Artificial Potential Fields for Safe Reinforcement Learning

in Flight Control Systems

S. Milosevic 4279816

Master of Science Thesis

TU Delft Delft University of Technology

Faculty of Aerospace Engineering

Artificial Potential Fields for Safe Reinforcement Learning
in Flight Control Systems

Master of Science Thesis

S. Milosevic 4279816

October 23, 2020

Faculty of Aerospace Engineering · Delft University of Technology


TU Delft
Delft
University of
Technology

Table of Contents

List of Figures

List of Tables

Preface

Automation has impacted the aviation industry in more than one way. Since the invention of the autopilot capable of steering the aircraft without any input, the strive for autonomy has made its way to almost all of the operations in the cockpit, as well as on the ground. The Flight Management System can calculate the route and fuel consumption with a few button presses. The Air Traffic Controllers need to perform a fraction of the calculations and communicate far less than their ancestors. Even the crew roster is processed automatically nowadays. The strive for autonomy continues.

This report documents a research project conducted as part of the Master Thesis in Aerospace Engineering. It is a survey of literature, methods and implementations of intelligent guidance control systems for aerospace applications. The motivation behind this project is in the ever-increasing autonomization in the aerospace industry. In order for the current trends to continue, safety precautions must be made to accommodate further implementations of autonomous systems, particularly in the flight control subspace. Advanced flight control systems have become a highly researched topic, with a primary focus on the safety of an aircraft and possible advantages of advanced systems over the conventional, gain-scheduled PID controllers. This project examines one such method of flight control, employing the use of Artificial Potential Field methods to supplement a Reinforcement Learning controller. The resulting algorithm provides safety restrictions, such that Reinforcement Learning can perform its exploratory tasks while ensuring a certain level of safety of the aircraft is guaranteed.

Chapter 1

Introduction

1-1 Motivation

Since the dawn of aerospace, automation was seen as an in-flight helping hand. The first gyroscopic autopilot was unveiled in 1914, but did little more than keep the aircraft level. Today's aircraft are capable of performing entire flight routes autonomously using the Flight Management System. The aerospace industry looks to be heading toward an increasingly more autonomous future, with the promise of reduced labor costs and less human-error incidents. So far automation has greatly benefited the industry, enabling flight crews to spend more time on auxiliary tasks such as route planning and ground communication, instead of continuously providing manual control to the aircraft. However, it is argued that automation can lead to degradation of skills required to perform the tasks being automated, resulting in a reduced effectiveness in times of emergencies [9, 10].

Making an airliner fully autonomous is currently relatively difficult, as the technology of automated guidance is not yet at a level of sophistication necessary to ensure uncompromised safety. Neither are the passengers psychologically comfortable enough to board a pilotless aircraft [11]. However, it is possible to apply autonomous control systems to smaller, *unmanned*, aircraft. In the past decade, Unmanned Aerial Vehicles (UAVs) have experienced rapid adoption by both the military and the commercial market, be it for professional or recreational use. With the growing number of airliners being produced, as well as the rise in the number of UAVs around the world, it is worth exploring the idea of a fully autonomous aerospace control system. Developing autonomous guidance systems for unmanned vehicles paves way for autonomy in other, more stringent aerospace applications, such as passenger flights.

Automating a UAV system, even without a crew and passengers increasing the safety design requirements, nonetheless presents challenges that need to be addressed. Safety is still considered a top priority; preventing the vehicle from entering a non-recoverable state needs to be implemented as a critical requirement. Avoiding obstacles which would lead to fatal outcomes needs to be achieved in real-time, in a robust and error-free manner. Machine Learning

techniques are proposed to be used in place of conventional controllers, enabling autonomous control without human input. Other advanced flight control methods used today are model-based, requiring an accurate aerodynamics model of an aircraft. An example of such a method is Nonlinear Dynamic Inversion [12], which works on the principle of cancelling non-linearities to create a linear or a partially linear system. While this advanced control method outperforms a PID gain-scheduled controller, which relies on linearizing the frequently-experienced flight conditions, it nevertheless works based on a fixed aircraft model, and cannot take into account uncertainties beyond the identified model. Machine Learning presents a solution to these limitations, since it is both robust and adaptable, and able to learn new dynamics and take them into account in future iterations.

This research project aims to improve safety and reliability of autonomous aerial vehicles, by combining existing techniques used in Machine Learning and Obstacle Avoidance. Both Machine Learning and Obstacle Avoidance are well researched topics, albeit seldom combined and used in aerospace systems. With the widespread and rapid adoption of Machine Learning in industries ranging from medicine to natural language processing, the technology will inevitably see aerospace applications. These aerospace applications are still bound within laboratory environments. The following research project has a general aim of furthering the technology, and bringing it closer to real-world feasibility.

1-2 Reinforcement Learning

Machine Learning (ML) emerged as a scientific field at a cross point of computer science, neuroscience, psychology and statistics, and aims to reproduce and mimic natural thinking through algorithmic programming [2]. ML is a method which can be used to address some of the challenges of complex control, beyond the predominantly used PID controllers [1].

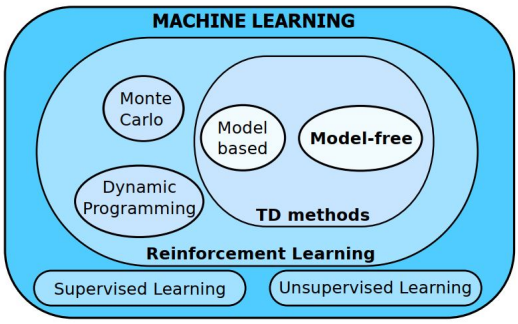


Figure 1-1: Machine Learning Classification [1]

Figure 1-1 shows the taxonomy of the prominent ML subtopics. Supervised learning relies on labelled training data provided by the operator, which is then used to extrapolate a response to unknown situations. While useful in statistical analysis, this learning method is not suitable for interactive environments, as the collected training data would need to be representative of all possible interactions. Unsupervised learning is a useful tool in finding structure and patterns in unlabelled and uncategorized data, but falls short beyond classifying data, the learning cannot be used to reinforce certain behaviour [2]. RL has its limitations too- but its reward-based learning process gives the algorithm flexibility, and the step-by-step nature of iteration, which is lacking in (un)supervised learning, is more suitable for real-time control systems. Further breakdown of various Machine Learning techniques is beyond the scope of this project, but in essence, Reinforcement Learning (RL) is the most suitable for a real-time unpredictable environment, as is the case in autonomous guidance.

RL has been a topic of research since the early days of computing, first being explicitly mentioned in 1965. While the computing approach Waltz and Fu [13] materialized half a century ago differs from today's state-of-the-art, the fundamental goal has remained the same - to design a control system capable of developing and improving its decision-making using reinforcement signals.

A RL problem typically consists of an agent inside a manipulable environment, and a reward system which "encourages" the agent to make the right decision- the agent is not explicitly told which action to take, rather, it is given a choice of actions, which it explores without additional user input, such that the total cumulative sum of rewards is maximized. This is the integral part of reinforcement learning- maximizing the collected rewards, while also exploring unknown parts of the environment.

General Properties of Reinforcement Learning

Reinforcement Learning algorithms can generally be described as *model-independent*, *autonomous*, and *adaptable* [1]. These 3 properties form the basis for the argument of using RL for guidance tasks.

The process is model-independent only in the so-called model-free RL algorithms (as opposed to the model-based ones). The model-free method can be applied in scenarios which lack the entire model of the environment or the system dynamics. This is of great importance for aerospace applications, as a complete model of the system is not always available, or is difficult to construct (for example, aerodynamic coupling or change in weight/centre of gravity).

The autonomy property derives from the reward system. A reward function is the only external input to a Reinforcement Learning problem, a user-defined function which the algorithm tries maximize. In order for the RL technique to be effective, the reward function must be well defined for the specific problem at hand, but once a satisfactory function is implemented the algorithm performs subsequent steps without any further input. The adaptability also derives from the reward function, as any change in the environment will be reflected by a change in the reward function. If in a simple environment there exists a goal state, and said goal state changes position, this change will directly change the reward function too, and the agent will adapt its previous strategy to the new scenario.

Challenges with Reinforcement Learning

The above-mentioned benefits present Reinforcement Learning methods as a feasible solution to the problem of autonomous guidance for aerospace applications. Nevertheless, RL methods present challenges that need be addressed.

The primary challenge faced by RL techniques is safety, as mentioned above the basis of RL is balancing exploration and maximizing collected rewards (or exploitation), and exploring unknown environments is inherently unsafe. This factor prevents a direct application to real-life use cases, as aircraft cannot learn from trial-and-error while in flight.

On the other hand, using simulated data presents a different challenge, namely dissimilarities between real and simulated data can result in wrong behaviors being learned by the agent. Controlled physical environments, such as a dedicated area in a lab setting, are seen as a

compromise between simulated and real-life learning environments. These controlled settings are equally inefficient in translating real-world learning scenarios, even more so as a human operator will usually be present to prevent fatal behavior, de-autonomising the learning in the process.

Another challenge arises in the efficiency of online learning. The aptly named *curse of dimensionality* [14] is an observation that computational time increases exponentially with an increase in the number of states in the environment. Online efficiency in an open world will therefore greatly suffer, as the number of states would make online learning too slow to execute actions in real-time.

For aerospace systems, the most crucial challenge is the one of safety, which in the standard form of RL is compromised due to the need to explore the environment, and perform trial-and-error decision-making to discover the optimal solution. Previous research has tried addressing this [4, 6, 15, 16], with positive results. Nevertheless, improvements can still be contributed, particularly for dynamic environments where adaptive strategies could result in better performance.

1-2-1 Artificial Potential Field Methods

Employing Artificial Potential Field (APF) methods in obstacle avoidance problems can be traced back in 1980s [17], but have since seen a steady interest in research, due to their applicability in guidance tasks in various environments. Most research has focused on guiding mobile robots and ship navigation, with a potential for aerospace applications recently being identified [5, 6].

Potential Field Methods model obstacles and objectives as virtual energy sources and sinks, respectively, which either emit or absorb said energy. This energy potential is used as information about the environment, and can direct the agent away from dangerous regions and toward the objective. Integrating this information within a Reinforcement Learning algorithm has shown improvements in convergence to the objective [7], and having an obstacle-repulsing ability could contribute to a safer learning environment for an RL agent. Furthermore, the APF construction simplicity, as is demonstrated in Section 2-3, results in a computationally inexpensive addition to RL, making it suitable for online learning.

1-3 Problem Statement & Research Questions

Given the above challenges, and their potential solutions, the goal of this research project is to combine the concepts of RL and APF Methods, such that the resulting implementation is safer than pure RL. Thus the most important Research Question follows, with intermediate steps listed as sub-questions below.

To which degree would using Potential Field Methods affect safety and performance of a Reinforcement Learning agent?

RQ1 What is the current state-of-the-art research in:

(a) Reinforcement Learning

(b) Artificial Potential Field Methods

RQ2 What are the feasible approaches of integrating Artificial Potential Field Methods with Reinforcement Learning?

(a) How is safety defined for an aerospace system?

(b) At which level of RL does applying APF Methods produce the most significant increase in safety?

(c) How can APF Methods address the issues of trial-and-error within RL?

RQ3 What are the measurable improvements in the safety of flight when using Potential Fields, relative to pure Reinforcement Learning?

(a) What is a real-life scenario which can be used in a simulation in order to validate the results?

(b) How is safety parametrized?

(c) How repeatable is the learning in different scenarios?

The experimental setup is to test at which point in the RL algorithm does applying the APF method result in the biggest overall safety improvement. The novelty of this research is in creating a real-time simulation of an autonomous aerospace controller, with validation provided by using an existing model of the General Dynamics F16 fighter jet. The literature is all-encompassing on these two concepts, RL and APF, but they have seldom been used together in the context of aerial vehicles. Previous research on this topic [6] focused on the discretized method for quadrotor control, which, although proved a working theory, can be expanded to show the algorithm's performance on different models.

1-4 Report Structure

A discussion of the findings of the research follows in preceding Sections. Part I summarizes the most relevant findings in a scientific article format. Methods and results of the applications of the methods are presented here.

Following the scientific article, Part II contains the literature survey in Chapter 2, integrations procedures of the algorithms in Chapter 3, and preliminary results of the algorithm application onto a simple model in Chapter 4. Part III contains additional results relevant to the flight controller model described in the scientific article, focusing on intra-episodic behaviour of the algorithm throughout the learning process. The report is closed off with conclusionary remarks in Part IV.

Part I

Scientific Article

Artificial Potential Fields for Safe Reinforcement Learning in Flight Control Systems

S. Milosevic*

Reinforcement Learning (RL) methods have become a topic of interest for performing guidance and navigation tasks, due to potential adaptability and autonomy improvements within dynamic systems. Nevertheless, a core component of RL is an agent exploring the environment it finds itself in, resulting in an intrinsic violation of the agent's safety. A subfield of Safe Reinforcement Learning (SRL) has emerged in an attempt to address the safety issues introduced by the agent's need to explore. Artificial Potential Field (APF) methods have also been studied extensively for the purposes of guidance and navigation, but seldom have the two methods, RL & APF, been examined when combined. This research focuses on integrating APF information within a RL controller for aerospace flight control applications. Different methods of integration are compared, by implementing APF information at varying levels of the RL algorithm- at the value function level, in the policy, and by exploration modification. Experiments show a decrease in collisions in the learning stage with a slight reduction in performance, relative to flat Reinforcement Learning.

I. Nomenclature

α_L	=	Learning rate
γ	=	Discount rate
ε	=	Exploration rate
$\rho(x)$	=	Euclidean distance from the agent to the goal
APF	=	Artificial Potential Field
DP	=	Dynamic programming
FSS	=	Fatal State Space
LTF	=	Lead-to-Fatal states
PID	=	Proportional-Integral-Derivative (controller)
QPF	=	Value (Q) function-based Artificial Potential Field
$Q(x, a)$	=	Action-value function
RL	=	Reinforcement Learning
RPF	=	Reward function-based Artificial Potential Field
SRL	=	Safe Reinforcement Learning
SSS	=	Safe State Space
TD	=	Temporal Difference
UAV	=	Unmanned Aerial Vehicle
$\mathbb{U}(x)$	=	Artificial Potential Field function

II. Introduction

Autonomous control systems have been an integral part in aviation industry's rise and prominence: the autopilot, the flight management computer, the crew scheduling algorithms, they all contribute to smoothly operating airliners the world has become accustomed to. With a newfound proliferation in the Unmanned Aerial Vehicles (UAV), autonomising flight control in its entirety has become a hot topic of research [1, 2].

In aircraft flight control, most current controllers operate using PID gain scheduling to conform to different operating scenarios. This gives an acceptable performance for an aircraft control system, but results in a suboptimal response when the aircraft dynamics are changed, and in conditions the controllers was not specifically designed for. Reinforcement

*Student, Control & Simulation, Delft University of Technology

Learning (RL) presents a potential solution for flight control autonomy; it is an adaptive method able to tackle the challenge of flight control, albeit with a few caveats. RL is reliant on exploration in order to determine the best set of actions to be taken in a particular scenario, resulting in decreased safety during the learning phase. Therefore, a way to increase safety during exploration was formulated through the use of Artificial Potential Field (APF) methods, a guidance and navigation tool that has seen extensive research [3–5] in the past few decades, but the combination of APF and RL is rarely explored. APF methods work on the principle of energy sources and sinks. A goal state is represented as an energy sink, while any obstacle is modelled as if it were emitting energy. The resulting energy field is then used to determine the best set of actions to be taken, through gradient descent. These methods have been previously applied to nagivation problems [6–8], particularly in maritime applications [3], but seldom in tandem with Reinforcement Learning. APF comes with its own pitfalls; namely local minima can prevent the agent from reaching the goal due to being stuck, a challenge which RL could tackle with its exploratory nature. This research will examine whether combining the theoretical approaches of APF and RL leads to a feasible solution for improving the safety of flight, relative to a pure Reinforcement Learning controller.

Related Work

As concepts, both RL and APF have been studied extensively for decades. RL has gained a newfound interest in the scientific community in the past decade, as computing power necessary for such calculations has only recently been achieved at scale. APF has been researched for maritime navigation purposes since 1980s [9] with limited commercial success, but it too has found its niche recently, the concept getting evolved over time to account for edge-case scenarios.

Safe Reinforcement Learning (SRL) has been categorized thoroughly by Garcia et al. [10], it has been defined as maximizing of expected returns in a RL problem while ensuring reasonable system performance and complying with safety constraints during learning and algorithm use. SRL is subdivided into criteria optimization and exploration modification. The agent's behaviour during the learning phase is dictated by a set of criteria which are being optimized-adding a safety criteria to it, in shape of risk-sensitive requirements, maximizing worst-case scenario returns, or constraining the available policy space. These strategies can be used to prevent the agent from entering fatal scenarios whilst learning. Exploration can also be influenced by providing the agent external knowledge through preloading the value matrix, letting a human operator perform the task while the agent observes, or adding fail-safe mechanisms which would pause learning if an unsafe situation is detected.

Increasing safety in learning has been a major focus in aerospace RL applications, due to RL's inherent lack of safety during exploration. In previous research, Hierarchical and Curriculum RL were applied to improve the efficiency of online learning through a subdivision of tasks [11, 12].

A relevant research to this project was presented by Xie et al. [13] in the RL probems with Artificial Potential Fields, wherein a virtual waterflow method was used to overcome the local minimum problem faced when using APF methods. The virtual waterflow method identifies whether the current state is a local minimum by comparing the potential energy to its surrounding states. If a local minimum is detected, the potential field energy of that state is increased to match the surrounding states, synonymous to water filling up in places where it cannot flow to a level of lower potential. This method helps overcome APF method's local minima limitation. APF methods have also been researched for remote piloting of UAVs [14], being utilized in a force feedback mechanism on the pilot's joystick, simulating control surfaces' forces and helping the pilots feel the sensations of flying more than if they would only have visual cues.

The contribution of this paper is to present an Artificial Potential Field framework, built onto a Reinforcement Learning controller for a safer autonomous exploration of the environment.

Fundamental concepts are introduced in Section III, elaborating core components of RL and APF. Integration of these concepts is presented in Section IV, combining RL and APF to create Safe RL algorithms. Preliminary results of a gridworld simulation are shown in Section V, followed by additional results and discussion of a flight controller application in Section VI. Lastly, the article is closed off with conclusive remarks in Section VII.

III. Fundamentals of Reinforcement Learning and Artificial Potential Field methods

In this section, a general framework for Reinforcement Learning (RL) and Artificial Potential Field (APF) methods is presented. RL is a Machine Learning concept which has proven to be useful in decision-making tasks involving an agent situated within a specified environment. APF is a theory based on modelling objects as field emitters and goals as field sinks, resulting in a state-space environment with energy contours, which can be used in guiding agents around obstacles in unknown scenarios.

Reinforcement Learning (RL) is a scientific field focused on decision-making tasks, which emerged as a multi-disciplinary idea, combining theories from computer science, psychology, neuroscience, and statistics. The core concept of RL is reproduction and mimicry of animal information and task processing abilities through algorithmic programming.

Safe Reinforcement Learning (SRL) is a subfield which emerged to address the lack of safety during RL exploration. It operates on the premise of defining safety constraints during the learning process, while maintaining adequate system performance. Various implementations have been explored in previous research, mainly split into exploration and optimality criterion modifications[10]. In this research, Artificial Potential Field information is to be integrated within a Reinforcement Learning's existing algorithm, such that the exploration will be restricted depending on the safety of the current and surrounding states.

A. Reinforcement Learning

The simplest form of a Reinforcement Learning problem comprises of an environment in which an agent resides, capable of taking actions based on the information it perceives from the environment [15]. This information comes in the form of a reward function, which the agent iterates through each time step. Over multiple episodes, the agent is able to learn which actions lead to the highest cumulative reward, creating the optimal path through the environment. An RL problem is defined as a Markov Decision Process (MDP)- meaning that any future event is only based on the current environment-agent conditions, and does not depend on past events and positions. An MDP is defined as a tuple $[\mathbb{X}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \gamma]$ consisting of the state space $\mathbb{X}$, action space $\mathbb{A}$, transition probability function $\mathbb{T}$, reward function $\mathbb{R}$ and the discount factor γ. The discount factor, $0 \leq \gamma \leq 1$, determines the future rewards' influence on the current action, formulating the discounted return G_t:

$$G_t = \sum_{t=0}^{\inf} \gamma^t \cdot R_t \tag{1}$$

A policy is a mapping of the states to actions that are executable at said state, $\pi(x) : \mathbb{X} \mapsto \mathbb{A}$. For a given state $X_t = x$, the control policy can be used in constructing the so-called value function:

$$V^{\pi}(x) = \mathbb{E}[G_t | X_t = x] = \mathbb{E}[\sum_{t=0}^{\inf} \gamma^t \cdot R_t | X_t = x] \tag{2}$$

which can be rewritten so that it can be used to iteratively evaluate the states and policies. The resulting equation is called the Bellman Equation:

$$V^{\pi}(x) = \mathbb{E}[R_t + \gamma V^{\pi}(X_{t+1}) | X_t = x] \tag{3}$$

Bellman optimality equations are found using the optimal policies, namely when the transitional probability $p(x', r|x, a)$ encompasses the weighted sum of probabilities of selecting an action $a \in \mathbb{A}$. The resulting equations are equivalent to the weighted sum of expectations of the expected values. The value function can be expressed as follows:

$$V^*(x) = \max_{a \in \mathbb{A}} \sum_{x', r} p(x', r|x, a) \left[r(x, a) + \gamma V^*(x') \right], \quad \texttt{for all } x \in \mathbb{X} \tag{4}$$

while the action-value function is shown in Equation 5.

$$Q^*(x, a) = \sum_{x', r} p(x', r|x, a) \left[r(x, a) + \gamma \max_{a \in \mathbb{A}} Q^*(x') \right], \quad \texttt{for all } x \in \mathbb{X}, a \in \mathbb{A} \tag{5}$$

While the value function $V^*(x)$ is only a function of the state, the action-value function $Q^*(x, a)$ takes into consideration the actions that were taken to get to said state.

The Bellman optimality equation is applied in its discretized form in Dynamic Programming (DP), first estimating the state values for the defined policy, then improving the policy itself. After the state value is estimated, the policy can be improved by acting greedily, selecting the action of highest return:

$$V_{k+1}(x) = \max_{a \in \mathbb{A}} \sum_{x'} p(x', r|x, a) \left[r(x, a) + \gamma V_k(x') \right] \tag{6}$$

Monte Carlo (MC) methods are analogous to DP, but rely on sample data to compute the average returns in episodic steps. MC methods, due to their sampling nature, can function even in incomplete environments, though their once-per-episode update rate limits their use cases. Combining the concepts of DP and MC methods enables Temporal Difference (TD) Learning, allowing the agent to estimate values based on previous estimates (a DP feature) in partially known environments (MC methods feature). Shown in Equation 7, the recursive TD learning equation is initialized arbitrarily, with a learning rate $0 < \alpha_L < 1$ determining the influence of the new estimate on the current one.

$$V(X_t) \leftarrow V(X_t) + \alpha_L[R_{t+1} + \gamma V(X_{t+1}) - V(X_t)] \tag{7}$$

The general structure of the algorithm is shown in Figure 1.

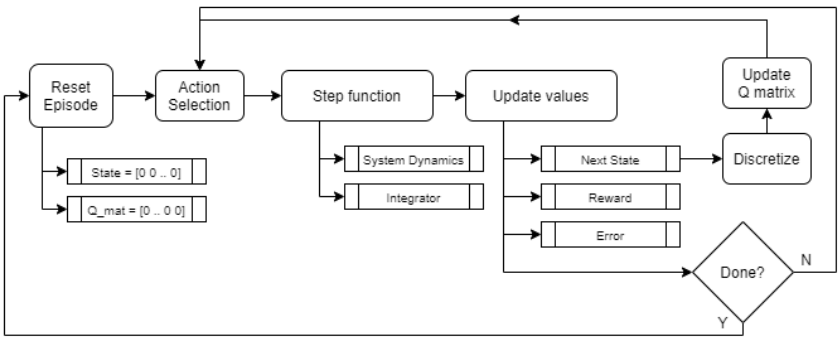


Fig. 1 Flow diagram for Temporal Difference Learning

An integral part of Reinforcement Learning is the exploration necessary for the agent to discover the environment around it. This is an inherently unsafe practice, since unknown states, while being explored, can lead to unsafe or unrecoverable states. Particularly in the aeropace domain, the agent needs to be guaranteed a certain level of safety for the algorithm to be feasible.

1. Safe Reinforcement Learning

Avoiding unsafe scenarios while learning is formalized as Safe Reinforcement Learning (SRL). Methods have been developed to limit the agent from entering states which compromise its safety. Two subgroups of SRL have emerged: optimization criterion modification (OCM) and exploration reduction. [10]

In OCM, the expected return is modified to take into consideration the safety of a set of trajectories followed during an episode. Worst-case criterion, or minimax, searches for a policy with maximum expected return for the worst possible outcome. For the maximum expected return of:

$$\max_{\pi \in \Pi} E_\pi(\mathbb{R}_t) = \max_{\pi \in \Pi} E_\pi(\sum_{t=0}^{\infty} \gamma^t r_t) \tag{8}$$

the worst-case criterion [16], following a trajectory $\omega \in \Omega^\pi$, would be:

$$\max_{\pi \in \Pi} \min_{\omega \in \Omega^\pi} E_{\pi,\omega}(\mathbb{R}_t) = \max_{\pi \in \Pi} \min_{\omega \in \Omega^\pi} E_{\pi,\omega}(\sum_{t=0}^{\infty} \gamma^t r_t) \tag{9}$$

Extending this concept to Q-learning, the minmax Q-learning algorithm [16] can be expressed as:

$$\hat{Q}(s_t, a_t) = \min(\hat{Q}(s_t, a_t), r_{t+1} + \gamma \max_{a_{t+1} \in A} \hat{Q}(s_{t+1}, a_{t+1})) \tag{10}$$

Constrained criterion is similarly an OCM approach, but rather than modifying the expected return, the MDP is expanded to $[\mathbb{X}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \gamma, \mathbb{C}]$, where $\mathbb{C}$ is a constraining element. This constrains the policy space such that an allowable policy π is within an allowable policy space Γ, where $\Gamma \subset \Pi$ contains the policies satisfying constraints $c_i \in \mathbb{C}$. The policy space Γ can be constrained by limiting the variance of return, or by requiring a higher minimum expectation of return for a policy to be deemed acceptable. [17, 18]

Exploration reduction, the other subgroup of SRL, is primarily focussed on intervening in the exploration process, either by preventing the agent to explore manually, by priming the Value function by human demonstrations, or by having a human in the loop. The autonomy of the algorithm is compromised, though agent's survivability is drastically improved, particularly in the case of human intervention when unsafe maneuvers are imminent, accompanied by a negative reward to discourage repeating the same mistake [19].

In the aerospace domain, SRL has been applied in the so-called SHERPA (Safety Handling with Risk Perception Algorithm) [20], which splits the environment into sets of Fatal State Spaces (FSS) and Safe State Spaces (SSS). When in Lead-to-Fatal (LTF) states (LTS$\subset$SSS), those states which are not fatal but are certain to result in the agent reaching FSS, the algorithm triggers an ideal backup policy. A further expansion of the algorithm, called OptiSHERPA [21], was introduced to account for the problems for which the SHERPA model was insufficient in handling the risk perception. The OptiSHERPA model added evasive strategies as options to the agent, allowing it to avoid states of imminent danger.

B. Artificial Potential Fields

Artificial Potential Field (APF) methods are primarily used for obstacle avoidance purposes in navigational problems. The concept is based on energy sources and sinks, which repel or emit potential energy. From this energy, actions can be derived such that the navigating agent goes toward the goal and away from any obstacles it might face. The agent will seek a state of lowest potential based on a gradient descent search method by minimizing the potential function. [13]

The energy potential at any given state, $\mathbb{U}(x)$, is given by:

$$\mathbb{U}(x) = \mathbb{U}(x)_{obs} + \mathbb{U}(x)_{goal} \tag{11}$$

with the obstacle (*obs*) emitting a negative potential $\mathbb{U}(x)_{obs}$, modelled by Equation 12.

$$\mathbb{U}(x)_{obs} = \begin{cases} k_r \left[\frac{1}{\rho(x)} - \frac{1}{\rho_0}\right]^2, & \text{if } \rho(x) \leqslant \rho_0. \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

The goal state, on the other hand, emits an attractive potential:

$$\mathbb{U}(x)_{goal} = \begin{cases} \frac{1}{2} k_g \rho^2_{goal}(x), & \text{if } \rho_{goal}(x) \leqslant d. \\ d k_g \rho_{goal}(x), & \text{otherwise.} \end{cases} \tag{13}$$

where $\rho(x) = |x - x_{obs}|$ is the euclidean distance from the current state to the obstacle and ρ_0 is the influence distance of the APF. ρ_{goal} is the distance to the goal state, with d representing the quadratic radius of influence from the goal state. k_r and k_g are proportional gains of the two functions. [13]

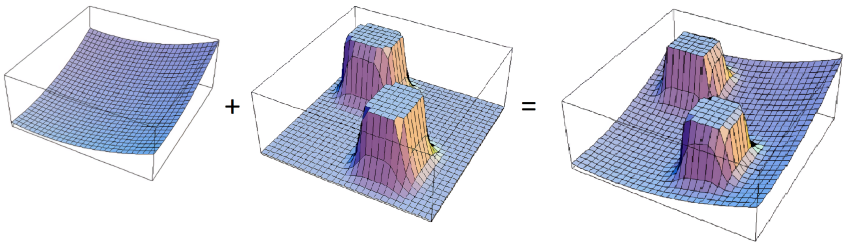


Fig. 2 An illustration of a combination of attractive and repulsive APFs, adapted from [22]

Figure 2 shows the individual APF produced by Equations 12 and 13, with $\mathbb{U}(x)_{goal}$ shown in the leftmost subplot, $\mathbb{U}(x)_{obs}$ of two individual obstacles shown in the middle subplot, and their resulting combined field shown in the rightmost subplot. ρ_0 affects the slope of the obstacle APF, while d is used as a cut-off for qudratic growth of APF for states far away from the goal state.

