burdened with some cost. Hence, while in RL problems the optimisation goal is to maximise the reward, in DP problems, the goal is to minimise the cost. Though the terminology is somewhat different, any RL problem can be viewed as a DP problem and vice versa. Consequently, the methodology between the two fields is mostly shared and the two communities have greatly influenced each other. Some of the most popular DP/RL algorithms are value iteration [16], relative value iteration [13], Q-learning [44,45] and the actor-critic framework [46]. A noteworthy extension to RL that has shown great potential is Deep Reinforcement Learning (DRL) [47], where Deep Learning [48, 49] is used to enhance the capabilities of RL. The most noteworthy application of DRL that caused its rise in popularity, is its success in learning how to play Atari games [50].

In recent years, as RL has gained popularity, researchers have employed it to address various issues within AM. Model-based RL has been applied on implementing corrective actions on a robot wire arm AM system [51] while in [52] a data-driven RL algorithm was trained for defect prevention, that also utilised prior knowledge about the process. The problem of scheduling multiple AM machines was solved using RL in [53] and in [54] RL was used to optimise laser power and velocity in metal AM. RL has also been used in toolpath generation [55, 56] and process monitoring through imaging [57] and acoustic emissions [58]. Traditionally, RL techniques are designed for systems with discrete state and action spaces and all the above studies are applied to such systems, or use discretisation of the state and action spaces, to obtain the desired structure. DRL algorithms for continuous control have been the subject of recent research, with the most popular developments being the Proximal Policy Optimisation (PPO) algorithm [59] and the Deep Deterministic Policy Gradient (DDPG) algorithm [60]. In the AM space, PPO has been used for optimising melt pool depth in [61] and tool path in [62]. Variations of DDPG have been used to control the motion of a Robot arm AM system [63] and a tower-crane AM system [64].

While one half of controlling AM processes is choosing appropriate control algorithms, the other half is choosing the right aspect of the process to control. Our ultimate goal is to minimise the number of faulty prototypes, while maintaining a safe and efficient process. There are many different causes of defects in AM processes [65, 66] but a primary factor is the temperature at different steps of the proces [67, 68]. This may be the temperature of the build-plate, the temperature of the chamber in the case of enclosed systems, the melting temperature inside the nozzle in extrusion systems or the laser temperature in laser-based metal AM. Temperature can influence the quality of the finished product in terms of the fusion between the layers, the porosity, and the solidification cracking [65]. This implies that there is a strong need for well designed control of the temperatures throughout the AM process.

Motivated by all the above, we dedicate our efforts in designing efficient controllers for the extrusion temperature in AM systems. In particular, we study a Big Area Additive Manufacturing (BAAM) system, due to the complexity of the heating system in its extruder, as opposed to commercial extrusion-based systems. We design controllers both in a model-based and a data-driven manner. This highlights the need for a process model, which we specifically choose to be a model in state-space form. In Optimal Control applications, state space models are very popular, however they have not been widely used in AM applications. Previous work in state space modelling in manufacturing includes the identification of a state space model and its use to design a controller for temperature regulation [69], the development of a state space model for product quality [70], and the use of a state space model for observer design of temperature states within an AM-produced part [71].

1.2 Outline of the thesis

The thesis is organised as follows:

In Chapter 2 we provide a thorough introduction to Reinforcement Learning and Dynamic Programming. We give the underlying structure of classic RL and DP problems, which are traditionally formulated as a Markov Decision Process [72], and the main tool in solving these problems, the Bellman Optimality Equation [16]. Next, we introduce some of the most noteworthy RL and DP algorithms, starting with the Relative Value Iteration, a very popular DP algorithm. We then present the Q-learning algorithm and its most notable extension, the Deep Q Network. Following is a popular family of RL algorithms, namely Actor-Critic methods. Finally, we discuss approaches to solving continuous RL problems, which is the focus of our work in the consecutive chapters.

In Chapter 3 we address the problem of controlling the temperature in the extruder of a Big Area Additive Manufacturing (BAAM) system. In particular, we consider the Linear Quadratic Tracking (LQT) framework for the control design, where we aim for the temperature to track a given reference signal, a goal quantified through a quadratic performance index. We design 3 different feedback controllers to solve this problem, two model-based and one data-driven. In the case where we consider a process model to be available, we explore both the finite horizon scenario and the infinite horizon scenario. We then use the infinite horizon control design as a basis for designing a data-driven controller, that learns directly from data using Reinforcement Learning techniques. We present our findings through simulations from a state space model of the heating system. We simulate the system's behaviour after the application of all three controllers and find that it is possible to achieve comparable performance both in the model-based and data-driven case. This is a very encouraging result, as accurate models are very rare in AM.

The ability to train effective controllers strictly from data, overcomes this limitation and paves the road towards intelligent manufacturing.

Chapter 4 presents an extension to the work done in Chapter 3. We again solve the problem of temperature control in a BAAM system but this time we have one important additional limitation. We no longer assume access to the internal system state, instead we only observe the state through sensors, in the form of measurements. This creates the need for a state estimation step, which in the case of model-based control can be performed with the help of a State Observer. In the model-free RL-based scenario, we include a state estimation step in the control design, through a record of past inputs and outputs. This significantly increases the complexity of the problem, which highlights the need for generating "good" data. As the RL-based controller learns information about the system dynamics, we require data rich in information for training. We provide a detailed discussion on ways to generate "good" data, with which we can obtain a data-driven controller that achieves parity with the model-based, Observer-based control. Finally, we address an important factor in the controller's performance, the design of the optimisation problem and more specifically the weighting parameters in the quadratic performance index. As there is no systematic way to select these parameters, we explore the application of Bayesian Optimisation and find that we can achieve improved tracking performance and a significant reduction in the overall cost. These results are another important step towards fully autonomous manufacturing processes that can learn optimal behaviours from process data with no prior knowledge of the system's dynamics.

In Chapter 5, we solve the temperature control problem while removing all assumptions about the system structure or the performance index. To that end, we employ the Deep Deterministic Policy Gradient (DDPG) algorithm [60] to train an optimal controller. DDPG is a popular Deep Reinforcement Learning algorithm combining Actor-Critic methods and Q-learning. We train two Deep Neural Networks (DNNs), one tasked with learning the optimal policy and the other with learning the value function, the tool that quantifies how "good" each policy is. This is done in a completely data-driven way and the versatility of DNNs allows for the use of complicated forms for the value function. In our application, we consider a constrained optimisation problem resulting in a value function that is no longer quadratic, which makes the methods of the previous chapters not applicable. We use the same linear state space model of the previous chapters to simulate the system's behaviour and generate data to train a DDPG agent. We were successful in learning a policy that closely tracks the reference signal, within a small error margin. This result showcases that an autonomous system trained on its own past behaviour, with no prior knowledge of its environment is a real possibility.

In Chapter 6 we offer a summary of our findings and conclude this thesis with topics for future research.

Finally, in Appendix A we present a side research project conducted during the course of this PhD. With the outbreak of the COVID-19 pandemic, there was a large focus on modelling, predicting and controlling the spread of the virus. As the vaccines started becoming available, an important problem to be solved was their optimal allocation. In our work, we introduce a new compartmental model that describes the transmission dynamics of the virus. We consider two age groups and several compartments representing different stages in the vaccination process. We then use Optimal Control techniques to obtain a vaccination control program that effectively minimised the total infections. Using data from Ireland's COVID-19 Data Hub, we simulate the evolution of infections with and without vaccination control, both in the case of strict lockdown measures and in the case where the restrictions are loosened. Our findings showcase that vaccinating both age groups simultaneously can be very beneficial, which is different to the approach taken by the state. Comparing our findings with the vaccination policy taken by the Irish government, we found that similar results would be possible through our obtained optimal vaccination strategy while using significantly fewer resources.

The methods proposed in Chapters 3, 4 and Appendix A were implemented using R and the approach taken in Chapter 5 is implemented in Python. All R and Python scripts are available on the author's Github `https://github.com/elenizavrakli`.

2 Reinforcement Learning basics

Reinforcement Learning (RL) is a Machine learning tool concerned with solving sequential optimisation problems using intelligent decision-making. This means that the goal of RL is to train an *agent* to take optimal actions in a specific environment, in order to achieve high rewards. An RL agent, much like a biological agent, learns from its past experience, namely by interacting with its environment, taking actions and receiving rewards corresponding to those actions.

The dynamics of a reinforcement learning problem can be formally described with the help of **Markov Decision Processes** [73].

Definition 1 *A Markov Decision Process (MDP) is a tuple $\langle S, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ where*

- *S is a finite set of states*
- *$\mathcal{A}$ is a finite set of actions*
- *$\mathcal{P}$ is a state transition probability matrix whose elements are of the form*

$$p_{ss'}(a) = \mathbb{P}\left[S_{t+1} = s' | S_t = s, A_t = a\right] \tag{2.1}$$

  i.e., the probability of the next state being s' given that the current state is s and the action that was chosen is a.

- *$\mathcal{R}$ is a reward function of the form*

$$r(s, a) = \mathbb{E}\left[R_t | S_t = s, A_t = a\right] \tag{2.2}$$

  i.e., the expected reward given that the current state is s and the action that was chosen is a.

It should be noted that MDPs are generally studied in the context of finite and discrete state and action spaces, as in the definition above, but they can also be generalised to continuous state and action spaces [74].

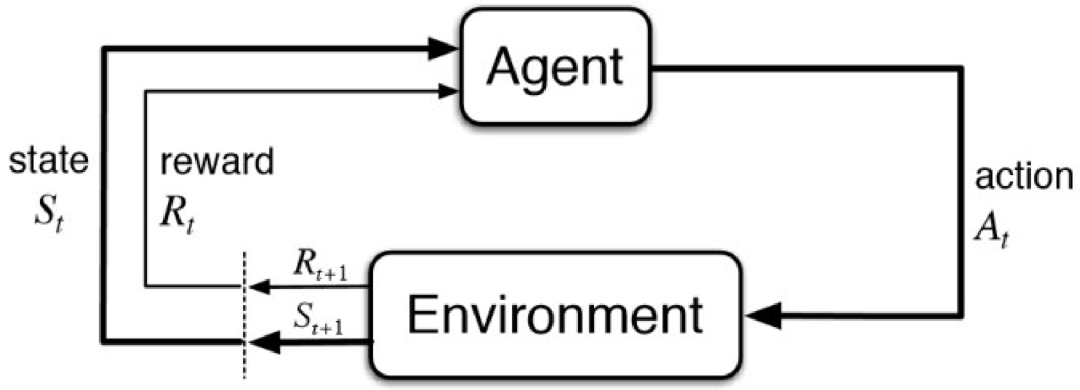


Figure 2.1: Dynamics of a Reinforcement Learning problem. At state S_t the agent receives a reward R_t and interacts with the environment by taking an action A_t.

This type of decision process is called *Markov* because we are assuming that each state includes all the necessary information from the previous steps to inform our decision, i.e., satisfies the **Markov property**. At each time step t, the agent

- observes the environment's state $S_t = s \in S$,
- selects and executes an action $A_t = a \in \mathcal{A}$,
- receives a numerical reward $R_t = r(s, a) \in \mathcal{R}$ and
- transitions to a new state $S_{t+1} = s' \in S$ with probability $p_{ss'}(a)$.

The goal of the agent is to maximise the cumulative rewards it receives from the environment. The theory of reinforcement learning is based on the *reward hypothesis.*

Definition 2 (Reward Hypothesis) *[45] All goals can be described by the maximisation of the expected cumulative reward.*

What we are hoping to accomplish in any reinforcement learning problem is to find the best action to take at each given state in order to obtain the best possible future reward. In other words, we are looking for a function of the state $\pi(s)$ that informs us of the best action to take at said state. This function is called a **policy function** and we are trying to determine the optimal one. A policy can either be deterministic ($\pi(s) = a$) or stochastic ($\pi(a|s) = \mathbb{P}[A = a|S = s]$). When our policy is independent of the time step it is called *stationary* and it is the type of policy that will be the focus of our study.

Definition 3 *A policy is called proper if the probability of being at an absorbing state converges to 1 as time converges to infinity. By absorbing state, we refer to a state that once it is reached, our agent can not transition to any other state.*

The basic tool used to determine optimal policies is **value functions**. Informally, a value function informs us about how rewarding the combination of a given state and action is. Formally, it is a prediction of future rewards if we follow a certain policy. There are two types of value functions:

→ The *state value function*: The expected reward starting from state s and then following policy π.

$$v_\pi(s) = \mathbb{E}_\pi\left[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots | S_t = s\right] \tag{2.3}$$

→ The *state-action value function*: The expected reward starting from state s, taking action a and then following policy π.

$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots | S_t = s, A_t = a\right] \tag{2.4}$$

γ is a small constant $0 < \gamma \leq 1$ known as the *discount factor*, which is used to indicate that we are more interested in immediate rewards over future rewards.

The type of value function depends on the algorithm chosen to target each problem. If the agent follows policy π and averages the returns for each state that it visits then a state value function is preferable. If on the other hand, our algorithm keeps separate averages for each action taken at each state, then a state-action value function is more suitable. An interesting feature of these two types of value functions is that one can be obtained from the other, utilising their definition with the use of the following relations:

$$\begin{aligned} v_\pi(s) &= \sum_{a\in\mathcal{A}} \pi(a|s) q_\pi(s,a) \qquad (2.5) \\ q_\pi(s,a) &= r(s,a) + \sum_{s'\in S} p_{ss'}(a) v_\pi(s') \qquad (2.6) \end{aligned}$$

Regardless of the specific problem we are studying, the key idea motivating the optimization process is the *principle of optimality*:

Definition 4 (Principle of optimality) *[16] An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*
More formally: Let $\left\{a_0^*, \ldots, a_{N-1}^*\right\}$ *be an optimal action sequence, which together with* s_0 *determines the corresponding state sequence* $\{s_1^*, \ldots, s_N^*\}$*. Consider the sub-problem whereby we start at* s_k^* *at time* k *and wish to maximise the "reward-to-go" (or minimise*

the "cost-to-go") from time k to time N,

$$r_k(s_k^*, a_k) + \sum_{m=k+1}^{N-1} r_m(s_m, a_m) + r_N(s_N)$$

over $\{a_k, \ldots, a_{N-1}\}$ with $a_m \in A_m(s_m),\ m = k, \ldots, N-1$. Then the truncated optimal action sequence $\left\{a_k^*, \ldots, a_{N-1}^*\right\}$ is optimal for this sub-problem.

A policy is considered to be optimal when it maximizes the corresponding value function, i.e., achieves the best possible reward for each state. An optimal policy is denoted by π^* and the optimal value function that is obtained by following the optimal policy can be denoted by $v^* = v_{\pi^*}$ or $q^* = q_{\pi^*}$

The principle of optimality can be expressed mathematically by the **Bellman optimality equation** [13, 45]. For the two types of value functions the respective Bellman equation will be:

$$v^*(s) = \mathbb{E}\left[R_t + \gamma v^*(S_{t+1}) | S_t = s\right] \tag{2.7}$$

$$q^*(s,a) = \mathbb{E}\left[R_t + \gamma q^*(S_{t+1}, A_{t+1}) | S_t = s, A_t = a\right] \tag{2.8}$$

In an ideal setting, we would also have access to a model describing the system, i.e., the transition probability matrix and the reward function. That is not always the case. Depending on whether there is a model available or not, the algorithm can be classified as *model-based* or *model-free.*

There are two main approaches to learning optimal policies, **Value based** and **Policy based** algorithms.

Value based methods concentrate on iteratively determining (or approximating) the optimal value function and from that extracting the optimal policy. These methods start with some initial value for each state and generate a sequence of value functions that converges to the optimal. While the iterations are taking place, there is no explicit policy. One can be derived at any step if the algorithm is stopped but there is no guarantee that it will be admissible. The optimal policy is obtained once the algorithm has converged by extracting the actions that have achieved the maximal reward for each state (or the minimal cost). Examples of value based methods are the *relative value iteration* algorithm and the *Q-learning algorithm.*

Policy based methods on the other hand start with a policy π_0 and generate a sequence of policies $\pi_1, \pi_2, \ldots$ that converge to the optimal one. Given a policy π_k, we start by evaluating it, computing $v_{\pi_k}(s)$ for each state s. In the case where the transition probabilities are known [43], $v_{\pi_k}(s)$ is evaluated using the following equation:

$$v_{\pi_k}(s) = \sum_{s'=0}^{n} p_{ss'}(\pi_k(s)) \left(r(s, \pi_k(s)) + \gamma v_{\pi_k}(s')\right) \tag{2.9}$$

where $0, \ldots, n$ denote the states of the environment. Next, we perform a *policy improvement step* by acting greedily with respect to $v_{\pi_k}(s)$. With the transition probabilities available, a new policy π_{k+1} is obtained by:

$$\pi_{k+1}(s) = \arg\max_a \sum_{s'=0}^{n} p_{ss'}(a) \left(r(s, a) + \gamma v_{\pi_k}(s')\right) \tag{2.10}$$

The process is repeated with π_{k+1} in the place of π_k until $v_{\pi_{k+1}}(s) = v_{\pi_k}(s)$ in which case π_k is the optimal policy. The most popular example of policy based methods is the *Actor-Critic* method.

2.1 Relative Value Iteration

Value iteration [13, 73] is a value-based and model-based approach coming from the field of *control theory* and is based on the *dynamic programming* algorithm for solving decision-making problems. This implies that traditionally it is formulated as a cost minimisation algorithm as opposed to a reward maximisation algorithm. The available information in this setting is:

- The possible states: $\{s_1, \ldots, s_n\}$ which for simplicity we represent as $\{1, \ldots, n\}$
- The available controls (actions) depending on the state: $a \in A(i)$
- The transition probabilities $p_{ij}(a)$
- A set of *admissible policies* of the form $\pi = \{\mu_0, \mu_1, \ldots\}$, $\mu_k(i) \in A(i)$. This means that for each state i, there is a set of admissible actions $A(i)$ and the functions μ_t choose at each time step t an action dependent on the current state.
- A cost function depending on the state and the action at each time step: $g(i, \mu(i))$. The purpose of the cost function is the same as that of the reward function as described in 2.2 with the difference that our problem is now a minimisation instead of a maximisation problem.

The value function in this setting is defined as the **average cost** per stage starting from a state i:

$$v_\pi(i) = \lim_{N\to\infty} \frac{1}{N} \mathbb{E}\left\{ \sum_{k=0}^{N-1} g(s_k, \mu_k(s_k)) | s_0 = i \right\}$$

In this setting, stage refers to the current time step/iteration. This means that at each stage, the average cost depends on the current policy. We are aiming to determine an *optimal stationary policy* $\pi^* = \{\mu, \mu, \dots\}$, independent of time, that achieves the *optimal average cost* which is denoted by λ^*. This is done iteratively by executing the *dynamic programming algorithm* starting from an arbitrary initial value for v, denoted by v_0. When our iterations converge we will have obtained the optimal average cost from which we can extract the optimal policy. This cost at each iteration step k has to satisfy the *Bellman equation* which for this problem would be:

$$v_{k+1}(i) = \min_{a\in A(i)} \left[g(i,a) + \sum_{j=1}^{n} p_{ij}(a) v_k(j) \right] \tag{2.11}$$

We are expecting v_k to converge to the optimal average cost as the iteration steps $k \to \infty$. This method is known as the **value iteration algorithm**. However, it is not uncommon for some of the components of v_k to diverge to ∞ or $-\infty$ thus complicating the calculations. This drawback is overcome by the introduction of the **relative value iteration algorithm**. We consider the vectors:

$$h_k(i) = v_k(i) - v_k(s)$$

where s is some arbitrarily chosen fixed state. By substituting equation (2.11) in the above definition we obtain:

$$\begin{aligned}
h_{k+1}(i) &= v_{k+1}(i) - v_{k+1}(s) \\
&= \min_{a\in A(i)} \left[g(i,a) + \sum_{j=1}^{n} p_{ij}(a) v_k(j) \right] - \min_{a\in A(s)} \left[g(i,a) + \sum_{j=1}^{n} p_{sj}(a) v_k(j) \right] \\
&= \min_{a\in A(i)} \left[g(i,a) + \sum_{j=1}^{n} p_{ij}(a) h_k(j) \right] - \min_{a\in A(s)} \left[g(s,a) + \sum_{j=1}^{n} p_{sj}(a) h_k(j) \right]
\end{aligned}$$

If the algorithm converges to some vector h, we have:

$$\lambda + h(i) = \min_{a\in A(i)} \left[g(i,a) + \sum_{j=1}^{n} p_{ij}(a) h_(j) \right]$$

where

$$\lambda = \min_{a\in A(s)} \left[g(s,a) + \sum_{j=1}^{n} p_{sj}(a) h_(j) \right]$$

is the optimal average cost per stage for all initial states.

The convergence of the relative value iteration is extensively studied in [13] and [75]. The following proposition provides good intuition to some conditions for convergence.

Proposition 1 *[75] Assume that there exists a positive integer m such that for every admissible policy $\pi = \{\mu_0, \mu_1, \dots\}$, there exists an $\epsilon > 0$ and a state s such that*

$$\left[P_{\mu_m} P_{\mu_{m-1}} \dots P_{\mu_1}\right]_{is} \geq \epsilon, \quad i = 1, \dots, n \tag{2.12}$$

$$\left[P_{\mu_{m-1}} P_{\mu_{m-2}} \dots P_{\mu_0}\right]_{is} \geq \epsilon, \quad i = 1, \dots, n \tag{2.13}$$

where P_{μ_j} denotes the transition probability matrix coressponding to action μ_j and $[\cdot]_{is}$ denotes the element of the ith row and the sth column of the corresponding matrix. Fix a state t and consider the relative value iteration algorithm

$$h_{k+1}(i) = v_{k+1}(i) - v_{k+1}(t), \ , i = 1, \dots, n \tag{2.14}$$

where $h_0(i)$ are arbitrary scalars. Then the sequence $\{h_k\}$ converges to a vector h satisfying

$$\min_{a \in A(t)} \left[g(t,a) + \sum_{j=1}^{n} p_{tj}(a) h_(j) \right] + h(i) = \min_{a \in A(i)} \left[g(i,a) + \sum_{j=1}^{n} p_{ij}(a) h_(j) \right]$$

so that $\lambda = \min_{a \in A(t)} \left[g(t,a) + \sum_{j=1}^{n} p_{tj}(a) h_(j) \right]$ is equal to the optimal average cost per stage for all initial states and h is an associated differential cost vector.

Conditions (2.12) and (2.13) imply that there exists a state that is reachable from every other state within a finite number of steps with nonzero probability. Algorithm 1 gives a step-by-step description of executing the relative value iteration algorithm until convergence to the optimal average cost and the optimal policy.

2.2 Q-learning

Q-learning [45] refers to a family of RL algorithms that involve the use of Q-functions, namely the state-action value function form of (2.4). The algorithm that is most commonly referred to as Q-learning or Q-value iteration is a method and was first introduced by *Watkins in 1989* [44]. Being an RL algorithm implies that it is formulated as a maximisation of rewards problem as opposed to a minimisation of costs. It can be applied to problems where the transition probabilities are not available and it is a value-based and off-policy method. By off-policy, we mean that the agent learns the value function without

```
Algorithm 1: Relative value iteration
   Result: Optimal policy $\pi$ and optimal average reward $\lambda$
   Input: Transition probabilities $p_{ij}(a)$, Set of admissible policies $A(i)$, Reward
          function $g(i,a)$, Accuracy $\epsilon$, Fixed state $s$
 1 Randomly initialise $h$ for all states;
 2 $h'(i) =$
     $\min_{a\in A(i)} \left[g(i,a) + \sum_{j=1}^{n} p_{ij}(a)h_0(j)\right] - \min_{a\in A(s)} \left[g(s,a) + \sum_{j=1}^{n} p_{sj}(a)h_0(j)\right]$
     for all states $i$;
 3 while $|h - h'| \geq \epsilon$ do
 4     $h = h'$;
 5     $h'(i) =$
         $\min_{a\in A(i)} \left[g(i,a) + \sum_{j=1}^{n} p_{ij}(a)h(j)\right] - \min_{a\in A(s)} \left[g(s,a) + \sum_{j=1}^{n} p_{sj}(a)h(j)\right]$
         for all states $i$;
 6 end
 7 $\pi(i) = \arg\min_{a\in A(i)} \left[g(i,a) + \sum_{j=1}^{n} p_{ij}(a)h'(j)\right]$ for all states $i$
   $\lambda = \min_{a\in A(s)} \left[g(s,a) + \sum_{j=1}^{n} p_{sj}(a)h(j)\right]$
```

necessarily following the same policy throughout the training. We consider a state-action value function $q_\pi(s,a)$ of action a in state s under policy π to be the expected return starting from state s, performing action a and following policy π thereafter.

$$q_\pi(s,a) = \mathbb{E}_\pi \left[R_t + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a\right]$$

The values of $q(s,a)$ for each state s_t are called the **Q-values** and are stored in the **Q-table**. The Q-table is initialized to zero or some other arbitrary value, and after enough iterations will converge to the optimum values, under certain sufficient conditions. That means that when the learning is complete, we can simply look at the values for a given state and decide on the best possible action for that state, thus deriving the optimal policy.

The agent starts exploring the environment (or playing the game) many times, studying the rewards it gets for each action and uses them to update the Q-table. The formula for these updates is:

$$Q^{new}(s_t, a_t) = (1-\alpha) \cdot Q(s_t, a_t) + \alpha \cdot \Big(r(s_t, a_t) + \gamma \cdot \max_a Q(s_{t+1}, a)\Big) \tag{2.15}$$

where

- $Q^{new}(s_t, a_t)$ is the new Q-value for the particular state-action pair,
- $Q(s_t, a_t)$ is the old Q-value,

- $\max_a Q(s_{t+1}, a)$ is an estimate of the optimal future value,
- $r(s_t, a_t)$ is the reward obtained for the particular state-action pair,
- γ is the discount factor, $0 < \gamma \leq 1$,
- α is the learning rate. The meaning of this factor is to determine how much of the old values we will retain. Generally α is chosen as $0 < \alpha < 1$. A learning rate of 0 will mean that there is nothing new to be learnt while a learning rate of 1 will mean that the old value will be completely discarded.

The Q-table continually gets updated until the agent reaches some terminal state that will cause it to stop moving. Then it will reset to the initial state and the algorithm runs again. Each repetition of the algorithm from initial to terminal state is called an **episode**. We would ideally like to let the updates run for an infinite number of episodes, as that will allow the q-values to converge to their optimals, as we will see at the end of the section. So depending on the problem we are trying to solve, we choose a number of episodes as large as possible. To make sure that no infinite loops occur, within each episode, we also introduce a maximum number of steps (updates) for the case where a terminal state is not reached.

Finally, an important notion in Q-learning is the trade-off between *exploration and exploitation*. By the term exploration, we mean the tendency our agent has to further explore the environment and by exploitation, we refer to how much our agent tends to exploit the knowledge it has already gained. Naturally, at the start of the learning process, our agent will need to explore more than exploit, since it has no prior knowledge of its environment. However that need for exploration is reduced as the learning progresses. To deal with this trade-off we introduce an *exploration factor* ϵ to control the way our agent chooses its next action. ϵ is a number between 0 and 1 decreasing after each iteration of the learning process (or episode), according to a chosen *decay factor*. The way our agent will choose an action at time t can be described by:

$$\mathrm{a}_t = \begin{cases} \arg\max_a Q(s,a), & R > \epsilon \\ \text{Random } a \in A, & R \leq \epsilon \end{cases} \tag{2.16}$$

where R is a randomly generated number between 0 and 1.

An interesting feature of the Q-learning algorithm as opposed to the relative value iteration algorithm is that it requires a specific initial state from which it can start exploring the environment. However, in order to obtain a Q-table somewhat independent of the initial state, we can sample a different initial state at the start of each episode. That will cause the Q-table to be filled in its entirety but it will cause our algorithm to take much longer to converge. This means that if we are interested in learning the best policy starting only from one specific initial state, it is better to let the algorithm run without sampling different starting states for each episode. If, on the other hand, we aim to use the results of the algorithm to obtain the optimal policy from more than one initial state, then the sampling approach will generate an appropriate Q-table.

The convergence of the Q-learning algorithm has been studied along with the original introduction of the algorithm in [44]. Further results were presented in [76], which can be summarised by the following theorem.

Theorem 1 *[76] The q-values $Q(s, a)$ converge to the optimal $q^*(s, a)$ with probability 1, for every s and a in each of the following cases:*

(1) If $\gamma < 1$.

(2) If $\gamma = 1$, the Q-table is initialised to all zeros and all policies are proper.

(3) If $\gamma = 1$, the Q-table is initialised to all zeros, the q-values are guaranteed to be bounded with probability 1 and the following assumption holds:

- *There exists at least one proper stationary policy.*
- *Every improper stationary policy yields an infinite expected cost for at least one initial state.*

In most cases, we choose a discount factor $\gamma < 1$, in which case the theorem implies that the q-values should converge to their optimal values. However, it is important to note that the Q-table is updated only when a certain state is visited and if that state is not visited, that entry is not updated. Since in practice, we are using a finite number of steps and episodes, it is possible that certain states are not visited frequently enough for them to converge to their respective optimals. Hence we might obtain some sub-optimal policy instead of the true optimal. Once again, the importance of using a large number of episodes is highlighted. Algorithm 2 provides the steps to implementing the Q-learning algorithm and obtaining the optimal Q-table.

2.2.1 Deep Q Learning

While Q-learning is a very powerful algorithm that allows for the learning of optimal policies with no prior knowledge of the system's dynamics, it has one major drawback. The algorithm is built on the Q-table, in which estimates of the Q-function values for each state-action pair are stored. When dealing with problems that have large state spaces, this table becomes very large, making it a major storage issue. Given that most modern problems are complicated, implying large dimensions, a different approach needs to be introduced. The role of the Q-table is to estimate the Q-values which essentially is an

Algorithm 2: Q-learning

```
   Result: Q-table
   Input: Number of episodes $N$, max steps per episode $m$, discount factor $\gamma$,
          learning rate $\alpha$, maximum exploration rate $\epsilon_{\max}$, maximum exploration
          rate $\epsilon_{\min}$, exploration decay rate $\delta$, termination state $s_T$
 1 Initialise Q-table to all zeros;
 2 Set exploration rate $\epsilon = \epsilon_{\max}$;
 3 for episode in 1:N do
 4     t=1;
 5     Sample an initial state $s_1$ ;
 6     while $s_t \neq s_T$ and $t < m$ do
 7         Choose $a_t$ according to (2.16);
 8         Receive reward $r(s_t, a_t)$ from environment;
 9         Sample the next state $s_{t+1}$ from environment;
10         Update Q-table value corresponding to the pair $(s_t, a_t)$ according to
             $Q^{new}(s_t, a_t) = (1-\alpha) \cdot Q(s_t, a_t) + \alpha \cdot [r(s_t, a_t) + \gamma \cdot \max_a Q(s_{t+1}, a)]$;
11         t=t+1;
12     end
13     if $\epsilon > \epsilon_{\min}$ then
14         $\epsilon = \epsilon * \delta$;
15     end
16 end
```

estimate of the value function. Considering this, the Q-table can be replaced by a function approximator, with the most popular choice being Deep Neural Networks (DNN)s [49]. This combination of Deep learning and Q-learning gave rise to the Deep Q Learning algorithm, also known as the Deep Q Network (DQN) [77].

The main idea for this algorithm is to use a Deep Neural Network (DNN) in place of the Q-table, which receives the current state as input and produces the q-values for all possible actions as outputs. The DQN is updated using the same formula as the Q-table (2.15), but there are a few adjustments that need to be made, which stem from the way neural networks operate and learn. Two main adjustments need to be made for the DQN to be successfully trained, the introduction of a **Replay Buffer** and a **Target Network**.

Replay Buffer

In each update step (2.15), we require information about the current state, the action taken, the reward received and the next state that the agent transitions to. This information can be represented as a tuple $\{s_t, a_t, r_t, s_{t+1}\}$. If we were to train the network on the data gathered at each time step, we would be training with sequential data. However, DNNs are designed to learn better when trained on independent and identically distributed (i.i.d.) data. To get as close to that as possible, we introduce a Replay Buffer, which is a memory where we store data tuples of the form $\{s_t, a_t, r_t, s_{t+1}\}$ while "playing the game". At each training step, we randomly sample a batch of data from the Buffer and use that to train the DQN. This way, the data used for training are no longer sequential and hence more appropriate for training.

Target Network

Looking at the update formula (2.15) again, we may notice that the DQN itself is needed to produce its updated value. This may cause the network to "chase its own tail" and as a result end up not learning efficiently. To remedy this, we introduce a second network, called the Target Network, which is identical to the DQN in configuration and initialisation. The target network is used to produce the target values, namely the right-hand side of equation (2.15), and its weights get updated periodically to match the weights of the DQN.

These two structures can ensure the successful training of a DQN, a step-by-step description of which is given in Algorithm 3. Notice the freedom in the initialisation of the neural networks, the loss function for training and the optimiser for backpropagation of the loss. Depending on the environment, different choices may be appropriate.

2.3 Actor-Critic algorithms

The Actor-Critic framework [42, 46, 78] refers to a family of policy-based algorithms used to learn optimal policies in environments where the dynamics are unknown. The main idea is instead of using one algorithm which learns the optimal value function and from that extracting the optimal policy, we use two algorithms that interact and learn from each other, one called the *Actor*, tasked with learning the optimal policy, and a *Critic*, tasked with learning the value function. In simple terms, the Actor decides how to act in the environment, and the Critic judges how good these actions were in order to inform the next actions the Actor chooses. Similarly to Q-learning and Deep Q Learning, Actor-Critic algorithms can be designed both with the help of lookup tables when the state space is small, or with Deep Neural Networks when the dimensions increase. Actor-critic algorithms are also very popular in the case where the policy function is not deterministic. In the following, we present the steps to training an Actor-Critic algorithm for the case where the resulting policy is deterministic.

The Actor learns a policy π mapping states to actions. The Critic learns a state value function (2.3) v_π and evaluates the new state after each action is selected by the Actor.

Algorithm 3: Deep Q Learning

```
   Result: DQN
   Input: Number of episodes $N$, Max steps per episode $m$, Discount factor $\gamma$,
          Learning rate $\alpha$, Maximum exploration rate $\epsilon_{\max}$, Maximum exploration
          rate $\epsilon_{\min}$, Exploration decay rate $\delta$, Termination state $s_T$, Batch size $bs$,
          Period for equalising the NNs $\tau$, Chosen initialisation, loss function and
          back-propagation optimiser for NNs.
 1 Initialise DQN and Target network (denoted as TN) to be identical;
 2 Set exploration rate $\epsilon = \epsilon_{\max}$;
 3 for episode in 1:N do
 4     t=1;
 5     Sample an initial state $s_1$ ;
 6     while $s_t \neq s_T$ and $t < m$ do
 7         Choose $a_t$ according to (2.16);
 8         Receive reward $r(s_t, a_t)$ from environment ;
 9         Sample the next state $s_{t+1}$ from environment;
10         Augment the replay buffer by the tuple $\{s_t, a_t, r_t, s_{t+1}\}$;
11         Sample a batch of size $bs$ from the replay buffer;
12         for i in 1:bs do
13             Extract $i$-th tuple $\{s^i, a^i, r^i, s^i_{next}\}$;
14             Set the target for the terminal node of the DQN corresponding to
                action $a^i$ to be
                $(1-\alpha) \cdot TN_{pred}(s^i, a^i) + \alpha \cdot \left[r^i + \gamma \cdot \max_a TN_{pred}(s_{next}, a)\right]$;
15             Set the targets for all other terminal nodes of the DQN equal to the
                corresponding predictions from the TN.;
16             Train the DQN by back-propagating the loss between the target values
                and the predictions from the DQN.
17         end
18         t=t+1;
19     end
20     if $\epsilon > \epsilon_{\min}$ then
21         $\epsilon = \epsilon * \delta$;
22     end
23     if episode mod $\tau == 0$ then
24         Set TN=DQN;
25     end
26 end
```

That evaluation happens through the error

$$\delta_t = r_t + \gamma v(s_{t+1}) - v(s_t) \tag{2.17}$$

where v is the current value function estimate that the critic has learned, r_t is the reward obtained by applying the selected action a_t at state s_t and transitioning to state s_{t+1} and γ is the discount factor. If this error is positive, the selection of a_t at s_t is encouraged, while if it is negative, it is discouraged. The updates to the actor happen according to

$$\pi^{new}(s_t) = (1-\alpha)\pi(s_t) + \alpha\delta_t \tag{2.18}$$

where α is the learning rate chosen for the actor. For the critic updates, let β be a chosen learning rate, then

$$v^{new}(s_t) = (1-\beta)v(s_t) + \beta[r_t + \gamma v(s_{t+1})]. \tag{2.19}$$

The learning process happens similarly to Q-learning where we "play the game" for many episodes starting from an initial state that is either predetermined by the environment or randomly chosen until a termination state is reached or the maximum number of steps is completed. Algorithm 4 provides the steps to implementing the above Actor-Critic algorithm.

Algorithm 4: Actor-Critic

```
Result: trained Actor and Critic
Input: Number of episodes $N$, max steps per episode $m$, discount factor $\gamma$,
       learning rate for actor $\alpha$, learning rate for critic $\beta$, termination state $s_T$
1  Initialise Actor ($\pi$) and Critic ($v$) to all zeros;
2  for episode in 1:N do
3      t=1;
4      Sample an initial state $s_1$ ;
5      while $s_t \neq s_T$ and $t < m$ do
6          Choose action using the actor $a_t = \pi(s_t)$;
7          Receive reward $r_t$ from environment;
8          Sample the next state $s_{t+1}$ from environment;
9          Update the actor using $\pi^{new}(s_t) = (1-\alpha)\pi(s_t) + \alpha\delta_t$;
10         Update the critic using $v^{new}(s_t) = (1-\beta)v(s_t) + \beta[r_t + \gamma v(s_{t+1})]$;
11         t=t+1;
12     end
13 end
```

2.4 Continuous Reinforcement Learning

Most Reinforcement Learning algorithms were initially designed for problems with finite and discrete state and action spaces, such as the famous Atari games application [50]. However, applying RL to solving control problems immediately creates the need for algorithms that can handle continuous state and action spaces, as discrete ones are very rare in physical systems.

As discussed at the beginning of the chapter, RL techniques are shared with the field of Dynamic Programming, where the focus is the optimisation of value functions with the use of the Bellman Equation. When the value function has a particular structure that can be exploited, then this information can be used to obtain approximation formulas for the solution. For example, if the value function is in quadratic form, then LQR [14] techniques can be used to obtain a closed form of the solution. The terms of the closed form can then be estimated using only process data, without any need for prior knowledge about the system's dynamics [79, 80]. Works like these are the inspiration behind our work in Chapters 3 and 4.

