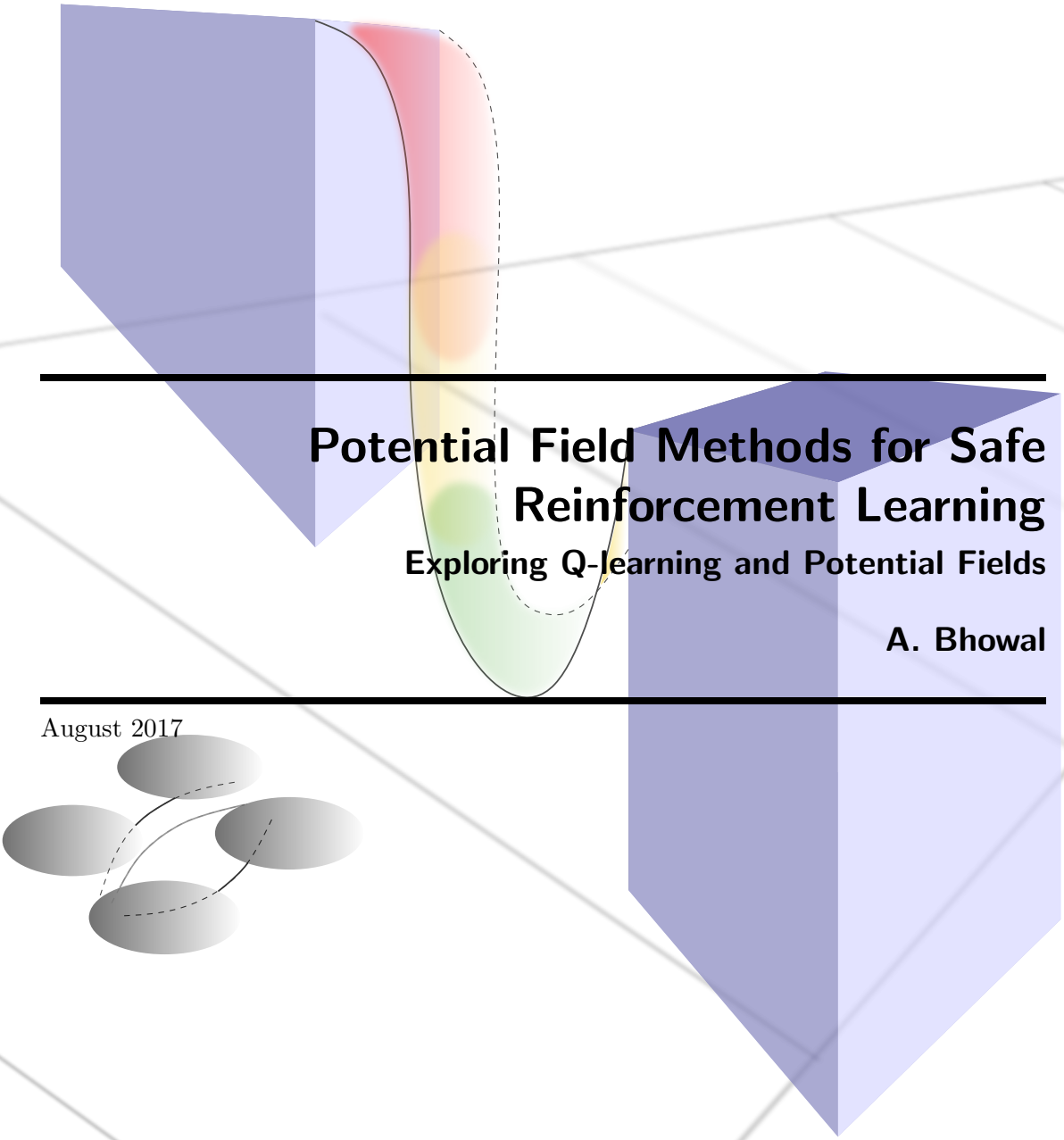
Master of Science Thesis



Potential Field Methods for Safe Reinforcement Learning

Exploring Q-learning and Potential Fields

A. Bhowal

August 2017

Faculty of Aerospace Engineering · Delft University of Technology

TU Delft Delft University of Technology

Challenge the future

Potential Field Methods for Safe Reinforcement Learning

Exploring Q-learning and Potential Fields

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering at Delft University of Technology