Where APF methods commonly fail is in the edge-case scenarios with lots of obstacles preventing the agent from advancing further, i.e. reaching a local minimum state from which any move would result in an increase in the negative field experienced by the agent. Mitigating this behaviour has been extensively studied, and a possible solution could be found by combining the APF techniques with Reinforcement Learning.

Addressing these shortfalls has been previously achieved using various methods. Potential Gradient Descent Algorithm (PGDA) [8], Virtual Waterflow Method [13], and Harmonic Potential Functions [3, 4] are some of the methods used in overcoming local minima problems and oscillations induced by multiple sources of potential fields causing interference in the environment. PGDA and Virtual Waterflow Methods both eliminate local minima by overriding a discovered minimum's state value with a higher one, equating it to its surroundings. Harmonic Potential Functions use Laplace functions which prevent spontaneous local minima from forming.

In the aerospace domain, APF methods have previously been applied to flight control and in interface design for UAV navigation [6, 14]. The UAV navigation application aids the remote pilot by providing similar feedback which would be experienced by a pilot onboard the aircraft, giving a sensation of resistance of the control surfaces during manoeuvres. The force feedback strength was coupled to the UAV's velocity, producing a realistic sensation of flight, and contributing to the fidelity of flight. APF methods have also been applied to real-time collision avoidance path planning algorithms for UAVs. Mac et al. [5] propose a modified APF method by including the agent's velocity in the Potential Field formation so that the global minimum is reached if and only if the agent has zero velocity at the goal state. Scherer et al. [23] use a global and a local planner for planning the route taken by an autonomous helicopter- the global APF creating a general route the UAV is to follow toward the goal with the local APF acts to prevent collision with obstacles, while maintaining the general course toward the preset goal.

IV. Safe Reinforcement Learning using Artificial Potential Fields

Various methods of integrating Artificial Potential Field information are introduced to the standard Reinforcement Learning algorithm. With safety being a top priority, the resulting algorithm's performance was expected to be less than that of a pure RL controller, since a trade-off exists between the two measures.

A. Reward function Modification

One method of APF integration is to influence the reward function based on the strength of the Potential Field the agent experienced. Integrating PF information with the reward function is done with a tunable parameter k_{PF} in order to normalize the reward values with the potential field strength, as seen in Equation 14.

$$R_{PF}(x) = R(x) + k_{RPF} \cdot \mathbb{U}(x) \tag{14}$$

The reward function, though having direct influence on the agent's performance, is predominantly used for instructing the agent toward the final end-state, or goal. It is considered bad practice to modify the reward equation extensively for purposes other than defining the end goal [15], hence a value function modification is also examined.

B. Value function Modification

Similar to the Reward function, the Value function modification is done by directly influencing the value of the state by the strength of the potential field. Unlike the reward function, the Value function does not interfere with the goal formation, nevertheless its influence on learning is more direct, with the potential for the agent to learn the effects of the Potential Field faster than with the reward function modification. Equation 15 shows the modified Value function with the APF information integrated.

$$\mathbb{V}^{\pi}_{PF}(x) = \mathbb{V}^{\pi}(x) + k_{VPF} \cdot \mathbb{U}(x) \tag{15}$$

A step further, the Action-Value (Q) function can be used in order to improve on the Value function integration. Namely, adding the action component to the APF integration will improve the algorithm's ability to determine the best possible path in the environment. The resulting Equation 16 will provide a state estimation featuring not only the states which would lead to a high cumulated reward, but also the actions that need to be taken at those states, removing ambiguities in decision-making and leading to a faster convergence. This entails that the agent has an higher survivability during learning.

$$\mathbb{Q}^{\pi}_{PF}(x, a) = \mathbb{Q}^{\pi}(x, a) + k_{QPF} \cdot \mathbb{U}(x, a) \tag{16}$$

C. Policy Modification

In addition to influencing the Value functions, APF information is also integrated in the policy, guiding the agent to avoid unsafe subspaces of the environment directly through decision-making.

Policy modification is implemented in addition to the Value modification discussed in the previous Section, as the Value function is a determining factor is algorithm's performance; with only a policy APF component, the algorithm performs on par with the pure RL controller, while combining Policy and Value function modifications has produced results outperforming them individually.

1. Backtracking

In the backtracking policy, the agent will perform as it was under the ε-greedy policy (agent selects a random action $\varepsilon \cdot 100\%$ of the time, the rest of the time selecting the action with the highest action-value score), unless high strengths of the potential field are experienced in the current state. A tuning gain is necessary in order to normalize the Potential Field energy values to the values normally found in the Value matrix.

The backtracking algorithm has a two-fold mechanism for increasing safety of the agent: At low values of APF energy, the exploration is reduced by halving ε. When high Potential Field energy is detected, meaning the agent is approaching an obstacle and still heading toward it, the backtracking algorithm forces the agent to go in the opposite direction. One of the side effects of this safety mechanism is the agent does not experience the negative effects of committing a dangerous action- for example, the agent would not experience hitting the obstacle and receiving a large negative reward due to unwanted behaviour- and therefore the previous action is given a negative reward through the Value function matrix instead.

Backtracking on its own is a sound safety measurement conceptually, but could inevitably experience edge cases where it fails in preventing the agent of dangerous manoeuvers, particularly in more complex control systems. Hence, backtracking is better suited to be used in conjunction with another policy.

2. Softmax Selection

A softmax function outputs a normalized probability distribution of the input vectors, meaning the sum of the probabilities will be equal to 1. A softmax policy has shown to be useful in settings where multiple options are of equal or similar quality, where ε-greedy policy would result in reinforcing a single transition. Being differentiable, the softmax policy also improves decision-making in gradient-descent methods such as APF.

The base equation for a softmax policy is shown in Equation 17, with $p(a|x)$ denoting the probability of selecting action a while in state x. $Q(x, a)$ is the action-value matrix and τ is a temperature parameter which controls the spread of the probability distribution; as τ goes to infinity the individual probabilities become equivalent.

$$p(a|x) = \frac{e^{\frac{Q(x,a)}{\tau}}}{\sum_{b=1}^{n} e^{\frac{Q(x,b)}{\tau}}} \tag{17}$$

Integrating APF information in the softmax function can be done by multiplying the action-value parameter or adding the APF component to it in the exponent. It has been determined both from literature [24, 25] and experiments that adding the APF component to the exponent, without the influence of the τ parameter, is the most effective strategy.

$$p(a|x) = \frac{e^{(\frac{Q(x,a)}{\tau} + k \cdot U(x))}}{\sum_{b=1}^{n} e^{(\frac{Q(x,b)}{\tau} + k \cdot U(x))}} \tag{18}$$

The APF modified softmax policy is shown in Equation 18, with $U(x)$ as the Potential Field component influenced by tuning gain k which normalizes the APF strength to the $Q(x, a)$ component values.

V. Preliminary Results

The methods described in Section IV are first implemented on a simple system in this Section, followed by a flight controller application in Section VI.

A 2-dimensional gridworld environment has been setup with scattered obstacles, to simulate an agent navigating around, from initialization until reaching the goal state. The experiment was first conducted with a pure Reinforcement Learning controller, followed with a RL controller featuring embedded Potential Field information. Furthermore, a

variety of integration methods of the potential field information were utilized to compare them relative to one another and relative to the pure RL performance.

A 14x14 grid is setup as shown in Figure 3. The agent can move one state per step, either horizontally or vertically, and is penalized with -1 reward per move. Reaching the goal state (green) counts as a successful eposide, giving the agent a reward of 100 and terminating the episode. Hitting an obstacle (black) gives the agent a reward of -100, also terminating the episode in the process. Artificial Potential Fields are introduced, as shown in Figure 4, and the agent is penalized with -2 and -3 reward per step for yellow and red fields, respectively.

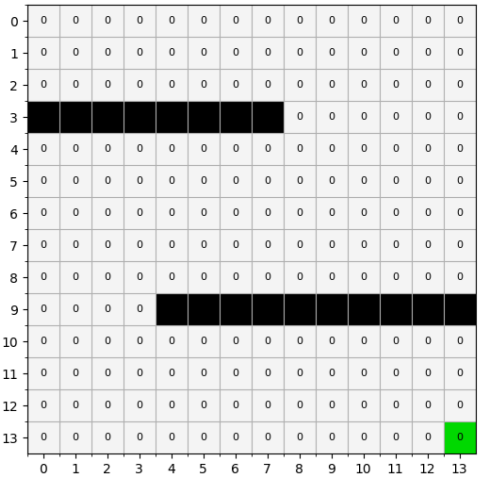


Fig. 3 Gridworld environment with obstacles (black) used in simulation

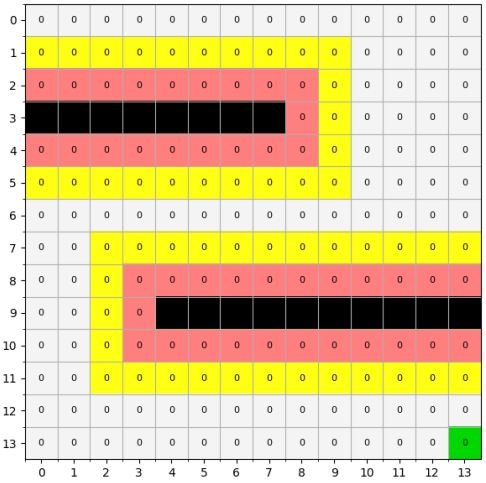


Fig. 4 Gridworld environment with APF of varying strength marked in red and yellow

350 episodes are run per simulation (300 ε-greedy followed by 50 greedy episodes), each episode being initalized at the [0,0] state (top-left corner of the environment). The episodes are capped at 300 steps taken per episode, to simulate the act of exhaustion (running out of fuel/battery/energy). The parameters used for Equation 7 are $[\alpha_L = 0.9, \gamma = 0.9]$, with an ε-greedy policy acting as the decision-maker with $\varepsilon = 0.1$. ε-greedy policy explores the environment $\varepsilon \cdot 100\%$ of the time, the remainder of the time acting greedily (picking the transition which has the highest value of the return).

Temporal Difference learning is utilized in the form of Q-learning, wherein a matrix (Q-matrix) is used to store the action-value function data for each combination of states and actions. A policy, such as ε-greedy, is used to select an action from a list of available ones. The process is iterated until the environment sends a signal informing the agent an episode is complete, then again until the designated number of episodes has been reached. The algorithm is shown in Algorithm 1.

Algorithm 1: Temporal Difference (Q) Learning

Initialize Q(s,a) arbitrarily;
Choose **a** from **s** using policy (ε-greedy or softmax);
while *i* < *total episodes* **do**
  Take action **a**, observe **r**,**s'**;
  Choose **a'** from **s'** using policy on Q(s,a);
  Q(s,a) $\Leftarrow$ **Q(s,a)**+α [**r** + γ **Q(s',a')** - **Q(s,a)**];
  s $\leftarrow$ **s'**, **a** $\leftarrow$ **a' end**
  Until **s** is terminal

The quality of algorithms with and without APF information integrated is compared relative to one another. The performance is measured based on the total reward collected and number of steps taken during an episode, as well as on a comparison of individual state visits and triggers of termination of an episode.

In Figure 5, rewards per episode are compared between a pure RL controller, and ones featuring reward and value (Q) function modifiers. The modifiers improve performance in terms of convergence speed and consistency of results, the consistency is further illustrated in Figure 7, as modified algorithms make no collisions after episode 75, while pure

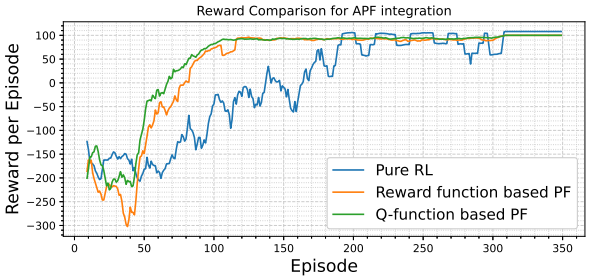


Fig. 5 Rewards per episode, smoothed rolling average of 10 episodes

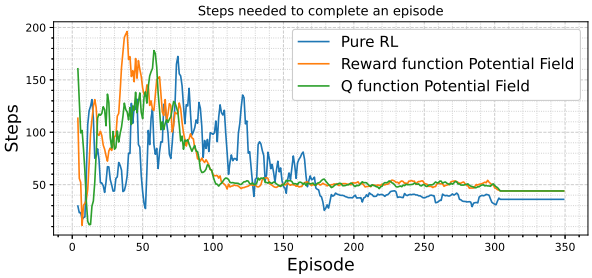


Fig. 6 Steps taken per episode, smoother rolling average of 10 episodes

RL controller's episodes periodically end with a collision throughout training.

Figure 6 tracks the number of steps taken per episode. Once again, modified algorithms tended to converge to a stable number of steps faster than the pure RL one, albeit having to take more steps per episode due to their aversion of the APF around obstacles (see Figure 8). Pure RL also heavily fluctuates in step count due to still not having learned where the goal is- referring back to Figure 7, modified algorithms find the goal state in less time and reinforce the pathways- pure RL is still searching for the positive reward provided by the goal state at a point when the modified algorithms have already visited the goal state multiple episodes in a row.

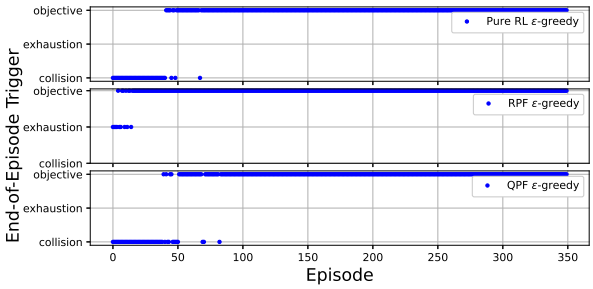


Fig. 7 Trigger for episode termination, optimization criterion

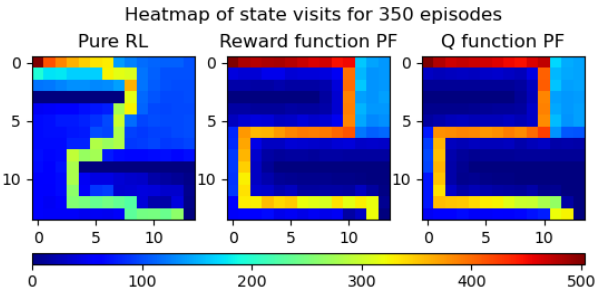


Fig. 8 Heatmap of state visits for different algorithms

With this comparison, it was confirmed that Value function modifier (QPF) not only outperforms a pure RL controller, but also the reward function modifier (RPF). Therefore, the Value function modified algorithm will further be used as a base for policy modification comparisons.

The policies chosen for the comparison are standard ε-greedy used above, with the addition of backtracking and softmax policies introduced in Section IV.C. However, it was found that the softmax policy has a smaller tolerance to the simulation parameters, performing inadequately for the original $\alpha_L = 0.9, \gamma = 0.9$ setting. With parameters being set to $\alpha_L = 0.8, \gamma = 0.95$, the performance was comparable, although higher and lower values of the parameters made the softmax policy's performance unacceptable. The exploration rate used in ε-greedy policies was the same as in the previous comparison, $\varepsilon = 0.1$.

Figures 9 and 10 show the rewards and steps taken per episode for different policies. The softmax policy underperformed relative to its counterparts in terms of learning convergence, taking more time to learn the policy toward the goal state. Furthermore, exhaustion (reaching the limit for steps in an episode, Figure 11) was prevalent with the softmax policy, meaning that the agent spent a lot of its time making inefficient exploration in the environment. On the other hand, the softmax policy resulted in less obstacle collisions than the pure ε-greedy policy- proving slight potential in its use with obstacle avoidance, albeit with better tuning required to eliminate the inefficient exploration.

On the other hand, the backtracking policy performed as expected- avoiding obstacles entirely. Reaching exhaustion during early stages of learning instead, as shown in Figure 11, the agent eventually finds its way around obstacles, barely ever reaching close to the obstacle state, as shown in Figure 12. In this Figure, it can also be seen that the backtracking policy reinforced an error in the path selection, as it also involved inefficient exploration in the top-right corner of the map. These actions were later selected for their quality once the agent reached the goal in one of the early episodes,

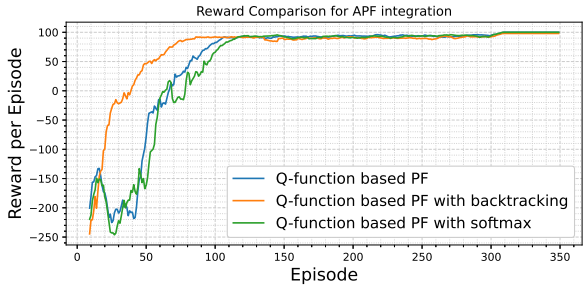


Fig. 9 Rewards per episode, smoothed rolling average of 10 episodes

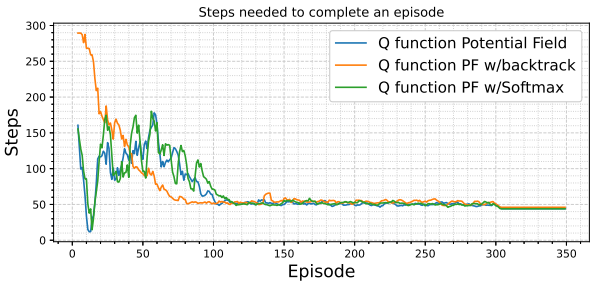


Fig. 10 Steps taken per episode, smoother rolling average of 10 episodes

having previously visited these states. Another reason for this inefficiency is the high discount factor γ being used in these simulations- the agent's performance drops off as γ approaches a value of 1, as shown in the sensitivity analysis in Figure 14.

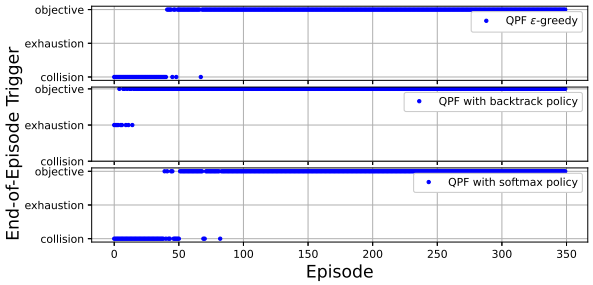


Fig. 11 Trigger for episode termination, policy modification

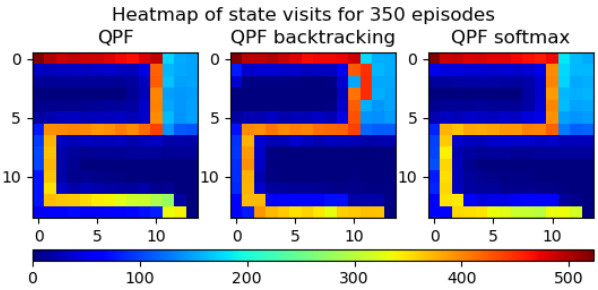


Fig. 12 Heatmap of state visits for different algorithms, policy modification

The backtracking policy exhibits the fastest convergence to optimal path out of the three algorithms compared above. The softmax policy does not rely on exploration once the optimal path is learned, only exploiting the existing knowledge after reaching the goal state once. The backtracking policy outperforms the other policies in steps taken, reward collected per episode, and most importantly, the number of obstacle collisions, making it suitable for aerospace applications.

A. Sensitivity Analysis

A sensitivity analysis is conducted by varying parameters individually while keeping others constant. The nominal conditions were chosen as in the previous section, $\alpha_L = 0.8, \gamma = 0.85, \varepsilon = 0.1$. The performance metric was the reward cumulated throughout the simulation, each simulation consisting of 300 ε-greedy episodes, followed by 50 greedy episodes.

Figure 13 shows the sensitivity analysis for the value and reward function modifiers. Compared to the pure Reinforcement Learning algorithm, the modified algorithms perform generally better, apart from the regions away from the nominal and most likely used parameter values (learning and decay rate are usually set close to 1, while exploration rate is just above 0). The decay rate having a negative inflection past $\gamma = 0.9$ can be attributed to the algorithm learning a wrong policy, and due to the decay having such a strong effect, the agent learns a wrong policy and keeps repeating it.

Figure 14 shows the sensitivity analysis for policy modifiers. The decay rate sensitivity analysis illustrates why the softmax policy did not perform at the nominal conditions at which ε-greedy policy is optimal- its performance is acceptable only for values higher than $\gamma = 0.9$. Softmax also did not utilize the exlporation rate, instead depending on probability distributions, hence it is uncorrelated with ε. It can also be seen that the backtracking policy outperform ε-greedy policy for the entire range around nominal values, making the algorithm more stable and predictable in a wider range of settings.

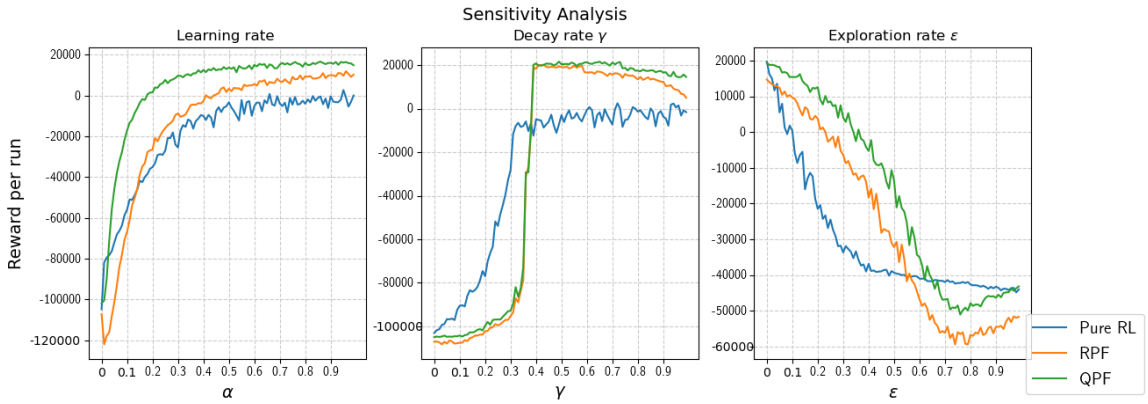


Fig. 13 Simulation parameter sensitivity analysis for reward and value function modifiers, RPF and QPF respectively

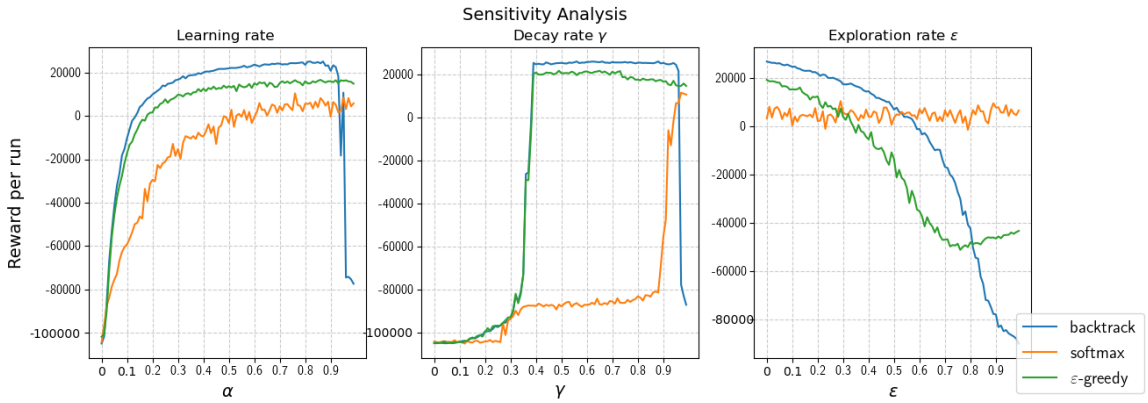


Fig. 14 Simulation parameter sensitivity analysis for policy modifers

VI. Flight Controller Application

The results obtained from the gridworld simulation demonstrate the effectiveness of integrating Artificial Potential Field methods to a Reinforcement Learning controller for ensuring an agent's safety during learning. On the basis of previous findings, autonomous learning using RL with APF is applied in the context of flight control. System Dynamics of the model used for flight control simulation are shown in Section VI.A, an overview of the simulation setup is presented in Section VI.B, followed by a discussion of the results in Section VI.C.

A. F-16 Simulation Model

The system used in the following experiment is a model of the General Dynamics F-16 fighter jet, which is reduced to an Linear Time-Invariant (LTI) system to the longitudinal states of interest. The state vector $\mathbf{x}$ is comprised of the true airspeed $V_t\,[\mathtt{ft/s}]$, angle of attack $\alpha\,[\mathtt{deg}]$, pitch rate $q\,[\mathtt{deg}]$, and altitude $h\,[\mathtt{ft}]$. The system is linearized at a nominal flight condition of $V_t = 900[\mathtt{ft/s}]$ at an altitude of $20,000[\mathtt{ft}]$. The resulting LTI system, acquired from Russel's Simulink model [26], is shown in Equations 19 and 20.

$$\begin{bmatrix} \dot{V}_t \\ \dot{\alpha} \\ \dot{\theta} \\ \dot{q} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} -0.012 & 37.06 & -32.17 & 0.1413 & 0 \\ 0 & -0.9748 & 0 & 0.9505 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & -4.276 & 0 & -1.317 & 0 \\ 0 & -900 & 900 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} V_t \\ \alpha \\ \theta \\ q \\ h \end{bmatrix} + \begin{bmatrix} 0.3549 \\ -0.0021 \\ 0 \\ -0.3139 \\ 0 \end{bmatrix} \cdot \delta_{el} \tag{19}$$

$$\begin{bmatrix} V_t \\ \alpha \\ \theta \\ q \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 57.2958 & 0 & 0 & 0 \\ 0 & 0 & 57.2958 & 0 & 0 \\ 0 & & 0 & 57.2958 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} V_t \\ \alpha \\ \theta \\ q \\ h \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot \delta_{el} \tag{20}$$

The system input is the elevator deflection angle δ_{el}, which is a discrete signal provided by the RL algorithm. Due to the system being linearized for the particular flight condition, the model is initialized with small random variations around the given condition, with variations of $1[\texttt{deg}]$ for the angular states and $50[\texttt{ft}]$ and $5[\texttt{ft/s}]$ for altitude and airspeed, respectively, from $\mathbf{x} = [900, 0, 0, 0, 20000]^T$.

B. Simulation Framework

Each simulation is comprised of 3000 episodes, with each episode running for 100 seconds in $dt = 0.2[sec]$ increments. The RL agent has access to the entire state-space, utilizing the angular states and the altitude in composing the Q-learning matrix.

The primary reference utilized by the algorithm is the pitch rate, with the altitude being tracked simultaneously. Pitch rate tracking is used due to it being the fasest adapting, and most direct, way of influencing the aircraft's attitude. Altitude tracking is used for APF integration, and is directly inferred as a safety metric. A sinusoidal pitch rate reference and an steady altitude reference used are given in Equation 21.

$$\texttt{q}_{\texttt{ref}} = \frac{100}{180} \cdot \pi \cdot sin(0.1 \cdot \pi \cdot \texttt{dt})[\texttt{deg}], h_{\texttt{ref}} = 20000[\texttt{ft}] \tag{21}$$

The Q-learning matrix's dimensions are defined by the number of indices for each state, with boundaries defined for state values and the range subdivided into discrete sections. There exists a significant trade-off between resolution and performance - for a highly discretized Q-matrix, the agent does not get a chance to explore all of the combinations of states, while for large indices the accuracy is impacted due to each index encompassing a wider range of state values. The boundaries and index resolutions for the recorded states are shown in Table 1

Table 1 Q-matrix boundaries and index resolutions

	min	max	resolution
α [deg]	-4	4	10
θ [deg]	-4	4	10
$q_{\texttt{err}}$ [deg]	-2	-2	10
$h_{\texttt{err}}$ [ft]	-500	500	20
δ_{el} [deg]	0.75	0.75	11

The resulting Q-matrix has a size of $2.2 \cdot 10^5$ state-action fields, initialized as a zero matrix at the start of each simulation. Temporal Difference learning is conducted as in the previous Section, using Algorithm 1, albeit the reward function and the APF implementation were expanded to encompass the more complex environment. Hyperparameters used in the algorithm are $[\alpha_L, \gamma, \varepsilon] = [0.9, 0.5, 0.1]$, selected empirically by examining system response for hyperparameter values commonly used in RL problems (relatively high learning and decay rates and exploration rate approaching 0). Reward function is modelled around the $q_{\texttt{err}}$ and $h_{\texttt{err}}$ values, with a negative quadratic and absolute rewards, respectively. The error functions are the distances between the current state and the reference state calculated at each timestep. Further positive reward is given when the error is minimized- in the range of (-1,1) degrees and (-50,50) feet from the reference, respectively.

The algorithm is run as follows: each simulation starts with ε-greedy policy being used to select an action, repeated every timestep. A greedy policy is instigated each 50th episode for performance tracking purposes, as well as at the end of the learning phase in the last 300 episodes. This allows for intra-learning performance to be measured between the algorithms at fixed intervals, without the exploration introducing randomness to the data. Furthermore, exploration is

reduced through the learning process, specifically at episode numbers =500, 1000. This enables the agent to explore more of its environment early in the simulation, and exploit the knowledge once an optimal policy is already known.

