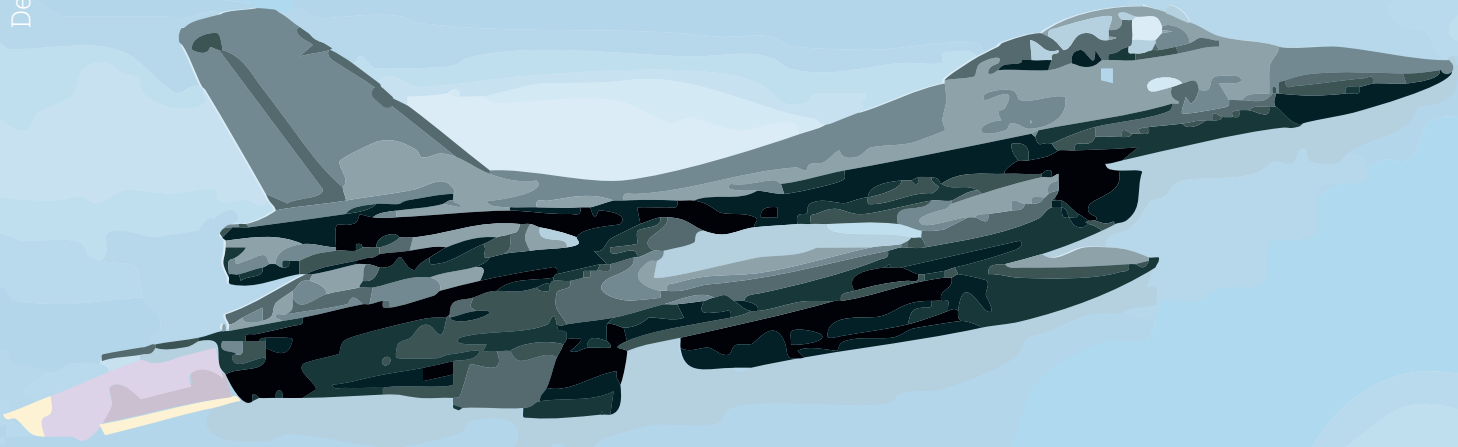
Robust Reinforcement Learning for Flight Control

Model-free fault-tolerant flight control

José Bernardo
Pinto de Moura Leite da Cunha

Delft University of Technology



TU Delft

Robust Reinforcement Learning for Flight Control

Model-free fault-tolerant flight control

by

José Bernardo
Pinto de Moura Leite da Cunha

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on June 30th, 2026 at 10:30 AM.

Thesis committee:

Chair:	Dr. Marilena Pavel
Supervisors:	Dr. ir. Erik-Jan van Kampen
	Dr. Eng. Spilios Theodoulis
External examiner:	Dr. ir. Erwin Mooij
Place:	Faculty of Aerospace Engineering, TU Delft
Project duration:	September 2025 - June 2026
Student number:	5216087

Cover image: illustration of General Dynamics F-16 fighter aircraft by Mayline Goëb, 2026

TU Delft

Preface

This thesis project marks the end of a nearly year-long project delving into robust control and reinforcement learning, but also as the culmination of my six year journey as a student at TU Delft. Having gone through the challenge of this thesis, but also that which is studying Aerospace Engineering at TU Delft, I look back with great pleasure and pride at the work I have done. However, I have not gone through this time alone, and so I would be amiss not to express my gratitude to those who have helped me complete this goal.

I could not have completed this thesis without the guidance and support from my supervisors, Dr. ir. Erik-Jan van Kampen and Dr. Eng. Spilios Theodoulis, to whom I must thank for the opportunity to take on this topic, but also for your expertise and feedback throughout this project.

I would also like to thank my family: my parents Ana and Luís, my brother António, my aunt Carla and my cousin Francisca, and my grandmother Maria de Lourdes; for their unconditional support and readiness to help me in whichever way I needed, and for continuously pushing me to become better. I could not have done any of this without you! Also to my friends here in Delft, I thank you for your companionship over the years, having made my time here so much richer.

Lastly, I am deeply grateful to my incredible girlfriend Mayline, who gave me constant motivation and encouragement throughout this project, as well as your patience with my explanations of my thesis. Thank you for all your support and love!

It is my hope that this project can serve as a useful stepping stone for others wishing to explore the combination of robust control and reinforcement learning, and as such, I leave the code for this project available to be used as seen fit, if it is useful[1].

José Cunha
Delft, June 2026

[1] github.com/RobustReinforcementLearning

Contents

Nomenclature

2-ZSG Two Player Zero-Sum Game

ADP Approximate Dynamic Programming

ARE Algebraic Ricatti Equation

CT Continuous Time

DHP Dual Heuristic Dynamic Programming

DT Discrete Time

HJI Hamilton-Jacobi-Isaacs

IQC Integral Quadratic Constraint

IRL Integral Reinforcement Learning

ISC Initial Stabilising Controller

LFT Linear Fractional Transformation

LMI Linear Matrix Inequalities

LQR Linear Quadratic Regulator

LQ Linear Quadratic

LSDP Loop Shaping Design Process

LTI Linear Time Invariant

MDP Markov Decision Process

NN Neural Network

NR Noisy Action Robust

PPO Proximal Policy Optimisation

PR Probabilistic Action Robust

RARL Robust Adversarial Reinforcement Learning

RC Robust Control

RLAC Robust Lyapunov Actor-Critic

RL Reinforcement Learning

RMDP Robust Markov Decision Process

RRL Robust Reinforcement Learning

SAC Soft Actor-Critic

ZSG Zero-Sum Game

1 Introduction

1.1. Background and motivation

The aviation industry has profited from an increasingly positive safety record, with a decrease of global aviation accident rates in 2024 compared to its pre-pandemic levels. Nonetheless, the number of fatal accidents is relatively high, with 2024 recording the highest number in the past six years. In fact, around 40% of fatal accidents in 2024 (coinciding with over 80% of fatalities) were caused by loss of control or system malfunctions/failures (ICAO, 2025). To improve these levels, further automation techniques can be implemented into the cockpit, namely to reduce the number of accidents caused by system malfunction by improving the fault-tolerance of aircraft.

A greater focus on automation has been a novel pursuit of the aviation industry, with unmanned aircraft systems that may not even have a pilot on board or even be remotely controlled but fully autonomous and relying on sensor information to fly. Vehicles such as new urban air mobility aircraft require a very high level of automation, which is safety-critical, to avoid any failures, especially for complex urban environments. In order to fly safely, the flight control system of these unmanned aerial vehicles requires robustness against the aircraft's model uncertainties and external disturbances, efficient fault detection and isolation systems, and be reconfigurable depending on actuator fault occurrence or aircraft damage (Ducard, 2009).

To address the need for robustness against model uncertainties and disturbances, robust control techniques can be used. $\mathcal{H}_\infty$ control is one of the main approaches, wherein a controller is designed to be robust against a series of uncertainties, while complying with stability and performance requirements. These types of controllers have seen widespread use for flight control (Schoon & Theodoulis, 2025), due to its mathematically rigorous guarantees on robustness and stability. Typically, controllers are tuned for linearised plant dynamics at specific operating points, leading to certain regions of the flight envelope that may be out of reach of the controller. Moreover, for typical $\mathcal{H}_\infty$ control methods (e.g. mixed sensitivity, μ-synthesis, loop shaping), a model of the aircraft plant in question is required to synthesise the controller, and accessing such a model is increasingly difficult when dynamics become highly nonlinear or tightly coupled. As these controllers rely on a known model of the system, it is not the best suited for fault-tolerant control, as the uncertainty representation may fail to capture the sudden, large changes in dynamics caused by a fault.

To address this reduced adaptability, adaptive control techniques can be used to adjust to unforeseen changes in plant dynamics. Unlike a robust controller that is designed to operate under worst-case conditions (often sacrificing performance during nominal conditions), adaptive controllers aim to perform an online estimation of the process uncertainty and produce a control input to minimise the deviations from desired behaviour (Lavretsky & Wise, 2024). However, adaptive control methods still require knowledge of the plant dynamics for fault-tolerance, and even sometimes of the failed plant dynamics, which again may be difficult to obtain accurately.

Another control paradigm that incorporates this adaptability to unforeseen circumstances is reinforce-

ment learning (RL). This bio-inspired data-driven control framework is capable of learning control actions (policies) through interactions with the environment. This framework can be used to remove the dependence on a model of the system, being the main setbacks of using a purely robust control approach. Using RL as a control paradigm, fault-tolerance can be achieved in two main ways: robustness in offline training or adaptation of the controller with online training. With carefully trained RL agents, some generalisation power can be attained through offline learning, leading to a control policy that is able to account for faults. On the other hand, part of learning can instead be performed online, where the parameters of the control policy are updated in-flight when there is a change to the (unknown) dynamics model, leading to an adaptive optimal control law (Lewis et al., 2012).

However, applying RL methods directly shows some of the shortcomings of these methods. Due to the difficulty and safety concerns of training RL agents in the real system, simulators are often employed, though the physics in these simulators cannot perfectly match that of the real system. This *sim-to-real* gap often results in the unsuccessful transfer of the learned policy to the real system, if the policy is not explicitly robust to modelling errors (Pinto et al., 2017). This can be somewhat addressed with heuristic methods, such as creating a rich training environment to explore the state and action spaces to improve the RL agent's generalisation power (at the cost of increased training time), though typical RL methods do not provide any guarantees of convergence to an optimal control (as they can be stuck in local optima).

With regards to online adaptation of the RL agent for fault-tolerance, this approach also comes with clear disadvantages. The RL algorithm should be able to maintain stability during the online learning phase, not only to reach an optimal controller after a series of training steps, a type of stability which is often omitted in typical RL methods (Buşoniu et al., 2018). In fact, most RL methods do not present any guarantees of stability, and can even learn policies that destabilise the system when online training is applied (Zhang et al., 2020a).

These shortcomings of both the robust control and reinforcement learning paradigms have led to the following question on how to synergise these two control approaches to achieve fault-tolerant control, namely how to improve the robustness and stability of reinforcement learning algorithms to make it safely adaptable to large sudden variations in the dynamics model (Khargonekar & Dahleh, 2018). For such a task, the approaches from robust reinforcement learning (RRL) are proposed, which merge the adaptability and expressiveness of RL with the stability and theoretical guarantees of robust control theory. The aim is to find an RL algorithm that is both provably robust to environmental disturbances and model uncertainties, but also presents provable stability guarantees during training, while not relying on the system dynamics to learn the control actions. RRL controllers have the potential to increase aircraft autonomy by improving upon the fault-tolerance of typical RL controllers and better ensuring the safety of the aircraft during online training.

1.2. Structure

The report is structured as follows. In Part I, a literature study reviewing the state-of-the-art of ADP for adaptive robust control and robust reinforcement learning (among others) is presented, followed by the research proposal for this Thesis, with the definition of the objective and research questions, along with the proposed planning. Part II provides the main results as part of the scientific article, presenting some foundations of robust control, the ADP methodology used, and its results. Additional results are given in Part III, where in Chapter 5 a sensitivity analysis is performed for the ADP learning algorithms that were used. A comparison with other ADP approaches prevalent in the literature is given in Chapter 6 (acting as validation for the methods used) and Chapter 7. The report is concluded in Part IV, where the final conclusions and reflections of the research objective and questions are presented in Chapter 8, while recommendations for future work are offered in Chapter 9.

Part I

Literature Study
&
Research Proposal

2

Literature Review

This literature review presents an overview of the fundamental principles of reinforcement learning and robust control theory (namely $\mathcal{H}_\infty$ control), and introduces the different concepts of combining robust control techniques with reinforcement learning, serving as the foundation for the methods used in this research. Section 2.1 introduces the concepts of reinforcement learning, while Section 2.2 introduces those of robust control. Following this, the principles of combining these two control paradigms are outlined in Section 2.3, describing the model-free solutions of $\mathcal{H}_\infty$ controllers, RL control with robustness properties, RL with stability guarantees and synergistic (model-based and model-free) methods.

2.1. Reinforcement learning

Reinforcement learning (RL) is one of the three main machine learning paradigms, comprised of a bio-inspired framework to learn complex tasks. RL consists of the interaction between an agent and its environment, and a learning process to train the agent to perform a task. The following subsections detail the important basic concepts of reinforcement learning.

2.1.1. Markov decision process

Through its interaction with the environment, the agent will receive from the environment at time step t its current state (s_t) and a reward (r_t) for being in that state. Given these values, the agent will take an action (a_t) in the environment leading to a new state (s_{t+1}). These discrete series of interactions can be summarised with Figure 2.1.

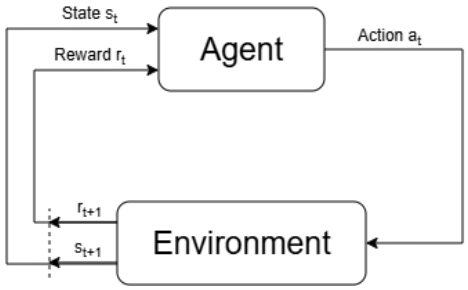


Figure 2.1: *Agent-environment interaction of reinforcement learning (Sutton & Barto, 1998).*

A Markov Decision Process (MDP) is a decision-making framework, which can be used in reinforcement learning to characterise the agent-environment interaction. An MDP is characterised by the Markov property, wherein the future evolution of a state is independent of the past values of the state, and the transition probabilities of reaching a new state are determined from observations (current state and rewards) and the action taken (Sutton & Barto, 1998). The MDP is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the sets of all possible states and actions, and the system dynamics

are defined by the mapping $\mathcal{P}$ from each state and action set to a new action. The reward function R and the discount factor γ define how each action is rewarded, and how much to consider future rewards.

2.1.2. Rewards, policy and value

The RL agent will learn by trying to maximise a reward, given instantaneously to the agent when performing an action at a current state. The reward is designed to aid the agent in achieving a certain task, essentially telling the agent which actions are encouraged and which are not. The reward function is shaped by the designer to capture the intricacies of the environment and to lead the agent to the best possible actions.

Instead of simply maximising the instantaneous reward, the agent will try to maximise the cumulative reward, as for complex tasks, it may be beneficial to forgo a current reward to reach an even higher future reward. This behaviour can be captured by the discount factor $\gamma \in [0, 1)$ in the cumulative reward seen in Equation 2.1. The time horizon T can be defined as the maximum time of operation, which can be infinite.

$$R_t = r_{t+1} + \gamma r_{t+2} + ... + \gamma^T r_{t+T+1} = \sum_{k=0}^{T} \gamma^k r_{t+k+1} = r_{t+1} + \gamma R_{t+1} \tag{2.1}$$

The discount factor is a hyperparameter to be manually tuned, to tell the agent how much to favour or ignore future or long-term rewards (depending on whether it is increased or decreased, respectively). As can be seen in Equation 2.1, the cumulative reward has a recursive nature, as the current value depends on the future cumulative rewards.

To represent the transition probabilities from one state to another, the policy π is introduced, defined in Equation 2.2. This is a law that tells the agent what action to take at each state. The goal of the reinforcement learning agent is to find the optimal policy π^* which maximises the discounted cumulative rewards.

To better express how desirable different states are, the state value function (V^π is introduced, defined in Equation 2.3. The value is the expected cumulative reward given the current state while following policy π. Another representation of the quality of each state and action pair is the Q-function or action-value function (seen in Equation 2.4). This takes into account both the current state and the action taken when following π.

$$\pi(a_t|s_t) = \mathcal{P}(a_t|s_t) \tag{2.2}$$

$$V^\pi(s_t) = \mathbb{E}_\pi[R_t|s_t] = \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k r_{t+k+1} \middle| s_t \right] \tag{2.3}$$

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi[R_t|s_t, a_t] = \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k r_{t+k+1} \middle| s_t, a_t \right] \tag{2.4}$$

A policy is said to be optimal when it leads to higher or equal cumulative rewards than any other policy. When following this optimal policy π^*, the optimal value and Q-functions are reached, yielding Equation 2.5.

$$V^*(s_t) = \max_\pi V^\pi(s_t) \qquad Q^*(s_t, a_t) = \max_\pi Q^\pi(s_t, a_t) \tag{2.5}$$

To learn the optimal policy, a learning process is required to *teach* the agent, from multiple interactions with the environment, what the best actions are to take at each state. Update laws are used to iteratively alter the policy until a solution to the RL problem is found (i.e. learning π^*). However, convergence to

the optimal policy is only guaranteed if every action is executed infinitely often for each state (Sutton & Barto, 1998). Therefore, finding an admissible solution from a limited exploration of the state and action space is important.

With a greedy policy - always taking the best action at each state according to the value function - the known rewards will be maximised. Nonetheless, there is no guarantee that the actions which are known are the best actions, as some might not have been tried. Exploring new actions will incur a risk as the short-term reward is reduced, though the long-term reward may eventually increase. This trade-off between exploring the state and action spaces and exploiting the best known actions is called the *exploration vs. exploitation dilemma*. Stochastic policies can be used to tackle this trade-off, sampling an action from a distribution for a given state $a_t \sim \pi(a_t|s_t)$, and as such, exploration is guaranteed since each action has a non-zero probability of being selected (Moos et al., 2022).

2.1.3. Taxonomy of RL algorithms

As previously mentioned, a learning process is needed to find an admissible policy to the RL problem. This process can be split into many different classifications of learning algorithms (Dong et al., 2020), detailed below. While there is a distinction between model-based (more akin to dynamic programming) and model-free RL algorithms, the methods in the following sections will focus solely on model-free RL (where in some algorithms, model-based methods are required separate to the main RL algorithm).

Offline and online learning

RL algorithms can be categorised depending on when the learning is performed. Online learning allows the RL agent to learn while performing its actions (Sutton & Barto, 1998) - for the flight control case, this would entail learning in-flight. Online RL therefore requires stricter safety considerations, as learning instabilities for critical systems may not be permitted, and also requires better sample-efficiency, as fewer data points may be available for learning (Dong et al., 2020). This perspective can be classified as an adaptive control, where the control policy is being updated while the agent is performing a task.

Offline learning instead occurs prior to the agent being deployed, usually inferring that training of the RL agent be done in a simulation of the environment, yet still in a model-free way, as the agent does not have knowledge of its system dynamics or environment. In essence, the parameters defining the control policy are updated prior to being deployed and left unchanged after deployment. As learning is done in a simulation, safety considerations are not critical, and more time can be given to the agent to learn from interactions with the simulation. However, there is often a mismatch between the simulation and reality (i.e. the *sim-to-real* gap), leading to stability issues when deploying offline trained RL agents to the real system - this can be improved upon with robust reinforcement learning. The definition of online and offline RL can differ in the literature, wherein offline RL can be considered as collecting a large dataset once, and using that to train the agent to then be deployed (Sutton & Barto, 1998). Throughout this research, offline RL refers to training being performed prior to deployment, and the control policy parameters remain unchanged during deployment.

There can also be a combination of offline and online trained agents, where initially a policy is learned offline in the simulation, with long training times and carefully crafted exploration of the state and action spaces, and then the policy can be improved online, leading to some adaptability of the policy model parameters in case it is required (e.g. during an actuator fault).

Discrete and continuous spaces

Reinforcement learning algorithms can be classified according to the size of the state and action spaces of the system/environment where the learning is occurring. These spaces can be either discrete or continuous, and solution methods can differ greatly between them. Algorithms that tackle discrete state or action spaces with low dimension (often called tabular methods) are the basis for other RL algorithms (Sutton & Barto, 1998), even though they are not suitable for the task being addressed in this research, as for high-dimensional state/action spaces, the size of the 'tables' required to store the value functions for these tabular methods increases exponentially.

Algorithms specifically designed to tackle problems with large continuous spaces often rely on function approximation methods to parametrise these state and action spaces, instead of large tables. Usually, this is done in the form of neural networks (NN) to parametrise the value function and/or policy and possibly also the model approximation of the environment.

Value-based and policy-based

Again, RL algorithms can be split into two main categories: value-based methods and policy-based methods. Value-based methods optimise the state-value ($V^{\pi}(s)$) or action-value ($Q^{\pi}(s,a)$) functions, and then using this optimal value is used to derive the optimal policy ($\pi^* \approx \arg\max_{\pi} Q^{\pi}$). These methods lead to better sample efficiency and small variance in the value function, though with the disadvantage that they cannot usually handle continuous action spaces (e.g. such as Deep Q-Network (Mnih et al., 2015)), and the greedy strategy when deriving the policy can result in overestimation.

Policy-based methods instead optimise the policy directly, iteratively updating the policy until the cumulative discounted rewards are maximised. This leads to a more direct and simple parametrisation of the policy (e.g. using neural networks), and unlike value-based methods, are suitable for high-dimensional/continuous action spaces. A common approach are policy gradient methods (e.g. REINFORCE, Williams (1992)), which use the gradient of the loss function with respect to the parameters of the policy to update them.

A combination of value-based and policy-based methods, where both the value function and the policy are directly estimated and optimised, are deemed the actor-critic (AC) methods. The actor uses policy-based methods to learn the policy function, while the critic uses value-based methods to learn the value or Q-function. Although these methods enjoy both the advantages of value- and policy-based methods, such as being applicable to discrete or continuous action spaces, and reducing sampling variance, there are also the disadvantages of overestimation of the value by the critic and a lack of exploration by the actor. However, several state-of-the-art AC methods, such as Proximal Policy Optimisation (PPO) (Schulman et al., 2017) or Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016) aim to address these issues.

On-policy and off-policy

The main difference between on- and off-policy RL algorithms stems from which policy is actually used when the optimal policy is being calculated. On-policy algorithms improve the policy that is used to generate the agent's actions, and use this collected data to evaluate and improve the policy (i.e. the policy that interacts with the environment is the policy which is being improved each step by the RL algorithm). On the other hand, off-policy methods use a different policy to generate the data than the one used to evaluate and improve it (i.e. use a greedy policy when calculating the improvements for the new one). SARSA (Sutton & Barto, 1998) is one of the common classic RL on-policy algorithms, selecting an action based on the current policy, and using the new state to improve the current policy. The off-policy version of this algorithm is Q-learning (Watkins & Dayan, 1992), which instead uses the max operation and greedy policy to select the action, requiring steps to calculate the maximum for which the old policy is used for interaction with the environment instead of the updated action. The difference in their Q-function updates can be seen in Equation 2.6.

$$\begin{aligned} Q(s_t,a_t) &\xleftarrow{\text{SARSA}} Q(s_t,a_t) + \alpha r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)] \\ Q(s_t,a_t) &\xleftarrow{\text{Q-learn}} Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)] \end{aligned} \tag{2.6}$$

Examples of RL algorithms

With the different taxonomies of algorithms differentiated, the various model-free RL algorithms that exist in the literature can be classified, as seen in Figure 2.2 for some of the algorithms of greater interest. As previously mentioned, SARSA and Q-learning are algorithms which are applicable for discrete state and action spaces, limiting their applicability for the high-dimensional landscape of flight control. Deep Q-Networks (DQN) are an extension of Q-learning, using deep neural networks for the Q-function approximation, allowing for continuous state spaces to be used, though only for smaller numbers of discrete actions.

For both continuous state and action spaces, a series of algorithms exist, which usually apply the actor-critic structure to both learn the policy and the value function. DDPG can be seen as an extension of DQN for continuous action spaces, learning a deterministic policy to maximise an NN-approximated Q-function, and applying both a replay buffer (set of previous states and actions) and a target network (time-delayed copy of the value/policy networks) to improve learning stability. Soft Actor-Critic (SAC) (Haarnoja et al., 2019) is also an off-policy algorithm which instead learns a stochastic policy and uses entropy regularisation (trade-off between the expected rewards and the entropy/randomness of the

system). SAC has already been applied to fault-tolerant flight control of fixed-wing aircraft (Dally & Kampen, 2022; Homola et al., 2025). Considering the on-policy algorithms, PPO is one of the state-of-the-art algorithms, using a clamping term in its loss function to be optimised, removing incentives for a new policy to get far from the previous policy, and improving stability. PPO has also been widely used for flight control applications (Zhang et al., 2025a; Wu & Litt, 2023).

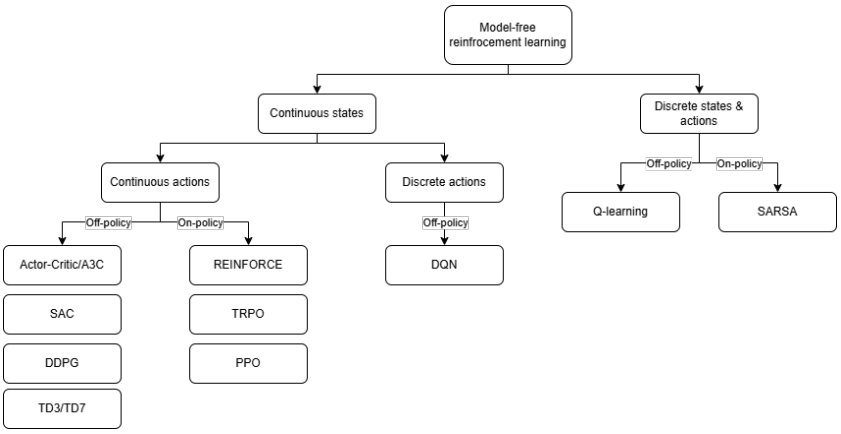


Figure 2.2: *Classification of various model-free reinforcement learning algorithms (AlMahamid & Grolinger, 2022).*

2.2. Robust control

In general, the goal of control theory is to create a controller that stabilises a dynamical system. These systems can be perturbed by environmental disturbances and uncertainties (as uncertainty of the parameters in the system and modelling errors), and as such, robustness to these disturbances and uncertainties is desirable. Robust control aims to address this by providing techniques to design controllers robust to both disturbances and uncertainties (Zhou & Doyle, 1998). Various techniques arise in robust control to solve this problem, with many centred around the $\mathcal{H}_\infty$ control problem, described in the following section.

2.2.1. $\mathcal{H}_\infty$ control framework

The framework for robust control can be summarised by a linear fractional transformation (LFT) framework, where a closed-loop system can be placed in the standard form shown in Figure 2.3. Here, the plant, or P, is the interconnection matrix, K is the controller, and Δ is the set of all possible uncertainties (which can be structured or unstructured). The goal is to find a controller K such that the effect of uncertainty in Δ on system performance is suppressed.