A. Bhowal

August 2017

Faculty of Aerospace Engineering · Delft University of Technology

Delft University Of Technology
Department Of
Control and Simulation

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Potential Field Methods for Safe Reinforcement Learning"** by **A. Bhowal** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: August 2017

Supervisor: _____
Dr.ir. Erik-Jan van Kampen

Supervisor: _____
Tommaso Mannucci, MSc

Reader: _____
Dr.ir. Q. P. Chu

Reader: _____
ir. Jos Sinke

Acknowledgements

This work is the culmination of five years of learning, growth and self-discovery, none of which would have been possible without the friends and acquaintances I have had the pleasure to meet and get to know. You've all made me a better person, one way or another.

I would like to thank my parents for always encouraging me to explore. Your support is a model that I will take with me, regardless of what is to come. To Babai, for making me realise the importance of emotional intelligence, and to Ma, for being the most supportive shoulder a son could ask for. To Ryan, who's always up for a discussion, be it about space, football or Minecraft.

Special thanks to Erik-Jan and Tommaso, for making the world of Reinforcement Learning seem so accessible. The discussions we've had over this thesis period have been genuinely enjoyable and have led me to appreciate the fact that we definitely are at the brink of an AI revolution.

This is for those who have been a part of this journey and for those who continue to do so.

Thank you.

Delft, The Netherlands — A. Bhowal
August 2017

List of Symbols and Abbreviations

Symbols

α	Learning Rate
β	Risk Sensitivity Parameter
ϵ	Exploration Factor
γ	Discount Factor
κ	Potential Sensitivity Parameter
π	Action Selection Policy
τ	Softmax Temperature Parameter
a_k	Agent action at time step k
G	Goal State
k	Discrete Time Index
m	Potential Merge Factor
$p(a\|s)$	Probability of selecting action a, given that the agent is in state s
$P(s)$	Potential value in state s
q	Potential Scaling Factor
$Q_t(s,a)$	Q-value for state s, with action a, at time t
r_avg	Performance Metric
R_k	Return over a period of episodes
r_k	Immediate Reward
S	Start State
s_k	Discretised State
$safmet$	Safety Metric

Abbreviations

APF	Artificial Potential Fields
DP	Dynamic Programming
GNRON	Goal Non-Reachable with Obstacle Nearby
LfD	Learning from Demonstration
LTF	Lead-to-fatal
MC	Monte Carlo
MDP	Markov Decision Process
PI-SRL	Policy Improvement through Safe Reinforcement Learning
RL	Reinforcement Learning
SHERPA	Safety Handling Exploration with Risk Perception Algorithm
UAV	Unmanned Aerial Vehicle

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Reinforcement Learning (RL) has demonstrated its potential as a powerful model-free framework to accomplish tasks relevant to the aerospace field, such as control [1], planning and sequential decision making [10]. The RL agent is intelligent by virtue of the fact that it can process information sensed from its environment and use this to constantly modify its subsequent actions. In order to perceive the state of the environment and ensure a steady flow of information, the agent needs to explore. On the scale of a small simulation with a few states, this exploration does not benefit from any restrictions. However, keeping in mind a realistic aerospace scenario such as autonomous navigation of an Unmanned Aerial Vehicle (UAV), the margins for error are low and it is always assumed that risks and uncertainty are high.

In order to mitigate these risks, various methods for 'Safe Exploration' within RL have been proposed and developed [7]. This thesis goes into detail about one of these methods, which involves the use of Artificial Potential Fields (APF). Before the notion of safe exploration can be expanded, however, 'safety' itself must be defined and put into context. This is detailed in the first section. Introducing the concept of APFs as a promising solution in the second section lays the groundwork for the problem statement. This is introduced in the third section and the introductory chapter is then concluded with a roadmap, detailing the scope of this thesis.

1.1 Safety in RL

The very first analogy presented by Sutton and Barto [23] in their seminal work, while introducing 'the Reinforcement Learning problem', is that of a playing infant that 'exercises its sensorimotor connection to its environment to produce a wealth of information about cause and effect and what to do in order to achieve its goals'. It is not difficult then to imagine this inexperienced infant carrying out a certain action that, directly or otherwise, leads to it harming itself. However, avoiding these undesirable situations and finding the best solution to a certain problem would involve the infant exploring and trying out new

things. Traditionally with RL, the discussion at this point leads towards the trade-off between exploration and exploitation. 'The agent has to *exploit* what it already knows in order to obtain its reward, but it also has to *explore* in order to make better action selections in the future.' [23].