1. Artificial Potential Field Formulation

The Artificial Potential Field is used to contain the agent within a specified altitude range, providing a safety constraint during learning. For the selected flight condition of $[V_t, h] = [900[\mathtt{ft/s}], 20000[\mathtt{ft}]]$ the APF is constructed as follows:

$$\mathtt{U}(h) = k_U * \left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)^2 \tag{22}$$

with the parameters equalling to: $\rho(h) = \mathtt{abs}|h - 20000|$, $\rho_0 = 35$, and $k_U = 0.5 \cdot 10^5$. The parameters were chosen such that the resulting APF addition to the Q-matrix neither overwhelms nor becomes negligible to the initial action-value function.

The APF is used firstly as a Q-function modifier, with the potential field strength being added to the Q-matrix at the current state. A backtracking algorithm is added to be compared with the standard ε-greedy policy. The backtracking policy is a modified ε-greedy policy which follows the same action selection procedure if no APF is present. In case APF is non-zero, the policy first halves the exploration, as an APF presence signifies a significant deviation from the reference altitude. If the strength of the field continues to rise, i.e. further deviation is experienced, the backtracking is activated, limiting action selection to half of the available actions, and pointing the agent back to a safe state space region.

C. Results and Discussion

Post-simulation processing includes analyzing the rewards gained per episode, comparison of the tracked states and inputs and an analysis of the performance and safety of the agent for given algorithms. Algorithm performance is defined with the agent's ability to track the reference pitch rate q, since it is the agent's primary reference. Safety is defined by the agent's altitude deviation from the initial condition, since the APF's primary purpose is keeping the agent within a specified altitude range. Therefore, as the agent goes further away from the designated altitude range, the safety of the algorithm is reduced.

First, a pure RL controller is compared to the QPF (Q-function based Potential Field) integration, followed by a comparison of the QPF model with one featuring a policy modification, namely, backtracking. Reward-based Potential Field modifier and the softmax policy were not considered due to their lack of performance demonstrated in the previous section. The following results are presented as an average of the entire simulation run per algorithm, as well as end-of-training performance.

1. Pure Reinforcement Learning vs. Q-function based Artificial Potential Field modifier

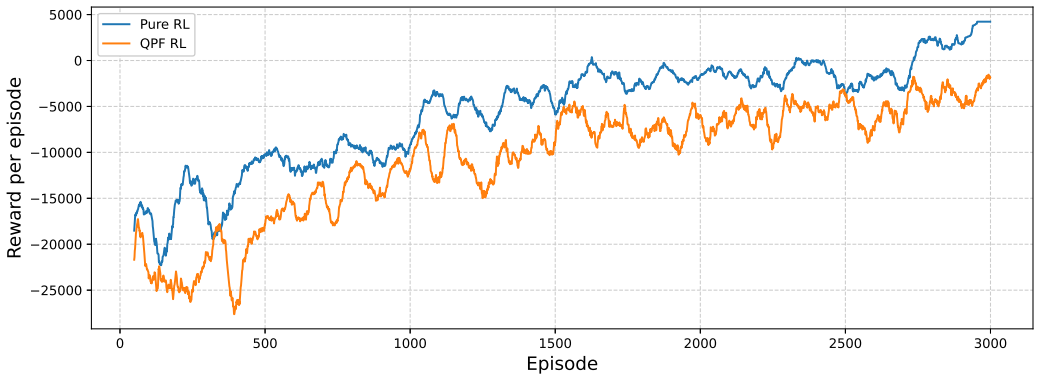


Fig. 15 Reward per episode, for pure RL (blue) and QPF (orange), 50 episode rolling average

Comparing the reward collected per episode during the learning phase, as shown in Figure 15, it can be noted that the introduction of APF information to the Q-function reduces the algorithm's previous performance, primarily due to the pitch rate tracking having less of an influence on decision making, as seen in Figure 17.

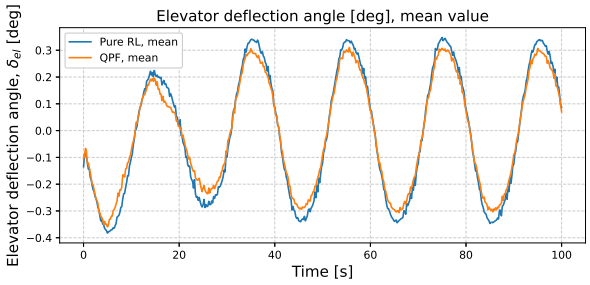


Fig. 16 Elevator deflection provided by the policy, mean value

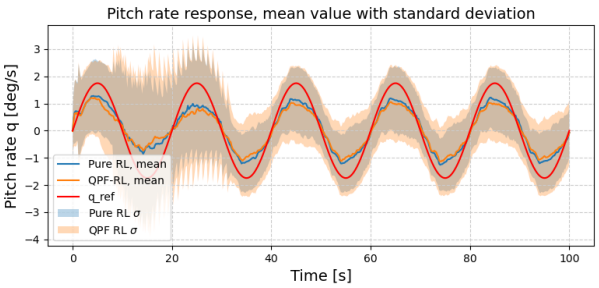


Fig. 17 Response pitch rate with the reference pitch rate in green, mean value

The LTI reduced model of the F16 has a positive bias in the pitch angle, seen in Figure 19, contributing to an increasing altitude during the simulation. In Figure 20 the altitude is shown alongside an APF function on the right. The altitude of the QPF modified controller follows the bounds of the potential field, correcting for the positive θ bias in the model. The result is a more level aircraft throughout the majority of the simulation. A trade-off in performance tracking of the pitch angle, as well as angle of attack (Figure 18), allows the agent to keep within a designated safe altitude.

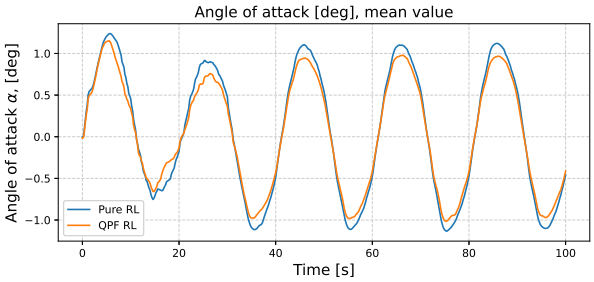


Fig. 18 Angle of attack response, mean value

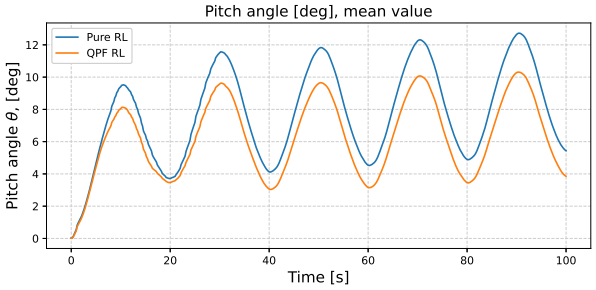


Fig. 19 Pitch angle response, mean value

The mean altitude response values over the entire simulation run and end-of-learning stage are shown in Figures 21 and 22. Here, the APF influence can be seen to decrease the aircraft's tendency pitch upwards, resulting in a more level aircraft overall. The end-of-training standard deviaton for the pure RL controller is smaller due to the agent settling to an optimal policy, as can be seen in the last episodes in Figure 15

The performance trade-off is visible throughout the learning process, showed in Figures 23 and 24 over the entire simulation run. Rewards collected per episode show the learning of the policy as well as the effect of adding APF information. As it can be seen through error tracking, the QPF algorithm manages better altitude tracking at the expense of pitch rate tracking.

From the provided data, it can be concluded that the addition of the Artificial Potential Field to the Q-function does increase the safety of the agent during learning.

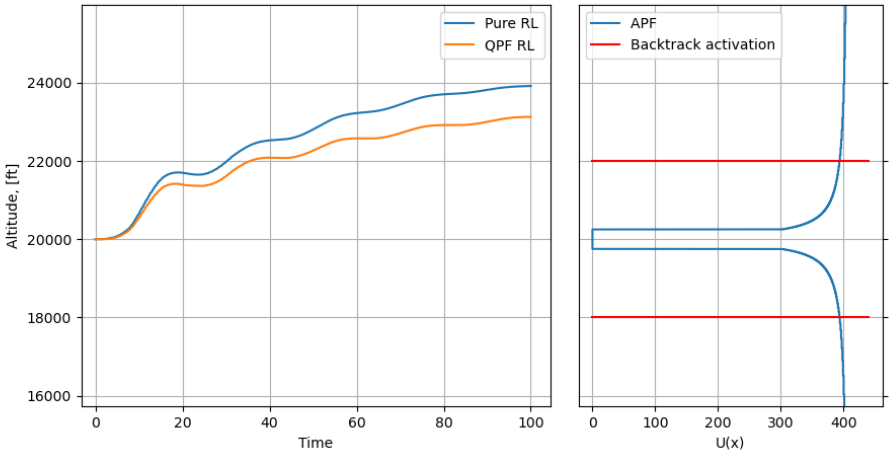


Fig. 20 Altitude response(left) and Artificial Potential Field range(right), episode 3000

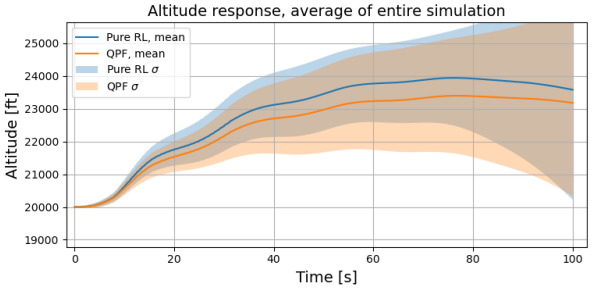


Fig. 21 Altitude response, mean over 3000 episodes

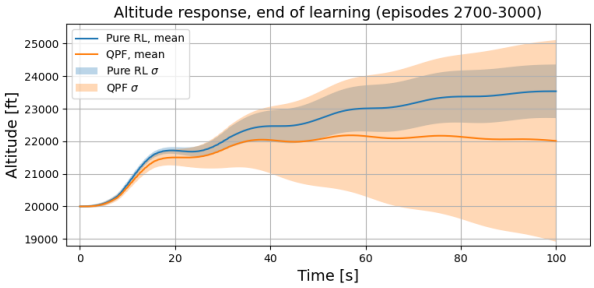


Fig. 22 Altitude response, end of learning mean (episodes 2700-3000)

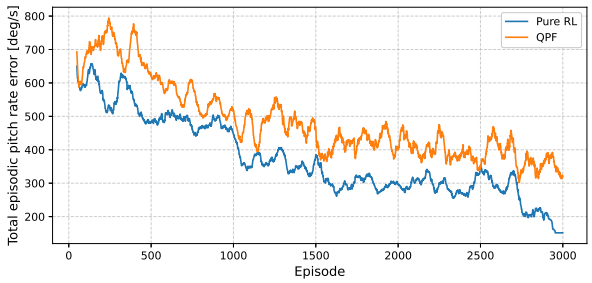


Fig. 23 Pitch rate error (from reference), total per episode, rolling average of 50 episodes

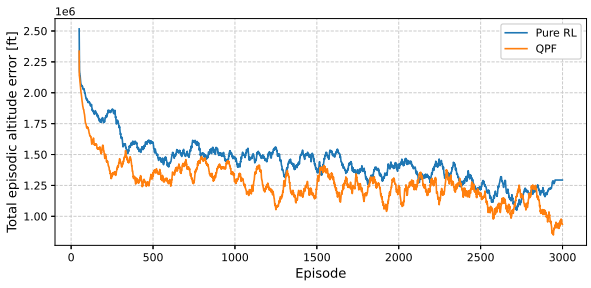


Fig. 24 Altitude error (from reference), total per episode, rolling average of 50 episodes

2. Q-function based Artificial Potential Field modifier vs Backtracking Policy

In this Section, the backtracking policy is compared to ε-greedy policy, which was used in the Section VI.C.1 in both algorithms. Backtracking policy provides the agent a safe state space surrounding its initial flight condition to learn in an environment where exploration is still possible, yet it is safer due to the imposed virtual restrictions.

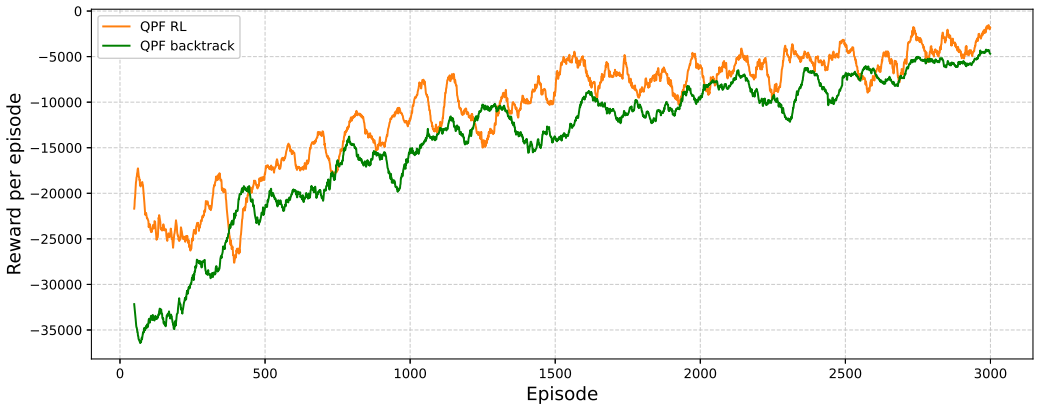


Fig. 25 Reward per episode for QPF ε-greedy policy (blue) and backtracking policy (orange)

Examining the episodic rewards in Figure 25, the learning progression is similar throughout the simulation, with backtracking having a steadier, less volatile, learning curve. This is a direct consequence of the limited exploration imposed by the backtracking restriction- the agent is not allowed to surpass certain boundaries and is instructed to backtrack, hence there is less variation, particularly with the negative rewards collected with large deviations from the reference pitch rate and altitude. The initial response of the backtracking policy performing worse than the ε-greedy one is also due to the lack of exploration, which is particularly important in the early learning stage.

The performance of the two policies is similar throughout the simulation, particularly in the greedy phase at the end of the simulation. A side effect of the restrictive environment can be seen in the backtracking algorithm underperforming at the end of learning, in Figure 30, the altitude tracking is more accurate when using ε-greedy policy. This can be further elaborated with Figure 34 showing the error throughout training. The backtracking policy returns to the same level of safety as the ε-greedy policy in the greedy stage where backtracking is disabled. During the learning though, backtracking serves its purpose in keeping the agent contained within set bounds, increasing its safety.

Comparing the angular states α, q and θ in Figures 28, 27 and 29, respectively, the performance of both algorithms is very similar, with slight deviations, since the greedy policy follows APF information only based on the Q-matrix modifier, omitting the implemented backtracking. Throughout training, the pitch rate tracking by the ε-greedy policy is relatively better, as seen in Figure 33, once again due to the limitation placed by the backtracking policy to stick within a specified range- preventing the agent from following the reference pitch rate.

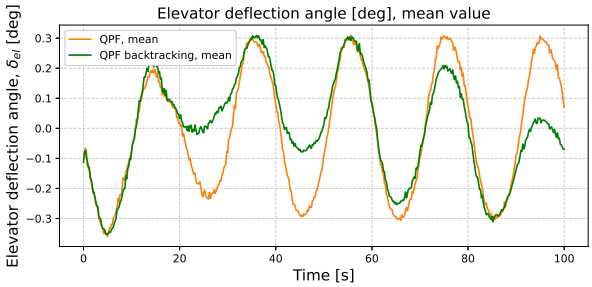


Fig. 26 Elevator deflection provided by the policy, mean value

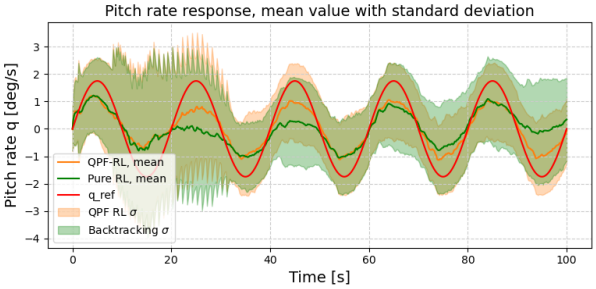


Fig. 27 Response pitch rate with the reference pitch rate in green, mean value

By the end of the learning stage, once the agent begins greedily exploiting the gained knowledge from learning, it

has not interacted with the states outside its designated range imposed by the backtracking. Due to the accumulated negative reward in the explored states, the agent is also more likely to go for the less explored states outside of the backtracking range, once that restriction is removed, since those states will have a comparatively higher Q-function value due to not being visited. In this case, the pitch rate tracking prevails over the APF information, resulting in less pitch rate error, as seen in Figure 27 in the last 20 seconds of the episode.

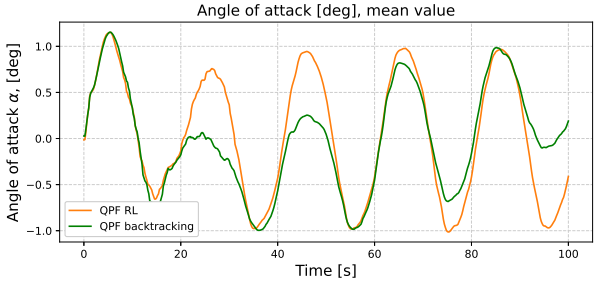


Fig. 28 Angle of attack response, mean value

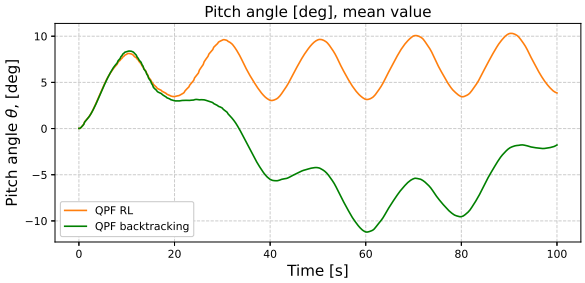


Fig. 29 Pitch angle response, mean value

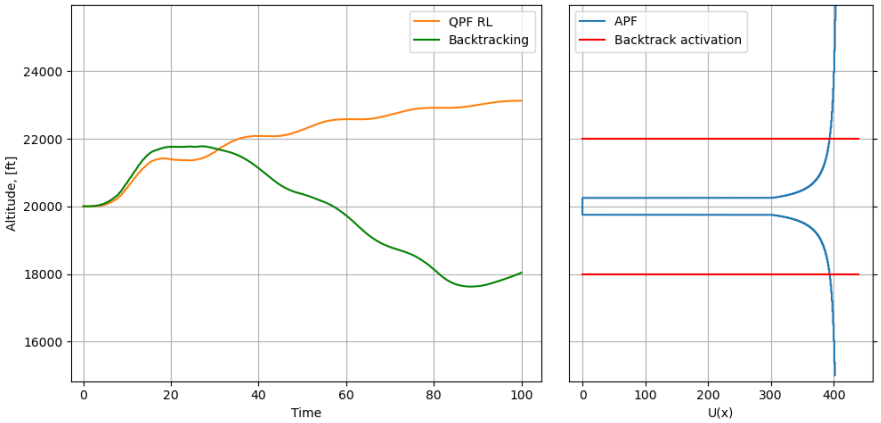


Fig. 30 Altitude response(left) and Artificial Potential Field range(right), episode 3000

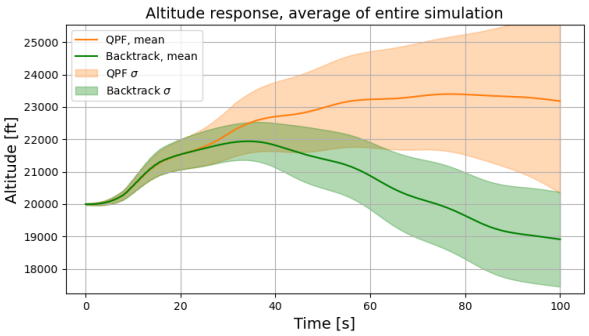


Fig. 31 Altitude response, mean over 3000 episodes

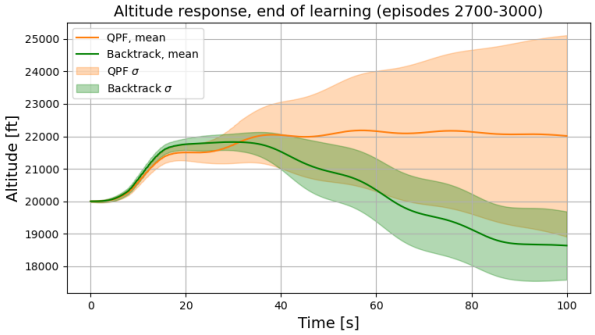


Fig. 32 Altitude response, end of learning mean (episodes 2700-3000)

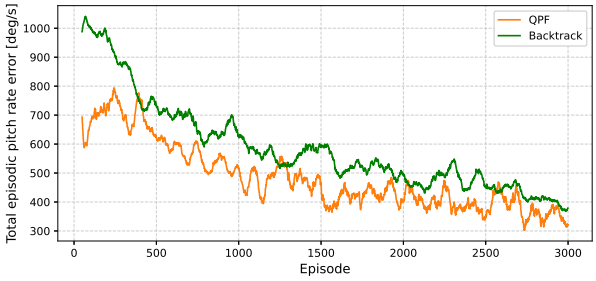


Fig. 33 pitch rate error per episode (from reference) throughout the simulation run

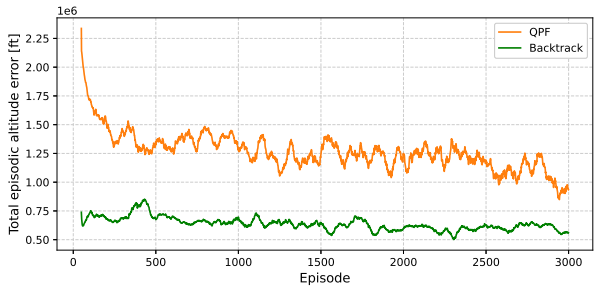


Fig. 34 altitude error per episode (from reference) throughout the simulation run

3. Sensitivity Analysis

The Reinforcement Learning hyperparameters were varied throughout the range of [0,1] to examine their impact on the algorithm's performance. The sensitivity curves draw similarities with the Sensitivity analysis conducted during the preliminary study, in Figures 13 and 14, albeit with some differences.

Figure 35 shows the learning rate's effect on the algorithms' performance- unlike with the gridworld environment, here a continually rising learning rate does not result in a continually better performance, particularly for the pure RL controller. Learning rate's effect on the training is crucial to the algorithm's performance, representing how fast the agent learns from the rewards experienced. A higher learning rate results in the agent learning faster, but a learning rate value close to $\alpha_L = 1$ will result in the Q-matrix values approaching each other, resulting in a deadlock scenario with decision making. This can be seen more clearly in Figure 14 where the performance drastically drops as the value approaches $\alpha_L = 1$; in Figure 35 this is not evident due to the larger step size between simulations.

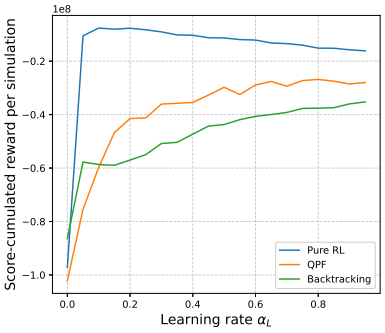


Fig. 35 Learning rate α_L sensitivity analysis

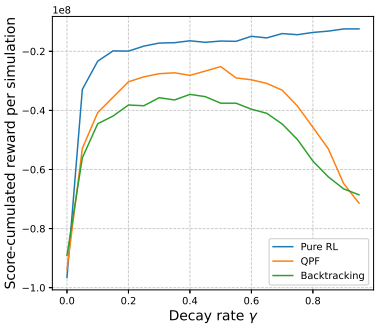


Fig. 36 Decay rate γ sensitivity analysis

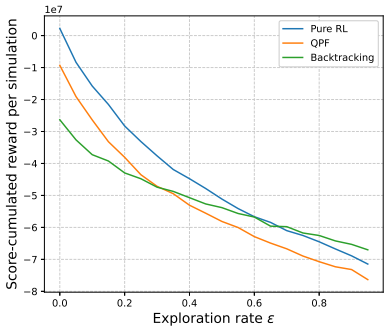


Fig. 37 Exploration rate ε sensitivity analysis

In Figure 36, decay rate γ sensitivity is different than the previous findings, in which a value of $\gamma = 0.4$ was an activation level for the algorithm to start performing. Here, there is a steady increase in algorithm's performance with an increase in the decay rate, due to its function of carrying the reinforcement back through the states that were previously visited. The falloff of performance for γ approaching 1 for algorithms utilizing APF integration is due to the agent receiving negative reward for violating the APF bounds, and carrying that reward further back to the previously visited states. States visited early on during an episode will receive negative rewards early in the simulation run, resulting in the agent avoiding them throughout the learning process, resulting in suboptimal performance. The initially selected value of $\gamma = 0.5$ was therefore close to an optimal decay rate.

Lastly, exploration rate sensitivity is shown in Figure 37. Exploration rate is best left close to 0 since only a small fraction of the time is necessary to successfully explore the environment, even when the environment is comprised of 10^5 state combinations. Backtracking policy has a performance relatively worse than the ε-greedy policies, with or without the inclusion of APF information, due to the boundaries preventing it from fully exploting positive reward by following the pitch rate reference. However, backtracking has a smaller slope, eventually outperforming both ε-greedy policies with an increasing exploration rate. This is due to the additional layer of safety, even with high exploration, resulting in a smaller decrease in performance per an increaase in exploration- even with the agent exploring half of the time, leaving the operating altitude window will still be prevented by backtracking.

VII. Conclusion

This paper has demonstrated Safe Reinforcement Learning with Artificial Potential Field methods to be a viable strategy for increasing the safety of a Reinforcement Learning controller. RL and APF compliment each other to produce an algorithm which outperforms a pure RL controller, APF acting as a safety measure to reduce the risks introduced by RL's exploratory nature, and RL overcoming APF method's main issue of getting stuck in local minima with the use of exploration.

APF integration within RL has been first introduced within a obstacle-filled gridworld environment. APF was integrated in the reward and value functions, the latter proving to be a better implementation. Then policy-based APF modifications were tested, comparing a standard ε-greedy policy to a softmax and backtracking policies with APF integration. The backtracking policy completely eliminated obstacle collisions, providing further proof of its potential

to increase the agent's safety during exploration.

Having established its potential, the combination of APF and RL was applied to a flight controller. A discretized LTI model of the F16 fighter jet aircraft was used to test the algorithm's capabilities for flight control purposes. The model was reduced longitudinally, with the RL controller being applied to the elevator deflection input. Proving to be more effective than the standard RL controller, the model was expanded to accommodate APF information with the decision-making policy function, in the form of a backtracking algorithm capable of keeping the agent/aircraft within a bounded safe region. The safety was measured based on the agent's ability to stay witin a safe distance from the initial altitude, while performance was determined based on tracking of a predefined reference pitch rate. A trade-off exists between safety and performance due to the model having a positive bias in the pitch angle, resulting in a steady increase in altitude. The APF modified RL algorithm was able to correct for the altitude drift and restrain the aircraft's altitude response to a predefined safe state space.

The results show that potential application of APF to RL can improve the safety aspect of integrating exploration-based learning algorithms to aerospace control problems. Many challenges were not explored in this research, addressing these challenges could bring the algorithm's performance and safety to a level necessary for real-time decision-making. The backtracking implementation prevented the agent from exposing itself to dangerous situations in both preliminary findings as well as in the flight controller application. This example shows promising results in terms of restricting the agent while simultaneously allowing it to safely explore within predefined bounds. Furthermore, hyperparameter optimization of the above research was trivial, with exploration rates being varied in set intervals. A tunable hyperparameter optimization, with varying rates of learning and decay, could improve the algorithm's performance and safety further. Combining APF methods with more complex RL models, such as deep neural networks, could also be utilized to develop the algorithm's efficiency further.

In terms of the simulation setup, a more realistic environment can be used to better understand the algorithm's performance in a real-life scenario. Disturbance inputs were not explored in this research, and future research into the topic of Artificial Potential Fields and Reinforcement Learning could benefit from a stochastic modelling of wind gusts, for example. Furthermore, the algorithms used here were not preloaded with any a-priori knowledge of the environment or the dynamic system being controller, initial information which could improve the algorithm's performance and safety aspects further. The current state-of-the-art research in Reinforcement Learning is focused on the utilization of deep neural networks; an implementation of the above research featuring neural networks in place of the discrete action-value matrix could further the efficiency and speed up learning, while maintaining a level of safety necessary for aerospace applications.

References

[1] Santoso, F., Garratt, M. A., and Anavatti, S. G., "State-of-the-Art Intelligent Flight Control Systems in Unmanned Aerial Vehicles," *IEEE Transactions on Automation Science and Engineering*, Vol. 15, No. 2, 2018, pp. 613–627. https://doi.org/10.1109/TASE.2017.2651109.

[2] Zhang, X., Xian, B., Zhao, B., and Zhang, Y., "Autonomous Flight Control of a Nano Quadrotor Helicopter in a GPS-Denied Environment Using On-Board Vision," *IEEE Transactions on Industrial Electronics*, Vol. 62, No. 10, 2015, pp. 6392–6403. https://doi.org/10.1109/TIE.2015.2420036.

[3] Shi, C., Zhang, M., and Peng, J., "Harmonic potential field method for autonomous ship navigation," *ITST 2007 - 7th International Conference on Intelligent Transport Systems Telecommunications, Proceedings*, 2007, pp. 471–476. https://doi.org/10.1109/ITST.2007.4295916.

[4] Kim, J. O., and Khosla, P. K., "Real-Time Obstacle Avoidance Using Harmonic Potential Functions," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 3, 1992, pp. 338–349. https://doi.org/10.1109/70.143352.

[5] Mac, T. T., Copot, C., Hernandez, A., and De Keyser, R., "Improved potential field method for unknown obstacle avoidance using UAV in indoor environment," *2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 2016, pp. 345–350. https://doi.org/10.1109/SAMI.2016.7423032.