From the algorithms introduced in the sections above, the only one that could be applicable in continuous problems is the Actor-Critic framework, which belongs to the wider family of Policy Gradient algorithms [81]. The main difference between algorithms in that family, to algorithms in the Q-learning family is the approach taken to choose the best action to take at each state. In the Q-learning family, we heuristically determine the best action by directly looking at the value function output for all possible actions, and choose the one that gives the highest return. This approach by its nature does not allow for infinitely large action spaces. Policy Gradient methods on the other hand are, as the name suggests, based on the idea of using the gradient of the policy itself to "push" the policy towards the action that maximises the value function at each state.

In recent years, two very powerful algorithms have emerged that can solve RL problems with continuous state and action spaces, the Proximal Policy Optimisation [59] and the Deep Deterministic Policy Gradient [60]. The Proximal Policy Optimisation (PPO) algorithm is an improvement upon Policy Gradient algorithms which lack robustness. The main idea in PPO is to refine the current policy based on data generated by an older policy while also making sure the ratio between the two policies falls within a specific "comfortable zone". This way, we discourage drastic changes in the learned policy, leading to smoother learning. The Deep Deterministic Policy Gradient (DDPG) algorithm combines the main strengths of Deep Q-learning and Actor-Critic methods. In this setting, there are 4 networks in total, the actor, the critic and the corresponding target networks. The training happens by sampling batches from a replay buffer, so as to have data as close to being i.i.d. as possible. The weights of the actor network are updated through the policy gradient, while the critic is updated using the sampled policy from the target actor network, and the estimate of value function given by the target critic network. The DDPG algorithm inspired our work in Chapter 5.

3 Data-driven Linear Quadratic Tracking based Temperature Control of a Big Area Additive Manufacturing System

Abstract

Designing efficient closed-loop control algorithms is a key issue in Additive Manufacturing (AM), as various aspects of the AM process require continuous monitoring and regulation, with temperature being a particularly significant factor. Here we study closed-loop control of a state space temperature model with a focus on both model-based and data-driven methods. We demonstrate these approaches using a simulator of the temperature evolution in the extruder of a Big Area Additive Manufacturing system (BAAM). We perform an in-depth comparison of the performance of these methods using the simulator. We find that we can learn an effective controller using solely simulated process data. Our approach achieves parity in performance compared to model-based controllers and so lessens the need for estimating a large number of parameters of the intricate and complicated process model. We believe this result is an important step towards autonomous intelligent manufacturing.

3.1 Introduction

Additive Manufacturing (AM) is shaping up to be one of the most promising methods in manufacturing, with its applications ranging from medicine, aerospace, and architecture to most aspects of daily life [2–4]. AM has become popular partially because of its ability to handle very complex geometries and due to bringing manufacturing into the hands of the consumer. However, while AM processes are becoming more widely adopted, there is still a substantial lack of efficient closed-loop control algorithms [5]. More specifically, many aspects of the process require constant monitoring and control, some specific to a type of AM, others more universal to manufacturing systems. These aspects include temperature, printing speed, layer thickness, material viscosity and flow, and extrusion pressure, among others.

Control theory [10, 11] offers an array of methods that address the problem of controlling a system's behaviour. Most classic approaches involve designing controllers using a model of the system dynamics. This presents a very important limitation, as an exact model is not always available. The lack of knowledge of the system dynamics gave rise to the area of data-driven control [34–37]. Reinforcement Learning (RL) [41, 42] is a family of data-driven Machine Learning algorithms that address the problem of finding optimal policies within complex and uncertain environments through maximising rewards (or minimising costs). In the Control Theory literature, optimal policies are created via Dynamic Programming [13, 16, 43] with many similarities to RL but with differing terminology. Whilst some RL approaches have been developed in a deterministic fashion assuming knowledge of the system dynamics, most RL applications are data-driven and model-free.

Some previous approaches have applied control algorithms to address specific challenges in metal AM, including the design of Proportional Integral Derivative (PID) controllers for the cooling rate [21] and laser power [22] in laser-based AM, the melt pool in Selective Laser Melting [26], and layer height for Laser Metal Deposition [24]. In recent years, with RL gaining in popularity, researchers have applied it to solve various AM-related problems including improving process monitoring through imaging [57], acoustic emissions [58], and solving the machine scheduling problem [53]. Additionally, RL has been used in toolpath generation [55] and optimising melt pool depth [61]. An RL framework has also been used in a model-based application for implementing corrective actions on a robot wire arm AM system [51].

We study the problem of designing closed-loop controllers for a state space temperature model. State space models are very useful in the design of optimal controllers [10, 11] but have not been widely used in AM applications. Previous work in state space modelling in manufacturing includes the identification of a state space model and its use to design a controller for temperature regulation [69], the development of a state space model for product quality [70], and the use of a state space model for observer design of temperature states within an AM-produced part [71].

The control objective throughout our chapter can be considered as the optimisation of a quadratic performance index, namely the Linear Quadratic Tracking (LQT) problem. Our chapter is structured as follows. Section 3.2 provides a brief introduction to the family of manufacturing systems that are the subject of this work. Section 3.3 introduces the problem statement and the particular system that is used as a case study. We introduce a model describing the temperatures within the extruder head of a Big Area Additive

Manufacturing (BAAM) system, in state space form. In Section 3.4, we perform an in-depth analysis of model-based control methods. We explore the finite horizon LQT controller, and for the infinite-horizon case, we create a model-based RL controller. The model-based methods designed in this section can be applied to any state space model. In Section 3.5, we develop a model-free, data-driven controller. We use a discrete-time, continuous state and action space Q-learning algorithm. Finally, in Section 3.6, we present the setup of our simulations, followed by a detailed comparison of the performance of all the methods. Apart from simulating the system's behaviour after each controller is implemented, the system model is used for the design of all model-based controllers, and for generating the data that will be used to train the data-driven controller. We provide a thorough discussion of the results in Section 3.7.

3.2 BAAM - Big Area Additive Manufacturing heating systems

BAAM style 3D printing systems are becoming more widely utilised in the areas of large-scale polymer and composite manufacturing. This technology is a derivation of the MEX (Material Extrusion) technology prevalent in desktop sized 3D printers (Examples include Prusa i3 and Ultimaker S5). Unlike the majority of desktop MEX systems which contain one heating source (Typically a ceramic heater cartridge) and a sensor (Thermistor, Thermocouple or similar), BAAM systems contain multiple heaters at different points along the length of their heated melt zone. The primary reason for this is the substantially larger volume of material that is processed through a BAAM extruder per unit of time, which can range from 2 - 20 kg per hour, compared with desktop systems which would average 0.05 kg per hour. As polymers are almost exclusively insulative materials (although sometimes containing conductive additives such as Carbon black or metallic powders), heat transfer from the melt zone wall into the polymer itself is poor, and thus it is necessary to increase the length of time the polymer resides inside the melt zone so as to evenly melt the polymer granules. In order to reach the desired residency times whilst maintaining a high throughput of material, melt zones are staggered across a long heated bore (which can commonly range from 0.2 to 0.5 meters in length).

The benefit of a multi-zone heating system is the accurate control of the melt pool, which is typically arranged in a steadily increasing temperature so as to slowly increase polymer temperature until the optimum state is reached for extrusion. An example of this heating system is visually represented in Figure 3.1. It is therefore vital that a robust method of controlling these temperatures is used, particularly as this technology is adopted for printing of PEEK, PEI and other high-value polymers, where the cost of a print failure can be significant. Finally, to ensure that a temperature is maintained throughout the printing process, it is essential that an adaptive technique be adopted to ensure that temperature variance is minimal, even as unavoidable systems changes such as flow rate, ambient temperature, power supply etc occur during the print.

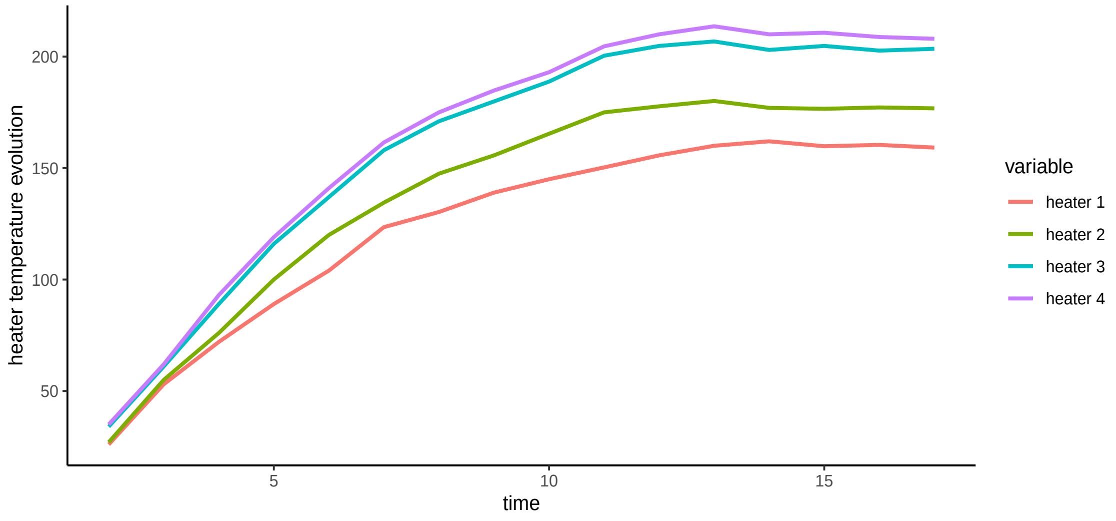


Figure 3.1: Example of heater staggering for a BAAM system using PLA, consisting of 4 heating zones to melt the material.

3.3 Linear Quadratic Tracking Control of a State-Space Temperature System

We focus on the heating and melting of the material, specifically PLA during an MEX process. To that end, we will be using a model introduced in [69]. It is a discrete-time linear model in state space form and was obtained using system identification methods using input and response data from a BAAM system. Specifically, we are dealing with a model of the temperatures within the extruder of the system. The extruder comprises five key components, namely the hopper, the screw, the barrel, the hose, and the nozzle, with thermal energy being supplied by four heaters in the barrel, one in the hose and one in the nozzle. An AC motor powers the rotation of the screw, which affects the temperature in each thermal cell as well as the adjacent cells. Figure 3.2 shows the heating system within the extruder, as well as indicates where the inputs are applied. The dynamics of the system can be expressed in standard state space form as:

$$x(t+1) = Ax(t) + Bu(t),\ t \geq t_0 \tag{3.1}$$

where $x(t) \in \mathbb{R}^n$ is the system state vector at time t and $u(t) \in \mathbb{R}^m$ is the system input at time t. $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are the state and input matrices respectively. In our case of the extruder temperature model, $x(t) \in \mathbb{R}^6$ is the set of the temperatures in each thermal cell and $u(t) \in \mathbb{R}^7$ is the input provided by the heaters and the motor for each time point t.

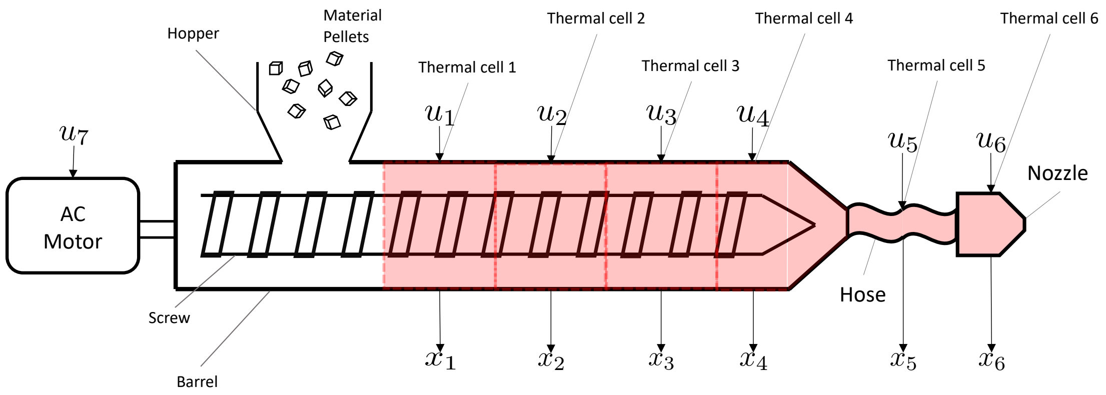


Figure 3.2: Extruder setup for BAAM system. Inputs 1 through 6 represent the heat applied to each thermal cell and u_7 represents the input from the motor. There are four heating zones inside the barrel, one on the hose and one on the nozzle. The red dotted lines denote the heating zones in the barrel. The system states represent the temperatures in each thermal cell.

The control objective is defined as the tracking of a reference signal $r(t) \in \mathbb{R}^n$ and can be expressed mathematically as in [10] through the performance index

$$V_f(x, r, u) = \sum_{t=t_0+1}^{T} \{[x(t) - r(t)]^T Q[x(t) - r(t)] + u^T(t-1)Ru(t-1)\} \tag{3.2}$$

which quantifies how well the reference signal is tracked under input u, starting from time t_0 until the end of a time horizon T. Q and R are matrices applying appropriate weights to the tracking error and the control function respectively. To ensure that the optimisation can converge, Q is required to be a positive-semidefinite matrix, while R needs to be positive definite. In certain applications, there is either no specifically defined time horizon, or it is too large. In those cases the infinite horizon performance index can be defined as in [79]

$$V_{inf}(x, r, u) = \sum_{t=t_0+1}^{\infty} \gamma^{t-t_0-1}\{[x(t) - r(t)]^T Q[x(t) - r(t)] + u^T(t)Ru(t)\} \tag{3.3}$$

where $0 < \gamma \leq 1$ is the discount factor, whose role is to emphasise short-term costs as opposed to costs in the more distant future. In the RL literature, the performance index is often called the value function.

The optimal tracking problem is defined as the search for the optimal control u^* such that the state x tracks the reference r while minimising the performance index. In the next sections, we explore solutions to the optimal tracking problem for the finite and infinite horizon cases. We consider the ideal scenario of having access to the system dynamics as well as the more realistic scenario of an unknown model. In the latter case, the problem is solved strictly using data.

3.4 Model-based Control

The main idea in model-based control is the design of a controller offline, using knowledge from the model that describes the system. The controller is then brought online, interacting with the system, using the state of the system to adjust its input in order to achieve the desired behaviour.

3.4.1 Finite Horizon

Assume the finite horizon, discrete, linear time-invariant (LTI) system defined in (3.1) and the corresponding finite horizon performance index (3.2). The solution to the optimal tracking problem, as thoroughly studied in [10], can be found as follows:

$$u^*(t) = -(B^T S(t+1)B + R)^{-1} B^T [S(t+1)Ax(t) + b(t+1) - Qr(t+1)] \tag{3.4}$$

for all $t_0 \leq t \leq T-1$, where

$$\begin{aligned} b(t) &= (A^T + KB^T)(b(t+1) - Qr(t+1)), \quad \text{with } b(T) = 0 \text{ and} && (3.5) \\ K^T &= -(B^T S(t+1)B + R)^{-1} B^T S(t+1)A && (3.6) \end{aligned}$$

and

$$\begin{aligned} S(t) &= A^T\{S(t+1) - S(t+1)B(B^T S(t+1)B + R)^{-1} B^T S(t+1)\}A + Q, \\ & \quad S(T) = Q && (3.7) \end{aligned}$$

In order to obtain a formula for $u^*(t)$ for all $t_0 \leq t \leq T-1$, given system (3.1) and performance index (3.2), Equations (3.5) and (3.7) need to be solved offline, backwards in time to determine $b(t)$ and $S(t)$ for all t. Then the control u^* can be used online, through Equation (3.4), for all t. A step-by-step description of the solution to the finite-horizon LQT problem is given in Algorithm 5.

An important prerequisite for the design of the above controller is the predefined time horizon as it is necessary for the calculation of the terms $b(t)$ and $S(t)$. However,

Algorithm 5: Finite-Horizon LQT

```
   Result: Optimal controller $u^*(t)$ and optimal trajectory $x^*(t)$
   Input: State and input matrices $A, B$, Weighting matrices for performance index
          $Q, R$, Reference trajectory $r(t)$, initial state $x(t_0)$, Time horizon $T$
 1 Initialise $b(T) = 0$ and $S(T) = Q$;
 2 for $t = T-1, t_0$ do
 3     $K^T = -(B^T S(t+1)B + R)^{-1} B^T S(t+1)A$;
 4     $b(t) = (A^T + KB^T)(b(t+1) - Qr(t+1))$;
 5     $S(t) = A^T[S(t+1) - S(t+1)B(B^T S(t+1)B + R)^{-1}\ B^T S(t+1)]A + Q$ ;
 6 end
 7 for $t = t_0, T-1$ do
 8     $u^*(t) = -(B^T S(t+1)B + R)^{-1} B^T [S(t+1)Ax(t) + b(t+1)\ - Qr(t+1)]$;
 9     $x^*(t+1) = Ax^*(t) + Bu^*(t)$ ;
10 end
```

this approach is not feasible in cases where the horizon is not explicit. This led to the development of infinite horizon approaches.

3.4.2 Infinite Horizon

The infinite Horizon Optimal Tracking Problem can be addressed in a slightly different way from the previous case. Specifically, the performance index is now of the form (3.3). Assume that the reference signal $r(t) \in \mathbb{R}^n$ is generated by

$$r(t+1) = Fr(t). \tag{3.8}$$

Consider the augmented state, including the system state and reference,

$$X(t) = \begin{bmatrix} x(t) \\ r(t) \end{bmatrix}$$

and construct the augmented system state equation

$$X(t+1) = \begin{bmatrix} x(t+1) \\ r(t+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} x(t) \\ r(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) = TX(t) + B_1 u(t). \tag{3.9}$$

The performance index (3.3) can be written in terms of the augmented state as

$$V_{inf}(X, u) = \sum_{t=k}^{\infty} \gamma^{t-k} \{X^T(t) Q_1 X(t) + u^T(t) R u(t)\} \tag{3.10}$$

where $Q_1 = \begin{bmatrix} I & -I \end{bmatrix}^T Q \begin{bmatrix} I & -I \end{bmatrix}$. The solution to the optimal tracking problem is a policy of the form [79]:

$$u(t) = -KX(t) \tag{3.11}$$

where

$$K = (R + \gamma B_1^T P B_1)^{-1} \gamma B_1^T P T \tag{3.12}$$

and P is the solution to the Algebraic Riccati Equation (ARE):

$$Q_1 - P + \gamma T^T P T - \gamma^2 T^T P B_1 (R + \gamma B_1^T P B_1)^{-1} B_1^T P T = 0. \tag{3.13}$$

An alternative to solving the ARE can be obtained with through Reinforcement Learning (RL). The Lemma proven in [79,80], states that the value function (3.10) can be written in quadratic form

$$V(x, r, u) = V(X) = \frac{1}{2} X^T(t) P X(t) \tag{3.14}$$

for some matrix $P = P^T > 0$, any stabilising policy (3.11) and reference signal (3.8). This form of the value function gives rise to the LQT Bellman equation

$$\begin{aligned} X^T(t)PX(t) &= X^T(t)Q_1X(t) + u^T(t)Ru(t) \\ &\quad + \gamma X^T(t+1)PX(t+1). \end{aligned} \tag{3.15}$$

Consider an arbitrarily chosen stabilizing policy K^* in (3.15). Then (3.15) can be transormed into the Lyapunov equation

$$P = Q_1 + (K^*)^T R K^* + \gamma (T + B_1 K^*)^T P (T + B_1 K^*). \tag{3.16}$$

Iteratively solving the Lyapunov equation while updating the optimal control estimate can effectively lead to the solution to the Optimal Tracking Problem. This result was shown in [82] where P^j was proven to converge to the optimal solution of the LQT problem, and K^J was shown to be stabilising at each step j.

A step-by-step description of the solution to the infinite-horizon problem can be found in Algorithm 6.

3.5 Data-driven Q-Learning Control

In the previous sections, the Linear Quadratic Tracking problem was thoroughly studied in the case where a model of the system of interest is available. However, this is not the case in most applications. In most instances, a model is either entirely unknown or not accurate enough to be considered ground truth. This is the motivation behind the constantly

Algorithm 6: Infinite-Horizon LQT with RL

```
   Result: Optimal control gain $K^*$, optimal controller $u^*(t)$ and optimal
           trajectory $x^*(t)$
   Input: State and input matrices $A, B$, Weighting matrices for performance index
          $Q, R$, Initial state $x(t_0)$, Initial reference signal $r(t_0)$, Discount factor $\gamma$,
          Initial control policy $K^0$, Error threshold $\epsilon$, Simulation length $N$
 1 Create augmented system matrices $T = \begin{bmatrix} A & 0 \\ 0 & F \end{bmatrix}, B_1 = \begin{bmatrix} B \\ 0 \end{bmatrix}$;
 2 Augment weighting matrix $Q_1 = \begin{bmatrix} I & -I \end{bmatrix}^T Q \begin{bmatrix} I & -I \end{bmatrix}$;
 3 Set $K^{new} = K^0$ and randomly initialise $K$;
 4 while $\|K - K^{new}\| > \epsilon$ do
 5     Set $K = K^{new}$. ;
 6     Solve the LQT Lyapunov equation
         $P = Q_1 + K^T R K + \gamma(T - B_1 K)^T P (T - B_1 K)$;
 7     Compute the control gain estimate $K^{new} = (R + \gamma B_1^T P B_1)^{-1} \gamma B_1^T P T$;
 8 end
 9 Set $K^* = K$;
10 Design initial augmented state $X(t_0) = \begin{bmatrix} x(t_0) \\ r(t_0) \end{bmatrix}$;
11 for $t = t_0, N$ do
12     $u^*(t) = -K^* X(t)$;
13     $X^*(t+1) = T X^*(t) + B_1 u^*(t)$;
14     Retrieve optimal state $x^*(t)$;
15 end
```

developing field of data-driven control. In this section, we explore Reinforcement Learning based algorithms, specifically ones in the Q-learning [44] family, which are named after the use of Q-functions as value functions.

We explore an online Q-learning solution to the LQT problem that is dependent solely on acquired data of the state trajectories [79]. The data is used to estimate the solution to the algebraic Riccati equation (3.13) following the analysis described below.

Let our system be described by the augmented state equation of the previous section

$$X(t+1) = TX(t) + B_1 u(t) \tag{3.17}$$

and let the weighting matrices Q_1 and R be defined as in (3.10). The Q-function is defined with the help of the Bellman equation as the sum of the current cost and the discounted future cost.

$$Q(x, r, u) = \frac{1}{2} X^T(t) Q_1 X(t) + \frac{1}{2} u^T(t) R u(t) + \frac{1}{2} \gamma X^T(t+1) P X(t+1) \tag{3.18}$$

where P is the solution to (3.13). Using the system dynamics, the Q-function can be written as:

$$\begin{aligned} Q(X,u) &= \frac{1}{2}X^T(t)Q_1X(t) + \frac{1}{2}u^T(t)Ru(t) && (3.19)\\ &+\frac{1}{2}\gamma(TX(t)+B_1u(t))^TP(TX(t)+B_1u(t)) && (3.20)\\ &= \frac{1}{2}\begin{bmatrix}X(t)\\u(t)\end{bmatrix}^T\begin{bmatrix}Q_1+\gamma T^TPT & \gamma T^TPB_1\\ \gamma B_1^TPT & R+\gamma B_1^TPB_1\end{bmatrix}\begin{bmatrix}X(t)\\u(t)\end{bmatrix} && (3.21)\\ &= \frac{1}{2}\begin{bmatrix}X(t)\\u(t)\end{bmatrix}^T H\begin{bmatrix}X(t)\\u(t)\end{bmatrix} && (3.22)\end{aligned}$$

where H is the symmetric kernel matrix. H can be written in block form as:

$$H = \begin{bmatrix}H_{XX} & H_{Xu}\\ H_{uX} & H_{uu}\end{bmatrix} \tag{3.23}$$

Due to the quadratic form of the Q-function, its minimisation can be reduced to the solution of $\frac{\partial Q(X(t),u(t))}{\partial u(t)} = 0$. This yields

$$u(t) = -H_{uu}^{-1}H_{uX}X(t) = -(R+\gamma B_1^TPB_1)^{-1}\gamma B_1^TPTX(t) \tag{3.24}$$

which is the same result as in equations (3.11) and (3.12). It is evident that the key to obtaining the optimal control policy in this formulation is the kernel matrix H. This matrix is what will be estimated through the Q-learning algorithm.

Setting $Z(t) = \begin{bmatrix}X(t)\\u(t)\end{bmatrix}$, the Q-function becomes

$$Q(Z) = \frac{1}{2}Z^T(t)HZ(t) \tag{3.25}$$

and consequently, (3.18) can be written as

$$Z^T(t)HZ(t) = X^T(t)Q_1X(t) + u^T(t)Ru(t) + Z^T(t+1)HZ(t+1). \tag{3.26}$$

In the above expression, the only element that requires knowledge of the system dynamics is H. However, it can be approximated using obtained data, namely a set of state trajectories with the corresponding references and inputs (x,r,u). The weighting matrices Q_1, R are predefined by the design of the optimisation problem. In order to estimate H, it needs to be isolated, meaning that the quadratic form of (3.26) needs to be transformed into an equivalent form. We start by vectorising (3.26) and then use the "vector trick" associated with the Kronecker product.

$$\begin{aligned}
vec\left(Z^T(t)HZ(t)\right) &= X^T(t)Q_1X(t) + u^T(t)Ru(t) \\
&\quad +\gamma vec\left(Z^T(t+1)HZ(t+1)\right) && (3.27) \\
\left(Z^T(t)\otimes Z^T(t)\right)vec(H) &= X^T(t)Q_1X(t) + u^T(t)Ru(t) \\
&\quad +\gamma\left(Z^T(t+1)\otimes Z^T(t+1)\right)vec(H) && (3.28)
\end{aligned}$$

where the vec operator stacks the columns of a matrix to create a single column.

Equation (3.28) can be iteratively solved through the *Value Iteration* algorithm [42] which consists of two steps:

(1) **Policy Evaluation**

$$\begin{aligned}
\left(Z^T(t)\otimes Z^T(t)\right)vec(H^{i+1}) &= X^T(t)Q_1X(t) + u^T(t)Ru(t) && (3.29) \\
&\quad +\gamma\left(\hat{Z}^T(t+1)\otimes \hat{Z}^T(t+1)\right)vec(H^i)
\end{aligned}$$

where $\hat{Z}(t+1) = \left\{X(t+1), u^i(t+1)\right\}$

(2) **Policy Improvement**

$$u^{i+1}(t) = -(H_{uu}^{i+1})^{-1}H_{uX}^{i+1}X(t) \qquad (3.30)$$

The Policy evaluation step can be solved through the Least Squares algorithm using measured data $Z(t), Z(t+1)$ and calculating the cost term $X^T(t)Q_1X(t) + u^T(t)Ru(t)$ for each data point. Matrix H is an $(2n+m)\times(2n+m)$ symmetric matrix which means that its determination is a problem with $(2n+m)\times(2n+m+1)/2$ degrees of freedom. Hence, at least $(2n+m)\times(2n+m+1)/2$ data points are required to solve (3.29). It is worth noting that in practice, many more data points are usually needed, especially when dealing with larger state and action spaces. Another important note is the need to include a regularisation step in the Least Squares algorithm. When the data are highly correlated, the inversion step becomes numerically unstable. This problem can be reduced with the choice of an appropriate regularisation parameter μ. A step-by-step description of the solution to the LQT problem using Q-learning for the state feedback case can be found in Algorithm 7.

3.6 Simulations and Numerical Results

The model used throughout the simulations is identified in [69]. In particular, the system matrices $A \in \mathbb{R}^{6\times 6}$ and $B \in \mathbb{R}^{6\times 7}$ were determined through system identification ap-

Algorithm 7: State Feedback Q-Learning

```
Result: Kernel matrix $H$, optimal controller $u^*(t)$ and optimal trajectory $x^*(t)$
Input: Weighting matrices for performance index $Q_1, R$, regularisation
       parameter $\mu$, Discount factor $\gamma$, initial kernel matrix estimate $H^0$, Error
       threshold $\epsilon$
1  Collect $N >> (2n+m) \times (2n+m+1)/2$ consecutive data tuples
2  $Z(t) = \{X(t), u(t)\}$ where $X(t) = \{x(t), r(t)\}$;
3  For each $Z(t)$, calculate the Kronecker product $K_Z(t) := Z^T(t) \otimes Z^T(t)$;
4  Set $H^{new} = H^0$ and randomly initialise $H$;
5  while $\|H - H^{new}\| > \epsilon$ do
6      Set $H = H^{new}$ ;
7      for $t = 1, N$ do
8          Determine $u^i(t+1) = -(H_{uu})^{-1} H_{uX} X(t+1)$;
9          Create the updated augmented state $\hat{Z}(t+1) = \left\{X(t+1), u^i(t+1)\right\}$ and
            the corresponding Kronecker product $K_{\hat{Z}}(t+1) = \hat{Z}^T(t+1) \otimes \hat{Z}^T(t+1)$;
10         Compute the future expected cost term $c(t) = X^T(t)Q_1X(t)$
            $+u^T(t)Ru(t) + \gamma K_{\hat{Z}}(t+1)vec(H)$;
11     end
12     Calculate the sums $L = \sum_1^N K_Z^T(t)K_Z(t)$ and $S = \sum_1^N K_Z^T(t)c(t)$ ;
13     Regularise matrix L: $L_r = L + \mu I$;
14     Produce new H estimate $H^{new} = L_r^{-1} S$;
15 end
```

proaches using data from a real BAAM system. The system is comprised of a FANUC S-420iF industrial robot arm, equipped with a thermoplastic extrusion machine and a heated industrial hose. Two system identification techniques are discussed, namely the Output-error Parametric Model Estimation method and the subspace identification method. The first method is used to identify the temperature model, namely the linear state space model describing the effect that the heater signals and the motor power have on the temperatures in each thermal zone, as represented in (3.1). The Output-error Parametric Model Estimation technique comprises 4 main steps, The parameterisation of the model, the formulation of the estimation objective as an optimisation problem, the solution of the optimisation problem using numerical methods, and finally, the evaluation of the accuracy of the obtained model. The name of this approach implies the formulation of the optimisation problem to estimate the model parameters. The optimal parameter vector minimises the expected squared norm of output error between the actual system output and the output of the parameterised model [83]. Following these steps, the system matrices were found to be:

$$A = \begin{bmatrix} 0.992 & 0.0018 & 0 & 0 & 0 & 0 \\ 0.0023 & 0.9919 & 0.0043 & 0 & 0 & 0 \\ 0 & -0.0042 & 1.0009 & 0.0024 & 0 & 0 \\ 0 & 0 & 0.0013 & 0.9979 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9972 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9953 \end{bmatrix} \tag{3.31}$$

$$B = \begin{bmatrix} 1.0033 & 0 & 0 & 0 & 0 & 0 & -0.2175 \\ 0 & 1.0460 & 0 & 0 & 0 & 0 & -0.0788 \\ 0 & 0 & 1.0326 & 0 & 0 & 0 & -0.0020 \\ 0 & 0 & 0 & 0.4798 & 0 & 0 & -0.0669 \\ 0 & 0 & 0 & 0 & 0.8882 & 0 & 0.1273 \\ 0 & 0 & 0 & 0 & 0 & 1.1699 & -0.1792 \end{bmatrix} \tag{3.32}$$

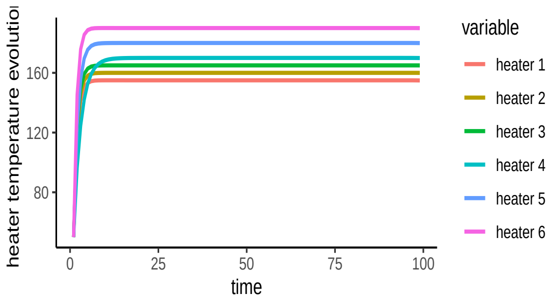
This model will be used for two main purposes. Firstly, it will be the basis around which the model-based control algorithms will be built. Secondly, it will be used throughout the simulation steps, generating data with which to train the data-driven algorithms.

It must be noted that the use of linear state space models to represent temperature evolution comes with certain limiting considerations. In general, using models that are simpler in form to describe complex dynamics inevitably means that many factors are ignored, or assumed to be constant. In particular, given that linearisations simplify the dynamics of a non-linear system at equilibrium, the accuracy greatly depends on the point around which the linearisation takes place. This implies that the system dynamics over a large range of temperatures may not be accurately described through a linear model. However, these considerations are beyond the scope of this thesis, and for the purpose of our analyses we will assume the accurate representation of the extruder heating system by the linear state space system (3.1).

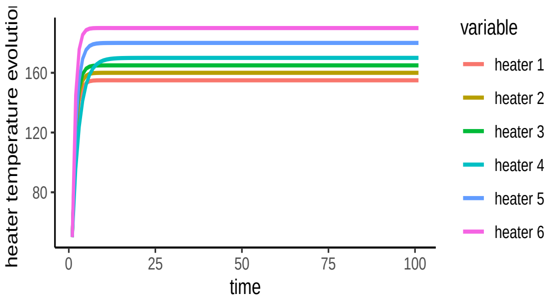
We set up an optimisation problem to be solved with all methods mentioned in the above sections. The goal of the optimisation is to control the behaviour of the system introduced in Section 3.3 so as to maintain the temperatures of all system states at some predefined set of increasing values, similar to Figure 3.1. We chose the reference trajectory to be the vector of increasing values: $r(t) = (155, 160, 165, 170, 180, 190) \in \mathbb{R}^6$ for all time points t, so that the last two heaters bring PLA within its melting temperature range. This also implies that the reference generator matrix $F = I_6$.

Starting with the model-based methods, we initially focus on the finite-horizon LQT problem. The parameters that need to be determined for the implementation of Algorithm 5 are the weighting matrices Q and R, the initial state $x(t_0)$ and the time horizon T. For Q, R we chose identity matrices of appropriate dimensions, namely $Q = I_6$ and $R = I_7$. The initial state was arbitrarily chosen to be $x(t_0) = (50, \ldots, 50) \in \mathbb{R}^6$ and the time horizon was chosen to be $T = 100$. Being the most reliable and efficient controller, implementation of Algorithm 5 results in an ideal system behaviour, where the optimal is achieved and maintained within very few steps. Specifically, all 6 heaters achieve temperatures within less than 0.09°C of the optimal temperature vector, within 16 steps. The values that the heater temperatures converge to are $x^* = (154.918, 159.985, 164.992, 169.922, 179.926, 189.955)$. This result can be seen in Figure 3.3a.

Next, we explore the case of the infinite time horizon, while still assuming full knowledge of the system dynamics, making use of Algorithm 6. The weighting matrices for the performance index Q, R and the initial state $x(t_0)$ are chosen to be the same as the finite horizon case. There are a few additional parameters that need to be determined. The discount factor is chosen as $\gamma = 0.99$, and the initial control policy K^0 is chosen as a 7×12 matrix populated by normal random numbers with mean 0 and standard deviation 10. Finally, the error threshold needed for the estimation of the control policy K is chosen to be $\epsilon = 0.1$. The RL-based controller is successful in bringing and maintaining all temperatures within 0.04 degrees of the optimal and more specifically, the temperature vector converges to the vector $x^* = (154.972, 159.996, 164.966, 169.989, 179.997, 189.955)$. 17 time steps are needed for the heaters to approach within 0.1 °C of the optimal and finally settle to their final values after 36 steps. This result is visualised in Figure 3.3b where we plot the state trajectories for the first 100 time steps.



(a) Finite Horizon LQT



(b) Infinite Horizon LQT

Figure 3.3: Model based control. Both algorithms successfully bring and maintain the heater temperatures within a small margin of the optimal values. In the finite horizon case, the convergence is achieved more quickly, while in the infinite horizon case, more time steps are required before the temperatures converge to the optimal.

Table 3.1 summarises all the parameters used in the model-based approaches to produce Figure 3.3.

Finally, we focus on the Q-learning algorithm introduced in Section 3.5. As this is a

Table of parameters - Model based methods		
Parameter	Symbol	Value
Universal parameters		
Tracking reference	r	$(155, 160, 165, 170, 180, 190)$
Initial state	$x(t_0)$	$(50, \ldots, 50)$
Tracking error weighting matrix	Q	I_6
Input weighting matrix	R	I_7
Finite Horizon LQT		
Time horizon	T	100
Infinite Horizon LQT		
Discount factor	γ	0.99
Error threshold	ϵ	0.1

Table 3.1: Parameters chosen for model-based control implementations. The information listed includes the parameter names, the corresponding symbol and the value that was chosen, grouped by the algorithms in which they were used.

learning algorithm, there is no access to the model of the system for the design of the controller, i.e., matrices A, B are unknown. Instead, the model introduced in Section 3.3 is used only as a data generator. In a real application of this learning approach, the data will be made available by sensorising the process itself, meaning that the steps regarding the data generation will not be necessary.

The simulated data needs to be varied enough so that it holds sufficient information about the system. This can be achieved by applying an input that is persistently exciting [35, 84], which can be achieved by the inclusion of probing noise. For our application, we consider the following random vectors

$$\omega_1 \sim \mathcal{N}(0, \sqrt{\sigma} I_7), \omega_2, \ldots \omega_8 \sim \mathcal{N}(0, I_7) \tag{3.33}$$

where $\sigma = 0.5$. We design the probing noise vector for time t by summing sinusoidal signals of various frequencies and periods, as follows:

$$\begin{aligned} \omega_{pr}(t) \quad = \quad & \omega_1 + 10 \sin\left(\frac{\omega_2 \pi t}{5}\right) + 8 \sin\left(\frac{2\omega_3 \pi t}{5}\right) + 7 \sin\left(\frac{3\omega_4 \pi t}{5}\right) \\ & + 6 \sin\left(\frac{4\omega_5 \pi t}{5}\right) + 4 \sin\left(\frac{5\omega_6 \pi t}{5}\right) + 3 \sin\left(\frac{6\omega_7 \pi t}{5}\right) \\ & + 0.5 \sin\left(\frac{7\omega_8 \pi t}{5}\right) \end{aligned} \tag{3.34}$$

It is important that the process remains stable during the simulations so that the states do not end up diverging towards infinity. This means that the input, apart from being persistently exciting, also needs to be stabilising. To that end, we use the following stabilising gain:

$$K = \begin{bmatrix} 0.7395 & -0.0076 & -0.0003 & -0.0264 & 0.0194 & -0.0170 \\ -0.0076 & 0.7430 & 0.0031 & -0.0093 & 0.0068 & -0.0060 \\ -0.0003 & -0.0033 & 0.7599 & 0.0021 & 0.0002 & -0.0002 \\ -0.0126 & -0.0042 & 0.0016 & 1.0971 & 0.0092 & -0.0079 \\ 0.0171 & 0.0058 & 0.0002 & 0.0170 & 0.8179 & 0.0108 \\ -0.0198 & -0.0067 & -0.0002 & -0.0193 & 0.0143 & 0.6823 \\ -0.1525 & -0.0519 & -0.0018 & -0.1412 & 0.1091 & -0.0977 \end{bmatrix} \tag{3.35}$$

In a real-life application, a stabilising gain would not need to be independently determined. The process would be tuned from the manufacturer to avoid divergence that would cause damage to the equipment. In our case, we can use the Matlab Control System Toolbox, along with the system representation (3.31), to obtain a stabilising gain. In particular, the toolbox offers two of the most popular approaches to stabilising a system's behaviour, namely the Pole Placement method and the Linear Quadratic Regulator (LQR). To obtain the gain (3.35), we used the LQR approach.