In the case of this thesis, however, the interest lies more with making sure that in this 'trial and error' process of getting to know its environment, the infant does not cause itself any harm. Considering now a UAV as the RL agent, during learning, it can experience unsafe, damage inducing and possibly fatal occurrences, especially with high complexity tasks, such as with uncertain systems and/or unknown environments. The concept of safety has been incorporated within RL in two major branches, the 'optimisation criterion', and the 'exploration process' [7]. The first deals with optimising pre-defined performance metrics (minimise costs, maximise rewards etc.), while the latter tries to prevent dangerous exploratory actions by the agent. Approaches such as 'Safety Handling Exploration with Risk Perception Algorithm (SHERPA)' have been shown to enable an agent to successfully explore its environment under limited prediction capabilities. This algorithm was also validated on a quadrotor model, further solidifying the applicability of this research to an aerospace context [17]. A more thorough treatment and classification of Safe RL is carried out in Ch. 3.

1.2 Artificial Potential Fields

Artificial Potential Field (APF) methods are a mature branch of control theory that can perform state and obstacle evasion. Conventionally, however, APFs have been used extensively for path planning of robotic manipulators and mobile robots [24]. The key concept in APF methods is the allocation of a potential function that increases when the system nears an undesired state or condition. If the corresponding generalised forces are used to control the system, undesired states can be avoided. A challenge of APF methods is that the potential must be feasible, i.e., the control surfaces and actuators must be able to apply the generalized forces as indicated by the potential. Once an appropriate potential is defined, APF methods are reliable and autonomous in providing safety. Thus, applying these methods to RL agents appears to be a promising strategy for Safe Exploration.

1.3 Problem Statement and Research Goals

This thesis investigates the use of APF methods in the context of RL agents acting on uncertain systems and partially known or unknown environments. Despite APF methods being used frequently in path planning tasks, that is not per-se the goal here. Instead, these methods will be adapted and optimised to aid in the following of more generalised goals, such as attitude or velocity targets. The APF approach will be required not only to provide safety, but also to accommodate a degree of freedom for the agent, in order to allow learning through state exploration. The following problem statement is used as a leading question for this thesis:

To what extent can Artificial Potential Fields (APF) be used to increase safety of exploration within Reinforcement Learning (RL)?

Taking this as a starting point, the research goals to be considered in further detail in this thesis are identified as follows:

1. Can APF methods be combined with RL in such a way that measurable improvements in terms of *safety* are observed?
   (a) Which *previous work* has addressed the combination of APF and RL methods?
   (b) How can APF methods be applied to *uncertain systems in partially known environments*?
   (c) How has *safety* been classified within RL?
   (d) How can APF methods best accommodate the *trial-and-error* nature of RL?
   (e) Do RL methods using APF information show a decrease in *computational time, complexity and performance*, while ensuring an increase in *safe action selection*?
2. How can APF methods be simulated within an RL environment in such a way that a definitive conclusion about real-life feasibility can be made?
   (a) Which of the available model-free RL algorithms presents the best return in terms of implementation complexity and adaptability?
   (b) To what extent can the APF concept be implemented in a discretised manner while still retaining its key characteristics?
   (c) To what extent can definitive statements about safety be made based on metrics such as *convergence*, *collision frequency* and *computation time*?
   (d) To what extent can 'realism' be approached within the simulation (e.g., increasing the state-space or making it more continuous) in order to verify the results and prove feasibility?

Leading on from this, Fig. 1.1 presents an overview of the steps taken in the thesis. After introducing the basics of RL and defining some standard terminology in Ch. 2, safety is further elaborated upon in Ch. 3. APF methods are then considered and put into the context of the overall goal in Ch. 4. Once the theory has been treated in sufficient detail, preliminary simulation results are presented and discussed in Ch. 5. Ch. 6 then presents the research method by discussing and justifying the steps to be taken from this point forward. This leads to the main findings of the thesis, which are discussed in detail in Ch. 7 and Ch. 8, which make up the penultimate part of the document. Conclusions and recommendations for future work are then presented in Ch. 9.