[6] Liu, Y., and Zhao, Y., "A virtual-waypoint based artificial potential field method for UAV path planning," *CGNCC 2016 - 2016 IEEE Chinese Guidance, Navigation and Control Conference*, 2017, pp. 949–953. https://doi.org/10.1109/CGNCC.2016.7828913.

[7] Yang, X., Yang, W., Zhang, H., Chang, H., Chen, C. Y., and Zhang, S., "A new method for robot path planning based artificial potential field," *Proceedings of the 2016 IEEE 11th Conference on Industrial Electronics and Applications, ICIEA 2016*, , No. December 2014, 2016, pp. 1294–1299. https://doi.org/10.1109/ICIEA.2016.7603784.

[8] Bounini, F., Gingras, D., Pollart, H., and Gruyer, D., "Modified artificial potential field method for online path planning applications," *IEEE Intelligent Vehicles Symposium, Proceedings*, , No. Iv, 2017, pp. 180–185. https://doi.org/10.1109/IVS.2017.7995717.

[9] Khatib, O., "Real-time Obstacle avoidance for Manipulators and Mobile Robots," 1985.

[10] Garcia, J., and Andez, F. F., "A Comprehensive Survey on Safe Reinforcement Learning," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 29, No. 4, 2018. https://doi.org/10.1109/TNNLS.2017.2654539, URL http://ieeexplore.ieee.org/document/7842559/.

[11] Verbist, S., Mannucci, T., and Van Kampen, E.-J., "The Actor-Judge Method: safe state exploration for Hierarchical Reinforcement Learning Controllers," *2018 AIAA Information Systems-AIAA Infotech @ Aerospace*, , No. January, 2018. https://doi.org/10.2514/6.2018-1634, URL https://arc.aiaa.org/doi/10.2514/6.2018-1634.

[12] Mannucci, T., van Kampen, E., de Visser, C. C., and Chu, Q. P., "Hierarchically structured controllers for safe UAV reinforcement learning applications," *AIAA Information Systems-AIAA Infotech at Aerospace, 2017*, , No. January, 2017, pp. 1–13. https://doi.org/10.2514/6.2017-0791.

[13] Li-juan, XIE, XIE Guang-rong, CHEN Huan-wen, L. X.-l., "Solution to reinforcement learning problems with artificial potential field," *Springer*, Vol. 21, No. 10, 2008, pp. 402–410. https://doi.org/10.1007/s11771.

[14] Lam, T. M., Boschloo, H. W., Mulder, M., and Van Paassen, M. M., "Artificial force field for haptic feedback in UAV teleoperation," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, Vol. 39, No. 6, 2009, pp. 1316–1330. https://doi.org/10.1109/TSMCA.2009.2028239.

[15] Sutton, R. S., and Barto, A. G., *Introduction to Reinforcement Learning*, 1st ed., MIT Press, Cambridge, MA, USA, 1998.

[16] Heger, M., "Consideration of Risk in Reinforcement Learning," *Machine Learning Proceedings 1994*, 1994, pp. 105–111. https://doi.org/10.1016/b978-1-55860-335-6.50021-0.

[17] Geibel, P., "Reinforcement learning for MDPs with constraints," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 4212 LNAI, 2006, pp. 646–653. https://doi.org/10.1007/11871842_63.

[18] Tamar, A., Di Castro, D., and Mannor, S., "Policy gradients with variance related risk criteria," *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, Vol. 1, 2012, pp. 935–942.

[19] Quintia, P., Iglesias, R., Rodriguez, M. A., and Regueiro, C. V., "Learning on real robots from experience and simple user feedback," *Journal of Physical Agents*, Vol. 7, No. 1, 2013, pp. 56–64. https://doi.org/10.14198/jopha.2013.7.1.08.

[20] Mannucci, T., Van Kampen, E.-J., de Visser, C. C., and Chu, Q. P., "SHERPA: a safe exploration algorithm for Reinforcement Learning controllers," *AIAA Guidance, Navigation, and Control Conference*, , No. January, 2015. https://doi.org/10.2514/6.2015-1757, URL http://arc.aiaa.org/doi/10.2514/6.2015-1757.

[21] Mannucci, T., van Kampen, E., de Visser, C., and Chu, Q., "Safe Exploration Algorithms for Reinforcement Learning Controllers," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 29, No. 4, 2018, pp. 1069–1081. https://doi.org/10.1109/TNNLS.2017.2654539.

[22] Saranli, U., "Lecture notes in Robot Motion and Control Planning," , August 2015.

[23] Scherer, S., Singh, S., Chamberlain, L., and Elgersma, M., "Flying fast and low among obstacles: Methodology and experiments," *International Journal of Robotics Research*, Vol. 27, No. 5, 2008, pp. 549–574. https://doi.org/10.1177/0278364908090949.

[24] Bhowal, A., "Potential Field Methods for Safe Reinforcement Learning," *Delft University of Technology*, 2017.

[25] Asadi, K., and Littman, M. L., "An alternative softmax operator for reinforcement learning," *34th International Conference on Machine Learning, ICML 2017*, Vol. 1, No. Property 2, 2017, pp. 360–369.

[26] Russell, R., "Non-linear F-16 Simulation using Simulink and Matlab," *Dynamics and Control Branch, NASA Langley Research Center*, 2003.

Part II

Preliminary Research and Simulation Study

Chapter 2

Literature Study

The following chapter elaborates on the theoretical background of Reinforcement Learning (RL) and Potential Fields(PF), the two techniques used to construct a guidance algorithm for the purpose of autonomous obstacles avoidance in Unmanned Aerial Vehicles (UAVs). Firstly, the basic concepts of RL are laid out in Section 2-1, with techniques used to create algorithms presented following, in Section 2-1-2. Potential Fields Methods, are discussed in Section 2-3.

2-1 Reinforcement Learning

Reinforcement Learning (RL) is a bio-inspired approach to decision making in unknown scenarios. RL relies on reward signals from the surroundings to determine subsequent steps, just like an animal does in a natural environment. For example, if an animal comes across a piece of fruit and eats it, positive reward signals (dopamine release in its brain) teach it that this action has a positive outcome, and encourages repeating said action again [18].

2-1-1 Basic Concepts of Reinforcement Learning

At the core of any Reinforcement Learning problem exists an environment, within which an agent resides. The agent is able to perform actions, which in turn change its state in the environment. A simple graphic illustrating this concept is shown in Figure 2-1.

The agent selects an action based on its current state, and the reward received from previous experiences. The environment provides the agent with all the available states, as well as the rewards received by being in each state. Using this information, the agent then tries to maximize the total reward received over its lifetime.

Reward Function The reward function is a mapping of the reward that is to be received at a particular state. Each state has a corresponding reward which is sent as a signal to the agent once the agent is in that particular state.

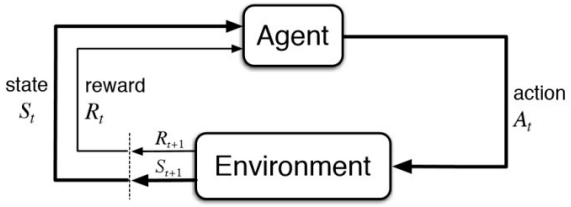


Figure 2-1: Agent-Environment interaction in RL [2]

The reward function is designed such that the desired result would lead to the agent maximizing the total collected reward. The reward is annotated as $\mathbb{R}^a_{ss'}$, i.e. the expected reward for taking action a in the state s, resulting in a transition to state s'. Designing the reward function correctly, relative to the experiment and the subject being tested, is crucial for the RL algorithm in solving the problem.

Episodes The agent's lifetime is defined as an episode, and lasts from the moment the agent starts exploring the environment (initial state), until the agent has completed its goal (terminal state). Limitations are introduced such that an episode can only last a certain amount of steps, in case the agent does not find the goal. This prevents a scenario in which an episode never ends. Framing the problem in episodes allows the RL agent to store data from previous experiences, and act in the following episodes using this stored information.

Each episode consists of a number of iterations, each iteration corresponding to the agent executing a single action.

Markov Decision Processes A fundamental assumption in a RL problem is its independence from past events. The knowledge gained from previous experiences is stored, but it is irrelevant (to the agent) how and when this information was obtained. A Markov Decision Process (MDP) structures this assumption into a mathematical framework. MDP is a construction of the current state of the environment, which includes all the past experiences, but does not depend on them. An good example on an MDP is the chessboard during a game of chess–at a given move, the player looks for the best strategy for the current setup of the chessboard, and does not rely on previous setups of the board.

In the context of RL, a MDP is a tuple of state-spaces $[\mathbb{X}, \mathbb{A}, \mathbb{T}, \mathbb{R}]$, wherein:

- $\mathbb{X}$ represents all of the possible states
- $\mathbb{A}$ represents the combination of state-actions, all of the actions which can be taken at each state
- $\mathbb{T}$ is the transition probability function, which is used to introduce stochasticity to the system. The transition function dictates the probability of the agent executing the action it has selected
- $\mathbb{R}$ represents the reward function, which is composed of the possible combination of state-action pairs, and the corresponding reward for executing a specific action when in a specific state

Delayed Reward RL can act without any human input, once the program is initiated. Instead of explicit instructions, the reward function encourages the agent to make a decision, rather than instruct it. The agent, however, must first explore the environment in order to determine the rewards provided by being in each state, and choosing each action. Only then can the agent start maximizing its total cumulated reward. However, an action which returns a relatively small reward has the potential to give a much larger one further down the chain of actions, meaning that if the agent settles for a local maximum reward, it will never achieve a global maximum in the collected rewards. Hence the agent must keep exploring, while also exploiting the known rewards. This is an integral part of Reinforcement Learning, the need to explore the environment through trial-and-error, but also cumulate total rewards such that they are maximized [2].

The cumulated sum of rewards received during agent's lifetime (i.e. an episode) is represented by the Expected Return $\mathbb{R}_t$:

$$\mathbb{R}_t = \sum_{t=0}^{\infty} \gamma^t r_t \tag{2-1}$$

where γ is the discount factor ($|\gamma| < 1$), which determines the influence of future rewards on the current action selection- higher γ leads to a long-term look-ahead, increasing the weight of future rewards on the current action). r_t represents the reward assigned to state s.

Expanding Equation 2-1 leads to a recursive Equation 2-2, which provides a way to relate the returns of successive states.

$$\mathbb{R}_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots$$

$$\mathbb{R}_t = r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \cdots)$$

$$\mathbb{R}_t = r_{t+1} + \gamma \mathbb{R}_{t+1} \tag{2-2}$$

Policy Defining the interactions of states and actions in a structured way is done with the use of a policy, which is a mapping of the states to actions that are to be executed at all states. A policy is denoted by $\pi(s) : \mathbb{X} \mapsto \mathbb{A}$; it is an instruction for the agent to take the action $a = \pi(s)$ when in state s.

Exploration in RL dictates for a policy to have a certain degree of randomness. This concept is defined in the so-called ε-greedy policy, which picks a random action (from the available pool of actions) $\frac{\varepsilon}{100}$ of the time. On the other hand, a greedy policy omits the exploration, and can be used post-ε-greedy run to fully exploit an already-explored environment. Other less optimal policies exist, such as a fully random action generator, but as the results produced cannot be optimal, these alternatives are not taken into further consideration.

Value Function While the rewards provide an immediate feedback about the quality of action taken by the agent, a Value Function provides information about the long-term quality of a specific action while in a specific state. The Value Function $V^{\pi}(s)$ gives the expected return for the agent starting in state s and following policy π:

$$V^{\pi}(s) = E_{\pi}\{\mathtt{R}_t | s_t = s\} \tag{2-3}$$

As is the case with the optimal policy, an optimal value function, denoted by $V*$, is going to produce the maximum cumulated reward in agent's lifetime.

Action-Value (Q)Function The Action-Value Function, commonly referred to as Q-function, represents the value of state-action pairs, as opposed to the Value function relating only to the states. By including the specific actions at each state, the agent is better suited for picking an action which would maximize the returns.

$$Q^{\pi}(s,a) = E_{\pi}\{\mathtt{R}_t | s_t = s, a_t = a\} \tag{2-4}$$

The Q-function, including the information on which actions at which state has the highest value, is more information dense than its corresponding Value Function. The result is a faster convergence rate and a more efficient optimal policy.

Bellman Equation The iterative setup of RL warrants a recursive process in receiving the return, which is achieved using the Bellman Equation. Bellman stated himself that "a solution is not merely a set of functions of time, or a set of numbers, but a rule telling the decisionmaker what to do; a policy" [19], referring to the Dynamic Programming (DP) model (Section 2-1-2) he was developing.

The Bellman equation combines the concepts of a value function and a policy, resulting in an iterative function which can express the value of a state for an action taken. This can then be used to discover the values of a policy.

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r(s,a) + \gamma V^{\pi}(s') \right], \quad \texttt{for all } s \in \mathbb{S} \tag{2-5}$$

The value $V^{\pi}(s)$ is given by the expected reward and the discounted future rewards $\gamma V^{\pi}(s')$, shown in Equation 2-5 in the innermost part. This total value is subject to transitioning probability $p(s', r|s, a)$, the product of which is itself subject to the weighted sum of probabilities of selecting an action $a \in \mathbb{A}$. An optimal policy would result in a Bellman Optimality Equation, shown in Equation 2-6

$$V^{\pi}(s) = \max_{a \in \mathbb{A}} \sum_{s',} p(s',r|s,a) \left[r(s,a) + \gamma V^{\pi}(s') \right], \quad \texttt{for all } s \in \mathbb{S} \tag{2-6}$$

Similarly, the Bellman Equations can be found for the Action-Value Function, as shown in Equations 2-7 and 2-8.

$$Q^{\pi}(s,a) = \sum_{s'} p(s', r|s, a) \left[r(s,a) + \gamma \sum_{a'} \pi(s', a') Q^{\pi}(s', a') \right], \quad \texttt{for all } s \in \mathbb{S} a \in \mathbb{A} \tag{2-7}$$

$$Q^{*}(s) = \sum_{s',} p(s', r|s, a) \left[r(s,a) + \gamma \max_{a \in \mathbb{A}} Q^{*}(s') \right], \quad \texttt{for all } s \in \mathbb{S}, a \in \mathbb{A} \tag{2-8}$$

2-1-2 Reinforcement Learning Methods

Dynamic Programming

Dynamic Programming (DP) is a set of algorithms designed to compute optimal policies for a perfect MDP model of the environment. Although rarely applicable in RL due to it being computationally expensive, the DP algorithms provide a theoretical basis for other methods to be introduced below.

DP works on the principle of estimating the state values for a defined policy (policy evaluation), which are then used to improve the policy itself (policy improvement) [2]. The Bellman Equation (Equation 2-5) is used in iteration to estimate the state value function, after which the policy is improved by acting greedily in selecting the action of highest return. A discretized Bellman Optimality Equation used in DP is shown in Equation 2-9

$$V_{k+1}(s) = \max_{a \in \mathbb{A}} \sum_{s'} p(s', r|s, a) \left[r(s,a) + \gamma V_k(s') \right] \tag{2-9}$$

Monte Carlo Methods

Monte Carlo (MC) methods are used to solve RL problems using averaging sample returns. That is, MC methods rely on sample data, simulated or from an actual environment. Using a sample made of corresponding states, rewards and actions, MC methods can learn value functions and policies, in episodic steps. The sampling nature of MC methods enables its use in scenarios where there is no complete model of the environment.

Compared to DP, MC methods are computationally more efficient, but are limited in practice due to their episodic updates- the value function is only updated one step at a time, once the episode is complete. This limits its use cases in real-time problems. Additionally, exploration is hard to integrate with the policy, since the policy is to select the currently best action.

Temporal Difference Learning

Temporal Difference (TD) Learning is a combination of ideas introduced in Dynamic Programming and Monte Carlo Methods. As in MC methods, in TD the agent is able to learn without a complete model of the environment. On the other hand, similarly to DP, TD Learning can estimate values based on the previous estimates (bootstrapping). An action-value function is created arbitrarily, usually as a zero matrix, which is then updated using a provided policy.

The value is updated using the reward received by the agent, and is achieved using the following recursive equation:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{2-10}$$

wherein α represents the learning rate ($0 < \alpha < 1$), which determines the influence of the new estimate on the current value estimate.

Learning through temporal difference is relatively slow, as only a single state is updated in each iteration. Eligibility traces combat this by keeping a memory of previous actions executed to get to the current state. These values are decayed each episode for all state-action pairs, apart from the current one, which is increased in eligibility. This method simulates a reinforcement in the value of the states of the current path taken, while decreasing the values of others, resulting in a quicker learning time, and further back-propagation of learning in the state values.

Briefly introduced in Section 2-2, Safe Reinforcement Learning (SRL) is the process of maximizing returns (analogous to flat RL), while obeying predefined safety metrics [20]. The goal of this research project is to create a SRL controller utilizing Potential Field Methods, discussed in Section 2-3. The following chapter will elaborate on ways PF Methods can be integrated with a flat Reinforcement Learning algorithm, such that the safety performance of the agent is improved.

Reinforcement Learning, on its own, requires an initial exploration of the environment in order to discover the reward function throughout the environment. While necessary for determining an optimal policy, exploration is also a notable risk in scenarios where the agent is a valuable asset, such as a UAV or an aircraft. In sensitive scenarios, the most optimal policy might put the agent in risk-prone situations. For example, the quickest path through a cliff-ridden terrain could feature moving alongside the cliff, and in an exploratory attempt the agent could hit that cliff, resulting in no positive learning. If instead, the agent got a reduced reward when approaching the cliff, it would learn a different, less optimal but safer policy. The resulting trade-off for a few additional steps would be an increase in the agent's survivability.

Potential Field Methods have previously been applied in guidance and navigation problems, but the idea of combining them with Reinforcement Learning has seldomly been explored [7], with previous implementations using either classic control theory (PID controllers), genetic algorithms, or similar alternative control methods.

Section 2-2-2 illustrates the methods used in creating SRL algorithms, followed by the integration of Potential Field Methods with the SRL methods in Section 3-1.

2-2 RL State of the Art

The above-discussed concepts of Reinforcement Learning are well-researched and provide a foundation for more complex implementations of learning agents. While RL performs well in theory, it faces challenges when applied to practical solutions. This is due to its inherent lack of safety created by the need to explore the environment, as well as the inefficiency in storing the state value information of excessively large environments.

The advantages of using RL far outweigh these drawbacks, as is evident by the continuous improvements in the research field.

2-2-1 Deep Q Learning

Applying RL algorithms to state-spaces of increasing dimensions leads to an exponential increase in computational power (and time) required to solve the problem at hand. This has been noted by Bellman [21] , who referred to the phenomenon as the "curse of dimensionality" [14]. Substantial effort has been put into solving this limitation, since real-world applications usually require large environments. Solving these environments becomes so time consuming that a real-time execution is computationally infeasible.

Deep Reinforcement Learning, or Deep Q Learning, is a solution that has emerged to address this. Namely, Deep Q Learning replaces the standard Q-matrix, which allocates the Q-values used in Temporal Difference Methods, with a convolutional neural network. This multi-layered neural network outputs a vector, $Q(s, a, \theta)$, where θ is the network parameter, such that the output presents a solution in terms of an action a to be taken when in state s.

The benefit of using Deep Q Learning is most prominent in identifying information from captured images. The most prominent research into learning from images is by Mnih et al. [22], and focuses on playing Atari games, with the input to the deep network being only the image on the screen, which gives a controller action output.

Deep Q Learning has seen limited applications in aerospace, so far having been applied to detecting imagery from a UAV-based camera [23], and even more limited in use in flight control for autonomous landing of UAVs on aircraft carriers [24]. Even in these applications, the modeling of environments is trivial, in the latter one taking into account only the effects of wind and wave motion. While Deep Q Learning has been proven to work and is being actively used in various applications, further research in the aerospace domain is necessary to demonstrate its viability.

2-2-2 Safe Reinforcement Learning

Safe Reinforcement Learning (SRL) concerns the inclusion of damage or risk avoidance on top of the existing exploration and exploitation strategies of a RL agent. In research, it is argued that RL is modeled after animal learning behavior [25], yet animals do not learn by blindly exploring their environments. Animals are pre-programmed for certain behaviors, such as the hypothesized fear of snakes in primates [26], or they learn under teacher supervision in early stages of life. Similarly, a RL agent would have a higher chance of survival, or3 success rate, if it is equipped with safeguards while learning.

SRL is formally defined as maximizing the expected return in Reinforcement Learning problems, while ensuring reasonable system performance and respecting safety constraints during learning and use of the algorithm. Defining the safety constraints is subjective, and different criteria can be used to determine whether an action or a policy is safe. Garcia et al. [20] categorized different approaches to SRL, and have split them into either of the 2 subgroups - modifying the optimization criteria or the exploration process. The policy can be constrained to disallow dangerous actions, by constraining the policy space, for example. Exploration can

be modified by giving advice to the agent in the early learning stages, or by providing initial knowledge in the form of a preloaded Q-matrix. With safety being the primary topic of this reseach project, these concepts are elaborated in greater detail in Chapter 3.

These approaches can be applied to almost any RL application, but are of particular importance to aerospace applications, where unsafe exploration can lead to unrecoverable and fatal scenarios. Previous research in RL for aerospace applications has largely focused on improving safety [1, 4, 16, 27], yet in most of the research it is suggested that potential improvements in online learning efficiency can be made. The most common problem that is identified is to do with the exploratory nature of RL, and the resulting tendency to reach unrecoverable situations, such as crashing or exceeding the flight envelope. Safe Reinforcement Learning techniques, such as pre-loading the value matrix or human intervention in dangerous situations, could show improvements in flight control performance.

SRL Methods

Various approaches have been found in literature on how to increase safety in RL. Garcia et al. [20] have classified these approaches in 2 general categories, namely modifying either the optimization criterion or the exploration process. Below are presented such methods of improving safety in Reinforcement Learning, particularly the ones which are suitable for aerospace models.

Optimization Criterion Modification

Optimization criteria refers to the expected return function, seen earlier in Equation 2-1:

$$\mathbb{R}_t = \sum_{t=0}^{\infty} \gamma^t r_t$$

The pre-defined goal in RL is to maximize the expected return, but flat RL does not concern with the involved risks in achieving the maximum return possible. A policy which results in a high return sum might have large variances in returns across episodes, which implies instability and hence risk. Mitigating this risk can be achieved in various different approaches, generally by modifying the parameters which define the optimality equation.

Firstly, for the purpose of this discussion, a risk-neutral criterion is defined as the expectation of the return:

$$\max_{\pi \in \Pi} E_\pi(\mathbb{R}_t) = \max_{\pi \in \Pi} E_\pi(\sum_{t=0}^{\infty} \gamma^t r_t) \tag{2-11}$$

Other criteria can be selected, such as the Value or the Q function; for simplicity the expectation of return will be used to demonstrate different methods of improving safety.

Worst-case Criterion

Introduced by Heger [3], in the worst-case criterion the objective is to find a policy in which the expectation of the return is maximized for the worst outcome possible during the learning process. This expectation is given by:

$$\max_{\pi \in \Pi} \max_{w \in \Omega^{\pi}} E_{\pi,w}(\mathbb{R}_t) = \max_{\pi \in \Pi} \max_{w \in \Omega^{\pi}} E_{\pi,w}(\sum_{t=0}^{\infty} \gamma^t r_t) \tag{2-12}$$

where Ω^{π} is a set of trajectories $(s_0, a_0, s_1, a_1, ..)$ that occur under policy π, and w represents a trajectory from this set. In a given situation with a high-reward trajectory which features risk (in the form of uncertainties or potential negative rewards) and a trajectory with limited risk but a lower reward, $E_{\pi,w}$ will be higher for the safer route. On the contrary, the most optimal path in flat RL would not take uncertainties into account, and choose the high reward/ high risk trajectory as optimal. The optimality is therefore changed to account for the risk.

This concepts has also been extended to Q-learning by Heger, formalized as $\hat{\text{Q}}$-learning, which utilizes the same minimax approach as is seen in the expectations of the return in Equation 2-12. Equation 2-13 demonstrates this.

$$\hat{Q}(s_t, a_t) = \min(\hat{Q}(s_t, a_t), r_{t+1} + \gamma \max_{a_{t+1} \in A} \hat{Q}(s_{t+1}, a_{t+1})) \tag{2-13}$$

This is essentially a lower bound of the original Q-learning function, which tends to make the decision making too precautious, resulting in the agent being exposed to more harm in pursuit of safety (for example, due to exhaustion). Gaskett [28] proposed an approach named β-pessimistic Q-learning, shown in Equation 2-14 which combines the pessimism of $\hat{\text{Q}}$-learning with the optimism of the standard Q-learning.

$$Q_{\beta}(s_t, a_t) = Q_{\beta}(s_t, a_t) + \alpha(r_{t+1} + \gamma((1-\beta) \max_{a_{t+1} \in A} Q_{\beta}(s_{t+1}, a_{t+1}) + \beta \min_{a_{t+1} \in A} Q_{\beta}(s_{t+1}, a_{t+1}))) \tag{2-14}$$

Experimentally it has been determined that for a value of $\beta = 0.5$, $\beta \in [0, 1]$, the equation reaches the same levels of pessimism as $\hat{\text{Q}}$-learning. The addition of a tunable parameter increases its applicability to problems, and as a result, the survivability of the agent.

Constrained Criterion

A different approach to ensuring safety compliance is to dynamically limit the policy space, depending on the environmental context. The constrained criterion extends a Markov process (MDP) such that an additional element is added to the existing tuple $[\mathbb{X}, \mathbb{A}, \mathbb{T}, \mathbb{R}]$ to include a constraining element $\mathbb{C}$, which needs to be satisfied. This criterion ensures that a selected policy π is within the allowable policy space Γ. Figure 2-2 illustrates the policy space $\Gamma \subset \Pi$, which contains the policies that satisfy the constraints $c_i \in \mathbb{C}$ [29].

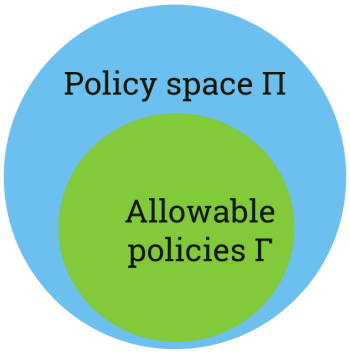


Figure 2-2: Restricting the available policy space [3]

The goal then is to maximize the expectation of return while respecting the constraints:

$$\max_{\pi \in \Pi} E_\pi(\mathbb{R}_t) \quad \forall \; c_i \in \mathbb{C} \tag{2-15}$$

or more concisely:

$$\max_{\pi \in \Gamma} E_\pi(\mathbb{R}_t) \tag{2-16}$$

The policy space Γ can be based on different constraining factors. A typical constraint used in literature is the minimal expecat ion of return [29, 30], with the Γ policy space consisting of all the policies that meet the criteria $E_\pi(\mathbb{R}_t) > \alpha$, where α is the tuning parameter for the minimal allowable return. Tamar et al. [31] based the constraint on variance, such that

The policy space Γ can be based on various constraints, such as ensuring a minimum expectation of return or a maximum allowed variance of return. Constraining the policy space will therefore decrease interactions by the agent with dangerous states.

Exploration Reduction

Instead of optimizing for the agent's policy, an alternative is to focus on the exploration process itself, and design it with risk preventing measures. In flat RL framework, the agent explores the environment non-discriminatively. To an extent, restricting exploration tends to improve the agent's probability of achieving the objective, but without sufficient exploration the agent can fail to discover the objective in the first place. A selectively restricted exploration process could provide the necessary exploration when needed, and a safer policy when in dangerous situations.

A common effort in modifying the exploration process is by implementing existing knowledge, either initially or through 'human teacher' guidance [20]. Providing initial knowledge can be done by using human inputs for the initial learning stage, such that the initial Value function already has a potential solution pre-programmed within itself. However, the agent utilizes exploration throughout the learning process, as is common in when using the popular ε-greedy policy, wherein the agent chooses to explore ε of the time (for example $\varepsilon = 0.1$ hence the agent explores 10% of the time) instead of exploiting the known high reward path. In this case, additional human input can be used either to prevent the agent from going into harmful states.

Similarly, the Value function can be derived from a set of demonstration runs, done offline by a human, before commencing the learning. This provides the agent with an initial safety net, but comes with challenges. Firstly, the autonomy of the agent is sacrificed, since the instructions are pre-programmed by a human. Furthermore, the agent might encounter situations that diverge from the initial demonstration, in which case it is once again exposed to exploring in unknown scenarios.

Lastly, the exploration can be modified such that there is always a human in the loop. This results in the agent seeking advice when necessary, or having a fail-safe mechanism in the form of a human override, which would prevent the agent from accessing dangerous states. This

can be limited to the initial exploration stage before the agent learns the policy. Clouse [32] proposes a policy in which the agent seeks advice when Q-values in the current state all have similar values. Quintia et al. [33] instead propose a policy in which a human can stop the exploration whenever they see the agent entering dangerous states, give a negative reward for that action and override it with their own action. These approaches proved to be effective, but they nevertheless depend on the human performance, and whether they deem an override necessary. Furthermore, this approach is time consuming as it requires a constant monitoring from a human operator.s

2-2-3 Safe Reinforcement Learning in Aerospace Control Systems

Safe Reinforcement Learning has seen an increasing interest in the aerospace research community [4, 6, 15, 16, 27]. Since pure RL algorithms lack the safety required to effectively solve an aerospace related problem, SRL methods present a possible solution.

SHERPA

Mannucci et al. [4] propose a SRL algorithm for aerospace applications which addresses problems of safety during exploration. The algorithm, named the Safety Handling Exploration with Risk Perception Algorithm (SHERPA), identifies measures to be taken to prevent fatal occurrences during exploration.