Considering all the above, the input to the system for time t can be defined as

$$u(t) = -Kx(t) + \omega_{pr}(t) \tag{3.36}$$

where ω_{pr} is the probing noise vector, as defined in (3.34). Starting with the same state initialisation as the previous cases, namely $x(t_0) = (50, \dots, 50)$ and initialising the input function as $u(t_0) = \omega_1$, we produce a set of 2000 data points that will be used for training.

In order to train the Q-learning based controller, a few more parameters need to be determined. Firstly, the weighting parameters of the value function are chosen as above, $Q = I_6$ and $R = I_7$. The discount factor was chosen to be $\gamma = 0.99$ and the regularisation parameter $\mu = 0.001$. Finally, the initial estimate of the kernel matrix is chosen to be $H^0 = I_{19}$. We train the algorithm until the error between the estimates of H becomes smaller than 0.001 or for a maximum of 30 iterations. Table 3.2 summarises all parameter values used for the implementation of Q-learning for the state feedback problem.

Figure 3.4 displays the state trajectories that result by applying to the system the controller learned with the use of Algorithm 7. The trajectories are plotted over 100 time steps. The states converge to the vector $x^* = (154.984, 159.998, 165.001169.994, 179.991, 189.991)$ which means they achieve temperatures within 0.02°C of the optimal. The state trajectories arrive within that error margin of the optimal in 20 time steps and convergence to their final values is completed within 36 steps. The results have a slight deviation from the optimal results produced by the

Table of parameters - Q-Learning		
Parameter	Symbol	Value
Tracking reference	r	$(155, 160, 165, 170, 180, 190)$
Initial state	$x(t_0)$	$(50, \dots, 50)$
Tracking error weighting matrix	Q	I_6
Input weighting matrix	R	I_7
Discount factor	γ	0.99
Regularisation parameter	μ	0.0001
Initial kernel matrix estimate	H^0	I_{19}

Table 3.2: Parameters chosen for the implementation of Q-learning. The information listed includes the parameter names, the corresponding symbol and the value that was chosen.

model-based methods but are close enough that plots 3.3b and 3.4 seem almost identical to the naked eye. This implies that the Q-learning algorithm converges to some sub-optimal kernel matrix H (and corresponding gain K). The Frobenious norm of the absolute difference between the optimal gain K^* as determined by Algorithm 6 and the estimated gain $\hat{K} = H_{uu}^{-1} H_{uX}$ from Algorithm 7 is 0.753. The mean absolute error of the individual elements populating the gain matrix is 0.0803.

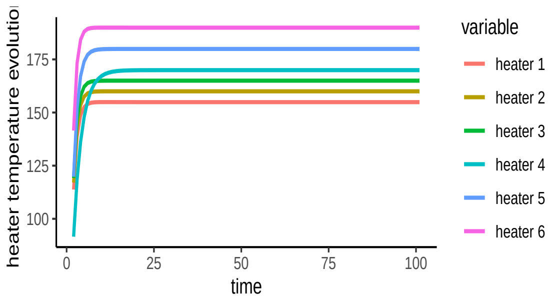


Figure 3.4: State Feedback Q-learning. The learning algorithm obtains a control gain that is sub-optimal but successfully brings and maintains all temperatures within 0.11 degrees of the reference.

A different choice in the discount factor γ would result in a different tracking performance. For example, choosing $\gamma = 0.95$ would cause the state trajectories to arrive and remain within 0.07°C of the optimal, and choosing $\gamma = 0.9$ would bring the temperatures within 0.14°C of the reference trajectory.

Another way to compare the performance of the Q-learning data-driven controller with its model-based counterpart is to look into the minimisation of the performance index (3.10). This is a particularly important measure of performance, considering that the data-driven controller seemingly performs better at tracking the optimal vector. However, for both algorithms, the optimisation goal is the minimisation of the infinite sum of discounted costs, which includes both the tracking cost and the cost associated with the input. Specifically, the model-based algorithm after 500 steps achieves a cumulative cost of $153,367$ while the data-driven method reaches the value of $153,743$. In around 900 steps both algorithms have achieved their final performance index which is $153,368$ and $153,745$ for the model-based and model-free cases respectively. Figure 3.5 shows the two cumulative costs over the first 500 time steps.

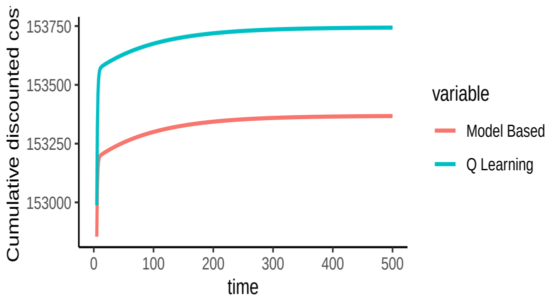


Figure 3.5: Comparison of the cumulative costs between the model-based and model-free algorithms. When the system dynamics are known, the performance index is more effectively minimised.

It is worth noting that some of the numerical results might slightly vary between different implementations, as they are dependent on the randomness introduced in the algorithm.

3.7 Conclusion

We have studied the problem of controlling a state space temperature system of material extrusion AM. Specifically, the system at hand was the heating system within the extruder of a BAAM system. The control objective was set to be bringing and maintaining the temperatures to some predetermined set of values. The problem was approached through different lenses, namely classic Control Theory techniques as well as Reinforcement Learning approaches. Initially, the focus was on model-based methods, exploring both the finite and infinite horizon cases, the first approached through the standard LQT algorithm, the second through Reinforcement Learning. Finally we explored the case of data-driven, model-free state feedback.

We showed that, in the case when an exact model of the system is available, the design of controllers is a deterministic process as it stems directly from the system dynamics. Both the finite and infinite horizon LQT controllers successfully achieve the optimisation goal, with a small error margin and within very few steps.

Given that obtaining exact models of physical processes is usually not possible, with that being evident also in Additive Manufacturing, the field of Data-driven Control is being very actively researched and developed. We approached the problem of optimising the temperatures within the extruder without knowledge of its exact dynamics, using Reinforcement Learning techniques and specifically Q-learning. The assumption is made that the entire state of the system is available for measurement. With an appropriate choice of parameters and a large training data set, the obtained controller was able to achieve the desired behaviour in the system within a few time steps. This is a very encouraging result since it implies that many systems in general and 3D printers in particular, can learn to optimise their behaviour simply by gathering data, without the need for complicated models. The more advanced data-driven controllers become, the closer the idea of truly Intelligent Manufacturing gets.

The results in this chapter are evidence to the fact that Reinforcement Learning, and more specifically Q-learning, can produce comparable results to model-based methods when controlling a state-space system. However, a few drawbacks of the approaches are highlighted. Firstly, the assumption that the entire state is exactly measurable is unrealistic. In real-life applications, measurements are obtained through sensors, which can produce errors or measure multiple states at once. Hence, output feedback approaches need to be investigated. Secondly, the training of the Q-learning algorithm requires a large amount of data which can be an issue, especially when dealing with more complicated systems. Machine learning approaches that are good at handling big data need to be explored, such as Deep Learning. These issues remain as future work.

4 Output Feedback Reinforcement Learning with Parameter Optimisation for Temperature Control in a Material Extrusion Additive Manufacturing system

Abstract

With the rapid development of Additive Manufacturing (AM) comes an urgent need for advanced monitoring and control of the process. Many aspects of the AM process play a significant role in the efficiency, accuracy and repeatability of the process, with temperature regulation being one of the most important ones. In this work, we solve the problem of optimal tracking control for a state space temperature model of a Big Area Additive Manufacturing (BAAM) system. In particular, we address the problem of designing a Linear Quadratic Tracking (LQT) controller when access to the exact system state is not possible, except in the form of measurements. We initially solve the problem with a model-based approach based on reinforcement learning concepts, with state estimation through an observer. We then design a model-free reinforcement-learning based controller with an internal state estimation step and demonstrate its performance through a simulator of the systems' behaviour. Our results showcase the possibility of achieving comparable results while learning optimal policies directly from process data, without the need for an accurate, intricate model of the process. We consider this outcome to be a significant stride towards autonomous intelligent manufacturing.

4.1 Introduction

Additive Manufacturing (AM) is becoming one of the most popular manufacturing techniques, with its appeal lying in the ability to build very complicated structures without the need for a mould and without the waste of a lot of materials. There is an array of different AM techniques, using different materials, and targeting different problems.

For example, on the industrial scale, metal-based AM is being adopted at an increasing scale due to its cost-effective and energy-saving nature [85–87]. Material extrusion (MEX) on the other hand, a technique using different polymers such as PLA, allows consumers to design and manufacture their own prototypes at home, making it more appealing to hobbyists and small business owners, neither of which would normally have access to traditional manufacturing techniques and equipment. While many different aspects of AM processes need to be monitored and controlled [6–8], efficient closed-loop control algorithms are still lacking in the AM space [5].

Control Theory techniques [10, 11], specifically proportional integral derivative (PID) controllers have been widely used to control various aspects of AM processes [19, 21, 22, 24, 26]. However, the lack of accurate models of the process is very challenging has led to the development of controllers tuned through information learned from process data. Reinforcement Learning (RL) [41, 42], and its Control Theory equivalent Dynamic Programming [13,16,43], offer an array of techniques for learning optimal strategies within intricate and unpredictable settings by maximizing rewards (or minimizing costs). In recent years, as RL has gained popularity, researchers have employed it to address various issues within AM [51, 53, 55–58, 61]. However, none of these papers use RL as a tool to design optimal feedback controllers. For more references we refer to our previous work [88], where we investigated the problem of temperature control in an AM system, when the exact system state is directly available.

In this chapter, we address the problem of temperature control for a MEX AM system without access to the exact system state, but with available sensor measurements only. Specifically, we control the temperatures within the extruder of a Big Area Additive Manufacturing (BAAM) system. We design a controller that solves the Linear Quadratic Tracking (LQT) problem, derived directly from the process model. However, we ultimately aim to have a controller that can learn optimal policies directly from process data. We utilise Reinforcement Learning techniques to design a data-driven LQT controller [79]. In the model-based case, the state can be estimated with the help of an observer [89]. In the case of the data-driven controller, we incorporate a data-driven state-estimation step in the controller, using a record of past inputs and outputs from the system only, inspired by the work done in [90]. Furthermore, we optimise the parameters used to define the LQT problem using Bayesian Optimisation [91, 92]. As there is no systematic way to choose these parameters, Bayesian Optimisation has been explored in the literature due to its ability to handle optimising complex objectives when trial-and-error approaches do not suffice [93–95]. Finally, while setting up our simulations, we provide detailed guidelines on how to generate appropriate data to train the data-driven controller. These data generation tools can be useful in many settings, in particular when one needs to train a learning algorithm for dynamical systems.

The structure of our chapter is as follows: In Section 4.2, we introduce the BAAM system we wish to control and define the optimisation problem to be solved. Section 4.3 deals with the design of the model-based state observer and controller, while in Section 4.4 we present the design of the RL-based data-driven controller which includes state estimation through input-output data. Section 4.5 introduces the main concepts of Bayesian Optimisation and Section 4.6 presents the results of our simulations, using both controllers, before and after Bayesian Optimisation. Finally, in Section 4.7 we provide a thorough discussion of our findings.

4.2 Linear Quadratic Output Feedback Control for a state-space system

Faulty prototypes often occur in AM processes due to issues related to temperature regulation in various subprocesses. More specifically, the cause of these faults may be the insufficient or excessive melting of the material, the unsuccessful binding between layers or the non-uniform heating of the build-plate which can stop the print from adhering properly.

In this work, we study the heating system of a Material Extrusion (MEX) system. In particular, we are focusing on the temperatures within the extruder of a Big Area Additive Manufacturing (BAAM) system. The extruder consists of 5 main parts, the hopper, the barrel, the hose, the nozzle and the AC motor. The hopper is where the material is inserted and gets fed to the barrel containing the screw which gets rotated by the motor. The screw pushes the material towards the hose and finally to the nozzle. There are four heaters placed along the barrel, one heater in the hose and one in the nozzle. The reason behind the use of multiple heaters in BAAM systems is to address the large volume of material that gets processed by the system. The areas where the heat is applied can be viewed as thermal cells, each influencing their neighbouring cells and all getting influenced by the motor. We adopt a discrete time, linear state-space representation of the system inspired by the system model obtained in [69] through system identification methods, using real process data. We make the further assumption that the temperatures within the extruder are not directly available, instead, a set of measurements can be obtained through a number of sensor measurements of the process.

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{4.1}$$

where $t \geq t_0$ is the discrete time step, $x(t) \in \mathbb{R}^n$ is the system state at time t, $u(t) \in \mathbb{R}^m$ is the system input at t and $y(t) \in \mathbb{R}^p$ is the system output obtained as measurements.

Matrices $A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}$ and $C \in \mathbb{R}^{p \times n}$ are the state, input and measurement matrices respectively. For our system, there are $m = 7$ inputs, the first 6 representing the load applied to each thermal cell by the corresponding heater and the last relaying the input applied by the motor. The system state is a vector of length $n = 6$ representing the temperature within each thermal cell. We assume that the sensor system obtains $p = 5$ measurements of the internal states of the system. A visual representation of the system setup can be seen in Figure 4.1.

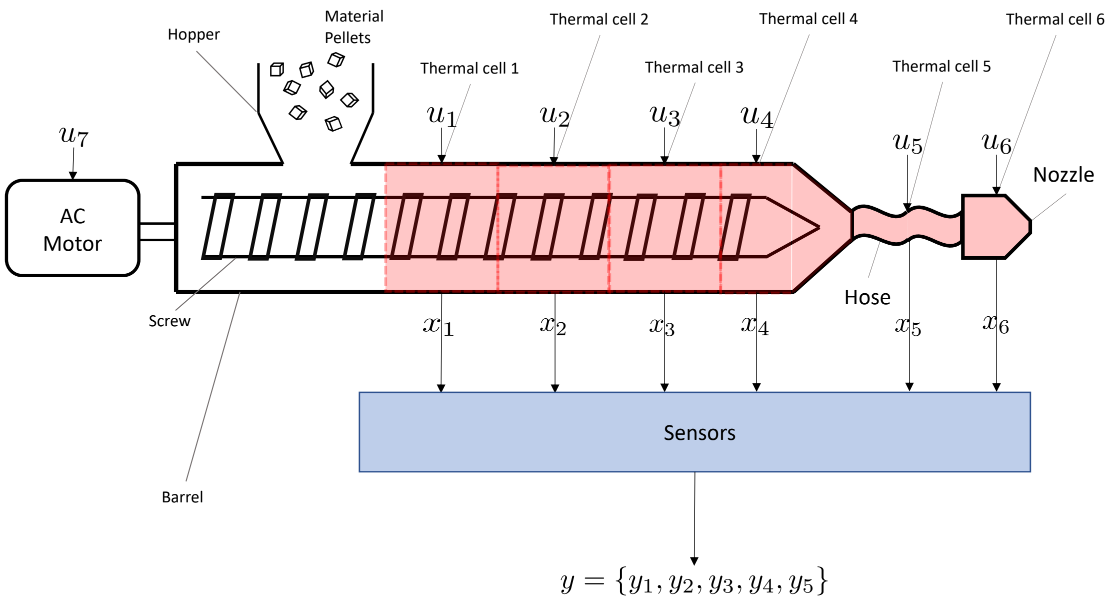


Figure 4.1: Heating system for a BAAM extruder. The inputs include the heat applied in each thermal cell and the input provided by the AC motor rotating the screw. The system states represent the temperatures within the extruder, which are measured by the sensors.

The temperature control problem is defined as the infinite horizon optimal tracking problem, where the goal is to obtain the optimal input (or control) u^* such that the system output accurately tracks the reference signal $r(t) \in \mathbb{R}^p$, generated according to matrix $F \in \mathbb{R}^{p \times p}$

$$r(t+1) = Fr(t). \tag{4.2}$$

In most applications $r(t)$ is chosen to be constant, namely $F = I_p$ or $r(t) = r \in \mathbb{R}$ for all t. However, the general, time-varying reference signal (4.2) can also be adopted in our formulation. The control objective can be formally expressed as the minimisation of the performance index defined as the total sum of discounted costs over an infinite horizon, namely:

$$\begin{aligned}\min_{\mathbf{u}} J(x(0),\mathbf{u}) &= \min_{\mathbf{u}} \sum_{t=0}^{\infty} \gamma^t \left\{[y(t)-r(t)]^T Q[y(t)-r(t)] + u^T(t)Ru(t)\right\} \\ &= \min_{\mathbf{u}} \sum_{t=0}^{\infty} \gamma^t \left\{[Cx(t)-r(t)]^T Q[Cx(t)-r(t)] + u^T(t)Ru(t)\right\}\end{aligned} \tag{4.3}$$

where $0 < \gamma \leq 1$ is the discount factor, putting greater emphasis on short-term costs. $Q \in \mathbb{R}^{p\times p}$ and $R \in \mathbb{R}^{m\times m}$ are positive definite symmetric matrices applying weight to the tracking error and the input cost respectively. $x(0)$ gives rise to all future states $x(t)$ and we define $\mathbf{u}$ as the infinite sequence containing all inputs $\{u(0), u(1), u(2), \dots\}$, also known as a *policy*. This means that the minimisation problem is equivalent to the search for an optimal policy $\mathbf{u}^*$. In the Reinforcement Learning literature [13, 16, 42], the minimisation problem is addressed through the principle of optimality:

Definition 5 (Principle of optimality) *[16] An optimal policy has the property that whatever the initial state and initial input are, the remaining inputs must constitute an optimal policy with regard to the state resulting from the first decision.*

To utilise the principle of optimality, we define the *Value Function* as the cost-to-go at time t, namely the sum of discounted future costs from time t onwards.

$$\begin{aligned}V(x(t),\mathbf{u}_t) &= \sum_{k=t}^{\infty} \gamma^{k-t} \left\{[y(k)-r(k)]^T Q[y(k)-r(k)] + u^T(k)Ru(k)\right\} \\ &= \sum_{k=t}^{\infty} \gamma^{k-t} \left\{[Cx(k)-r(k)]^T Q[Cx(k)-r(k)] + u^T(k)Ru(k)\right\}\end{aligned} \tag{4.4}$$

where $\mathbf{u}_t = \{u(t), u(t+1), u(t+2), \dots\}$. The Value function allows for the principle of optimality to be expressed mathematically as the **Bellman optimality equation** [13,42]. Let $c(x(t), u(t)) = [Cx(t)-r(t)]^T Q[Cx(t)-r(t)] + u^T(t)Ru(t)$ be the cost at time t. Then for the optimal policy $\mathbf{u}^*$, the principle of optimality yields

$$V(x(t),\mathbf{u}_t^*) = c(x(t), u^*(t)) + \gamma V(x(t+1), \mathbf{u}_{t+1}^*) \tag{4.5}$$

for all t. In the following sections, we explore two different approaches to solving the optimal tracking problem. The first is based on the design of a state observer while the second utilises a record of past inputs and outputs to estimate the internal state. While the observer design requires knowledge of the system dynamics, the second approach can be adapted to be completely model free and data-driven.

4.3 State Observer based Output Feedback

An important issue in the design of efficient controllers is the lack of access to the exact system state $x(t)$. Assuming knowledge of the system dynamics (4.1), a state observer can be implemented to provide an informed estimate of the state, that can then be used for control design. This observer is also known as a closed loop observer or a Luenberger observer [11, 89].

In practice, the Luenberger observer is another state space system, made up of the same state, input and output matrices as the original system, where the state equation has an additional term associated with the error between the system output and the observer output.

$$\begin{aligned} \hat{x}(t+1) &= A\hat{x}(t) + Bu(t) + L\left[y(t) - \hat{y}(t)\right] \\ \hat{y}(t) &= C\hat{x}(t) \end{aligned} \tag{4.6}$$

The matrix $L \in \mathbb{R}^{n \times p}$ is known as the observer gain. The goal is to have an observer that emulates the behaviour of the original system as closely as possible. In control theory terms, we need to find a gain matrix L such that the matrix $A - LC$ is stable [11]. This problem can be approached as a pole placement problem, where the objective is to place the poles of $A - LC$ inside the unit circle. For that to be possible, the system needs to be *completely state observable* [11]. The observability condition requires the column rank of the matrix

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{4.7}$$

to be n. If this condition is satisfied, the system state can be fully determined from the system inputs and outputs. One approach to solve the pole placement problem is to choose a small positive number τ and calculate the observer gain through:

$$L = AC^T\Psi^{-1} \text{ where } \Psi = CC^T + \tau I_p \tag{4.8}$$

Once L is determined, the state estimate $\hat{x}$ can be used in place of x to design a controller. This means that the value function (4.4) can be rewritten as follows:

$$V_{obs}(x(t), \mathbf{u}_t) = \sum_{k=t}^{\infty} \gamma^{k-t} \left\{ [C\hat{x}(k) - r(k)]^T Q [C\hat{x}(k) - r(k)] + u^T(k) R u(k) \right\} \tag{4.9}$$

For the optimal control problem to be successfully solved, the system needs to be *controllable* [11]. System (4.1) is said to be controllable when the matrix

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix} \tag{4.10}$$

known as the controllability matrix, has full row rank n. The controllability condition implies that the system can be controlled to have the desired behaviour within a finite number of steps, through an appropriate choice of control function u.

For the infinite horizon LQT problem (4.9), a controller can be designed by adopting the approach in [79]. Consider the augmented state

$$X_{obs}(t) = \begin{bmatrix} \hat{x}(t) \\ r(t) \end{bmatrix} \tag{4.11}$$

and the corresponding system dynamics

$$X_{obs}(t+1) = TX_{obs}(t) + B_1 u(t). \tag{4.12}$$

where $T = \begin{bmatrix} A & 0 \\ 0 & F \end{bmatrix}$ and $B_1 = \begin{bmatrix} B \\ 0 \end{bmatrix}$. Using (4.11), (4.9) can be rewritten as

$$V_{obs}(X(t), \mathbf{u}_t) = \sum_{k=t}^{\infty} \gamma^{k-t} \{ X_{obs}^T(k) Q_1 X_{obs}(k) + u^T(k) R u(k) \} \tag{4.13}$$

where

$$Q_1 = \begin{bmatrix} C^TQC & -C^TQ \\ -QC & Q \end{bmatrix}. \tag{4.14}$$

The optimal policy is in feedback form of the augmented system state

$$u^*(t) = -KX_{obs}(t). \tag{4.15}$$

where K is the optimal gain. K is commonly obtained through the solution of an Algebraic Riccati Equation (ARE) which can be a challenging task and computationally expensive. An alternative approach to determining K, particularly useful for large-scale models, is through Reinforcement Learning (RL) techniques, namely, by utilising the Bellman equation [41–43]. In order to form the Bellman equation, the following lemma is needed.

Lemma 1 *[79] For the optimal tracking problem with value function of the form* (4.9) *and reference generated by $r_{obs}(t+1) = F_{obs} x_{obs}(t)$, then for any stabilising policy of the* *form (4.15), the value function can be written in quadratic form as*

$$V_{obs}(x(t),\mathbf{u}_t) = V_{obs}(X(t)) = \frac{1}{2}X_{obs}^T(t)PX(_{obs}t) \tag{4.16}$$

for some matrix $P = P^T > 0$.

Utilising the quadratic form of the value function, it can be rewritten as the Bellman equation

$$X_{obs}^T(t)PX_{obs}(t) = X_{obs}^T(t)Q_1X_{obs}(t) + u^T(t)Ru(t) + \gamma X_{obs}^T(t+1)PX_{obs}(t+1). \tag{4.17}$$

Consider a stabilising policy $\hat{K}^*$ in (4.17). A matrix K is said to be stabilising when the matrix $A-BK$ is stable, namely has eigenvalues strictly within the unit circle [10]. Using the augmented system dynamics (4.12), we obtain the Lyapunov equation

$$P = Q_1 + (\hat{K}^*)^TR\hat{K}^* + \gamma(T + B_1\hat{K}^*)^TP(T + B_1\hat{K}^*). \tag{4.18}$$

Iterative solutions of the Lyapunov equation while updating the optimal control gain estimate where

$$\hat{K}^* = (R + \gamma B_1^TPB_1)^{-1}\gamma B_1^TPT \tag{4.19}$$

and the corresponding control signal $\hat{u}^* = -\hat{K}^*X_{obs}$ can effectively lead to the solution to the Optimal Tracking Problem. A step-by-step description of the RL based solution to the LQT problem using state estimation through a state observer is given in Algorithm 8.

If the initial conditions given to the observer are the same or close in value to the true initial conditions of the system, the observer estimates the state very closely within a few time steps. The further away the initial values are, the longer it takes for the observer to estimate the state correctly, so the longer it takes for the system's behaviour to be optimised, as the controller is designed based on the state estimates.

While the observer is a powerful estimator of the state, it includes an important assumption which is access to the system dynamics, namely matrices A, B and C. This is an unrealistic assumption in many applications, where a model of the process is either not available or not accurate enough. In the next section, we explore another approach for control design with state estimation.

4.4 Output Feedback using Input-Output data

Consider the performance index (4.4) to be minimised with r generated as in (3.8). Similarly to the previous section, we augment the state with the reference and construct the

Algorithm 8: LQT control with state estimation through Luenberger observer

```
Result: State estimate $\hat{x}(t)$, Optimal control gain $K^*$, optimal controller $u^*(t)$
Input: State input and output matrices $A, B, C$, Weighting matrices for
       performance index $Q, R$, Initial state estimate $\hat{x}(t_0)$, small positive
       number $\tau$, Initial reference signal $r(t_0)$, Discount factor $\gamma$, Initial control
       policy $K^0$, Error threshold $\epsilon$,, Simulation length $N$
1  Create augmented system matrices $T = \begin{bmatrix} A & 0 \\ 0 & F \end{bmatrix}, B_1 = \begin{bmatrix} B \\ 0 \end{bmatrix}$;
2  Augment weighting matrix $Q_1 = \begin{bmatrix} C^TQC & -C^TQ \\ -QC & Q \end{bmatrix}$.;
3  Set $K^{new} = K^0$ and randomly initialise $K$;
4  while $\|K - K^{new}\| > \epsilon$ do
5      Set $K = K^{new}$;
6      Solve the LQT Lyapunov equation
         $P = Q_1 + K^TRK + \gamma(T - B_1K)^TP(T - B_1K)$;
7      Compute the control gain estimate $K^{new} = (R + \gamma B_1^TPB_1)^{-1}\,\gamma B_1^TPT$;
8  end
9  Set $K^* = K$;
10 for $t = t_0, N$ do
11     Design augmented state $X_{obs}(t) = \begin{bmatrix} \hat{x}(t) \\ r(t) \end{bmatrix}$;
12     Obtain system input $u^*(t) = -K^*X_{obs}(t)$ ;
13     Obtain output estimation $\hat{y}(t) = C\hat{x}(t)$;
14     Observe system output $y(t) = Cx(t)$;
15     Determine $L = AC^T\Psi^{-1}$ where $\Psi = CC^T + \tau I_p$;
16     State estimate $\hat{x}(t+1) = A\hat{x}(t) + Bu^*(t) + L(y(t) - \hat{y}(t))$;
17     System state update (not measurable) $x(t+1) = Ax(t) + Bu^*(t)$;
18     Reference signal update $r(t+1) = Fr(t)$;
19 end
```

augmented system dynamics:

$$X(t+1) = \begin{bmatrix} x(t+1) \\ r(t+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} x(t) \\ r(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) = TX(t) + B_1 u(t). \tag{4.20}$$

The value function can be written in terms of X as

$$V(X(t), \mathbf{u}_t) = \sum_{k=t}^{\infty} \gamma^{k-t} \{X^T(k) Q_1 X(k) + u^T(k) R u(k)\} \tag{4.21}$$

where Q_1 defined as in (4.14). Assuming a feedback control policy, Lemma 1 allows for 4.21 to be written as

$$V(x(t), \mathbf{u}_t) = V(X(t)) = \frac{1}{2} X^T(t) P_1 X(t) \tag{4.22}$$

which gives rise to the Bellman equation

$$X^T(t) P_1 X(t) = X^T(t) Q_1 X(t) + u^T(t) R u(t) + \gamma X^T(t+1) P_1 X(t+1). \tag{4.23}$$

Unlike the previous section, in addition to not having access to the internal system state, we no longer assume knowledge of the system dynamics. Instead, we estimate the state through a record of past inputs and outputs, under the assumption that the system (4.1) is controllable and observable. Consider a time horizon N and recursively express the state $x(t)$ in terms of past inputs $u(t-1), \dots, u(t-N)$

$$\begin{aligned} x(t) &= Ax(t-1) + Bu(t-1) \\ &= A^2 x(t-1) + ABu(t-2) + Bu(t-1) \\ &\dots \\ &= A^N x(t-N) + \begin{bmatrix} B & AB & A^2B & \dots & A^{N-1}B \end{bmatrix} \begin{bmatrix} u(t-1) \\ u(t-2) \\ \vdots \\ u(t-N) \end{bmatrix} \\ &:= A^N x(t-N) + U_N \bar{u}(t-1, t-N). \end{aligned} \tag{4.24}$$

Similarly, the system output can be expressed recursively as

$$y(t) = Cx(t) = CA^N x(t-N) + CU_N \bar{u}(t-1, t-N). \tag{4.25}$$

Consider a sequence of N outputs

$$\bar{y}(t-1,t-N) = \begin{bmatrix} y(t-1) \\ y(t-2) \\ y(t-3) \\ \vdots \\ y(t-N) \end{bmatrix} \tag{4.26}$$

which can be expressed in terms of $x(t-N)$ and $\bar{u}(t-1,t-N)$ as

$$\begin{aligned} \bar{y}(t-1,t-N) &= \begin{bmatrix} CA^{N-1} \\ CA^{N-2} \\ \vdots \\ CA \\ C \end{bmatrix} x(t-N) \\ &+ \begin{bmatrix} 0 & CB & CAB & \dots & CA^{N-2}B \\ 0 & 0 & CB & \dots & CA^{N-3}B \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & CB \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} u(t-1) \\ u(t-2) \\ \vdots \\ u(t-N) \end{bmatrix} \\ &:= W_N x(t-N) + D_N \bar{u}(t-1,t-N) \end{aligned} \tag{4.27}$$

Matrix W_N has full rank for $N \geq O$ where O is the rank of the observability matrix (4.7). Since we assume our system to be observable, then $O = n$. If N is chosen to be equal to n, matrix U_N is equal to the controllability matrix $\mathcal{C}$ (4.10) and W_N is equal to the observability matrix $\mathcal{O}$ (4.7) flipped from top to bottom.

We aim to express the state $x(t)$ in terms of past inputs and outputs. Based on (4.24), $x(t-N)$ needs to be replaced by an expression in terms of $u(t-1), \dots, u(t-N)$ and $y(t-1), \dots, y(t-N)$. This can be achieved by solving (4.25) for $x(t-N)$. To that end, consider the generalised Moore-Penrose inverse [96, 97] of W_N, $W_N^+ = (W_N^T W_N)^{-1} W_N^T$. Then

$$x(t-N) = W_N^+(\bar{y}(t-1,t-N) - D_N \bar{u}(t-1,t-N)). \tag{4.28}$$

Considering that $r(t) = F^N r(t-N)$, the augmented state $X(t)$ can be written as:

$$\begin{aligned}
X(t) &= \begin{bmatrix} A^N & 0 \\ 0 & F^N \end{bmatrix} \begin{bmatrix} x(t-N) \\ r(t-N) \end{bmatrix} + \begin{bmatrix} U_N \\ 0 \end{bmatrix} \bar{u}(t-1, t-N) \\
&= \begin{bmatrix} A^N & 0 \\ 0 & F^N \end{bmatrix} \begin{bmatrix} W_N^+(\bar{y}(t-1,t-N) - D_N \bar{u}(t-1,t-N)) \\ r(t-N) \end{bmatrix} \\
&\quad + \begin{bmatrix} U_N \\ 0 \end{bmatrix} \bar{u}(t-1,t-N) \\
&= \begin{bmatrix} U_N - A^N W_N^+ D_N & A^N W_N^+ & 0 \\ 0 & 0 & F^N \end{bmatrix} \begin{bmatrix} \bar{u}(t-1,t-N) \\ \bar{y}(t-1,t-N) \\ r(t-N) \end{bmatrix} \\
&:= M \begin{bmatrix} \bar{u}(t-1,t-N) \\ \bar{y}(t-1,t-N) \\ r(t-N) \end{bmatrix} \qquad (4.29)
\end{aligned}$$

The above result establishes that the system state can be expressed in terms of past inputs and outputs with the help of matrix M. Let $Z(t) = \begin{bmatrix} X(t) \\ u(t) \end{bmatrix}$ and consider the quadratic form of the value function 4.22. It can be rewritten as

$$V(Z(t)) = \frac{1}{2} Z^T(t) H Z(t) \qquad (4.30)$$

with $H = \begin{bmatrix} Q_1 + \gamma T^T P_1 T & \gamma T^T P_1 B_1 \\ \gamma B_1^T P_1 T & R + \gamma B_1^T P_1 B_1 \end{bmatrix}$. Using 4.29 in 4.30 we obtain

$$\begin{aligned}
V(Z(t)) &= \frac{1}{2} \begin{bmatrix} X(t) \\ u(t) \end{bmatrix}^T H \begin{bmatrix} X(t) \\ u(t) \end{bmatrix} \\
&= \frac{1}{2} \begin{bmatrix} M \begin{bmatrix} \bar{u}(t-1,t-N) \\ \bar{y}(t-1,t-N) \\ r(t-N) \end{bmatrix} \\ u(t) \end{bmatrix}^T H \begin{bmatrix} M \begin{bmatrix} \bar{u}(t-1,t-N) \\ \bar{y}(t-1,t-N) \\ r(t-N) \end{bmatrix} \\ u(t) \end{bmatrix} \qquad (4.31)
\end{aligned}$$

Set $\bar{Z}(t) = \begin{bmatrix} \bar{u}(t-1,t-N) \\ \bar{y}(t-1,t-N) \\ r(t-N) \\ u(t) \end{bmatrix}$ and $\bar{H} = \begin{bmatrix} M & 0 \\ 0 & I_m \end{bmatrix}^T H \begin{bmatrix} M & 0 \\ 0 & I_m \end{bmatrix}$. Then we can rewrite

4.30 as

$$\begin{aligned} V(\bar{Z}(t)) &= \frac{1}{2}\bar{Z}^T(t)\bar{H}\bar{Z}(t) \\ &= \frac{1}{2}\begin{bmatrix} \bar{u}(t-1,t-N+1) \\ \bar{y}(t,t-N+1) \\ r(t-N+1) \\ u(t) \end{bmatrix}^T \begin{bmatrix} H_{\bar{u}\bar{u}} & H_{\bar{u}\bar{y}} & H_{\bar{u}r} & H_{\bar{u}u} \\ H_{\bar{y}\bar{u}} & H_{\bar{y}\bar{y}} & H_{\bar{y}r} & H_{\bar{y}u} \\ H_{r\bar{u}} & H_{r\bar{y}} & H_{rr} & H_{ru} \\ H_{u\bar{u}} & H_{u\bar{y}} & H_{ur} & H_{uu} \end{bmatrix} \begin{bmatrix} \bar{u}(t-1,t-N+1) \\ \bar{y}(t,t-N+1) \\ r(t-N+1) \\ u(t) \end{bmatrix} \end{aligned} \tag{4.32}$$

The decomposition of $\bar{H}$ in blocks of the form H_{ij} is according to the dimensions of each set of i, j. For example block $H_{\bar{u}\bar{u}}$ will be a $Nm \times Nm$ matrix and block $H_{u\bar{y}}$ will be a $m \times Np$ matrix.

Due to the quadratic form of the value function, the optimality condition $\frac{\partial V}{\partial u} = 0$ is sufficient to ensure its minimisation. Solving for $u(t)$ yields

$$u(t) = -H_{uu}^{-1}(H_{u\bar{u}}\bar{u}(t-1,t-N) + H_{u\bar{y}}\bar{y}(t-N,t-N) + H_{ur}r(t-N)). \tag{4.33}$$

If the system model is available, 4.33 can be used to directly determine the optimal LQT controller using input-output information. The kernel matrix $\bar{H}$ can be directly determined using

$$\begin{aligned} \bar{H} &= \begin{bmatrix} M & 0 \\ 0 & I_m \end{bmatrix}^T H \begin{bmatrix} M & 0 \\ 0 & I_m \end{bmatrix} = \begin{bmatrix} U_N - A^N W_N^+ D_N & A^N W_N^+ & 0 & 0 \\ 0 & 0 & F^N & 0 \\ 0 & 0 & 0 & I_m \end{bmatrix}^T \\ & \begin{bmatrix} Q_1 + \gamma T^T P_1 T & \gamma T^T P_1 B_1 \\ \gamma B_1^T P_1 T & R + \gamma B_1^T P_1 B_1 \end{bmatrix} \begin{bmatrix} U_N - A^N W_N^+ D_N & A^N W_N^+ & 0 & 0 \\ 0 & 0 & F^N & 0 \\ 0 & 0 & 0 & I_m \end{bmatrix} \end{aligned} \tag{4.34}$$

where P_1 is obtained by solving the standard LQT problem as in 4.18.