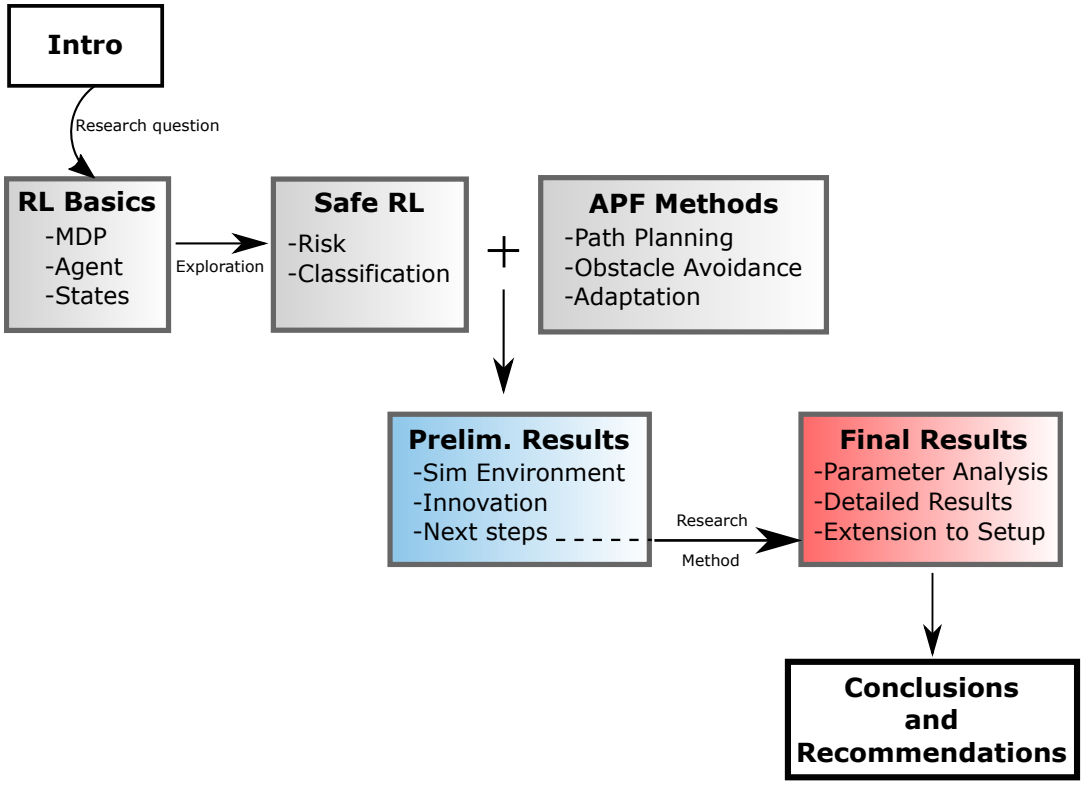


Figure 1.1: Thesis Roadmap

Part I

Article

Increasing Safety of Autonomous Exploration by combining Artificial Potential Fields with Reinforcement Learning

Abhranil Bhowal*

Delft University of Technology, Delft, Zuid Holland, 2629HS, The Netherlands

A Reinforcement Learning (RL) agent learns about its environment through exploration. For most physical applications such as search and rescue UAVs, this exploration must take place with safety in mind. Unregulated exploration, especially at the beginning of a run, will lead to fatal situations such as crashes. One approach to mitigating these risks is by using Artificial Potential Fields (APFs). Various approaches to effectively use the potential information gathered by the agent are proposed, tested and discussed. The agent is placed in an environment-model-free setting, where it is still provided with knowledge of its own dynamics. A gridworld simulation is developed using MATLAB to test the interoperability of APFs with Q-learning. It is shown that safety of exploration benefits from adding this layer of information to the agents' decision making process. In effect, the Q-table gets updated more efficiently due to the agent explicitly knowing of high potential 'dangerous' states.