The approach is based on identifying a Fatal State Space (FSS), and deriving a Safe State Space (SSS) from the remainder. When an agent is approaching the FSS, a backup policy is initiated which is designed to bring the agent back to safety. This is done by identifying so-called Lead-to-Fatal (LTF) states, which although not a part of the FSS set, eventually would lead the agent to a fatal state with a probability of one. Hence during exploration, these LTF states must be avoided. In case a LTF state is reached, the 'ideal' backup action is performed so that the agent returns to an already explored state. A flowchart of SHERPA is shown in Figure 2-3, it demonstrates the decision making process when selecting an action. Multiple checks are made before the action is taken, and once it is, the information is stored in a new backup, replacing the previous one. Furthermore, if the iteration limit is reached, the agent automatically goes for the backed up strategy.

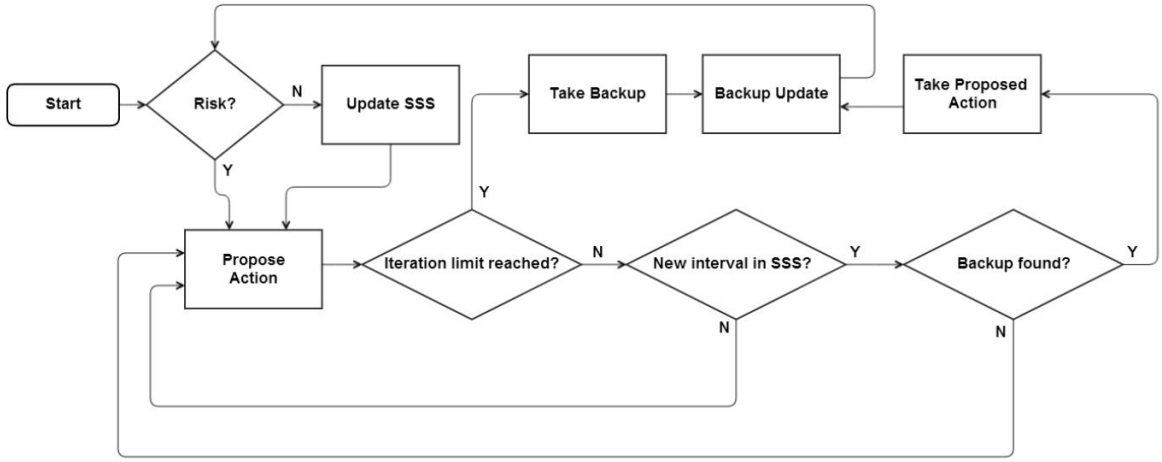


Figure 2-3: Flowchart of the SHERPA procedure [4]

Mannucci et al. conclude that this algorithm guarantees safety for an indefinite period of time. This approach can be used in conjunction with the Potential Field information to construct a safe algorithm for an aerospace application. The following section illustrates these algorithms.

2-3 Potential Field Methods

Using Potential Fields (PF) methods is a tried-and-true method of integrating obstacle avoidance in navigational algorithms. The principle of PF methods is based on using energy sources and sinks, which repel and attract the agent in an environment, respectively, in order to perform maneuvers around obstacles.

The strength of the field informs the agent about its surroundings. In order to utilize potential field information, the agent must be equipped with appropriate sensors that would enable the agent to determine the distance to obstacles. Since this project is focused on the algorithm itself, it is assumed that such a sensor is integrated in the system.

Potential Fields, also titled Potential Functions, Artificial Potential Field Method or similarly, can be constructed mathematically as follows:

1. The total potential at any given point in the environment is equal to the combined potential of positive and negative fields

$$U(x) = U(x)_{obs} + U(x)_{goal} \tag{2-17}$$

   with $U(x)$ representing the potential in state x

2. The obstacle states emit a negative potential field in their surrounding, with the field defined as follows:

$$U(x)_{obs} = \begin{cases} k_r \left[\frac{1}{\rho(x)} - \frac{1}{\rho_0}\right]^2, & \text{if } \rho(x) \leqslant \rho_0. \\ 0, & \text{otherwise.} \end{cases} \tag{2-18}$$

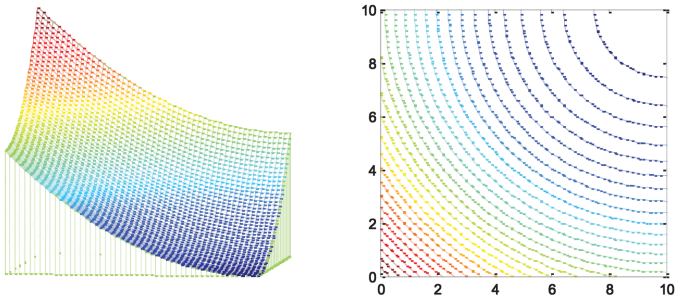
   where k_r is a tunable gain, $\rho(x) = ||x - x_{obs}||$ is the distance to the obstacle, and ρ_0 is a parameter which changes

3. The goal state emits an attractive potential field, defined as:

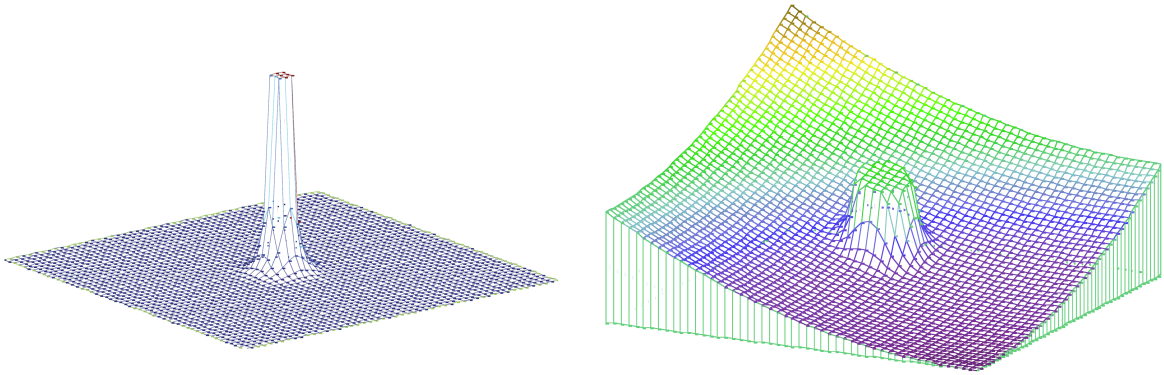
$$U(x)_{goal} = \begin{cases} \frac{1}{2} k_g \rho_{goal}^2, & \text{if } \rho_{goal}(x) \leqslant d. \\ d k_g \rho_{goal}(x), & \text{otherwise.} \end{cases} \tag{2-19}$$

   where k_g is a function gain, d is the drop-off radius after which the potential field drops in strength, and $\rho_{goal}(x)$ is the shortest distance between the agent's current state and the goal state.

The concept is visually illustrated in Figure 2-4 (adapted from [5]), with the goal and obstacle potential fields first illustrated individually in Figures 2-4a and 2-4b and the final composition of the two combined in Figure 2-4c. Note that in this example, the potential fields originate from only two sources, while a thorough analysis of the coupled interaction between repulsive and attractive potential fields would consist of multiple obstacles.



(a) Potential Field emmitted by the goal, $U(x)_{goal}$



(b) Potential Field emitted by an obstacle, $U(x)_{obs}$ **(c)** Combined Potenti1al Field of a goal and an obstacle

Figure 2-4: Example Potential Field composition, adapted from [5]

A common pitfall of PF methods is its inability to recognize an obstacle from afar, meaning it will only act on the information once it is sufficiently near an obstacle, or close to the goal. Another one is the tendency to get stuck in local minima, or situations where every state surrounding the current state leads to a higher potential, for example in a scenario where a horseshoe-shaped wall is present, as shown in Figure 2-5. Different methods of mitigating these scenarios have been proposed in literature [7, 34, 35], including backtracking out of a dead-end scenario, using harmonic functions which act dynamically based on the agent's behaviour, and mimicking water dynamics in flowing from places of higher potential to lower ones.

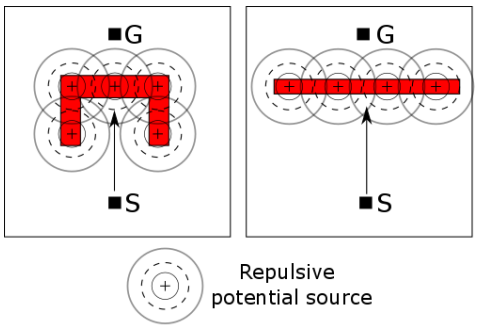


Figure 2-5: Dead-end due to a local minimum, agent going from the start (S) to goal (G) state [6]

2-4 PF Methods State of the Art

Potential Field Methods, having been formally introduced by Khatib in 1985 [17], have since appeared commonly in guidance-based research. The theory of potential fields has slowly evolved to address the above-mentioned challenges, with new approaches more suitable for real-world applications, and particularly the aerospace domain [5].

One of the criticisms of PF Methods was the inability to process obstacles fast enough to be used in real-time, resulting in oscillations when faced with confined spaces and multiple nearby obstacles [36]. This issue emerged with classic control theory, and could potentially be resolved with the use of Reinforcement Learning.

The most common challenge with PF Methods has been the local minimum scenario, a case where the agent gets stuck due to its surrounding states all having a lower potential than its current one. This happens if the agent finds itself in between walls, for example, or in edge cases in the environment. Modifications to a pure PF Method are made such that these cases are mitigated, by using various techniques such as Potential Gradient Descent Algorithm (PGDA) [37] and the waterflow method [7]. PGDA method proposed by Bounini et al. creates a new potential field whenever a local minimum is reached, assigning a negative potential value to the local minimum state, and iterating until the algorithm converges. Similarly, the waterflow method emulates the environment and the potential fields in it as pools of 'virtual water', whenever the local minima are reached, their potential is leveled out with the surrounding states' potential, thus eliminating that particular local minimum. The waterflow concept is particularly of interest since it uses Reinforcement Learning in combination with Potential Field theory to solve the given problem.

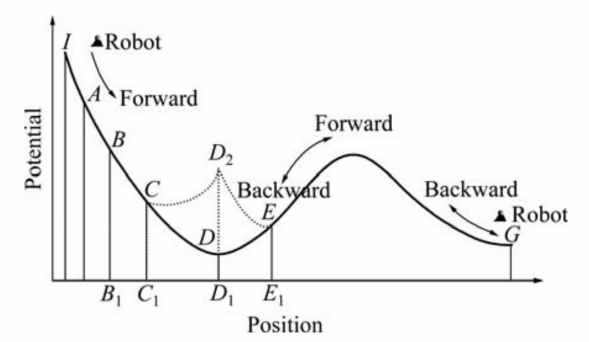


Figure 2-6: Virtual waterflow method of overcoming local minima. When the agent is surrounded by states of higher potential, the potential at that stete is incrased until it is leveled out with its surroundings [7]

Elsewhere, a solution to the local minima problem is found in Harmonic Potential functions [35, 38, 39], wherein the potential field is modeled as a solvable Laplace function. This constraint guarantees that local minima cannot spontaneously emerge. For example, a potential function $U(x, y)$ in a two-dimensional environment would need to satisfy the Laplace Equation:

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0 \tag{2-20}$$

Over a given region $\Omega \subset R^n$, the above Equation is satisfied iff U(x,y) attains its maxima and minima only on the boundary of Ω.

Harmonic Potential Functions appear in aerospace applications for UAV obstacle avoidance [40, 41], albeit utilizing conventional control theory. Scherer et al. [41] have seen robust results even in relatively high speeds for a small UAV (up to $10m/s$), but reported the same previously-mentioned oscillatory behaviour when faced with multiple obstacles.

Potential Field Methods have also been applied in Interface design for UAV control. Lam et al. [8] developed a haptic interface for remote UAV operations, wherein the human operator experiences haptic feedback forces on the controller based on the impending obstacles in the UAV's environment. Haptic information is constructed from an artificial force field, which maps obstacles in the environment to repulsive forces in the controller. The idea is to use the haptic feedback in order to compensate the lack of stimuli a UAV pilot has relative to an onboard pilot, while also unloading the visual information processing. Figure 2-7 is an insert from this specific research, showing the agent (UAV) as the circle at origin. The difference in contours between the two examples comes from the adjustable parameters being able to control the size, shape and gradient based on the agent's velocity.

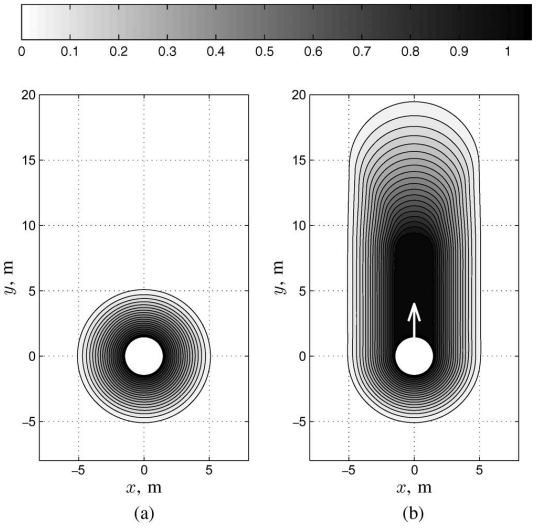


Figure 2-7: Potential Fields for Haptic Feedback, (a)$v = 0m/s$, (b) $v = 4m/s$, adapted from [8]

2-5 Chapter Summary

This chapter presented a literature study into Reinforcement Learning concepts and Potential Field Methods. Reinforcement Learning is an algorithmic approach to decision making, which tries to model action taking on animal behaviour, by providing the agent, or learner, with rewards for achieving a predetermined goal. The agent is tasked to explore the environment it is in, in the meantime creating a map of behaviour that is good and rewarded, and bad and punished, such that a desired behaviour is learned over time, without a direct instruction to do or follow specific orders. The end result for the agent is to maximize the cummulated rewards received throughout its lifetime. Although exploiting the environment will result in potentially highest cummulated rewards, it needs to be balanced by exploration of the environment, otherwise the agent might not learn the optimal policy. The current state of the art in RL features a heavy focus on improving the safety of the agent during learning; due to the nature of RL the agent inevitably needs to explore the environment, potentially leading to the agent finding itself in a dangerous situation. Advances in safety of the agent make RL more suitable to applications in aerospace systems.

Potential Field Methods is a common guidance tool used for navigational purposes, specifically in obstacle avoidance. PF Methdods are used to model obstacles as energy emitting points in

the environment, and goals as energy-absorbing elements. The result map of the environment is then sensed by the agent, which can then be used in navigating around it.

These two separate theories have seldom been combined, even less so for purposes of flight control, although some efforts have been made in using it for the purpose of creating a safer learning. This Chapter covers the first Research Question, **RQ1**, from Section 1-3, addressing the current state of the art of the research field, in terms of both Reinforcement Learning and Potential Field Methods.

Chapter 3

Safe Reinforcement Learning using Potential Field Methods

The following chapter explores possible methods of integrating Potential Field Methods with Reinforcement Learning to produce a Safe(r) Reinforcement Learning algorithm. The SRL methods described in Section 2-2-2 are to be combined with the basic principles of Potential Fields, Section 2-3, to create an algorithm which can be compared with pure RL methods.

3-1 Potential Field Integration

The theory described so far in Chapter 2 gives a basis for integrating Potential Field information within a Reinforcement Learning algorithm, with a goal of improving the safety of the agent during the learning process. PF information can be integrated in different ways- it can be designed to directly affect the reward or the Value function, or to modify the exploration process, such that when in dangerous states the randomness is decreased or the action selection is overridden by a fail-safe mechanism.

3-1-1 Reward and Value function based Potential Field Integration

The PF information, in the form of a potential function $U(s)$ representing the strength of the field sensed at a particular state s, can be incorporated at the reward function level, such that the total reward received in a particular state reflects the presence of a potential field.

For a simple reward function being equal to $r(s)$ for a state s, the modified reward function will take the form of:

$$r_{tot}(s) = r(s) + k \cdot U(s) \tag{3-1}$$

$r_{tot}(s)$ now represents the new total reward for a particular state s, with k being a tuning parameter. This tuning is required to normalize the potential field strength with other rewards, otherwise the potential field would have very little impact, or cause the algorithm to act sporadically.

Reward-based Potential Field integration can prevent the agent from entering dangerous states, by giving a variable amount of negative feedback when approaching these states. However, the due to delayed rewards, as well as the discount factor being present when evaluating the Expected Return (Equation 2-1), the method can further be improved. Furthermore, modifying the reward function in this way can result in the agent prioritizing the potential information over other inputs from the environment [2], so a better way to impart this information is in the value function or the policy.

Using the Value function (Equation 2-6) as a starting point, introduced as a Bellman Equation in Section 2-1-1:

$$V^{\pi}(s) = \max_{a \in \mathbb{A}} \sum_{s',r} p(s', r|s, a) \left[r(s, a) + \gamma V^{\pi}(s') \right], \quad \texttt{for all } s \in \mathbb{S}$$

In the above equation, the reward R_t is iterated through the $V^{\pi}(s)$ recursion. It follows that if Potential Field information is injected at the Value function level, it would result in a more direct approach:

$$V^{\pi}_{tot}(s) = V^{\pi}(s) + k \cdot U(s) \tag{3-2}$$

Compared to the reward-level integration, the Value-based PF integration will ensure a quicker learning rate since the information does not pass through the decay factor γ. Furthermore, this ensures that the reward function is left with the sole purpose of describing the wanted end result, with which PF information will not interfere.

Q function-based Potential Field Integration

Similarly to the above-mentioned Value function modification, the Action-value (Q) function can be augmented to accommodate the potential field information. An added benefit of using the Q-function is the additional information about a particular action being selected. Therefore, the potential field information can be based not only on the current state, but also on the action at that state.

The Q-function takes the following form, similar to the iterable equation described in the Temporal Difference description in Section 2-1-2.

$$Q(s, a) \leftarrow Q(s, a) + \alpha[R(s) + \gamma \max_{a \in \mathbb{A}} Q(s', a') - Q(s, a)] \tag{3-3}$$

Addition of PF information is done right after performing the above iteration, by adding the value of the potential at the current state.

$$Q(s, a) = Q(s, a) + k \cdot U(s, a) \tag{3-4}$$

Comparing the above equation to the Value function PF integration, Equation 3-2, it can be noted that the Q-function PF integration has the potential for higher accuracy due to the action component being included. The result is that, at a particular state the PF information can differ depending on the action taken. For example, a particular state might be under the influence of a PF emitting from a nearby obstacle, but if the action is driving the agent away or alongside the obstacle, the PF can be lower than if the agent were heading directly into the obstacle.

3-1-2 Action space Potential Field Integration

The implementations so far have been based on augmenting the value matrices, which are, although effective, limited in scope. These algorithms can further increase the effectiveness of the use of PF information, since the policy dictates the decision making at a more direct level relative to the value and Q functions.

Backtracking

Backtracking is a policy based on the ε-greedy policy, with PF information used to backstep out of dangerous situations. In normal conditions, the policy performs in the same way as a normal ε-greedy policy, shown in Algorithm 1

Algorithm 1 ε-greedy policy

```
1: procedure EPSILON_GREEDY(Q_matrix, state, ε)
2:     if rand[0,1] < ε then
3:         return rand.choice(1,4)
4:     else
5:         return action = argmax(Q_matrix[state])
6:     end if
7: end procedure
```

The additional PF information is used to expand the ε-greedy policy by reducing the exploration rate when a certain PF strength is identified, and if the agent yet experiences a stronger potential, backtracking is initiated based on the information of the previous action and previous state. The action taken to arrive in this state is also rewarded with a highly negative reward, so that it is not repeated in future episodes.

Algorithm 2 Backtracking ε-greedy policy

```
procedure EPSILON_GREEDY(Q_matrix, state, ε, environment, last state, last action)
    extract PF info from env,state
    if PF low then
        if rand[0,1] < 0.5 * ε then
            return action = rand.choice(1,4)
        else
            return action = argmax(Q_matrix[state])
        end if
    end if
    if PF high then Initiate backtrack
        Q_matrix[last_state][last_action] -= reward
        return action = opposite(last_action)
    end if
    else execute standard ε-greedy policy
end procedure
```

Softmax

The softmax policy is based on the concept of the softmax function, which has an input of multiple vectors and output of a normalized probability distribution of the input vectors. The output vectors will all be in the interval (0,1) and their sum will be equal to 1, all the while the magnitudes of the input vectors will be comparatively the same with the output ones. This concept is often applied in neural networks due to the necessity to order the parameters in a consistent interval, but have been applied in some Reinforcement Learning policies [6,42].

The softmax policy is shown in Equation 3-5 below, with τ being a temperature parameter, which controls the spread of probabilities- as τ approaches 0 the output probabilities will be more sparsely separated, and as it goes to infinity the probabilities approach being equal to each other:

$$p(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{b=1}^{n} e^{\frac{Q(s,b)}{\tau}}} \tag{3-5}$$

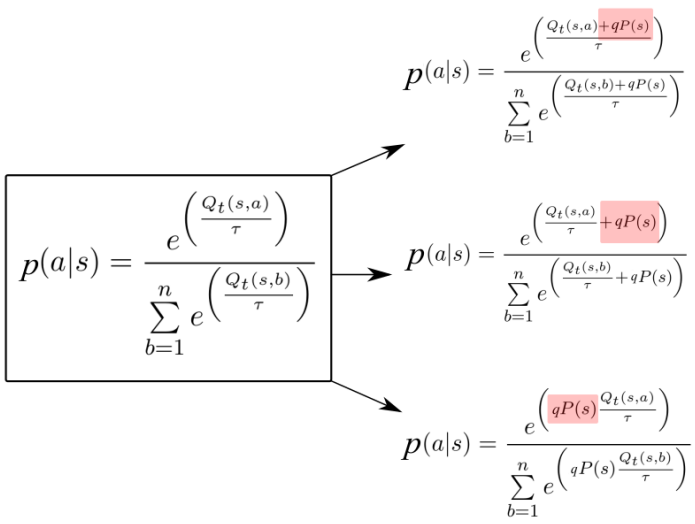


Figure 3-1: Integrating Potential Field information within the softmax policy [6]

Integrating the PF information within the softmax policy can be done in multiple ways. Figure 3-1 illustrates some of the potential solutions- with $P(s)$ representing the PF information, and q the tuning parameter for the relative strength of the PF. Care has to be taken so that the PF information is neither overwhelming the pre-existing Q-function values, nor that it is negligible relative to it.

3-2 Chapter Summary

This chapter introduced the concept of Safe Reinforcement Learning and the methods used in creating such algorithms. The methods were then applied in implementations featuring Potential Fields. Different approaches were shown with respect to the level of PF information integration, whether this information is integrated on a reward or Q-function level, or on a policy level. These algorithms are to be used in the following chapter to produce simulations and preliminary results, showcasing their relative performance to a flat RL implementation.

Chapter 4

Simulation Study into Reinforcement Learning with Potential Fields

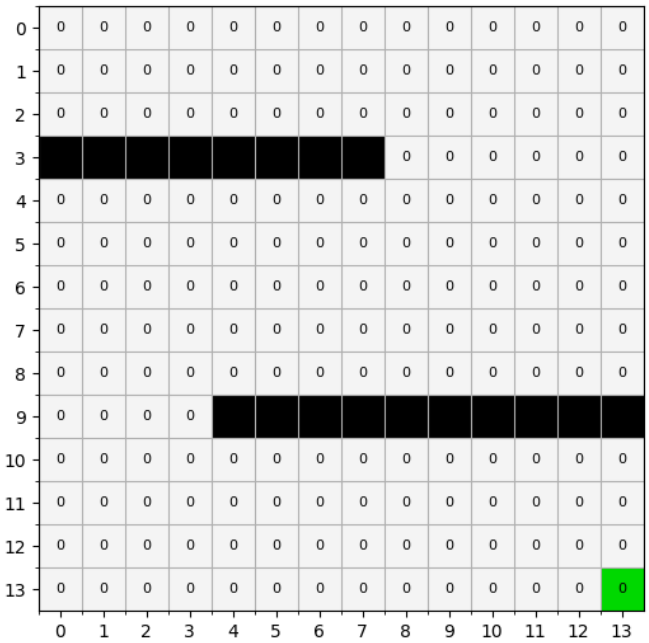


Figure 4-1: $n \times n$ gridworld used in preliminary simulations; black boxes are impermeable obstacles, green state represents the objective

A simulation study is created to evaluate the feasibility of the approaches proposed in Chapter 3. The simulation is a two dimensional gridworld environment, shown in Figure 4-1, featuring

obstacles around which the agent navigates in order to reach the goal state. Each step taken results in a reward of -1, even in boundary cases where the agent attempts to go leave the environment, in which case it stays in the same state and receives the negative reward. Hitting an obstacle terminates the current episode and results in a reward of -100 and reaching the goal state gives the agent a reward of 143.

The potential field theory presented in the previous chapters is simplified for the purpose of this simulation, and a discretized model is made such that the obstacles (black states in the gridworld) are surrounded by states with varying negative rewards- closer to the wall equating to a larger negative reward. Figure 4-2 illustrates the potential field states, with yellow and red states carrying a reward of -2 and -3, respectively.

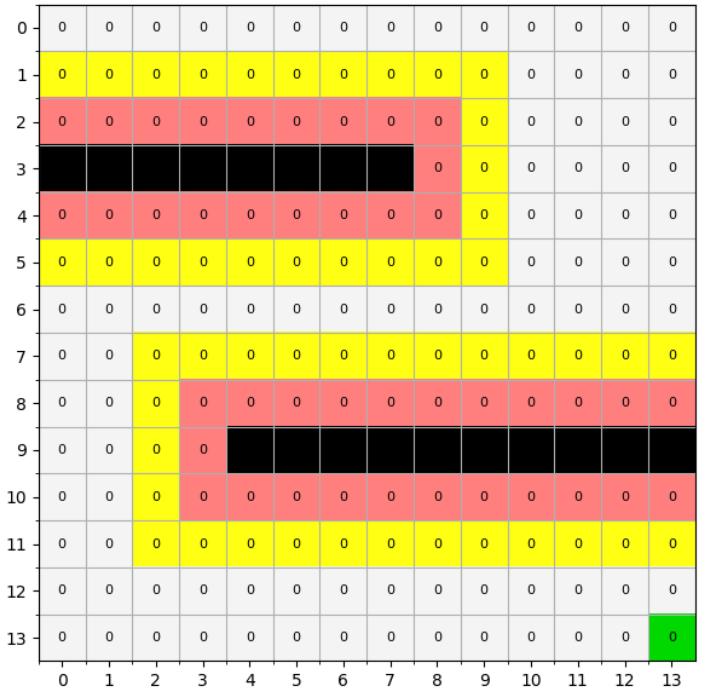


Figure 4-2: $n \times n$ gridworld used in preliminary simulations; black boxes are impermeable obstacles, green states represents the objective, yellow and red states indicate the potential field information

4-1 Preliminary Results

Different implementations of Potential Fields within Reinforcement Learning, as described in Section 3-1, are quantitavely compared below. The comparison is based on multiple factors, namely the total collected reward per episode, the number of steps per episode, the end-of-episode trigger, and the distribution of state visits.

The cummulated reward per episode presents a straight-forward performance comparison, given that maximizing cummulated rewards is a primitive concept of RL. Putting a high negative reward on collisions gives the agent incentive to not repeat such behaviour, albeit it is unwanted to begin with.

The step count measure the convergence of the algorithm to the minimum steps required to find the objective. It is expected that the step number is high during the initial period of learning, due to the exploratory nature of RL, hence all of the algorithms will show a decreasing trend in this metric. The end-of-episode trigger, which can either be collision, exhaustion (reaching maximum steps per episode), or reaching the goal state, provides further detail when accompanied by the step count analysis.

Lastly, tracking the visits of each state can clearly visualize agent's behaviour, and demonstrate the performance of the augmented algorithm relative to the pure RL implementation.

4-1-1 Nominal Conditions and Simulation Setup

A nominal condition is chosen such that the performance of the simulation is maximized. The exploration rate, for example, should be set as low as possible, while the learning and decay rates should be as high as possible. These statements are based on literature, and will be further examined in a sensitivity analysis in Section 4-2

Table 4-1: Nominal Simulation Conditions

	Nominal condition
Learning rate, $\alpha \in [0, 1]$	0.9
Decay rate, $\gamma \in [0, 1]$	0.9
Exploration rate, $\varepsilon \in [0, 1]$	0.1
Steps per episode	300
Episodes per run	300 ε-greedy, followed by 50 greedy

4-1-2 Reward and Value Function PF integration

A comparison is made between the implementations of Potential Field information at either the reward or the value (Q) function. The results are compared against a pure RL solution, which features no PF information.

Figure 4-3 shows the reward function comparison. The data from each run is averaged across the episodes using a rolling mean to better illustrate the progressive improvement in cummulating rewards. As is evident from the figure, both reward and Q function PF implementations show improvements in convergence speed, i.e. they maximize the cummulated rewards sooner than pure RL. Consistency of results is also improved, since the pure RL algorithm periodically sees a drop in performance due to exploration in later stages, which also leads to collisions.

Figure 4-4 illustrates this consistency more clearly- as it can be seen in the topmost graph the collisions are more present throughout the entire lifetime of the agent, while adding PF information results in no collisions past episode 71.