We are interested in the case when a reliable model of the system is not available, meaning matrices A, B and C are unknown. In this case, the kernel matrix $\bar{H}$ needs to be estimated strictly from data. To that end, consider the Bellman equation derived from the quadratic form 4.30

$$\begin{aligned} \bar{Z}(t)^T \bar{H} \bar{Z}(t) &= (y(t) - r(t))^T Q (y(t) - r(t)) + u^T(t) R u(t) \\ &\quad + \gamma \bar{Z}(t+1)^T \bar{H} \bar{Z}(t+1). \end{aligned} \tag{4.35}$$

To obtain an approximation formula for $\bar{H}$, it needs to be isolated from the quadratic form. This can be achieved by firstly vectorising (4.35):

$$\begin{aligned} vec\left(\bar{Z}^T(t)\bar{H}\bar{Z}(t)\right) &= (y(t)-r(t))^T Q(y(t)-r(t)) + u^T(t)Ru(t) \\ &\quad +\gamma vec\left(\bar{Z}^T(t+1)\bar{H}\bar{Z}(t+1)\right) \end{aligned} \tag{4.36}$$

where the *vec* operator stacks the columns of a matrix to create a single column. We then apply the "vector trick" associated with the Kronecker product

$$\begin{aligned} \left(\bar{Z}^T(t)\otimes\bar{Z}^T(t)\right) vec(\bar{H}) &= (y(t)-r(t))^T Q(y(t)-r(t)) + u^T(t)Ru(t) \\ &\quad +\gamma\left(\bar{Z}^T(t+1)\otimes\bar{Z}^T(t+1)\right) vec(\bar{H}). \end{aligned} \tag{4.37}$$

To solve 4.37, we may employ the Policy Iteration (PI) or Value Iteration (VI) algorithm [42]. In the standard Reinforcement Learning framework, where the problem is modelled as a Markov Decision Process with discrete state and action spaces, there are two main differences between the algorithms: Firstly, the PI assumes that the initial policy is an admissible one while VI has no such requirement. The second difference lies in the way the optimal policy is built. VI focuses on optimising the value function over all possible actions and extracting the optimal policy at the end. PI updates the value function based on the current policy estimate and extracts a new, better policy afterwards, iteratively until the optimal is reached. In the case of continuous state and action spaces, such as our control problem, we will focus on the VI algorithm, mainly due to the lack of need for an initial stabilising controller (admissible policy). When the system dynamics are not known, the design of such a controller becomes very challenging, though some work has been done in recent literature on obtaining stabilising controllers through measured data [98]. The algorithm iterates between a Policy Evaluation and a Policy Improvement step. We start with an arbitrarily chosen initial kernel matrix $\bar{H}^0$ that obtains an initial policy $u^0(t)$. We then iterate between the following two steps:

(1) **Policy Evaluation**

$$\begin{aligned} \left(\bar{Z}^T(t)\otimes\bar{Z}^T(t)\right) vec(\bar{H}^{i+1}) &= (y(t)-r(t))^T Q(y(t)-r(t)) + u^T(t)Ru(t) \\ &\quad +\gamma\left(\hat{\bar{Z}}^T(t+1)\otimes\hat{\bar{Z}}^T(t+1)\right) vec(\bar{H}^i) \end{aligned}$$

(2) **Policy Improvement**

$$\begin{aligned} u^{i+1}(t) &= -(H_{uu}^{i+1})^{-1}(H_{u\bar{u}}\bar{u}^{i+1}(t-1,t-N) + H_{u\bar{y}}^{i+1}\bar{y}(t-1,t-N) \\ &\quad +H_{ur}^{i+1}r(t-N)) \end{aligned}$$

The Policy evaluation step can be solved through the Least Squares (LS) algorithm using measured data $\bar{Z}(t)$ and calculating the corresponding cost term $(y(t) - r(t))^T Q(y(t) - r(t)) + u^T(t)Ru(t)$. It is worth noting that for the discounted future cost term $\gamma\left(\hat{\bar{Z}}^T(t+1) \otimes \hat{\bar{Z}}^T(t+1)\right) vec(\bar{H}^i)$, we are not using the measured input $u(t+1)$ but instead the estimated optimal input $u^i(t+1)$, namely $\hat{\bar{Z}}(t+1) = \begin{bmatrix}\bar{u}(t, t-N+1) & \bar{y}(t, t-N+1) & r(t-N+1) & u^i(t+1)\end{bmatrix}^T$. In RL terms, this yields the expected future reward based on the current policy. Matrix $\bar{H}$ is an $((N+1)m + (N+1)p) \times ((N+1)m + (N+1)p)$ symmetric matrix which means that its determination is a problem that requires a minimum of $((N+1)m+(N+1)p) \times ((N+1)m+(N+1)p+1)/2$ data points. In practice, many more data points are usually needed, especially when dealing with more complicated systems with larger state and action spaces. When the data is highly correlated, which is often the case with sequential input output data from a control system, the inversion step in the LS algorithm becomes challenging. This can be rectified with a regularized LS approach, which includes an appropriate regularisation parameter μ. A step by step description of the method introduced in this section is given in Algorithm 9.

From a software point of view, the implementation of Algorithm 9 can become very computationally expensive with larger system dimensions. There are a few ways to work around this problem. The construction of matrix L in step (12) consists of the Kronecker product terms obtained directly from the dataset. This means that L can be determined only once outside the training iterations. Similarly for its regularised version L_r and more importantly for the inverse L_r^{-1}. Matrix inversion is a very expensive operation and an approximation of the inverse might be necessary. Specifically, we chose to calculate the inverse with the use of the Cholesky decomposition [99].

4.5 Parameter Tuning through Bayesian Optimisation

There are multiple aspects that influence the performance of a controller, both in the model-based scenario and especially in the model-free, data-driven scenario. In the model-based case, these aspects concern mainly the control objective design. In the case of data-driven control, the resulting controller's performance is influenced as much by the characteristics of the data as by the definition of the optimisation problem. In the next section we discuss the setup of our simulations so as to generate informative data, that can be used to effectively train a controller. In this section, we focus on optimising the design of the performance index, which dictates the control objective.

Apart from the system dynamics (4.1), the main parameters influencing the optimisation are the weighting matrices Q and R in the performance index (4.3). However, there is no systematic way to choose the optimal values for these matrices, as their direct relation-

Algorithm 9: LQT output-feedback control using input-output measured data

```
   Result: Optimal kernel matrix estimate $H^*$, optimal controller estimate $u^*(t)$
   Input: Weighting matrices for performance index $Q_1, R$, regularisation
          parameter $\mu$, Reference signal $r(t)$, Discount factor $\gamma$, Initial kernel
          matrix $H^0$, Error threshold $\epsilon_{rl}$, time horizon $N$
 1 Collect $M \gg ((N+1)m + (N+1)p) \times ((N+1)m + (N+1)p)/2$ consecutive data
     tuples $\{y(t), u(t)\}$;
 2 Create vectors $\bar{u}(t-1, t-N), \bar{y}(t-1, t-N)$ and
     $\bar{Z}(t) = \begin{bmatrix}\bar{u}(t-1, t-N) & \bar{y}(t-1, t-N) & r(t-N) & u(t)\end{bmatrix}^T$;
 3 For each $\bar{Z}(t)$, calculate the Kronecker product $K_{\bar{Z}}(t) := \bar{Z}^T(t) \otimes \bar{Z}^T(t)$;
 4 Set $H^{new} = H^0$ and randomly initialise $H$;
 5 while $\|H - H^{new}\| \leq \epsilon_{rl}$ do
 6     $H = H^{new}$;
 7     for t=N,M do
 8         Determine
             $u^{est}(t+1) = -(H_{uu})^{-1}(H_{u\bar{u}}\bar{u}(t-1, t-N) + H_{u\bar{y}}\bar{y}(t-1, t-N) + H_{ur}r(t-N))$;
 9         Create the updated augmented state
             $\hat{\bar{Z}}(t+1) = \begin{bmatrix}\bar{u}(t, t-N+1) & \bar{y}(t, t-N+1) & r(t-N+1) & u^{est}(t+1)\end{bmatrix}^T$
           and the corresponding Kronecker product
           $K_{\hat{\bar{Z}}}(t+1) = \hat{\bar{Z}}^T(t+1) \otimes \hat{\bar{Z}}^T(t+1)$;
10         Compute the future expected cost term $c(t) = X^T(t)Q_1X(t)$
             $+u^T(t)Ru(t) + \gamma K_{\hat{\bar{Z}}}(t+1)vec(H)$;
11     end
12     Calculate the sums $L = \sum_{t=N+1}^{M} K_Z^T(t)K_Z(t)$ and $S = \sum_{t=N+1}^{M} K_Z^T(t)c(t)$ ;
13     Regularise matrix L: $L_r = L + \mu I$;
14     Produce new H estimate $H^{new} = L_r^{-1}S$;
15 end
```

ship to the controller performance cannot be expressed by a known function. Hence, they are commonly chosen empirically by the engineer, or for simpler problems, an exhaustive search may be performed to obtain the best choice. For our system, an exhaustive search would not be feasible, especially in the model-free case where the calculations become vastly more computationally expensive due to the system augmentation. Motivated by this we explore Bayesian Optimisation [91,92,100], which is a powerful optimization tool in which uncertainty over the objective function is typically represented as a Gaussian process (GP) [101].

In particular, we want to design a fitness function which quantifies the effect of Q and R on the controller's performance. We consider a controller to have good performance if it tracks the reference signal closely, whilst not applying costly inputs to the system. A choice for the fitness function could be the performance index itself as defined in (4.3). There are two main drawbacks to this choice. Firstly, an infinite sum is not feasible, which means it would need to be replaced by a finite sum corresponding to the length of the experiment. Secondly, using (4.3) as the fitness function to be minimised, may imply that using very small values for Q and R is sufficient to have a good performance. Such a choice, however, would only ensure that the sum of discounted costs is small, without necessarily guaranteeing good tracking performance and low input costs. Hence, we define the simpler fitness function

$$F(\theta) = \sum_{t=0}^{l} \gamma^t (\|y(t) - r(t)\|^2 + \|u(t)\|^2) \tag{4.38}$$

where $\|\cdot\|$ denotes the Euclidian norm and l is the length of the experiment. $\theta \in \mathcal{D}$ represents the parameterised choice of Q and R sampled from a bounded search domain $\mathcal{D}$. Considering that Q and R are symmetric positive definite matrices, we can choose them to be diagonal matrices with all positive entries, as is common practice for simplicity. This means that θ can be written as $\theta = \{q_1, \dots, q_p, r_1, \dots, r_m\}$ and $\mathcal{D} \subset \mathbb{R}_+^{p+m}$. This choice for F allows us to investigate the tracking and input costs as an indirect result of Q and R, with the weighting matrices being used in the Bellman equation to obtain the optimal policy and thus the optimal trajectory.

We want to minimise the fitness function with respect to θ. F can be modelled through a GP prior as

$$F(\theta) \sim \mathcal{GP}(\mu(\theta), k(\theta, \theta')) \tag{4.39}$$

The GP is characterised by a mean $\mu(\theta)$ and a covariance or kernel function $k(\theta, \theta')$.

Consider a sample of l points $\{x_1, \dots, x_l\}$ and calculate the covariance for each pair $\{x_i, x_j\}$. The mean for this initial sample may be chosen as $\mu_0 = 0$ which is a common choice for the mean when no prior information is available. For the kernel, we choose one of the most popular and simpler kernel functions which is the *Square Exponential* or *Gaussian* kernel

$$k_0(\theta, \theta') = \exp(-\|\theta - \theta'\|^2) \text{ where } \|\theta - \theta'\|^2 = \sum_{i=1}^{p+m} (\theta_i - \theta_i')^2. \tag{4.40}$$

All the above yield the prior distribution

$$F(x_{1:l}) \sim \mathcal{N}(\mu_0(x_{1:l}), k_0(x_{1:l})) \tag{4.41}$$

where $F(x_{1:l}) = \{F(x_1), \dots, F(x_l)\}$, $\mu_0(x_{1:l}) = 0_l$ and $k_0(x_{1:l}) = \{k(x_1, x_1), \dots, k(x_1, x_l); \dots; k(x_l, x_1), \dots, k(x_l, x_l)\}$. Once the prior distribution has been determined for the initial sample of points, we can evaluate F at a new point x and compute the conditional distribution of $F(x)$ using Bayes' rule

$$F(x)|F(x_{1:l}) \sim \mathcal{N}(\mu_l(x), k_l(x)) \tag{4.42}$$

where $\mu_l(x) = k_0(x, x_{1:l})k_0(x_{1:l}, x_{1:l})^{-1}(F(x_{1:l}) - \mu_0(x_{1:l})) + \mu_0(x)$ and $k_l(x) = k_0(x, x) - k_0(x, x_{1:l})k_0(x_{1:l}, x_{1:l})^{-1}k_0(x_{1:l}, x)$. Distribution (4.42) is also known as the *posterior probability distribution.* The more points that are sampled, the more informed the posterior becomes.

Returning to the optimisation problem at hand, we are looking for a choice of Q and R that minimise F. The tool used in Bayesian Optimisation to find such a point is known as the *acquisition function.* Acquisition functions provide a candidate for the best point to sample next, based on the current estimate for the distribution. There are a few different choices for acquisition functions, the most common including *Expected Improvement, Probability of Improvement* and *Gaussian Process Upper Confidence Bound.* For our application, we will be using the latter, defined as follows

$$a_{UCB}(x; x_{1:l}) = \mu_l(x) - \epsilon k_l(x) \tag{4.43}$$

where ϵ is a parameter balancing exploration against exploitation. The next point to sample V is chosen as $x_{next} = \arg\min a_{UCB}$. The new point can be used to update the posterior distribution, while resulting in a more informed acquisition function that will in turn suggest a better point to sample next. The process is repeated for a predefined number of steps and the optimal is extracted at the end if M points have been sampled in total, as

$$x_{best} = \arg\min_x \mu_M(x). \tag{4.44}$$

4.6 Simulation setup and results

In this section we perform numerical simulations to investigate the behaviour of the extruder heating system introduced in Section 4.2 after a controller is introduced to the system. We assume partial access to the system state in the form of measurements and explore both the case of full knowledge of the system dynamics and the case where a system model is unknown. For our simulations, we need a model of the extruder dynamics which will be used for the design of the model-based controller and to generate data to train the data-driven controller. In [69], system identification methods were used to obtain the following state and input matrices for a BAAM system:

$$\begin{aligned} A &= \begin{bmatrix} 0.992 & 0.0018 & 0 & 0 & 0 & 0 \\ 0.0023 & 0.9919 & 0.0043 & 0 & 0 & 0 \\ 0 & -0.0042 & 1.0009 & 0.0024 & 0 & 0 \\ 0 & 0 & 0.0013 & 0.9979 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9972 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9953 \end{bmatrix} \\ B &= \begin{bmatrix} 1.0033 & 0 & 0 & 0 & 0 & 0 & -0.2175 \\ 0 & 1.0460 & 0 & 0 & 0 & 0 & -0.0788 \\ 0 & 0 & 1.0326 & 0 & 0 & 0 & -0.0020 \\ 0 & 0 & 0 & 0.4798 & 0 & 0 & -0.0669 \\ 0 & 0 & 0 & 0 & 0.8882 & 0 & 0.1273 \\ 0 & 0 & 0 & 0 & 0 & 1.1699 & -0.1792 \end{bmatrix} \end{aligned} \tag{4.45}$$

In addition, we design an output matrix for the system as follows:

$$C = \begin{bmatrix} 0.992 & 0.00018 & 0 & 0 & -0.0001 & 0 \\ 0.0023 & 1.3 & 0.0043 & 0 & 0 & 0 \\ 0 & -0.0042 & 1.0109 & 0.0024 & 0 & 0.201 \\ 0 & 0 & 0.0013 & 0.989 & 0.00031 & 0.64 \\ 0 & 0 & 0 & 0 & 0.923 & 0.3 \end{bmatrix} \tag{4.46}$$

We can easily verify that the system described by the above matrices is controllable and observable. As highlighted in the previous section, the most important design parameters for the optimisation are the weighting matrices Q and R. Before performing Bayesian Optimisation to obtain the best values, we chose as a baseline, identity matrices of appropriate dimensions, namely $Q = I_p$ and $R = I_m$.

In BAAM systems, the multiple heaters located across the barrel heat the material in steadily increasing temperatures until the material reaches a suitable temperature for extrusion. For our simulations, we assume the polymer used is PLA, whose melting temperature ranges between 160 and 180 °C. With this in mind, we set the reference signal to be $r(t) = (150, 160, 170, 175, 180), \forall t \geq t_0$. This means that the reference generating matrix is $F = I_5$. This reference will be used throughout the simulations, both for the model-based controller and the data-driven case.

4.6.1 State Observer based Output Feedback

We first focus on the approach introduced in Section 4.3, starting with the design of the State Observer, for which we choose $\tau = 0.002$. For the controller design, we use a discount factor $\gamma = 0.99$ and an error threshold for convergence $\epsilon = 0.01$. We arbitrarily initialise the state estimate with $\hat{x}(t_0) = (50, \dots, 50)$ and the control policy with random normally distributed numbers as $K^0 = (k_1, \dots, k_{n+p})$ where $k_i \sim \mathcal{N}(0, 10)$. Finally, even though we assume that the system state is not directly available, we need to initialise it so that it can be updated and measured during the simulations. That starting point is chosen to be $x(t_0) = (20, \dots, 20)$. Table 4.1 summarises all parameter values used for the observer-based simulations.

Table of parameters - Observer based output feedback		
Parameter	Symbol	Value
Tracking reference	r	$(150, 160, 170, 175, 180)$
Initial state	$x(t_0)$	$(20, \dots, 20)$
Initial observer state	$\hat{x}(t_0)$	$(50, \dots, 50)$
Observer pole	τ	0.002
Error threshold	ϵ	0.01
Initial control gain	$K^0 = \{k_i\}_{i=1}^{n+p}$	$k_i \sim \mathcal{N}(0, 10)$

Table 4.1: Parameters chosen for observer-based control implementations. The information listed includes the parameter names, the corresponding symbol and the value that was chosen.

Figure 4.2 shows the output trajectories obtained as a result of applying the model-based controller for 100 time steps, as described in Algorithm 8. The controller manages to bring all measurements within 0.03°C of the optimal. The measured temperatures arrive within 0.1 degrees of the target within only 10 steps and remain there. The values that the measurements finally converge to are $y^* = (149.9798, 159.9932, 169.9795, 174.9815, 179.9859)$. The error between the final measurement values and the reference, calculated through the Euclidean norm, is $\|r - y^*\|_2 = 0.0376$. The value of the performance index obtained by this controller, approximated as the discounted sum of the first 100 cost terms is 183,362.5. When calculated over 1000 steps, it reaches 183,436.2. The performance index trajectory for 100 time steps can be seen in Figure 4.3a. The good performance of the controller is due to the fact that the observer approximates the system state very closely, thanks to the knowledge of the system dynamics. The estimation error, calculated as the Euclidean norm $e(t) = \|\hat{x}(t) - x(t)\|_2$, is decreasing with each time step, reaching the value 0.008 in 100 steps. It is worth noting that the closer the initial state estimate is to the initial state, the faster the error decreases. This decreasing trend can be seen in Figure 4.3b where we plot the estimation error for the first 100 time steps.

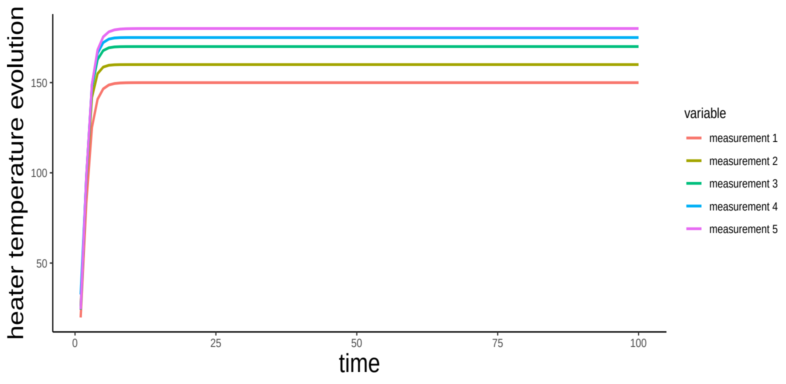
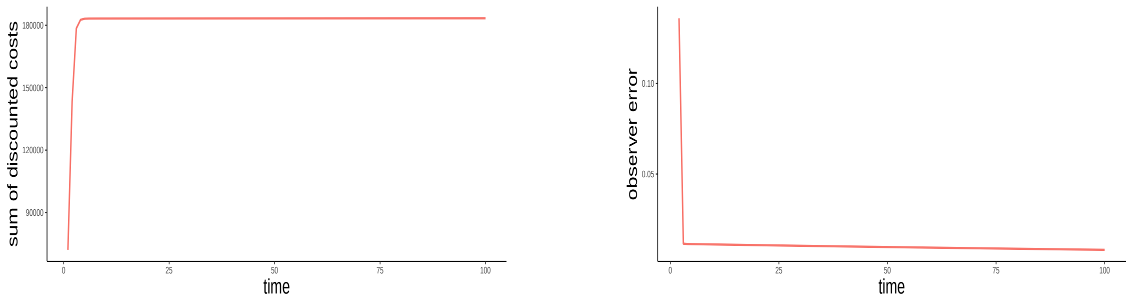


Figure 4.2: Observer-based Output Feedback. The system output achieves temperatures very close to the optimal within a few steps.



(a) Sum of discounted cost terms for 100 time steps.

(b) Estimation error for the system state by the observer.

Figure 4.3: Performance index and state estimation error for the Observer-based Output feedback.

We perform Bayesian Optimisation to obtain better choices for the weighting matrices Q and R and study the effect they have on our results. For simplicity, as is common practice, we assume these matrices are diagonal and positive definite. Through some trial and error attempts, we find that the best range of values for the diagonal entries is positive numbers ≤ 1. To avoid numerical instability, we assume a lower limit for these values to be 0.01. Additionally, we want to put a higher emphasis on the tracking of the reference signal. This is expressed by choosing appropriate upper limits for the entries of each matrix. Specifically, we chose 1 as the upper limit for Q and 0.4 as the upper limit for R.

We run the Bayesian Optimisation procedure as described in Section 4.5 with 50 initial randomly sampled points for 100 iterations. The optimal values obtained for the matrices Q and R rounded up to 3 decimal points are

$$Q^* = \begin{bmatrix} 0.943 & 0 & 0 & 0 & 0 \\ 0 & 0.762 & 0 & 0 & 0 \\ 0 & 0 & 0.542 & 0 & 0 \\ 0 & 0 & 0 & 0.420 & 0 \\ 0 & 0 & 0 & 0 & 0.514 \end{bmatrix} \tag{4.47}$$

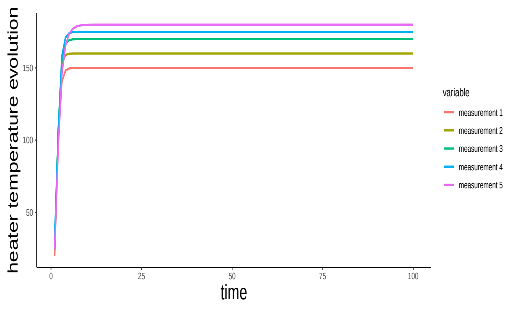
and

$$R^* = \begin{bmatrix} 0.300 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.270 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.281 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.092 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.054 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.269 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.318 \end{bmatrix} \tag{4.48}$$

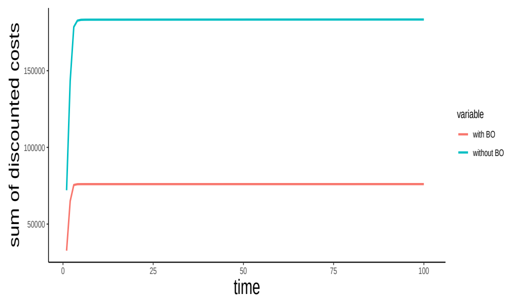
Using the optimised values Q^* and R^* we obtain the updated trajectories seen in Figure 4.4a. The values that the outputs converge to are: $(149.9940, 159.9946, 169.9843, 174.9873, 179.9834)$ which means they all arrive within 0.02°C of the optimal (cf 0.03°C for the previous version). The error of the final values from the optimal is $\|r - y^*\|_2 = 0.0274$, verifying that the tracking performance is improved. Additionally, the performance index is significantly lower in this case, as can be seen in Figure 4.4b. Specifically, the performance index obtains the value 76,060.36 after 100 steps and converges to 76,072.6 over 1000 steps. This means that the updated weighting matrices result in a 58.5% reduction of the performance index.

4.6.2 Data generation

When designing and training model-free, data-driven controllers, much like any learning algorithm, it is important to have data that is rich in information. In the case of the algorithm introduced in Section 4.4, the iterative algorithm learns the kernel matrix H through input-output tuples. Matrix H includes information of the system dynamics as seen in Equation (4.34), meaning that learning H is equivalent to learning the system



(a) System output trajectories



(b) Comparison of performance index before and after using Bayesian Optimisation

Figure 4.4: Output trajectories and performance index comparison for the Observer-based Output feedback where the weighting matrices Q and R were obtained through Bayesian Optimisation. The output trajectories are similar to the default case while the performance index is minimised more effectively through the Bayesian Optimisation approach.

model. This means that the problem can also be viewed as a system identification problem [83].

An important advantage of working with simulated data generated from an assumed model is that we can ensure the data is appropriate to train our algorithms. The way to achieve that is by using a well-designed input signal to the system. Firstly, we need the system output to stay bounded and not diverge towards infinite values, in which case the data would not be usable. To that end, we can design the input to be in feedback form

$$u(t) = -K_{data}x(t) \tag{4.49}$$

where the feedback gain K_{data} is stabilising. In the case where the training data is obtained from a real process, this step is not applicable, however, the process would be tuned from the manufacturer to avoid extreme divergence. In the case of the system introduced in 4.45 and 4.46, we can use the *Matlab Control System Toolbox* to obtain a stabilising gain. In particular, two of the most popular approaches to stabilising a system's behaviour are the Pole Placement method and the Linear Quadratic Regulator

(LQR). Using the LQR approach, we obtain the gain:

$$K_{data} = \begin{bmatrix} 0.7395 & -0.0076 & -0.0003 & -0.0264 & 0.0194 & -0.0170 \\ -0.0076 & 0.7430 & 0.0031 & -0.0093 & 0.0068 & -0.0060 \\ -0.0003 & -0.0033 & 0.7599 & 0.0021 & 0.0002 & -0.0002 \\ -0.0126 & -0.0042 & 0.0016 & 1.0971 & 0.0092 & -0.0079 \\ 0.0171 & 0.0058 & 0.0002 & 0.0170 & 0.8179 & 0.0108 \\ -0.0198 & -0.0067 & -0.0002 & -0.0193 & 0.0143 & 0.6823 \\ -0.1525 & -0.0519 & -0.0018 & -0.1412 & 0.1091 & -0.0977 \end{bmatrix} \tag{4.50}$$

Secondly, to ensure the output data is varied enough to efficiently train the control algorithm, we require the input signal to not be 'too simple'. This requirement can be fulfilled by the addition of an appropriate probing noise signal to the input, meaning that it will now be of the form

$$u(t) = -K_{data}x(t) + \omega_{pr}. \tag{4.51}$$

We define the probing noise as a sum of a Gaussian noise signal and sinusoidal signals, of varying ranges and frequencies. Specifically, to ensure great variety in the data, we chose ranges between 10 and 100. The choice of sampling frequencies is more particular, where an appropriate choice is the system's bandwidth, i.e. the part of frequency domain where most of signal's energy is contained. Formally, a system's bandwidth is the frequency at which the magnitude of the system's response drops below 0.707 (-3dB). In the case of multi-input multi-output (MIMO) systems, the bandwidth is the frequency where the maximum singular value of the frequency response crosses 0.707 from below [102]. In Matlab, this value can be easily obtained for any system using the *sigma* function. The bandwidth for our system is 1.65, so we use it as the maximum frequency from which to sample sinusoidal signals. Finally, we add a normally distributed random vector to our noise signal, to add an extra element of randomness. We consider the variance for this vector to be $\sigma = 1.5$. Bringing all the above elements together, we have

$$\begin{aligned} \omega_{pr} &= \omega_1 + 100\sin(\omega_2 t) + 90\sin(\omega_3 t) + 80\sin(\omega_4 t) + 70\sin(\omega_5 t) \\ &\quad +60\sin(\omega_6 t) + 50\sin(\omega_7 t) + 40\sin(\omega_8 t) + 30\sin(\omega_9 t) \\ &\quad +20\sin(\omega_{10} t) + 10\sin(\omega_{11} t) \end{aligned} \tag{4.52}$$

where

$$\omega_1 \sim \mathcal{N}(0, \sqrt{\sigma} I_7),\ \omega_2 = (16.5, \dots, 16.5) \in \mathbb{R}^7,\ \text{and } \omega_3, \dots \omega_{11} \sim \mathcal{U}_7(0, 1.65). \tag{4.53}$$

4.6.3 Data-driven Output Feedback

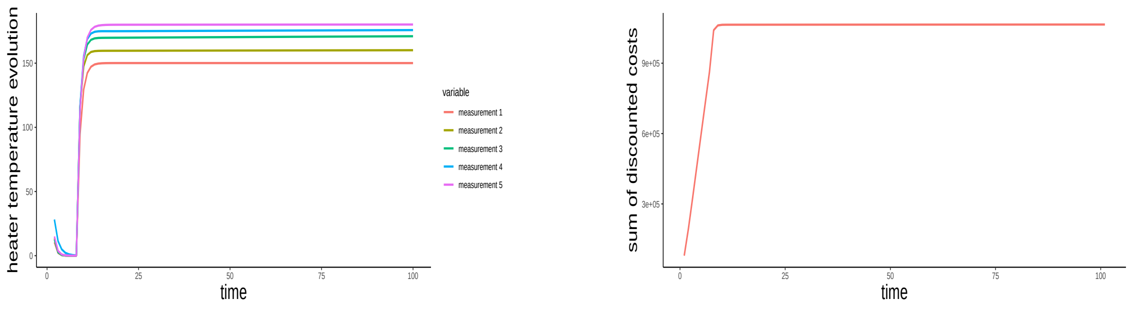
Using the input signal designed in the previous section, we can obtain data tuples $\{y(t), u(t)\}$ to train the data-driven RL based controller introduced in Section 4.4. In particular, we generate $M = 15000$ samples and set a maximum number of iterations to be 1000 in case the algorithm does not converge according to the error threshold chosen as $\epsilon_{rl} = 0.001$. As the system is fully observable, we choose the time horizon N for the augmented data tuples to be $N = 6$. We choose the discount factor to be $\gamma = 0.99$ and the regularisation parameter to be $\mu = 0.0001$. Finally we initialise the kernel matrix H as the identity matrix of appropriate dimensions, namely $H^0 = I_{(N+1)m+(N+1)p}$. We run two sets of simulations, one with the weighting matrices Q, R chosen arbitrarily, and one using the values for Q and R obtained through Bayesian Optimisation.

Table of parameters - Model based methods		
Parameter	Symbol	Value
Universal parameters		
Tracking reference	r	$(150, 160, 170, 175, 180)$
Initial state	$x(t_0)$	$(50, \ldots, 50)$
Discount factor	γ	0.99
Error threshold	ϵ_{rl}	0.001
Regularisation parameter	μ	0.0001
Time horizon	N	6
Initial kernel matrix	H^0	$I_{(N+1)m+(N+1)p}$

Table 4.2: Parameters chosen for data-driven RL based implementations. The information listed includes the parameter names, the corresponding symbol and the value that was chosen.

Firstly, we focus on the case of the arbitrarily chosen weighting matrices Q and R. We chose them to be identity matrices of appropriate dimensions, namely $Q = I_5$ and $R = I_7$. Figure 4.5 shows the output trajectories obtained through the data-driven controller described in Algorithm 9, and the corresponding performance index. The measurements converge to values within 1°C of the optimal, specifically to the vector $y^* = (150.0857, 160.1181, 171.0027, 175.7926, 180.1416)$. The performance index, defined as the infinite sum of discounted costs, is estimated by a large finite sum which for 1000 steps is 1,074,985. We plot it for the first 100 steps, in which period the sum has obtained the value 1,065,077.

We now perform Bayesian Optimisation on Q and R using the same upper and lower bounds for the search space as in Section 4.6.1, namely we chose 0.1 as the lower bound for both matrices and chose 1 as the upper bound for Q and 0.4 as the upper bound for R. We generate a grid of 50 random initial samples and then train the algorithm for 100



(a) System output trajectories (b) Performance Index

Figure 4.5: Output trajectories and the performance index the data-driven Output feedback algorithm using input-output data, where the weighting matrices Q and R were chosen as identity matrices of appropriate dimensions.

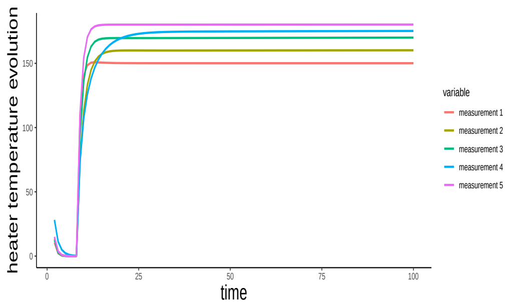
iterations. The optimal values found for Q and R rounded to 3 decimal points are

$$Q^* = \begin{bmatrix} 0.174 & 0 & 0 & 0 & 0 \\ 0 & 0.056 & 0 & 0 & 0 \\ 0 & 0 & 0.010 & 0 & 0 \\ 0 & 0 & 0 & 0.010 & 0 \\ 0 & 0 & 0 & 0 & 0.160 \end{bmatrix} \tag{4.54}$$

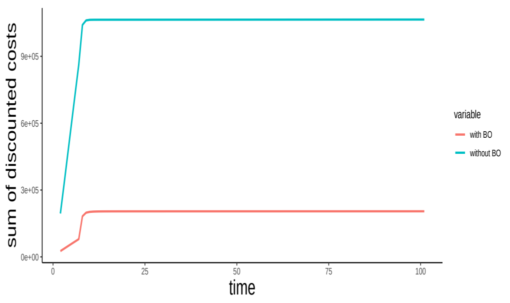
and

$$R^* = \begin{bmatrix} 0.145 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.389 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.020 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.116 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.099 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.316 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.010 \end{bmatrix}. \tag{4.55}$$

With these values for Q and R, Algorithm 9 obtains a controller that produces the output seen in Figure 4.6a. The measurements converge to the vector $y^* = (150.069, 160.154, 170.011, 175.199, 180.176)$ after 100 steps, namely they converge within 0.2°C of the optimal. This means that tracking performance is significantly improved compared to the previous case. In addition, the overall cost is much lower, as can be seen by the comparison of the two performance indices, before and after Bayesian Optimisation, in Figure 4.6b. In particular, the new infinite sum of discounted costs, when approximated by the first 100 steps, obtains the value $204,635.7$, while the value reached after 1000 steps is $206,358.8$. This means that Bayesian Optimisation achieves an 80.8% decrease in the cost, as expressed through the performance index.



(a) System output trajectories after Bayesian Optimisation



(b) Cost Comparison before and after Bayesian Optimisation

Figure 4.6: (a) Output trajectories where the weighting matrices Q and R were obtained through Bayesian Optimisation. (b) A comparison of the performance index before and after the optimisation of Q and R. There is a significant decrease of the total cost after Bayesian Optimisation.

It is important to note that in Figure 4.6b, the two quantities being compared are being weighted by the corresponding weighting matrices Q and R. The matrices resulting from Bayesian Optimisation are populated by elements that are significantly smaller than 1, which implies that the weights applied to the cost terms are much smaller in the Bayesian Optimisation case. This makes the comparison biased and does not provide clarity on the improvement that Bayesian Optimisation offers. To get a clear picture of this, we look at the individual unweighted cost terms achieved in both cases, after convergence. Table 4.3 displays the input, tracking and total costs after the first 10,000 steps in the simulation, after convergence is achieved. By input, tracking and total costs we refer to the terms $\|u\|_2$, $\|r-y^*\|_2$ and $\|u\|_2+\|r-y^*\|_2$ respectively, where $\|\cdot\|_2$ denotes the Euclidean norm. The comparison of the individual unweighted costs proves that Bayesian Optimisation achieves a significant reduction in all costs, namely an 87.5% reduction of the total cost and an 87.8% and 86.6% reduction of the input and tracking costs respectively.

Comparison of costs		
Cost	Before B.O.	After B.O.
Input	12.840	1.726
Tracking	68.309	8.363
Total	81.149	10.089

Table 4.3: Comparison of unweighted costs before and after Bayesian Optimisation. The weighting matrices obtained through Bayesian Optimisation result in learning a controller that achieves much lower costs after convergence.

Different implementations might yield slightly different numerical results, depending on the randomness introduced into the algorithm. In particular, randomness is introduced in the data generation step, as well as the sampling for Bayesian Optimisation. However, as long as the learning algorithm is designed according to Algorithm 9 and the data is generated appropriately, as described in Section 4.6.2, the resulting controller will successfully track the reference signal, and Bayesian Optimisation will consistently reduce the cost while maintaining good tracking performance.

4.7 Conclusion

In this work, we investigated the problem of temperature LQT control within the extruder of a Big Area Additive Manufacturing system, through tracking a reference signal. In particular, we focused on the case where direct access to the system state is not possible, instead, some measurements of the state are available. We considered a state-space model identified and validated based on real data describing the system dynamics, and used it to simulate the system's behaviour and response to different controllers. The lack of access to the system state introduces the need for a state estimation step, to better inform the controller. Assuming the system model is available, the task of state estimation can be performed through the implementation of a state observer. The resulting state estimate is then used to inform a feedback controller, designed using model-based Reinforcement Learning Techniques, with the Bellman equation being the main tool.

Next, we investigated solving the Output Feedback problem in a model-free and data-driven way. We used the state space temperature model for data generation but no longer assumed it to be available for the control design. We instead used Least Squares to train a controller directly from process data, while incorporating a state estimation procedure in the algorithm. As with many learning algorithms, the data quality is really important when training data-driven controllers. For this reason, we provide an in depth discussion on the input signals that would generate "good" data when applied to the system. Finally, we explored the use of Bayesian Optimisation to improve the choice of weighting parameters in the performance index.

We presented our results before and after the implementation of Bayesian Optimisation, both for the model-based and data-driven controllers. For the model-based scenario, the obtained controller successfully tracks the reference signal, which was expected given the access to full information about the system. We were able to achieve comparable results while alleviating the need for knowledge of the system dynamics. This is a very encouraging result, considering how challenging it is to have accurate models of AM processes. Our results also indicate the improvement Bayesian Optimisation offers, as expressed through the significant reduction of the performance index, namely a 58.5% improvement in the model-based case and an 80.8% improvement in the data-driven case. The improvement can also be highlighted by examining the individual unweighted costs after convergence, where Bayesian Optimisation improves the total cost by 87.5%.

While our approach is a valuable tool to solving the LQT problem, it presents certain limitations. The basis of our approach is the quadratic form of the objective function (performance index). When defining a control problem, the objective function can take many forms, especially when introducing constraints into the problem. For example, we can impose constraints on the output response of the system so that the values follow steadily increasing trajectories and the dip seen in Figures 4.6a and 4.5a is avoided. Another important drawback of our approach stems from its mathematical and computational complexity. In particular, the implementation of Least Squares for the estimation of kernel matrix H involves matrix operations between very large matrices. Normally, the more complex the system to be controlled is, the larger the state space needed to describe it is. This means that the matrices involved in the introduced method can become significantly larger than the ones in our case study.

The above concerns can be addressed through the introduction of more abstract Reinforcement Learning algorithms, based on incorporating Deep Neural Networks [43]. These are powerful function approximators that can handle large amounts of data, thus being able to handle much more complicated systems and alieviating the need for a particular form to the value function. These approaches are widely known as Deep Reinforcement Learning methods. A very popular one among them is the Actor-Critic framework [46], where one network approximates the value function (critic) and another produces the new action to be taken (actor). We leave these approaches as future work which we explore elsewhere.