Nomenclature

α	Learning rate
γ	Discount factor
ϵ	Exploration factor
κ	Potential Sensitivity parameter
τ	Softmax Temperature parameter
q	Potential scaling factor
APF	Artificial Potential Field
G	Goal state
$p(a\|s)$	Probability of selecting action a, given that the agent is in state s
$P(s)$	Potential value in state s
$Q_t(s,a)$	Q-value for state s, with action a, at time t
RL	Reinforcement Learning
S	Start state

I. Introduction

Exploration is a significant aspect of any Reinforcement Learning (RL) algorithm, and this is made especially evident in situations where safety is paramount. A Search and rescue (SAR) drone surveying its environment will be expected not to collide with the unknown obstacles in its path, from take-off to goal location. An RL agent trained on an algorithm which views safety as a key performance indicator, is expected to successfully achieve this objective. The RL agent is perceptive by virtue of the fact that it can process information sensed from its environment and use this to constantly modify its subsequent actions. In order to perceive the state of the environment and ensure a steady flow of information, the agent needs

*MSc student, Aerospace Control and Simulation, Delft.

to explore. Keeping in mind a realistic aerospace scenario such as autonomous navigation of an Unmanned Aerial Vehicle (UAV), the margins for error are low and it is always assumed that risks and uncertainty are high.

In order to mitigate these risks, various methods for 'Safe Exploration' within RL have been proposed and developed. This paper goes into detail about one of these methods, which involves the use of Artificial Potential Fields (APF). Applying the concept of 'safety' to RL has been treated by labelling[1–3] , ergodicity,[4] costs[2,5] and variance of expected return.[2] Furthermore, safe RL approaches are classified into the *Optimisation Criterion*[6,7] and the *Exploration Process*[6,8] . APFs, being a mature field of research, have been used on manipulators and mobile robots,[9] while combining them with RL is still a growing field of research.[10]

Investigating the use of APF methods in the context of RL agents acting on uncertain systems and partially known or unknown environments is the leading motivation for this paper. Despite APF methods being used frequently in path planning tasks, that is not per-se the goal here. Instead, these methods are adapted to aid in the following of more generalised goals, such as attitude or velocity targets. The APF approach is required not only to provide safety, but also to accommodate a degree of freedom for the agent, in order to allow learning through state exploration.

Four levels of combining APFs with a Q-Learning based algorithm are proposed, ranging from a purely potential based approach to one that is solely reward based. These are then analysed and tested in a gridworld simulation including inertia effects. The paper first introduces the fundamentals of this research, namely Safe RL and APFs in Section II, after which the experiment and analysis approach are detailed in Section III. Section IV then presents the results and the paper is concluded, with recommendations for future work, in Section V.

II. Fundamentals

II.A. Safe Reinforcement Learning

The concept of safety has been incorporated within RL in two major branches, the 'optimisation criterion', and the 'exploration process'.[6] The first deals with optimising pre-defined performance metrics (minimise costs, maximise rewards etc.), while the latter tries to prevent dangerous exploratory actions on the part of the agent. Algorithms such as 'Safety Handling Exploration with Risk Perception Algorithm (SHERPA)[11]' have been shown to enable an agent to successfully explore its environment under limited prediction capabilities. This algorithm is also validated on a quadrotor model, further solidifying the applicability of this research to an aerospace context. It must be noted here that 'safety' as a concept within RL is still a growing field. However, some attempts have been made to propose and generalise the definitions of various types of safety. This is presented in the following sections.

Types of Safety

'Safe Exploration' in this context could be intuitively thought of as any 'schematic or algorithm that enables the agent to avoid fatal states and suboptimal lead-to-fatal (LTF) states[11] while rapidly learning about the environment and its dynamics'. LTF states are defined as those that, 'when visited, will lead the agent to end up in the fatal state space with probability one'.[11] Pecka and Svoboda[2] published an overview of safe exploration in RL in 2014, where they collected the various definitions used so far in this non-standardised field and a short discussion of each follows.

Labelling: Referring to a transition from state s to s0 while carrying out the action a and receiving reward r with the tuple <s; a; r; s0 >, Hans[1] defines four broad regions in order of decreasing safety, namely, 'safe', 'critical', 'supercritical' and 'fatal'. A transition is fatal if r is below a certain threshold, while a safe policy leads only to safe states by using non-fatal transitions. In a 'supercritical' state, 'there exists no policy that would guarantee no fatal transition for an agent starting from state s',[2] i.e., the state that the agent currently is in, s, can, with a very high probability, give rise to a fatal transition. Finally, the state from which the agent chose to take a supercritical or fatal action is itself called 'critical'.