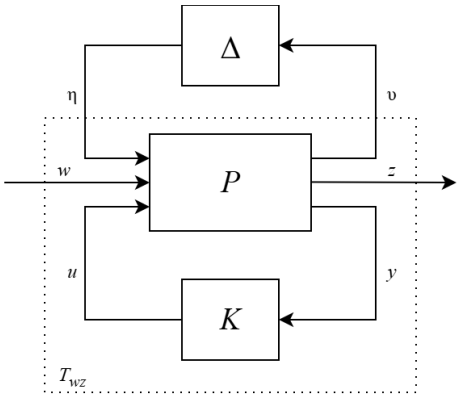


Figure 2.3: *General LFT framework, adapted from (Zhou & Doyle, 1998).*

Considering the lower blocks of the general system given in Figure 2.3, the open-loop system and

feedback controller can be described by Equations 2.7 and 2.8.

$$\begin{bmatrix} z \\ y \end{bmatrix} = P(s) \begin{bmatrix} w \\ u \end{bmatrix} = \begin{bmatrix} P_{zw}(s) & P_{zu}(s) \\ P_{yw}(s) & P_{yu}(s) \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix} \tag{2.7}$$

$$u = K(s)y \tag{2.8}$$

Considering now the system in closed-loop, described by the lower LFT of Figure 2.3, it can be characterised by Equation 2.9, where $T_{zw}(s)$ is the transfer function from the disturbances w to the performance outputs z. Now, the initial goal is to synthesise a controller that nominally stabilises the closed-loop system, while also minimising the $\mathcal{H}_\infty$ norm of the closed-loop transfer function $T_{zw}(s)$ (i.e. ensure an $\mathcal{H}_\infty$ performance level $\gamma > 0$).

$$\begin{aligned} z &= \mathcal{F}_l[P(s), K(s)]w = T_{zw}(s)w \\ T_{zw}(s) &= P_{zw} + P_{zu}K(I - P_{yu}K)^{-1}P_{yw} \end{aligned} \tag{2.9}$$

The $\mathcal{H}_\infty$ norm (denoted by $\|\cdot\|_\infty$) is defined as the peak value of the largest singular value over frequency of the frequency response of the system in question. As seen in Equation 2.10, the $\mathcal{H}_\infty$ norm can have both a frequency-domain and time-domain definition, considering the L_2 norms of the performance output and disturbance signals. By reducing $\|T_{wz}(s)\|_\infty$ via the controller K, the energy of the output signal $z(t)$ is reduced for all disturbances $w(t)$ with bounded energy (i.e. $\|w(t)\|_\infty < 1/\gamma_w$, $\gamma_w > 0$)

$$\begin{aligned} \|T_{wz}(s)\|_\infty &= \max_\omega \bar{\sigma}[T_{wz}(j\omega)] \\ &= \max_\omega \max_{w(\omega)\neq 0} \frac{\|z(\omega)\|_2}{\|w(\omega)\|_2} \\ &= \max_{w(t)\neq 0} \frac{\|z(t)\|_2}{\|w(t)\|_2} \qquad < \gamma \end{aligned} \tag{2.10}$$

Various tools exist to synthesise this nominal $\mathcal{H}_\infty$ controller. Full order controllers can be built via Algebraic Ricatti Equations (ARE) (Zhou & Doyle, 1998) or Linear Matrix Inequalities (LMI) (Gahinet & Apkarian, 1994), leading to unstructured controllers. To create structured controllers instead, non-convex optimisation techniques can be employed to tune the parameters of fixed-structure controllers (Apkarian & Noll, 2017). As will be mentioned in Section 2.3, other techniques such as reinforcement learning can also be used to synthesise an $\mathcal{H}_\infty$ controller.

2.2.2. Robust controller design methods

$\mathcal{H}_\infty$ synthesis will lead to controllers that comply with nominal stability and performance requirements (disturbance rejection or reference tracking, noise attenuation and control signal reduction). In addition to these requirements, a robust controller must also provide robust stability and robust performance to potentially unstructured or structured/parametric uncertainties. Three main methods exist to design a robust $\mathcal{H}_\infty$ controller, namely: $\mathcal{H}_\infty$ mixed sensitivity, μ-synthesis and $\mathcal{H}_\infty$ loop shaping.

Mixed sensitivity introduces frequency-dependent weighting functions which are used to shape the singular values of several closed-loop transfer functions, namely the 'gang-of-six'. High-level objectives such as low-frequency disturbance rejection, high-frequency noise attenuation and reduced control actions and importantly robust stability to unstructured uncertainty, can be converted into a desired shape of these closed-loop transfer functions. A series of weighting filters can be designed to reflect these requirements, and a controller can be created using the aforementioned $\mathcal{H}_\infty$ controller synthesis techniques.

While the robust stability and performance criteria can be posed in terms of the 'gang-of-six', a more unified framework can be used using the structured singular value μ. This value can be used as a robustness analysis tool in the frequency domain (Zhou & Doyle, 1998), in terms of the maximum structured singular value of a particular/structured system. It can be used to create a robust controller

through the μ-synthesis method. This method involves a sequential two-parameter minimisation called D-K iteration, using $\mathcal{H}_\infty$ synthesis for a scaled in the K-step, and updating the scaling function $D(s)$ in the D-step (Skogestad & Postlethwaite, 2005). Even though each step is a convex optimisation problem, when joined, convexity is not guaranteed, and the iterations may converge to a local optimum, so care must be taken.

An additional robust controller design technique is the $\mathcal{H}_\infty$ loop shaping design process (LSDP) (Zhou & Doyle, 1998). For the mixed sensitivity controller, the 'gang-of-six' closed-loop transfer functions are weighted to achieve the nominal and robust stability and performance requirements. However, it can be shown that these requirements can be satisfied by shaping only the open-loop transfer functions within some frequency range, by designing two filters W_1 and W_2. These weights, however, can be particularly difficult and time-consuming to design, especially for non-diagonal weights. With the weights designed, a controller can then be synthesised using the mentioned $\mathcal{H}_\infty$ controller synthesis techniques.

2.2.3. Solving $\mathcal{H}_\infty$ controller as two-player zero-sum game

With a robust $\mathcal{H}_\infty$ controller, the aim is to design it to be robust against environmental disturbances and model uncertainties. This can involve solving a set of AREs to yield the desired controller. As hinted at in subsection 2.2.2, other 'non-classical' solution methods can be employed to synthesise an $\mathcal{H}_\infty$ controller. One class of such methods involves casting the $\mathcal{H}_\infty$ control problem as a two-player zero-sum game (ZSG) (Başar & Bernhard, 2008) or an equivalent min-max problem (Morimoto & Doya, 2005), wherein one player tries to minimise a cost function via the control action and the other player tries to maximise the cost via the disturbance.

To solve an $\mathcal{H}_\infty$ control problem, one must minimise the $\mathcal{H}_\infty$ norm of the closed-loop transfer function T_{wz} in Equation 2.10 with a stabilising controller K, which is equivalent to solving a two-player ZSG. To solve the ZSG, a performance function or value function (similar to the one described in Equation 2.3) can be constructed (which will depend on whether the problem is in continuous or discrete time), and its structure can be seen in Equation 2.11 for CT (Morimoto & Doya, 2005).

$$V(x,u,w) = \int_0^\infty \left(z^\top(t)z(t) - \gamma^2 w^\top(t)w(t)\right) dt \leq 0 \tag{2.11}$$

The goal of solving the ZSG is to find the optimal value function V^*. The condition for this optimal value function is the Hamilton-Jacobi-Isaacs (HJI) equation, which can be used to derive the optimal control policy u^* (essentially encoding the feedback control action attributed to the controller K), and the worst-case disturbance w^* rejected by the controller. As seen in Equation 2.12, this equation involves the system dynamics f (assuming here a CT nonlinear system $\dot{x} = f(x,u,w)$), and the two-player ZSG structure can be seen in the min-max operation.

$$\min_u \max_w \left[z^\top z - \gamma^2 w^\top w + \frac{\partial V^*}{\partial x} f(x,u,w) \right] = 0 \tag{2.12}$$

However, the HJI equation is a nonlinear partial differential equation that is, in general, not possible to solve analytically (Başar & Bernhard, 2008). For a linear system model with quadratic rewards (i.e. $r = z^T z - \gamma^2 w^T w$), the value function can be given also in quadratic form ($V = x^T P x$), where P is the solution to an ARE, and the optimal action and disturbance policies can be derived from the system matrices and P - showing the equivalence of this method with the ARE solution method mentioned in Section 2.2. This solution is, however, not generally the case for nonlinear systems or non-quadratic rewards. Instead of solving it analytically, approximation methods can be employed to solve the HJI equation. Numerical approximation methods have been proposed (Bea, 1998; Ferreira et al., 2008; Huang & Lin, 1995) that require the system dynamics.

This ZSG approach to robust control shall be further explored in the following section, as it comprises the framework behind many of the reinforcement learning approaches to robust control.

2.3. Combining robust control with reinforcement learning

In the previous sections, some details were given on methods from reinforcement learning and robust control theory. Despite the success classic reinforcement learning has had as a control technique (including for flight control), algorithms often struggle with robustness to parameter uncertainty, generalizability to more complex environments and stability during the learning process. On the other hand, robust control has found abundant success in flight control tasks, being able to tackle the issues with robustness to disturbances and uncertainties; however, it suffers from limitations with adaptability, and usually requires the system model to synthesise controllers (which may not always be available).

Combining techniques from reinforcement learning (RL) and robust control (RC) could prove advantageous, using the advantages of each control framework while trying to avoid the drawbacks of each. In this spirit, some different possibilities of the blend of RL and RC are explored in this section, and their possible applications to flight control are investigated.

2.3.1. Approximate dynamic programming to learn $\mathcal{H}_\infty$ controllers

More recently, approximate dynamic programming (ADP) has been used to solve the ZSG. ADP is an application of reinforcement learning for solving feedback control problems, wherein it does not compute the exact dynamic programming solution, but learns an approximation of it. A large variety of ADP methods for solving the $\mathcal{H}_\infty$ control problem related to a two-player ZSG exist in the literature. As these algorithms are essentially an application of reinforcement learning, the different taxonomies of RL algorithms described in Section 2.1 lead to numerous ADP methods with slight differences in solution method, related to model dependence, on/off-policy updates, online/offline learning, continuous/discrete-time systems, and control problem (disturbance rejection, tracking, constrained controls).

For model-based solution methods, Al-Tamimi et al. (2007b) uses an online Dual Heuristic Dynamic Programming (DHP) implementation for linear DT systems, that requires the A, B and E matrices to iteratively calculate the controller (showing convergence to the optimal controller). The reliance on the drift dynamics $f(x)$ was relaxed for CT nonlinear affine systems with the online SPUA algorithm (Wu & Luo, 2012), using neural networks (NN) as function approximators. For both the DT and CT ZSGs, ADP has been successful in solving the $\mathcal{H}_\infty$ control problem, requiring the full or partial system dynamics (Luo et al., 2015; Yasini et al., 2015; Wang et al., 2017; Ngo & Godtliebsen, 2020).

However, in practice, it is often difficult or not possible to have access to the exact model dynamics, and hence a totally model-free implementation is beneficial. Vrabie and Lewis (2011) proposed a model-free ADP to find the solution to the linear quadratic ZSG (which reduced to an ARE, and a similar method is proposed in Li et al. (2014)), whereas Al-Tamimi et al. (2007a) used Q-learning (a type of RL/ADP) for the same DT case. Other authors have also achieved a model-free $\mathcal{H}_\infty$ controller via solving a ZSG for linear DT systems, and have applied it to simple linear systems (Kiumarsi et al., 2017; Liu et al., 2020), output feedback instead of state feedback (Fan et al., 2018) and autonomous driving demand systems (Aalipour & Khani, 2023).

For the nonlinear ZSG, online solutions have been found for the disturbance rejection control problem (Qin et al., 2016) and for the tracking control problem (Modares et al., 2015), with an off-policy ADP algorithm for CT systems. In Modares et al. (2015), a three NN actor-disturber-critic structure was implemented to approximate the control policy, disturbance policy and value function, respectively, with the NN weights trained with a least-squares approach. However, the convergence of the NN approximation is not proved, which is later shown in (Zhu et al., 2017).

Nonetheless, these algorithms (Modares et al., 2015; Qin et al., 2016; Zhu et al., 2017; Li et al., 2014; Perrusquía & Yu, 2021; Lin et al., 2023; Zhang et al., 2018) require either long initial data collection times or an initial stabilising control action, which may be difficult to always attain in a model-free way. In this sense, Vamvoudakis (2017) used an online Q-learning algorithm to relax the dependence on an initial stabilising control action for online learning. Another result uses a modified version of the DDPG algorithm and an actor-disturber-critic architecture (Lee & Lee, 2025), to solve the 2-ZSG, applying it to quadcopter tracking control, without any reliance on initial stabilising controllers, but is trained offline.

The game-based methods described herein lead to $\mathcal{H}_\infty$ controllers that can be learned online or offline from interacting with the environment (i.e. without the need to know the system dynamics), and also show convergence to $\mathcal{H}_\infty$ controllers designed in a model-based fashion. These ADP methods present robustness to external disturbances; however, for robustness to model uncertainties, care must be taken when designing the weights used for controller synthesis.

2.3.2. RL-control with robustness properties

Model-free reinforcement learning (RL) algorithms are usually trained in a simulation, with an agent being able to interact with a simulated environment. However, there may be an inherent difference between the simulator and the real world, as the modelling parameters cannot be perfectly created to reflect the real-world testing environment. Such differences may come from modelling and approximation errors when creating the model, real-world parameters changing in real-time with interactions that remain unmodelled, or adversarial disturbances coming from the environment. Such a mismatch between the modelled and real environment – even if relatively small – can significantly degrade the performance of the trained RL agent. To address this *sim-to-real* gap, robustness to these unmodelled disturbances and uncertainties must be considered (Moos et al., 2022).

Moos et al. (2022) presents a similarity between robust reinforcement learning (RRL) methods to improve the robustness of RL algorithms to uncertainties, modelling errors and disturbances - variability in the transition probabilities. This variability can be achieved in four main approaches, each of them representing a component of the state transition: (i) *Transition robust* - changing the transition probability function directly (altering P); (ii) *Disturbance robust* - applying environmental disturbances to represent changes in the model parameters (altering s_{t+1}); (iii) *Action robust* - manipulating the actions of the agent with an adversary (altering a_t); or (iv) *Observation robust* - manipulating the agent's observations of the state (altering s_t). Using these four approaches, several methods can be found in the literature that aim to improve the robustness of RL, with varying levels of system description (low or high dimensional, DT or CT) and guarantees of stability and convergence.

The Robust Markov Decision Process (RMDP) framework was proposed by Iyengar (2005) and Nilim and El Ghaoui (2005) to improve the robustness of the MDP (described in Section 2.1). While the MDP considers a fixed transition probability function P, such that $s_{t+1} \sim P(s_t, a_t)$, the RMDP formulation extends this to a set of transition probability functions $\mathcal{P}$ (where $P \in \mathcal{P}$, therefore being a transition robust design), therefore considering an uncertainty set of model parameters (assuming that the unknown actual model parameters lie in this set). The objective is to then find an optimal robust policy which maximises the reward under the worst model. Similarly to the treatment given in the previous section, where the $\mathcal{H}_\infty$ control problem can be solved through the lens of a two-player ZSG, the RMDP can be likewise seen as a min-max problem, maximising the reward while minimising the performance of the model. This formulation of the RMDP requires knowledge of the system dynamics and is therefore not suitable for model-free RL. Additionally, this result is only tackled for the low-dimensional/tabular case, and it is unclear how to obtain the mentioned uncertainty sets (Tessler et al., 2019).

Due to the *curse of dimensionality*, the RMDP formulation of Iyengar (2005) becomes intractable for high-dimensional state-spaces. To combat this, linear or nonlinear function approximators can be used, with deep neural networks (nonlinear approximators) already being commonplace for non-robust reinforcement learning. Linear approximators instead maintain better convergence guarantees while also mitigating the *curse of dimensionality*. Panaganti and Kalathil (2021) used linear approximators (instead of deep NNs) for the Q-function in a model-free robust MDP, where an uncertainty set of transition probability functions was created regarding a confidence region around an unknown nominal model, and a least-squares policy iteration algorithm was used to find the optimal robust policy. Wang and Zou (2021) use a similar approach, though with a different definition of the uncertainty set being used to 'robustify' Q-learning and a temporal-difference learning scheme.

Another recent approach involving transition robust RL aims to provide a more general robustness framework (for both discrete and continuous domains), by considering an uncertainty set of candidate transition probabilities (or dynamics models) bounded by an ϵ-Wasserstein ball around some learned reference dynamics (Abdullah et al., 2019). This result uses deep neural networks to parametrise the policies and transition models, leading to agents which can determine robust policies under worst-case

models/disturbances, using Wasserstein distance constraints to bound the search spaces and ensure convergence to feasible transition models, using ϵ as a 'degree of robustness' hyperparameter. This Wasserstein Robust Reinforcement Learning (WR^{2}L) algorithm was compared against other RRL and non-robust RL frameworks in various MuJoCo environments (Brockman et al., 2016), outperforming these methods in the given benchmarks.

Developed in parallel with the definition of the RMDP, which leads to robustness through variability in the transition probability function itself, Morimoto and Doya (2005) considered robustness by applying environmental disturbances to the agent, to represent both input disturbances, and modelling errors (i.e. action robust RL). This approach stems from the $\mathcal{H}_\infty$ control setting, where again the problem is set up as finding the min-max solution of a value function, as in Equations 2.11 and 2.12. Morimoto and Doya (2005) propose a model-free actor-disturber-critic architecture, leading to a min-max game where the actor aims to minimise the cost through the control policy and the disturber aims to maximise it through the disturbance policy. Function approximators (radial basis functions) are used to represent the action and disturbance policies and the value function, respectively. This scheme is applied to some low-dimensional nonlinear systems, leading to robustness to changes in model parameters. It was also shown that this RRL method coincides with the $\mathcal{H}_\infty$ control solution in the linear-quadratic case, as further detailed in (Lewis et al., 2012).

Building on the results of robust reinforcement learning, Pinto et al. (2017) propose a framework for robustness to disturbances and modelling errors by explicitly considering adversarial disturbances to the actor. The MDP is modified to define a 2-ZSG between an actor (π) and a destabilising adversary ($\bar{\pi}$), such that $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \bar{\mathcal{A}}, \mathcal{P}, R, \gamma)$, where $\bar{\mathcal{A}}$ is the set of adversary actions. This Robust Adversarial Reinforcement Learning (RARL) algorithm is designed for the aforementioned Markov game with continuous state and action spaces and uses deep neural networks as nonlinear function approximators, making this scheme amenable to a large variety of applications. RARL performs its learning in an offline fashion, optimising the two agents sequentially, initially learning the actor policy while holding the adversary fixed, and then vice-versa, repeating until convergence is reached. The algorithm was tested experimentally in several MuJoCo domains, achieving robustness against adversarial disturbances and model parameter changes. However, Pinto et al. (2017) do not give theoretical guarantees of stability or convergence (Tessler et al., 2019; Zhang et al., 2020a), limiting its safety and application to online training.

The results of RARL have also been applied by other authors attempting to improve on its results. One flaw with learned adversarial disturbances is that the level of variation seen during training of the disturbance policy may not be large enough, wherein high-risk/catastrophic disturbance policies may not be actively sought out or encountered by the adversary. Thus, the risk-averse RARL algorithm (Pan et al., 2019) tackles this by introducing a notion of risk (through the variance of an ensemble of value functions), where the agent will be risk-averse and the adversary risk-seeking - though this requires the agent to be trained alone initially to be able to cope with the risk-seeking behaviour of the adversary. This offline scheme results in fewer catastrophic events during testing on a discrete autonomous driving model. Another interesting use of the RARL framework was its application to performing model-free μ-synthesis (Keivan et al., 2022). The D-K-iteration (with static D scaling) mentioned in Section 2.2 was performed in a model-free fashion, where the K-step uses a modified RARL algorithm (Zhang et al., 2020a) and the D-step uses a model-free finite difference approximation.

Another approach to introducing robustness to reinforcement learning is to add disturbances not to the state (as previously discussed) but to the action (i.e. action robust RL). Applying perturbations to the action directly also leads to robustness, as deviations in the action space also simulate environmental changes. This concept was applied by Tessler et al. (2019), proposing two modifications to the MDP, the probabilistic action robust (PR) and noisy action robust (NR) MDP, with a joint policy mixing the actions of an agent and adversary (hence adding perturbations to the action). These two MDP formulations differ in how they encode action robustness, where the PR-MDP results in a deterministic policy (Equation 2.13) that encodes robustness to sudden/impulse adversarial actions, while the NR-MDP results in a stochastic policy (Equation 2.14) that encodes robustness to constant interrupting disturbances. For both methods, the effect of the adversary is controlled solely through the α hyperparameter.

$$\pi_P^{mix} = (1-\alpha)\pi + \alpha\bar{\pi} \tag{2.13}$$

$$\pi_N^{mix} = \mathbb{E}_{b\sim\pi,\bar{b}\sim\bar{\pi}}\left[\mathbb{1}_{a=(1-\alpha)b+\alpha\bar{b}}\right] \tag{2.14}$$

These action robust MDPs were combined with DDPG to create an offline RRL algorithm, the action robust DDPG (AR-DDPG, with two variants depending on whether the PR- or NR-MDP are used). This algorithm was applied to various MuJoCo robotics domains, empirically resulting in better robustness than the baseline DDPG algorithm, even leading to better performance without perturbations (i.e. with $\alpha = 0$) than the baseline. The advantage of this method over the transition robust designs is that it does not require any definition or knowledge of the uncertainty set, as it is implicitly defined through α. In another study (Deshpande et al., 2021), AR-DDPG was applied to quadcopter control for a waypoint navigation task considering the presence of both environment as well as parametric uncertainties, showing promising robustness results. Another study also used the concept of adversarial attacks to create an RRL framework (Tan et al., 2020), which uses additive perturbations to the action during training to improve robustness.

Modelled in a similar fashion to action robust RL designs, observation robust designs can be used to give variability to the state transition probabilities (and hence robustness to disturbances and uncertainties). This can be done by generating adversarial attacks on the agent's observations, in a similar manner to how generative adversarial networks work. This was done in several studies, by adding noise to the state observation (which can be randomly sampled or gradient dependent) and applied to Atari games (Huang et al., 2017) and several MuJoCo domains (Pattanaik et al., 2017). Another approach perturbs the agent observations with a bounded function, aiming for better robustness to sensor noise (Zhang et al., 2020b). This method can be applied to other non-robust RL algorithms (e.g. PPO or DDPG) to improve them.

2.3.3. Reinforcement learning with stability guarantees

One of the limitations of reinforcement learning mentioned in Section 2.1 is its lack of theoretical stability and convergence guarantees during training. This can lead to the controller diverging during online training, destabilising the system, and severely reducing the safety of implementing a classic reinforcement learning-based controller. Even though these controllers can find applicability without the guarantees of stability during training (even for flight control (Dally & Kampen, 2022)), it would be advantageous to find methods that assure stability during training, leading to safer RL.

One approach to enforcing stability in reinforcement learning is the one by Kretchmar et al. (2001). The aim of this research was to learn robust controllers with improved performance through RL while observing actual controlled behaviour (i.e. online learning), while guaranteeing that learning remains stable even as the network parameters are being altered during training. Integral quadratic constraints (IQC) are a tool for verifying the stability of a system with uncertainty and capture uncertainty descriptions. Two IQCs are used to convert the nonlinear dynamics of the hidden layer of a NN into an uncertainty function, and another one to account for the slow time-varying network weights by bounding their rate of change.

With these two IQCs, a nominal robust controller can be constructed from the system dynamics, guaranteeing the closed-loop stability of the system, and a feedforward NN is added in parallel. The two IQCs are applied, IQC-analysis can be used to calculate the maximum allowed perturbation for the network weights still leading to a stable controller, and these perturbation bounds are used to safely adjust the NN weights, which are trained with Q-learning.

One of the main limitations of this result is its dependence on knowledge of the system dynamics. Even if the nominal robust controller were synthesised in a model-free manner, the IQCs that provide the boundedness of the learning process require the model to be computed. Other results that use the same IQC framework to inject stability guarantees into the RL process (Anderson et al., 2007; Qin et al., 2012; Jin & Lavaei, 2018) lead to the same limitation.

Connected with the notion of placing bounds on the gradient of NN weights, as previously mentioned, bounds can also be placed on the control action provided by a reinforcement learning controller. The aim of this idea is to enforce the same provable robustness criteria of robust control in reinforcement learning control policies. Donti et al. (2021) tackle this problem by projecting the outputs of a nominal NN-based controller onto a space of stabilising control actions constructed from robust control specifications.

Starting from any common RL algorithm (in this case PPO (Schulman et al., 2017)), the resultant NN-parametrised policy ($\hat{\pi}_\theta$) can be transformed into a robust policy given a projection $\mathcal{P}$ onto some safe set $\mathcal{C}(x)$ (set of all actions for a given state x guarantees to satisfy an exponential stability condition), as seen in Equation 2.15. This approach was applied to several nonlinear dynamic systems, such as a cart-pole system, a planar quadrotor and a micro power grid, showing performance improvements under adversarial disturbances compared to 'classic' PPO and RARL (Pinto et al., 2017).

$$\pi_\theta(x) = \mathcal{P}_{\mathcal{C}(x)}(\hat{\pi}_\theta(x)) \tag{2.15}$$

However, this result falls into the same limitation as its IQC-based counterparts, where even though the base reinforcement learning algorithm does not require the model to be trained, creating the safe set $\mathcal{C}(x)$ and the associated projection $\mathcal{P}_{\mathcal{C}(x)}$ does require the system dynamics.

As noted, the previously mentioned methods to inject robust stability guarantees into reinforcement learning require the system dynamics to construct the necessary bounds and safety projections that lead to the guarantees. Inspired by this lack of fully autonomous robust control design with reinforcement learning, Wang et al. (2022) introduces a novel scheme for solving the nonlinear robust control problem online in a model-free way. The result from Donti et al. (2021) is learned without using the system dynamics and the reliance on an initial admissible control is removed. Three NNs are employed to learn the control policy, the value function and an additional model-dependent term.

As opposed to attempting to learn the projections onto safe sets, Wang et al. (2022) instead proposes a new value function ($\xi^\pi(x)$) formulation that encodes the same robustness objective in Equation 2.16. The new value function is defined as the infinite-horizon integral of the utility function $\bar{U}$, which is applied to remove the need for an initial stabilising control action.

$$\dot{V}^\pi(x) \leq -\lambda V^\pi(x) \tag{2.16}$$

The proposed method was applied to the stabilisation of a simple unstable nonlinear system, to the tracking control of a deep submergence rescue vehicle and to the stabilisation of a real rotary-inverted-pendulum, testing the *sim-to-real* gap of the learning algorithm. The proposed method was compared with an off-policy ZSG $\mathcal{H}_\infty$ controller (Luo et al., 2015) and a Q-learning algorithm (Vamvoudakis, 2017) for the first two control tasks, leading to better disturbance attenuation and stable learning online (as opposed to the off-policy ZSG $\mathcal{H}_\infty$ controller, which learned the wrong policy for the first task when no initial admissible control is given).