5 Deep Reinforcement Learning based Temperature Control for Material Extrusion

Abstract

With the wider adoption of Additive Manufacturing (AM) both in industrial settings and in everyday life, comes a strong need to ensure the safety, repeatability, and cost efficiency of the process. This need can be addressed by the development of efficient monitoring and control algorithms to ensure all aspects of the process are performing according to certain expectations. The lack of exact and simple process models in AM makes the use of traditional model-based control approaches inapplicable. We propose a Deep Reinforcement Learning approach using the Deep Deterministic Policy Gradient (DDPG) algorithm, to learn an optimal controller without any prior knowledge of the system's dynamics, purely based on available measurements. Our results indicate that we can have efficient controllers that satisfy a constrained optimisation criterion and learn directly from process data.

5.1 Introduction

Additive Manufacturing (AM) is the technology currently at the front stage of the manufacturing field. It has revolutionised the field, thanks to its flexibility in building complicated structures with minimal waste of material. AM technologies are being integrated into various applications such as the design of parts in aerospace industries [86, 103, 104] and the design of biomedical devices [105, 106]. They are also becoming increasingly popular as educational tools used by institutions, as well as becoming a favourite for hobbyists, that now have the power of manufacturing in their own hands. Despite its overall promising future, there remains one main barrier to the wide adoption of AM, the lack of efficient monitoring and feedback control of the process [8, 9, 107].

Traditionally, the design of feedback controllers for AM would be approached through standard Control Theory techniques [19, 21, 22, 24, 26]. However, the scarcity of accurate models of AM processes has inspired the adoption of Reinforcement Learning (RL) [41,42] to solve control problems by formulating them as sequential decision-making problems. An important extension to RL is Deep Reinforcement Learning (DRL) [77] which involves the incorporation of Deep Neural Networks (DNNs) [49] as function approximators, allowing RL to scale decision-making problems with large state and action spaces that were previously intractable. DRL offers another advantage over standard RL, the ability to handle continuous actions, something that many RL approaches by design cannot do. Some previous work has explored applying RL concepts, mainly value functions and the Bellman Optimality Equation [16], to solve Optimal Control problems when the value function can be assumed to have a quadratic form [79, 80]. Our previous work involved applying these concepts to controlling the temperature in Material Extrusion AM systems both in the fully observable case [88] and when the system state needs to be estimated [108, 109]. However, while this algorithm can achieve very good performance, the assumption of a particular form for the value function is an important limitation that we are aiming to overcome in this work. The use of DRL for continuous control can achieve this, as these approaches involve approximating the value function. The flexibility and adaptability of DNNs allow for value functions that may be non-linear, involve additional constraints etc. Two of the most popular DRL algorithms that have been recently developed are the Proximal Policy Optimisation (PPO) algorithm [59] and the Deep Deterministic Policy Gradient (DDPG) algorithm [60]. Some early attempts have been made at using these algorithms in the context of AM. Specifically, PPO has been used for optimising the melt pool depth in [61] and tool path in [62]. An extension of DDPG called Twin Delayed DDPG [110], has been used to control the motion of a Robot arm AM system [63] and a tower-crane AM system [64].

In this work, we explore the use of the DDPG algorithm for controlling the temperatures within the extruder of a Big Area Additive Manufacturing (BAAM) system that uses Material Extrusion technology. We use a state-space model of the heating system introduced in recent literature [69] to simulate the behaviour of the system and produce data to train the data-driven controller. The reason behind the use of simulations is the need to interact with the system to produce new data as the optimal policy gets updated. Contrary to other Machine Learning approaches that can be trained on previously acquired data, RL agents need to interact with their environment during training. To interact and explore their environments efficiently, they need to try a wide range of actions and observe the corresponding rewards. If this process were to happen in a real-life system, there is the risk of exploring actions that can damage the machine and waste a lot of material and resources.

In Section 5.2 we introduce the BAAM temperature system to be controlled as well as the control objective. Section 5.3 consists of the main ideas of the DDPG algorithm and its adaptation to solving the temperature regulation problem. In Section 5.4 we provide the details of our simulations and the results obtained from them. Finally, in 5.5 we offer some concluding remarks and future research directions.

5.2 Material extrusion temperature model and control

We consider the heating system within the extruder of a Big Area Additive Manufacturing (BAAM) system. BAAM technology involves the adaptation of Material Extrusion to large-scale manufacturing with the use of polymers and composite materials. Due to the large volume of material used in BAAM applications, the heating process requires multiple steps, to ensure even melting of the material before extrusion. This is achieved by the inclusion of multiple heaters along the extruder that melt the material in stages, achieving the ideal extrusion temperature by the time the material reaches the nozzle.

A typical BAAM extruder consists of 5 main parts, the hopper, the screw, the barrel, the hose and the nozzle, as can be seen in Figure 5.1. The material gets fed into the extruder through the hopper and gets pushed along the barrel with the screw, which gets rotated with the help of an AC motor. The heating of the material takes place in the barrel, specifically in 4 consecutive heating zones, then gets fed into the hose, where it is further heated, and finally reaches the nozzle where it reaches its final melting temperature for extrusion. The temperature of the heating zones is dependent on the heat input applied to each of them and their neighbouring zones, as well as the input from the AC motor. A state space temperature model of this temperature system was introduced in [69] and has the form

$$x(t+1) = Ax(t) + Bu(t) \tag{5.1}$$

where $x(t) \in \mathbb{R}^6$ is the system state at time t, representing the temperatures in each thermal zone, and $u(t) \in \mathbb{R}^7$ is the system input at time t, namely the heat applied to the 6 zones and the motor input. $A \in \mathbb{R}^{6\times 6}$ and $B \in \mathbb{R}^{6\times 7}$ are the state and input matrices respectively that have been estimated through system identification methods from real process data in [69].

We aim to control the temperatures within the extruder in order to ensure the proper gradual melting of the material. In other words, our goal is to design a controller of the system state $x(t)$ so that it tracks a certain reference signal $r(t) \in \mathbb{R}^6$ representing the ideal temperatures for each thermal cell. This objective can be expressed in terms of a

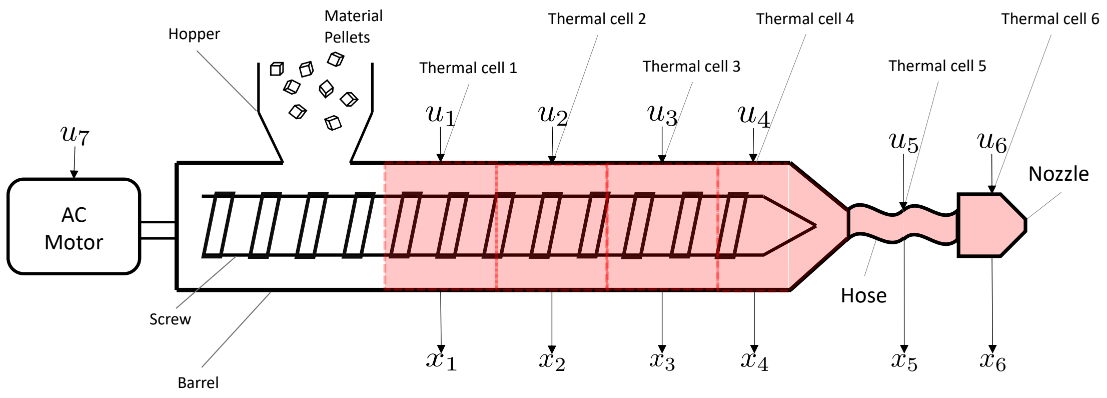


Figure 5.1: Heating system in the extruder of a BAAM system. The heater consists of 6 heating zones, or thermal cells. The temperature at each thermal cell gets influenced by its neighbouring cells as well as the motor rotating the extruder screw.

quadratic performance index to be minimised (subject to (5.1))

$$J(x(0), \mathbf{u}) = \sum_{t=0}^{\infty} \gamma^t \left\{ [x(t) - r(t)]^T Q [x(t) - r(t)] + u^T(t) R u(t) \right\} \tag{5.2}$$

where $\mathbf{u}$ is the control policy to be optimised, $Q \in \mathbb{R}^{6\times 6}$ is a weighting matrix for the tracking error, and where we also include a cost term associated with the input weighted by matrix $R \in \mathbb{R}^{7\times 7}$. We also assume the time horizon has not been predefined, so we consider it to be infinite, which creates the need for a discount factor $0 < \gamma < 1$ that attributes a higher importance to immediate costs over future ones. Furthermore, we impose constraints upon the upper and lower limits of the temperature. The physical interpretation of these constraints can be attributed to the machine itself, the cost of achieving such extremes or the strain caused to the material by excessive heating and cooling. Consider b_u to be the upper limit and b_l the lower limit for the state. Then the constrained optimisation problem can be written as

$$\begin{aligned} \min \quad & J(x(0), \mathbf{u}) \\ \text{subject to} \quad & x(t) \leq b_u, \ \forall t \\ & x(t) \geq b_l, \ \forall t. \end{aligned} \tag{5.3}$$

In the unconstrained case, a formula for the analytical solution to the minimisation problem can be found, due to the convenient quadratic form of the performance index. The Least Squares algorithm can then be used to estimate the terms of the formula, using process data [88]. With the complexity of the problem increased by the introduction of constraints, a more sophisticated algorithm is needed to learn the optimal policy.

5.3 Deep Deterministic Policy Gradient

The Deep Deterministic Policy Gradient (DDPG) [60] is a powerful Deep Reinforcement Learning algorithm capable of handling continuous state and action spaces, complicated environments and value functions. This algorithm combines two of the most popular DRL frameworks, the Actor-Critic framework [46] and Q-learning [44] or more specifically Deep Q-learning [77].

DDPG consists of two Deep Neural Networks, the Actor and the Critic. The Actor is tasked with learning the optimal policy and the Critic with learning the optimal value function. During training, the Actor chooses actions to perform at each state, and the Critic judges how good that action was, by producing an estimate of the value function for that particular state-action pair. By action in the context of RL, we refer to the system inputs in Control Theory terms. In what follows, we will be using the two terms interchangeably. The action is deterministic in terms of the state, which means that at each state the action to be taken is the same every time the state is visited. In classic RL and DRL problems, the value functions quantify how good a state-action pair is through the rewards received. In our application, however, instead of acquiring rewards that we intend to maximise, we are burdened with costs that need to be minimised. Specifically, following the definition of the constrained optimisation problem (5.3), the cost of choosing action $u(t)$ at state $x(t)$ is

$$c(t) = [x(t) - r(t)]^T Q[x(t) - r(t)] + u^T(t)Ru(t) + \lambda_1(x(t) - b_u) + \lambda_2(b_l - x(t)) \tag{5.4}$$

where λ_1, λ_2 are constants, serving the same purpose as Lagrange multipliers in Optimal Control Theory [10].

The value function is defined as the expected sum of discounted future costs, starting from state $x(t)$, choosing action $u(t)$ and then following the deterministic policy $\mu(x)$ onwards

$$q_\mu(x(t), u(t)) = \mathbb{E}_\mu[c(t) + \gamma c(t+1) + \gamma^2 c(t+2) + \dots |x(t), u(t)]. \tag{5.5}$$

The main tool used in RL to optimise value function is the Bellman Equation [16]. It is the mathematical expression of the Optimality Principle, which states that an optimal policy for state $x(t)$ will also be optimal for the state resulting from the action $u(t)$

$$q_\mu(x(t), u(t)) = \mathbb{E}\left[c(t) + \gamma q_\mu(x(t+1), \mu(x(t+1)))\right]. \tag{5.6}$$

In order to train the Actor and the Critic networks, we employ two important techniques associated with training Deep Q Networks (DQN), the Replay Buffer and the incorporation of Target Networks. The target networks are two additional DNNs, identical to the Actor and the Critic in structure, used to generate the target values for training. In particular, considering that the Critic is estimating the value function q_μ and the Actor is estimating the policy μ, if we were to use them to calculate the right-hand side of the Bellman equation (5.6), it would lead to great instability. The cause of this instability is the overestimation of the Q-values, an effect commonly referred to as "catastrophic forgetting". What this means is that the network gets continuously updated, not allowing itself time to consider the effect of the actions that have recently taken place. To circumvent this issue, we use two target networks, whose structure is identical to the original networks. The use of target networks is a technique first introduced in [50] and has since been widely adopted in the training of Deep RL agents. Instead of updating them after each training instance, we use "soft updates" to update their weights. Let θ_a, θ_c be the weights of the actor and the critic respectively, and θ'_a, θ'_c the weights of the respective target networks. The weight updates happen according to

$$\begin{aligned} \theta'_a &\leftarrow \tau\theta_a + (1-\tau)\theta'_a \\ \theta'_c &\leftarrow \tau\theta_c + (1-\tau)\theta'_c \end{aligned} \tag{5.7}$$

where τ is a very small positive number.

The Replay Buffer is a tool meant to aid us in training the networks on data that are i.i.d., or as close to i.i.d., as possible. At each iteration step t, we store a tuple of the form $\{x(t), u(t), c(t), x(t+1)\}$ in a memory buffer, and sample a minibatch of size N of tuples to train the DNNs with. For each tuple in the minibatch, $\{x_i, u_i, c_i, x'_i\}$, calculate the right hand side of (5.6) using the target networks

$$y_i = c_i + \gamma q^{target}(x'_i, \mu^{target}(x'_i)) \tag{5.8}$$

which will be the target value for the Critic, associated with the input pair (x_i, u_i). The critic gets trained by back-propagating the error $(q(x_i, u_i) - y_i)^2$ for each $i \in 1, \dots N$, where $q(x_i, u_i)$ is the current value function estimate produced by the critic. To train with the entire minibatch, we calculate all the target values according to (5.8) and back-propagate the mean squared error $L = \frac{1}{N}\sum_i (q(x_i, u_i) - y_i)^2$.

The Actor on the other hand gets updated with the use of the deterministic policy gradient [81]. Let θ_a be the Actor's weights and θ_c be the Critic's weights. Considering a minibatch of size N, the sampled gradient of the performance index in terms of the policy is

$$\nabla_{\theta_a} J \approx \frac{1}{N}\sum_i \nabla_a q(x, a|\theta_c)|_{x=x_i, a=\mu(x_i)} \nabla_{\theta_a}\mu(x|\theta_a)|_{x_i} \tag{5.9}$$

where μ is the current learnt policy from the actor. From an implementation point of view, we first calculate the recommended actions $\mu(x_i)$ from the Actor for each state x_i in a minibatch and then obtain the value function estimate from the Critic for each pair $(x_i, \mu(x_i))$. The mean of the value function estimates $q(x_i, \mu(x_i))$ gets backpropagated through the actor network to update its weights towards the minimisation of q.

A final but very important aspect of training a DDPG agent, or any DRL agent, is the environment exploration. For the agent to effectively learn the environment's dynamics, it needs to sufficiently explore it, something that might not be possible if the actions taken are only the ones generated by the Actor. For this reason, we include a noise process $\mathcal{M}$ to the action at time t:

$$u(t) = \mu(x(t)) + \mathcal{M} \tag{5.10}$$

The noise process $\mathcal{M}$ is chosen according to the application at hand. Integrating all of the above elements, Algorithm 10 gives the steps to implementing the DDPG algorithm within a certain environment.

5.4 Simulation setup and results

We set up a simulation environment of the extruder temperatures using the model (5.1) and the system matrices identified in [69] through system identification methods, using process data. The state and input matrices used are:

$$A = \begin{bmatrix} 0.992 & 0.0018 & 0 & 0 & 0 & 0 \\ 0.0023 & 0.9919 & 0.0043 & 0 & 0 & 0 \\ 0 & -0.0042 & 1.0009 & 0.0024 & 0 & 0 \\ 0 & 0 & 0.0013 & 0.9979 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9972 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9953 \end{bmatrix} \tag{5.11}$$

$$B = \begin{bmatrix} 1.0033 & 0 & 0 & 0 & 0 & 0 & -0.2175 \\ 0 & 1.0460 & 0 & 0 & 0 & 0 & -0.0788 \\ 0 & 0 & 1.0326 & 0 & 0 & 0 & -0.0020 \\ 0 & 0 & 0 & 0.4798 & 0 & 0 & -0.0669 \\ 0 & 0 & 0 & 0 & 0.8882 & 0 & 0.1273 \\ 0 & 0 & 0 & 0 & 0 & 1.1699 & -0.1792 \end{bmatrix} \tag{5.12}$$

We design the environment to be bounded, with the range of admissible actions to be $(0, 1)$ and the range of temperatures the heaters can reach to be $(50, 300)$ °C.

We consider the extruded material to be PLA, whose melting point is between 170 °C and 180 °C. We set the reference signal $r(t) = (150, 160, 165, 170, 175, 180)$ for all t,

Algorithm 10: DDPG algorithm

```
   Result: Actor and Critic Networks
   Input: Number of episodes $n$, Max steps per episode $m$, Discount factor $\gamma$,
          Batch size $N$, Soft updates parameter $\tau$, Initial state $x(0)$, Noise term
          process $\mathcal{M}$
1  Initialise Actor $\mu(x|\theta_a)$ and Critic $q(x,u|\theta_c)$with weights $\theta_a$ and $\theta_c$;
2  Initialise target networks $\mu^{target}$ and $q^{target}$ with weights $\theta'_a \leftarrow \theta_a$ and $\theta'_c \leftarrow \theta_c$;
3  Initialise replay buffer;
4  for episode = 0,n-1 do
5      for t = 0,m-1 do
6          Choose action according to $u(t) = \mu(x(t)) + \mathcal{M}$;
7          Receive cost $c(t)$ from environment;
8          Sample the next state $x(t+1)$ from environment;
9          Augment the replay buffer by the tuple $\{x(t), u(t), c(t), x(t+1)\}$;
10         Sample a batch of size $N$ from the replay buffer;
11         for i in 1:N do
12             Extract $i$-th tuple $\{x_i, u_i, c_i, x'_i\}$;
13             Set the target for the critic $y_i = c_i + \gamma q^{target}(x'_i, \mu^{target}(x'_i))$ ;
14             Obtain recommended actions $\mu(x_i)$ from the Actor and corresponding
                 value function estimates from the Critic $q_i = q(x_i, \mu(x_i))$
15         end
16         Update the critic by minimising the loss $L = \frac{1}{N}\sum_i (q(x_i, u_i) - y_i)^2$;
17         Update the actor by minimising the mean $q_i$ ;
18         Update the target networks:
               $\theta'_a \leftarrow \tau\theta_a + (1-\tau)\theta'_a$
               $\theta'_c \leftarrow \tau\theta_c + (1-\tau)\theta'_c$
19     end
20 end
```

so that the final heating of the material that happens in the nozzle, will bring it to its melting temperature. We further assume that the the upper and lower bounds for the state are $b_u = (220, \dots, 220) \in \mathbb{R}^6$ and $b_l = (110, \dots, 110) \in \mathbb{R}^6$, indicating that it is overly costly for the machine to exceed those bounds. To further test our algorithm's ability to remain within the bounds, before each episode we reset the environment to a state outside the minimum bound, $x(0) = (60, \dots, 60) \in \mathbb{R}^6$. For the cost function (5.4) we choose multipliers $\lambda_1 = \lambda_2 = 0.1$ and the weighting matrices $Q = I_6$ and $R = I_7$. The value function discount factor is chosen to be $\gamma = 0.99$ and the soft updates parameter $\tau = 0.005$.

Both the Actor and the Critic Networks have two fully connected layers of 1000 and 800 nodes respectively. The learning rates, determining how quickly the DNNs' weights get updated, are chosen to be $a = 0.002$ and $\beta = 0.0005$ for the Actor and Critic respectively. The networks get trained for a total of $n = 600$ episodes of $m = 1000$ time steps each using minibatches of size $N = 128$. In order to ensure stable training of the networks, we also standardise the data before training with each minibatch.

In the original DDPG paper [60], the action noise is generated from an Ornstein-Uhlenbeck process [111]. However, a simple normal distribution can be as effective, an assertion supported by empirical evidence that was comprehensively shown in a comparative study [112]. We chose a mean of 0 and a standard deviation of 0.3. In order to provide an even wider variety of data for training, we generate data in a uniformly random manner, within the predetermined environment limits, for the first 30 episodes. This means that we randomly pick an action between $(0, 1)$ for each state, apply it to the system and observe the cost and new state. After the first 30 episodes, the data gets generated from employing actions as in (5.10), with $\mathcal{M} \sim \mathcal{N}(0, 0.3)$. Table 5.1 lists all the parameters chosen for our simulations.

Within 600 episodes of 1000 time steps each, the losses of the two DNNs quickly decrease, showing the convergence to a local minimum. The two losses throughout training can be seen in Figure 5.2.

The agent successfully learns a policy that can bring and maintain the heater temperatures very close to their optimal values, as can be seen in Figure 5.3. In particular, the heaters temperatures converge to the vector $x^*(t) = (143.496, 160.609, 163.017, 157.298, 167.876, 161.216)$, which means all heaters achieve temperatures within 18 °C of the optimal. Additionally, even though all states are initialised outside the desired bounds, they are all brought and kept within the bounds very quickly. It is also important to note that the learnt policy brings the temperatures within 5°C of their final values in 373 time steps and within 1°C in 494 steps.

While these results are very encouraging, they still deviate from the optimal behaviour we have defined for our system. This can be attributed to the sensitivity of the DDPG

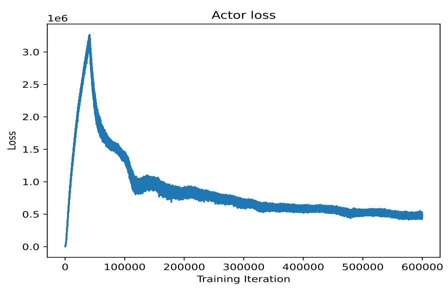
Table of parameters		
Parameter	Symbol	Value
Tracking reference	r	$(150, 160, 165, 170, 175, 180)$
Initial state	$x(0)$	$(60, \ldots, 60)$
State upper bound	b_u	$(220, \ldots, 220)$
State lower bound	b_l	$(110, \ldots, 110)$
Multiplier for upper bound	λ_1	0.1
Multiplier for lower bound	λ_2	0.1
Tracking error weighting matrix	Q	I_6
Input weighting matrix	R	I_7
Discount factor	γ	0.99
Batch size	N	128
Soft updates parameter	τ	0.005
Actor learning rate	α	0.002
Critic learning rate	β	0.0005
Noise term	$\mathcal{M}$	$\sim \mathcal{N}(0, 0.3)$
Number of episodes	n	600
Max steps per episode	m	1000

Table 5.1: Parameter values chosen for the implementation of the DDPG algorithm.

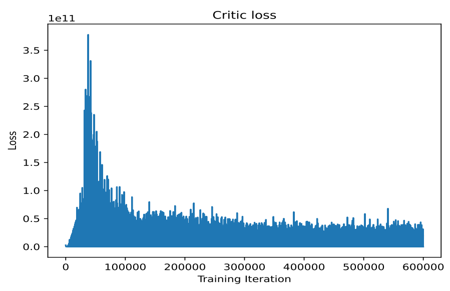
algorithm to the choice of hyper-parameters, inherited by the use of DNNs. DNNs require fine-tuning of their hyper-parameters to adjust to each individual task. In particular, the range of the state and action spaces, the complexity of the system's dynamics, and the form of the value function are the most important factors that influence the problem formulation. As a result, different choices for the above factors would require completely different choices for the design parameters. Furthermore, the performance of the DDPG algorithm, as with any Machine Learning algorithm, is greatly influenced by the quality of the data used for training. While we ensured the data is varied and the inputs used to generate it sufficiently explore the environment, it is evident that more careful consideration of the input choice is required.

5.5 Conclusion

In this work, we addressed the problem of temperature control in the extruder of a Big Area Additive Manufacturing system using Deep Reinforcement Learning. We use a linear state space model as a simulator of the heating system's behaviour and design a data-driven controller that optimises that behaviour according to a constrained optimisation performance index. The controller is designed based on the DDPG algorithm, a very powerful Deep Reinforcement Learning algorithm that learns optimal policies in complicated environments. The algorithm was able to learn a policy that brought and maintained the



(a)



(b)

Figure 5.2: Loss function for the Actor (a) and the Critic (b) networks. Both losses get significantly reduced over time, indicating the successful training of the networks.

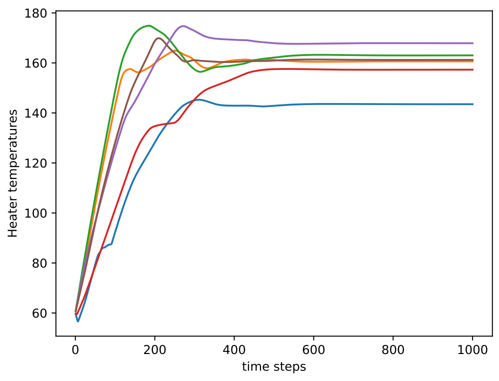


Figure 5.3: Heater temperatures after applying the learned policy for 1000 iterations. All temperatures remain within the constraints imposed by our cost function, and they are all brought and maintained within small error margins of their optimal reference values.

temperatures within a very small error margin of their predefined optimals.

The versatility of the DDPG algorithm allows for training optimal controllers under uncertain conditions, with no prior knowledge about its environment. However, the lack of prior knowledge results in suboptimal behaviour, which is the main drawback of this approach. It is important to highlight the sensitivity that this algorithm has to the configuration of the neural networks as well as the inputs used to generate the data for training. Given the large number of hyper-parameters involved in this method, as well as the wide variety of "tricks" that could be employed to boost training, the improvement of the algorithm's performance remains an open research question.

There are several extensions to our work worth exploring. In the context of simulation work, using more complex nonlinear models as simulators would result in a vastly more complicated environment and would further test the algorithm's capabilities to learn optimal policies. An additional intriguing extension would be the incorporation of more advanced optimisation criteria, namely introducing additional non-linearities and constraints in the problem formulation. Finally, testing the learning capabilities and performance of this algorithm in a real-life scenario would be an exciting research direction. We leave these extensions as future work which we will pursue elsewhere.

6 Conclusion

In this thesis we have introduced a number of methods for closed-loop temperature control in Additive Manufacturing (AM). With the rise and wider adaptation of AM, there is an increasing need for effective feedback controllers that ensure the safety and repeatability of the process. However, similarly to many other physical systems, AM systems are difficult to accurately model. Even when an accurate model of the process is available, it is usually too complex for traditional closed-loop control design. This obstacle can be overcome with the adoption of data-driven control techniques and specifically Reinforcement Learning (RL). In our work, we designed and implemented RL-based closed-loop controllers and compared them with their model-based counterparts, whenever they were available. In this final Chapter we revisit these methods introduced in Chapters 3-5, discuss their performance and their limitations, as motivation for future research.

In Chapter 3, we formulate the control problem as a Linear Quadratic Tracking (LQT) problem and solve it by first assuming knowledge of a model describing the system's dynamics. We explored both the finite horizon and the infinite horizon case. Thanks to the form of the performance index, an analytical solution to the problem can be found. The form of the analytical solution was then used as the basis for designing an RL-based data-driven controller that estimates the system model in order to obtain the optimal policy. The estimation step involves the implementation of the Least Squares algorithm. Through simulations, we showcased the performance of all controllers. We found that the data-driven controller was on a par with its model-based counterpart. This is a very important step towards autonomous intelligent manufacturing.

While our approach is very powerful and provides encouraging results, it comes with certain drawbacks that result from the limiting assumptions that were made. Firstly, the choice of the LQT framework limits the application of this approach to linear systems optimised under quadratic costs. This is due to the fact that the estimation formula for the optimal policy is based on the analytical solution of the LQT. Linear models are very convenient for control design but are a very rare occurrence in physical systems.

Similarly, quadratic objective functions are useful due to their convexity but they are often not appropriate to express all optimisation objectives. The development of data-driven controllers, that can learn optimal policies in more complicated environments, is an important extension to our work. Finally, we assumed that the system state was fully available for observation, an assumption that is not realistic in many physical systems, as the state is usually measured through sensors. This limitation is addressed in the next chapter.

In Chapter 4, we explore the case of output feedback, namely the case where the system state is not fully observable. We use the LQT framework again and we incorporate a state estimation step in the control design. In the case where a process model is available, we design a state observer. In the model-free case, the state is estimated through a record of past inputs and outputs, which significantly increases the dimension of the problem. We highlighted the importance of generating appropriate data for training and provided a detailed analysis of the best techniques we used for generating data in our simulations. We were able to train a controller that performs comparably to the model-based case, while learning strictly from the input-output data. In addition to the data-driven control design, we explored the use of Bayesian Optimisation to improve the design of the optimisation objective. Specifically, we optimised the design of the weighting matrices in the quadratic performance index. Our experiments verified the value of this approach, as it led to higher tracking performance as well as lower overall costs.

The main advantage of the introduced method is its ability to learn optimal policies without prior knowledge about the system and without direct access to the internal system state. Additionally, the description of "good" data generation can be a very important guide for many intelligent manufacturing applications. A lot of AM processes are very costly and time-consuming, so it is important to design the experiments correctly so as not to waste resources. However, similarly to the method introduced in Chapter 3, this approach is also based on the assumption of a linear model and a quadratic performance index. The search for more versatile algorithms that can handle complicated environments is the most important extension to this work. The other important drawback of this approach stems from the use of the Least Squares algorithm. It involves a matrix inversion step that becomes computationally expensive and slow as the problem's dimension increases. This drawback was evident in the work of Chapter 3 as well, but the state augmentation through past inputs and outputs in Chapter 4 made this issue more prevalent. If this approach were to be applied to systems with much higher state and action spaces, the inversion step would become almost intractable, which motivates the need for algorithms that circumvent that step. All the above concerns can be addressed by the use of Deep Neural Networks. They are powerful function approximators, that can handle non-linear dynamics and non-quadratic value functions.

In Chapter 5 we explored the use of Deep Reinforcement Learning, and specifically the Deep Deterministic Policy Gradient (DDPG) algorithm for solving the temperature control problem. We trained two Deep Neural Networks (DNNs), the Actor and the Critic, learning the optimal policy and the value function respectively. They interact with each other, while exploring the environment, learning from past experience how to act optimally. To showcase the flexibility of this approach, we consider a value function that is no longer quadratic. We introduce constraints on the system's response, which is imperative in real-life applications, where if certain bounds are exceeded it can cause damage to the machine, or increase the cost beyond an acceptable amount. With the help of the Critic, the Actor learns a policy that successfully tracks the reference signal within a small error margin. This approach does not need any prior assumptions about the system or the control design.

Even though we used the same linear model for our simulations as the previous chapters, the DDPG algorithm can handle complicated non-linear systems, with appropriate tuning of its parameters. This is a big advantage of this method over the previous ones. However, the existence of many different parameters to be tuned, makes it difficult to find the best values as there is no systematic way to chose them. Applying this approach to more complicated problems is a research venue worth exploring. Apart from using non-linear models and non-quadratic value functions, the complexity can be introduced in terms of the data that is used. In our applications the data is numerical, as it is referring to temperatures. Depending on the problem, the data can be of different formats such as visual, acoustic etc. Different data formats would need the introduction of more complicated layers in the DNNs, for example convolutional layers.

All the methods we have introduced throughout the thesis have proven to be very powerful and successful in designing effective data-driven controllers. They were all tested through simulations, which was a useful tool when wanting to compare them to a baseline, model-based situation. The next step would be to test all these algorithms in a real-life scenario, to control the temperatures in a real AM system. Training these algorithms requires a large amount of data and long computations which means the machine in question would need to operate for a long period of time and repeat experiments multiple times. This can be a costly process that will also produce a lot of waste. Furthermore, the agent needs to thoroughly explore the environment during training, by trying a variety of different actions. This has the potential to damage the machine, if the actions explored are close to or beyond certain safe bounds. These issues need to be considered and addressed in future research. One course of action could be to pre-train a RL controller using simulations, based on the most accurate model available, and then bring it online to continue training from process data. That way, the advantages of both model-based and data-driven methods can be exploited.

Finally, we note that the code used for the implementations of the methods in Chapters 3-5 is freely available on `https://github.com/elenizavrakli`. This ensures all results are reproducible and the methods are accessible to future practitioners. We remark that our code has not been optimised in terms of computational speed, with a few exceptions such as the matrix inversion step in the Least Squares algorithm in 4. Further improvements would allow the wider adoption of these methods, especially for real-time control applications.

Bibliography

[1] A Savini and GG Savini. A short history of 3d printing, a technological revolution just started. In *2015 ICOHTEC/IEEE international history of high-technologies and their socio-cultural contexts conference (HISTELCON)*, pages 1–8. IEEE, 2015.

[2] Barry Berman. 3-d printing: The new industrial revolution. *Business Horizons*, 55(2):155–162, 2012.

[3] C. Buchanan and L. Gardner. Metal 3d printing in construction: A review of methods, research, applications, opportunities and challenges. *Engineering Structures*, 180:332–348, 2019.

[4] Ugur M. Dilberoglu, Bahar Gharehpapagh, Ulas Yaman, and Melik Dolen. The role of additive manufacturing in the era of industry 4.0. *Procedia Manufacturing*, 11:545–554, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[5] Francisco Jose Mercado Rivera and Alvaro Jose Rojas Arciniegas. Additive manufacturing methods: techniques, materials, and closed-loop control applications. *The International Journal of Advanced Manufacturing Technology*, 109:17–31, 2020.

[6] Katayoon Taherkhani, Osazee Ero, Farima Liravi, Sahar Toorandaz, and Ehsan Toyserkani. On the application of in-situ monitoring systems and machine learning algorithms for developing quality assurance platforms in laser powder bed fusion: A review. *Journal of Manufacturing Processes*, 99:848–897, 2023.

[7] Yanzhou Fu, Austin R.J. Downey, Lang Yuan, Tianyu Zhang, Avery Pratt, and Yunusa Balogun. Machine learning algorithms for defect detection in metal laser-based additive manufacturing: A review. *Journal of Manufacturing Processes*, 75:693–710, 2022.

[8] Hugo Lhachemi, Ammar Malik, and Robert Shorten. Augmented reality, cyber-physical systems, and feedback control for additive manufacturing: A review. *IEEE Access*, 7:50119–50135, 2019.

[9] Gustavo Tapia and Alaa Elwany. A review on process monitoring and control in metal-based additive manufacturing. *Journal of Manufacturing Science and Engineering*, 136(6):060801, 2014.

[10] Brian DO Anderson and John B Moore. *Optimal control: linear quadratic methods.* Courier Corporation, 2007.

[11] Katsuhiko Ogata et al. *Modern control engineering*, volume 5. Prentice hall Upper Saddle River, NJ, 2010.

[12] Maude Josée Blondin, Panos M Pardalos, and Javier Sanchis Sáez. Computational intelligence and optimization methods for control engineering. 2019.

[13] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[14] Rudolf Emil Kalman et al. Contributions to the theory of optimal control. *Bol. soc. mat. mexicana*, 5(2):102–119, 1960.

[15] LS PONTRJAGIN. The mathematical theory of optimal processes. *Interscience*, 1962.

[16] Richard Bellman. *Dynamic programming.* Princeton University Press, 1956.

[17] Dieter Fingerle. Autogenic melt temperature control system for plastic extrusion. *Journal of Elastomers & Plastics*, 10(4):293–310, 1978.

[18] MH Costin, PA Taylor, and JD Wright. On the dynamics and control of a plasticating extruder. *Polymer Engineering & Science*, 22(17):1095–1106, 1982.

[19] Fabio Previdi, Sergio M Savaresi, and Angiolino Panarotto. Design of a feedback control system for real-time control of flow in a single-screw extruder. *Control engineering practice*, 14(9):1111–1121, 2006.

[20] Bao Kha Nguyen, Gerard McNally, and Alan Clarke. Real time measurement and control of viscosity for extrusion processes using recycled materials. *Polymer degradation and stability*, 102:212–221, 2014.

[21] Mohammad H. Farshidianfar, Amir Khajepour, and Adrian Gerlich. Real-time control of microstructure in laser additive manufacturing. *The International Journal of Advanced Manufacturing Technology*, 82(5):1173–1186, 2016.

[22] Dongming Hu and Radovan Kovacevic. Sensing, modeling and control for laser-based additive manufacturing. *International Journal of Machine Tools and Manufacture*, 43(1):51–60, 2003.

[23] Yaoyu Ding, James Warton, and Radovan Kovacevic. Development of sensing and control system for robotized laser-based direct metal addition system. *Additive Manufacturing*, 10:24–35, 2016.

[24] Patrick M. Sammons, Michelle L. Gegel, Douglas A. Bristow, and Robert G. Landers. Repetitive process control of additive manufacturing with application to laser metal deposition. *IEEE Transactions on Control Systems Technology*, 27(2):566–575, 2019.

[25] Charalabos Doumanidis and Yong-Min Kwak. Geometry modeling and control by infrared and laser sensing in thermal manufacturing with material deposition. *J. Manuf. Sci. Eng.*, 123(1):45–52, 2001.

[26] J-P Kruth, Peter Mercelis, Jonas Van Vaerenbergh, and Tom Craeghs. Feedback control of selective laser melting. In *Virtual and Rapid Manufacturing*, pages 521–528. Crc Press, 2007.

[27] Lie Tang and Robert G Landers. Melt pool temperature control for laser metal deposition processes—part i: Online temperature control. 2010.

[28] Qian Wang, Jianyi Li, Michael Gouge, Abdalla R Nassar, Panagiotis Michaleris, and Edward W Reutzel. Physics-based multivariable modeling and feedback linearization control of melt-pool geometry and temperature in directed energy deposition. *Journal of Manufacturing Science and Engineering*, 139(2):021013, 2017.

[29] Harry Bikas, Panagiotis Stavropoulos, and George Chryssolouris. Additive manufacturing methods and modelling approaches: a critical review. *The International Journal of Advanced Manufacturing Technology*, 83:389–405, 2016.

[30] Jing Jiang, Shengping Wen, and Guoping Zhao. A melt temperature pid controller based on rbf neural network. In *2008 ISECS International Colloquium on Computing, Communication, Control, and Management*, volume 2, pages 172–175. IEEE, 2008.

[31] Xiyue Zhao, Robert G Landers, and Ming C Leu. Adaptive extrusion force control of freeze-form extrusion fabrication processes. 2010.

[32] Lawrence W Funke and James P Schmiedeler. Control of final part dimensions in polymer extrusion using a variable-geometry die. *Journal of Manufacturing Science and Engineering*, 140(8):081001, 2018.

[33] Patrick M Sammons, Douglas A Bristow, and Robert G Landers. Height dependent laser metal deposition process modeling. *Journal of Manufacturing Science and engineering*, 135(5):054501, 2013.

[34] Zhong-Sheng Hou and Zhuo Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, 235:3–35, 2013.

[35] Claudio De Persis and Pietro Tesi. Formulas for data-driven control: Stabilization, optimality, and robustness. *IEEE Transactions on Automatic Control*, 65(3):909–924, 2019.

[36] Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, 2017.

[37] Dario Piga, Simone Formentin, and Alberto Bemporad. Direct data-driven control of constrained systems. *IEEE Transactions on Control Systems Technology*, 26(4):1422–1429, 2017.