Garcia and Fernandes introduce the Policy Improvement through Safe RL (PI-SRL) algorithm which aims to 'improve baseline policies in dangerous domains using RL'.[3] In order to determine what is safe, four regions are defined, 'Known States', Unknown States, Non-Error States and Error States. The algorithm first learns the 'known-space' from the baseline policy, which is assumed to be safe and suboptimal. The

'unknown' spaces are then adjusted in order to explore, while avoiding the 'error' areas. The exploration is carried out by 'perturbing the state-action trajectories with the addition of Gaussian random noise'. A risk-parameter σ is defined which allows the user to set how much noise (and therefore risk) he/she is willing to add to the exploration process.

Most other definitions of 'safety' in literature in terms of labels follow from the ones discussed above but the crux of the matter is that there are safe states, fatal states and critical states, the latter being ones which have the potential to lead to a fatal situation.

Ergodicity: Moldovan and Abbeel present an algorithm that 'guarantees safe, but potentially suboptimal exploration'[4] by formulating safety through ergodicity. An ergodic MDP is defined as one where an agent can (with probability of at-least a certain δ), by following a suitable policy, reach any state, starting from any other state. The physical interpretation here is that any 'mistake' can be reversed. Safe policies are defined as those that 'preserve ergodicity with some well controlled probability'. However, ergodicity must be carefully defined and discussed in terms of applicability. A UAV could find itself in a situation where, due to a dynamic environmental change or a system fault, it cannot return back to its original state. This violates ergodicity but does not automatically mean that the UAV is in an unsafe state.

Costs: Heger[5] presents a discussion on why using policies derived from minimising the expected total discounted cost is not always reliable in his paper on considering risk in RL. Here, a cost is assigned for taking an action/being in a state and the worst-case cost of the generated policies is minimised. Pecka brings up the valid point that using the cost method 'leads only to the safest possible policies, which are not necessarily safe'.[2] Therefore, any formulation involving cost assignment must be carefully set up.

Variance of Expected Return: This is somewhat of an extension to the cost minimisation method of ensuring safety, as it recommends minimising both cost and its variance. A safe policy is one that minimizes the number of critical actions, since 'fatal transitions are expected to yield much larger costs than safe transitions, increasing the variance significantly'.[2] This is also expected to be a highly restrictive approach as there could be situation where the change in returned cost from episode to episode is high, without necessarily putting the agent in a fatal situation.

Safe RL Approaches

Approaches to safe RL have been segmented into two distinct sections in literature, namely algorithms that influence the *Optimisation criterion* and ones that modify the *Exploration process*.[6] Despite these two branches, a modification to the optimisation criteria will invariably influence the exploration process. However, the first branch considers those methods that include some form of risk in the optimisation criterion while the second branch considers those methods where there is a fixed optimisation criterion, but the exploration process is modified to include risk.

Optimisation Criterion: Finding a function that guides which actions to take in which states (optimal control policy), while optimising a defined criterion is the basis for most RL problems. One proposal is the **Worst-Case Criterion** where the objective is '*to compute a control policy that maximises the expectation of the return with respect to the worst case scenario incurred in the learning process*[6].' Using the minimax equation, this approach mitigates variability due to system stochasticity and/or parameter uncertainty. This is an attempt at decreasing risk and selection of actions that lead to undesirable states. Any approach that includes a parameter that allows the '*sensitivity to the risk*' to be controlled, falls under the **Risk-Sensitive Criterion**. There are methods based on exponential functions and on the weighted sum of return and risk. This approach attempts to avoid catastrophic situations even if their probability of occurrence is very small. However, typical behaviours here include the underestimation of risk due to the ignorance of improbable but severe events. Furthermore, 'mean-variance' optimisation 'can directly lead to counterintuitive policies' (cited from Garcia,[6] and inferred from a study on mean-variance optimisation infinite horizon MDPs[7]).