Motivated by the guarantees of robust $\mathcal{H}_\infty$ control theory, Deep Q-learning (a form of model-free RL) is placed in a control-oriented perspective, wherein the learning process is converted into an uncertain LTI system, which can then be explicitly controlled with a robust controller (Varga et al., 2023). While combining robust control theory and reinforcement learning has been applied to improve the controller performance, Varga et al. (2023) use it to improve the training performance instead, improving the convergence properties of learning. Firstly, the Q-learning process is converted into an uncertain LTI system with the neural tangent kernel (NTK). Using this model, classical $\mathcal{H}_\infty$ control techniques (as the ones described in Section 2.2) can be used to control this system. The authors create both a full-order mixed sensitivity $\mathcal{H}_\infty$ controller and a fixed-structure $\mathcal{H}_\infty$ controller. Unlike the model-based and model-free synergistic methods later described in subsection 2.3.4, this approach does not require an actual model of the environment, system or transition probabilities to construct the controller, as the NTK-LTI does not describe the system itself but the learning process of Deep Q-learning.

Within the approach of the disturbance robust RL designs described in the previous subsection 2.3.2, while adding an adversarial disturber to the learning process empirically improves the robustness to disturbances and uncertainties, those methods provide no theoretical guarantees of either the robustness itself, nor of the stability of the learning process (Moos et al., 2022; Zhang et al., 2020a) (i.e. while training, the agent may learn a destabilising policy, which would limit its applicability to an online training algorithm).

Such a limitation of disturbance robust RL was approached by Han et al. (2020), who propose the Robust Lyapunov Actor-Critic (RLAC) algorithm. This framework learns a Lyapunov function as the critic policy gradient, which guarantees the mean square stability during learning, leading to a learned policy that is robustly stable. The offline algorithm also uses an adversarial disturber to select the worst disturbance for a given state, allowing the agent to learn a policy that is robust against disturbances and uncertainties, similar to the RARL (Pinto et al., 2017) approach. RLAC was applied to a simulated cart-pole system, and compared against RARL, SAC (non-robust RL) and a model-based LQR controller, leading to robustness to impulsive disturbances and parametric uncertainty.

Another study has further addressed the lack of robustness and stability guarantees in disturbance robust RL, namely for RARL. The conventional RARL framework can learn a destabilising policy if the initial control policy does not exhibit a robust stability property against the adversarial policy (Zhang et al., 2020a). The RARL framework is then re-examined under the linear quadratic (LQ) robust control setting, to establish some theoretical guarantees of convergence, robust performance and stability during training. This results in an LQ double-loop RARL variant that provably converges to the optimal cost, while maintaining robust stability online, while requiring a robustly stabilising initial control policy. However, this initial policy can be learned from system data instead of using model-based $\mathcal{H}_\infty$ control methods. Unlike RARL, where the agent and adversary are updated sequentially, this LQ-RARL uses a double loop algorithm, where an outer-loop policy gradient update improves the agent's policy and the inner-loop solves the adversary policy for a fixed agent policy.

2.3.4. Synergistic methods

As mentioned in Section 2.2, classic robust control techniques require the system dynamics to be described, which may be intractable to accurately create for complex high-dimensional systems. Nonetheless, when a model of the system is available, its limitations in terms of modelling errors and uncertainties can be improved upon by combining a model-based controller (e.g. $\mathcal{H}_\infty$ robust controller) with a model-free reinforcement learning agent.

In many studies, synergising model-based and model-free control techniques, an adaptive augmentation is designed to improve upon the performance of the robust controller. In the ProxFly project (Zhang et al., 2025b), a cascaded model-based controller is compensated by an offline residual reinforcement learning algorithm (trained using PPO). This scheme was successfully applied to a challenging quadcopter control problem, where a main quadcopter is disturbed by the downwash of other smaller quadcopters (large disturbances), or the docking of a smaller one on the main quadcopter (large parameter uncertainty as the inertia of the quadcopter being controlled is altered). Furthermore, Pi et al. (2021) also apply PPO to a quadrotor control task; instead of using residual reinforcement learning, they design a model-based disturbance observer and compensator to improve robustness to environmental disturbances.

Also using the approach of residual reinforcement learning (wherein corrective actions are provided to a base controller), a model-based mixed sensitivity $\mathcal{H}_\infty$ controller mixed with residual RL (trained using PPO) was used for a blimp control task (Zuo et al., 2023). A mixing logic was used where a learnable parameter q controls how to combine the two control signals to improve performance and maintain robustness, following the logic $a_{mixed} = (1-q)a_{RL} + qu_{\mathcal{H}_\infty}$. Other studies applied a similar combination of $\mathcal{H}_\infty$ control and PPO top autonomous vehicle motion (Lelkó et al., 2023, 2024), though with a different mixing logic of actions, which results in safer control by placing the robust controller as the baseline and minimising the difference between the output of the mixer and the output of the RL agent. Also to improve the safety of an applied DDPG-based controller, one study implements a switching logic between the RL controller and a base LQR controller for the flight test validation of the control system (Shukla et al., 2020), switching from the DDPG-based to the LQR-based when the RL agent outputs are detected to propagate the aircraft into an unsafe zone.

3

Research Proposal

This research proposal aims to first identify the gaps in research of robust reinforcement learning, arising from the literature study in Chapter 2, and then address these gaps with the research questions defined in Section 3.1, followed by a plan of how to answer the same.

3.1. Research gap and questions

The general goal of this research is to investigate the combination of reinforcement learning and robust control techniques, combining the adaptability and robustness properties of these two methods, such that they can be applied to fault-tolerant control of an aircraft.

From the literature review, two approaches to this combination seem most promising for model-free flight control: robust adversarial reinforcement learning (RARL) techniques and approximate dynamic programming (ADP) to learn $\mathcal{H}_\infty$ controllers. RARL aims to apply a robust control approach to RL by learning competing control and disturbance policy using typical RL formulations (e.g., two competing SAC controllers). While these can learn control policies which are robust to disturbances and are adaptable to faults (as is also somewhat present in typical RL controllers), RARL policies suffer from poor stability during learning and have reduced guarantees of convergence, and are hence applied offline. Even though RARL techniques have been applied to quadcopter control (Deshpande et al., 2021; Lee & Lee, 2025), and to control of fixed-wing aircraft (Marquis et al., 2025; Liu et al., 2025), the fault-tolerant control problem has not been addressed.

Model-based $\mathcal{H}_\infty$ controllers have instead reduced adaptability to faults, as the faulty model is typically not readily available, making it harder to find a new controller online. ADP aims to improve this adaptability by removing the dependence on knowledge of the system dynamics, and as such, an $\mathcal{H}_\infty$ controller can be learned online. This formulation does profit from guarantees of convergence to an $\mathcal{H}_\infty$ controller designed with access to the system dynamics, and can be implemented online to perform an adaptation of the controller in either an off-policy or on-policy fashion. ADP approaches (namely integral reinforcement learning) have been applied to learn $\mathcal{H}_\infty$ controllers for the linearised F16 short period dynamics, for both regulation and tracking control problems, though fault-tolerance has not been approached directly.

These knowledge gaps form the basis for the research objective and research questions outlined below.

Research Objective

To enable fault-tolerance through the development of a model-free flight control system for a fixed-wing aircraft, by investigating how robust control techniques can be coupled with reinforcement learning.

From this research objective, three main research questions were identified, each of them detailed below with respective sub-questions.

Research Question 1

RQ1 Which robust reinforcement learning (RRL) framework is best suited to maximise tracking performance of an aircraft under disturbances, uncertainty and actuator faults?

RQ1.1 Which RRL formulations (e.g. $\mathcal{H}_\infty$ as min-max problem, robust adversarial RL, stability bounded/certified RL) are best suited to capture the dynamics of a simple system and maintain robust stability?

RQ1.2 To what extent does the RRL controller maintain stable performance in the presence of large model parameter deviations for a simple system?

Research Question 2

RQ2 How can the proposed RRL framework be integrated with the aircraft simulation model for fault-tolerant flight control?

RQ2.1 What type of control task and faults is the system subject to, and which control architecture should be designed for this?

Research Question 3

RQ3 What performance and robustness indicators are most appropriate for the RRL controller?

RQ3.1 How well does the RRL controller perform under simple manoeuvres (e.g. tracking accuracy)?

RQ3.2 How well does the controller adapt itself to faults or large model parameter changes?

3.2. Research plan

This section provides a brief outline of the plan of how each research question will be answered. **RQ1.1** was tackled in Chapter 2 with the literature study, examining the different approaches in the literature to combining robust control and reinforcement learning, and which ones are best suited for model-free fault-tolerant control. **RQ1.2** will be approached by implementing the results for example systems given in ADP/RL literature (e.g., cart-pole system or the linearised F16 short-period dynamics), evaluating how well these learned controllers perform in the presence of model uncertainties and disturbances.

RQ2.1 proceeds with the selection of an aircraft model and fault type which can be applied to the model. A system should be chosen to demonstrate the capabilities of the robust reinforcement learning algorithm in adapting a robust controller after a fault occurs, for the chosen tracking control task. The controller learning algorithm will then be implemented for the selected aircraft and the fault to be evaluated. As such, **RQ2** will be answered.

RQ3 aims to evaluate the performance of the learned controller, in terms of performance after learning has occurred, but also its stability during learning. This will be achieved by evaluating the tracking performance of a benchmark model-based controller and comparing it to that of the learned controller. The convergence of the controller during learning will also be analysed to evaluate the adaptation of the controller after a fault occurs.

The research is divided into three main phases. The literature study and research proposal take place in the first 10 weeks of the project. In the Midterm Research Phase (taking place between weeks 10 and 20), a selection of robust reinforcement learning algorithms will be made to be further assessed. Two to three results stemming from the research described in Section 2.3 will be replicated and evaluated for their suitability to encompass the research objective. Following this, in the Green Light Research Phase (from weeks 20 to 34), the chosen algorithm will be applied to the control of an aircraft, which shall also be selected at this stage. The control task and faults to be examined will be chosen, and an appropriate control system architecture shall be determined for this aircraft. From there, the robust reinforcement learning algorithm shall be implemented to design the control policy. Furthermore, the performance of the trained RRL control system will be investigated in terms of tracking accuracy and robustness to disturbances and model uncertainty, but also in terms of fault-tolerance. A more detailed description of the research timeline is given in Appendix A

Part II

Scientific Article

Model-free state-feedback $\mathcal{H}_\infty$ control using integral reinforcement learning for online adaptation to faults

José B. P. M. L. Cunha*

***Abstract* - $\mathcal{H}_\infty$ robust control is a powerful tool to design controllers robust against disturbance and model uncertainties. However, when a fault occurs in the system, $\mathcal{H}_\infty$ controllers typically have reduced adaptability to the fault, possibly requiring costly system identification steps to adapt the controller to the unknown faulty model. In this research, integral reinforcement learning (IRL) is applied to the $\mathcal{H}_\infty$ tracking control problem, to adapt a controller online without any knowledge of system dynamics, where only system trajectory data is available to the learning algorithm. Both off-policy and on-policy IRL methods are applied to a pitch-rate tracker for the linearised F-16 short period dynamics, after a fault of 50% reduction in elevator actuator effectiveness has occurred. The adapted controller converges to the model-based controller for the faulty system within $15\,\mathrm{s}$, and its tracking performance is compared to that of the controller for the nominal system.**

1 Introduction

Robust $\mathcal{H}_\infty$ control is a controller design tool which provides stability even in the presence of bounded disturbances and model uncertainties. Typical $\mathcal{H}_\infty$ controller synthesis involves knowledge of the system dynamics and is performed offline (Zhou & Doyle, 1998; Skogestad & Postlethwaite, 2005). However, when a fault occurs in the system, it can prove difficult for the controller to adapt to the fault (Lavretsky & Wise, 2024), as the model of the faulty system is typically not available online. As such, to improve the safety and autonomy of aircraft using robust controllers, it would be advantageous to find methods which improve the online adaptability of robust controllers.

Adaptive dynamic programming (ADP) is a reinforcement learning (RL) formulation (Buşoniu et al., 2010) which has been used extensively for feedback control (Lewis et al., 2012). These methods aim to use approximations of the value function (whether by a linear model or nonlinear function approximators) to learn control strategies with the use of online measured system data instead of relying solely on the system model.

ADP has been very successfully applied to optimal control problems (Vrabie et al., 2013), wherein the goal is to find a control policy which minimises a cost function or performance index. The advantage of using ADP for optimal control is that a control policy can be obtained in an iterative fashion without having to directly solve the algebraic Riccati equation (ARE) for linear systems or the Hamilton-Jacobi-Bellman (HJB) equation for nonlinear systems, even when the system dynamics are not known (Jiang & Jiang, 2013). For discrete-time (DT) systems, several ADP formulations have been developed and applied to various control problems, such as using dual heuristic dynamic programming (DHP) and iterative DHP (Werbos, 1992), Q-learning for output feedback (Lewis & Vamvoudakis, 2011) and incremental ADP for flight control (Zhou et al., 2018; de Alvear Cárdenas et al., 2021). However, due to the different structures of the HJB and ARE between DT and CT systems, the previous results cannot be readily applied to CT systems. Nonetheless, several CT approaches have been developed, by first discretising the system and using DT-ADP methods (Doya, 2000), model-free optimal control via integral reinforcement learning (IRL) (Jiang & Jiang, 2012; Lewis et al., 2012), Q-learning applied to CT systems (Vamvoudakis, 2017) and for tracking control with constrained inputs (Kiumarsi et al., 2016; Park et al., 2019).

$\mathcal{H}_\infty$ control presents a different approach than optimal control. Game theory provides a different insight into the design of $\mathcal{H}_\infty$ controllers. The equilibrium solution of a two-player zero-sum game (ZSG) - between a control policy and a disturbance policy - provides the solution of an $\mathcal{H}_\infty$ controller (Başar & Bernhard, 2008). ADP has been used to find the solution of this ZSG online, and with a reduced dependence on the system dynamics. In general, many ADP algorithms devised to solve the $\mathcal{H}_\infty$ control problem rely on the input dynamics of the system. These methods are referred to as model-based, while other methods extend to completely unknown dynamics (referred to as model-free). For DT systems, Al-Tamimi et al. (2007a) presented a model-based DHP algorithm to solve the ZSG, while others proposed a Q-learning approach (Al-Tamimi et al., 2007b; Kim & Lewis, 2010) or a policy iteration approach (Kiumarsi et al., 2017) to make the algorithm model-free. Other approaches have also been taken to find an $\mathcal{H}_\infty$ controller for fully unknown DT systems for a series of applications, namely using global DHP for regulation of nonlinear systems (Zhong et al., 2018), finding an output feedback controller for linear systems using policy iteration ADP (Fan et al., 2018; Valadbeigi et al., 2020) and for tracking control problems for linear (Liu et al., 2020) and nonlinear systems (Hou et al., 2020).

Similarly, the structure of the associated ARE or the Hamilton-Jacobi-Isaacs equation differs between DT and

*MSc student, Faculty of Aerospace Engineering, Control and Simulation Section, Delft University of Technology

CT systems, and hence different methods were developed. Morimoto and Doya (2005) proposed an ADP approach to solve an $\mathcal{H}_\infty$ control problem for CT systems, though requiring full knowledge of the system dynamics. Removing the dependence on the system matrix, Vrabie and Lewis (2011) used CT-HDP to solve the ZSG online for linear systems, while a simultaneous control and disturbance policy update algorithm was proposed for nonlinear systems (Wu & Luo, 2012; Luo et al., 2015).

Extending the ADP algorithms to a model-free solution to an $\mathcal{H}_\infty$ controller, Li et al. (2014) applied IRL to find an online on-policy solution for a regulation problem for linear CT systems, while Qin et al. (2016) used the same IRL approach but for off-policy learning. This result was later extended to nonlinear systems (Zhu et al., 2017; Perrusquía & Yu, 2021). For a tracking control problem, Modares et al. (2015) used an off-policy ADP method to remove the dependence on the system dynamics, which was later extended to tracking with input constraints for linear systems (Ahmadi et al., 2025) and with saturating actuators for nonlinear systems (Zhang et al., 2018). A similarity between these aforementioned $\mathcal{H}_\infty$ tracking controllers is that they require a discount factor to be used in the cost function to ensure that it is always bounded, and the controller structure consists of augmenting the system with the reference dynamics to be tracked. However, the discount term alters the disturbance attenuation condition present in $\mathcal{H}_\infty$ control, and is only valid for certain values of the discount factor (Modares et al., 2015). Thus, to better align with classic $\mathcal{H}_\infty$ controller design, the disturbance attenuation condition should be left unchanged, and a different controller structure chosen.

The contribution of this research is to improve upon the adaptability of robust controllers by developing a model-free adaptation to a nominal controller after a fault occurs. This research explores the use of adaptive dynamic programming (notably using IRL) for robust flight control, using the linearised short-period dynamics of an F16. Unlike the discounted tracking method used in the literature, this research approaches reference tracking by augmenting the system with an integrator, without the use of a discount factor.

This research is structured as follows. In Section 2, the foundations of using the ARE to synthesise an $\mathcal{H}_\infty$ controller for CT linear systems are explained and extended to the solution of a ZSG. In Section 3, a model-free IRL algorithm is derived, and two variants (off-policy or on-policy) are applied online using a least-squares method to find the controller. Section 4 details the application of these IRL procedures to the online adaptation of an F16 pitch-rate tracking controller after an elevator fault has occurred, without knowledge of the system dynamics. Finally, the conclusions are presented in Section 5.

Notation: $\|\cdot\|$ and $\|\cdot\|_2$ denote the vector L_1 and L_2 norms respectively, while $\|\cdot\|_\infty$ denotes the system $\mathcal{H}_\infty$-norm. Let $\mathbf{vec}(\cdot)$ denote the matrix operation where the columns of a matrix are stacked on top of one another, returning a vector. Let $\otimes$ denote the Kronecker product between two vectors or matrices. Let $\mathbf{diag}(\cdot)$ represent a diagonal matrix with these elements on the diagonals.

2 $\mathcal{H}_\infty$ CONTROLLER SYNTHESIS USING THE ARE

Consider the following class of linear time-invariant (LTI) continuous-time systems, described by

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B_u u(t) + B_w w(t) \\ y(t) &= C_y x(t) \\ z(t) &= C_z x(t) + D_{zu} u(t) + D_{zw} w(t) \end{aligned} \tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the system state, $u(t) \in \mathbb{R}^m$ the control inputs, $w(t) \in \mathbb{R}^q$ the external disturbances, $y(t) \in \mathbb{R}^n$ the plant outputs, and $z(t) \in \mathbb{R}^{n+m}$ the performance output. A, B_u, B_w, C_y, C_z, D_{zu} and D_{zw} are constant matrices.

The $\mathcal{H}_\infty$ control problem considers a class of external disturbance signals with bounded energy such that $w(t) \in L_2[0,\infty)$. For the remainder of the work herein, the control problem being considered is to find a state feedback control law which nominally stabilises the closed-loop system, while also minimising the $\mathcal{H}_\infty$-norm of the closed-loop transfer function from the external disturbance to the performance outputs ($T_{zw}(s)$). The latter condition can also be considered as complying with the following disturbance attenuation condition for all bounded energy disturbances

$$\int_0^\infty \left(\|z(t)\|^2\right) dt \leq \gamma^2 \int_0^\infty \left(\|w(t)\|^2\right) dt \tag{2}$$

where $\gamma > 0$ is a prescribed level of disturbance attenuation (or performance level). This disturbance attenuation condition can also be seen in terms of the $\mathcal{H}_\infty$ -norm of the closed-loop transfer function $T_{wz}(s)$

$$\begin{aligned} \frac{\sqrt{\int_0^\infty \|z(t)\|^2 dt}}{\sqrt{\int_0^\infty \|w(t)\|^2 dt}} = \frac{\|z(t)\|_2}{\|w(t)\|_2} \leq \gamma \quad \forall w(t) \in L_2[0,\infty) \\ \sup_{w \neq 0} \frac{\|z(t)\|_2}{\|w(t)\|_2} = \|T_{wz}(s)\|_\infty \leq \gamma \end{aligned} \tag{3}$$

The disturbance attenuation condition holds for a prescribed attenuation level $\gamma \geq \gamma^*$, where γ^* is the optimal (or lowest) attenuation level for which Equation 2 can hold.

The virtual performance output $z(t)$ can also be thought of as a weighted function of the state and control input with static weights, such that

$$\begin{aligned} \|z(t)\|^2 &= x(t)^\top Q x(t) + u(t)^\top R u(t) \\ z(t) &= \begin{bmatrix} \sqrt{Q}x \\ \sqrt{R}u \end{bmatrix} \end{aligned} \tag{4}$$

where $Q = C_z^\top C_z > 0$ and $R > 0$, meaning that both matrices must be positive semi-definite.

To find a suboptimal controller which stabilises the closed-loop system and which satisfies the disturbance attenuation condition in Equation 2, the following controller K can be used

$$K = R^{-1}B_u^\top P \qquad u(t) = -Kx(t) \tag{5}$$

where P is the solution to the associated Algebraic Riccati Equation (ARE), for given values of γ, Q and R

$$A^\top P + PA - PB_u R^{-1} B_u^\top P + \gamma^{-2} PB_w B_w^\top P + Q = 0 \tag{6}$$

As the prescribed disturbance attenuation level used to solve the ARE may not necessarily be the optimal value γ^*, a γ-iteration technique (Zhou et al., 1996) can be used to find the smallest $\gamma > 0$ for which a solution to Equation 6 still exists. However, as γ approaches its optimal value, the solution to the ARE approaches its feasibility boundary (as a $\gamma \geq \gamma^*$ is required for a stabilising solution to the ARE), leading to extremely large gains of the controller, which are not practical to apply. This occurs as the Hamiltonian matrix associated with this ARE will have eigenvalues on the imaginary axis when $\gamma = \gamma^*$, violating the condition for the ARE to have a stabilising solution, leading to the large gains when γ is close to optimal (Zhou & Doyle, 1998). Instead, a sufficiently small value of γ (i.e., taking fewer steps in the iteration process) can be used for more practical applications (Benner et al., 2007).

The weights in Q must also be carefully selected to reflect the control task (e.g., regulation or tracking) and the performance that is required from the system, thus ensuring that the optimal γ^* of the closed-loop system is close to one. As both Q and R scale the optimal performance level, R is typically set to one, leaving Q as the main tuning parameter for the controller.

In many $\mathcal{H}_\infty$ control problems, to better capture the design requirements (i.e. disturbance rejection, reference tracking, noise attenuation and control signal reduction), frequency-dependent weighting functions are used to shape the performance outputs, better reflecting how different signals and performance objectives should be emphasised or attenuated across a range of frequencies (Zhou & Doyle, 1998). However, in this case, only constant weights (through $\sqrt{Q}$ and $\sqrt{R}$) are applied, as the learning algorithm shown in Section 3 works in the time-domain. This has the limitation that as the $\mathcal{H}_\infty$ controller works only to minimise the peak of the closed-loop transfer function T_{wz}, it does not shape the relevant closed-loop sensitivity functions, leading to uniform performance across frequencies, poorer robustness margins and typically higher gain controllers.

The $\mathcal{H}_\infty$ control problem can also be reframed as a two-player Zero Sum Game (ZSG) (Başar & Bernhard, 2008), wherein one player tries to minimise a cost function while another player attempts to maximise it. In this case, a control policy $u(t)$ minimises a cost while a disturbance policy $w(t)$ maximises it. Then the solution to the controllers/policies can be found as the saddle point stabilising equilibrium of the ZSG. To match the $\mathcal{H}_\infty$ control problem, an infinite horizon cost function can be made such that it is quadratic in the states, controls and disturbances, by combining Equation 2 and 4

$$\begin{aligned} J(x,u,w) &= \int_0^\infty r(x,u,w)\ d\tau \\ &= \int_0^\infty \left(x^\top Q x + u^\top R u - \gamma^2 w^\top w\right)\ d\tau \end{aligned} \tag{7}$$

The optimal value function can therefore be found by solving the ZSG in Equation 8, resulting from the cost function under the optimal control and disturbance policies (u^*, w^*).

$$V^*(x) = \min_u \max_w J(x,u,w) \tag{8}$$

When the system has linear dynamics and an infinite horizon quadratic value function is used, the equilibrium solution (u^*, w^*) to the ZSG is equivalent to solving the ARE in Equation 6 (Başar & Bernhard, 2008; Vrabie & Lewis, 2011), leading to the same solution for the controller as Equation 5. The optimal value function V^* is found from the ARE solution (being quadratic in the state), and the optimal policies follow from it,

$$\begin{aligned} V^* &= x^\top P x \\ u^* &= -\frac{1}{2} R^{-1} B_u^\top \nabla V^* = -R^{-1} B_u^\top P x \\ w^* &= \frac{1}{2} \gamma^{-2} B_w^\top \nabla V^* = \gamma^{-2} B_w^\top P x \end{aligned} \tag{9}$$

where $\nabla V^* = \partial V^* / \partial x$. The value function is equivalent to the cost function under specified control and disturbance policies, $V(x) = J(x,u,w)|_{u,\,w}$.

One approach to solve the ARE (or equivalently solve the ZSG arising from the $\mathcal{H}_\infty$ control problem) is using the Newton-Kleinman method (Kleinman, 1968), a description of which is given in Algorithm 1. This approach iteratively updates the controller gains K and L until convergence to the analytic solution of the ARE.

First, the solution to the controller gains in terms of the ARE solution $K = R^{-1}B_u^\top P$ and $L = \gamma^{-2} B_w^\top P$ are applied to Equation 6

$$A^\top P + PA - PB_u K + PB_w L + Q = 0 \tag{10}$$

Following this, the closed-loop dynamics are substituted into the above equation, where $A_{cl} = A - B_u K + B_w L$, resulting in the following Lyapunov equation

$$A_{cl}^\top P + P A_{cl} + Q_L = 0 \tag{11}$$

$$\begin{aligned} Q_L &= K^\top B_u^\top P - L^\top B_w^\top P + Q \\ &= K^\top R K - \gamma^2 L^\top L + Q \end{aligned} \tag{12}$$

The Newton-Kleinman algorithm requires initial controller gains K_0 and L_0 which render the closed-loop system stable. From there, an initial solution to the above Lyapunov equation P_0 is found. With this solution, the new controller gains K_1 and L_1 are calculated according to Equation 14. This is repeated, until P_i converges.