[38] Tom M Mitchell. *Machine learning*. McGraw Hill, 1997.

[39] Zhi-Hua Zhou. *Machine learning*. Springer Nature, 2021.

[40] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[41] Jennie Si, Andrew G Barto, Warren B Powell, and Don Wunsch. *Handbook of learning and approximate dynamic programming*, volume 2. John Wiley & Sons, 2004.

[42] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[43] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

[44] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

[45] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[46] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[47] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[48] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[49] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[50] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[51] Audelia G Dharmawan, Yi Xiong, Shaohui Foong, and Gim Song Soh. A model-based reinforcement learning and correction framework for process control of robotic wire arc additive manufacturing. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4030–4036. IEEE, 2020.

[52] Jihoon Chung, Bo Shen, Andrew Chung Chee Law, and Zhenyu James Kong. Reinforcement learning-based defect mitigation for quality assurance of additive manufacturing. *Journal of Manufacturing Systems*, 65:822–835, 2022.

[53] Mirko Alicastro, Daniele Ferone, Paola Festa, Serena Fugaro, and Tommaso Pastore. A reinforcement learning iterated local search for makespan minimization in additive manufacturing machine scheduling problems. *Computers & Operations Research*, 131:105272, 2021.

[54] Susheel Dharmadhikari, Nandana Menon, and Amrita Basak. A reinforcement learning approach for process parameter optimization in additive manufacturing. *Additive Manufacturing*, 71:103556, 2023.

[55] Steven Patrick, Andrzej Nycz, and Mark Noakes. Reinforcement learning for generating toolpaths in additive manufacturing. In *2018 International Solid Freeform Fabrication Symposium*. University of Texas at Austin, 2018.

[56] Jan Petrik and Markus Bambach. Reinforcement learning and optimization based path planning for thin-walled structures in wire arc additive manufacturing. *Journal of Manufacturing Processes*, 93:75–89, 2023.

[57] Bing Yao, Farhad Imani, and Hui Yang. Markov decision process for image-guided additive manufacturing. *IEEE Robotics and Automation Letters*, 3(4):2792–2798, 2018.

[58] Kilian Wasmer, Tri Le-Quang, Bastian Meylan, and Sergey A Shevchik. In situ quality monitoring in am using acoustic emission: A reinforcement learning approach. *Journal of Materials Engineering and Performance*, 28:666–672, 2019.

[59] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[60] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[61] Francis Ogoke and Amir Barati Farimani. Thermal control of laser powder bed fusion using deep reinforcement learning. *Additive Manufacturing*, 46:102033, 2021.

[62] Mojtaba Mozaffar, Ablodghani Ebrahimi, and Jian Cao. Toolpath design for additive manufacturing using deep reinforcement learning. *arXiv preprint arXiv:2009.14365*, 2020.

[63] Benjamin Felbrich, Tim Schork, and Achim Menges. Autonomous robotic additive manufacturing through distributed model-free deep reinforcement learning in computational design environments. *Construction Robotics*, 6(1):15–37, 2022.

[64] Fabio Parisi, Valentino Sangiorgio, Nicola Parisi, Agostino M Mangini, Maria Pia Fanti, and Jose M Adam. A new concept for large additive manufacturing in construction: tower crane-based 3d printing controlled by deep reinforcement learning. *Construction Innovation*, 2023.

[65] MC Brennan, JS Keist, and TA Palmer. Defects in metal additive manufacturing processes, 2021.

[66] Arit Das, Camden A Chatham, Jacob J Fallon, Callie E Zawaski, Eric L Gilmer, Christopher B Williams, and Michael J Bortner. Current understanding and challenges in high temperature additive manufacturing of engineering thermoplastic polymers. *Additive Manufacturing*, 34:101218, 2020.

[67] Sun Xiaoyong, Cao Liangcheng, Ma Honglin, Gao Peng, Bai Zhanwei, and Li Cheng. Experimental analysis of high temperature peek materials on 3d printing test. In *2017 9th International conference on measuring technology and mechatronics automation (ICMTMA)*, pages 13–16. IEEE, 2017.

[68] Austin Lee, Mathew Wynn, Liam Quigley, Marco Salviato, and Navid Zobeiry. Effect of temperature history during additive manufacturing on crystalline morphology of peek. *Advances in Industrial and Manufacturing Engineering*, 4:100085, 2022.

[69] Didier Gootjes. Applying feedback control to improve 3d printing quality. Master's thesis, Delf University of Technology, 2017.

[70] Stoyan Stoyanov and Chris Bailey. Machine learning for additive manufacturing of electronics. In *2017 40th international spring seminar on electronics technology (ISSE)*, pages 1–6. IEEE, 2017.

[71] Nathaniel Wood and David J. Hoelzle. On the feasibility of a temperature state observer for powder bed fusion additive manufacturing. In *2018 Annual American Control Conference (ACC)*, pages 321–328, 2018.

[72] Frédérick Garcia and Emmanuel Rachelson. Markov decision processes. *Markov Decision Processes in Artificial Intelligence*, pages 1–38, 2013.

[73] Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957.

[74] François Dufour and Tomás Prieto-Rumeau. Approximation of markov decision processes with general state space. *Journal of Mathematical Analysis and applications*, 388(2):1254–1267, 2012.

[75] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 2. Athena scientific Belmont, MA, 1995.

[76] John N Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202, 1994.

[77] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[78] Abhijit Gosavi. A tutorial for reinforcement learning. *The State University of New York at Buffalo*, 2017.

[79] Bahare Kiumarsi, Frank L Lewis, Hamidreza Modares, Ali Karimpour, and Mohammad-Bagher Naghibi-Sistani. Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics. *Automatica*, 50(4):1167–1175, 2014.

[80] Frank L. Lewis, Draguna Vrabie, and Kyriakos G. Vamvoudakis. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine*, 32(6):76–105, 2012.

[81] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

[82] Gary Hewer. An iterative technique for the computation of the steady state gains for the discrete optimal regulator. *IEEE Transactions on Automatic Control*, 16(4):382–384, 1971.

[83] Lennart Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer, 1998.

[84] J Yuan and W Wonham. Probing signals for model reference identification. *IEEE Transactions on Automatic Control*, 22(4):530–538, 1977.

[85] William E Frazier. Metal additive manufacturing: a review. *Journal of Materials Engineering and performance*, 23:1917–1928, 2014.

[86] Adrián Uriondo, Manuel Esperon-Miguez, and Suresh Perinpanayagam. The present and future of additive manufacturing in the aerospace sector: A review of important aspects. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 229(11):2132–2147, 2015.

[87] Dong Dong Gu, Wilhelm Meiners, Konrad Wissenbach, and Reinhart Poprawe. Laser additive manufacturing of metallic components: materials, processes and mechanisms. *International materials reviews*, 57(3):133–164, 2012.

[88] Eleni Zavrakli, Andrew Parnell, Andrew Dickson, and Subhrakanti Dey. Data-driven Linear Quadratic Tracking based Temperature Control of a Big Area Additive Manufacturing System. *arXiv preprint arXiv:2307.07039*, 2023.

[89] David Luenberger. An introduction to observers. *IEEE Transactions on automatic control*, 16(6):596–602, 1971.

[90] Cong Chen, Weijie Sun, Guangyue Zhao, and Yunjian Peng. Reinforcement Q-Learning Incorporated With Internal Model Method for Output Feedback Tracking Control of Unknown Linear Systems. *IEEE Access*, 8:134456–134467, 2020.

[91] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[92] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[93] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty: An experimental comparison on a dynamic bipedal walker. *Annals of Mathematics and Artificial Intelligence*, 76:5–23, 2016.

[94] Alonso Marco, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe. Automatic LQR tuning based on Gaussian process global optimization.

In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 270–277. IEEE, 2016.

[95] Kou Miyamoto, Jinhua She, Daiki Sato, and Nobuaki Yasuo. Automatic determination of LQR weighting matrices for active structural control. *Engineering Structures*, 174:308–321, 2018.

[96] Eliakim H Moore. On the reciprocal of the general algebraic matrix. *Bull. Am. Math. Soc.*, 26:394–395, 1920.

[97] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955.

[98] Victor G Lopez, Mohammad Alsalti, and Matthias A Müller. Efficient off-policy Q-learning for data-based discrete-time lqr problems. *IEEE Transactions on Automatic Control*, 2023.

[99] Aravindh Krishnamoorthy and Deepak Menon. Matrix inversion using Cholesky decomposition. In *2013 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pages 70–72, 2013.

[100] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[101] Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian processes for machine learning*, volume 1. Springer, 2006.

[102] Stanley M Shinners. *Modern control system theory and design.* John Wiley & Sons, 1998.

[103] Byron Blakey-Milner, Paul Gradl, Glen Snedden, Michael Brooks, Jean Pitot, Elena Lopez, Martin Leary, Filippo Berto, and Anton Du Plessis. Metal additive manufacturing in aerospace: A review. *Materials & Design*, 209:110008, 2021.

[104] Joel C Najmon, Sajjad Raeisi, and Andres Tovar. Review of additive manufacturing technologies and applications in the aerospace industry. *Additive manufacturing for the aerospace industry*, pages 7–31, 2019.

[105] Sunpreet Singh and Seeram Ramakrishna. Biomedical applications of additive manufacturing: present and future. *Current opinion in biomedical engineering*, 2:105–115, 2017.

[106] M Vignesh, G Ranjith Kumar, M Sathishkumar, M Manikandan, G Rajyalakshmi, R Ramanujam, and N Arivazhagan. Development of biomedical implants through additive manufacturing: A review. *Journal of Materials Engineering and Performance*, 30:4735–4744, 2021.

[107] Edward W Reutzel and Abdalla R Nassar. A survey of sensing and control systems for machine and process monitoring of directed-energy, metal-based additive manufacturing. *Rapid Prototyping Journal*, 21(2):159–167, 2015.

[108] Eleni Zavrakli, Andrew Parnell, and Subhrakanti Dey. Output feedback reinforcement learning for temperature control in a fused deposition modelling additive manufacturing system. In *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 1483–1487, 2023.

[109] Eleni Zavrakli, Andrew Parnell, and Subhrakanti Dey. Output feedback reinforcement learning with parameter optimisation for temperature control in a material extrusion additive manufacturing system. *arXiv preprint arXiv:2310.03599*, 2023.

[110] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[111] George E Uhlenbeck and Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

[112] Jakob Hollenstein, Sayantan Auddy, Matteo Saveriano, Erwan Renaudo, and Justus Piater. Action noise in off-policy deep reinforcement learning: Impact on exploration and performance. *arXiv preprint arXiv:2206.03787*, 2022.

A
Optimal age-specific vaccination control for COVID-19: an Irish case study

Abstract

The outbreak of a novel coronavirus causing severe acute respiratory syndrome in December 2019 has escalated into a worldwide pandemic. In this work, we propose a compartmental model to describe the dynamics of transmission of infection and use it to obtain the optimal vaccination control. The model accounts for the various stages of the vaccination, and the optimisation is focused on minimising the infections to protect the population and relieve the healthcare system. As a case study, we selected the Republic of Ireland. We use data provided by Ireland's COVID-19 Data-Hub and simulate the evolution of the pandemic with and without the vaccination in place for two different scenarios, one representative of a national lockdown situation and the other indicating looser restrictions in place. One of the main findings of our work is that the optimal approach would involve a vaccination programme where the older population is vaccinated in larger numbers earlier while simultaneously part of the younger population also gets vaccinated to lower the risk of transmission between groups. We compare our simulated results with those of the vaccination policy taken by the Irish government to explore the advantages of our optimisation method. Our comparison suggests that a similar reduction in cases may have been possible even with a reduced set of vaccinations available for use.

Code availability

The code used for our simulations was written in R and can be found at: `https://github.com/elenizavrakli/Optimal-Age-Specific-Vaccination-Control`

A.1 Introduction

In late 2019, an outbreak of pneumonia of unknown cause was reported in the city of Wuhan in the Hubei province of China [1–7]. The virus was named Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2) by the World Health Organisation [8] and the disease that it causes is referred to as COVID-19 [9]. The disease quickly became a source of international worry as it spread around China with most countries around the world following [10]. By March 2020, most countries in the world had confirmed cases of COVID-19, including the Republic of Ireland, where the first confirmed case was on February 29th 2020 [11], the same day that WHO raised the risk warning for the virus to "very high" [12]. In the absence of an effective treatment or a vaccine, governments worldwide started implementing protective measures with most of them announcing a national lockdown to try and control the spread of the virus and reduce the strain on their healthcare systems. In Ireland, the first restrictive measures and social distancing guidelines were first announced on March 12th 2020 and were initially intended to last until March 29th. While countries tried to control the virus and protect their citizens, the scientific community committed to developing an effective vaccine to put an end to the pandemic. At the same time, researchers dedicated themselves to study, model and predict the evolution of the pandemic as well as investigate non-pharmaceutical intervention methods to control the spread of the virus [13–25].

On November 9th 2020 Pfizer and BioNTech [26, 27] announced a vaccine candidate that successfully completed the clinical trials and was 90% effective in preventing infection from the virus. Shortly after, two more vaccines were announced, namely the one by Moderna [28] and the one by AstraZeneca [29]. All three vaccines got approved by the European Union [30–32], and a fourth vaccine was granted commercial authorisation in March 2021, namely the Jcovden one (previously known as Janssen) [33]. As of 2023, there are seven different vaccines approved for use by European citizens, with more in development [34]. Worldwide, certain states have made different decisions regarding available vaccines. Since the start of 2021, a vaccination roll-out commenced in most countries, including Ireland, taking into account the number of available vaccines and the level of risk that different groups of people are considered to be in. As of 2022, first-world countries have for the most part completed the first round of vaccinations and are in the process of administering booster shots. On the other hand, as of the appendix composition in 2023, many countries are still in very early stages of vaccination [35]. Given this situation, a need that naturally arises is that of a way to determine the optimal vaccination strategy, especially in the cases where the resources to reduce the severity of the pandemic are still limited [36, 37]. This idea was the main motivation for our study. In addition to improving the current state of the pandemic in certain countries, our work can be adapted for use in the unfortunate scenario of future pandemics or COVID-19 variants that require the distribution of new vaccines from scratch.

Modelling, predicting, and controlling the behaviour of epidemics has been a widely studied area [38, 39]. A very prominent example is the pandemic influenza, a virus that caused an outbreak of severe pneumonia in 2009, commonly known as ”swine-flu” [40]. Several models were developed to evaluate the implementation of mitigation strategies [41–44], with a great focus on optimising these strategies. Since the outbreak of COVID-19 and the wider availability of the vaccines, a number of studies have been conducted on modelling and simulating the evolution of the pandemic under different scenarios, in order to draw conclusions on the best mitigation strategies, often considering different strategies for different age groups. In [45] the main questions that were investigated relate to the minimum vaccination coverage for each older age group before starting vaccination for a subsequent group, the optimal interval between administering vaccine doses and the corresponding minimum number of vaccine doses that can make the implementation of this interval possible. [46] considered two age groups and two different mathematical models to describe the dynamics of transmission. Using these models, the authors drew conclusions about vaccination strategies based on different assumptions on the efficacy of the vaccines and the transmission rate of the infection. In [47], UK based simulations were conducted to assess factors that influence the success of any vaccination program, considering 3 types of vaccines, each targeting a different aspect of infection. Optimal control [48–51] is an important tool that allows for optimising these mitigation strategies in a systematic way, according to a minimisation objective. In [52], 3 different objectives where considered, using South Korea as a case study and considering 16 age groups. In [53] two objectives are explored, namely the minimisation of infections and the minimisation of deaths, and the problem is solved through the interior point method, for China, Brazil, and Indonesia. In [54], the authors consider two age groups and use both vaccination and medical treatment as control variables and find that combining both works best in reducing infection and deaths. In [55], China is used as a case study and 3 control measures are considered vaccination, isolation and nucleic acid testing. Forecast and impact of vaccination during the third wave in Pakistan is studied in [56], along with estimation of some epidemiological parameters. Finally, in [57] optimal control is applied when 3 different intervention methods are considered, namely isolation, vaccination and treatment.

In this work, we introduce a new model that describes the transmission dynamics of the virus among different groups of the population in a more complete way compared to the well known SEIR model [13,58–60] which is commonly used in the study of epidemics. In the context of COVID-19, SEIR-based models have been widely used for forecasting purposes in works such as [61–63]. Specifically, in [61] the focus is on simulating infections using data from Singapore, while in [62] data from many different countries are being used. In [63], the authors use an SEIR model to estimate the distribution of time delays in reported data, in Italy and Austria. The novelty of our approach lies in the introduction of new compartments to the model accounting for stages of the vaccination process as well as two different age groups. Using this model as our baseline, we explore a way to determine the optimal vaccination strategy, applying through optimal control theory methods. The control action applied to the system is considered to be the vaccination percentage for each age group. Using these tools, we studied the evolution of the pandemic in the Republic of Ireland, starting from January 2021 when the vaccination roll-out began, using data provided by *Ireland's COVID-19 Data-Hub* [64] and parameter values according to the work done in [65]. We obtained the optimal vaccination strategy based on estimates of the initial conditions, which is similar to the course of action taken by the state. However, we find that it is beneficial to start vaccinating people under the age of 65 in parallel with people in older age groups but in smaller numbers, as opposed to exclusively vaccinating the older population first. We compare our obtained strategy with the course of action taken by the state and comment on the benefits and disadvantages of both approaches.

In Section A.2 we introduce our model and express it as a system of ordinary differential equations. In Section A.3 we use optimal control theory techniques to obtain the optimal vaccination strategy, given our model. In Section A.4 we produce two sets of simulations to compare the evolution of the pandemic with and without the vaccination strategy in place, under strict and loose restrictions. In Section A.5 we present a comparative study of the strategy obtained through our optimisation against the strategy chosen by the government. Finally, in Section A.6 we discuss our findings, the possible drawbacks of our method, and extensions to our approach that are worth exploring.

A.2 Model

The model most commonly used in the study of epidemics was first introduced by *M'Kendrick* in [59, 60] and is known as the SIR model. This model suggests that every member of a population can be considered to belong in one of the three compartments: Susceptible (S), Infectious (I) or Recovered (R). In the study of COVID-19, an extra compartment is usually added to the SIR model, namely the Exposed (E) compartment, consisting of the people who have been in contact with the virus but have not developed any symptoms. This is known as the SEIR model [13]. All compartments can be indexed by time (t), since they express the number of individuals in each compartment for each moment in time. In a closed population with no births and no deaths, the model can be expressed as a system of ordinary differential equations as follows:

$$\begin{aligned}
\frac{dS_t}{dt} &= -\frac{\beta I_t}{N} S_t \\
\frac{dE_t}{dt} &= \frac{\beta I_t}{N} S_t - \sigma E_t \\
\frac{dI_t}{dt} &= \sigma E_t - \gamma I_t \\
\frac{dR_t}{dt} &= \gamma I_t
\end{aligned} \tag{A.1}$$

where $N = S_t + E_t + I_t + R_t$ is the total population which we assume constant, namely we are dealing with a closed population. β is the infectious rate, which expresses the probability that a susceptible person gets infected by an infectious person, σ is the incubation rate, meaning the rate at which an exposed individual becomes infectious and γ is the recovery rate calculated as the inverse of the average time of infection.

In this study, we use a model that takes the idea of the SEIR model one step further, with additional compartments related to whether an individual has been vaccinated and whether the vaccination was effective, hence protecting the individual. The new set of states consists of the following: Susceptible not yet vaccinated (S); Received the vaccine; waiting for it to take effect (V); Received the vaccine but it was not effective (N); Susceptible, refusing or unable to receive the vaccine (U); Exposed, infectious but still asymptomatic (E); Infectious symptomatic (I); Recovered or deceased (R); Protected from vaccine (P).

A full table of the notations used throughout the appendix can be found in A.7. In addition, we assume that a susceptible individual may get infected by exposed as well as infectious individuals. Furthermore, we consider two different age groups, those over 65 years old and those younger than 65. The reasons for this distinction can be found in [66], the main ones being that 80% of the hospitalised individuals are over the age of 65 and this age group has a 23 times higher risk of death compared to those under 65. This results in two identical models, one for the younger population and one for the older which allows for the introduction of different parameters to account for the way the virus affects people of different ages. Figure A.1 is a depiction of our model for an individual age cohort (for brevity we do not show the full model configuration). The state configuration is exactly the same in both cases with the difference being in the transition rates from one state to the other.

We distinguish the states of the two models by adding an O or a Y as an subscript to

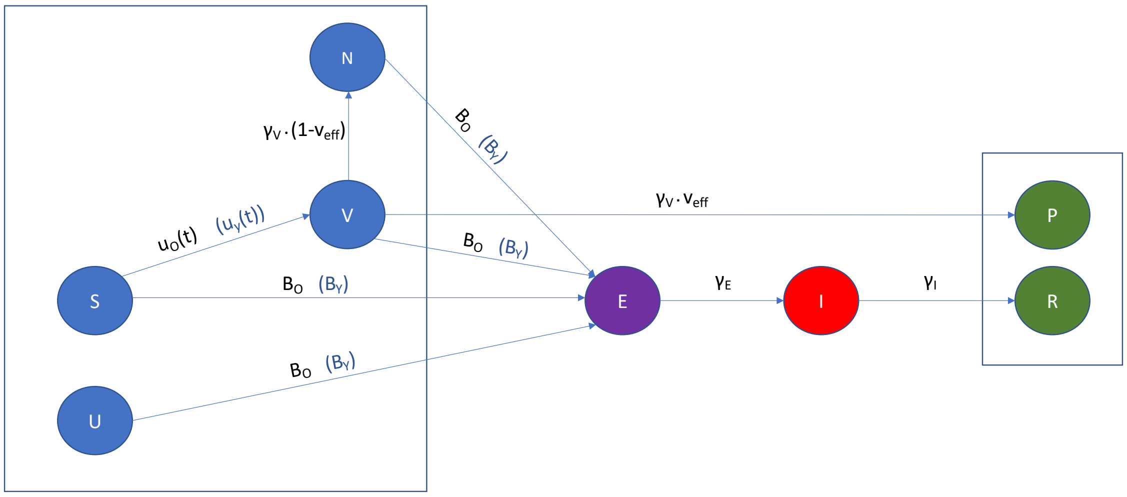


Figure A.1: Vaccination Compartmental model
The model compartments (states) are indicated by the coloured circles. Blue indicates states at which one might get infected by an exposed or infectious person and green indicates the states at which a person is considered to be safe from infection. The parameters describing the transmission rates are indicated on the arcs between the states. The models for the over and under 65 populations (indicated by O and Y subscripts) are identical with the only differences being the age specific control functions $u_O(t), u_Y(t)$ which express the percentage of the susceptible population to be vaccinated at each time point, and the rates of infection from an exposed or infectious individual $B_O = \frac{\beta_{OO}(E_O(t)+I_O(t))}{T_O} + \frac{\beta_{YO}(E_Y(t)+I_Y(t))}{T_Y}$ and $B_Y = \frac{\beta_{OY}(E_O(t)+I_O(t))}{T_O} + \frac{\beta_{YY}(E_Y(t)+I_Y(t))}{T_Y}$.

the state name to indicate the older or younger than 65 age group respectively e.g. S_O, V_Y etc. Since the states are time dependent, we can write them in function notation e.g. $S_O(t)$, $V_Y(t)$, however, for brevity, we will omit the (t) in the equations that follow. We also use the notation T_O, T_Y to indicate the total number of people in each age group, which is assumed to not be time dependent. The two populations influence each other in the sense that a susceptible person in any of the two groups may be infected by an exposed or infectious person from either group. For brevity, we will indicate the older than 65 age group as o65 and the younger than 65 age group as y65. We will be using upper-case Roman characters to indicate the model states and lower-case Greek characters to indicate model parameters.

The model is expressed as a system of ordinary differential equations (ODEs), each of them describing the evolution through time of one of the states. The ODEs describing the dynamics of our model are given by:

$$
\begin{aligned}
\frac{dS_O}{dt} &= -\left(\frac{\beta_{OO}(E_O + I_O)}{T_O} + \frac{\beta_{YO}(E_Y + I_Y)}{T_Y}\right) S_O - u_O S_O \\
\frac{dS_Y}{dt} &= -\left(\frac{\beta_{OY}(E_O + I_O)}{T_O} + \frac{\beta_{YY}(E_Y + I_Y)}{T_Y}\right) S_Y - u_Y S_Y
\end{aligned}
$$

$$\begin{aligned}
\frac{dV_O}{dt} &= u_O S_O - \left(\frac{\beta_{OO}(E_O+I_O)}{T_O} + \frac{\beta_{YO}(E_Y+I_Y)}{T_Y}\right) V_O \\
&\quad -(1-\alpha_V)\gamma_V V_O - \gamma_V \alpha_V V_O \\
\frac{dV_Y}{dt} &= u_Y S_Y - \left(\frac{\beta_{OY}(E_O+I_O)}{T_O} + \frac{\beta_{YY}(E_Y+I_Y)}{T_Y}\right) V_Y \\
&\quad -(1-\alpha_V)\gamma_V V_Y - \gamma_V \alpha_V V_Y \\
\frac{dN_O}{dt} &= (1-\alpha_V)\gamma_V V_O - \left(\frac{\beta_{OO}(E_O+I_O)}{T_O} + \frac{\beta_{YO}(E_Y+I_Y)}{T_Y}\right) N_O \\
\frac{dN_Y}{dt} &= (1-\alpha_V)\gamma_V V_Y - \left(\frac{\beta_{OY}(E_O+I_O)}{T_O} + \frac{\beta_{YY}(E_Y+I_Y)}{T_Y}\right) N_Y \\
\frac{dU_O}{dt} &= -\left(\frac{\beta_{OO}(E_O+I_O)}{T_O} + \frac{\beta_{YO}(E_Y+I_Y)}{T_Y}\right) U_O \\
\frac{dU_Y}{dt} &= -\left(\frac{\beta_{OY}\cdot(E_O+I_O)}{T_O} + \frac{\beta_{YY}\cdot(E_Y+I_Y)}{T_Y}\right) U_Y \\
\frac{dE_O}{dt} &= \left(\frac{\beta_{OO}(E_O+I_O)}{T_O} + \frac{\beta_{YO}(E_Y+I_Y)}{T_Y}\right) (S_O+V_O+N_O+U_O) \\
&\quad -\gamma_E E_O \\
\frac{dE_Y}{dt} &= \left(\frac{\beta_{OY}(E_O+I_O)}{T_O} + \frac{\beta_{YY}(E_Y+I_Y)}{T_Y}\right) (S_Y+V_Y+N_Y+U_Y) \\
&\quad -\gamma_E E_Y \\
\frac{dI_O}{dt} &= \gamma_E E_O - \gamma_I I_O \\
\frac{dI_Y}{dt} &= \gamma_E E_Y - \gamma_I I_Y \\
\frac{dR_O}{dt} &= \gamma_I I_O \\
\frac{dR_Y}{dt} &= \gamma_I I_Y \\
\frac{dP_O}{dt} &= \gamma_V \alpha_V \cdot V_O \\
\frac{dP_Y}{dt} &= \gamma_V \alpha_V \cdot V_Y
\end{aligned} \tag{A.2}$$

where T_O, T_Y denote the total number of people in the over and under 65 years old populations respectively. These populations are divided in those willing and those unwilling to get vaccinated (S_O, S_Y, U_O, U_Y). This division happens based on estimates of the percentages of people who refuse the vaccine in each age group. The only option for individuals in the U compartments, apart from staying there if the pandemic ends early, is to eventually get infected and be moved to the exposed compartments E.

The terms u_O, u_Y are the control functions and represent the percentage of the old and young population respectively to be vaccinated at each time point, which means that they are also time dependent. Each control function is applied to the respective susceptible population (S_O, S_Y), moving that percentage of the population to the respective vaccinated compartment (V_O, V_Y). The optimal values for u_O, u_Y are obtained through the application of the Optimal Control techniques [48, 67] discussed in the next section. Any person in the states (S, V, U, N) is considered susceptible to getting infected by an infectious or exposed person (E, I). That is because they are either not yet chosen for vaccination, they received the vaccine and are waiting for it to take effect, they received the vaccine and it was ineffective, or they chose not to get vaccinated. The terms $\beta_{ij}, i, j \in \{O, Y\}$ describe the transmission rates of the infection between the age groups and they are analogous to the β parameter in the standard SEIR model. Specifically:

- β_{OO} = rate at which an o65 susceptible person becomes infected by an o65 exposed or o65 infected person
- β_{YO} = rate at which an o65 susceptible person becomes infected by an y65 exposed or y65 infected person
- β_{OY} = rate at which a y65 susceptible person becomes infected by an o65 exposed or o65 infected person
- β_{YY} = rate at which a y65 susceptible person becomes infected by an y65 exposed or y65 infected person

There are three rates taken into consideration in the model, the first being γ_E which is the rate at which an exposed person becomes symptomatic. This rate can be calculated as the inverse of the mean holding time to develop symptoms and become infectious. The next rate is γ_I which is the rate at which an infectious person recovers and can be calculated as the inverse of the mean time to recovery. The final rate considered in the model is γ_V which is the rate at which vaccination becomes effective and can be calculated as the inverse of the mean holding time until protected from the vaccine. The last parameter influencing the model is α_V which is the vaccine effectiveness. This is expressed as the percentage of people who become protected (P_O, P_Y) from the vaccine at a rate γ_V, out of those who have received it (V_O, V_Y).

A.3 Optimal control

Optimal control theory [48,50,67] is the study of strategies to obtain the control function that optimises a certain objective over a time horizon, possibly under suitable constraints. These types of techniques have been widely adopted to biological systems in general [38] but also more specifically to obtain optimal vaccination strategies when dealing with viruses and epidemics [44, 68–70]. More specifically, *Pontryagin's Maximum Principle* [51, 71, 72] is the main tool that is used when dealing with a problem whose dynamics are described by a set of Ordinary Differential Equations.

An optimisation problem can be expressed as the problem of minimising an objective (cost) functional under certain constraints. Let $f(t, x, u)$ denote the objective functional. Also, let $g(t, x, u)$ denote the state equation (or set of state equations) of our system. Using f and g we can form a Hamiltonian function [49, 50] as follows:

$$H(t, x, u, \lambda) = f(t, x, u) + \lambda(t)g(t, x, u) \tag{A.3}$$

where λ is a continuous function of time(t). For simplicity, we will write $\lambda(t)$ as just λ in the expressions that follow.

For the COVID-19 control problem and our spcific model, we can express the goal of our optimisation as the minimisation of the number of infectious individuals, at a minimal cost via vaccination, within a certain time frame $[0, T]$. That goal can be expressed with the help the following objective functional to be minimised:

$$\mathcal{F}(U(t)) = \int_0^T \left[I_O(t) + I_Y(t) + \frac{W_O}{2}u_O^2(t) + \frac{W_Y}{2}u_Y^2(t) \right] dt \tag{A.4}$$

where $U(t) = (u_O(t), u_Y(t))$ and $W_O, W_Y \geq 0$ are the age specific weight constants enforcing the severity of the optimisation constraints. This type of choice of the objective functional as a non-negative increasing function of the control inputs, is common practice in optimal control applications [44, 50, 69]. The control functions are squared in order to ensure the convexity of the functional, guaranteeing a sufficient condition for the existence of an optimal solution (to be proved later in detail). Consider $X(t) = (\ S_O(t),\ V_O(t),\ N_O(t), U_O(t), E_O(t), I_O(t), R_O(t), P_O(t), S_Y(t), V_Y(t), N_Y(t), U_Y(t), E_Y(t), I_Y(t), R_Y(t), P_Y(t)\ )$. We are looking for the optimal pair of solutions $(U^*(t), X^*(t))$, i.e. the optimal control U^* and the corresponding trajectory X^* when U^* is applied, such that

$$\mathcal{F}(U^*(t)) = \min_{\Omega} \mathcal{F}(U(t)) \tag{A.5}$$

where $\Omega = \left\{U(t) \in L^2(O, T)^2 \| a \leq u_O(t), u_Y(t) \leq b, t \in [0, T]\right\}$, a and b are the upper and lower bounds for the control function and can usually be expressed as real values, and T is the time horizon for our optimisation.

There are a few different approaches that can be taken in defining the constrained optimisation problem and are worth mentioning. We can include weighting factors to the infectious compartments (I_O, I_Y) in the objective functional $\mathcal{F}$ that correspond to the mortality rates for each age group or more factors that generally model the cost that large number of infections can have in the healthcare system. Additionally, when it comes to bounding the control functions, we can chose a constraint of the form $a \leq u_O(t)+u_Y(t) \leq b$ which can be interpreted as the total percentage of vaccinations being bounded as opposed to bounding them per age group. This kind of constraint would result in an extra term in the objective functional and the Hamiltonian function having the form $\lambda \cdot (u_O + u_Y)$.

The following theorem describes the main conditions that when satisfied can lead us to the optimal control that solves the minimisation problem.

Theorem 2 (Pontryagin's maximum principle) *Given the Hamiltonian*

$$H(t, x, u, \lambda) = f(t, x, u) + \lambda g(t, x, u), \qquad \text{(A.6)}$$

then the following conditions are satisfied by the optimal control $u^* \in \Omega = \big\{U(t) \in L^2(O,T)^2 \| a \leq u_O(t), u_Y$

$$\frac{\partial H}{\partial u}\begin{cases} = 0 & \text{at } u^* \text{ if } \quad a < u^* < b \\ \geq 0 & \text{at } u^* \text{ if } \quad u^* = a \\ \leq 0 & \text{at } u^* \text{ if } \quad u^* = b \end{cases} \qquad \textit{Optimality Condition} \quad \text{(A.7)}$$

$$\dot{\lambda} = -\frac{\partial H}{\partial x} \qquad \textit{Adjoint Equation} \quad \text{(A.8)}$$

$$\lambda(T) = 0 \qquad \textit{Transversality Condition} \quad \text{(A.9)}$$

$$\dot{x} = g(t, x, u), \quad x(0) = x_0 \qquad \textit{Dynamics of state equation} \quad \text{(A.10)}$$

where $\dot{\lambda} = \partial\lambda/\partial t$ *and* $\dot{x} = \partial x/\partial t$.

In order to apply the maximum principle we first need to define the Hamiltonian function, omitting for brevity the time dependence from the state notations:

$$\begin{aligned}
H &= \left[I_O + I_Y + \frac{W_O}{2}u_O^2 + \frac{W_Y}{2}u_Y^2\right] \\
&+ \lambda_{S_O}\left\{-\left(\frac{\beta_{OO}(E_O + I_O)}{T_O} + \frac{\beta_{YO}(E_Y + I_Y)}{T_Y}\right)S_O - u_O S_O\right\} \\
&+ \lambda_{S_Y}\left\{-\left(\frac{\beta_{OY}(E_O + I_O)}{T_O} + \frac{\beta_{YY}(E_Y + I_Y)}{T_Y}\right)S_Y - u_Y S_Y\right\} \\
&+ \lambda_{V_O}\left\{u_O S_O - \left(\frac{\beta_{OO}(E_O + I_O)}{T_O} + \frac{\beta_{YO}(E_Y + I_Y)}{T_Y}\right)V_O - \gamma_V V_O \right. \\
&\quad \left. -\alpha_V V_O\right\} \\
&+ \lambda_{V_Y}\left\{u_Y S_Y - \left(\frac{\beta_{OY}(E_O + I_O)}{T_O} + \frac{\beta_{YY}(E_Y + I_Y)}{T_Y}\right)V_Y - \gamma_V V_Y \right. \\
&\quad \left. -\alpha_V V_Y\right\} \\
&+ \lambda_{N_O}\left\{\gamma_V V_O - \left(\frac{\beta_{OO}(E_O + I_O)}{T_O} + \frac{\beta_{YO}(E_Y + I_Y)}{T_Y}\right)N_O\right\} \\
&+ \lambda_{N_Y}\left\{\gamma_V V_Y - \left(\frac{\beta_{OY}(E_O + I_O)}{T_O} + \frac{\beta_{YY}(E_Y + I_Y)}{T_Y}\right)N_Y\right\} \\
&- \lambda_{U_O}\left(\frac{\beta_{OO}(E_O + I_O)}{T_O} + \frac{\beta_{YO}(E_Y + I_Y)}{T_Y}\right)U_O
\end{aligned}$$

$$
\begin{aligned}
&- \lambda_{U_Y}\left(\frac{\beta_{OY}(E_O+I_O)}{T_O}+\frac{\beta_{YY}(E_Y+I_Y)}{T_Y}\right)U_Y\\
&+ \lambda_{E_O}\left\{\left(\frac{\beta_{OO}(E_O+I_O)}{T_O}+\frac{\beta_{YO}(E_Y+I_Y)}{T_Y}\right)(S_O+V_O+N_O+U_O)\right.\\
&\left.-\gamma_E E_O\right\}\\
&+ \lambda_{E_Y}\left\{\left(\frac{\beta_{OY}(E_O+I_O)}{T_O}+\frac{\beta_{YY}(E_Y+I_Y)}{T_Y}\right)(S_Y+V_Y+N_Y+U_Y)\right.\\
&\left.-\gamma_E E_Y\right\}\\
&+ \lambda_{I_O}\left\{\gamma_E E_O-\gamma_I I_O\right\}\\
&+ \lambda_{I_Y}\left\{\gamma_E E_Y-\gamma_I I_Y\right\}
\end{aligned}
\qquad \text{A.11}
$$

where $\Lambda := \{\ \lambda_{S_O}, \lambda_{S_Y}, \lambda_{V_O}, \lambda_{V_Y}, \lambda_{N_O}, \lambda_{N_Y}, \lambda_{U_O}, \lambda_{U_Y}, \lambda_{E_O}, \lambda_{E_Y}, \lambda_{I_O}, \lambda_{I_Y}\ \}$ is a set of continuous functions of time ($\Lambda(t)$), given the states and controls. These functions have a key role in our optimisation technique.