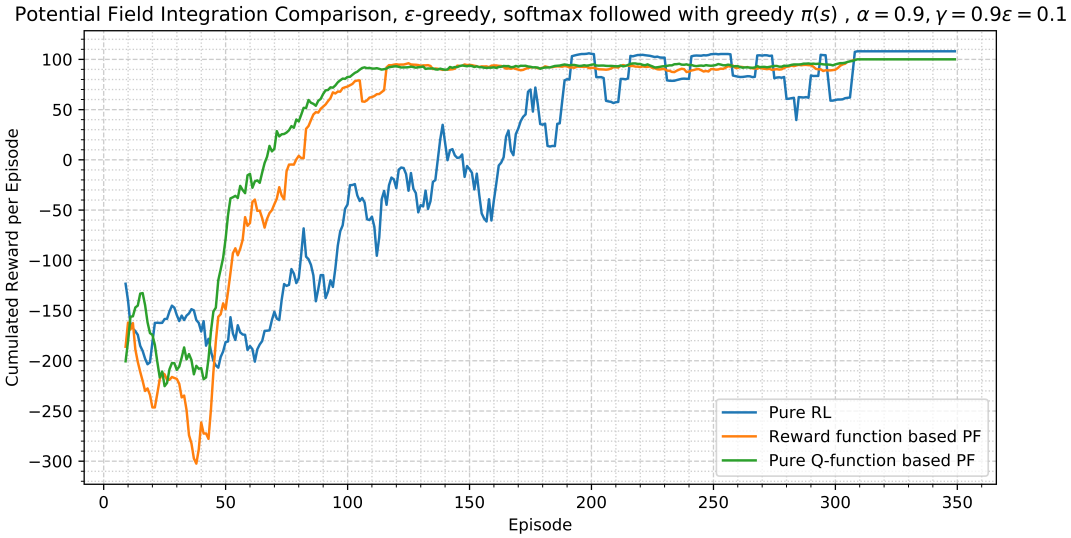


Figure 4-3: Cummulated rewards per episode (Smoothed average over 10 episodes)

Figure 4-5 shows the steps taken per episode, with a rolling average applied. Interestingly, the number of steps first increases before converging to the optimal path. Together with the end-of-episode trigger information from Figure 4-4, it can be seen that this is due to episodes ending in a collision with an obstacle. The Q-based PF (QPF) algorithm outperforms the reward-based PF (RPF) and pure RL algorithms in both the steps taken as well as cummulated reward. Peculiarly, RPF is the only one reaching the maximum allowed steps per episode (pure RL does reach the 'exhaustion' limit once according to Figure 4-4), while pure RL and QPF have a somewhat similar progression of learning. Possible explanation is that the learning is slower in RPF due to the conflict of experiencing the rewards from the potential fields, as opposed to only receiving a reward upon reaching the goal state. It is, however, clear that Potential Field information improves the performance of thea algorithm, as the pure RL takes the highest number of episodes to learn the correct path around the obstacles.

Pure RL finds the shortest route, as can be seen in it taking the least number of steps to objective in the greedy part of learning (last 50 episodes). The occasional dip below the optimum number of steps also shows that the agent still collides with obstacles late in the learning process, whereas PF-based solutions stay above their optimum number of steps taken until reaching the greedy stage. This is due to the agent still exploring the environment 10% of the time, leading it to sporadically go off-course, but still avoid obstacles due to the potential fields. The heatmaps shown in Figure 4-6 illustrate show the shortest path taken, since this path inevitably has the most state visits. In this figure, a clear outline of potential fields can be seen in the center and right heatmaps, since these states will get visited less frequently, compared to the pure RL heatmap where the distribution of state visits is more uniform.

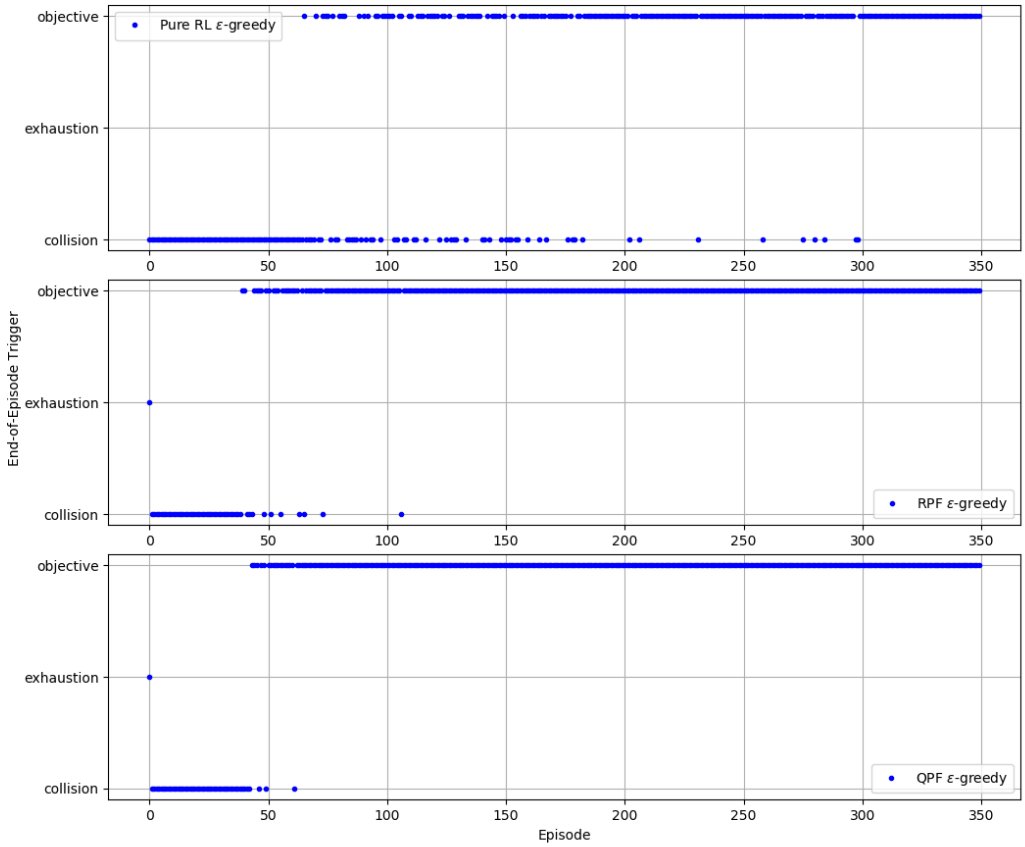


Figure 4-4: End of episode trigger

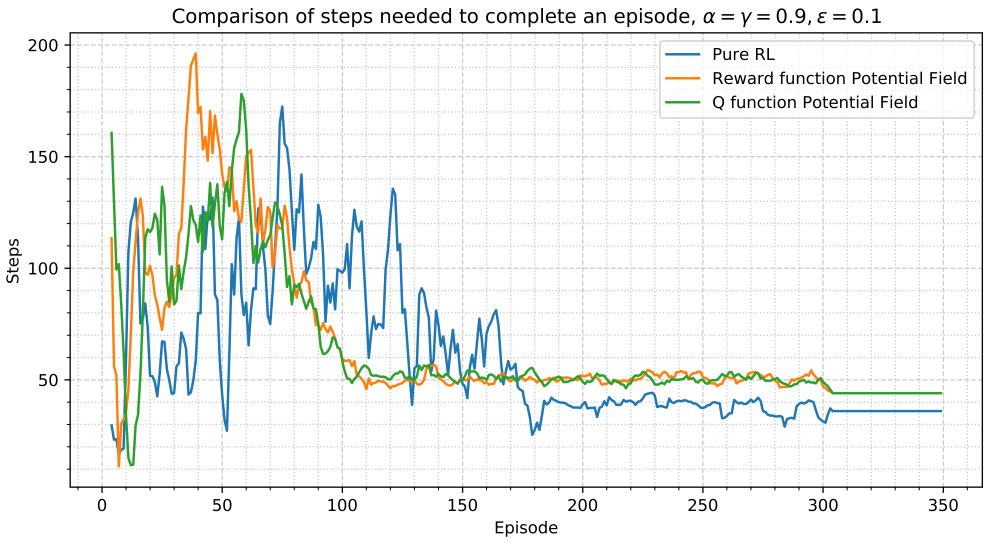


Figure 4-5: Steps taken per episode (Smoothed average over 10 episodes)

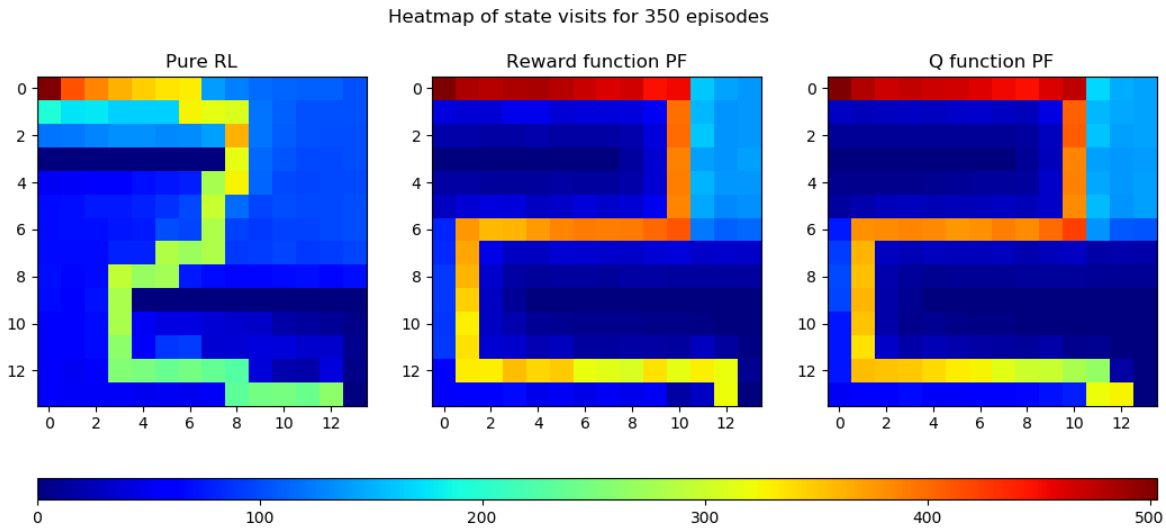


Figure 4-6: Heatmap of state visits, overlaid on the gridworld environment

4-1-3 Policy based PF integration

Having determined that the Q-function based PF integration outperforms both pure Reinforcement Learning and reward function based PF, it is chosen to be used for further development and analysis. In this section, QPF algorithm is used to compare different policy implementations. Namely, the previously used ε-greedy policy is compared with backtracking ε-greedy and softmax-based PF policies.

Comparing the performance of softmax implementations

Before comparing the performance of the different algorithms, multiple ways of integrating PF information within a softmax policy have been described in Section 3-1-2, Figure 3-1. Previous research [6] has established that multiplying the PF with the Q-function values is ineffective, therefore the other two methods of integration are to be compared, namely:

$$p_1(a|s) = \frac{e^{\frac{Q(s,a)+q\cdot P(S)}{\tau}}}{\sum_{b=1}^{n} e^{\frac{Q(s,b)+q\cdot P(S)}{\tau}}} \tag{4-1}$$

and

$$p_2(a|s) = \frac{e^{\frac{Q(s,a)}{\tau}+q\cdot P(S)}}{\sum_{b=1}^{n} e^{\frac{Q(s,b)}{\tau}+q\cdot P(S)}} \tag{4-2}$$

The only difference between these two implementations is whether the temperature parameter affects the PF information. For the sake of brevity, Equations 4-1 and 4-2 will hereinafter be called softmax1 (p1) and softmax2 (p2) respectively. Table 4-2 shows the nominal conditions chosen for this simulation. τ and q values have been tuned such that the potential field information is neither overwhelming nor negligible in comparison to the Q-values attained during learning.

Table 4-2: Softmax Nominal Simulation Conditions

	Nominal condition
Learning rate, $\alpha \in [0, 1]$	0.9
Decay rate, $\gamma \in [0, 1]$	0.9
Exploration rate, $\varepsilon \in [0, 1]$	0.1
Steps per episode	300
Episodes per run	500 ε-greedy, followed by 50 greedy
τ	0.1
q	10

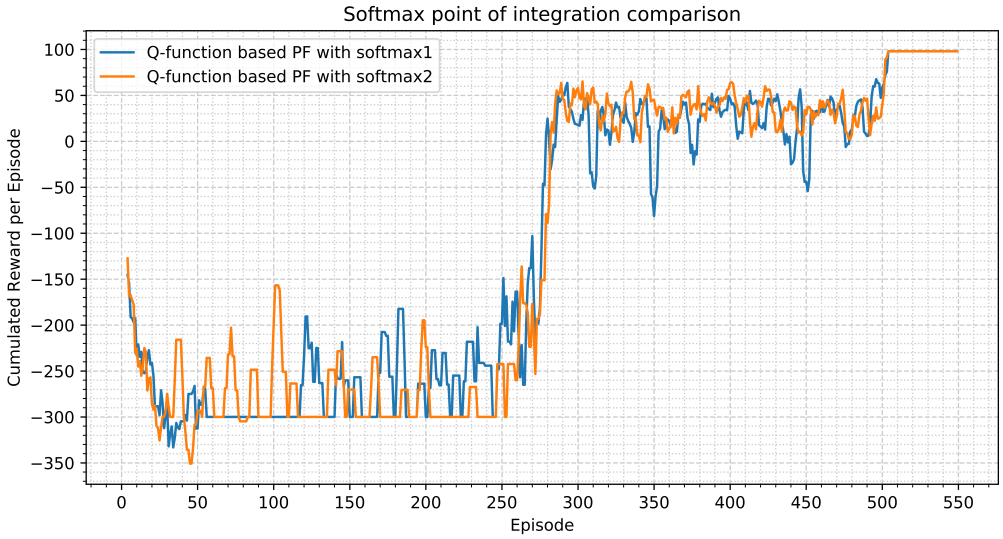


Figure 4-7: Softmax point of PF integration comparison, p1 in blue and p2 in orange, cummulated rewards per episode (Smoothed average over 5 episodes)

Figure 4-7 shows a comparison in cummulated rewards of the two differing softmax implementations. The temperature and PF tuning parameters have been set to $\tau = 1, q = 10$ in order to equalize the PF influence with the Q-values. The total number of episodes has been increased from the nominal conditions to 500 episodes (+50 greedy episodes) due to the slower performance of the softmax algorithm relative to ε-greedy, and the smoothing average has been decrease to 5 episodes to provide a more detailed comparison. The performance of softmax2 algorithm is marginally better, albeit very similar to softmax1. In the first part, up until episode 250, softmax 2 achieves higher rewards and has more episodes in which the step limit is not reached (the reward being -300 means that the maximum number of steps has been reached and the episode is terminated). Furthermore, softmax2 has less variations in results in the second part (episodes 300-500). Therefore, softmax2 will be used in the remained of the comparison. It is also noted that if the maximum number of steps has been increased the performance might improve, but for the sake of comparing the results with the other implementations it is kept constant at 300 steps per episode.

Figure 4-8 shows the cummulated reward comparison. With the earlier softmax analysis, it was to be expected that the QPF and QPF with backtracking algorithms would outperform the softmax implementation. Interestingly, once the greedy policy is deployed after episode 300 the softmax policy does converge to the optimal solution, which signals that the Q-values derived from the softmax policy are similar to those of the other two policies. Possible explanation for the lack of performance can further be found in Figure 4-9, where the end-of-episode trigger for the majority of the softmax episodes is exhaustion. Reaching the goal state sporadically does mean that the Q-values are adjusted toward the solution, but the policy's ineffectiveness likely comes from getting stuck between obstacles. This was also exhibited by other algorithms, in Figure 4-6 the light blue states indicate the agent spent more time in those states relative to darker blue ones, but nevertheless it eventually learned to complete the path to the goal, unlike when softmax is used.

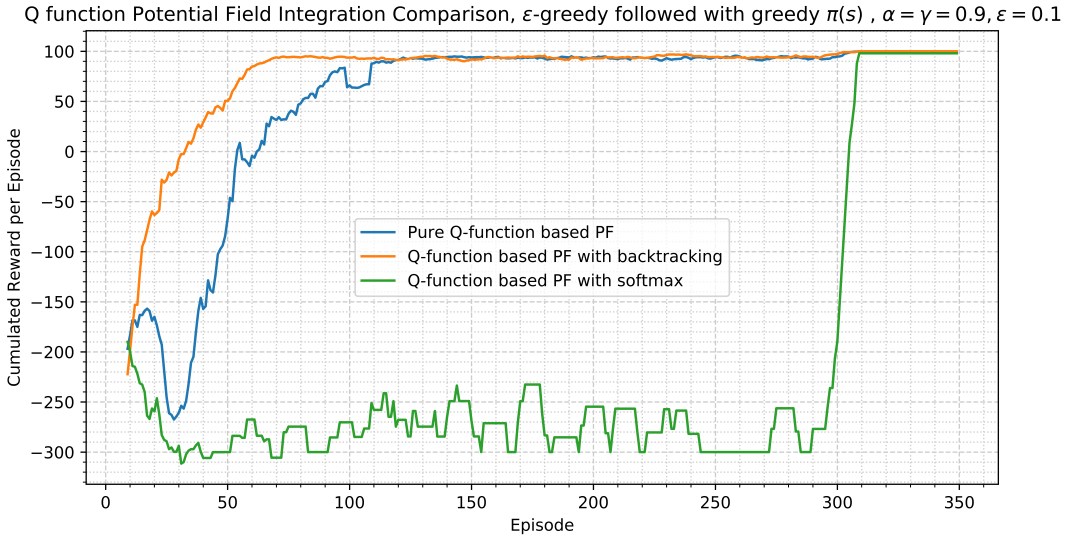


Figure 4-8: Cummulated rewards per episode, policy PF integration (Smoothed average over 10 episodes)

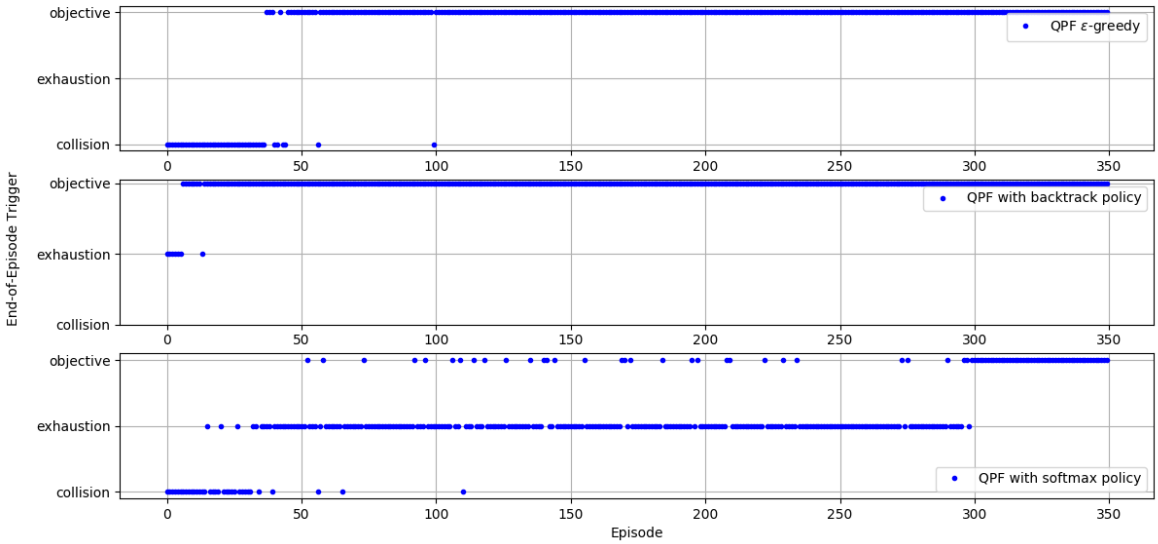


Figure 4-9: End of episode trigger, policy PF integration

Further analysis demonstrated that deviating from the nominal conditions when using softmax policy gives results that are similar to the other algorithms. Specifically, setting the learning rate to $\alpha = 0.8$ and decay to $\gamma = 0.95$ gives a convergence using softmax policy within the maximum number of episodes. Therefore, these values were used the other algorithms so that an equal comparison could be made.

Comparison of ε-greedy, backtracking and softmax policies

In Figure 4-10 the adjusted α and γ values are used to draw a comparison between the standard ε-greedy, backtracking ε-greedy and the softmax policies.

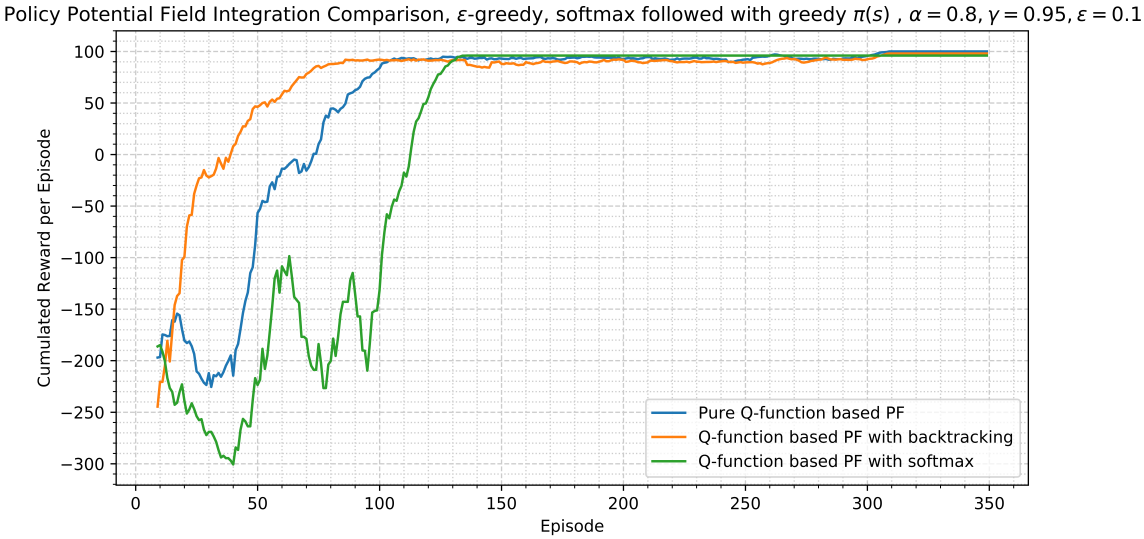


Figure 4-10: Cummulated rewards per episode, policy PF integration with adjusted parameters (Smoothed average over 10 episodes)

While the softmax policy still underperforms relative to the ε-greedy policies, the updated parameters give it comparable performance. The softmax policy does perform better in terms of variations once a correct policy is learned- once it converges there exist no deviations (episode 150 onwards).

The backtracking QPF policy outperforms the ε-greedy and softmax policies, and converges in the least number of episodes of the three. Interestingly, all three policies produce different optimal paths, as can be seen in the heatmap of state visits in Figure 4-13. The backtracking policy learns to avoid a particular state, as seen in the middle heatmap, due to the decay from entering into the potential field region, and subsequently being forced back and given a large negative Q-value for those actions. The softmax policy, on the other hand, does not utilize random exploration, once it has learned how to get to the goal it no longer explored the environment, choosing only to exploit the known path.

Figure 4-11 confirms the backtracking being the best performing algorithm in terms of safety. Having no collisions makes this algorithm the most suitable for aerospace applications of the three compared here. The limited episodes which end in exhaustion could still be reduced, either by changing the basic RL parameters, or augmenting the policy further.

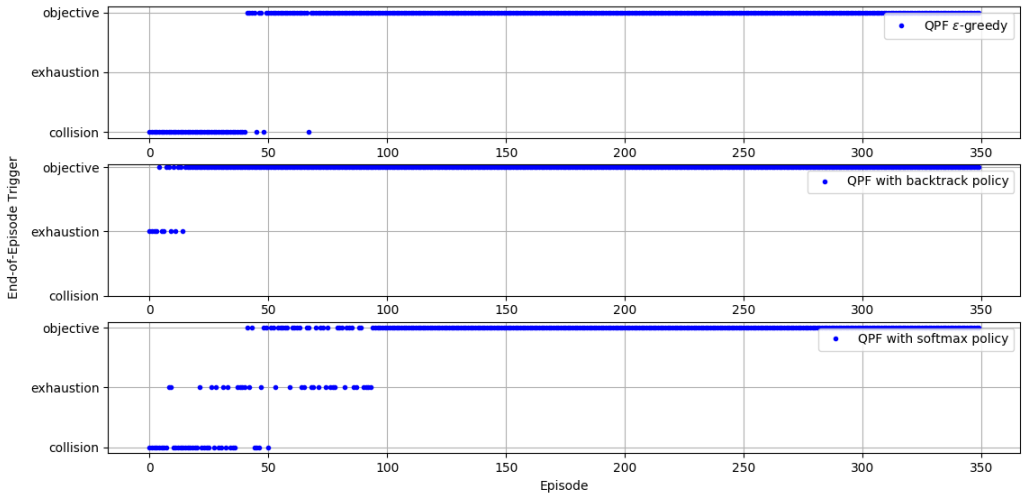


Figure 4-11: End of episode trigger, policy PF integration with adjusted parameters

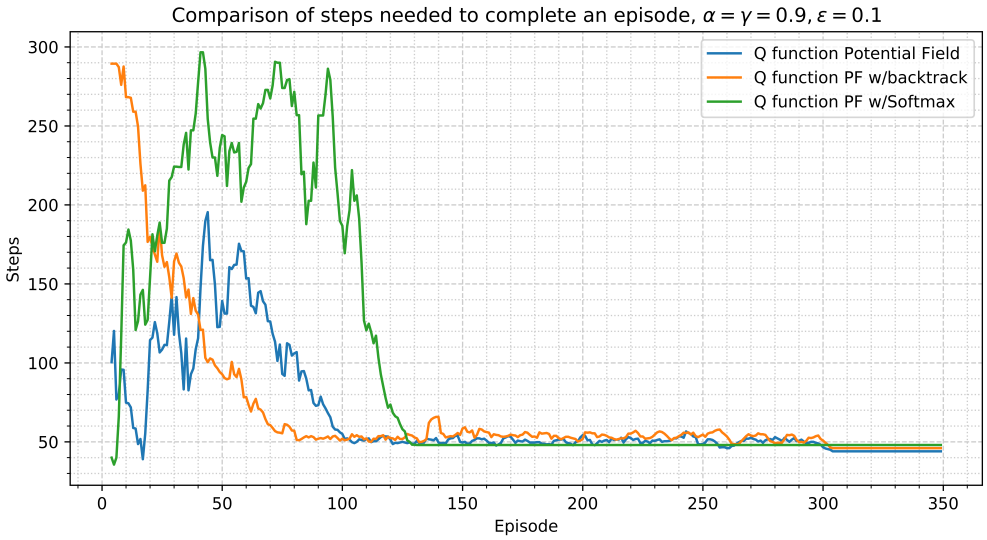


Figure 4-12: Steps taken per episode, policy PF integration (Smoothed average over 10 episodes)

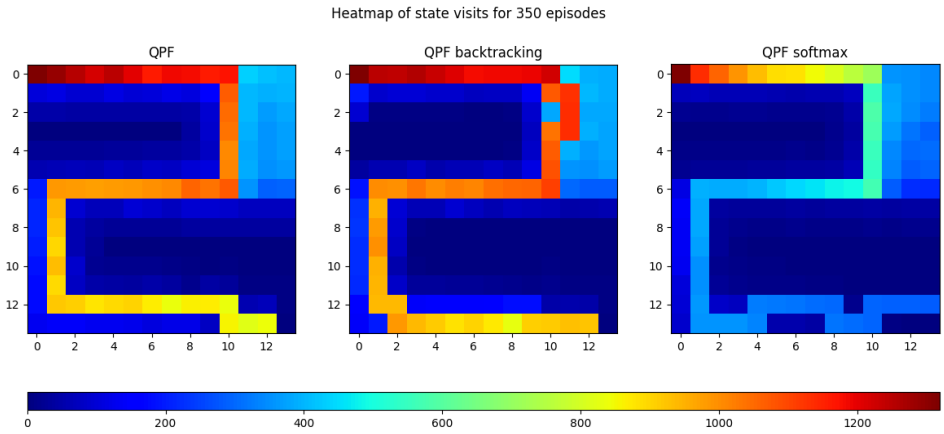


Figure 4-13: Heatmap of state visits

4-2 Sensitivity Analysis

A sensitivity analysis is conducted to determine the effects of changing the learning parameters. The parameters $\alpha, \gamma, \varepsilon$ are varied individually, in the interval (0,1) in increments of 0.01.

Based on the findings in the previous section, the nominal conditions for the parameters are described in Table 4-3.

Table 4-3: Sensitivity Analysis Nominal Simulation Conditions

	Nominal condition
Learning rate, $\alpha \in [0, 1]$	0.8
Decay rate, $\gamma \in [0, 1]$	0.95
Exploration rate, $\varepsilon \in [0, 1]$	0.1
Steps per episode	300
Episodes per run	300 ε-greedy, followed by 50 greedy

The comparison is based on the total cummulated rewards collected and steps taken during the entire run per each parameter varied. In other words, the rewards and steps are added up not for a single episode, but for all of the episodes in a given run. This determines the overall performance for a given parameter.