It should be noted that this offline algorithm requires knowledge of the system input dynamics B_u and B_w. In the following section, a different iterative method is used to find the solution to the ARE and the controller gains for systems with fully unknown dynamics.

Algorithm 1: Offline model-based iterative solution to the Algebraic Riccati Equation (Newton-Kleinman)

1. Select initial controller gains K_0 and L_0 which stabilise the closed loop system.

2. Find the solution P_i to the following Lyapunov equation, where $A_{cl,i} = A - B_u K_i + B_w L_i$ and $Q_{L,i} = Q - \gamma^2 L_i^\top L_i + K_i^\top R K_i$.

$$A_{cl,i}^\top P_i + P_i A_{cl,i} + Q_{L,i} = 0 \tag{13}$$

3. From this solution, calculate the new controllers

$$\begin{aligned} K_{i+1} &= R^{-1} B^\top P_i \\ L_{i+1} &= \gamma^{-2} B_w^\top P_i \end{aligned} \tag{14}$$

4. Repeat step **(2)** until convergence as

$$\lim_{i \to \infty} P_i = P \tag{15}$$

3 Integral Reinforcement Learning Solution to the ARE

Offline methods used to find an $\mathcal{H}_\infty$ controller by solving the associated ARE (such as the one discussed in Section 2 or using MATLAB's `icare`[1]) typically require full knowledge of the system dynamics. Approximate dynamic programming (ADP) has been used to learn the $\mathcal{H}_\infty$ controller that arises from solving an ARE for many control problems, namely regulation (Wu & Luo, 2012; Luo et al., 2015), tracking (Modares et al., 2015) and for systems with saturating actuators (Zhang et al., 2018; Hou et al., 2020). These were initially applied to partially unknown system dynamics (where only the input/noise matrices are required) (Vrabie & Lewis, 2011; Yasini et al., 2015) and further extended to remove the dependency on knowledge of the system dynamics.

[1]Solver for continuous-time algebraic Riccati equations, mathworks.com/icare (last accessed 08/05/2026)

In this section, an ADP method is derived for linear continuous-time systems with no knowledge of the system dynamics, using the zero-sum game approach to learn the ARE-based solution to an $\mathcal{H}_\infty$ controller. The online implementation of the algorithm is shown using a least-squares method, being applied in both an off-policy and on-policy manner.

3.1 Data-driven ADP for an $\mathcal{H}_\infty$ controller

Integral reinforcement learning (IRL) (Vrabie et al., 2013) is an ADP technique which aims to remove the dependency of an optimal or $\mathcal{H}_\infty$ controller on the system dynamics. The principle of IRL comes from first noting that the Bellman equation associated with Equation 7 (found by differentiating the cost function with respect to the state),

$$0 = r(x, u, w) + \nabla V^\top (Ax + B_u u + B_w w) \tag{16}$$

depends on the system dynamics. This equation is used to solve the optimal value function leading to the state feedback controllers u and w, which is equivalent to solving the ARE (Başar & Bernhard, 2008). Taking the integral of Equation 16 on the interval $[t, t+T]$ leads to the IRL Bellman equation,

$$\begin{aligned} 0 &= \int_t^{t+T} \left(r(x, u, w) + \nabla V^\top \dot{x} \right) \, d\tau \\ &= \int_t^{t+T} r(x, u, w) \, d\tau + \int_t^{t+T} \frac{d}{dt} V(x(\tau)) \, d\tau \end{aligned}$$

$$V(x(t)) = \int_t^{t+T} r(x, u, w) \, d\tau + V(x(t+T)) \tag{17}$$

which, unlike Equation 16, does not directly depend on the system dynamics, and is equivalent to the Bellman equation. From this principle, several algorithms (Lewis et al., 2012; Xiao et al., 2018; Moghadam & Lewis, 2019) have been developed to find the solution to the controllers. However, in the form shown in Equation 17, only the optimal value function is found, and to then calculate the controllers u and w, the system matrices B_u and B_w are required.

This can be seen as a policy iteration method, where the policy evaluation step consists of solving Equation 17 using admissible policies (which does not require knowledge of the system dynamics), followed by the policy improvement step, which uses the optimal value function to iteratively improve the policies (requiring the system dynamics). Value iteration algorithms have also been developed using IRL (Vrabie & Lewis, 2011). To then remove the need for knowledge of B_u and B_w, the policy evaluation and improvement steps can be combined (Li et al., 2014). Adapted from the work of Li et al. (2014) and Qin et al. (2016), the derivation of the model-free IRL method follows here.

First, the system dynamics in Equation 1 are rewritten in terms of the closed-loop dynamics, by applying the feedback

controllers $u_i = -K_i x$ and $w_i = L_i x$.

$$\begin{aligned}\dot{x} &= Ax + B_u u_i + B_w w_i + B_u(u - u_i) + B_w(w - w_i) \\ &= (A - B_u K_i + B_w L_i)x + B_u(u - u_i) + B_w(w - w_i) \\ &= A_{cl} + B_u(u - u_i) + B_w(w - w_i)\end{aligned} \tag{18}$$

The control and disturbance policies to be learned are denoted by u_i and w_i, while the values u, w are the control and disturbance inputs applied to the system. These do not have to be the same due to the introduction of excitation signals or the difference in the controller gains used during data collection when comparing off-policy and on-policy algorithms.

Considering that the value function in Equation 7 can be written in terms of the ARE solution, such that $V_i = x^\top P_i x$, its time derivative is calculated, and the value of $\dot{x}$ substituted from Equation 18

$$\begin{aligned}\dot{V}_i &= \dot{x}^\top P_i x + x^\top P_i \dot{x} \\ &= x^\top A_{cl}^\top P_i x + x^\top P_i A_{cl} x + 2(u - u_i)^\top B_u^\top P_i x \\ &\quad + 2(w - w_i)^\top B_w^\top P_i x\end{aligned} \tag{19}$$

From here, Equation 11 and Equation 12 can be used to remove the dependency on the knowledge of the system dynamics, leading to the following

$$\dot{V}_i = -x^\top Q_{L,i} x + 2(u - u_i)^\top R K_{i+1} x + 2(w - w_i)^\top \gamma^2 L_{i+1} x \tag{20}$$

Integrating Equation 20 between t and $t+T$ for a time interval $T > 0$ leads to

$$\begin{aligned}V_i(t+T) - V_i(t) &= x(t+T)^\top P_i x(t+T) - x(t)^\top P_i x(t) \\ &= -\int_t^{t+T} x^\top Q_{L,i} x \; d\tau \\ &\quad + 2\int_t^{t+T} (u - u_i)^\top R K_{i+1} x \; d\tau \\ &\quad + 2\gamma^2 \int_t^{t+T} (w - w_i)^\top L_{i+1} x \; d\tau\end{aligned} \tag{21}$$

Equation 21 describes the policy iteration algorithm used in IRL literature (Li et al., 2014; Qin et al., 2016), wherein the policy evaluation and improvement steps are combined into one equation, without the need for knowledge of B_u and B_w. This iterative method (along with other similar ADP and IRL algorithms) is shown to converge to the same controller as the one obtained from the ARE, as it can be seen as a model-free application of Algorithm 1 (Lewis et al., 2012; Li et al., 2014).

However, as Equation 21 is a one-dimensional equation, to find P_i, K_{i+1} and L_{i+1}, one requires at least $N_s \geq n(n+1)/2 + nm + nq$ equations, which are used to solve for the unknown parameters with a least-squares method. Equation 21 is converted into a form more amenable to a least-squares method, using the following identity: for two appropriately sized vectors a, b and matrix M, it follows that $a^\top M b = (a \otimes b)^\top \mathbf{vec}(M)$ (Magnus & Neudecker, 2019). The different components of Equation 21 are formatted into

$$\begin{aligned}(u - u_i)^\top R K_{i+1} x &= (u \otimes x)^\top (I_n \otimes R)\mathbf{vec}(K_{i+1}) \\ &\quad + (x \otimes x)^\top (I_n \otimes K_i^\top R)\mathbf{vec}(K_{i+1})\end{aligned} \tag{22}$$

$$\begin{aligned}(w - w_i)^\top R K_{i+1} x &= (w \otimes x)^\top \mathbf{vec}(L_{i+1}) \\ &\quad + (x \otimes x)^\top (I_n \otimes L_i^\top)\mathbf{vec}(L_{i+1})\end{aligned} \tag{23}$$

$$x^\top Q_{L,i} x = (x \otimes x)^\top \mathbf{vec}(Q_{L,i}) \tag{24}$$

The terms for $V_i = x^\top P_i x$ are similarly converted in Equation 25,

$$\begin{aligned}x(t+T)^\top P_i x(t+T) - x(t)^\top P_i x(t) = \\ (\bar{x}(t) - \bar{x}(t+T))^\top \hat{P}_i\end{aligned} \tag{25}$$

where $\hat{P}$ refers to the vector stacking the diagonal and upper triangular part of P, where the off-diagonal elements are equal to $2p_{ij}$, and $\bar{x}(t)$ denotes the Kronecker product quadratic polynomial basis vector with elements $\{x_i(t)x_j(t)\}_{i=1,n;j=i,n}$. As P is symmetric, the Kronecker product identity is not used, and thus $n(n-3)/2$ equations can be removed.

Combining the previous equations leads to Equation 21 in a form more suitable to be solved with least-squares, where one such equation can be constructed for each sample k within the time interval T, with a sampling rate of Δt,

$$\psi_k \begin{bmatrix} \hat{P}_i \\ \mathbf{vec}(K_{i+1}) \\ \mathbf{vec}(L_{i+1}) \end{bmatrix} = \theta_k \qquad k \in [0, 1, ..., N_s] \tag{26}$$

where

$$\theta_k = \int_{t+k\Delta t}^{t+(k+1)\Delta t} (x \otimes x)^\top \mathbf{vec}(Q_{L,i})$$

$$\psi_k = \begin{bmatrix} (\bar{x}(t) - \bar{x}(t+\Delta t))^\top, \\ \int_{t+k\Delta t}^{t+(k+1)\Delta t} (u \otimes x)^\top (I_n \otimes R) + (x \otimes x)^\top (I_n \otimes K_i^\top R)\, d\tau, \\ \int_{t+k\Delta t}^{t+(k+1)\Delta t} (w \otimes x)^\top + (x \otimes x)^\top (I_n \otimes L_i^\top)\, d\tau \end{bmatrix}$$

The N_s different samples can then be collected into a system of equations,

$$\Psi \begin{bmatrix} \hat{P}_i \\ \mathbf{vec}(K_{i+1}) \\ \mathbf{vec}(L_{i+1}) \end{bmatrix} = \Theta$$

where $\Psi = [\psi_0, \psi_1, ..., \psi_{N_s-1}]^\top$ and $\Theta = [\theta_0, \theta_1, ..., \theta_{N_s-1}]^\top$. The updated controller gains and value function are found from the least-squares solution, given that Ψ has full column rank (i.e., at least N_s linearly independent vectors),

$$\begin{bmatrix} \hat{P}_i \\ \mathbf{vec}(K_{i+1}) \\ \mathbf{vec}(L_{i+1}) \end{bmatrix} = \Psi^+ \Theta \tag{27}$$

where Ψ^+ refers to the Moore-Penrose pseudoinverse.

3.2 Importance of persistence of excitation

To ensure that the least-squares problem has a unique solution (i.e., rank$(\Psi) \geq N_s$), excitation signals e_u and e_w are injected into the control and disturbance inputs applied to the system during training, such that $u = -Kx + e_u$ and $w = Lx + e_w$. This can also be seen as satisfying a persistency of excitation (PE) condition (Narendra & Annaswamy, 1984), as sufficiently rich information is needed in Ψ and Θ such that K and L converge to the analytic gains found when solving the ARE directly. The IRL algorithm can be seen as performing an implicit system identification of the B_u and B_w matrices from the input-output data collected while learning P, K and L, and hence persistently exciting input signals are required.

In general, two distinct excitation signals (e_u and e_w) are required for convergence to the analytic solution. The IRL method is formulated as a ZSG between the control and disturbance policies, and hence, during learning, both K and L must be learned to find an equilibrium of the value function. Therefore, e_u is required to learn K and e_w is required to learn L, to both satisfy the PE condition to identify the controller gains, but also to ensure the full column rank condition needed for the least-squares solution.

For the IRL methods to converge to the analytic solution of the ARE, careful selection of the excitation signals is required. Even though the introduction of persistently exciting signals does not affect the convergence properties of the ADP methods (Li et al., 2014), the signals have a large effect on the accuracy of the convergence of the gains towards K_{are} and L_{are}. If the signals are not selected such that they excite the relevant eigenmodes of the system, then the gains will converge to completely different values than the analytic ones. Likewise, the signals must not be strong enough such that they destabilise the system or surpass the actuator limits of the system.

Some knowledge of the system may therefore be required to design the excitation signals, to ensure that there is enough information in the state trajectories and the control/disturbance inputs during training, such that a correct identification of the controller gains can be performed. For some systems, small zero-mean Gaussian noises may be sufficient to excite the system, while for others, multisine signals or frequency sweeps designed around the bandwidth of the system may be required.

Another important aspect of the choice of excitation signals relates to the trade-off between safety during learning and the speed of adaptation of the controller. To ensure that the states do not diverge too much during learning (thus leading the system into an unsafe operating regime), the magnitude of the excitation should be kept relatively small. On the other hand, too small an excitation may lead the collected data to not be informative enough during the specified measurement window to ensure that the controller converges to the analytic result. This leads to slower adaptation of the controller, and the measurement time needs to be increased. This concept is similar to the exploration versus exploitation dilemma in reinforcement learning (Vrabie et al., 2013).

3.3 Off- and on-policy algorithms

With Equation 21, the policy iteration and update step of the IRL method is defined, while Equation 27 can be applied iteratively to learn the controllers. Even though these equations form the basis for an online solution to the ARE using only input-output data, different approaches can be taken to learn the controllers, namely, with respect to how and when the policy is updated in the learning process. For an off-policy learning method, a different policy is used for collecting data than the one that is being updated, whereas on-policy methods improve the policy which is being used to generate the data.

Two different online model-free IRL algorithms were implemented, particularly an off-policy (Algorithm 2) and an on-policy (Algorithm 3) version. The main difference in the implementation of these algorithms comes from the definition of the $(u - u_i)$ and $(w - w_i)$ terms in Equation 21. For the off-policy algorithm, only the initial control and disturbance policies are used to generate the data (with the excitation signals added onto these), whereas for the on-policy algorithm, at each iteration, different policies are used. These differences are summarised below.

$$\begin{aligned}
\text{Off-policy}: \quad & u = -K_0 x + e_u \quad , \quad w = L_0 + e_w \\
& u - u_i = -K_0 x + e_u + K_i x \\
& w - w_i = L_0 x + e_w - L_i x \\
\text{On-policy}: \quad & u = -K_i x + e_u \quad , \quad w = L_i + e_w \\
& u - u_i = -K_i x + e_u + K_i x = e_u \\
& w - w_i = L_i x + e_w - L_i x \quad = e_w
\end{aligned}$$

The other main difference between the methods is at which point of the data collection step the policy iterations occur. For the off-policy algorithm, input-output data is collected during a large time window, after which all the policy iterations occur consecutively, reusing the same state trajectories, inputs and disturbances that were collected. On the

other hand, the on-policy algorithm collects data for a typically shorter time window, and performs only one policy iteration, after which more data is collected using the new policies and the process is repeated.

The number of iterations n_{iter} and the time window size T are some of the main hyperparameters to fine-tune the performance of the algorithms, along with the selection of excitation signals and the initialisation of the control gains (K_0, L_0) and the state (x_0). It should be noted that, as both algorithms are applied online, the initial state cannot be reset after each iteration for the on-policy version, which is a common technique to improve stability in ADP/RL methods (Andrychowicz et al., 2021).

Similarly to how the Newton-Kleinman algorithm requires initial stabilising controllers to converge to the ARE solution, this IRL policy iteration method also requires the initial controllers K_0 and L_0 to stabilise the closed-loop system. Such a requirement is needed for learning stability, as without an initial stabilising controller, the state trajectories could diverge when excited with e_u and e_w, as the closed-loop system would be unstable. Thus, the identification of the controllers from input-output data for the following iteration would not be correct, and the controller would not converge to the analytic ARE solution. In general, this condition can be quite complex to achieve in a completely model-free manner, especially if the system is not open-loop stable, as some information about the system dynamics would be needed to find an initial controller to be adapted.

Algorithm 2: Online off-policy model-free IRL for $\mathcal{H}_\infty$ control

1. Given the initial stabilising controllers with added excitation signals $u = -K_0 x + e_u$ and $w = L_0 x + e_w$, collect the state trajectories with these inputs for T seconds.
2. Solve Equation 27 to find P_i, K_{i+1} and L_{i+1}.
3. Re-use the state trajectories collected in step **(1)** and the new control gains to again solve Equation 27.
4. Repeat step **(3)** until the gains have converged.

4 Tracking control with fault adaptation

In this section, the $\mathcal{H}_\infty$ tracking control problem is introduced, and the associated ARE is first solved using the system model to find a state-feedback controller for the linearised F-16 short-period dynamics. Following this, the ADP learning methods described in Section 3 are applied to the tracking control problem, wherein a fault is first introduced to the system, and the controller is adapted online to this fault. Finally, the adapted controller is evaluated on a pitch-rate tracking task.

Algorithm 3: Online on-policy model-free IRL for $\mathcal{H}_\infty$ control

1. Given the initial stabilising controllers with added excitation signals $u = -K_0 x + e_u$ and $w = L_0 x + e_w$, collect the state trajectories with these inputs for T seconds.
2. Solve Equation 27 to find P_i, K_{i+1} and L_{i+1}.
3. Update the control and disturbance inputs with the new controller, such that for the i^{th} iteration, the inputs are $u = -K_i x + e_u$ and $w = L_i x + e_w$, and collect the state trajectories for another T seconds.
4. Again solve Equation 27 using the new collected state trajectories.
5. Repeat step **(3)** and **(4)** until the gains have converged.

4.1 *Tracking control problem*

Consider an LTI system as in Equation 1. To create a control augmentation system (CAS), such that one of the states tracks a reference signal, one can augment the state with an integral of the tracking error, such that $X = [x, x_i]^\top$, where r is a reference to be tracked and y_r is the output which should follow the reference. Therefore, the integral tracking error is defined as

$$x_i = \int_{t_0}^{t} \Big(r(t) - y_r(t) \Big) d\tau$$

With this extra state, the augmented plant dynamics can be constructed as

$$\begin{bmatrix} \dot{x} \\ \dot{x_i} \end{bmatrix} = \begin{bmatrix} A & \mathbf{0}_{n\times 1} \\ C_{yr} & 0 \end{bmatrix} \begin{bmatrix} x \\ x_i \end{bmatrix} + \begin{bmatrix} B_u \\ 0 \end{bmatrix} u + \begin{bmatrix} \mathbf{0}_{n\times 1} \\ 1 \end{bmatrix} r \quad (28)$$

The system that shall be used in this study to apply the IRL algorithms described in Section 3 is the linearised F-16 short-period dynamics (trimmed at steady and level flight with a speed of $V_T = 502\,\text{ft/s}$ and a centre of gravity position at $x_{\text{c.g.}} = 0.35\bar{c}$) (Stevens et al., 2016). The system matrices for this flight condition are given below, where the states $x = [\alpha, q, \delta_e]$ consist of the angle of attack (rad), the pitch rate (rad/s) and the elevator deflection (rad), and the input is the commanded elevator deflection $u = \delta_e^{\text{cmd}}$. A fast first-order actuator is used to model the elevator deflection, such that $\dot{\delta}_e = -20.2\delta_e + 20.2\delta_e^{\text{cmd}}$.

$$A = \begin{bmatrix} -1.01887 & 0.90506 & -0.12318 \\ 0.82225 & -1.07741 & -10.058 \\ 0 & 0 & -20.2 \end{bmatrix} \quad B_u = \begin{bmatrix} 0 \\ 0 \\ 20.2 \end{bmatrix}$$

$$C = \mathbb{I}_3 \qquad D = \mathbf{0}_{3\times 1}$$

The goal is to design a pitch-rate tracker CAS for this system, and hence the plant augmentation shown in Equation 28 is used, where $C_{yr} = [0, 1, 0]$ (as the second state, q, is the one that should track a reference).

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\delta}_e \\ \dot{q}_i \end{bmatrix} = \begin{bmatrix} -1.01887 & 0.90506 & -0.12318 & 0 \\ 0.82225 & -1.07741 & -10.058 & 0 \\ 0 & 0 & -20.2 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \delta_e \\ q_i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 20.2 \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} r \tag{29}$$

With access to the system matrices, a state-feedback controller can be synthesised using the ARE in Equation 6. As discussed in Section 3, the weights Q and R must be carefully designed to reflect the tracking control problem, and to ensure that a close-to-optimal controller is found with an attenuation level $\gamma \approx 1$. To this sense, the diagonal matrix Q (which will now have four entries on the diagonals due to the added integrator state) is chosen to penalise the q_i tracking error, but also apply some smaller penalty to the other states, as seen in Equation 30. The matrix R is selected as $R = 1$.

$$Q = \begin{bmatrix} q_\alpha = 0.5 & 0 & 0 & 0 \\ 0 & q_q = 0.5 & 0 & 0 \\ 0 & 0 & q_{\delta_e} = 0.051 & 0 \\ 0 & 0 & 0 & q_{q_i} = 12 \end{bmatrix} \tag{30}$$

With Q and R chosen for this control problem, the ARE can be solved, wherein a γ-iteration bisection algorithm is employed to find a controller such that $\|T_{wz}\|_\infty \approx \gamma^*$. The optimal performance level is found to be $\gamma^* = 0.976$, which leads to a controller or extremely large gains, which would not be practical to apply. Instead, as mentioned in Section 2, γ can be slightly increased away from the optimal value, leading to the following controller with $\gamma = 0.9998$.

$$K_{\text{are}} = \begin{bmatrix} 3.668 & -7.9121 & 2.951 & 52.36 \end{bmatrix} \tag{31}$$

This controller can then be evaluated on a simple pitch-rate tracking task, to ensure that the weighting matrices are being correctly chosen, and that the physical actuator limits of the model are not being surpassed (namely, for a commanded $1\,\text{deg/s}$ pitch-rate, the elevator deflection rate should not exceed $\pm 2.4\,\text{deg/s}$). Figure 1 shows the tracking response under the state feedback law $u = -K_{\text{are}}x$, leading to a fast rise time of $217\,\text{ms}$, while Figure 2 shows the elevator deflection rate, which is shown to be within limits.

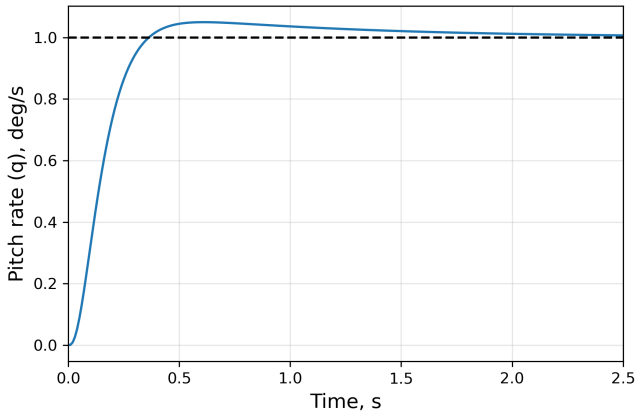


Figure 1: *Tracking response of the nominal system to a* $1\,\text{deg/s}$ *pitch-rate command using* K_{are}.

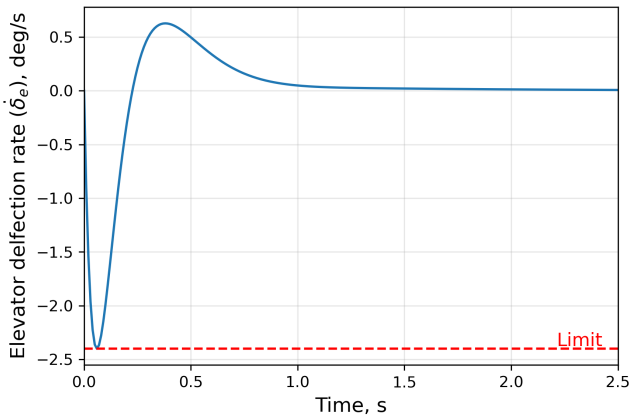


Figure 2: *Elevator deflection rate during pitch-rate tracking using* K_{are}.

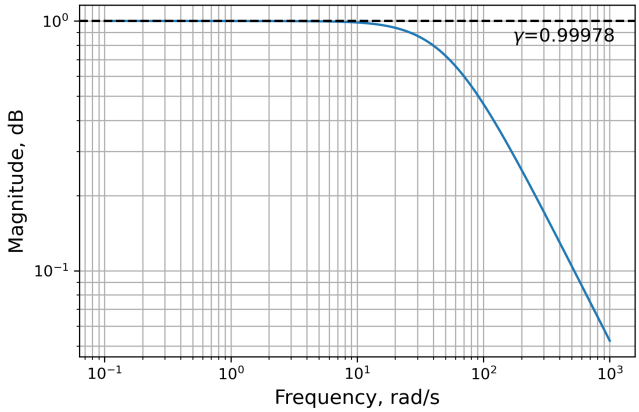


Figure 3: *Singular values plot of* $T_{wz}(s)$ *of nominal system using* K_{are}.

The singular values of the closed-loop transfer function $T_{wz}(s)$ are shown in Figure 3, where the effect of the $\mathcal{H}_\infty$

controller can be seen, reducing the peak of $T_{wz}(s)$ to the performance level γ. This plot also shows the limitation of only using constant weights Q and R, as $T_{wz}(s)$ cannot be shaped at low or high frequencies to better reflect the desired performance requirements, and hence only the peak of this transfer function is reduced.