Applying each of the conditions of the maximum principle (2) we obtain the following:

- From the optimality condition, the optimal control strategies are given by:

$$u_O^*(t)=\min\left\{\max\left\{a,\frac{S_O(t)}{W_O}\left(\lambda_{S_O}(t)-\lambda_{V_O}(t)\right)\right\},b\right\} \qquad \text{A.12}$$

$$u_Y^*(t)=\min\left\{\max\left\{a,\frac{S_Y(t)}{W_Y}\left(\lambda_{S_Y}(t)-\lambda_{V_Y}(t)\right)\right\},b\right\} \qquad \text{A.13}$$

- Adjoint Equation: $\dot{\Lambda}=-\partial H/\partial X$ which results in the following system of ODEs:

$$
\begin{aligned}
\dot{\lambda}_{S_O} &= \left(\frac{\beta_{OO}(E_O+I_O)}{T_O}+\frac{\beta_{YO}(E_Y+I_Y)}{T_Y}+u_O\right)\lambda_{S_O}-u_O\lambda_{V_O}\\
&- \left(\frac{\beta_{OO}(E_O+I_O)}{T_O}+\frac{\beta_{YO}(E_Y+I_Y)}{T_Y}\right)\lambda_{E_O} && \text{A.14}\\
\dot{\lambda}_{S_Y} &= \left(\frac{\beta_{OY}(E_O+I_O)}{T_O}+\frac{\beta_{YY}(E_Y+I_Y)}{T_Y}+u_Y\right)\lambda_{S_Y}-u_Y\lambda_{V_Y}\\
&- \left(\frac{\beta_{OY}(E_O+I_O)}{T_O}+\frac{\beta_{YY}(E_Y+I_Y)}{T_Y}\right)\lambda_{E_Y} && \text{A.15}\\
\dot{\lambda}_{V_O} &= \left(\frac{\beta_{OO}(E_O+I_O)}{T_O}+\frac{\beta_{YO}(E_Y+I_Y)}{T_Y}+\gamma_V\right)\lambda_{V_O}-(1-\alpha_V)\gamma_V\lambda_{N_O}\\
&- \left(\frac{\beta_{OO}(E_O+I_O)}{T_O}+\frac{\beta_{YO}(E_Y+I_Y)}{T_Y}\right)\lambda_{E_O} && \text{A.16}\\
\dot{\lambda}_{V_Y} &= \left(\frac{\beta_{OY}(E_O+I_O)}{T_O}+\frac{\beta_{YY}(E_Y+I_Y)}{T_Y}+\gamma_V\right)\lambda_{V_Y}-(1-\alpha_V)\gamma_V\lambda_{N_Y}
\end{aligned}
$$

$$
\begin{aligned}
& - \left(\frac{\beta_{OY}(E_O+I_O)}{T_O}+\frac{\beta_{YY}(E_Y+I_Y)}{T_Y}\right)\lambda_{E_Y} && \boxed{\text{A.17}}\\
\dot{\lambda}_{N_O} &= \left(\frac{\beta_{OO}(E_O+I_O)}{T_O}+\frac{\beta_{YO}(E_Y+I_Y)}{T_Y}\right)(\lambda_{N_O}-\lambda_{E_O}) && \boxed{\text{A.18}}\\
\dot{\lambda}_{N_Y} &= \left(\frac{\beta_{OY}(E_O+I_O)}{T_O}+\frac{\beta_{YY}(E_Y+I_Y)}{T_Y}\right)(\lambda_{N_Y}-\lambda_{E_Y}) && \boxed{\text{A.19}}\\
\dot{\lambda}_{U_O} &= \left(\frac{\beta_{OO}(E_O+I_O)}{T_O}+\frac{\beta_{YO}(E_Y+I_Y)}{T_Y}\right)(\lambda_{U_O}-\lambda_{E_O}) && \boxed{\text{A.20}}\\
\dot{\lambda}_{U_Y} &= \left(\frac{\beta_{OY}(E_O+I_O)}{T_O}+\frac{\beta_{YY}(E_Y+I_Y)}{T_Y}\right)(\lambda_{U_Y}-\lambda_{E_Y}) && \boxed{\text{A.21}}\\
\dot{\lambda}_{E_O} &= \frac{\beta_{OO}S_O}{T_O}(\lambda_{S_O}-\lambda_{E_O})+\frac{\beta_{OO}V_O}{T_O}(\lambda_{V_O}-\lambda_{E_O})\\
&+ \frac{\beta_{OO}N_O}{T_O}(\lambda_{N_O}-\lambda_{E_O})+\frac{\beta_{OO}U_O}{T_O}(\lambda_{U_O}-\lambda_{E_O})\\
&+ \frac{\beta_{OY}S_Y}{T_O}(\lambda_{S_Y}-\lambda_{E_Y})+\frac{\beta_{OY}YV_t}{T_O}(\lambda_{V_Y}-\lambda_{E_Y})\\
&+ \frac{\beta_{OY}N_Y}{T_O}(\lambda_{N_Y}-\lambda_{E_Y})+\frac{\beta_{OY}U_Y}{T_O}(\lambda_{U_Y}-\lambda_{E_Y})\\
&+ \gamma_E\lambda_{E_O}-\gamma_E\lambda_{I_O} && \boxed{\text{A.22}}\\
\dot{\lambda}_{E_Y} &= \frac{\beta_{YO}S_O}{T_Y}(\lambda_{S_O}-\lambda_{E_O})+\frac{\beta_{YO}V_O}{T_Y}(\lambda_{V_O}-\lambda_{E_O})\\
&+ \frac{\beta_{YO}N_O}{T_Y}(\lambda_{N_O}-\lambda_{E_O})+\frac{\beta_{YO}U_O}{T_Y}(\lambda_{U_O}-\lambda_{E_O})\\
&+ \frac{\beta_{YY}S_Y}{T_Y}(\lambda_{S_Y}-\lambda_{E_Y})+\frac{\beta_{YY}V_Y}{T_Y}(\lambda_{V_Y}-\lambda_{E_Y})\\
&+ \frac{\beta_{YY}N_Y}{T_Y}(\lambda_{N_Y}-\lambda_{E_Y})+\frac{\beta_{YY}U_Y}{T_Y}(\lambda_{U_Y}-\lambda_{E_Y})\\
&+ \gamma_E\lambda_{E_Y}-\gamma_E\lambda_{I_Y} && \boxed{\text{A.23}}\\
\dot{\lambda}_{I_O} &= \frac{\beta_{OO}S_O}{T_O}(\lambda_{S_O}-\lambda_{E_O})+\frac{\beta_{OO}V_O}{T_O}(\lambda_{V_O}-\lambda_{E_O})\\
&+ \frac{\beta_{OO}N_O}{T_O}(\lambda_{N_O}-\lambda_{E_O})+\frac{\beta_{OO}U_O}{T_O}(\lambda_{U_O}-\lambda_{E_O})\\
&+ \frac{\beta_{OY}S_Y}{T_Y}(\lambda_{S_Y}-\lambda_{E_Y})+\frac{\beta_{OY}V_Y}{T_O}(\lambda_{V_Y}-\lambda_{E_Y})\\
&+ \frac{\beta_{OY}N_Y}{T_O}(\lambda_{N_Y}-\lambda_{E_Y})+\frac{\beta_{OY}U_Y}{T_O}(\lambda_{U_Y}-\lambda_{E_Y})\\
&+ \gamma_I\lambda_{I_O}-1 && \boxed{\text{A.24}}\\
\dot{\lambda}_{I_Y} &= \frac{\beta_{YO}S_O}{T_Y}(\lambda_{S_O}-\lambda_{E_O})+\frac{\beta_{YO}V_O}{T_Y}(\lambda_{V_O}-\lambda_{E_O})\\
&+ \frac{\beta_{YO}N_O}{T_Y}(\lambda_{N_O}-\lambda_{E_O})+\frac{\beta_{YO}U_O}{T_Y}(\lambda_{U_O}-\lambda_{E_O})\\
&+ \frac{\beta_{YY}S_Y}{T_Y}(\lambda_{S_Y}-\lambda_{E_Y})+\frac{\beta_{YY}V_Y}{T_Y}(\lambda_{V_Y}-\lambda_{E_Y})
\end{aligned}
$$

$$
\begin{aligned}
&+ \quad \frac{\beta_{YY} N_Y}{T_Y}(\lambda_{N_Y} - \lambda_{E_Y}) + \frac{\beta_{YY} U_Y}{T_Y}(\lambda_{U_Y} - \lambda_{E_Y}) \\
&+ \quad \gamma_I \lambda_{I_Y} - 1 \qquad \text{(A.25)}
\end{aligned}
$$

where $\dot{\lambda}$ denotes $\partial\lambda/\partial t$

- The transversality condition: $\Lambda(T) = 0$ gives us a terminal condition for the system of adjoint ODEs. This condition implies that the adjoint system should be solved backwards [73] as opposed to the state system.
- The dynamics of the system are described by the system of state equations (A.2).

To ensure the existence of a solution to our optimisation problem, we need to examine the sufficient conditions as stated in [48], as also used in [74]. The relevant conditions and the corresponding existence proof for our model can be found in Appendix A.8.

Having obtained all the necessary optimality conditions and resulting equations, we can iterate between solving the state equations forwards in time and solving the adjoint equations backwards in time while updating the control functions after each iteration. With each update of the control functions, the state vectors approach their optimal trajectories, finally converging when the control functions converge to the optimal ones. We choose a small positive number δ as a threshold for the error between two consecutive solutions of the state system of ODEs. Algorithm 11 describes the steps that lead to the computation of an optimal control trajectory.

Algorithm 11: Determining the optimal control

```
  Result: Optimal control and trajectory $U^*, X^*$
  Input: Initial state $x_0$, Parameter values, Time horizon, Initial control,
         Convergence threshold $\delta$
1 Solve the system of state ODEs forwards in time to determine the state;
2 Solve the system of adjoint ODEs backwards in time;
3 Update the controls using (A.12) and (A.13) ;
4 Compute the error term $\frac{\|x_k - x_{k-1}\|_1}{\|x_{k-1}\|_1}$ ;
5 If the error term is smaller than some predetermined threshold $\delta$ then extract the
    optimal control and optimal trajectory. Otherwise repeat the process until
    convergence.
```

A.4 Simulation Setup and Results

We used our model to estimate the evolution of the virus in the Republic of Ireland, starting from January 2021 using data provided by [64] regarding the population of the country, the number of exposed, infected and recovered people in both age groups. Additionally, we applied the optimal control strategy on the same data to obtain the effect that the vaccination would have on the evolution of the pandemic, were it adopted by the state.

Table A.1 includes the initial values of the states $E_O, I_O, R_O, E_Y, I_Y, R_Y$. The rest of the states can be calculated with the use of this information and some of the parameter values. The total susceptible people in each age group are the people not yet vaccinated along with the people refusing or unable to receive the vaccine ($\mathbf{S} := S+U$). This total of susceptibles for each age group can be calculated by: $\mathbf{S} = T-(E+I+R)$. The separation between not yet vaccinated (S) and unwilling to get vaccinated (U) can be expressed as a percentage for each population, representing the refusal rate for the vaccine. Let r_O and r_Y be these percentages in the over 65 and under 65 populations respectively. Then $S_Y = (1-r_Y)\mathbf{S}$, $U_Y = r_Y\mathbf{S}$, $S_O = (1-r_O)\mathbf{S}$, $U_O = r_O\mathbf{S}$.

State (Compartment)	Number of people
Total y65 (T_Y)	4000000
Total o65 (T_O)	900000
Y65 exposed (E_Y)	2000
O65 exposed (E_O)	200
Y65 infected (I_Y)	2000
O65 infected (I_O)	200
Y65 recovered (R_Y)	200000
O65 recovered (R_O)	100000

Table A.1: Initial values for the model states

We produce two different simulations, to study the evolution of the pandemic, following the steps described in Algorithm 11. In the first case we assume strict measures are in place, meaning that the transmission rates of the virus among individuals is low. In the second case we assume minimal restrictions are in place, hence the transmission rates are much higher. The parameters used in both simulations are listed in Table A.2.

The three parameters t_E, t_I, t_V, describe the mean holding times in each of the states E, I and V. They are the inverses of the rates we have in our model (A.2), namely $\gamma_E, \gamma_I, \gamma_V$. Also, particularly interesting are the four RO parameters, that are related to how quickly the infection is transmitted between the members of the entire population. More specifically, these parameters are directly related to the terms β_{ij} in our model which describe the transmission rates, with the help of the three mean holding times:

$$\beta_{OO} = \frac{R0_{OO}}{t_E+t_I+t_V}, \beta_{YO} = \frac{R0_{YO}}{t_E+t_I+t_V}, \\ \beta_{OY} = \frac{R0_{OY}}{t_E+t_I+t_V}, \beta_{YY} = \frac{R0_{YY}}{t_E+t_I+t_V} \tag{A.26}$$

Parameter Description	Symbol	Case 1	Case 2	Unit
Mean no of o65 infected by an o65	$R0_{OO}$	1.2	8	No of people
Mean no of y65 infected by a y65	$R0_{YY}$	1.2	8	No of people
Mean no of y65 infected by an o65	$R0_{OY}$	0.9	4	No of people
Mean no of o65 infected by a y65	$R0_{YO}$	0.9	3	No of people
Mean holding time exposed in days	t_E	6.6		Days
Mean holding time infected in days	t_I	7.4		Days
Mean holding time vaccinated in days	t_V	14		Days
Vaccine effectiveness	α_V	0.9		Percentage
Upper bound for control function	b	0.3		Percentage
Refusal rate for o65	r_O	0.07		Percentage
Refusal rate for y65	r_Y	0.21		Percentage
Weight constant relating to o65 group	W_O	10^{11}		Unitless
Weight constant relating to y65 group	W_Y	10^{11}		Unitless
Error threshold for convergence	δ	0.0005		Unitless

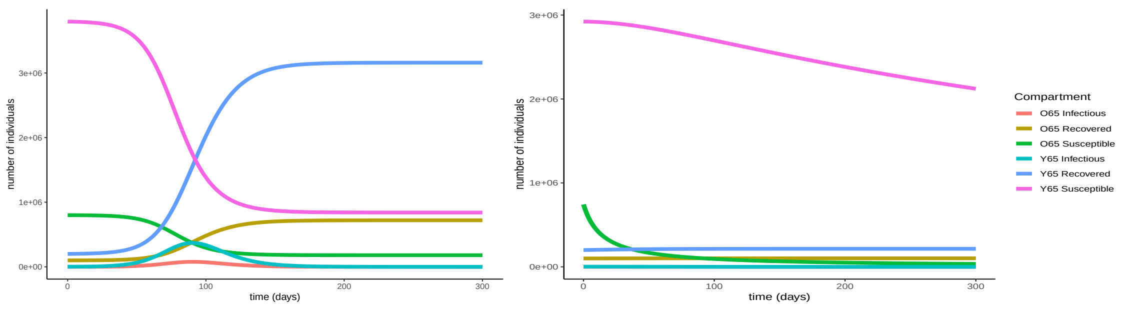
Table A.2: Parameter values for both simulations. The mean numbers of infections from each infected person within each age group and between age groups vary for the two simulations, while for the rest of the parameters we used the same values. Case 1 is representative of a situation where strict measures are in place and the results of the simulation can be found in subsection A.4.1. Case 2 is an example of a situation with minimal restrictions and is explored in subsection A.4.2.

Two different sets of $R0$ numbers are used to produce our two simulations and give an insight to the evolution of the pandemic under different levels of restrictions.

The values we use for the parameters are estimates specific to COVID-19 (mean holding times) [75], or based on estimates used in similar studies of other infectious diseases such as (weight constants) [44]. Parameter values specific to Ireland (such as transmission rates during and after lockdown), were chosen according to the work done in [65]. The upper bound for the control function (b) is chosen arbitrarily, however it is not a value that is ever obtained for either control function, thanks to the high weight constants used in the objective functional that model the cost of the vaccination and the severity of the constraints. An alternative approach would be for b to bound the sum of the control functions as opposed to each of them individually. The weight constants W_O, W_Y are chosen to be equal because we assume no difference in the cost of providing the vaccination to people in different age groups. However, the two groups can be weighed differently, explaining the cost of bringing the vaccination to remote locations, nursing homes etc. The refusal percentages for both age groups (r_O, r_Y) we used are based on a survey [76], the results of which suggest that 77% of the overall population of Ireland are willing to receive the vaccine and the the willingness is stronger in the over 65 population, with 93% of the group intending to get vaccinated. We included the percentage of people unsure about the vaccination (15% overall) in the refusal percentage (6%) along with the people who claimed to already be against receiving the vaccination. Finally, as we mentioned in section A.3, it is possible for one more set of parameters to be included in our model and more specifically the optimisation function, weighing the number of infectious individuals by the mortality rate for each age group.

A.4.1 Evolution of pandemic under tight restrictions

We simulate the evolution of the pandemic using the initial state values given in Table A.1 and the parameter values given in Table A.2. Specifically, the $R0$ numbers we use are given in the column named Case 1 and are an example of the way the virus is evolving under a strict restriction policy. In that case, the average number of infections within each age group is 1.2 and between age groups 0.9.

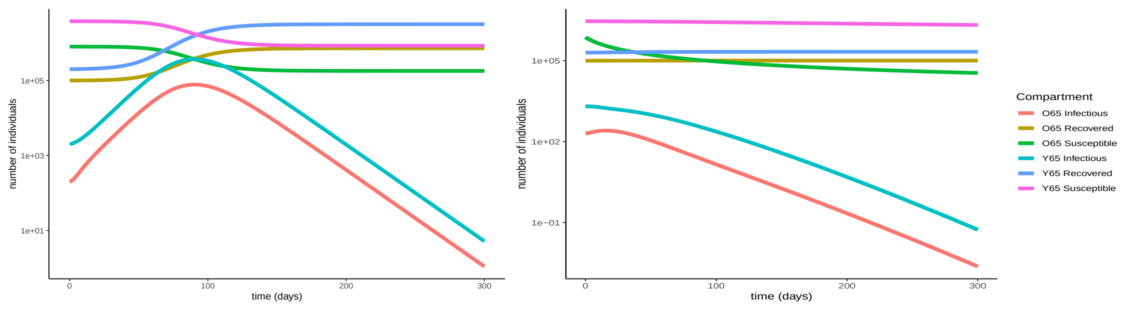


(a) Evolution of states without the vaccine

(b) Evolution of states with the vaccine

Figure A.2: Evolution of the pandemic with and without vaccination under strict measures. In the case where no vaccination policy is in place, there is an early peak for the infectious individuals in both age groups (red and turquoise curves). In the second graph the peaks are almost indistinguishable. This is due to the vaccination reducing both the total and the maximum number of infections that take place, hence flattening the curve. The number of susceptible people (pink and blue curves) in both age groups declines fast in the first graph, due to most of them getting infected, while in the second case the decline is much slower, as many individuals are getting vaccinated and thus protected from infection.

Figure A.2 shows the difference between the evolution of the pandemic when there is no control in place and when the optimal control is applied. Due to the transmission rates being quite low, the increase in the number of infectious individuals will not be particularly evident, especially when the vaccination is applied. For that reason we represented the same information also on the log scale in Figure A.3 to give a clearer picture of the difference an optimal vaccination strategy makes to the evolution of the pandemic. In the case where no control is applied there is a peak in the number of infectious individuals in both age groups and as a result, the number of susceptible people drops fast with the pandemic ending in a very short period of time, namely less than 250 days, with the



(a) Evolution of states without the vaccine (b) Evolution of states with the vaccine

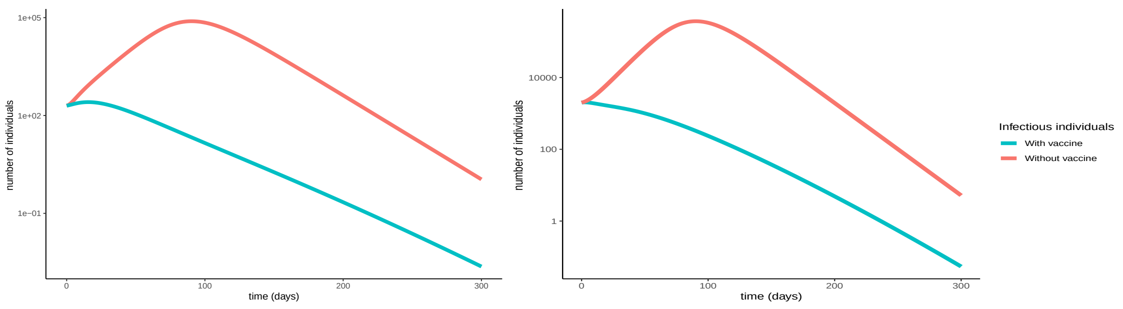
Figure A.3: Evolution of the pandemic with and without vaccination under strict measures. The graphs are produced in vertical log scale to give a clearer view of the peaks. In the first graph there are clear peaks in the number of infectious individuals in both age groups while in the second, the curves are clearly flattened thanks to the vaccination.

greatest part of the population having been infected. Specifically, 80% of the population over 65 (720,249) and 79% of the population under 65 (3,159,510) gets infected by the virus until the end of the pandemic. That means that around 79% of the total population of the country (3,879,759) will get infected by the virus when no vaccination strategy is in place.

On the other hand, when the optimal vaccination strategy is applied, the infectious curve is flattened to the point where it becomes nearly indistinguishable from the horizontal axis. This means that the number of people to become infected by the virus is substantially reduced as a result of the protection provided by the vaccine. In particular, only 11.3% (101,764) and 5.38% (215,132) of the over and under 65 years old groups respectively will get infected by the virus, resulting in a total 6.47% of the population (316,896). This is achieved thanks to the 70.58% (635,187) and 17.08% (683,044) of the over 65 and under 65 populations respectively receiving the vaccine and being successfully protected by it. In total, that is 26.9% of the population (1,318,231) being successfully vaccinated.

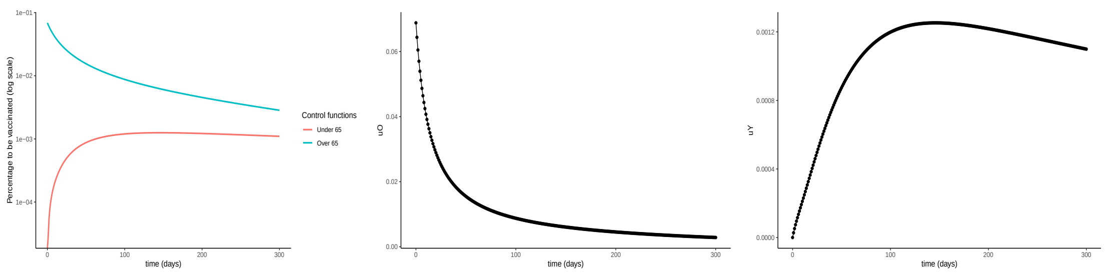
The difference that the control function makes to the number of infectious individuals (in the log scale) can be more clearly seen in Figure A.4. For both age groups, the curve is substantially flattened as a result of the vaccination. This is a visual representation of the reduction in the total number of infections and also showcases the fact that less people will be infected at the same time, causing less stress on the healthcare system.

The optimal vaccination strategy for each age group, i.e., the control functions that resulted from the optimisation technique described in section A.3 can be seen in Figure A.5. As expected, close to the start of our simulations, a high percentage of the susceptible population gets vaccinated to quickly get the pandemic under control, and then that percentage continuously declines. It is interesting to note how the o65 population



(a) Evolution of infectious in over 65 (b) Evolution of infectious in under 65

Figure A.4: Comparison of baseline curves (without vaccine) with optimal vaccination strategy curves for the infectious populations in both age groups under strict measures. The vertical scale is in log units. The vaccination successfully flattens the curve in both cases, resulting in fewer infections overall as well as fewer simultaneous infections.



(a) Optimal control functions (b) Optimal control for over65 (c) Optimal control for under65

Figure A.5: Optimal control functions for both age groups. A.5a presents both control functions on a vertical log scale. The o65 group gets vaccinated faster and in higher percentages, while the vaccination of the y65 group commences but proceeds slower and continues after the pandemic is under control.

gets vaccinated earlier and with higher percentages while the vaccination of the younger population is happening slower, with smaller percentages every day, and goes on after the pandemic is over as can be seen in Figure A.5. This is also evident in the percentages of the people in each age group that get vaccinated by the end of the simulation, with 70.58% and 17.08% for the over and under 65 groups respectively, as described earlier.

It is worth noting that the parameter setup used for this simulation is indicative of a national lockdown situation, which is an unrealistic scenario for this length of time. For this reason, we looked into a second parameter setup, discussed in the next section.

A.4.2 Evolution of pandemic under loose restrictions

We now simulate the evolution of the pandemic in Ireland, using the same initial conditions for the states as defined in Table A.1. We are interested in investigating the effect a different choice of parameters, specifically different $R0$ numbers, would have on our results. We are looking into values that are indicative of an extreme scenario where the measures in place are not strict or even non existent and each infected individual can infect multiple individuals before recovering. The column titled Case 2 in Table A.2 consists of values chosen for the parameters, while the rest of the parameters remain the same as in subsection A.4.1.

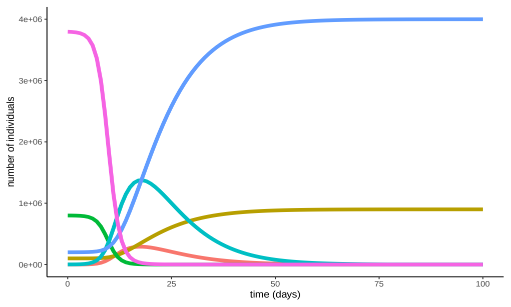
When no vaccination policy is in place, the curves of infectious individuals in both age groups present a very high peak due to the high transmission rates as can be seen in Figures A.6 and A.7. This peak is indicative of a period of time where a very large number of people are infected at the same time, a situation that would cause a huge strain on the country's healthcare system. The percentages of people that get infected in the over and under 65 group respectively are 99.995% (899,954) and 99.993% (3,999,713), resulting in a total 99.993% (4,899,667) of the population. This means that nearly every single resident of the country will get infected by the virus in a very short period of time, while at the same time paralysing the healthcare system.

That curve gets flattened, with a lower peak and greater spread, thanks to the optimal control functions, i.e. the optimal vaccination strategy. This would offer some relief to the healthcare system, as not all infections will be active at the same time. The percentages of people that get infected in the over and under 65 group respectively are 85.29% (767,614) and 99.1% (3,965,963), resulting in a total 96.6% of the population (4,733,577). These percentages, though reduced, are still very high and that is due to the fact that the chosen transmission rates are so high. For the same reason the percentages of the people that get successfully protected from the vaccine are relatively low, namely 14.2% (127,871) and 0.1% (3,861) for the over 65 and under 65 groups respectively, meaning a total 2.7% of the population (131,732) will be successfully vaccinated. There is an additional reason to why these percentages are so low. From the moment an individual receives the vaccine, there is a period equal to the *mean holding time vaccinated* (t_V) during which the person may still get infected. Due to the high transmission rates, a lot of vaccinated individuals will get infected, hence not making it to the protected state.

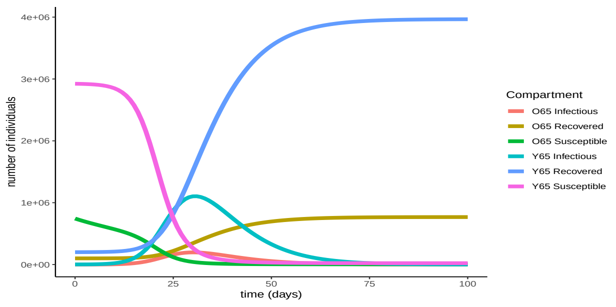
A side by side presentation of the evolution of the pandemic with and without vaccination is given in Figure A.6 and a comparison of the optimised with the baseline curves for the infectious populations in each age group are given in Figure A.7.

Figure A.8 shows the optimal control functions that can be obtained as the solution to our optimisation problem. Similar to the previous simulations, the control function curves present a peak at first, indicating a high percentage of the population being chosen for vaccination early on and that percentage getting reduced as time progresses.

It is interesting to note that in both cases above, the upper bound for the control functions is never reached. This is due to the high weights W_O, W_Y associated with u_O and u_Y in the objective functional A.4. The weights represent the cost of acquiring

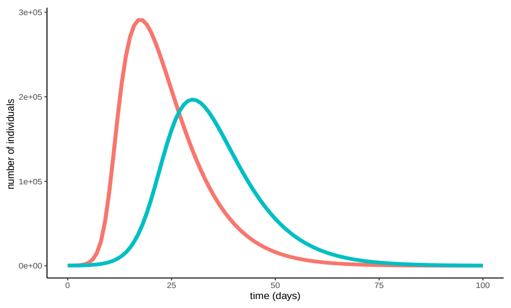


(a) Evolution of states without the vaccine

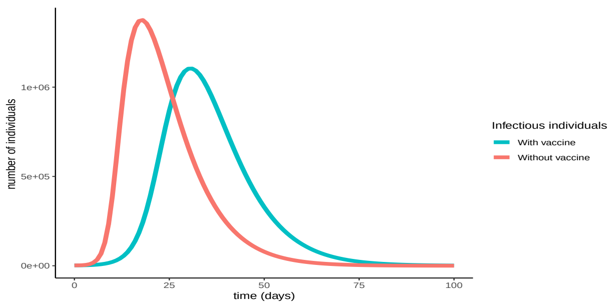


(b) Evolution of states with the vaccine

Figure A.6: Evolution of the pandemic with an without vaccination under loose measures. Without a vaccination policy in place, the curve of infectious individuals presents an earlier and higher peak as opposed to the case where the vaccination strategy is applied and the curve is flattened.



(a) Evolution of infectious in over 65



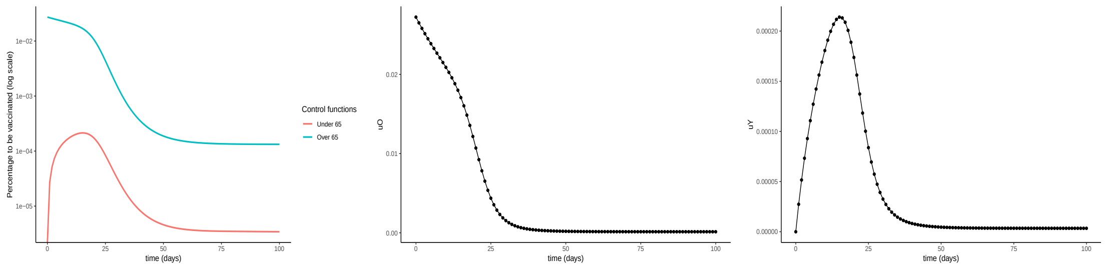
(b) Evolution of infectious in under 65

Figure A.7: Comparison of baseline curves (without vaccine) with optimal vaccination strategy curves for the infectious populations in both age groups under loose restrictions.

and distributing the vaccines, which was particularly high when they were first becoming available. We explored the option of choosing lower weights, namely 10^8 and 10^5 and ran the simulations for the case of loose restrictions, with all other parameters kept the same. In both cases, the function $u^O(t)$ was obtained to be constant and equal to the upper bound $b = 0.6$ for all t. When choosing $W_O = W_Y = 10^5$, the function $u^Y(t)$ is also found to be equal to b for all time points, while when $W_O = W_Y = 10^8$ looks similar to A.8c but the peak is taller, reaching about 0.04, and narrower. As expected, the curve of infection gets reduced and flattened more, the lower the weights get.

A.5 Comparative Study

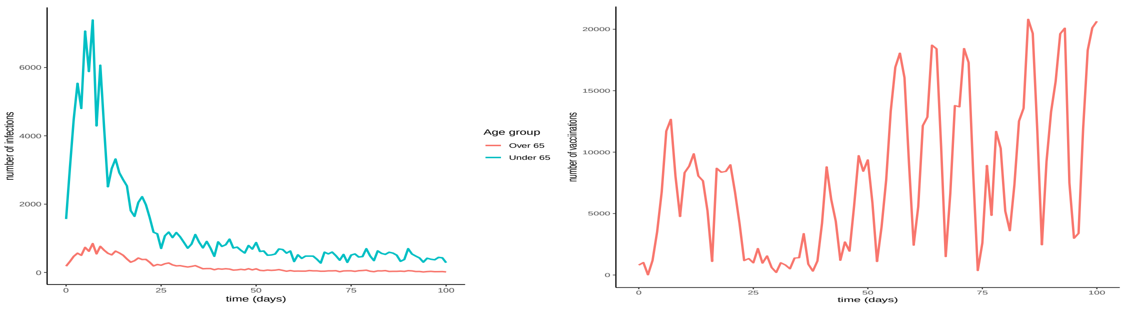
We now look at the effect that the optimal control obtained from our model would have on the evolution of the pandemic in Ireland by comparing our approach to the real infection numbers, given the vaccination strategy taken by the Irish government. In order for



(a) Optimal control functions (b) Optimal control for over65 (c) Optimal control for under65

Figure A.8: Optimal control functions for both age groups. A.8a presents both control functions in log scale.

the comparison to be sensible, we limit our simulations to a time frame of 100 days, thus ensuring that the transmission rates remain somewhat constant. This is needed because our model does not account for varying $R0$ numbers, while in reality these numbers can fluctuate depending on the restrictive measures in place (e.g. lockdowns) and the preventative measures taken, namely vaccinations. We obtained our data from [64], specifically the daily infection and vaccination numbers starting from January 1st 2021. These values are shown in FigureA.9.

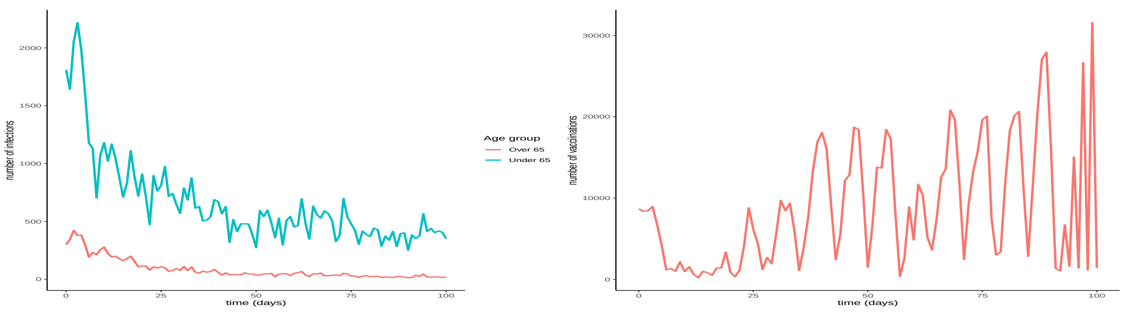


(a) Infectious individuals (b) Vaccines administered

Figure A.9: Infection and vaccination numbers for the Republic of Ireland starting from January 1st 2021

It is evident that there is a substantial peak in the infection numbers during the first few days which can be attributed to the looser measures in place before the Christmas holidays of 2020. Considering this, we produce a second set of comparative plots, starting the simulation just after the peak in infections, namely on January 18th instead of the 1st. The daily infection and vaccination numbers for a period of 100 days starting from January 18th can be seen in FigureA.10.

Our model requires a specific upper bound for the control functions in order for the optimisation objective to be defined, as seen in Equation A.5. One choice for that upper bound could be the average of the daily vaccinations over the time period we are studying.



(a) Infectious individuals (b) Vaccines administered

Figure A.10: Infection and vaccination numbers for the Republic of Ireland starting from January 18th 2021

However, it is evident from Figures A.9b and A.10b that there is a great fluctuation in the number of vaccines that were administered which can be attributed to the limited availability of vaccines at the start of 2021. This leads us to assume that the number of vaccines administered each day was the maximum of available vaccines for that day. With that in mind, we designed a time varying upper bound, using each daily vaccination number as the upper bound for the control function for that day. Additionally, while the data provides the total number of vaccines to be administered to the population each day, our model accounts for two age groups, each of them requiring a separate upper bound. To address this issue, we apply complimentary percentages of the daily vaccination numbers as upper bounds to each control function. Specifically, and in agreement with the policy taken by the Irish government, we consider a higher percentage of the vaccines to be made a available to the over 65 group. We choose 80% of the available vaccinations to be used for the over-65 group and let $V_{No}(t)$ be the number of vaccines administered at time (day) t. Then, for time t, the control functions are bounded as follows:

$$u_O(t)S_O(t) < 0.8 \cdot V_{No}(t) \text{ and } u_Y(t)S_Y(t) < 0.2 \cdot V_{No}(t) \tag{A.27}$$

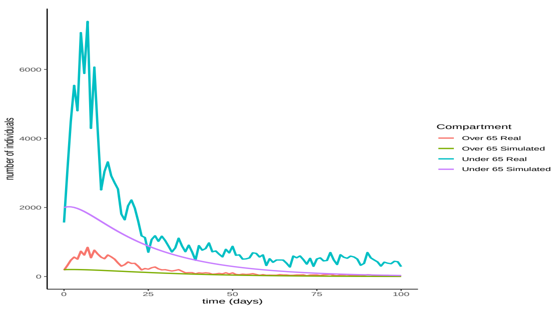
where each control function is multiplied by the corresponding susceptible population.

We next set the mean number of people infected by an infectious person, namely the R0 number. Our model includes 4 age-specific R0 numbers associated with the β parameters in our model, according to Equation (A.26). In reality, the R0 number greatly fluctuates depending on individual behaviours and government restrictions. For that reason, we explore three different choices for the set of age-specific R0 numbers. These choices are listed in Table A.3 along with references to the respective plots.

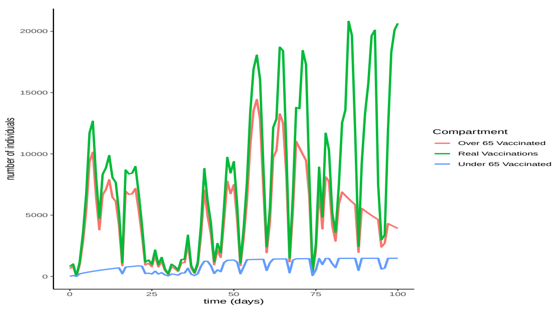
From Figures A.11 to A.14 it is evident that the optimal vaccination strategy obtained from our model is successful in quickly reducing the number of infections even in the case of an average R0 number of 1.9. This value for R0 is unrealistic, especially for January

$R0_{OO}$	$R0_{YY}$	$R0_{OY}$	$R0_{YO}$	Average $R0$	Plot before peak	Plot after peak
1	1	0.3	0.3	1.3	Figure A.11	Figure A.12
1	1	0.9	0.9	1.9	Figure A.15	Figure A.16
1.2	1.2	0.3	0.3	1.5	Figure A.13	Figure A.14

Table A.3: Chosen values for the $R0$ numbers (first four columns). The fifth column corresponds to the weighted average of the age-specific R0 numbers. The sixth and seventh column refer to the corresponding plots with the comparative simulation results, using Irish data before and after the peak in the number of infections respectively.

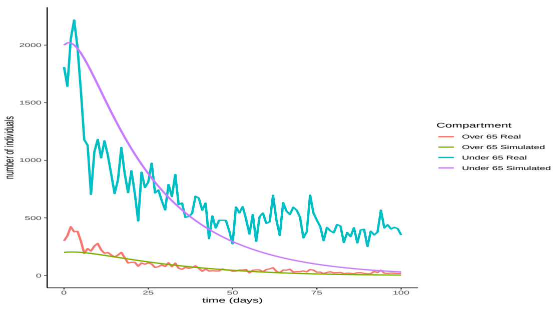


(a) Number of infections

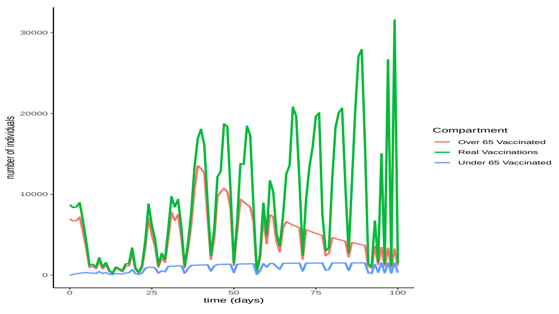


(b) Number of vaccinations

Figure A.11: Comparison of the infection and vaccination numbers for the Republic of Ireland starting from January 1st 2021. The R0 numbers used are equal to 1 within each age group and 0.3 between age groups.