Exploration Process: Methods classified here include the consideration of 'risk' within the exploration process while keeping the optimisation criterion unchanged. The motivation for doing this as opposed to classic RL exploration strategies that rely on some randomness, e.g., ϵ-greedy, is that these random policies require the agent to explore and learn from scratch, which will almost certainly lead to catastrophic actions

being taken. Furthermore, random exploration is time and resource consuming as irrelevant regions of the state-action space are explored. The bulk of the methods here lead on from the conjecture that 'it is impossible to completely avoid undesirable situations in high-risk environments without a certain amount of external knowledge'.[6] Three approaches to using external knowledge are to **Provide initial knowledge** to prompt the agents' exploration in the right direction, to **Derive a policy from a training set**, or to **Guide exploration through teacher advice**, where a 'teacher' model (generally a partial Q-function) is used to provide information when it is considered necessary. *Learning from Demonstration* (LfD) or apprenticeship learning is another often cited approach,[8] where performance is heavily based on the training set. If the initialisation does not provide information for all important states, the agent will end up with a suboptimal and possibly fatal policy.

II.B. Artificial Potential Fields

APFs have traditionally been used to develop obstacle avoidance strategies for manipulators and mobile robots. The APF itself is described by Khatib as a 'field of forces' where the desired goal position is an 'attractive pole' for the manipulator and obstacles are surrounded by 'repulsive surfaces'.[9] A defined potential function typically represents the generalised shape of this field and is made up of two distinct 'attractive' and 'repulsive' definitions. The former can be thought of as an energy well in a contoured surface which, by virtue of its shape, drives the agent to the bottom (point of lowest potential). The 'contours' in this surface come about due to the presence of repulsive potential barriers placed around obstacles (or fatal regions).

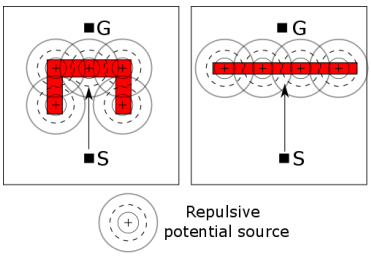


Figure 1. Examples of minima using APF (agent goes from start (S) to goal (G)

The idea of using APFs can be generalised to undesirable regions in the whole state-space that the agent should learn to avoid, not just physical obstacles. An RL agent is expected to benefit from the direction provided by an APF in its environment, at least in the context of increasing safety of exploration. Combining APFs and RL is not a very mature field of research and the issues identified with the use of APFs for robots are expected to extend to an RL agent as well. The key issue is that of the agent getting stuck in local minima, as illustrated in Fig. 1.

In order to mitigate this problem, the methods proposed in literature follow one or more of the following strategies.

- Following of the same path out of the local minimum (backtracking) and then using another strategy.
- Random movements in order to 'explore' itself out of the minimum.
- Using a procedural planner.
- Designing more complex APFs that are minima free (harmonic PF) or adaptive based on whether the minimum is local/global.

Keeping the aforementioned and the context of this paper in mind, a simulation is developed which allows for an analysis of the interaction between RL and APFs. This is not necessarily intended to be a study in designing the optimal artificial potential function and the values are therefore chosen procedurally and based on experience. The RL agent is expected to use the APF as a substitute for an environment model, which it is not provided in the beginning. Using this learnt information, it is expected that the action selection behaviour of the agent will inherently be safer than not using APFs.

III. Experimental Setup

In order to explore the feasibility of combining APFs with RL algorithms with a view to increase safe exploration, a MATLAB simulation environment is developed. The overall goal is to simulate an agent traversing an environment which includes states that are to be avoided, e.g., a UAV with an obstacle avoidance task. The choice to keep this simulation as a discrete, 2D world is taken in order to focus on the APF + RL combination rather than visual extensions, which can always be included in future iterations. Section III.A explains the model in more detail, after which the analysis approach to be taken is explained

in Section III.B. Finally, some implementation specifics are elaborated upon in Section III.C, followed by a discussion of the results in Section IV.

III.A. Simulation

The simulation includes four states, namely velocities and positions in x and y directions **Vx, Vy, x** and **y**. The environment is an 11x11 grid world, where some of the grid-squares are considered obstacles. The agent, represented by a black cell in Fig. 2 has the task of learning about its environment, and over time, maturing a policy that allows it to avoid the obstacles and reach its goal cell, G, with no overshoot, beginning from the start cell, S.

The goal state, **G** may be represented by Eq. 1. If the agent passes over the goal position in its run, it is said to have overshot the goal. This is elaborated upon in Fig. 3. Here, a represents the change in velocities taken as action by the agent. On the left, the agent already has velocity values of 3 in both x and y directions. The action taken is $(0, 0)$, namely no velocity change. This takes it through a path that goes over the goal state. The agent carries out this movement one step at a time, where, at the second step, the agent will be at the goal position. However, due to the fact that it still has residual speed, it will end up overshooting. The situation on the right is of a valid goal state. While there is no overshoot, it must be noted here that both scenarios have residual inertia. It is not imposed that the agent must reach the goal position with zero velocities, as discussed previously.