4.2 *Fault adaptation with IRL method*

Solving the ARE directly to find the state feedback tracking controller requires full knowledge of the system dynamics. To remove this reliance, the IRL algorithms described in Section 2 can be applied to this tracking control problem, to find an $\mathcal{H}_\infty$ controller online. As described in Section 1, the goal is to learn an online adaptation to the $\mathcal{H}_\infty$ controller after a fault is introduced, without knowledge of the system dynamics. For this research, the moment in which the fault occurs is assumed to be known, thus no fault detection methods are integrated. The fault in question is a 50% reduction in elevator actuator effectiveness, reflected in a reduction to the input matrix, and leading to the following actuator dynamics

$$B_u^f = 0.5 B_u \tag{32}$$

$$\dot{\delta}_e = -20.2\delta_e + 10.1\delta_e^{\text{cmd}} \tag{33}$$

Even though the faulty model is typically not available during flight, as a benchmark for the convergence of the IRL algorithms, the ARE can be solved for the faulty system, leading to the controller which should be achieved after learning. Due to the reduction in elevator actuator effectiveness, it is no longer feasible to request the same performance as with the nominal system, and hence, the Q weights should be reduced to lead to the same optimal γ as the nominal system. With the new weights chosen as $Q^f = \mathbf{diag}(0.25, 0.25, 0.025, 6)$, the ARE controller for the faulty system is the following, with a performance level of $\gamma = 0.991$

$$K_{\text{are}}^f = \begin{bmatrix} -0.07265 & -3.876 & 1.686 & 13.699 \end{bmatrix} \tag{34}$$

To better evaluate the stability robustness of these model-based controllers (comparing the nominal and faulty controllers), the gain and phase margins can be found from the Bode plots of the input loop transfer function $L_i(s) = K(s)G(s)$ and output loop transfer function $L_o = G(s)K(s)$, where $G(s)$ is the respective transfer matrix of the augmented plant defined in Equation 29 and Equation 32, and $K(s)$ is the respective controller. These margins are detailed in Table 1. Such high stability margins at the plant input are expected from a controller such as this, due to its similarities with an LQR controller for tracking (which also possesses excellent stability margins at the plant input).

Table 1: *Stability margins of input and output loop transfer functions for nominal and faulty systems with respective nominal and faulty model-based controllers.*

Margin	Nominal system	Faulty system
GM(L_i)	$+\infty$ dB	$+\infty$ dB
PM(L_i)	83.6 °	80.5 °
GM(L_{o,q_i})	22.2 dB	21.5 dB
PM(L_{o,q_i})	69.4 °	73.2 °

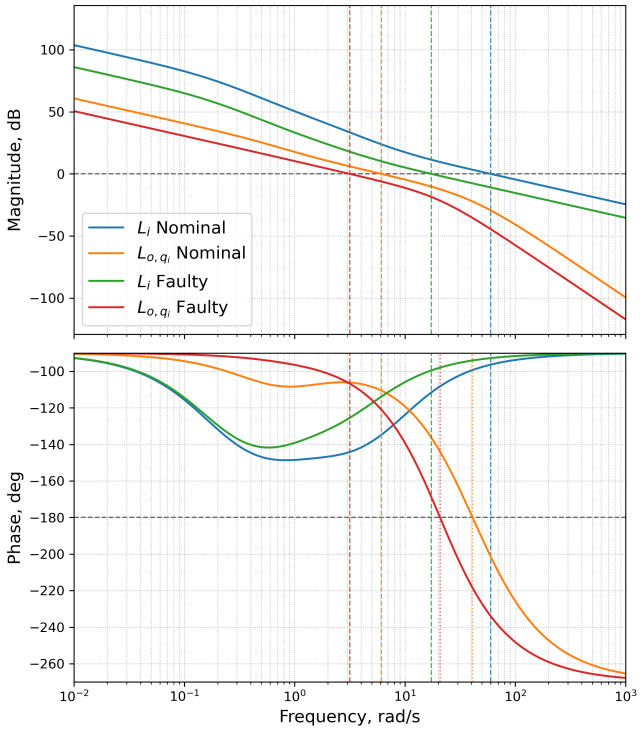


Figure 4: *Bode plots of input loop transfer function L_i of nominal system with K_{are} and faulty system with K_{are}^f*

To now adapt the controller for the nominal system without knowledge of the faulty system dynamics, one must first consider one of the main assumptions of the IRL methods used, namely the need for an initial stabilising controller. As the augmented system is not open-loop stable due to the pole at zero introduced by the integrator state, a null initial controller of all zeros cannot be chosen, and another initial controller must be found. This is not the case for a regulation problem for this system, as it is open-loop stable, a null initial controller can be used, and a controller can be learned in a fully model-free sense using ADP techniques, as has already been studied in the literature (Li et al., 2014; Qin et al., 2016).

As the aim of this research is to adapt a controller after a fault, a possible initial controller would be one used for the nominal system, shown in Equation 5. However, depending

on the fault, the controller used for the nominal system may not be stabilising for the faulty system, and another controller needs to be found to initialise the IRL algorithms. For the system and fault being treated here, the ARE-based controller for the nominal system stabilises the faulty system, and hence can be used as an initial controller.

Applying Algorithm 2 and Algorithm 3, an $\mathcal{H}_\infty$ tracking controller can be found using either off-policy or on-policy IRL, respectively. For the simulation of the system dynamics, a sampling rate of 100 Hz was used. Table 2 details the parameters selected for both algorithms. The persistence of

Table 2: *Parameters selected for IRL learning process of Algorithm 2 and Algorithm 3.*

Parameter	Off-policy	On-policy
T	15 s	2.5 s
n_{iter}	6	6
K_0 , L_0	ARE solution for nominal system	
x_0	$[0,0,0,0]^\top$	$[0,0,0,0]^\top$ for first iteration then $x(iT)$ for $(i+1)^{\text{th}}$ iteration

excitation condition is satisfied through the use of the excitation signals e_u and e_w. These were carefully chosen for this system, and hence also require some domain knowledge of the system, just as in the selection of the initial stabilising controllers. These were defined as the following multisine signals, also shown in Figure 5.

$$\begin{aligned} e_u(t) = \Big(&0.28\sin(0.7t+0.1) \;+\\ &0.18\sin(2.1t+0.8) \;+\\ &0.12\sin(4.8t+1.5) \;+\\ &0.08\sin(8t+0.2)\Big)\\ e_w(t) = \Big(&0.12\sin 0.5t+0.3) \;+\\ &0.08\sin(1.6t+1) \;+\\ &0.05\sin(3.6t+0.6) \;+\\ &0.03\sin(6.2t+1.8)\Big) \end{aligned} \tag{35}$$

Using the aforementioned settings, the two IRL algorithms yield the controllers in Equation 36, with very small differences to the model-based controller for the faulty system, where the off-policy version yields a controller which is slightly closer to K_{are}^f than the on-policy controller. The resulting performance levels of each controller are $\gamma_{\text{off}} = 1.02$ and $\gamma_{\text{on}} = 0.994$ respectively. The convergence of the controllers for the two algorithms is seen in Figure 6 and the state

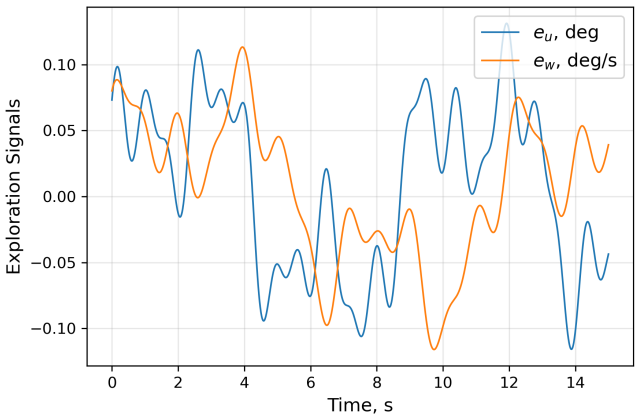


Figure 5: *Excitation signals e_u and e_w used during IRL process.*

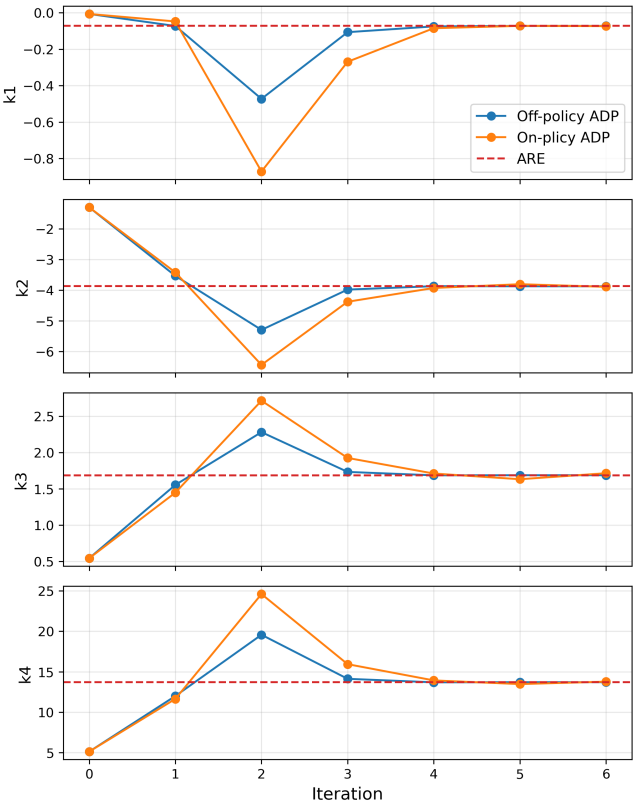


Figure 6: *Convergence of the controllers from the off-policy and on-policy algorithms to the analytic ARE-based solution.*

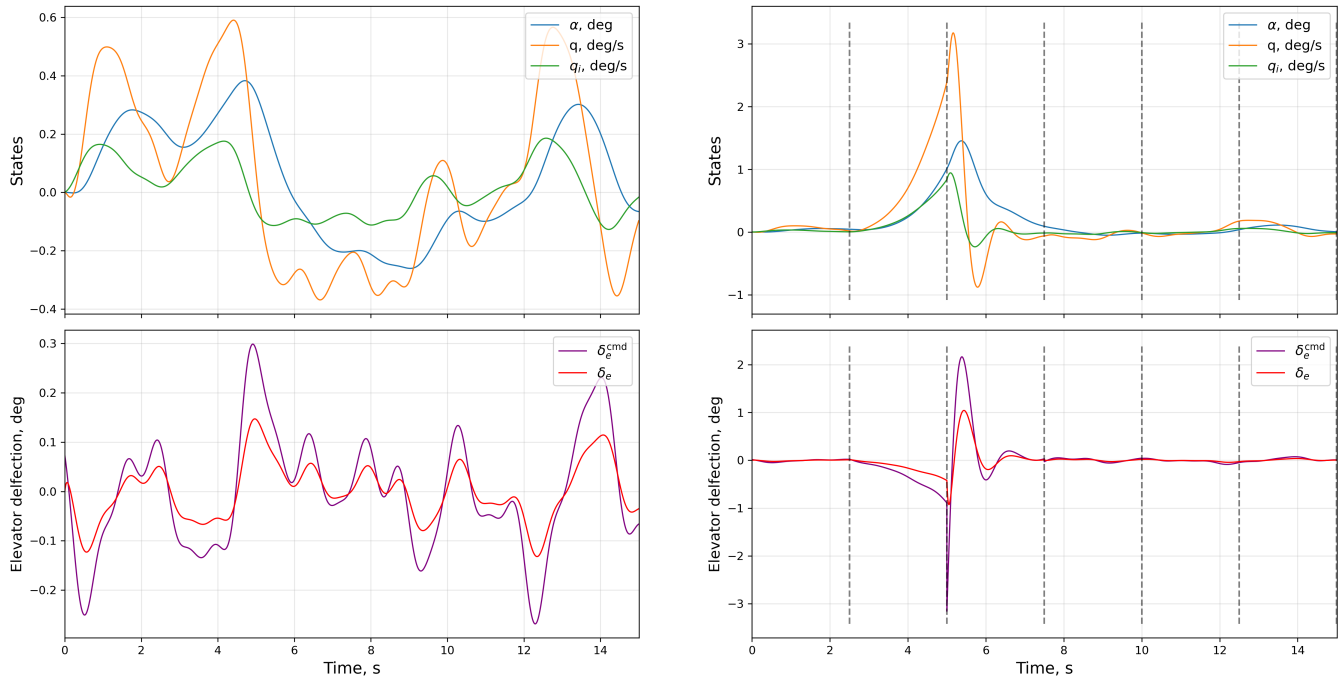
trajectories during learning in Figure 7.

$$\begin{aligned} K_{\text{off}} &= \begin{bmatrix} -0.07252 & -3.878 & 1.687 & 13.71 \end{bmatrix}\\ K_{\text{on}} &= \begin{bmatrix} -0.07374 & -3.897 & 1.715 & 13.77 \end{bmatrix}\\ ||K_{\text{off}} - K_{\text{are}}^f|| &= 0.00897\\ ||K_{\text{on}} - K_{\text{are}}^f|| &= 0.0852 \end{aligned} \tag{36}$$

From Figure 6, each controller has similar convergence properties, requiring 5 to 6 iterations of the algorithm to learn the controller. The first few iterations seem to overshoot the analytic solution; for the on-policy version, this is evident in Figure 7b as a large spike in the state trajectories. However, with further iterations, the controller does converge to the analytic ARE solution. One of the main differences in the learning process between the off-policy and on-policy IRL is seen in Figure 7, where the state trajectories remain relatively small for off-policy learning, while for the on-policy version, they diverge during one of the iterations. This demonstrates the previously described trade-off between safety and speed of adaptation, as the excitation signal needs to be exploratory enough during this time window for efficient adaptation of the controller, with the detriment that the states become large for some seconds. This does not occur during off-policy learning, as a single stabilising control policy is used to collect data, and the adaptation steps occur at the end of a large data collection step.

Plotting the movement of the system's closed-loop poles (seen in Figure 8) shows that during learning, the controller does not always stabilise the system, as after the first iteration, there is a pole on the right-hand plane. For off-policy learning, this destabilising controller does not affect safety during learning, as the controller is iterated upon until convergence to the analytic solution before it is used. Conversely, for on-policy learning, this destabilising controller can pose a safety threat as the state will start to diverge (as seen in Figure 7b). For this reason, the measurement window should be kept relatively low to not allow the state to diverge too aggressively, while also allowing the data to be informative enough for the adaptation to occur.

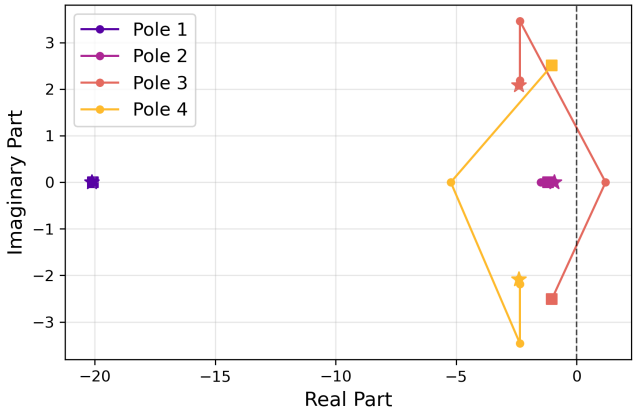
With the $\mathcal{H}_\infty$ state-feedback tracking controllers learned through the IRL methods, their tracking performance can be evaluated and compared to the analytic solution found through the ARE for the faulty system (K_{are}^f), and also to the controller for the nominal system (K_{are}). The response of the system to a tracking task for the various controllers can be seen in Figure 9. Due to the reduction in elevator actuator effectiveness and consequent reduction in the demanded tracking performance by altering Q, the rise time of the response is slowed down to $465\,\text{ms}$ for a $1\,\text{deg/s}$ pitch-rate command. The effect of reducing Q can be further seen in the response of the faulty system with the controller for the nominal system, wherein it achieves similar tracking performance to that of the nominal system, though requiring a much greater control effort, and violating the elevator deflection rate limits for this manoeuvre.



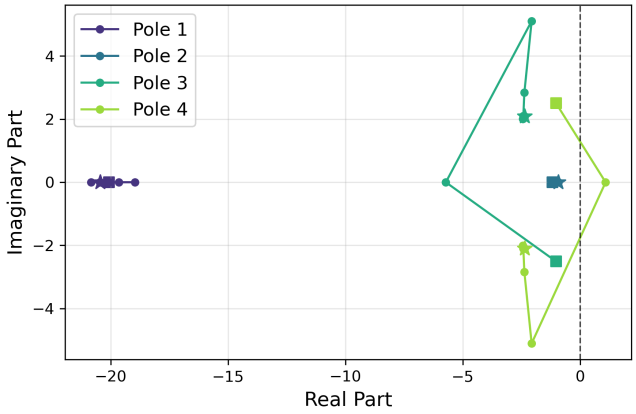
(a) *Off-policy IRL*

(b) *On-policy IRL*

Figure 7: *State trajectories during learning. For the off-policy version, this represents the single data collection step, whereas for the on-policy version, it represents the various data collection steps and subsequent controller update (separated by the vertical dashed lines).*

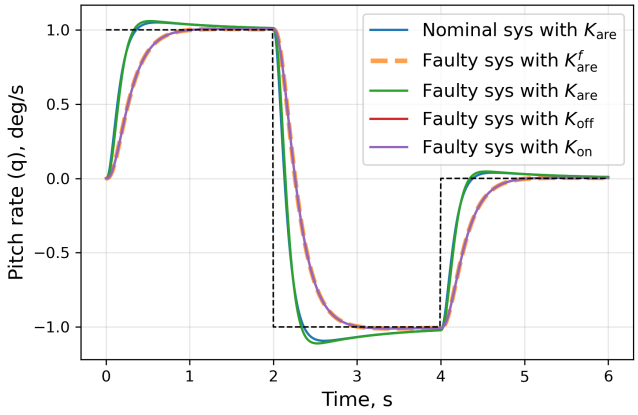


(a) *Off-policy IRL*

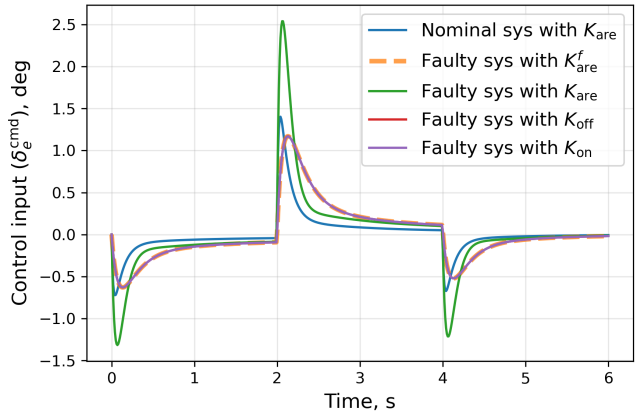


(b) *On-policy IRL*

Figure 8: *Movement of closed-loop poles during the learning processes. Iteration begins at the pole marked by a square and ends at the pole marked by a star.*



(a) *Pitch-rate response*



(b) *Control input*

Figure 9: *Pitch-rate reference tracking comparison of the analytic and learned controllers.*

One can also compare the closed-loop transfer functions $T_{wz}(s)$ of the systems with the different controllers, as seen in Figure 10. Since the ADP-learned controllers are quite similar to the analytic ARE-based controller for the faulty system, their closed-loop transfer functions are almost identical. Likewise, the performance levels of the nominal and faulty controllers should also be very similar, as the Q weights were altered for the faulty controller to account for this. For the faulty system using the nominal controller, this leads to a large performance level of $\gamma = 1.77$, as the demanded performance is too aggressive for the faulty system (which was also noted in Figure 9b).

5 Conclusion

In these results, integral reinforcement learning was applied to learn an online adaptation to an $\mathcal{H}_\infty$ tracking controller after a fault had occurred in the system, without any knowledge of the system dynamics. A continuous-time LTI model of an F-16's short-period dynamics was used for a pitch-rate tracking task, where a fault that reduced the elevator actuator effectiveness had occurred. Two model-free versions of the policy iteration learning method were applied, depending on when the control policy is updated (off-policy and on-policy), adapting the value function and the control and disturbance policies simultaneously. The learned state-feedback controller was shown to converge to the model-based controller, where the injection of excitation signals into the plant inputs allowed for the learning of the controller without access to the system matrices.

Although the IRL algorithms do not seem to require any information about the system model, some knowledge of the system is still required for the design of the excitation signals and for the choice of initial stabilising controller for the iterative scheme. Applying the online on-policy learning algorithm can also come at the expense of reduced safety, as the initially learned policies can be destabilising, and hence

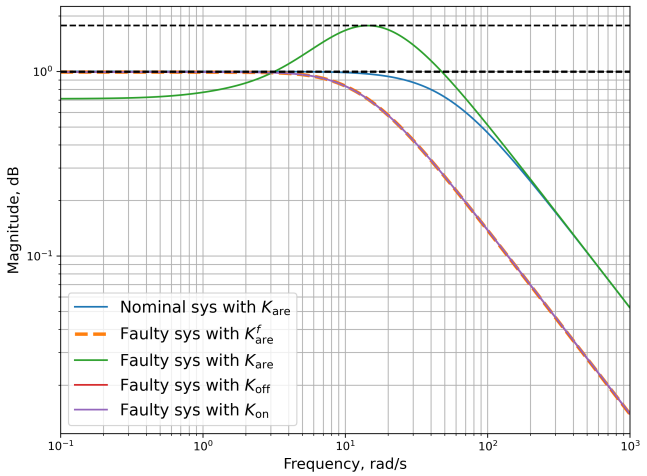


Figure 10: *Comparison of singular-value plots for the different closed-loop transfer functions $T_{wz}(s)$ using different controllers.*

the data collection windows between controller updates should be kept relatively small.

For future research into the use of ADP and reinforcement learning to find $\mathcal{H}_\infty$ controllers without access to the model, one could extend the methods described herein to nonlinear continuous-time systems, or to the coupled lateral and longitudinal dynamics of the F-16, as well as explore other control structures, namely output feedback or taking into account constrained/saturated inputs.

References

Ahmadi, P., Shahmansoorian, A., & Rahmani, M. (2025). Data-based H-infinity optimal tracking control of completely unknown linear systems under input constraints. *IET Control Theory & Applications*, *19*(1), e70022. https://doi.org/10.1049/cth2.70022

Al-Tamimi, A., Abu-Khalaf, M., & Lewis, F. L. (2007a). Adaptive critic designs for discrete-time zero-sum games with application to H-infinity control. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *37*(1), 240–247. https://doi.org/10.1109/TSMCB.2006.880135

Al-Tamimi, A., Lewis, F. L., & Abu-Khalaf, M. (2007b). Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control. *Automatica*, *43*(3), 473–481. https://doi.org/10.1016/j.automatica.2006.09.019

Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., & Bachem, O. (2021). What matters in on-policy reinforcement learning? a large-scale empirical study. *ICLR 2021: Ninth International Conference on Learning Representations*. https://doi.org/10.48550/arXiv.2006.05990

Başar, T., & Bernhard, P. (2008). *H-infinity optimal control and related minimax design problems*. Birkhäuser Boston. https://doi.org/10.1007/978-0-8176-4757-5

Benner, P., Byers, R., Mehrmann, V., & Xu, H. (2007). A robust numerical method for the gamma-iteration in h-infinity control. *Linear Algebra and its Applications*, *425*(2), 548–570. https://doi.org/10.1016/j.laa.2007.03.026

Buşoniu, L., De Schutter, B., & Babuška, R. (2010). Approximate dynamic programming and reinforcement learning. In R. Babuška & F. C. A. Groen (Eds.), *Interactive collaborative information systems* (pp. 3–44, Vol. 281). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-11688-9_1

de Alvear Cárdenas, J. I., Sun, B., & Van Kampen, E.-J. (2021). Intelligent adaptive control using LADP and IADP applied to f-16 aircraft with imperfect measurements. *AIAA Scitech 2021 Forum*. https://doi.org/10.2514/6.2021-1119

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation*, *12*(1), 219–245. https://doi.org/10.1162/089976600300015961

Fan, J., Li, Z., Jiang, Y., Chai, T., & Lewis, F. L. (2018). Model-free linear discrete-time system H-infinity control using input-output data. *2018 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 207–212. https://doi.org/10.1109/ICAMechS.2018.8506843

Hou, J., Wang, D., Liu, D., & Zhang, Y. (2020). Model-free H-infinity optimal tracking control of constrained nonlinear systems via an iterative adaptive learning algorithm. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*(11), 4097–4108. https://doi.org/10.1109/TSMC.2018.2863708

Jiang, Y., & Jiang, Z. P. (2012). Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. *Automatica*, *48*(10), 2699–2704. https://doi.org/10.1016/j.automatica.2012.06.096

Jiang, Z. P., & Jiang, Y. (2013). Robust adaptive dynamic programming for linear and nonlinear systems: An overview. *European Journal of Control*, *19*(5), 417–425. https://doi.org/10.1016/j.ejcon.2013.05.017

Kim, J. H., & Lewis, F. L. (2010). Model-free H-infinity control design for unknown linear discrete-time systems via Q-learning with LMI. *Automatica*, *46*(8), 1320–1326. https://doi.org/10.1016/j.automatica.2010.05.002

Kiumarsi, B., Modares, H., & Lewis, F. L. (2016). Optimal tracking control of uncertain systems: On-policy and off-policy reinforcement learning approaches. In K. G. Vamvoudakis & S. Jagannathan (Eds.), *Control of complex systems* (pp. 165–186). Butterworth-Heinemann. https://doi.org/10.1016/B978-0-12-805246-4.00005-7

Kiumarsi, B., Lewis, F. L., & Jiang, Z. P. (2017). H-infinity control of linear discrete-time systems: Off-policy reinforcement learning. *Automatica*, *78*, 144–152. https://doi.org/10.1016/j.automatica.2016.12.009

Kleinman, D. (1968). On an iterative technique for riccati equation computations. *IEEE Transactions on Automatic Control*, *13*(1), 114–115. https://doi.org/10.1109/TAC.1968.1098829

Lavretsky, E., & Wise, K. A. (2024). *Robust and adaptive control: With aerospace applications*. Springer International Publishing.

Lewis, F. L., & Vamvoudakis, K. G. (2011). Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *41*(1), 14–25. https://doi.org/10.1109/TSMCB.2010.2043839

Lewis, F. L., Vrabie, D., & Vamvoudakis, K. G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine*, *32*(6), 76–105. https://doi.org/10.1109/MCS.2012.2214134

Li, H., Liu, D., & Wang, D. (2014). Integral reinforcement learning for linear continuous-time zero-sum games with completely unknown dynamics. *IEEE Transactions on Automation Science and Engineering*, *11*(3), 706–714. https://doi.org/10.1109/TASE.2014.2300532

Liu, Y. Y., Wang, Z. S., & Shi, Z. (2020). H-infinity tracking control for linear discrete-time systems via reinforcement learning. *International Journal of Robust and Nonlinear Control*, *30*(1), 282–301. https://doi.org/10.1002/rnc.4762

Luo, B., Wu, H. N., & Huang, T. (2015). Off-policy reinforcement learning for H-infinity control design. *IEEE Transactions on Cybernetics*, *45*(1), 65–76. https://doi.org/10.1109/TCYB.2014.2319577

Magnus, J. R., & Neudecker, H. (2019). *Matrix differential calculus with applications in statistics and econometrics* (Third edition). John Wiley & Sons, Inc.