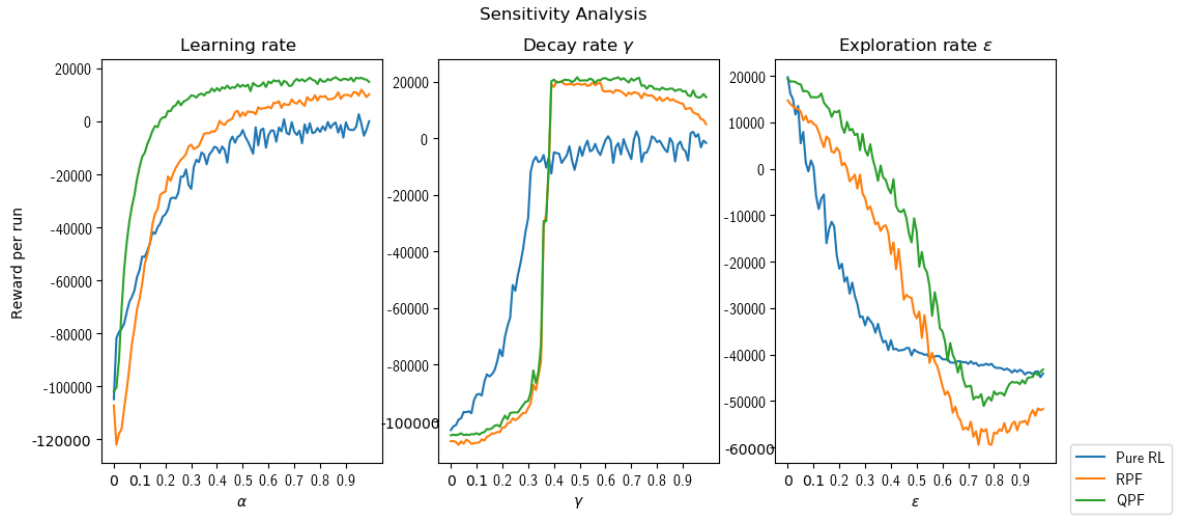


Figure 4-14: Sensitivity Analysis, reward per run

Figure 4-14 shows the effects of the variation of parameters on the total cummulated reward. The learning rate analysis shows a consistent improvement, and relative deviation between the different algorithms, as the learning rate is increased. For the decay rate, RPF and QPF algorithms coincide at approximately $\gamma = 0.45$ before diverging, with the QPF algorithm outperforming RPF. In all of the analyses, it is notable that the pure RL performs better in non-nonimal conditions, but as the nominal conditions are approached it falls behind the other two algorithms.

In the steps analysis shown in Figure 4-15, it is clear that the pure RL algorithm outperforms the algorithms with Potential Field information. Note that opposed to the cummulated reward analysis, here the best performing algorithm is the one with the lowest step count. The pure RL algorithm does not consider the harmful parts of the environment, and therefore takes the shortest path to the objective, as can be seen in the heatmap in Figure 4-6. With varying the exploration rate (rightmost graph), the PF integrated algorithms reach a peak around $\varepsilon = 0.6$ before convergin down to meet the pure RL performance at an exploration rate of $\varepsilon = 1$. First when the exploration rate is increased, the agent takes more and more steps as it explores the environment more than 50% of the time. Then, as the exploration begins to reach the maximum, less steps are taken due to collisions with obstacles, since the high exploration rate means that it is more likely to pick a random action instead of the optimal. The convergence of all algorithms at $\varepsilon = 1$ also signifies the worst performance, even though they have the least steps taken in a run, when the step information is added with the reward analysis from above, it is clear that for such high exploration values the resulting reward is the lowest.

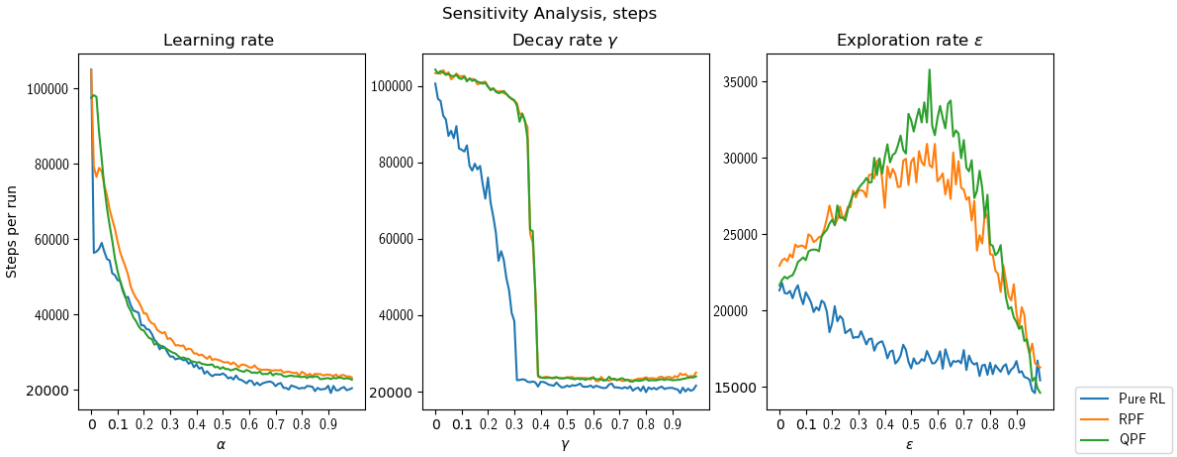


Figure 4-15: Sensitivity Analysis, steps per run

The effects of varying parameters were also made for the policy based PF integration. Figure 4-16 shows the cummulated reward analysis. Relative to the previous analysis in Figure 4-14, the results are more inconsistent relative to one another. The backtracking policy (blue line) outperforms the other two algorithms for the most part, but does break for learning rate values exceeding $\alpha = 0.9$. Similarly the decay rate of more than $\gamma = 0.9$ also breaks the backtracking algorithm. Examining the second Figure, 4-17, this lack of performance can be attributed to learning the wrong policy, for example getting stuck in a loop between two states of in a corner of the environment. While the algorithm does not perform at the extremes of the parameters, it however has the highest performance at nominal conditions. In the leftmost graph of Figure 4-14 it can be seen that the ε-greedy algorithm does outperform the backtracking one in terms of the least amount of steps taken, which can be attributed to the backtracking algorithm's cautious approach in dangerous situation, where it will retrace its steps back to the previous state, adding to the total number of steps taken.

As it was determined earlier with the softmax policy, it did not have a satisfactory performance for the previously selected nominal values of $\alpha = \gamma = 0.9, \varepsilon = 0.1$. This can be seen especially

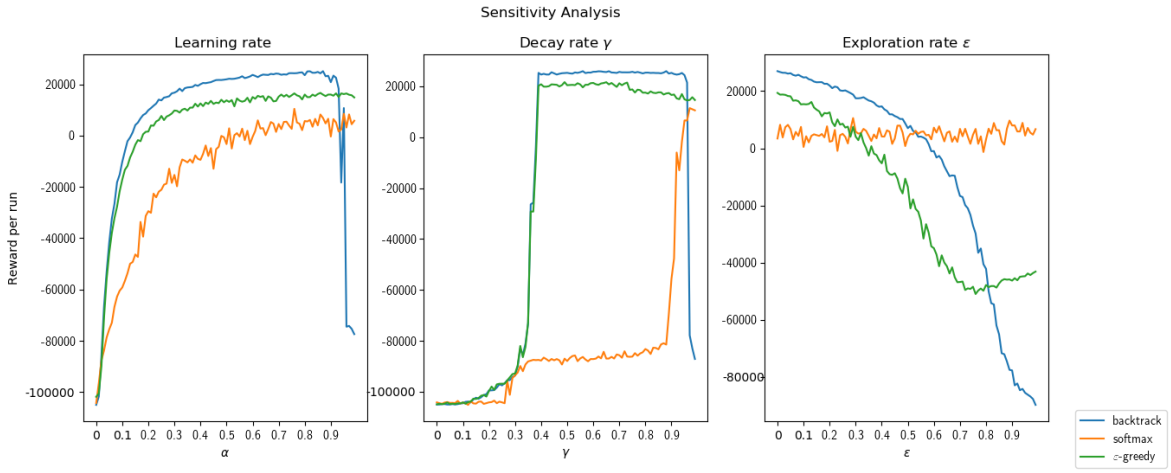


Figure 4-16: Policy Sensitivity Analysis, reward per run

in the decay rate analysis, as the softmax policy underperforms until it surpasses $\gamma = 0.9$. This illustrates the softmax policy's reliance on the past state visits, and linking up the states in order to learn a policy. Whereas the ε-greedy has a relatively stable performance in the γ range of (0.4,0.9), the softmax only performs adequately when the γ parameter is approaching 1. Last note is that the softmax does not utilize the exploration rate ε and therefore is independent in the analysis. The slight variations in their relation can be attributed to perturbations due to the initial random selection by the softmax policy when the Q-values are all equal.

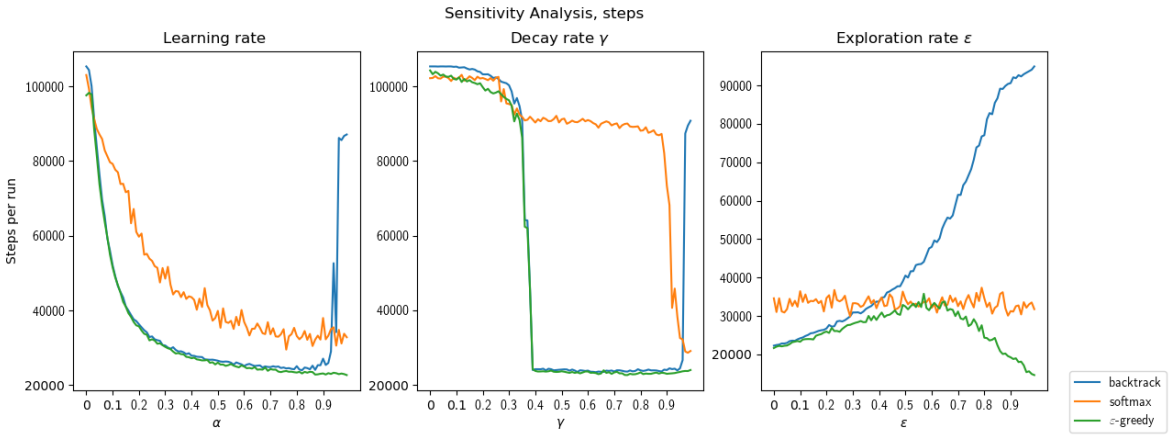


Figure 4-17: Policy Sensitivity Analysis, steps per run

4-3 Chapter Summary

In this chapter, a simulation study was conducted using a gridworld environment featuring various obstacles. The agent is tasked with reaching the goal state, without explicit instruction. Instead, reaching the goal state gives the agent a positive reward, which it uses to learn the actions needed to get to the goal state. Algorithms featuring Potential Field information within the Reinforcement Learning algorithm were compared with a pure Reinforcement Learning one. Different methods of integrating Potential Field information were presented, namely integrating it within the reward function, the Q-function, and numerous ways of utilizing the Potential Field information within the action policy space.

A performance analysis was conducted, comparing the results of the algorithms relative to one another. The analysis was based on the rewards collected and steps taken, as well as the number of collisions and completions of each algorithm. This was followed by a sensitivity analysis into varying the learning parameters, and the effects on the performance of all algorithms due to these variations.

This chapter addresses the second Research Question presented in Section 1-3, about the feasible approaches of integrating PF Methods within an RL algorithm. The safety definition (**RQ2a**) was established around the number of obstacle collisions of the agent - the less collisions during learning, the safer the algorithm. The level of integration which produces the highest increase in safety relative to the flat RL algorithm (**RQ2b**) is the QPF integration, where the PF information is implemented directly within the Q-function. Furthermore, the trial-and-error nature of RL (**RQ2c**) is addressed with the backtracking algorithm, which produces learning without any collisions, guaranteeing safety of the agent (sans the max iterations/exhaustion limit). These results validate the feasibility of the algorithm, giving way to implementing it within a more complex aerospace system.

Part III

Additional Results

Chapter 5

Flight Controller Application: Learning Progression

In this Chapter, the progression of learning of the Reinforcement Learning algorithm is elaborated. Given the trial-and-error nature of RL, the algorithm is bound to make errors during the course of finding the optimal solution. This is the driving reason behind implementing a safety feature in the form of Artificial Potential Fields.

The algorithms presented in the scientific article, attached in Part I, are examined throughout the simulation runs, with a more in-depth look into intra-episodic behaviour, performance, and safety of the agent during learning. The algorithm performance is compared between the 3 selected flight controllers, namely Pure RL ε-greedy, QPF ε-greedy (Q-function based Potential Filed modifier), and QPF with backtracking policy.

The simulation setup is to follow a given pitch rate reference shown in Equation 5-1, with an additional reference provided by the APF, centered around the initial condition altitude, shown in Equation 5-2. The equations parameters are $\rho_0 = \texttt{abs}|h - 20000|$, $\rho_0 = 35$, and $k_U = 0.5{\cdot}10^5$. During the simulation run of 3000 episodes, greedy episodes were run inbetween ε-greedy ones - each 50th episode as well as the last 300 episodes were greedy, in order to compare the performance without the effects of randomness introduced by exploration.

$$\texttt{q}_{\texttt{ref}} = \frac{100}{180} \cdot \pi \cdot sin(0.1 \cdot \pi \cdot \texttt{dt}) \tag{5-1}$$

$$\texttt{U}(h) = k_U * \left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)^2 \tag{5-2}$$

In Figure 5-1 the rewards gained per episode are shown. The pure RL controller outperforms both implementations of the APF, since it is able to follow the reference pitch rate without the interference of the APF signal. The backtracking policy initital performance suffers due to the lack of exploration imposed by the safety criterion- the agent is prevented from entering regions which are deemed unsafe, with the consequence of the agent not learning these states

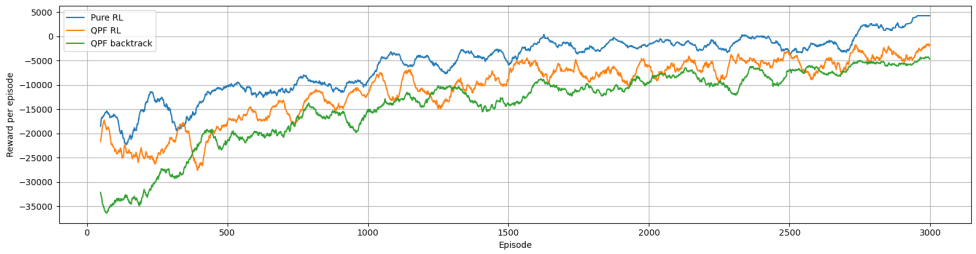


Figure 5-1: Reward per episode, pure RL vs QPF vs backtrack, 50 episode rolling average

result in a negative reward, the agent is therefore more likely to follow the same path in a later episode.

Figures 5-2 through 5-6 show the progression of learning for episodes 50, 250, 1500, 2500 and the last episode 2999. The angle of attack and pitch angle states do not follow a predefined reference, only reacting to the pitch rate tracking by the controller. It can be seen from the learning progression that the pitch rate tracking (second graph from top in above Figures) improves over time for all of the controllers, though deviations are noticeable in the backtracking policy controller due to the agent attempting to stick to the predesignated altitude. The backtracking policy controller's effectiveness is clearly visible in the bottom graph in Figures, the altitude response, where the controller instructs the agent to return once the altitude difference from the initial condition is more than 2000[ft]. When the difference in altitude is realized, the agent is quick to change its course and return to the safe region, at the expense of pitch rate tracking performance.

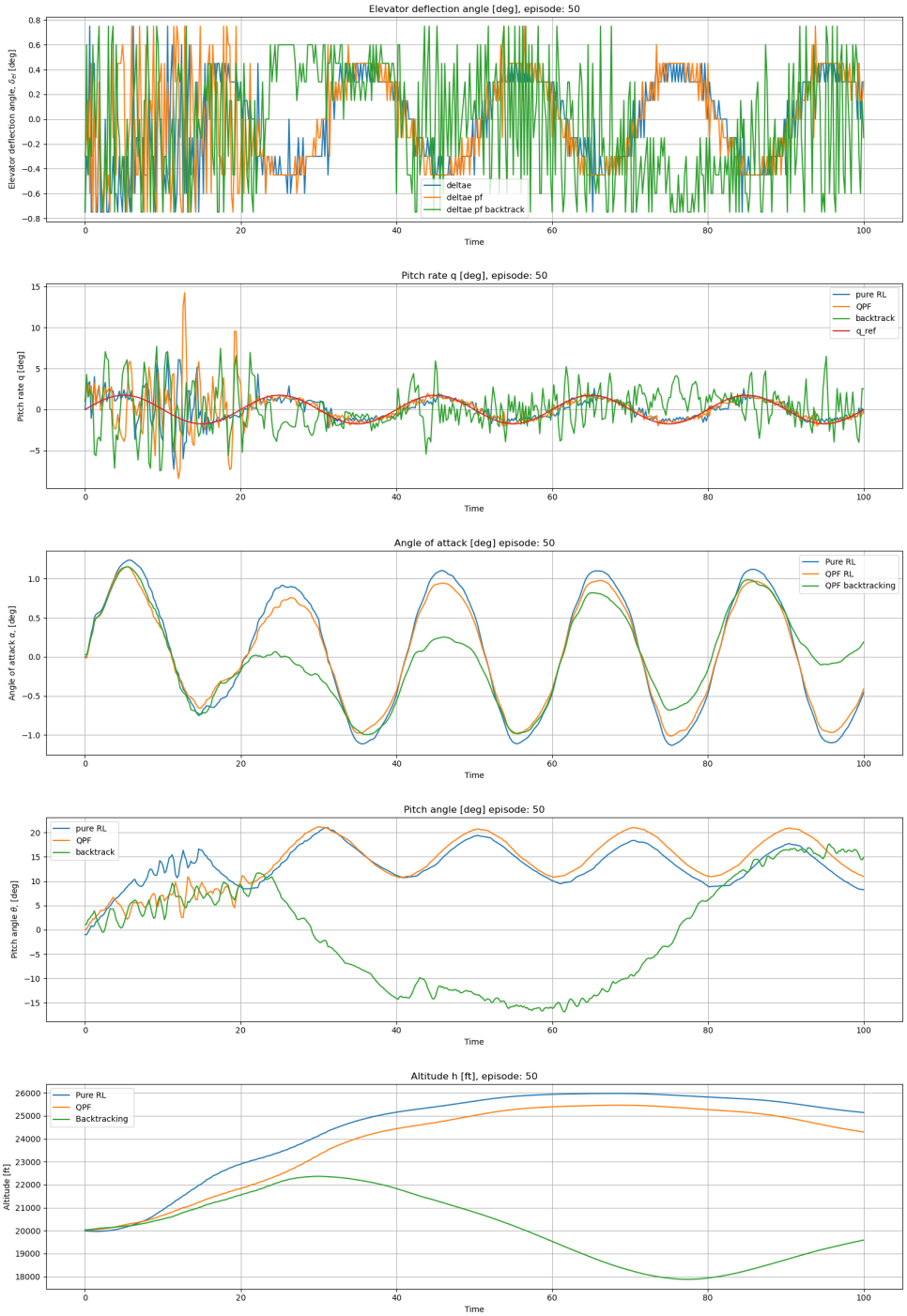


Figure 5-2: System response for episode 50, from top to bottom: input δ_{el}, pitch rate q, angle of attack α, pitch rate θ, altitude h

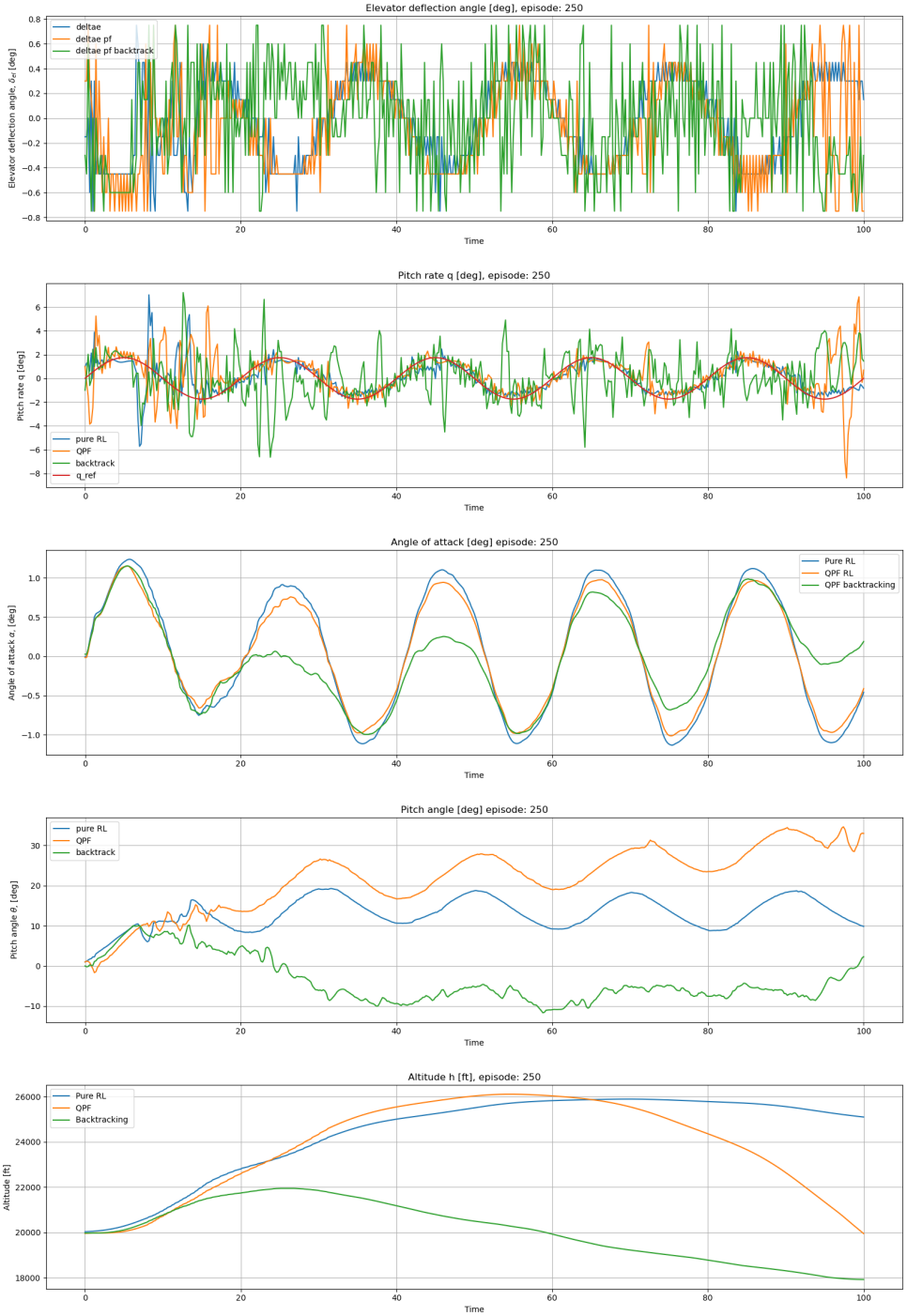


Figure 5-3: System response for episode 250, from top to bottom: input δ_{el}, pitch rate q, angle of attack α, pitch rate θ, altitude h

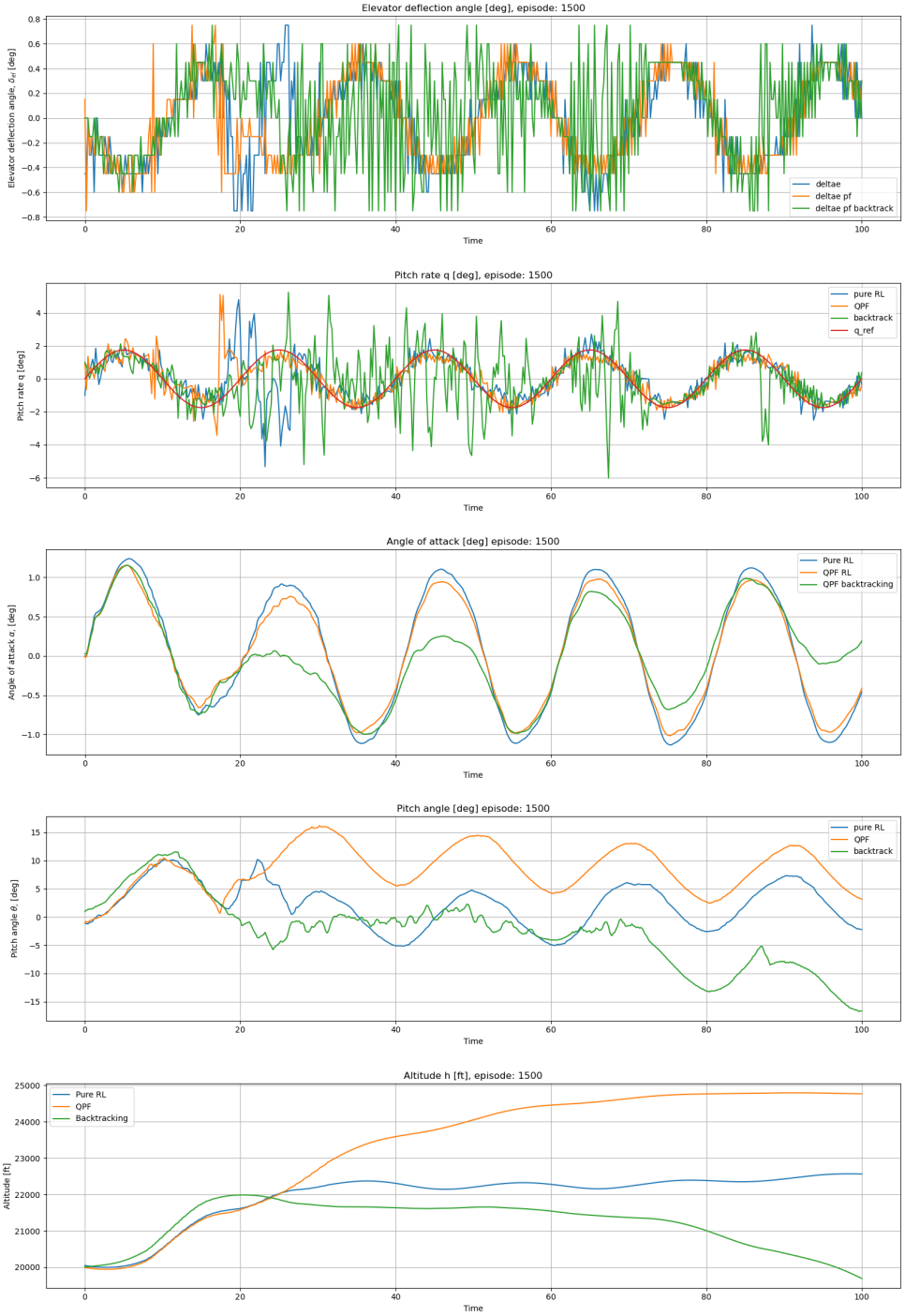


Figure 5-4: System response for episode 1500, from top to bottom: input δ_{el}, pitch rate q, angle of attack α, pitch rate θ, altitude h

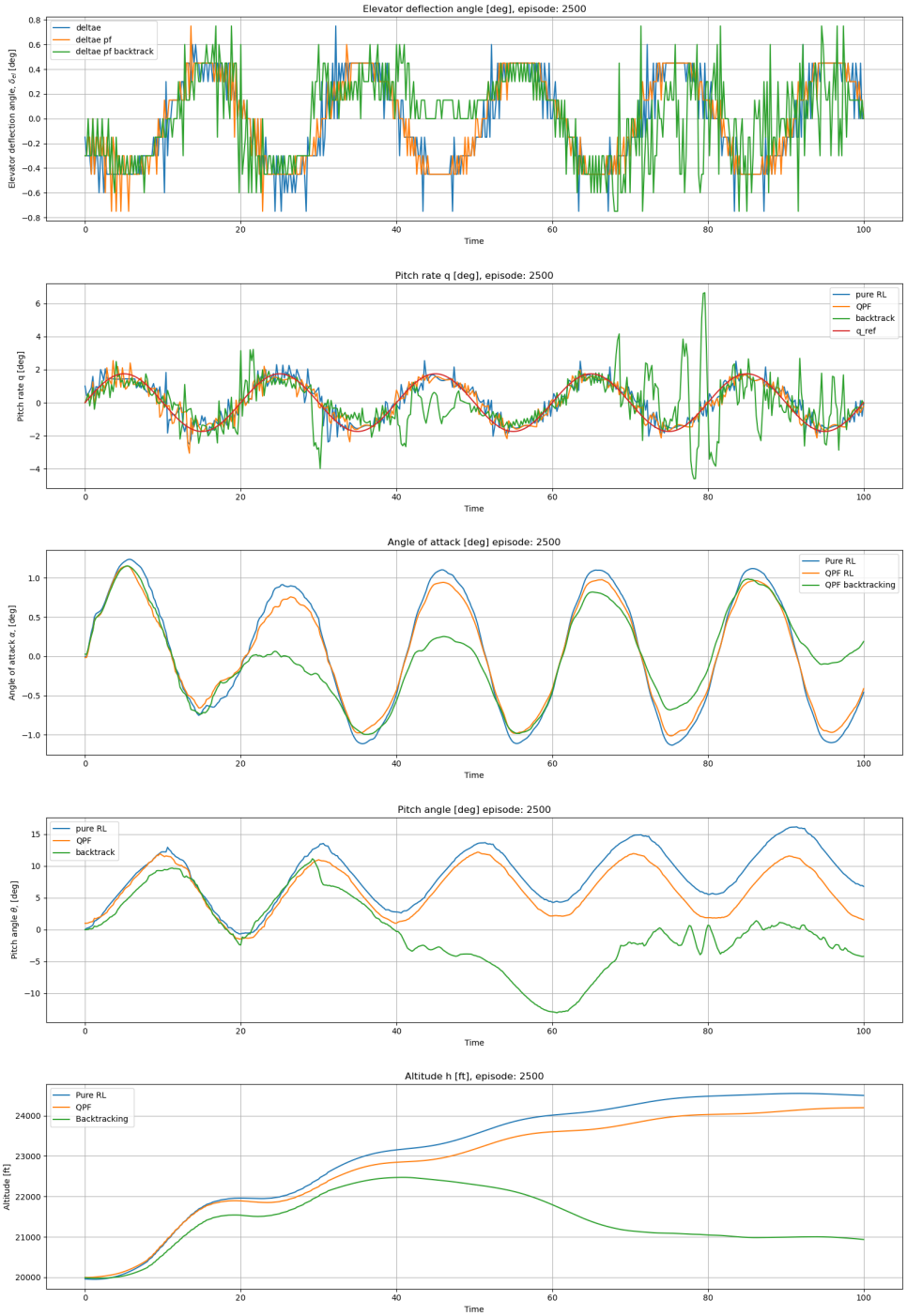


Figure 5-5: System response for episode 2500, from top to bottom: input δ_{el}, pitch rate q, angle of attack α, pitch rate θ, altitude h

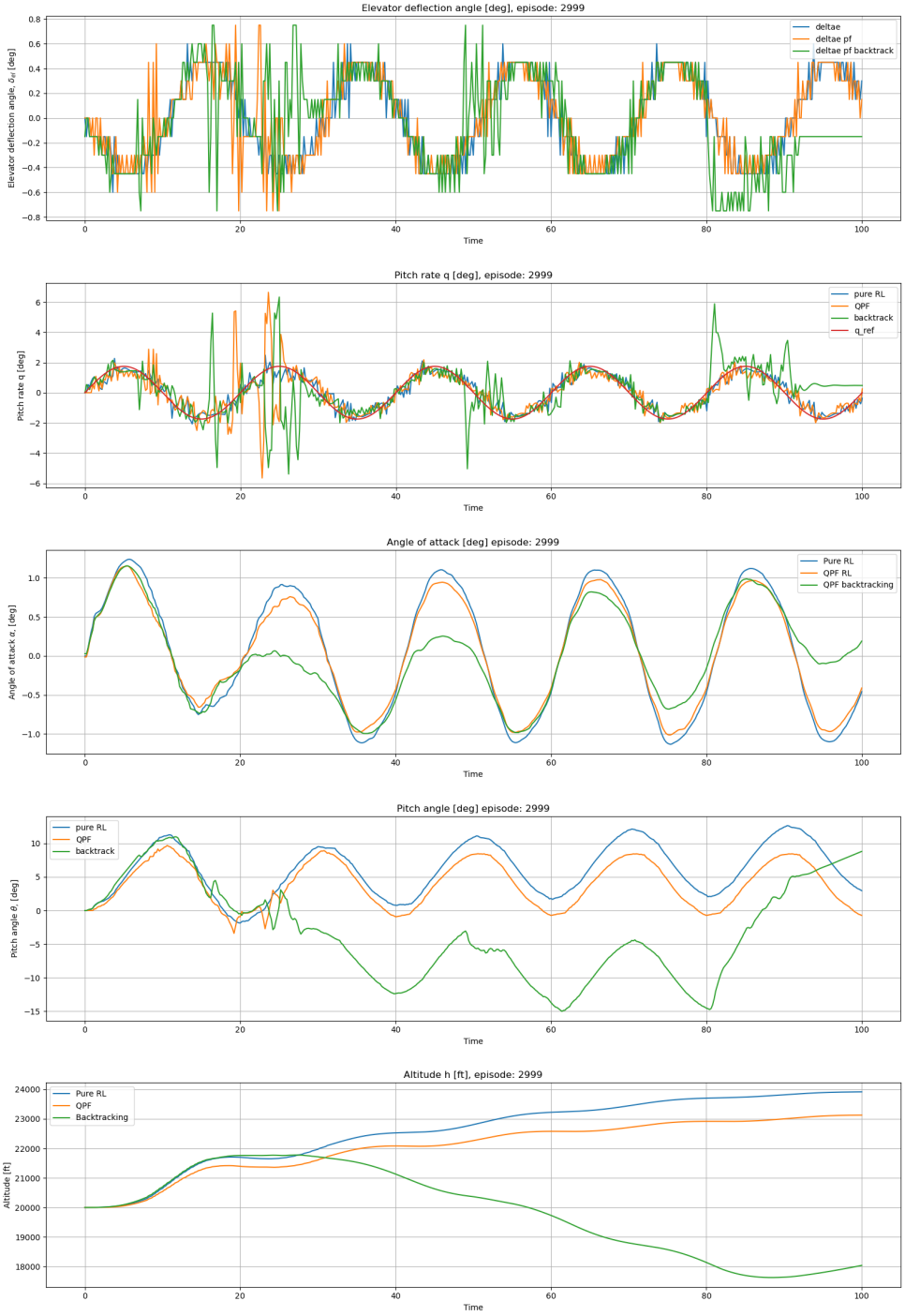


Figure 5-6: System response for episode 2999, from top to bottom: input δ_{el}, pitch rate q, angle of attack α, pitch rate θ, altitude h

Chapter 6

Flight Controller Application: Validation

A flight controller application of the Artificial Potential Field-infused Reinforcement Learning has been presented in the scientific article attached in Part I. In this Chapter, the developed algorithm is applied to the same model in a different flight condition to confirm its validity.