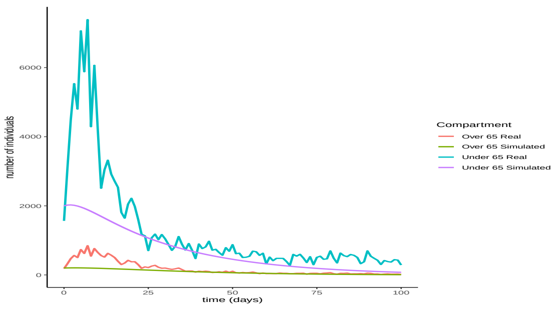
(a) Infected individuals



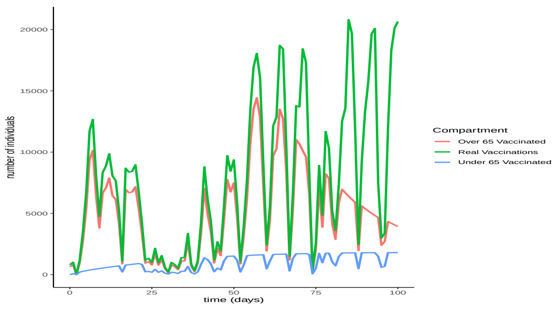
(b) Vaccines administered

Figure A.12: Comparison of the infection and vaccination numbers for the Republic of Ireland starting from January 18th 2021.The R0 numbers used are equal to 1 within each age group and 0.3 between age groups.

2021, when Ireland was under strict lockdown measures. However, in this extreme case, while the total number of infections is high compared to the real numbers, as expected, the decline of the infection curve is steep, as can be seen in Figure A.16. This implies that even if the country was not under total lockdown during these days, the optimal vaccination strategy would be able to bring the infections under control within a short amount of time.

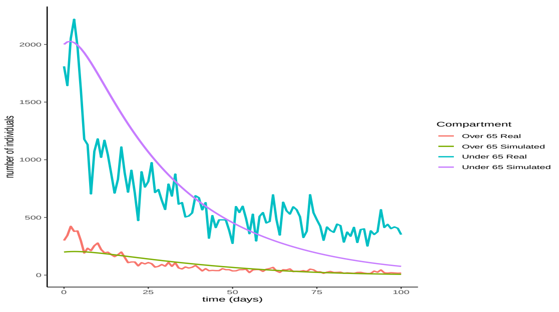


(a) Number of infections

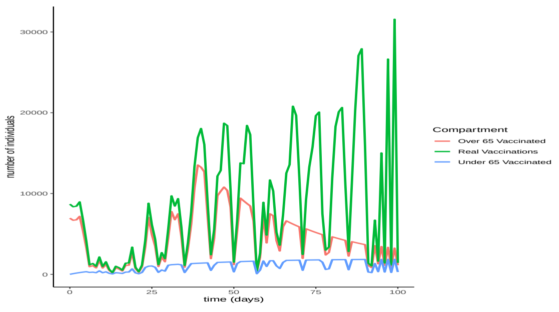


(b) Number of vaccinations

Figure A.13: Comparison of the infection and vaccination numbers for the Republic of Ireland starting from January 1st 2021. The R0 numbers used are equal to 1.2 within each age group and 0.3 between age groups.

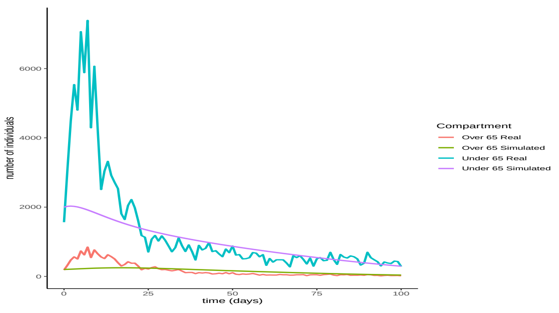


(a) Infected individuals

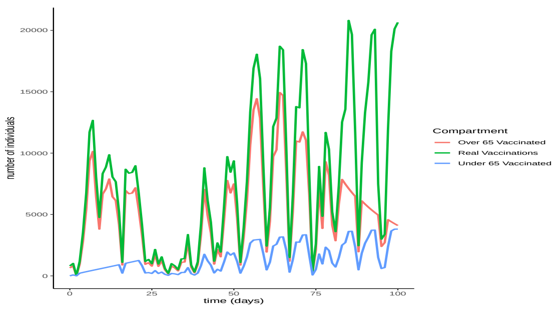


(b) Vaccines administered

Figure A.14: Comparison of the infection and vaccination numbers for the Republic of Ireland starting from January 18th 2021.The R0 numbers used are equal to 1.2 within each age group and 0.3 between age groups.



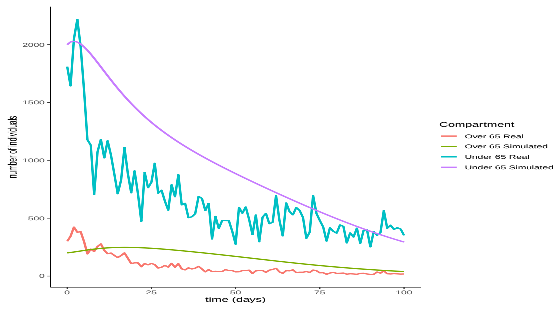
(a) Number of infections



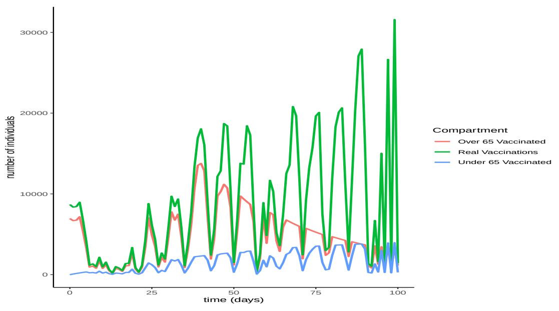
(b) Number of vaccinations

Figure A.15: Comparison of the infection and vaccination numbers for the Republic of Ireland starting from January 1st 2021. The R0 numbers used are equal to 1 within each age group and 0.9 between age groups.

The successful control of infections for all choices of the $R0$ numbers can be attributed to the distribution of the vaccines to both age groups. The policy adopted by the Irish



(a) Infected individuals



(b) Vaccines administered

Figure A.16: Comparison of the infection and vaccination numbers for the Republic of Ireland starting from January 18th 2021.The R0 numbers used are equal to 1 within each age group and 0.9 between age groups.

government mainly entailed administering vaccines to the over 65 population as are the higher risk group. However, it is evident from our simulation that a policy where a portion of the vaccines are additionally made available to the under 65 group can be more beneficial.

It is interesting to note that in every simulation, the total number of daily vaccinations resulting from the optimisation are often less than the real vaccination numbers, i.e. the daily maximum number of available vaccines. This is a very encouraging observation, considering the implication that less vaccines can be used to achieve comparable or even better results. It is a particularly significant result if the optimisation is targeted towards countries that have gained limited access to vaccines.

A.6 Discussion

Since the end of February 2020, the Republic of Ireland has been affected by the COVID-19 pandemic, like most countries in the world. For the greater part of the year no effective treatment or vaccine was present, leaving the government with one tool to attempt to flatten the curve of infection: the application of various levels of restrictions, depending on the occurrence of new daily cases. In light of the vaccines that became available near the end of 2020, a need arose for efficient vaccination plans based on the availability of the vaccines and the risk levels of different groups of the population.

In this work, we introduced a novel compartmental model which more adequately describes the dynamics of the virus compared to the approaches taken thus far in the literature. We added compartments representing the stages of infection and vaccination, and assumed two different age groups sharing the same model structure. Using this model as a baseline to describe the evolution of the pandemic in the Republic of Ireland, we applied Optimal Control methods to obtain a suggested vaccination strategy that minimises the number of infections under certain restrictions such as the available vaccines. We simulated the evolution of the pandemic with and without the vaccination strategy in place for two different scenarios, indicative of strict and loose restrictions respectively. The main conclusions we drew were:

(1) The optimal control strategy in both scenarios successfully flattens the curve of infectious individuals, ensuring that less people will get infected by the virus due to protection from the vaccine and, most importantly, that less people will be simultaneously infected thus avoiding exhausting the national healthcare system.

(2) In the case of low transmission rates, the vaccination not only flattens the curve but also significantly reduces the total number of infections, specifically from 80% to 11.3% for the over 65 group and from 79% to 5.38% for the under 65 group. However, the pandemic lasts a long time, meaning that a big part of the population remains in the susceptible compartment, not yet vaccinated but also not infected due to the low transmission rates, as opposed to being moved to the protected or recovered compartments.

(3) When the transmission rates are high however, the vaccination reduces the number of infections by a much smaller percentage, namely from 99.995% to 85.29% for the over 65 group and from 99.993% to 99.1% for the under 65 group. However it still succeeds in flattening the curve and ensuring less simultaneous infections, hence preventing a healthcare crisis. The pandemic in this case ends in a very short period of time with the biggest part of the population having been infected and eventually recovered.

(4) For both scenarios, the optimal strategy suggests focusing on vaccinating the older population in higher percentages first, while simultaneously vaccinating part of the younger population. This is a different approach to the one taken by many states, including Ireland, where the older population gets exclusively vaccinated first.

All of the above lead us to conclude that an approach involving a combination of strict and loose measures would be ideal, while the vaccination program is taking place. That would ensure that the infection doesn't spread to such a high percentage of the population while the restrictions are not as severe so as to make the situation unbearable for a certain length of time.

To verify our findings, we compared the results of our simulations to the real infection numbers that occurred during the beginning of 2021 in Ireland, when the vaccination process was starting across the country. We used three different sets of transmission numbers, each of them representative of different levels of restrictions. Our method was successful in bringing the infection numbers under control within a short amount of time and with a need for less resources than what was used. This result is particularly important as it stresses the importance of correct allocation of the available resources over the quantity of those resources. This means that even countries with limited access to vaccines are able to effectively reduce the number of infections.

When working with a compartmental model, an important question that may arise is in regards to the model's identifiability. This topic has been studied in recent literature [77, 78] and the results suggest that different model structures have different levels of identifiability. Specifically, parameters associated with non-observable compartments seem to be the most difficult to identify. This difficulty does not take away from the merit of our approach, since compartmental models allow for a deeper and more intuitive analysis of the dynamics of infectious diseases, especially one as novel as COVID-19.

While our approach led us to a lot of interesting conclusions, there are certain drawbacks. Our model assumes that all of the parameters remain constant for the duration of the simulations which is not a realistic assumption. For example the transmission rates, which were the source of the vastly different results in our two simulations, do not remain constant for long periods of time. The dynamics of infectious diseases are highly sensitive to transmission rates. While we explored two relatively extreme cases, we haven't explored the case where the transmission rates are variable, which would be a very interesting extension. Furthermore, our model is a deterministic model which means that there is no accounting for any uncertainty. However, it would be relatively simple to propagate uncertainty through the model by repeatedly running it with parameter values sourced from an appropriate probability distribution. In addition, we made the assumption that the available vaccines are bounded per age group instead of them being bounded as a total. However, it seems that bounding the available vaccines in their totality is closer to what occurred in reality. That would result in a different search space for the minimisation, or in other words, different constraints for the objective functional, which would translate to an extra term in the Hamiltonian function and consequently one additional adjoint equation.

In order to avoid the drawbacks described above, a stochastic model would have to be introduced or a model with variable parameters. A study on a time-varying stochastic SEIR model for the control of the Ebola virus can be found in [79]. Exploring different transmission rates and their influence on the system dynamics is an important extension to our work. This can be done either through sensitivity analysis, by producing simulations with a wide range of transmission rates, or introducing time-varying transmission rates. In addition, as we mentioned earlier, it is interesting to explore alternative expressions for the objective functional to be minimised. For example, different weights to the control functions for each age group can be introduced, modelling the cost of bringing the vaccine to different populations, or weight factors relating to the mortality rates of each group, to the numbers of infectious individuals. Moreover, the introduction of more compartments could enrich the model further, for instance a compartment for the hospitalised individuals or one for the deceased. In the last case, it is possible to define the objective functional so as to minimise the deaths as opposed to the infections. In our work, we focused on minimising infections so as to not cause a strain on the healthcare system, however, minimising deaths is an objective very crucial for life and is certainly important and worth exploring. Finally, notable extensions would be the introduction of more age groups and high risk groups, as well as multiple vaccines and their effectiveness. We leave these topics as future research topics which we pursue elsewhere.

A.7 Table of notations

Model states	**Symbol**
Susceptible not yet vaccinated (o65 and y65)	S_O, S_Y
Received vaccine, waiting for it to take effect (o65 and y65)	V_O, V_Y
Received vaccine but was not effective (o65 and y65)	N_O, N_Y
Susceptible, refusing or unable to receive vaccine (o65 and y65)	U_O, U_Y
Exposed (o65 and y65)	E_O, E_Y
Infectious (o65 and y65)	I_O, I_Y
Recovered (o65 and y65)	R_O, R_Y
Protected from vaccine (o65 and y65)	P_O, P_Y
Total number of people in age group (o65 and y65)	T_O, T_Y
Model Parameters	**Symbol**
Rate at which an o65 person infects an o65 person	β_{OO}
Average number of o65 people infected by an o65 person	$R0_{OO}$
Rate at which a y65 person infects an o65 person	β_{YO}
Average number of o65 people infected by a y65 person	$R0_{YO}$
Rate at which an o65 person infects a y65 person	β_{OY}
Average number of y65 people infected by an o65 person	$R0_{OY}$
Rate at which a y65 person infects a y65 person	β_{YY}
Average number of y65 people infected by a y65 person	$R0_{YY}$
Rate at which exposed becomes infected	γ_E
Rate at which infected becomes recovered	γ_I
Rate at which vaccinated becomes protected	γ_V
Vaccine effectiveness	α_V
Percentage of over 65s refusing the vaccine	r_O
Percentage of under 65s refusing the vaccine	r_Y
Control functions	**Symbol**
Percentage of over 65s to get vaccinated at time t	$u_O(t)$
Percentage of under 65s to get vaccinated at time t	$u_Y(t)$
Optimisation Functions and Parameters	**Symbol**
Hamiltonian function	H
Cost function	$\mathcal{F}$
Age specific weight constants	W_O, W_Y

A.8 Existence of an optimal solution

The state equations (A.2) can be written in compact form as

$$g(t,x,u) := \frac{dx(t)}{dt} \tag{A.28}$$

where x denotes the variable vector $x = (\ S_O, V_O, N_O, U_O, E_O, I_O, R_O, P_O, S_Y, V_Y, N_Y, U_Y, E_Y, I_Y, R_Y, P_Y\ )$. Firstly, we assume g satisfies the following:

(a) $|g(t,x,u)| \leq C_1(1+|x|+|u|)$

(b) $|g(t,x,u) - g(t,x',u)| \leq C_2|x-x'|(1+|u|)$

for some positive constants C_1, C_2 and for all t, x, x' and $u \in \Omega$. As stated in [48], when g is of the class C^1, meaning that it is differentiable with continuous first order partial derivatives, then the above conditions are satisfied by applying suitable bounds on the partial derivatives of g. It is possible to show that such bounds can be applied in the case of our epidemic model, from the boundedness of the control and state variables ensuring that Assumptions (a)-(b) are true.

Consider a performance index of the form:

$$\mathcal{F}(u(t)) = \int_0^T f(t,x(t),u(t))dt \tag{A.29}$$

The following theorem, provides the sufficient conditions for the existence of a solution to the optimal control problem.

Theorem 3 *[48] Suppose that assumptions a-b hold, that f is continuous and moreover that:*

(1) The set of solutions to the system equations (A.2) with corresponding control functions in Ω is nonempty.

(2) Ω is convex.

(3) f is convex on Ω and g can be written as $g(t,x,u) = a(t,x) + b(t,x)u$.

Then there exists an optimal control $u^*(t)$ with corresponding optimal trajectory $x^*(t)$ minimizing $\mathcal{F}(u(t))$.

We now prove the conditions of the theorem are satisfied for our model. As noted earlier, conditions (a)-(b) are satisfied. The performance index integrand is

$$f(t,x(t),u(t)) = I_O(t) + I_Y(t) + \frac{W_O}{2}u_O^2(t) + \frac{W_Y}{2}u_Y^2(t) \tag{A.30}$$

which is continuous, and $\Omega = \{u(t) \in L^2(O,T)^2 \| a \leq u_O(t), u_Y(t) \leq b, t \in [0,T]\}$. We now examine conditions 1-3.

(1) From the *Picard-Lindelöf* theorem [80], if the system equations are continuous and Lipschitz and the solutions to the system equations are bounded, then there is a unique solution corresponding to every admissible control in Ω. It is trivial to verify the continuity of the state equations A.2. Additionally, every element of the variable vector $x(t)$ is bounded by the corresponding age-group totals T_O, T_Y. In other words, $x(t) \in [O, \max\{T_O, T_Y\}]^{16}$ which is a compact and convex set. Finally, the Lipschitz property can be verified with the use of the following lemma from [81].

Lemma 2 *If the vector function $\mathbf{X}(\mathbf{x}, t)$ is of class C^1 in a compact convex domain D, then it satisfies the Lipschitz condition there.*

(2) From the *Heine-Borel* theorem, it follows that Ω is convex because as a closed and bounded set in $\mathbb{R}^2$ [82].

(3) The state equations are linearly dependent on the control functions $u_O(t)$ and $u_Y(t)$ so they can be written in the form $g(t,x,u) = a(t,x) + b(t,x)u$. The convexity of the integrand f in the objective functional, follows from the fact that it is quadratic in the control functions.

A.9 Summary of Simulation Code

The R code associated with this work can be found in on Github: `https://github.com/elenizavrakli/Optimal-Age-Specific-Vaccination-Control`. We provide two scripts in order to simulate the evolution of the pandemic with and without intervention through vaccination. In both scripts, we use two packages: *deSolve*, which provides methods for solving differential equations, and *tidyverse* which offers an array of tools for data manipulation, visualisation etc.

A.9.1 Baseline situation without vaccination

The script is organised in 6 blocks as follows:

(1) **Initialise system states and parameters.** Set initial values for the susceptible, exposed, infectious and recovered population in each age group. Additionally, introduce the refusal rates for the vaccine, the mean holding times for the exposed and infectious states and the R0 numbers which represent the dynamics of transmission amongst age groups. Finally, we set the time frame for the experiment.

(2) **Creating state and parameter vectors.** Bring all states together in a named vector using the initial values defined previously, while setting the initial values for all compartments associated with the vaccination equal to 0. Similarly, we define the parameter values used in the model according to the initialisation in the previous step.

(3) **Time sequence and control.** Define the time sequence for which to solve the system of ODEs and initialise the control functions to 0. In this case, they will never get updated, since we are exploring the baseline situation where no vaccination is applied.

(4) **Define the system of ODEs.** Create a function containing the system of ODEs, as defined in section A.2.

(5) **Solve the system of ODEs.** Use the ODE solver provided by the *deSolve* package to obtain the state trajectories. We also obtain the percentage of recovered people for each age group as well as the total percentage of the population that got infected.

(6) **Plotting the trajectories.** Plot the trajectories of the susceptible, infected and recovered compartments for both age groups. We also save some of the state vectors to produce comparisons with the case where the optimal vaccination strategy is implemented.

A.9.2 Case where optimal vaccination strategy in place

The script is organised similarly to the previous case. It is organised in 9 blocks as follows:

(1) **Initialise system states and parameters.** Same as before, set initial values states, parameters and the time frame for the experiment. There are some additional parameters associated with the vaccination, namely the vaccine effectiveness and mean holding time for the vaccine. Additionally, we define the upper bound for the control functions and the error threshold for the optimisation convergence.

(2) **Creating state and parameter vectors.** Same as the previous case. The states associated with the vaccine are again initialised to 0, because we assume the vaccination is only getting started now.

(3) **Time sequence and control.** Same as the baseline case. We need two time sequence, one increasing and one decreasing, indicating the solution of the state system forwards in time and the solution of the adjoint system backwards in time respectively. The control functions will get updated after every solution of the state and adjoint systems.

(4) **Define the state ODEs.**

(5) **Solve the state ODEs forwards in time.**

(6) **Define the adjoint ODEs.** Create a function containing the system of adjoint ODEs as obtained through optimal control in section A.3.

(7) **Solve the adjoint ODEs backwards in time.**

(8) **Repeat process until convergence.** Design a while loop which includes the updates of the control function after each solution of the state and adjoint system. The loop is exited when two consecutive state trajectories are obtained to be closer than the predetermined error threshold. We obtain the percentage of recovered people for each age group as well, the total percentage of the population that got infected, the percentage of vaccinated people for each age group and the total percentage of the population protected from the vaccine.

(9) **Plotting the trajectories.** Plot the trajectories of the susceptible, infected and recovered compartments for both age groups. We also plot the control function for each age group and then provide comparison plots for the infectious individuals with and without the optimal vaccination strategy in place, for both age groups.

Our code is designed to simulate the situation in the Republic of Ireland, meaning that we used initial values for the states and parameters, according to estimates of these numbers found in [64] and [65]. The code is written so that any values can be used, in order to study the effect that they have on the infection dynamics. The versatility of our code allows us to apply this approach to any country by making changes in the two first blocks in both of our R scripts. We can change the values associated with the population, the transmission rates, mean holding times, refusal rate etc.

Bibliography

[1] David S. Hui, Esam I Azhar, et al. The continuing 2019-nCoV epidemic threat of novel coronaviruses to global health ; The latest 2019 novel coronavirus outbreak in Wuhan, China. *International Journal of Infectious Diseases*, 91:264–266, 2021/03/11 2020.

[2] Pneumonia of unknown cause - China. WHO, `https://www.who.int/csr/don/05-january-2020-pneumonia-of-unkown-cause-china/en/`, 2020. Accessed: 2021-03-11.

[3] Na Zhu, Dingyu Zhang, et al. A Novel Coronavirus from Patients with Pneumonia in China, 2019. *New England Journal of Medicine*, 382(8):727–733, 2020. PMID: 31978945.

[4] Hongzhou Lu, Charles W. Stratton, and Yi-Wei Tang. Outbreak of pneumonia of unknown etiology in wuhan, china: The mystery and the miracle. *Journal of Medical Virology*, 92(4):401–402, 2020.

[5] Jasper Fuk-Woo Chan, Shuofeng Yuan, et al. A familial cluster of pneumonia associated with the 2019 novel coronavirus indicating person-to-person transmission: a study of a family cluster. *The Lancet*, 395(10223):514–523, 2021/03/11 2020.

[6] Fan Wu, Su Zhao, Bin Yu, et al. A new coronavirus associated with human respiratory disease in china. *Nature*, 579(7798):265–269, 2020.

[7] Qun Li, Xuhua Guan, Peng Wu, et al. Early transmission dynamics in wuhan, china, of novel coronavirus–infected pneumonia. *New England Journal of Medicine*, 382(13):1199–1207, 2020. PMID: 31995857.

[8] Novel coronavirus - China. WHO, `https://www.who.int/csr/don/12-january-2020-novel-coronavirus-china/en/`, 2020. Accessed: 2021-03-11.

[9] Naming the coronavirus disease (COVID-19) and the virus that causes it. WHO, `https://www.who.int/emergencies/diseases/novel-coronavirus-2019/`

technical-guidance/naming-the-coronavirus-disease-(covid-2019)-and-the-virus-that-causes-it, 2020. Accessed: 2021-03-11.

[10] Maged N. Kamel Boulos and Estella M. Geraghty. Geographical tracking and mapping of coronavirus disease covid-19/severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) epidemic and associated events around the world: how 21st century GIS technologies are supporting the global fight against outbreaks and epidemics. *International Journal of Health Geographics*, 19(1):8, 2020.

[11] Coronavirus: Man treated in Dublin hospital as officials trace contacts. The Irish Times, https://www.irishtimes.com/news/health/coronavirus-man-treated-in-dublin-hospital-as-officials-trace-contacts-1.4189306, 2020. Accessed: 2021-03-12.

[12] Coronavirus outbreak: Who raises risk warning to very high. The Irish Times, https://www.irishtimes.com/news/health/coronavirus-outbreak-who-raises-risk-warning-to-very-high-1.4188313, 2020. Accessed: 2021-03-12.

[13] Liangrong Peng, Wuyue Yang, Dongyan Zhang, Changjing Zhuge, and Liu Hong. Epidemic analysis of COVID-19 in China by dynamical modeling. *medRxiv*, 2020.

[14] Qun Li, Xuhua Guan, Peng Wu, et al. Early Transmission Dynamics in Wuhan, China, of Novel Coronavirus–Infected Pneumonia. *New England Journal of Medicine*, 382(13):1199–1207, 2020. PMID: 31995857.

[15] Shi Zhao, Qianyin Lin, Jinjun Ran, et al. Preliminary estimation of the basic reproduction number of novel coronavirus (2019-ncov) in china, from 2019 to 2020: A data-driven analysis in the early phase of the outbreak. *International journal of infectious diseases*, 92:214–217, 2020.

[16] Joseph T Wu, Kathy Leung, and Gabriel M Leung. Nowcasting and forecasting the potential domestic and international spread of the 2019-ncov outbreak originating in wuhan, china: a modelling study. *The Lancet*, 395(10225):689–697, 2020.

[17] Tao Liu, Jianxiong Hu, Min Kang, et al. Transmission dynamics of 2019 novel coronavirus (2019-ncov). *bioRxiv*, 2020.

[18] Mingwang Shen, Zhihang Peng, Yanni Xiao, and Lei Zhang. Modelling the epidemic trend of the 2019 novel coronavirus outbreak in China. *bioRxiv*, 2020.

[19] Adam J Kucharski, Timothy W Russell, Charlie Diamond, et al. Early dynamics of transmission and control of COVID-19: a mathematical modelling study. *The lancet infectious diseases*, 20(5):553–558, 2020.

[20] Yaqing Fang, Yiting Nie, and Marshare Penny. Transmission dynamics of the COVID-19 outbreak and effectiveness of government interventions: A data-driven analysis. *Journal of Medical Virology*, 92(6):645–659, 2020.

[21] Alberto Godio, Francesca Pace, and Andrea Vergnano. SEIR modeling of the Italian epidemic of SARS-CoV-2 using computational swarm intelligence. *International Journal of Environmental Research and Public Health*, 17(10):3535, 2020.

[22] Salisu M. Garba, Jean M.-S. Lubuma, and Berge Tsanou. Modeling the transmission dynamics of the covid-19 pandemic in south africa. *Mathematical Biosciences*, 328:108441, 2020.

[23] Damjan Manevski, Nina Ružić Gorenjec, Nataša Kejžar, and Rok Blagus. Modeling covid-19 pandemic using bayesian analysis with application to slovene data. *Mathematical Biosciences*, 329:108466, 2020.

[24] Lennon Ó Náraigh and Áine Byrne. Piecewise-constant optimal control strategies for controlling the outbreak of covid-19 in the irish population. *Mathematical Biosciences*, 330:108496, 2020.

[25] Calistus N. Ngonghala, Enahoro Iboi, Steffen Eikenberry, Matthew Scotch, Chandini Raina MacIntyre, Matthew H. Bonds, and Abba B. Gumel. Mathematical assessment of the impact of non-pharmaceutical interventions on curtailing the 2019 novel coronavirus. *Mathematical Biosciences*, 325:108364, 2020.

[26] Pfizer and BioNTech announce vaccine candidate against COVID-19 achieved success in first interim analysis from phase 3 study. Pfizer, `https://www.pfizer.com/news/press-release/press-release-detail/pfizer-and-biontech-announce-vaccine-candidate-against`, 2020. Accessed: 2021-03-16.

[27] Fernando P. Polack, Stephen J. Thomas, Nicholas Kitchin, et al. Safety and Efficacy of the BNT162b2 mRNA Covid-19 Vaccine. *New England Journal of Medicine*, 383(27):2603–2615, 2020. PMID: 33301246.

[28] Lindsey R. Baden, Hana M. El Sahly, Brandon Essink, et al. Efficacy and safety of the mrna-1273 sars-cov-2 vaccine. *New England Journal of Medicine*, 384(5):403–416, 2020. PMID: 33378609.

[29] Merryn Voysey, Sue Ann Costa Clemens, Shabir A Madhi, et al. Safety and efficacy of the ChAdox1 nCov-19 vaccine (AZD1222) against SARS-CoV-2: an interim analysis of four randomised controlled trials in Brazil, South Africa, and the UK. *The Lancet*, 397(10269):99–111, 2020.

[30] Comirnaty. European Medicines Agency, https://www.ema.europa.eu/en/medicines/human/EPAR/comirnaty, 2020. Accessed: 2021-03-16.

[31] COVID-19 Vaccine Moderna. European Medicines Agency, https://www.ema.europa.eu/en/medicines/human/EPAR/covid-19-vaccine-moderna#overview-section, 2020. Accessed: 2021-03-16.

[32] COVID-19 Vaccine AstraZeneca. European Medicines Agency, https://www.ema.europa.eu/en/medicines/human/EPAR/covid-19-vaccine-astrazeneca, 2020. Accessed: 2021-03-16.

[33] COVID-19 Vaccine Janssen. European Medicines Agency, https://www.ema.europa.eu/en/medicines/human/EPAR/covid-19-vaccine-janssen, 2020. Accessed: 2021-04-13.

[34] Safe COVID-19 vaccines for Europeans. European Commission, https://commission.europa.eu/strategy-and-policy/coronavirus-response/safe-covid-19-vaccines-europeans_enn, 2023. Accessed: 2023-07-13.

[35] Coronavirus (COVID-19) Vaccinations. Our World in Data, https://ourworldindata.org/covid-vaccinations, 2023. Accessed: 2023-07-17.

[36] Manuel Adrian Acuña-Zegarra, Saúl Díaz-Infante, David Baca-Carrasco, and Daniel Olmos Liceaga. Covid-19 optimal vaccination policies: A modeling study on efficacy, natural and vaccine-induced immunity responses. *Mathematical Biosciences*, page 108614, 2021.

[37] Isabelle J. Rao and Margaret L. Brandeau. Optimal allocation of limited vaccine to control an infectious disease: Simple analytical conditions. *Mathematical Biosciences*, 337:108621, 2021.

[38] Suzanne Lenhart and John T Workman. *Optimal control applied to biological models*. CRC press, 2007.

[39] Norman TJ Bailey et al. *The mathematical theory of infectious diseases and its applications*. Charles Griffin & Company Ltd, 5a Crendon Street, High Wycombe, Bucks HP13 6LE., 1975.

[40] Gerardo Chowell, Stefano M Bertozzi, M Arantxa Colchero, et al. Severe respiratory disease concurrent with the circulation of h1n1 influenza. *New England journal of medicine*, 361(7):674–679, 2009.

[41] Christopher S Bowman, Julien Arino, and Seyed M Moghadas. Evaluation of vaccination strategies during pandemic outbreaks. *Mathematical Biosciences & Engineering*, 8(1):113, 2011.

[42] Rajan Patel, Ira M Longini Jr, and M Elizabeth Halloran. Finding optimal vaccination strategies for pandemic influenza using genetic algorithms. *Journal of theoretical biology*, 234(2):201–212, 2005.

[43] Neil M. Ferguson, Derek A. T. Cummings, Christophe Fraser, et al. Strategies for mitigating an influenza pandemic. *Nature*, 442(7101):448–452, 2006.

[44] Sunmi Lee, Michael Golinski, and Gerardo Chowell. Modeling Optimal Age-Specific Vaccination Strategies Against Pandemic Influenza. *Bulletin of Mathematical Biology*, 2012.

[45] Leonardo Souto Ferreira, Gabriel Berg de Almeida, Marcelo Eduardo Borges, Lorena Mendes Simon, Silas Poloni, Ângela Maria Bagattini, Michelle Quarti Machado da Rosa, José Alexandre Felizola Diniz Filho, Ricardo de Souza Kuchenbecker, Suzi Alves Camey, et al. Modelling optimal vaccination strategies against covid-19 in a context of gamma variant predominance in brazil. *Vaccine*, 40(46):6616–6624, 2022.

[46] Gilberto González-Parra, Myladis R Cogollo, and Abraham J Arenas. Mathematical modeling to study optimal allocation of vaccines against covid-19 using an age-structured population. *Axioms*, 11(3):109, 2022.

[47] Sam Moore, Edward M Hill, Louise Dyson, Michael J Tildesley, and Matt J Keeling. Modelling optimal vaccination strategy for sars-cov-2 in the uk. *PLoS computational biology*, 17(5):e1008849, 2021.

[48] Wendell Fleming and Raymond Rishel. *Deterministic and Stochastic Optimal Control*. Springer-Verlag New York, 1975.

[49] Katsuhiko Ogata et al. *Modern control engineering*, volume 5. Prentice hall Upper Saddle River, NJ, 2010.

[50] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004.

[51] V.G. Boltyanski, R.V. Gamkrelidze, E.F. Mishchenko, and L.S. Pontryagin. The maximum principle in the theory of optimal processes of control. *IFAC Proceedings Volumes*, 1(1):464 – 469, 1960. 1st International IFAC Congress on Automatic and Remote Control, Moscow, USSR, 1960.

[52] Eunha Shim. Optimal allocation of the limited covid-19 vaccine supply in south korea. *Journal of clinical medicine*, 10(4):591, 2021.

[53] Cong Yang, Yali Yang, and Yang Li. Assessing vaccination priorities for different ages and age-specific vaccination strategies of covid-19 using an seir modelling approach. *Plos one*, 16(12):e0261236, 2021.

[54] Bishal Chhetri, DKK Vamsi, D Bhanu Prakash, S Balasubramanian, and Carani B Sanjeevi. Age structured mathematical modeling studies on covid-19 with respect to combined vaccination and medical treatment strategies. *Computational and Mathematical Biophysics*, 10(1):281–303, 2022.

[55] Tingting Li and Youming Guo. Modeling and optimal control of mutated covid-19 (delta strain) with imperfect vaccination. *Chaos, Solitons & Fractals*, 156:111825, 2022.

[56] Zhong-Hua Shen, Yu-Ming Chu, Muhammad Altaf Khan, Shabbir Muhammad, Omar A Al-Hartomy, and M Higazy. Mathematical modeling and optimal control of the covid-19 dynamics. *Results in Physics*, 31:105028, 2021.

[57] Arshad Alam Khan, Saif Ullah, and Rohul Amin. Optimal control analysis of covid-19 vaccine epidemic model: a case study. *The European Physical Journal Plus*, 137(1):1–25, 2022.

[58] Nicolas Bacaër. *A short history of mathematical population dynamics.* Springer Science & Business Media, 2011.

[59] A. G. M'kendrick. Applications of Mathematics to Medical Problems. *Proceedings of the Edinburgh Mathematical Society*, 1925.

[60] Roy M Anderson and Robert M May. Population biology of infectious diseases: Part I. *Nature*, 280(5721):361–367, 1979.

[61] Fu Teck Liew, Palash Ghosh, and Bibhas Chakraborty. Accounting for the role of asymptomatic patients in understanding the dynamics of the covid-19 pandemic: a case study from singapore. *Epidemiologic Methods*, 11(s1):20210031, 2022.

[62] Marwan Al-Raeei. Numerical simulation of the force of infection and the typical times of sars-cov-2 disease for different location countries. *Modeling Earth Systems and Environment*, 8(1):1443–1448, 2022.

[63] Nicola Guglielmi, Elisa Iacomini, and Alex Viguerie. Identification of time delays in covid-19 data. *Epidemiologic Methods*, 12(1):20220117, 2023.

[64] Ireland's COVID-19 Data Hub. Government of Ireland, `https://covid19ireland-geohive.hub.arcgis.com/pages/detailed-profile-of-cases`, 2020. Accessed: 2021-03-22.

[65] James P Gleeson, Thomas Brendan Murphy, Joseph D O'Brien, Nial Friel, Norma Bargary, and David JP O'Sullivan. Calibrating covid-19 susceptible-exposed-infected-removed models with time-varying effective contact rates. *Philosophical Transactions of the Royal Society A*, 380(2214):20210120, 2022.

[66] Amber L Mueller, Maeve S McNamara, and David A Sinclair. Why does COVID-19 disproportionately affect older people? *Aging (Albany NY)*, 12(10):9959, 2020.

[67] Lucien W Neustadt. *Optimization: a theory of necessary conditions.* Princeton University Press, 1976.

[68] Horst Behncke. Optimal control of deterministic epidemics. *Optimal Control Applications and Methods*, 21(6):269–285, 2000.

[69] Oluwaseun Sharomi and Tufail Malik. Optimal control in epidemiology. *Annals of Operations Research*, 251(1-2):55–71, 2017.

[70] Denise Kirschner, Suzanne Lenhart, and Steve Serbin. Optimal control of the chemotherapy of HIV. *Journal of mathematical biology*, 35(7):775–792, 1997.

[71] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes.* Routledge, 2018.

[72] Ernest Bruce Lee and Lawrence Markus. *Foundations of optimal control theory.* Robert E. Kreiger Publishing Co., Malabar, Florida, 1967.

[73] John Charles Butcher and Nicolette Goodwin. *Numerical methods for ordinary differential equations*, volume 2. Wiley Online Library, 2008.

[74] Holly Gaff and Elsa Schaefer. Optimal control applied to vaccination and treatment strategies for various epidemiological models. *Mathematical biosciences & engineering*, 6(3):469, 2009.

[75] D Cereda, M Tirani, F Rovida, et al. The early phase of the COVID-19 outbreak in Lombardy, Italy, 2020.

[76] Just 6 % of people will refuse COVID-19 vaccine, says Ipsos MRBI tracker survey for IPHA. Irish Pharmaceutical Healthcare Association, `https://www.ipha.ie/just-6-of-people-will-refuse-covid-19-vaccine-says-ipsos-mrbi-tracker-survey-for-ipha/`, 2020. Accessed: 2021-04-15.

[77] Gemma Massonis, Julio R Banga, and Alejandro F Villaverde. Structural identifiability and observability of compartmental models of the covid-19 pandemic. *Annual reviews in control*, 51:441–459, 2021.

[78] Luca Gallo, Mattia Frasca, Vito Latora, and Giovanni Russo. Lack of practical identifiability may hamper reliable predictions in covid-19 epidemic models. *Science advances*, 8(3):eabg5234, 2022.

[79] Phenyo E Lekone and Bärbel F Finkenstädt. Statistical inference in a stochastic epidemic seir model with control intervention: Ebola as a case study. *Biometrics*, 62(4):1170–1177, 2006.

[80] Earl A Coddington and Norman Levinson. *Theory of ordinary differential equations*. Tata McGraw-Hill Education, 1955.

[81] Garrett Birkhoff and Gian-Carlo Rota. *Ordinary differential equations*. John Wiley & Sons, 1978.

[82] Walter Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1976.