Modares, H., Lewis, F. L., & Jiang, Z. P. (2015). H-infinity tracking control of completely unknown continuous-time systems via off-policy reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(10), 2550–2562. https://doi.org/10.1109/TNNLS.2015.2441749

Moghadam, R., & Lewis, F. L. (2019). Output-feedback H-infinity quadratic tracking control of linear systems using reinforcement learning. *International Journal of Adaptive Control and Signal Processing*, *33*(2), 300–314. https://doi.org/10.1002/acs.2830

Morimoto, J., & Doya, K. (2005). Robust reinforcement learning. *Neural Computation*, *17*(2), 335–359. https://doi.org/10.1162/0899766053011528

Narendra, K. S., & Annaswamy, A. M. (1984). Persistent excitation in dynamical systems. *1984 American Control Conference*, 336–338. https://doi.org/10.23919/ACC.1984.4788399

Park, O., Shin, H., & Tsourdos, A. (2019). Linear quadratic tracker with integrator using integral reinforcement learning. *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, 31–36. https://doi.org/10.1109/REDUAS47371.2019.8999679

Perrusquía, A., & Yu, W. (2021). Continuous-time reinforcement learning for robust control under worst-case uncertainty. *International Journal of Systems Science*, *52*(4), 770–784. https://doi.org/10.1080/00207721.2020.1839142

Qin, C., Zhang, H., & Luo, Y. (2016). Model-free H-infinity control design for unknown continuous-time linear system using adaptive dynamic programming. *Asian Journal of Control*, *18*(2), 609–618. https://doi.org/10.1002/asjc.1102

Skogestad, S., & Postlethwaite, I. (2005). *Multivariable feedback control: Analysis and design*. John Wiley & Sons.

Stevens, B. L., Lewis, F. L., & Johnson, E. N. (2016). *Aircraft control and simulation: Dynamics, controls design, and autonomous systems* (Third edition). Wiley.

Valadbeigi, A. P., Sedigh, A. K., & Lewis, F. L. (2020). H-infinity static output-feedback control design for discrete-time systems using reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(2), 396–406. https://doi.org/10.1109/TNNLS.2019.2901889

Vamvoudakis, K. G. (2017). Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach. *Systems & Control Letters*, *100*, 14–20. https://doi.org/10.1016/j.sysconle.2016.12.003

Vrabie, D., & Lewis, F. (2011). Adaptive dynamic programming for online solution of a zero-sum differential game. *Journal of Control Theory and Applications*, *9*(3), 353–360. https://doi.org/10.1007/s11768-011-0166-4

Vrabie, D., Vamvoudakis, K. G., & Lewis, F. L. (2013). *Optimal adaptive control and differential games by reinforcement learning principles*. Institution of engineering; technology. https://doi.org/10.1049/PBCE081E

Werbos, P. J. (1992). Approximate dynamic programming for real-time control and neural modeling. In D. A. Sofge & D. A. White (Eds.), *Handbook of intelligent control* (pp. 493–525). Van Nostrand.

Wu, H. N., & Luo, B. (2012). Neural network based online simultaneous policy update algorithm for solving the HJI equation in nonlinear H-infinity control. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(12), 1884–1895. https://doi.org/10.1109/TNNLS.2012.2217349

Xiao, G., Zhang, H., Zhang, K., & Wen, Y. (2018). Value iteration based integral reinforcement learning approach for H-infinity controller design of continuous-time nonlinear systems. *Neurocomputing*, *285*, 51–59. https://doi.org/10.1016/j.neucom.2018.01.029

Yasini, S., Karimpour, A., Naghibi Sistani, M. B., & Modares, H. (2015). Online concurrent reinforcement learning algorithm to solve two-player zero-sum games for partially unknown nonlinear continuous-time systems. *International Journal of Adaptive Control and Signal Processing*, *29*(4), 473–493. https://doi.org/10.1002/acs.2485

Zhang, H., Cui, X., Luo, Y., & Jiang, H. (2018). Finite-horizon H-infinity tracking control for unknown nonlinear systems with saturating actuators. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(4), 1200–1212. https://doi.org/10.1109/TNNLS.2017.2669099

Zhong, X., He, H., Wang, D., & Ni, Z. (2018). Model-free adaptive control for unknown nonlinear zero-sum differential game. *IEEE Transactions on Cybernetics*, *48*(5), 1633–1646. https://doi.org/10.1109/TCYB.2017.2712617

Zhou, K., Doyle, J. C., & Glover, K. (1996). *Robust and optimal control* (Vol. 40). Prentice Hall.

Zhou, K., & Doyle, J. C. (1998). *Essentials of robust control* (Vol. 104). Prentice Hall.

Zhou, Y., Van Kampen, E.-J., & Chu, Q. P. (2018). Incremental approximate dynamic programming for nonlinear adaptive tracking control with partial observability. *Journal of Guidance, Control, and Dynamics*, *41*(12), 2554–2567. https://doi.org/10.2514/1.G003472

Zhu, Y., Zhao, D., & Li, X. (2017). Iterative adaptive dynamic programming for solving unknown nonlinear zero-sum game based on online data. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(3), 714–725. https://doi.org/10.1109/TNNLS.2016.2561300

Part III

Additional Results

5

Sensitivity Analysis of ADP Algorithms

Additional tests on the controller can aid to improve the performance of the learning algorithm during online learning. For this purpose, the parameters of the ADP learning algorithm employed in this research can be evaluated by performing a sensitivity analysis of the learning method to changes in these parameters. As such, not only should the performance of the algorithm be improved by this analysis, but also, further understanding of the ADP method can be gained.

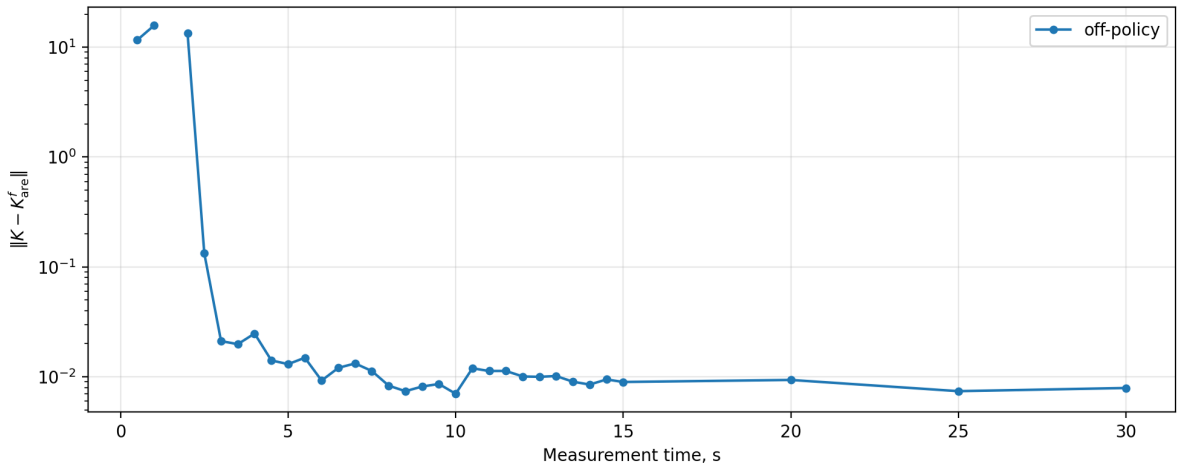
5.1. Sensitivity to measurement window size

The two main parameters that influence the iteration process of the IRL method are the measurement time T and the number of iterations n_{iter}. The measurement time refers to the time window in which system trajectory data is collected, which differs between on-policy and off-policy methods. For off-policy IRL, one measurement window of size T is used, followed by n_{iter} iterations, whereas for on-policy IRL, n_{iter} consecutive iterations are used, each with a different measurement time window of size T. As mentioned in Section 4.3.2, a trade-off between safety and speed of adaptation also encompasses the selection of the measurement window size. Too long a window incurs a slow adaptation, while too short a window may lead to not enough information being available to identify the controller from the data, which could result in an unsafe controller.

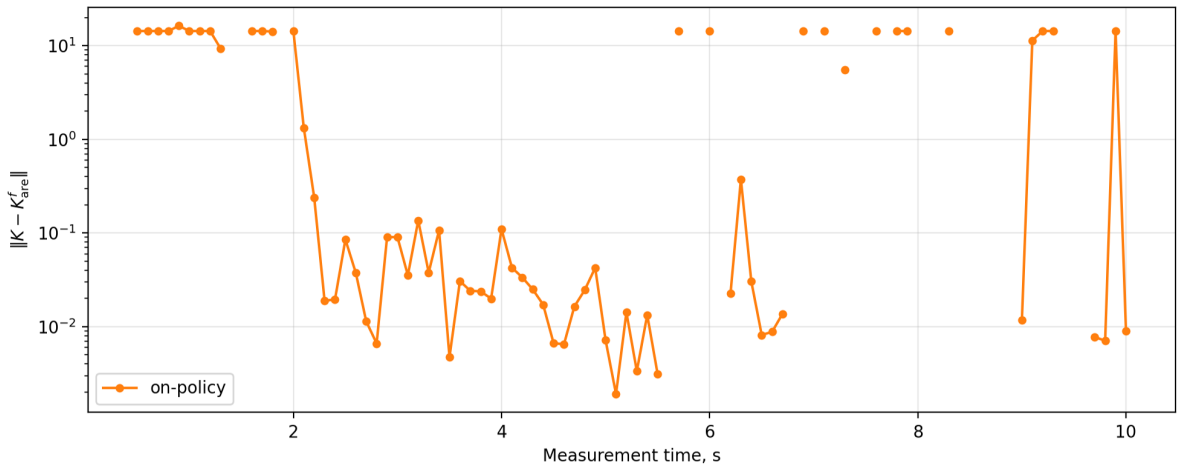
Referring to Table 4.1, the measurement window size used to adapt the IRL controller was $T = 15\,\text{s}$ and $T = 2.5\,\text{s}$ for off-policy and on-policy learning, respectively. For each policy iteration type, the measurement time was varied, and the convergence of the learned controller to the analytic model-based ARE controller was evaluated in terms of $||K_{\text{learned}} - K_{\text{are}}^{f}||$.

As seen in Figure 5.1, the measurement time does have a large effect on the accuracy of the convergence. For off-policy learning, very small measurement windows ($T < 3\,\text{s}$) lead to unsatisfactory convergence, as there is not enough information in the data to identify the controller (even though there are enough samples to solve the least-squares equations). As the measurement time increases, so does the accuracy of the learned controller. However, for much longer measurement times, the accuracy does not increase further, stagnating around the order of 10^{-3} to 10^{-2}, due to the effect of the excitation signals. Notably, a smaller measurement time of $10\,\text{s}$ is sufficient to provide better convergence accuracy.

Analysing the results for on-policy learning, it exhibits greater sensitivity to the size of the measurement window, as there is a greater consequence of learning destabilising controllers. If the measurement window is too small, there is also not enough information to iterate on the controller correctly (e.g., $T < 2\,\text{s}$), whereas if the measurement window is too large (e.g., $T > 7\,\text{s}$), then the state may diverge too far during learning for accurate iteration if a destabilising controller is found at some point of the iteration. From Figure 4.8, it can be seen that for the first iteration, the IRL algorithm will learn a desta-



(a) *Off-policy ADP*



(b) *On-policy ADP*

Figure 5.1: *Effect of measurement time T on convergence of controller. Instances where data is missing refer to where the controller failed to converge, due to a large divergence of the system states during learning. Enough iterations were used for each data point to allow for convergence to occur.*

bilising controller, and the state will start to diverge in the following measurement window. Therefore, if that window is too large, the controller will not converge to the model-based solution. The combination of the measurement window size and the excitation signals causes the on-policy IRL method to be more sensitive to changes in measurement time, though the range of $2.3\,\text{s}$ to $5.6\,\text{s}$ leads to accurate convergence.

5.2. Sensitivity to excitation signal selection

As mentioned in Section 4.3.2, the selection of the excitation signals can have a large bearing on the convergence of the learning process. The excitation must be selected such that there is sufficiently rich information in the data to identify the controller, but is not large enough to destabilise the system during learning.

For this purpose, several types of excitation signals were considered, namely three multisine signals (defined by Equation 5.1), a frequency sweep (defined by Equation 5.2), frequency sweep with noise and Gaussian noises with a slow ($\Delta t = 0.1\,\text{s}$) and fast ($\Delta t = 0.01\,\text{s}$) sampling rates. The following equations describe the excitation signals, and Figure 5.2 shows them during the total measurement

time window.

$$e_{\text{multisine}}(t) = \sum_i a_i \sin(\omega_i t + \phi_i) \tag{5.1}$$

$$e_{\text{sweep}}(t) = \cos\left(\omega_0 t + \frac{\omega_1 - \omega_0}{T} t^2\right) \tag{5.2}$$

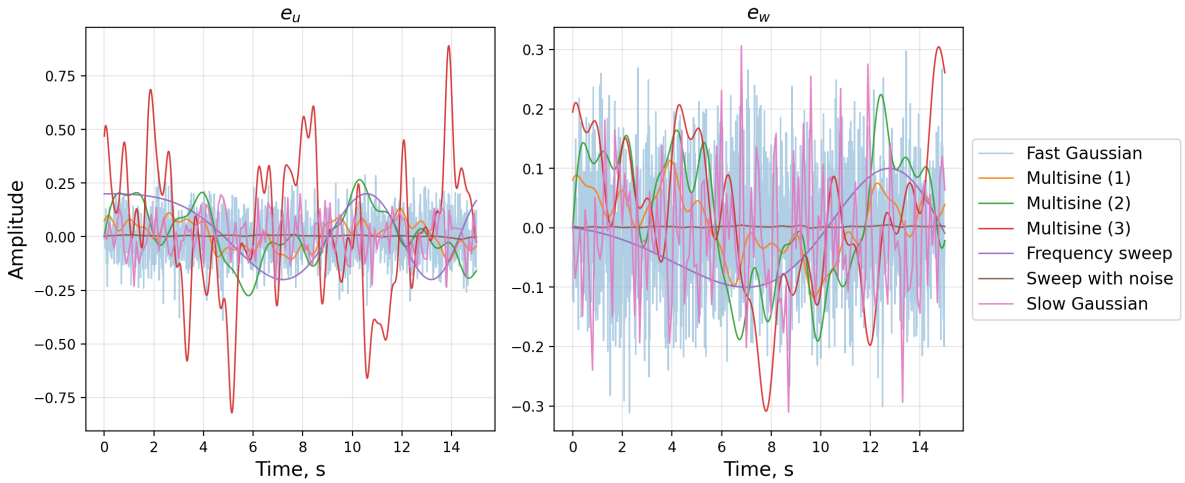


Figure 5.2: *Different excitation signals tested.*

Using these different excitation signals, the two IRL variants are applied to learn the $\mathcal{H}_\infty$ controller after a fault occurred, leading to the following convergence accuracies detailed in Table 5.1.

Table 5.1: *Effect of choice of excitation signal on controller convergence.*

Signal type	$\|\|K_{\text{off}} - K^f_{\text{are}}\|\|$	$\|\|K_{\text{on}} - K^f_{\text{are}}\|\|$
Multisine (Equation 4.35)	0.0089	0.085
Multisine (2)	0.0016	0.032
Multisine (3)	0.0053	0.0015
Frequency sweep	0.025	0.003
Sweep with noise	0.0071	0.13
Slow noise	0.048	0.01
Fast noise	unstable	unstable

As seen in Table 5.1, several classes of excitation signals can be used for the IRL algorithms, yielding good convergence to the model-based ARE solution. Of the solutions which converge, the worst is given by the sweep-with-noise excitation signal for on-policy learning. Even though there is a relatively larger difference between this learned controller and the analytic solution in 'gain-space', when considering the difference in pitch-rate tracking, the performance is almost identical (as shown in Figure 5.3), and such a difference in controller gain is acceptable.

However, one type of signal fails for the learning process, namely, the Gaussian noise with the same sampling time as the simulation. It should be expected that Gaussian noise causes the learning procedure to fail, as recalling Section 4.2, the $\mathcal{H}_\infty$ control problem only considers a class of disturbance signals with bounded energy. As Gaussian noise does not fall within the class of signals for which an $\mathcal{H}_\infty$ controller can attenuate, the system trajectory diverges during learning, and the controller fails to converge.

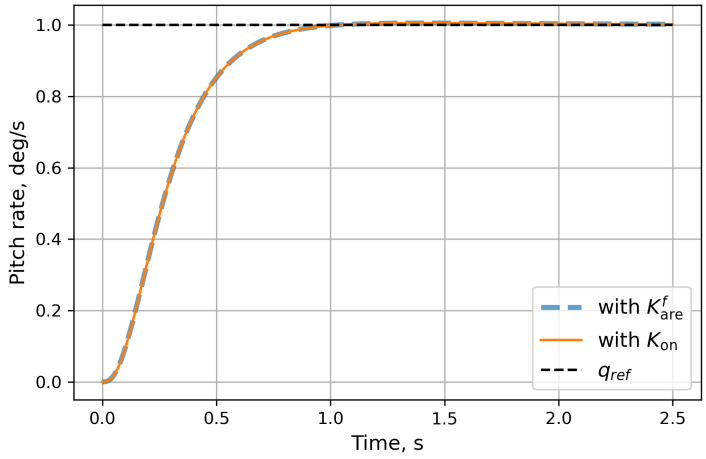


Figure 5.3: *Comparison of pitch-rate tracking response between the model-based ARE controller and the on-policy learned controller using the sweep-with-noise excitation signal.*

5.3. Sensitivity to fault magnitude

The type of fault also has a large effect on the accuracy of the learning process. In these results, only one type of fault is treated (reduced elevator actuator effectiveness), and hence, the severity of the fault can be altered by changing the magnitude of the reduction in effectiveness. As the fault becomes more severe, the controller for the faulty system differs more from the controller for the nominal system, which could have a negative effect on the learning process. Likewise, as the fault becomes increasingly severe, the nominal controller may no longer be stabilising for the faulty system, and cannot be used as an initial controller, causing the learning process to fail.

Figure 5.4 shows the convergence of the IRL controller for increasing fault severity, such that $B_u^f = (1-\rho)B_u$, where $\rho \in [0, 1]$ is a scaling factor on the elevator actuator effectiveness, representing the magnitude of the fault on the elevator. As the severity of the fault increases, the convergence accuracy to the respective ARE-based controller decreases until the fault becomes too severe and the learning process fails.

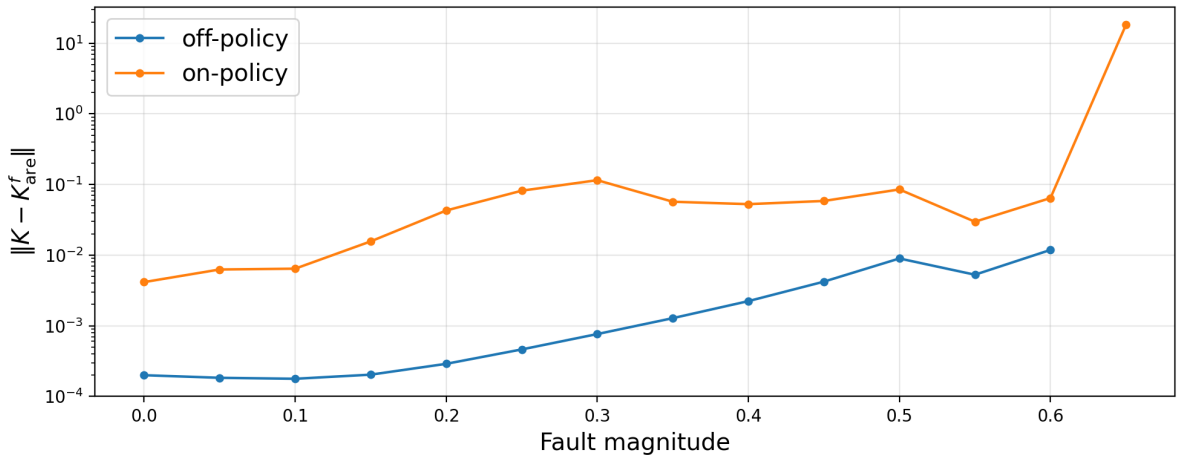


Figure 5.4: *Effect of fault magnitude on convergence of IRL controller.*

In fact, when the fault becomes severe enough, the closed-loop system becomes unstable using the model-based ARE faulty system controller with the scaled weights Q^f, highlighting the limitation that these weights are pre-specified prior to learning and are not learned online with the controller to adapt to the faulty system performance requirements.

6

Verification and Validation

6.1. Verification

To ensure that the ADP learning algorithm was implemented correctly, the convergence plot presented in Figure 4.6 is used as a reference. The ultimate goal of the IRL methods presented in Section 4.3 is to learn both the solution to the ARE associated with $\mathcal{H}_\infty$ control and the state-feedback controller from this solution. Therefore, to verify if the correct solution of the IRL algorithms, they can be compared to the solution of the ARE when solved using the system model. This was performed in Section 4.4, leading to very small differences between the learned and analytic controllers (in the order of 0.009 to 0.09). As such, the learning algorithms are considered to be correctly implemented.

6.2. Validation

To validate the implementation of the model-free $\mathcal{H}_\infty$ controller learning methods, the results can be compared to those published in the literature, namely considering the research of Li et al. (2014) and Qin et al. (2016), from which the main approach of this thesis stems.

In the results from Li et al. (2014), a linear model of a power system is used to study the on-policy IRL method, which is described in Equation 6.1. The goal of this IRL algorithm is not for tracking a signal, but regulation of the states to zero, unlike the results in Section 4.4. Nonetheless, the IRL approach to solving the ARE is the same, and performing validation on this control task should be applicable.

$$\begin{aligned}\dot{x} =& Ax + B_1 u + B_2 w \\ =& \begin{bmatrix} -0.0665 & 8 & 0 & 0 \\ 0 & -3.663 & 3.663 & 0 \\ -6.86 & 0 & -13.736 & -13.736 \\ 0.6 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 13.736 \\ 0 \end{bmatrix} u + \begin{bmatrix} -8 \\ 0 \\ 0 \\ 0 \end{bmatrix} w \end{aligned} \tag{6.1}$$

The following configurations are used in the paper: Q and R are identity matrices of appropriate dimension, $\gamma = 3.5$, $x_0 = [0.1, 0.2, 0.2, 0.1]^\top$, $T_{meas} = 0.5\,\mathrm{s}$ and small zero-mean Gaussian noises are used as the excitation signals. Using the system dynamics and specified Q, R and γ, the ARE is solved to yield the following state-feedback controller

$$K_{\text{are}} = \begin{bmatrix} 1.894 & 3.240 & 0.9563 & 1.313 \end{bmatrix}$$

Applying the on-policy learning algorithm (described in Section 4.3) with initial stabilising controllers selected as $K_0 = [0, 0, 0, 0]$ and $L_0 = [0, 0, 0, 0]$, yields the following controller,

$$K_{\text{IRL}} = \begin{bmatrix} 1.894 & 3.242 & 0.9596 & 1.317 \end{bmatrix}$$

where $\|K_{\text{IRL}} - K_{\text{are}}\| = 0.00641$. As can also be seen from Figure 6.1, the IRL method adapts the controller to the ARE-based solution, a result which is also provided by Li et al. (2014), thus validating the approach.

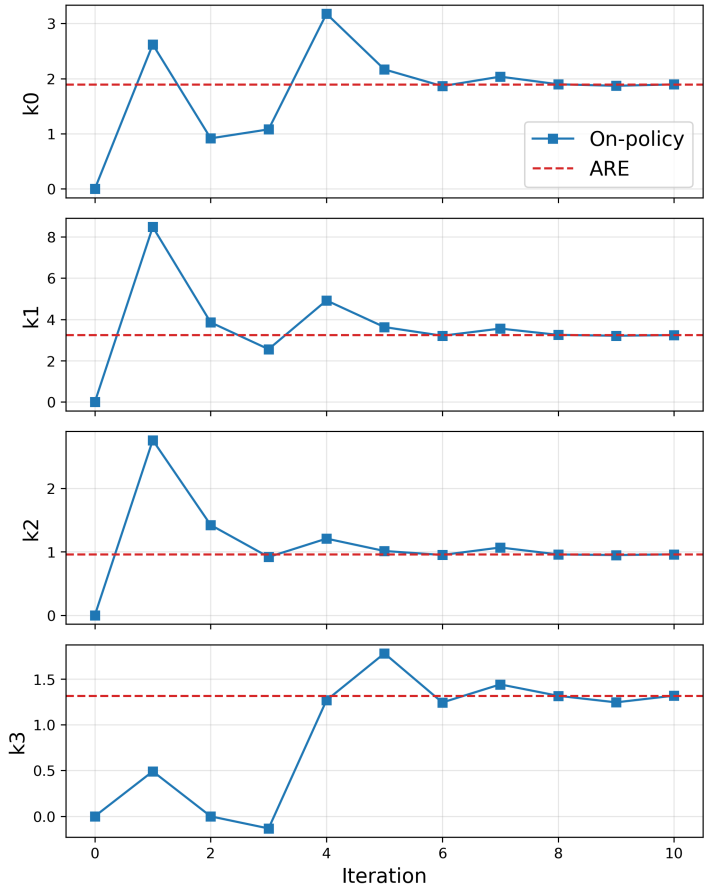


Figure 6.1: *Convergence of on-policy controller to the analytic ARE solution for the system given in Equation 6.1 (Li et al., 2014).*

It is important to note that, as this power system is open-loop stable, when learning a regulation controller using IRL, it is sufficient to set the initial controllers K_0 and L_0 as zeros, as these stabilise the closed-loop system. Therefore, for an open-loop stable system, the $\mathcal{H}_\infty$ controller learning approach can not only adapt a nominal controller in the presence of faults or parameter deviations (as demonstrated in Section 4.4), but also fully learn the $\mathcal{H}_\infty$ controller without having to use the model to design an initial stabilising control policy.