The flight condition for the validation model was selected for $[v_t, h] = [500[\texttt{ft/s}], 10000[\texttt{ft}]]$, with the resulting system dynamics shown below in Equations 6-1 and 6-2.

$$\begin{bmatrix} \dot{V}_t \\ \dot{\alpha} \\ \dot{\theta} \\ \dot{q} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} -0.01304 & -2.948 & -32.17 & -1.028 & 0 \\ 0 & -0.7506 & 0 & 0.9281 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & -1.836 & 0 & -1.027 & 0 \\ 0 & -900 & 900 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} V_t \\ \alpha \\ \theta \\ q \\ h \end{bmatrix} + \begin{bmatrix} 0.3549 \\ -0.0021 \\ 0 \\ -0.3139 \\ 0 \end{bmatrix} \cdot \delta_{el} \tag{6-1}$$

$$\begin{bmatrix} V_t \\ \alpha \\ \theta \\ q \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 57.2958 & 0 & 0 & 0 \\ 0 & 0 & 57.2958 & 0 & 0 \\ 0 & & 0 & 57.2958 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} V_t \\ \alpha \\ \theta \\ q \\ h \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \cdot \delta_{el} \tag{6-2}$$

In order to compare the data directly with the existing results, the APF function was kept the same, shown in Equation 6-3, with the parameters equalling to: $\rho_0 = \texttt{abs}|h - 10000|$, $\rho_0 = 35$, and $k_U = 0.5 \cdot 10^5$. The only difference is the reference altitude being changed from $20000[\texttt{ft}]$ to $10000[\texttt{ft}]$.

$$\texttt{U}(h) = k_U * \left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)^2 \tag{6-3}$$

The reward collected throughout the simulation run is shown in Figure 6-1.

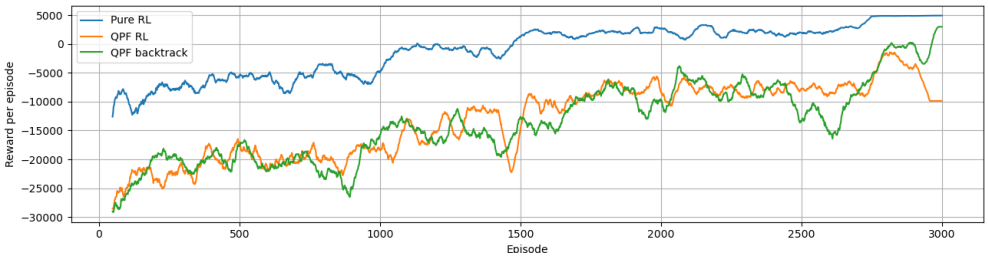


Figure 6-1: Reward per episode, pure RL vs QPF vs backtrack, 50 episode rolling average

The system response for the last episode of the simulation run is shown in Figure 6-2, with an APF construction compared to the final altitude response presented in Figure 6-3. Furthermore, mean response for altitude across the entire simulation and across the greedy episodes at the end of the simulation are shown in Figures 6-4 and 6-5, respectively.

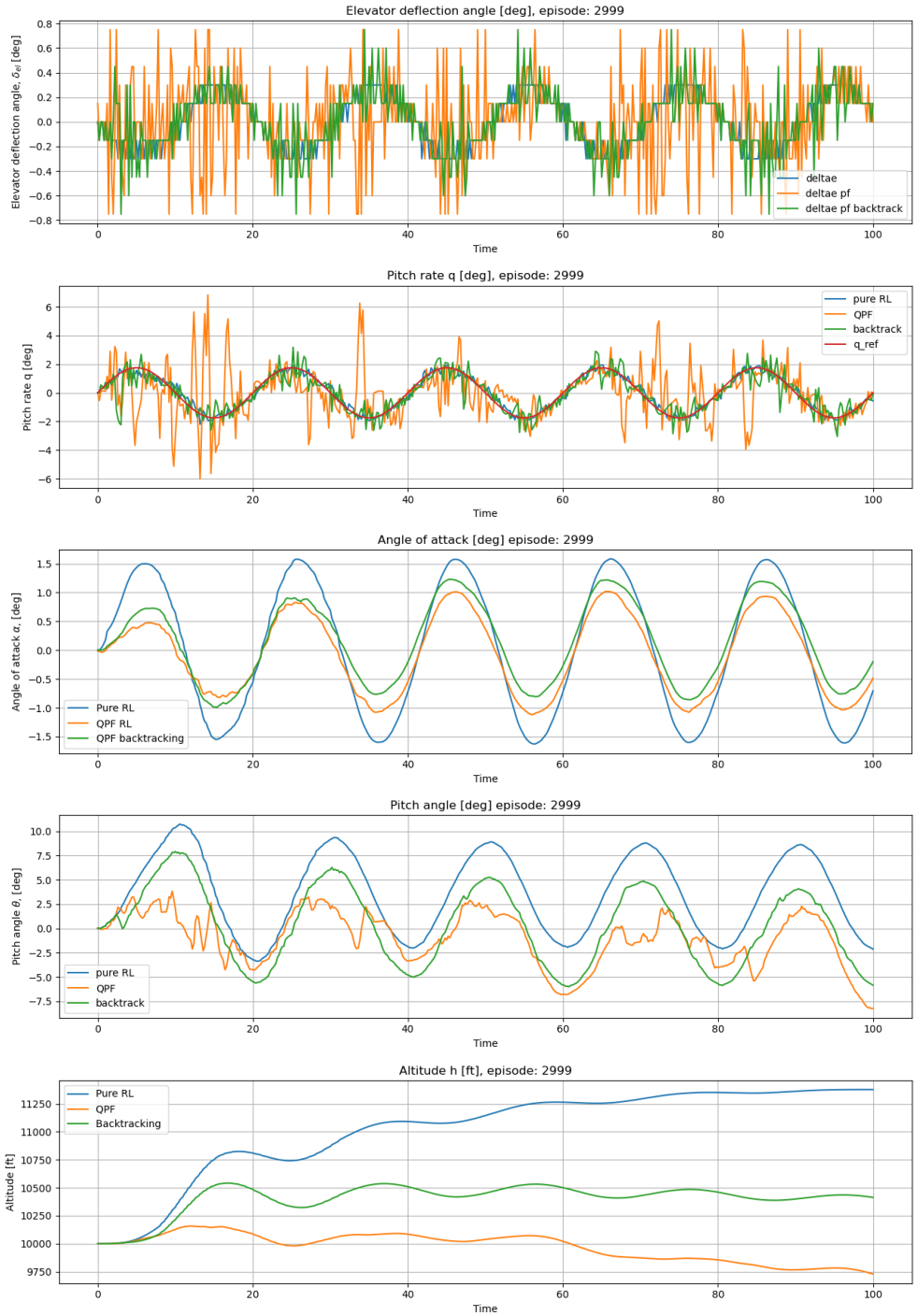


Figure 6-2: System response, episode 2999

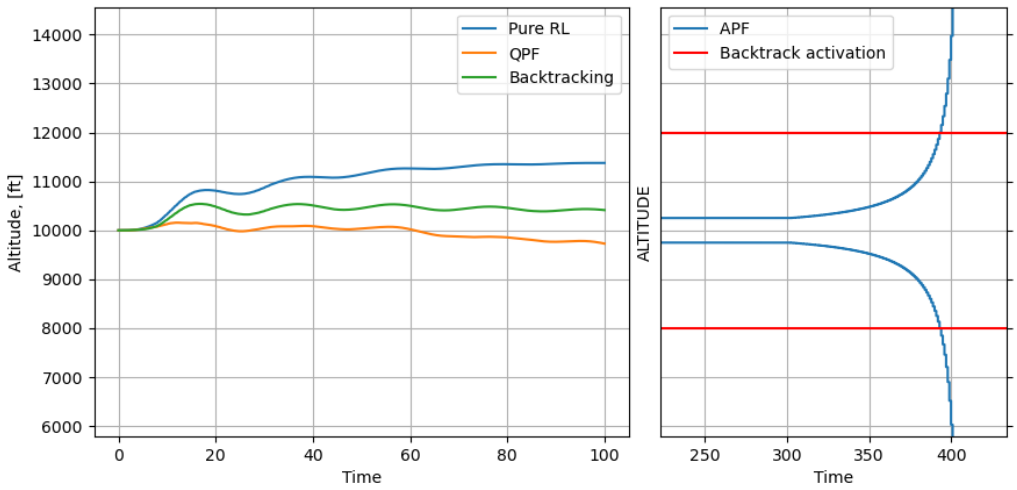


Figure 6-3: Altitude response of the validation model, last episode, with the potential field function shown on the right-hand side graph

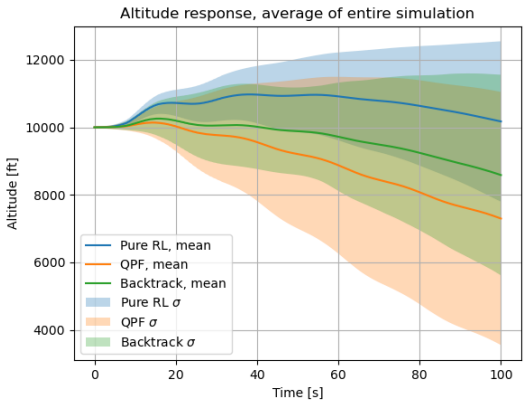


Figure 6-4: Altitude response, mean across all episodes

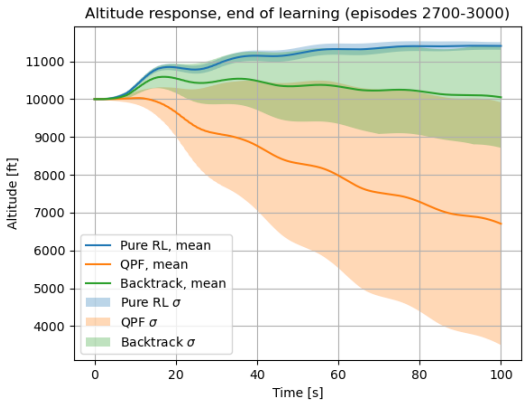


Figure 6-5: Altitude response, mean across greedy episodes (2700-3000)

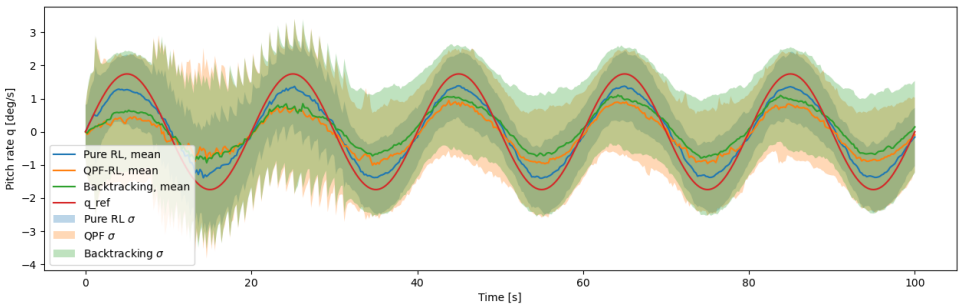


Figure 6-6: Mean pitch rate, with standard deviation marked by shaded areas

Part IV

Closing Remarks

Chapter 7

Conclusion & Recommendations

7-1 Conclusion

In conclusion, a Reinforcement Learning algorithm has been developed, which takes additional input in the form of Artificial Potential Field information directly from the environment. The developed algorithm has demonstrated increased safety in terms of obstacle avoidance when the agent is activated in an environment featuring various obstacles, having fewer collisions during learning.

A literature study was conducted into both concepts of Reinforcement Learning and Artificial Potential Field Methods. Artificial Potential Field Methods were integrated within Reinforcement Learning to create Safe Reinforcement Learning, with past research able to back up claims of better performance of the agent in harmful environments.

A simulation study was conducted with a gridworld environment featuring obstacles. A pure RL algorithm was initiated, then Artificial Potential Fields were added around obstacles, and the information provided by them was integrated in various configurations within the base RL algorithm. The results presented in Chapter 4 prove that the combination of RL and APF has made for a safer exploration and learning, particularly when the APF information has been implemented at the Q-function level, and backtracking out states with high potential was used. Additionally, a sensitivity analysis was carried out to determine the effects of changing parameters on the algorithm's performance.

Furthermore, the RL algorithm was then applied onto an aircraft flight model. A model of the General Dynamics F16 Fighting Falcon was reduced into a Linear Time-Invariant longitudinal model, having a single input state of the elevator deflection angle. Controllers featuring pure RL and Q-function-based APF modifier running ε-greedy policies were compared relative to each other, and relative to a controller operating with a backtracking policy. The backtracking policy acts as an additional layer of safety, instructing the agent to return to a safe state space once a threshold strength of the APF is sensed. The reward function was based on a sinusoidal pitch rate reference, while the APF was centered around the altitude of the initial

flight condition. Performance was defined based on the pitch-tracking ability, while safety was linked to the minimization of the altitude error.

Looking back at the Research Questions outlined in Section 1-3, so far **RQ1** and **RQ2** have been addressed, with the state of the art research (**RQ1**) being outlined in Chapter 2. **RQ2** concerning the approaches of integration of Artificial Potential Field Methods with Reinforcement Learning has been described in Chapters 3 and 4, the former evaluating the methods and the latter showing preliminary simulation results of the combined algorithms.

7-2 Research Questions

The Research Questions introduced in the introductory pages of this report are repeated below with further explanations and pointers to the relevant sections.

The original question was: **To which degree would using Artificial Potential Field Methods improve safety and reliability of a Reinforcement Learning agent?**

To answer this question, safety and reliability were defined for the simulations that were conducted. In the preliminary gridworld study, safety was defined by the agent's ability to avoid collisions with obstacles, and reliability was measured with the agent's path-planning and steps needed to reach the goal state. In Chapter 4 that the agent's safety was drastically improved, with APF-modified RL controllers reducing, and even completely avoiding, collisions with obstacles. The reliability did suffer as a trade-off, but only marginally, justified by the increase in the overall safety of the agent.

Applying this knowledge to an aerospace flight controller, and defining altitude tracking as a safety measure and pitch rate tracking as a performance/reliability metric, it was found that adding APF information to an RL controller increased the safety of the agent. The resulting algorithm, utilizing APF information within the Q-learning matrix as well as the decision-making policy, was able to keep the agent within a predetermined safe state space, that being a relatively narrow altitude window surrounding the altitude of the initial condition. The agent's performance did suffer, resulting in a reduced pitch rate tracking ability, but overall the aircraft managed to continue partially tracking the pitch rate, only deviating locally. Overall, the agent was able to stay within the designated safe state space with the introduction of a backtracking policy, which intructed the aircraft to perform evasive maneuvers when a threshold level of the APF has been reached.

The research sub-questions have been answered throughout the report:

RQ1 What is the current state-of-the-art research in:

(a) Reinforcement Learning - RL theoretical background has been extensively elaborated in Sections 2-1 and 2-1-2, followed by the state of the art of RL in Section 2-2

(b) Artificial Potential Field Methods - APF methods were introduced in Section 2-3, with the current state-of-the-art elaborated in Section 2-4

RQ2 What are the feasible approaches of integrating Artificial Potential Field Methods with Reinforcement Learning?

(a) How is safety defined for an aerospace system? - Safety was defined by the agent's ability to stay within predetermined error from the reference altitude, which the aircraft was initialized from.

(b) At which level of RL does applying APF Methods produce the most significant increase in safety? - Different combinations of APF integration were tested, namely by adding APF information to the reward and value functions, and basing the decision-making policy on the APF strength experienced by the agent. A combination of value function APF modifier and a policy utilizing APF strength to backtrack out of unsafe state spaces was found to be the most effective implementation, resulting in the agent staying within the safety bounds while still having the ability to explore the environment.

(c) How can APF Methods address the issues of trial-and-error within RL? - APF was found to reduce the negative aspects of the trial-and-error of RL, by limiting the agent from entering undesired states and deflecting from the predefined safe part of the environment. However, this does put a limit on the agent's ability to explore, hence a trade-off is made between safety and exploration.

RQ3 What are the measurable improvements in the safety of flight when using Artificial Potential Fields, relative to pure Reinforcement Learning?

(a) What is a real-life scenario which can be used in a simulation in order to validate the results? - A model of the General Dynamics F-16 Fighting Falcon was derived to be used in conjunction with a RL flight controller. APF methods were integrated with the RL controller to compare the performance and safety of the integrated algorithm to that of a pure RL controller.

(b) How is safety parametrized? - Safety was parametrized by the error of the model to the initial altitude

(c) How repeatable is the learning in different scenarios? - The results of the flight controller were validated against a similar model set to a different flight condition in Chapter 6, showing acceptable performance and complying with the safety requirements.

7-3 Future Recommendations

This research project has examined the effects of adding additional safety considerations to a Reinforcement Learning controller, which was the most critical problem identified in previous literature, and in the introductory pages of this report. While some of the challenges have been addressed, there remain other ones that are still open to solutions. Online efficiency still remains a crucial challenge in the whole research area of RL, as it is computationally expensive to process large state spaces.

In terms of the above-presented research, additional optimization can increase the safety further, and improve the algorithm performance. Hyperparameter optimization can also be improved, present research utilized exploration reduction throughout learning, for example, but additional tuning could result in a better performing algorithm. Additional algorithms which combine the introduced concepts, such as using a softmax policy with backtracking

or combining potential fields with deep neural networks, have also been left unexplored. Furthermore, the backtracking implementation has so far been simple, and further attention can be focused on expanding it such that it is comprised of multiple stages, such that it can steer the agent more effectively

In terms of the simulation setup, a more complex approach can be used to better understand the algorithm's performance in a real-life scenario. Disturbance inputs were not explored in this research, and future research into the topic of Artificial Potential Fields and Reinforcement Learning could benefit from a stochastic modelling of wind gusts, for example. Furthermore, the algorithms used here were not preloaded with any a-priori knowledge of the environment or the dynamic system being controller, initial information which could improve the algorithm's performance and safety aspects further. The current state-of-the-art research in the field of Reinforcement Learning has a strong focus on utilizing deep neural networks for computational efficiency- adding APF methods to the deep neural network matrices used (instead of the algebraic matrices used in this research) could also further the efficiency and speed up learning, while maintaining a level of safety necessary for aerospace applications.

Bibliography

[1] T. Mannucci, *Safe Online Robust Exploration for Reinforcement Learning Control of Unmanned Aerial Vehicles*. Delft University of Technology, 2017.

[2] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed., 1998.

[3] M. Heger, "Consideration of Risk in Reinforcement Learning," *Machine Learning Proceedings 1994*, pp. 105–111, 1994.

[4] T. Mannucci, E.-J. Van Kampen, C. C. de Visser, and Q. P. Chu, "SHERPA: a safe exploration algorithm for Reinforcement Learning controllers," *AIAA Guidance, Navigation, and Control Conference*, no. January, 2015.

[5] Y. Liu and Y. Zhao, "A virtual-waypoint based artificial potential field method for UAV path planning," *CGNCC 2016 - 2016 IEEE Chinese Guidance, Navigation and Control Conference*, pp. 949–953, 2017.

[6] A. Bhowal, "Potential Field Methods for Safe Reinforcement Learning," *Delft University of Technology*, 2017.

[7] L. X.-l. Li-juan, XIE, XIE Guang-rong, CHEN Huan-wen, "Solution to reinforcement learning problems with artificial potential field," *Springer*, vol. 21, no. 10, pp. 402–410, 2008.

[8] T. M. Lam, H. W. Boschloo, M. Mulder, and M. M. Van Paassen, "Artificial force field for haptic feedback in UAV teleoperation," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 39, no. 6, pp. 1316–1330, 2009.

[9] J. Archer, H. Keno, and Y. Kwon, "Effects of Automation in the Aircraft Cockpit Environment : Skill Degradation , Situation Awareness , Workload," 2012.

[10] N. Carr, *The Glass Cage: Automation and Us*. W. W. Norton & Company, 1st ed., 2014.

[11] S. Vance and A. Malik, "Analysis of factors that may be essential in the decision to fly on fully autonomous passenger airliners," *Journal of Advanced Transportation*, vol. 49, no. March 2015, pp. 829–854, 2015.

[12] S. H. Lane and R. F. Stengel, "Flight control design using non-linear inverse dynamics," *Automatica*, vol. 24, no. 4, pp. 471 – 483, 1988.

[13] M. Waltz and K. Fu, "A Heuristic Approach to Reinforcement Learning Control Systems," no. 4, 1965.

[14] E. Keogh and A. Mueen, *Curse of Dimensionality*, pp. 314–315. Boston, MA: Springer US, 2017.

[15] S. Verbist, T. Mannucci, and E.-J. Van Kampen, "The Actor-Judge Method: safe state exploration for Hierarchical Reinforcement Learning Controllers," *2018 AIAA Information Systems-AIAA Infotech @ Aerospace*, no. January, 2018.

[16] T. Mannucci, E. van Kampen, C. C. de Visser, and Q. P. Chu, "Hierarchically structured controllers for safe UAV reinforcement learning applications," *AIAA Information Systems-AIAA Infotech at Aerospace, 2017*, no. January, pp. 1–13, 2017.

[17] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," 1985.

[18] A. Barron, E. Søvik, and J. Cornish, "The Roles of Dopamine and Related Compounds in Reward-Seeking Behavior Across Animal Phyla," *Frontiers in Behavioral Neuroscience*, vol. 4, 2010.

[19] R. Bellman, *Eye of the Hurricane: An Autobiography*, vol. 17. World Scientific Pub Co Inc, 1985.

[20] J. Garcia and F. F. Andez, "A Comprehensive Survey on Safe Reinforcement Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 4, 2018.

[21] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," pp. 1–9, 2013.

[23] N. Ammour, H. Alhichri, Y. Bazi, B. Benjdira, N. Alajlan, and M. Zuair, "Deep learning approach for car detection in UAV imagery," *Remote Sensing*, vol. 9, no. 4, 2017.

[24] D. Zhou, J. Zhou, M. Zhang, D. Xiang, and Z. Zhong, "Deep learning for unmanned aerial vehicles landing carrier in different conditions," in *2017 18th International Conference on Advanced Robotics (ICAR)*, pp. 469–475, July 2017.

[25] N. D. Daw and P. N. Tobler, "Value Learning through Reinforcement: The Basics of Dopamine and Reinforcement Learning," *Neuroeconomics: Decision Making and the Brain: Second Edition*, pp. 283–298, 2013.

[26] A. Ohman, S. C. Soares, P. Juth, B. Lindstrom, and F. Esteves, "Evolutionary derived modulations of attention to two common fear stimuli: Serpents and hostile humans," *Journal of Cognitive Psychology*, vol. 24, no. 1, pp. 17–32, 2012.

[27] S. Verbist, "Safe Hierarchical Reinforcement Learning," 2017.

[28] C. Gaskett, "Reinforcement learning under circumstances beyond its control," *Proceedings of the international conference on computational intelligence for modelling control and automation*, 2003.

[29] P. Geibel, "Reinforcement learning for MDPs with constraints," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4212 LNAI, pp. 646–653, 2006.

[30] Y. Kadota, M. Kurano, and M. Yasuda, "Discounted Markov decision processes with utility constraints," *Computers and Mathematics with Applications*, vol. 51, no. 2, pp. 279–284, 2006.

[31] A. Tamar, D. Di Castro, and S. Mannor, "Policy gradients with variance related risk criteria," *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, vol. 1, pp. 935–942, 2012.

[32] J. A. Clouse and P. E. Utgoff, "On integrating apprentice learning and reinforcement learning," 1996.

[33] P. Quintia, R. Iglesias, M. A. Rodriguez, and C. V. Regueiro, "Learning on real robots from experience and simple user feedback," *Journal of Physical Agents*, vol. 7, no. 1, pp. 56–64, 2013.

[34] X. Yang, W. Yang, H. Zhang, H. Chang, C. Y. Chen, and S. Zhang, "A new method for robot path planning based artificial potential field," *Proceedings of the 2016 IEEE 11th Conference on Industrial Electronics and Applications, ICIEA 2016*, no. December 2014, pp. 1294–1299, 2016.

[35] C. Shi, M. Zhang, and J. Peng, "Harmonic potential field method for autonomous ship navigation," *ITST 2007 - 7th International Conference on Intelligent Transport Systems Telecommunications, Proceedings*, pp. 471–476, 2007.

[36] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pp. 1398–1404 vol.2, April 1991.

[37] F. Bounini, D. Gingras, H. Pollart, and D. Gruyer, "Modified artificial potential field method for online path planning applications," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 180–185, 2017.

[38] J. B. Burns, R. Weiss, and C. Connelly, "Path Planning Using Laplace's Equation," *University of Massachusetts at Amherst, Office of Naval Research*, pp. 1–14, 1994.

[39] J. O. Kim and P. K. Khosla, "Real-Time Obstacle Avoidance Using Harmonic Potential Functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 338–349, 1992.

[40] T. T. Mac, C. Copot, A. Hernandez, and R. De Keyser, "Improved potential field method for unknown obstacle avoidance using uav in indoor environment," in *2016 IEEE*

14th International Symposium on Applied Machine Intelligence and Informatics (SAMI), pp. 345–350, Jan 2016.

[41] S. Scherer, S. Singh, L. Chamberlain, and M. Elgersma, "Flying fast and low among obstacles: Methodology and experiments," *International Journal of Robotics Research*, vol. 27, no. 5, pp. 549–574, 2008.

[42] K. Asadi and M. L. Littman, "An alternative softmax operator for reinforcement learning," *34th International Conference on Machine Learning, ICML 2017*, vol. 1, no. Property 2, pp. 360–369, 2017.