6.2.1. Relation to learning an LQR controller

As mentioned in Section 4.3, two excitation signals were employed to apply the IRL algorithms without knowledge of the system dynamics. One excitation signal is applied to the control input (e_u), whereas the other one is applied to the disturbance signal (e_w). Further noted in Section 4.3.2 is the requirement that both excitation signals be present, so that the IRL algorithm can learn the equilibrium solution of the zero-sum game between the control and disturbance inputs and, hence, learn two controllers (K and L). The following equation is then integrated to serve as the basis for the IRL algorithms.

$$\dot{V}_i = -x^\top Q_{L,i} x + 2(u - u_i)^\top R K_{i+1} x + 2\gamma^{-2}(w - w_i)^\top L_{i+1} x \tag{6.2}$$

If instead, the excitation signal on the disturbance input is removed, and the learning process uses initial stabilising controllers of all zeros (as in the example used in Section 6.2), the time derivative of the value function is then defined by Equation 6.3. These conditions lead to $w_i = L_i x = 0$ (as each successive update L_{i+1} will also become all zeros) and $w = w_i + e_w = 0$.

$$\dot{V}_i = -x^\top (Q + K_i^\top R K_i) x + 2(u - u_i)^\top R K_{i+1} x \tag{6.3}$$

Integrating this time derivative on the interval $[t, t+T]$, and using the definition that $V_i = x^\top P_i x$, yields the following policy iteration algorithm, which leads to the solution to the state-feedback optimal (LQR) controller (Jiang & Jiang, 2012).

$$x(t+T)^\top P_i x(t+T) - x(t)^\top P_i x(t) = -\int_t^{t+T} x^\top (Q + K_i^\top R K_i) x \, d\tau + 2\int_t^{t+T} (u-u_i)^\top R K_{i+1} x \, d\tau \quad (6.4)$$

To show that this yields an LQR controller, one can begin from the HJI equation (described in Equation 6.5, which gives the state-feedback solution to the nonlinear $\mathcal{H}_\infty$ control problem (Ferreira et al., 2008). This equation is also equivalent to the Bellman equation shown in Equation 4.16.

$$x^\top Q x + u^\top R u - \gamma^2 w^\top w + \frac{\partial V}{\partial x}\dot{x} = 0 \quad (6.5)$$

With the condition that the disturbance input exploration signal is removed, the dependence of the HJI equation on the disturbance is also removed. For linear systems, the value function can be assumed to be positive semi-definite and defined as $V = x^\top P x$. Replacing this into the HJI equation yields the following HJB equation

$$x^\top Q x + u^\top R u + 2x^\top P(Ax + B_u u) = 0 \quad (6.6)$$

The optimal state-feedback control policy can be determined by minimising the HJB equation, which can be found explicitly as the following

$$u^* = \arg\min_u \left[x^\top Q x + u^\top R u + 2x^\top P(Ax + B_u u)\right] = -R^{-1} B_u^\top P x \quad (6.7)$$

From here, replacing the optimal control policy into the HJB equation, yields the Algebraic Riccati Equation associated with optimal control, whose solution P is used to calculate the LQR state-feedback controller.

$$A^\top P + PA + Q - PB_u R^{-1} B_u^\top P = 0 \quad (6.8)$$

Using the system given in Equation 6.1 as an example, with the same identity weighting matrices applied to it, the model-based LQR controller (solved using the associated ARE) is given as

$$K_{\text{lqr}} = \begin{bmatrix} 0.8267 & 1.701 & 0.7049 & 0.4142 \end{bmatrix}$$

Using the model-free IRL method described in Section 4.3, with initial stabilising controllers set to $K_0 = L_0 = [0, 0, 0, 0]$, e_u being a small zero-mean Gaussian noise and $e_w = 0$ (instead of also using an exploration signal for the disturbance input), leads to the following controller, which converged to the LQR controller.

$$K_{IRL} = \begin{bmatrix} 0.8267 & 1.699 & 0.7054 & 0.4141 \end{bmatrix} \qquad \|K_{\text{IRL}} - K_{\text{lqr}}\| = 0.00057$$

7

Tracking with Reference Dynamics

The integral reinforcement learning controller, which has been applied in the main results, provides tracking control by augmenting the system dynamics with an integrator of the error between the reference and the state that must track it. However, in ADP literature, it is more common to find tracking controllers which augment the state with certain reference dynamics, requiring the associated cost function (and consequently the disturbance attenuation condition) to be bounded by a discount factor (Modares et al., 2015; Park et al., 2019; Ahmadi et al., 2025). In this chapter, the differences between the current approach and the discounted-tracking approach are explored. In Section 7.1, the new augmented system is described, along with the new model-based solution to the discounted problem, while in Section 7.2, the results of learning a controller via the discounted-tracking approach are given, and compared to the main approach of this research

7.1. Augmented system and controller synthesis with ARE

Consider the following class of LTI continuous-time systems, with specified reference dynamics, where r is a reference to be tracked, defined by the command generator dynamics H.

$$\begin{aligned} \dot{x} &= Ax + B_u u + B_w w \\ \dot{r} &= Hr \end{aligned} \tag{7.1}$$

Define the error between the state and the reference $e = x - r$, and construct the augmented state consisting of the error and the reference signals $x_a = [e, r]$. Using this augmented state, the full dynamics of the augmented system are given in Equation 7.2.

$$\begin{aligned} \dot{x}_a = \begin{bmatrix} \dot{e} \\ \dot{r} \end{bmatrix} &= \begin{bmatrix} A & A-H \\ 0 & H \end{bmatrix} \begin{bmatrix} e \\ r \end{bmatrix} + \begin{bmatrix} B_u \\ 0 \end{bmatrix} u + \begin{bmatrix} B_w \\ 0 \end{bmatrix} w \\ &= \tilde{A}x_a + \tilde{B}_u u + \tilde{B}_w w \end{aligned} \tag{7.2}$$

To ensure that the performance/cost function associated with the disturbance attenuation condition in $\mathcal{H}_\infty$ control remains bounded in this control structure, an exponential discount factor $\alpha > 0$ is applied (Modares et al., 2015). This leads to the following disturbance attenuation condition (different from the typical one associated with $\mathcal{H}_\infty$ control)

$$\frac{\int_t^\infty e^{-\alpha(t-\tau)} \left\|z(\tau)\right\|^2}{\int_t^\infty e^{-\alpha(t-\tau)} \left\|w(\tau)\right\|^2} \leq \gamma^2 \tag{7.3}$$

where $\left\|z\right\|^2 = e^T Q e + u^T R u$, and also to the cost function derived from it

$$J(x_a, u, w) = \int_t^\infty e^{-\alpha(t-\tau)} \Big(x_a^T Q_a x_a + u^T R u - \gamma^t w^T w \Big) d\tau \tag{7.4}$$

$$Q_a = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} \tag{7.5}$$

Similarly to the main results, to find the $\mathcal{H}_\infty$ state-feedback controller which stabilises the system in Equation 7.1 and satisfies the disturbance attenuation condition in Equation 7.3 when the system dynamics are known, the solution to an ARE can be used. However, as there is now the addition of a bounding discount factor, the ARE inherits an additional term, as seen in Equation 7.6.

$$\tilde{A}^T P + P\tilde{A} + Q_a - P\tilde{B}_u R^{-1} \tilde{B}_u^T P + \gamma^{-2} P \tilde{B}_w \tilde{B}^T P - \alpha P = 0 \tag{7.6}$$

A non-zero discount factor is a necessary condition for this ARE to be solved, as for some reference dynamics, the augmented system matrix $\tilde{A}$ could have an unreachable subspace, making it uncontrollable, violating a necessary assumption for solving the ARE. Nonetheless, the value of α can be made arbitrarily small, leading to a very similar ARE solution to that with no discount factor.

7.2. IRL solution to $\mathcal{H}_\infty$ tracking controller with discount

Using the ARE in Equation 7.6 requires complete knowledge of the system dynamics. The integral reinforcement learning methods described in Section 4.3 can be used to remove the dependence on the system dynamics and learn the controller online. These can be readily applied to this augmented system, with small modifications to account for the additional discount factor. Recall Equation 4.21, which describes the policy iteration algorithm of IRL for $\mathcal{H}_\infty$ control. Using the discount factor to bound the cost function ultimately leads to the following policy update law (Modares et al., 2015), where $Q_{L,i} = Q_a - \gamma^2 L_i^T L_i + K_i^T R K_i$

$$\begin{aligned} e^{-\alpha(t-\tau)}\big(V(x_a(t+T) - V(x_a(t)\big) &= e^{-\alpha(t-\tau)}\big(x_a(t+T)^T P_i x_a(t+T) - x_a(t)^T P_i x_a(t)\big) \\ &= -\int_t^\infty e^{-\alpha(t-\tau)}\Big(x_a^T Q_{L,i} x_a\Big) d\tau + \\ &\quad 2\int_t^\infty e^{-\alpha(t-\tau)}\Big((u-u_i)^T R K_{i+1} x_a\Big) d\tau + \\ &\quad 2\gamma^2 \int_t^\infty e^{-\alpha(t-\tau)}\Big((w-w_i)^T L_{i+1} x_a\Big) d\tau \end{aligned} \tag{7.7}$$

In a similar fashion as Equations 4.26 and 4.27, a series of samples of the system trajectories can be collected, and a least-squares approach applied to iteratively solve for K_{i+1} and L_{i+1} until convergence to the ARE solution coming from Equation 7.6. As detailed in Section 4.3, the same conditions are required for convergence of the IRL controller to the analytic ARE solution, namely: using initial stabilising controllers K_0 and L_0; and the presence of excitation signals on the control and disturbance input (e_u and e_w) to ensure that the collected data is informative enough to identify the controllers. With this setup, both off-policy (Algorithm 2) and on-policy (Algorithm 3) learning methods can be applied to this discounted tracking problem, for the pitch-rate tracking of the linearised F16 short-period dynamics described in Section 4.4. The new augmented tracking system is defined in Equation 7.8, for constant reference dynamics $\dot{r} = 0$.

$$\begin{bmatrix} \dot{\alpha} \\ \dot{e}_q \\ \dot{\delta}_e \\ r_q \end{bmatrix} = \begin{bmatrix} -1.01887 & 0.90506 & -0.12318 & 0.90506 \\ 0.82225 & -1.07741 & -10.058 & -1.07741 \\ 0 & 0 & -20.2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ e_q \\ \delta_e \\ r_q \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 20.2 \\ 0 \end{bmatrix} u + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} w \tag{7.8}$$

A model-based controller can be found for the above system, by solving the ARE in Equation 7.6, considering a small discount factor $\alpha = 0.0001$, $Q_a = \mathbf{diag}(0, 20, 0, 0)$, $R = 1$ and $\gamma = 1$, leading to Equation 7.9. This is not the optimal $\mathcal{H}_\infty$ controller as the resulting γ is considerably smaller than 1, but it serves to illustrate the discounted tracking approach.

$$K_{\text{are}} = \begin{bmatrix} -0.1796 & -4.234 & 1.284 & 0.2381 \end{bmatrix} \tag{7.9}$$

To now apply the discounted-IRL algorithms to this system, it can be noted that the augmented system with the applied discount (wherein the system matrix becomes $A - 0.5\alpha\mathbb{I}_n$), is open-loop stable, and

therefore a null initial stabilising controller can be used to learn the state-feedback controller for both the nominal system and for a faulty system. Therefore, using $K_0 = \mathbf{0}$, $L_0 = \mathbf{0}$, $T_{\text{off}} = 25\,\text{s}$, $T_{\text{on}} = 2.5\,\text{s}$ and $n_{\text{iter}} = 10$, the following nominal controllers are learned. As seen from Equation 7.10, similar to the IRL approach taken in Section 4.4, the learned controllers are extremely similar to the model-based ARE controller. This could also be extended to adapting a controller after a fault occurs.

$$\begin{aligned} K_{\text{off}} &= \begin{bmatrix} -0.1797 & -4.233 & 1.286 & 0.2382 \end{bmatrix} & \|K_{\text{off}} - K_{\text{are}}\| &= 0.0014 \\ K_{\text{on}} &= \begin{bmatrix} -0.1795 & -4.235 & 1.273 & 0.2329 \end{bmatrix} & \|K_{\text{on}} - K_{\text{are}}\| &= 0.012 \end{aligned} \tag{7.10}$$

Another interesting result for this approach to learning a tracking controller comes from attempting to track a reference while learning using on-policy IRL. For the initial controller updates, tracking performance is unsatisfactory; however, as more updates are rolled out, the controller begins to converge to the model-based ARE controller and tracking performance improves, leading to the on-policy learned controller in Equation 7.10.

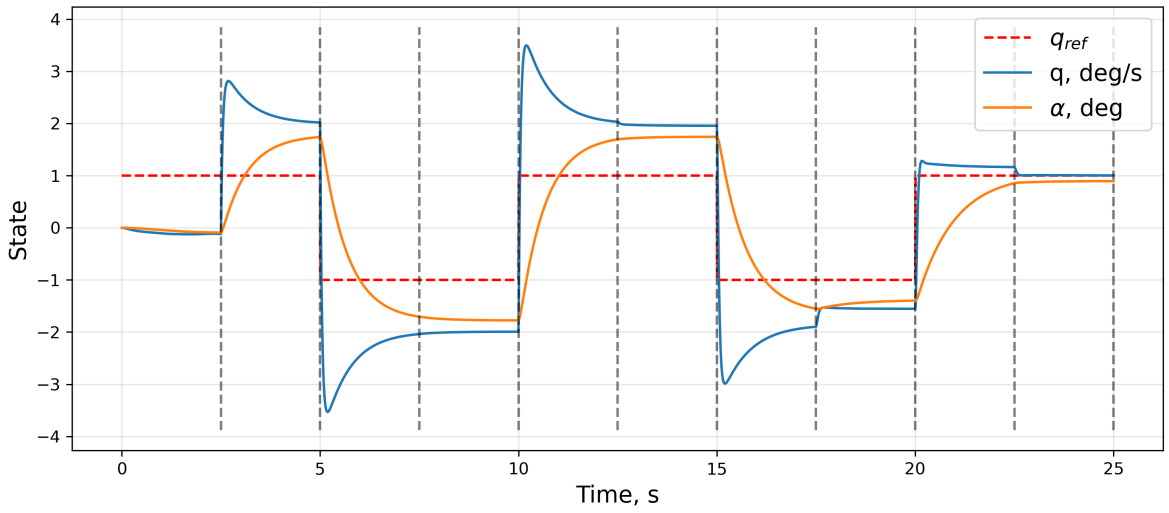


Figure 7.1: *On-policy discounted-IRL with pitch-rate reference tracking during the learning process. Policy updates are marked by the black dashed lines.*

Part IV

Closure

8

Conclusion

The aim of this research is to contribute to the development of flight control systems which combine the robust stability and performance of robust flight control laws with the online adaptability power of reinforcement learning. Such a robust reinforcement learning control system is leveraged to enable fault-tolerance of a flight controller, by adapting a robust controller online in response to a fault. This chapter reflects on the research questions posed at the start of this thesis, how the research objective has been met, and how it has contributed to the body of literature on robust reinforcement learning.

Research Question 1

RQ1 Which robust reinforcement learning (RRL) framework is best suited to maximise tracking performance of an aircraft under disturbances, uncertainty and actuator faults?

RQ1.1 Which RRL formulations (e.g. $\mathcal{H}_\infty$ as min-max problem, robust adversarial RL, stability bounded/certified RL) are best suited to capture the dynamics of a simple system and maintain robust stability?

RQ1.2 To what extent does the RRL controller maintain stable performance in the presence of large model parameter deviations for a simple system?

In Section 2.3, different frameworks of combining robust control and reinforcement learning are considered, namely using approximate dynamic programming (with an emphasis on a zero-sum game solution to the $\mathcal{H}_\infty$ control problem) to learn robust controllers, stability-bounded reinforcement learning policies, synergistic methods combining model-based robust control and model-free reinforcement learning, and reinforcement learning policies with robustness properties. From the research objective, it was determined that such a flight controller should be model-free and enable fault-tolerance through either offline or online methods. As such, greater focus is placed on methods applied to fully unknown system dynamics.

Moreover, Section 2.3 compares the two approaches of performing offline training to enable fault-tolerance and online adaptation to faults. RL-based methods are typically trained offline, as they have reduced stability guarantees during online learning, while ADP-based approaches are well-suited for online adaptation with high sample efficiency. From the series of methods detailed in Chapter 2, ADP was selected as the candidate framework for this research, as it not only provides guarantees of robustness (depending on the class of controller being learned) but also guarantees of convergence and stability during learning. This therefore provides an answer for *RQ1.1*

In Chapter 6, a regulation controller is synthesised using the ADP framework (following the work of Li et al. (2014) and Qin et al. (2016)), which is able to adapt online in the presence of a fault, stabilising the system. This preliminary analysis shows the power of the ADP framework to synthesise a robust adaptive controller, hence answering *RQ1.2*. Within Section 4.3, the ADP algorithm (using integral reinforcement learning) which is used to develop the flight controller is described, which answers **RQ1**.

Research Question 2

RQ2 How can the proposed RRL framework be integrated with the aircraft simulation model for fault-tolerant flight control?

RQ2.1 What type of control task and faults is the system subject to, and which control architecture should be designed for this?

With the goal of adapting an $\mathcal{H}_\infty$ controller online in the presence of a fault, a continuous-time LTI model of the F16 short-period dynamics was used. Even though this model is relatively simple, it is suitable as a benchmark for the applicability of the selected learning method. The applied ADP algorithm uses a state-feedback controller structure, adapting the controller after collecting state trajectory data during a time window. For this model, a pitch-rate tracking task was selected. The system dynamics were augmented with an integrator of the error between the pitch rate and the reference (as detailed in Section 4.4). After a controller is synthesised from the ARE, a fault to the elevator actuator effectiveness is applied, and the IRL method is applied to adapt the controller to the fault online. As such, *RQ2.1* and **RQ2** are answered.

Research Question 3

RQ3 What performance and robustness indicators are most appropriate for the RRL controller?

RQ3.1 How well does the RRL controller perform under simple manoeuvres (e.g. tracking accuracy)?

RQ3.2 How well does the controller adapt itself to faults or large model parameter changes?

Evaluating the learned controller on a $1\,\mathrm{deg/s}$ pitch-rate command tracking task (shown in Section 4.4), the responses of the analytic controller (solved with access to the faulty system dynamics using the ARE) and the learned controllers (for both off-policy and on-policy versions) were practically indistinguishable, pointing to the very high accuracy of the convergence to the analytic result during learning. In fact, the total difference (L1-norm) between the learned and analytic controller was only 0.009 and 0.09 for the off and on-policy algorithms, respectively, answering *RQ3.1*. These convergence properties demonstrate the ability of the IRL method to adapt a controller to a large change in the model, though, as shown in Chapter 5, when the fault is too intense, the states diverge before the learning algorithm is able to converge to an admissible controller, which provides an answer to *RQ3.2*. To analyse the robustness properties of the learned controller, its gain and phase margins can be found, yielding a gain margin of $+\infty$ and a phase margin of $80\,^\circ$ at the plant input, demonstrating good robustness margins.

Research Objective

To enable fault-tolerance through the development of a model-free flight control system for an aircraft, by investigating how robust control techniques can be coupled with reinforcement learning.

In conclusion, this research has demonstrated how robust control theory and reinforcement learning can be successfully combined to develop a model-free flight control framework capable of adapting to faults. Through the investigation of ADP learning methods, the work presented has shown that robust stability, disturbance rejection, and controller adaptability can be maintained without requiring a precise mathematical model of the aircraft dynamics, improving upon the difficulties which model-based robust controllers have in adapting to large changes in the model. Beyond the specific results obtained, this research contributes towards the broader advancement of safer, more resilient, and increasingly autonomous aerospace systems, where intelligent controllers capable of adapting online to failures and uncertainties will play a fundamental role in the future of aviation.

9

Recommendations

From the results presented in this research, a series of recommendations can be made for future research, to expand upon the body of work of robust reinforcement learning:

- One of the main assumptions which limits the application of this implementation of an IRL controller comes from the use of full state-feedback. Typically, not all states are available from measurements, and only the output of the system is given; hence, an output-feedback (OF) controller must be synthesised. While most ADP model-free methods to find $\mathcal{H}_\infty$ controllers online require full information of the system's internal states, there have been advancements for learning output-feedback controllers online, namely for optimal control problems (Lewis & Vamvoudakis, 2011; Na et al., 2017), and for $\mathcal{H}_\infty$ control problems for DT systems (Fan et al., 2018; Valadbeigi et al., 2020). For CT systems, Kartal et al. (2022) proposes a new solution to the model-free online $\mathcal{H}_\infty$ controller synthesis, which would be interesting to explore its application in future research.
- Another limitation of most IRL methods comes from the assumption of an initial stabilising controller (ISC) so that the learning process can converge to the analytic ARE solution to the $\mathcal{H}_\infty$ control problem. However, this ISC may not always be accessible, notably for fault-tolerant control, where the controller for the nominal system may be destabilising after a fault has occurred. Lamperski (2020) and Wang et al. (2022) proposed model-free ADP frameworks which do not require ISCs, though they are constructed for LQR optimal controllers, which could possibly be extended to $\mathcal{H}_\infty$ control. Aalipour and Khani (2023) provide a Q-learning approach which does not require an ISC, which could be extended to CT systems and for flight control applications.
- In general, an approach to expand upon the results of this research would be to apply the online IRL algorithms to more complex flight control problems, such a control augmentation system for altitude and attitude control of a jet aircraft that can adapt to faults, tracking control with constrained/saturating inputs (extending upon the results of Hou et al. (2020) or Ahmadi et al. (2025)) for the F16 lateral or coupled dynamics, or exploring other types of faults that could occur in these systems (e.g., icing, structural failures, or jammed actuators). Exploring how well these ADP learning methods scale to more complex control problems would be of interest.
- A limitation on the robustness properties of the ADP method applied herein is the use of static weights on the performance vector, such that there is no frequency shaping of the sensitivity functions. It would be of interest to apply frequency-dependent weighting functions to the $\mathcal{H}_\infty$ controller learning problem, perhaps considering data-driven $\mathcal{H}_\infty$ control in the frequency domain (Tabibian et al., 2025).
- As part of the procedure for learning an $\mathcal{H}_\infty$ controller after a fault occurred, the performance weights Q and R are adjusted for the faulty system to account for the reduced system performance, which may require information about the system dynamics to do so manually. This approach could be improved by learning the performance weights online to account for the fault (i.e. a co-design of the weights and controller, as in Pérez et al. (2022)) without knowledge of the system dynamics.

- An important aspect of fault-tolerant control is the use of fault detection techniques to be able to trigger adaptations to faults. These results assume that the moment the fault occurs is known *a priori*, and thus the integration of fault detection learning algorithms into the controller adaptation loop would be a natural extension.
- These results focus solely on a subset of robust reinforcement learning techniques, namely the use of approximate dynamic programming to learn $\mathcal{H}_\infty$ controllers online. As detailed in Section 2.3, there are many other promising approaches to learning robust control laws with reinforcement learning, such as using robust adversarial reinforcement learning (Pinto et al., 2017; Zhang et al., 2020a; Keivan et al., 2022) (though with possibly fewer convergence/stability guarantees) or using IQCs to ensure that learning remains stable and improve robust control laws online (Kretchmar et al., 2001; Qin et al., 2012).

References

Aalipour, A., & Khani, A. (2023). Data-Driven H-infinity Control with a Real-Time and Efficient Reinforcement Learning Algorithm: An Application to Autonomous Mobility-on-Demand Systems [arXiv preprint]. https://doi.org/10.48550/arXiv.2309.08880

Abdullah, M. A., Ren, H., Ammar, H. B., Milenkovic, V., Luo, R., Zhang, M., & Wang, J. (2019). Wasserstein Robust Reinforcement Learning [arXiv preprint]. https://doi.org/10.48550/arXiv.1907.13196

Ahmadi, P., Shahmansoorian, A., & Rahmani, M. (2025). Data-based H_∞ optimal tracking control of completely unknown linear systems under input constraints. *IET Control Theory & Applications*, *19*(1), e70022. https://doi.org/10.1049/cth2.70022

AlMahamid, F., & Grolinger, K. (2022). Reinforcement Learning Algorithms: An Overview and Classification. *IEEE canadian conference on electrical and computer engineering (CCECE)*, 1–7. https://doi.org/10.1109/CCECE53047.2021.9569056

Al-Tamimi, A., Lewis, F. L., & Abu-Khalaf, M. (2007a). Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control. *Automatica*, *43*(3), 473–481. https://doi.org/10.1016/j.automatica.2006.09.019

Al-Tamimi, A., Abu-Khalaf, M., & Lewis, F. L. (2007b). Adaptive Critic Designs for Discrete-Time Zero-Sum Games With Application to H∞ Control. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *37*(1), 240–247. https://doi.org/10.1109/TSMCB.2006.880135

Anderson, C. W., Young, P. M., Buehner, M. R., Knight, J. N., Bush, K. A., & Hittle, D. C. (2007). Robust Reinforcement Learning Control Using Integral Quadratic Constraints for Recurrent Neural Networks. *IEEE Transactions on Neural Networks*, *18*(4), 993–1002. https://doi.org/10.1109/TNN.2007.899520

Apkarian, P., & Noll, D. (2017). The H∞ Control Problem is Solved. *AerospaceLab Journal*, *Issue 13*. https://doi.org/10.12762/2017.AL13-01

Başar, T., & Bernhard, P. (2008). *H∞ Optimal Control and Related Minimax Design Problems*. Birkhäuser Boston. https://doi.org/10.1007/978-0-8176-4757-5

Bea, R. W. (1998). Successive Galerkin approximation algorithms for nonlinear optimal and robust control. *International Journal of Control*, *71*(5), 717–743. https://doi.org/10.1080/002071798221542

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym [arXiv preprint]. https://doi.org/10.48550/arXiv.1606.01540

Buşoniu, L., De Bruin, T., Tolić, D., Kober, J., & Palunko, I. (2018). Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*, *46*, 8–28. https://doi.org/10.1016/j.arcontrol.2018.09.005

Dally, K., & Kampen, E.-J. v. (2022). Soft Actor-Critic Deep Reinforcement Learning for Fault Tolerant Flight Control. *AIAA SCITECH 2022 Forum*, 2078. https://doi.org/10.2514/6.2022-2078

Deshpande, A. M., Minai, A. A., & Kumar, M. (2021). Robust Deep Reinforcement Learning for Quadcopter Control. *Modeling, Estimation and Control Conference MECC 2021*, *54*(20), 90–95. https://doi.org/10.1016/j.ifacol.2021.11.158

Dong, H., Ding, Z., & Zhang, S. (2020). *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Springer Singapore. https://doi.org/10.1007/978-981-15-4095-0

Donti, P. L., Roderick, M., Fazlyab, M., & Kolter, J. Z. (2021). Enforcing robust control guarantees within neural network policies [arXiv preprint]. https://doi.org/10.48550/arXiv.2011.08105

Ducard, G. J. (2009). *Fault-tolerant Flight Control and Guidance Systems*. Springer. https://doi.org/10.1007/978-1-84882-561-1

Fan, J., Li, Z., Jiang, Y., Chai, T., & Lewis, F. L. (2018). Model-Free Linear Discrete-Time System H∞ Control Using Input-Output Data. *2018 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 207–212. https://doi.org/10.1109/ICAMechS.2018.8506843

Ferreira, H. C., Rocha, P. H., & Sales, R. M. (2008). Nonlinear H∞ control and the Hamilton-Jacobi-Isaacs equation. *IFAC Proceedings Volumes*, *41*(2), 188–193. https://doi.org/10.3182/20080706-5-KR-1001.00032

Gahinet, P., & Apkarian, P. (1994). A linear matrix inequality approach to H∞ control. *International Journal of Robust and Nonlinear Control*, *4*(4), 421–448. https://doi.org/10.1002/rnc.4590040403

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., & Levine, S. (2019). Soft Actor-Critic Algorithms and Applications [arXiv preprint]. https://doi.org/10.48550/arXiv.1812.05905

Han, M., Tian, Y., Zhang, L., Wang, J., & Pan, W. (2020). H∞ Model-free Reinforcement Learning with Robust Stability Guarantee. *NeurIPS 2019 Workshop on Robot Learning: Control and Interaction in the Real World*. https://doi.org/10.48550/arXiv.1911.02875

Homola, M., Li, Y., & van Kampen, E.-J. (2025). Uncertainty-Driven Distributional Reinforcement Learning for Flight Control. *AIAA SciTech 2025 Forum*. https://doi.org/10.2514/6.2025-2793

Hou, J., Wang, D., Liu, D., & Zhang, Y. (2020). Model-Free H_∞ Optimal Tracking Control of Constrained Nonlinear Systems via an Iterative Adaptive Learning Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*(11), 4097–4108. https://doi.org/10.1109/TSMC.2018.2863708

Huang, J., & Lin, C.-F. (1995). Numerical approach to computing nonlinear H-infinity control laws. *Journal of Guidance, Control, and Dynamics*, *18*(5), 989–994. https://doi.org/10.2514/3.21495

Huang, S., Papernot, N., Goodfellow, I., Duan, Y., & Abbeel, P. (2017). Adversarial attacks on neural network policies [arXiv preprint]. https://doi.org/10.48550/arXiv.1702.02284

ICAO. (2025). *State of Global Aviation Safety* (tech. rep.). International Civial Aviation Organisation.

Iyengar, G. N. (2005). Robust Dynamic Programming. *Mathematics of Operations Research*, *30*(2), 257–280. https://doi.org/10.1287/moor.1040.0129

Jiang, Y., & Jiang, Z. P. (2012). Computational adaptive optimal control for continuous-time linear systems with completely unknown dynamics. *Automatica*, *48*(10), 2699–2704. https://doi.org/10.1016/j.automatica.2012.06.096

Jin, M., & Lavaei, J. (2018). Stability-certified reinforcement learning: A control-theoretic perspective. *IEEE Access*, *8*, 229086–229100. https://doi.org/10.48550/arXiv.1810.11505

Kartal, Y., Xue, W., Koru, A. T., Lewis, F. L., & Dogan, A. (2022). New Solution for H-Infinity Static Output-Feedback Control Using Integral Reinforcement Learning. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4221689

Keivan, D., Havens, A., Seiler, P., Dullerud, G., & Hu, B. (2022). Model-free μ synthesis via adversarial reinforcement learning. *American Control Conference 2022*, 3335–3341.

Khargonekar, P. P., & Dahleh, M. A. (2018). Advancing systems and control research in the era of ML and AI. *Annual Reviews in Control*, *45*, 1–4. https://doi.org/10.1016/j.arcontrol.2018.04.001

Kiumarsi, B., Lewis, F. L., & Jiang, Z.-P. (2017). H∞ control of linear discrete-time systems: Off-policy reinforcement learning. *Automatica*, *78*, 144–152. https://doi.org/10.1016/j.automatica.2016.12.009

Kretchmar, R. M., Young, P. M., Anderson, C. W., Hittle, D. C., Anderson, M. L., & Delnero, C. C. (2001). Robust reinforcement learning control with static and dynamic stability. *International Journal of Robust and Nonlinear Control*, *11*(15), 1469–1500. https://doi.org/10.1002/rnc.670

Lamperski, A. (2020). Computing stabilizing linear controllers via policy iteration. *59th IEEE Conference on Decision and Control (CDC)*, 1902–1907. https://doi.org/10.1109/CDC42340.2020.9304202

Lavretsky, E., & Wise, K. A. (2024). *Robust and adaptive control: With aerospace applications*. Springer International Publishing. https://doi.org/10.1007/978-3-031-38314-4

Lee, T., & Lee, D. (2025). Robust Deterministic Policy Gradient for Disturbance Attenuation and Its Application to Quadrotor Control [arXiv preprint]. https://doi.org/10.48550/arXiv.2502.21057

Lelkó, A., Németh, B., Fényes, D., & Gáspár, P. (2023). Integration of Robust Control with Reinforcement Learning for Safe Autonomous Vehicle Motion. *22nd IFAC World Congress*, *56*(2), 1101–1106.

Lelkó, A., Németh, B., Mihály, A., Sename, O., & Gáspár, P. (2024). Robust Reinforcement Learning-based Vehicle Control with Object Avoidance. *17th IFAC Symposium on Control of Transportation Systems CTS 2024*, *58*(10), 134–139. https://doi.org/10.1016/j.ifacol.2024.07.330

Lewis, F. L., & Vamvoudakis, K. G. (2011). Reinforcement learning for partially observable dynamic processes: Adaptive dynamic programming using measured output data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *41*(1), 14–25. https://doi.org/10.1109/TSMCB.2010.2043839

Lewis, F. L., Vrabie, D., & Vamvoudakis, K. G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine*, *32*(6), 76–105. https://doi.org/10.1109/MCS.2012.2214134

Li, H., Liu, D., & Wang, D. (2014). Integral reinforcement learning for linear continuous-time zero-sum games with completely unknown dynamics. *IEEE Transactions on Automation Science and Engineering*, *11*(3), 706–714. https://doi.org/10.1109/TASE.2014.2300532

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning [arXiv preprint]. https://doi.org/10.48550/arXiv.1509.02971

Lin, M., Zhao, B., & Liu, D. (2023). Policy gradient adaptive dynamic programming for nonlinear discrete-time zero-sum games with unknown dynamics. *Soft Computing*, *27*(9), 5781–5795. https://doi.org/10.1007/s00500-023-07817-6

Liu, Y. Y., Wang, Z. S., & Shi, Z. (2020). H∞ tracking control for linear discrete-time systems via reinforcement learning. *International Journal of Robust and Nonlinear Control*, *30*(1), 282–301. https://doi.org/10.1002/rnc.4762

Liu, Z., Hu, J., Zhai, P., Hu, K., Nian, L., & Zhang, L. (2025). A simulation framework for fixed-wing aircraft robust control via adversarial reinforcement learning. *2025 International Conference on Virtual Reality and Visualization (ICVRV)*, 983–988. https://doi.org/10.1109/ICVRV67992.2025.00171

Luo, B., Wu, H. N., & Huang, T. (2015). Off-Policy Reinforcement Learning for H∞ Control Design. *IEEE Transactions on Cybernetics*, *45*(1), 65–76. https://doi.org/10.1109/TCYB.2014.2319577

Marquis, D. J., Wilhelm, B., Muniraj, D., & Farhood, M. (2025). Adversarial reinforcement learning for robust control of fixed-wing aircraft under model uncertainty [arXiv preprint]. https://doi.org/10.48550/ARXIV.2510.16650

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Modares, H., Lewis, F. L., & Jiang, Z.-P. (2015). H∞ Tracking Control of Completely Unknown Continuous-Time Systems via Off-Policy Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(10), 2550–2562. https://doi.org/10.1109/TNNLS.2015.2441749

Moos, J., Hansel, K., Abdulsamad, H., Stark, S., Clever, D., & Peters, J. (2022). Robust Reinforcement Learning: A Review of Foundations and Recent Advances. *Machine Learning and Knowledge Extraction*, *4*(1), 276–315. https://doi.org/10.3390/make4010013

Morimoto, J., & Doya, K. (2005). Robust Reinforcement Learning. *Neural Computation*, *17*(2), 335–359. https://doi.org/10.1162/0899766053011528

Na, J., Herrmann, G., & Vamvoudakis, K. G. (2017). Adaptive optimal observer design via approximate dynamic programming. *2017 American Control Conference (ACC)*, 3288–3293. https://doi.org/10.23919/ACC.2017.7963454

Ngo, P. D., & Godtliebsen, F. (2020). Data-Driven Robust Control Using Reinforcement Learning. *Applied Sciences*, *12*(4), 2262. https://doi.org/10.3390/app12042262

Nilim, A., & El Ghaoui, L. (2005). Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Operations Research*, *53*(5), 780–798. https://doi.org/10.1287/opre.1050.0216

Pan, X., Seita, D., Gao, Y., & Canny, J. (2019). Risk Averse Robust Adversarial Reinforcement Learning. *International Conference on Robotics and Automation (ICRA) 2019*, 8522–8528. https://doi.org/10.48550/arXiv.1904.00511

Panaganti, K., & Kalathil, D. (2021). Robust Reinforcement Learning using Least Squares Policy Iteration with Provable Performance Guarantees. *38th International Conference on Machine Learning*, 511–520.

Park, O., Shin, H., & Tsourdos, A. (2019). Linear quadratic tracker with integrator using integral reinforcement learning. *Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, 31–36. https://doi.org/10.1109/REDUAS47371.2019.8999679

Pattanaik, A., Tang, Z., Liu, S., Bommannan, G., & Chowdhary, G. (2017). Robust Deep Reinforcement Learning with Adversarial Attacks. *AAMAS '18: 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2040–2042. https://doi.org/10.48550/arXiv.1712.03632

Pérez, C. A., Theodoulis, S., Sève, F., & Goerig, L. (2022). Automatic weighting filter tuning for robust flight control law design. *IFAC-PapersOnLine*, *55*(16), 400–405. https://doi.org/10.1016/j.ifacol.2022.09.057

Perrusquía, A., & Yu, W. (2021). Continuous-time reinforcement learning for robust control under worst-case uncertainty. *International Journal of Systems Science*, *52*(4), 770–784. https://doi.org/10.1080/00207721.2020.1839142

Pi, C.-H., Ye, W.-Y., & Cheng, S. (2021). Robust Quadrotor Control through Reinforcement Learning with Disturbance Compensation. *Applied Sciences*, *11*(7), 3257. https://doi.org/10.3390/app11073257

Pinto, L., Davidson, J., Sukthankar, R., & Gupta, A. (2017). Robust adversarial reinforcement learning. *ICML'17: 34th International conference on machine learning*, *70*, 2817–2826. https://doi.org/10.48550/arXiv.1703.02702

Qin, B., Li, P., Wang, X., & Wang, Z. (2012). Robust Reinforcement Learning Control and Its Application Based on IQC and PSO. *2012 Second International Conference on Intelligent System Design and Engineering Application*, 505–508. https://doi.org/10.1109/ISdea.2012.658

Qin, C., Zhang, H., & Luo, Y. (2016). Model-Free H∞ Control Design for Unknown Continuous-Time Linear System Using Adaptive Dynamic Programming. *Asian Journal of Control*, *18*(2), 609–618. https://doi.org/10.1002/asjc.1102

Schoon, D., & Theodoulis, S. (2025). Review of H∞ Static Output Feedback Controller Synthesis Methods for Fighter Aircraft Control. *AIAA SciTech 2025 Forum*. https://doi.org/10.2514/6.2025-2241

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms [arXiv preprint]. https://doi.org/10.48550/arXiv.1707.06347

Shukla, D., Lal, R., Hauptman, D., Keshmiri, S. S., Prabhakar, P., & Beckage, N. (2020). Flight Test Validation of a Safety-Critical Neural Network Based Longitudinal Controller for a Fixed-Wing UAS. *AIAA Aviation 2020 Forum*. https://doi.org/10.2514/6.2020-3093

Skogestad, S., & Postlethwaite, I. (2005). *Multivariable feedback control: Analysis and design*. John Wiley & Sons.

Sutton, R., & Barto, A. (1998). Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*, *9*(5), 1054–1054. https://doi.org/10.1109/TNN.1998.712192

Tabibian, S.-M., Ataei, M., & Koofigar, H.-R. (2025). Robust data driven control in frequency domain: A model-free approach. *ISA Transactions*, *164*, 1–13. https://doi.org/10.1016/j.isatra.2025.05.033

Tan, K. L., Esfandiari, Y., Lee, X. Y., Aakanksha, & Sarkar, S. (2020). Robustifying Reinforcement Learning Agents via Action Space Adversarial Training. *American Control Conference 2020*, 3959–3964. https://doi.org/10.48550/arXiv.2007.07176

Tessler, C., Efroni, Y., & Mannor, S. (2019). Action Robust Reinforcement Learning and Applications in Continuous Control. *International Conference on Machine Learning*, 6215–6224. https://doi.org/10.48550/arXiv.1901.09184

Valadbeigi, A. P., Sedigh, A. K., & Lewis, F. L. (2020). H∞ Static Output-Feedback Control Design for Discrete-Time Systems Using Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(2), 396–406. https://doi.org/10.1109/TNNLS.2019.2901889

Vamvoudakis, K. G. (2017). Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach. *Systems & Control Letters*, *100*, 14–20. https://doi.org/10.1016/j.sysconle.2016.12.003

Varga, B., Kulcsár, B., & Chehreghani, M. H. (2023). Deep Q-learning: A robust control approach. *International Journal of Robust and Nonlinear Control*, *33*(1), 526–544. https://doi.org/10.1002/rnc.6457

Vrabie, D., & Lewis, F. (2011). Adaptive dynamic programming for online solution of a zero-sum differential game. *Journal of Control Theory and Applications*, *9*(3), 353–360. https://doi.org/10.1007/s11768-011-0166-4

Wang, D., He, H., & Liu, D. (2017). Improving the critic learning for event-based nonlinear H_∞ control design. *IEEE Transactions on Cybernetics*, *47*(10), 3417–3428. https://doi.org/10.1109/TCYB.2017.2653800

Wang, Y., & Zou, S. (2021). Online Robust Reinforcement Learning with Model Uncertainty. *NIPS'21: 35th International Conference on Neural Information Processing Systems*, 7193–7206. https://doi.org/10.48550/arXiv.2109.14523

Wang, X., Deng, H., & Ye, X. (2022). Model-free nonlinear robust control design via online critic learning. *ISA Transactions*, *129*, 446–459. https://doi.org/10.1016/j.isatra.2021.12.017

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3), 279–292. https://doi.org/10.1007/BF00992698

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3), 229–256. https://doi.org/10.1007/BF00992696

Wu, H. N., & Luo, B. (2012). Neural network based online simultaneous policy update algorithm for solving the hji equation in nonlinear H_∞ control. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(12), 1884–1895. https://doi.org/10.1109/TNNLS.2012.2217349

Wu, K. C., & Litt, J. S. (2023). *Reinforcement Learning Approach to Flight Control Allocation with Distributed Electric Propulsion* (tech. rep. No. E-20165).

Yasini, S., Karimpour, A., Naghibi Sistani, M.-B., & Modares, H. (2015). Online concurrent reinforcement learning algorithm to solve two-player zero-sum games for partially unknown nonlinear continuous-time systems. *International Journal of Adaptive Control and Signal Processing*, *29*(4), 473–493. https://doi.org/10.1002/acs.2485

Zhang, H., Cui, X., Luo, Y., & Jiang, H. (2018). Finite-Horizon H∞ Tracking Control for Unknown Nonlinear Systems With Saturating Actuators. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(4), 1200–1212. https://doi.org/10.1109/TNNLS.2017.2669099

Zhang, K., Hu, B., & Basar, T. (2020a). On the Stability and Convergence of Robust Adversarial Reinforcement Learning: A Case Study on Linear Quadratic Systems. *Advances in Neural Information Processing Systems*, *33*, 22056–22068.

Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., & Hsieh, C. J. (2020b). Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations. *NeurIPS 2020: 34th Conference on Neural Information Processing Systems*, 21024–21037. https://doi.org/10.48550/arXiv.2003.08938

Zhang, J., Rivera, C. E. O., Tyni, K., & Nguyen, S. (2025a). AirPilot: Interpretable PPO-based DRL Auto-Tuned Nonlinear PID Drone Controller for Robust Autonomous Flights [arXiv preprint]. https://doi.org/10.48550/arXiv.2404.00204

Zhang, R., Zhang, D., & Mueller, M. W. (2025b). ProxFly: Robust Control for Close Proximity Quadcopter Flight via Residual Reinforcement Learning. *IEEE International Conference on Robotics and Automation (ICRA) 2025*, 13683–13689. https://doi.org/10.1109/ICRA55743.2025.11127714

Zhou, K., & Doyle, J. C. (1998). *Essentials of robust control* (Vol. 104). Prentice Hall.

Zhu, Y., Zhao, D., & Li, X. (2017). Iterative Adaptive Dynamic Programming for Solving Unknown Nonlinear Zero-Sum Game Based on Online Data. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(3), 714–725. https://doi.org/10.1109/TNNLS.2016.2561300

Zuo, Y., Liu, Y. T., & Ahmad, A. (2023). Autonomous Blimp Control via H-infinity Robust Deep Residual Reinforcement Learning [arXiv preprint]. https://doi.org/10.48550/arXiv.2303.13929

A

Proposed Planning

This appendix details the high-level timeline of the phases of the thesis, followed by the GANTT charts of the two research phases (Midterm and Green Light). The Finalisation phase entails the incorporation of final feedback given by the supervisors after the green-light has been given for the defence.

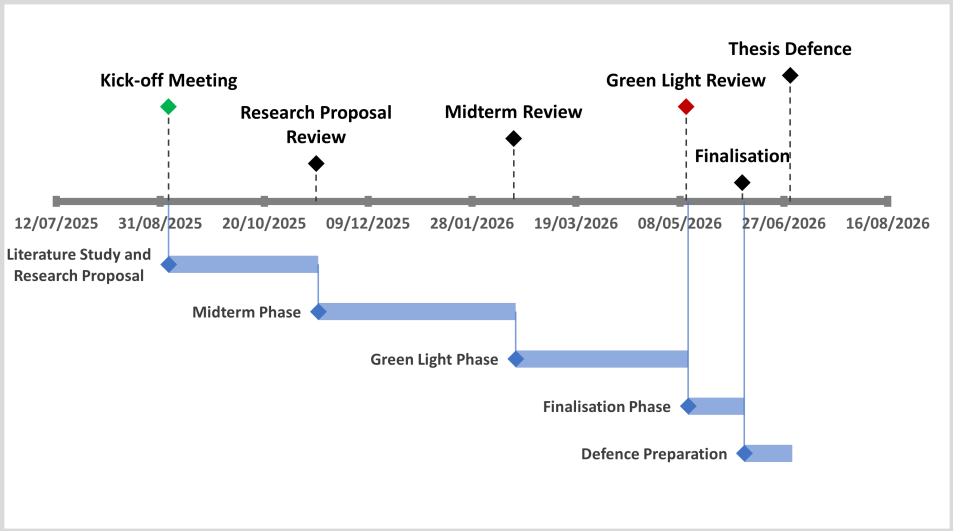


Figure A.1: *Overview of different phases and milestones of the Thesis project.*

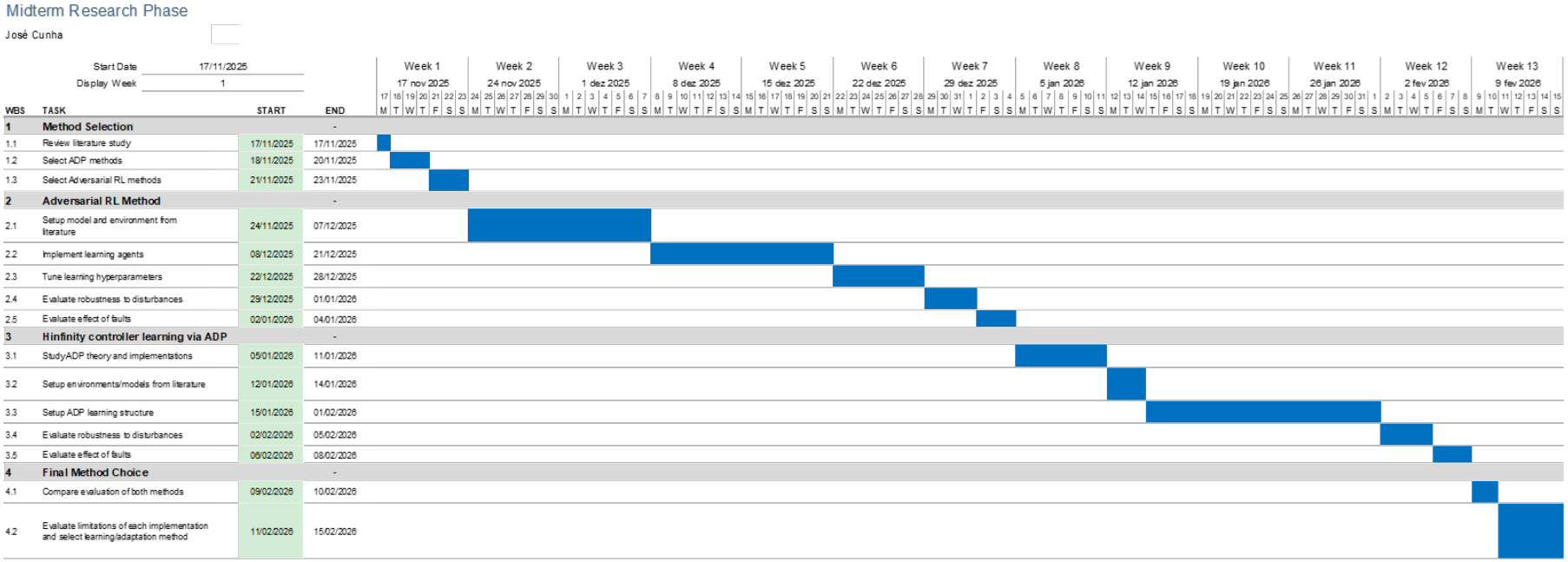


Figure A.2: *Midterm phase GANTT Chart*

Greenlight Research Phase

José Cunha

Start Date 16/02/2026

Display Week 1

WBS	TASK	START	END	DAYS
1	**Modelling**		-	
1.1	Select aircraft model	16/02/2026	17/02/2026	2
1.2	Select fault types	18/02/2026	19/02/2026	2
1.3	Select control problem	20/02/2026	22/02/2026	3
1.4	Finalise simulation architechture	16/02/2026	22/02/2026	7
2	**ADP Implementation**		-	
2.1	Implement ADP for regulation	23/02/2026	26/02/2026	4
2.2	Tune excitation signals and iteration parameters	27/02/2026	05/03/2026	7
2.3	Compare implementations of on and off policy learning	06/03/2026	08/03/2026	3
2.4	Adapt ADP learning to tracking control	09/03/2026	22/03/2026	14
3	**Adaptation to faults**		-	
3.1	ADP learning for adaptation to faults	23/03/2026	29/03/2026	7
3.2	Retune learning parameters for control task	30/03/2026	01/04/2026	3
4	**Controller evaluation**		-	
4.1	Evaluate nominal controller in terms of time-domain characteristics and robustness	02/04/2026	03/04/2026	2
4.2	Evaluate model-based faulty controller in terms of time-domain characteristics and robustness	04/04/2026	05/04/2026	2
4.3	Evaluate convergence of ADP method	04/04/2026	05/04/2026	2
4.4	Analyse robustness to disturbances and model uncertainties	04/04/2026	05/04/2026	2
4.5	Sensitivity analysis of different tuning parameters of ADP methdod	06/04/2026	12/04/2026	7
5	**Reporting**		-	
5.1	Draft report	06/04/2026	03/05/2026	2

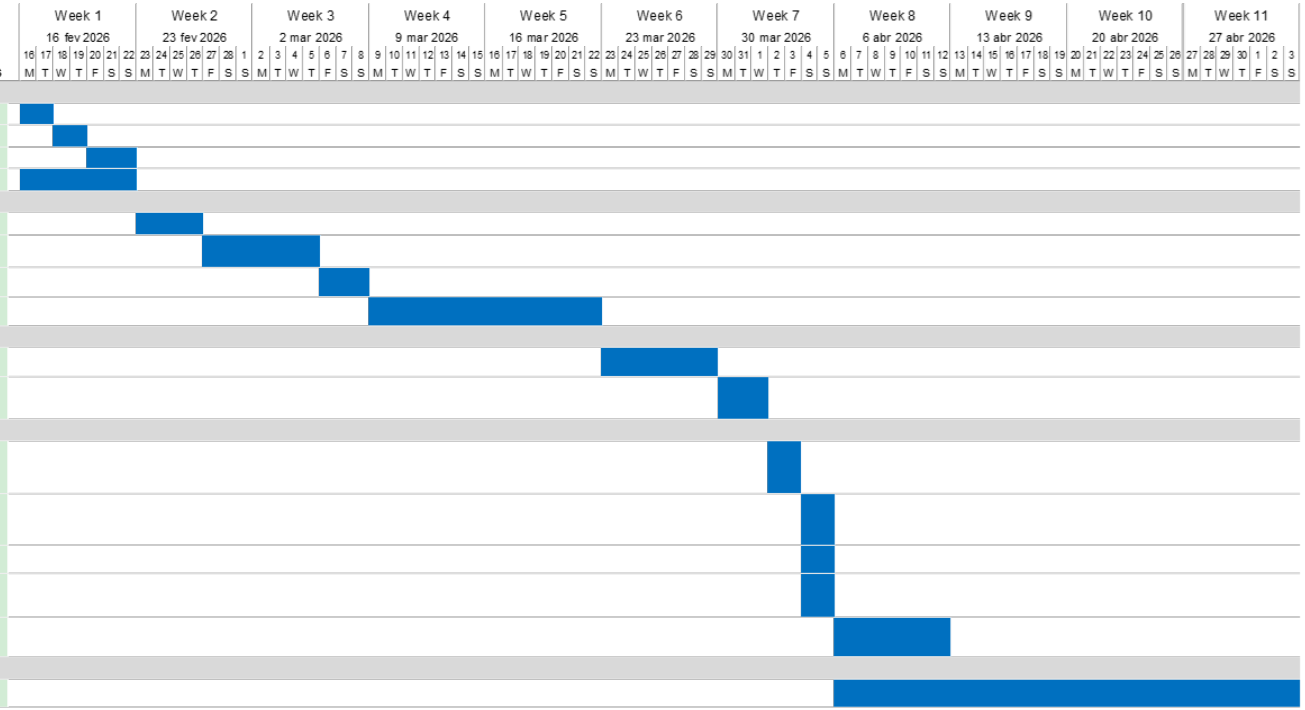


Figure A.3: *Green Light phase GANTT Chart*