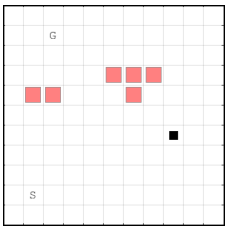


Figure 2. Gridworld with obstacles, start (S) and goal (G) positions

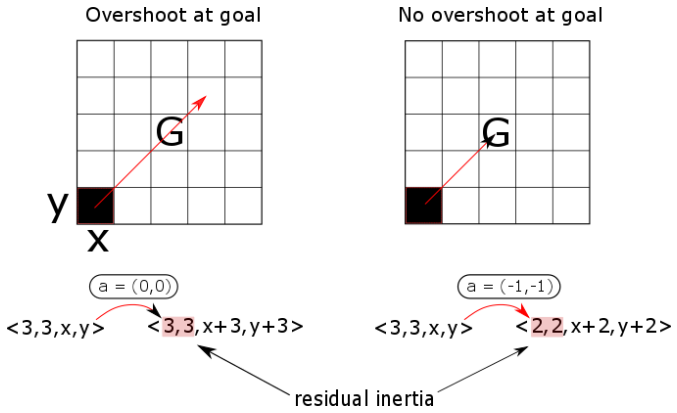


Figure 3. Goal Overshoot scenario

The goal state is made up of a user chosen position and velocity. For the latter, perhaps the most realistic case would be to impose a zero-velocity condition on the agent, i.e. it should arrive at the goal location with no residual speed. However, this is also the most restrictive and therefore difficult, condition. In order to ensure that the simulation results can be interpreted and compared, a less restrictive goal condition is set, namely that the agent should learn and condition its algorithm such that it arrives at the goal position with no overshoot instead of no velocity. In order to clarify this, it is imperative to explain the actions that are available to the agent. Every step, the agent has the option to increase or decrease its change in horizontal and vertical velocity, namely ΔV_x and ΔV_y. Once this is set, the agent can move one block at a time, where the number of blocks moved (and the direction) in that step is determined by the current position, velocity and change in velocity at that state.

$$\begin{aligned} G_{ideal} &\rightarrow < 0, 0, x_{goal}, y_{goal} > \\ G_{sim} &\rightarrow < V_x, V_y, x_{goal}, y_{goal} > \end{aligned} \tag{1}$$

For completeness, the range of possible values for this simulation can be seen in Fig. 4. There are 49 combinations of velocities that are possible in this simulation (7 from V_x and V_y each, the values of which both go from -3 to 3). As far as actions go, each step, the agent can choose from decreasing or increasing its

velocity in x and y direction by 1, or not changing it at all. For clarity, the whole set is written out in Fig. 4. These values are chosen based on an informal tradeoff between model size (and therefore computational complexity and time), and return in terms of effect observability. What is meant by this is that the chosen size still allows for the observation of potential based interaction effects. A bigger state-space by magnitude would not add much.

Regarding the implementation of an APF, there are two main questions to be tackled; how to store the observed 'potential' information, and what to do with it. Extending the Q-learning concept of assigning what is essentially an 'estimated utility' value to each state, a P-table is devised to store representations of potential for each state. This is an [nx1] table, where n is the total number of states (in this case, 5929 [7x7x11x11]), where the presence of potential in each state is recorded in the appropriate cell. The magnitude of this number represents the strength. It is assumed that the agent has sensors which allow it to see 1 step around its current position, much like any proximity sensor (vision based or otherwise) would work on a UAV. Consecutively, the code is written such that when the agent is next to an obstacle, it records its current position as a high potential area, the position slightly further away as a slightly lower potential area, and so on. The specific potential values that are assigned based on agent proximity to any obstacle (obs), are detailed in Table 1. This is set arbitrarily following the only requirement that the magnitude of the potential should decrease as the distance to the obstacle increases. The agent 'perceives' the 1-step away (in all directions) potentials and stores it in a [3x3] construct known as `Pnet`. It must also be noted here that, since the potential 'field' is only distance based, if a potential is detected at a certain state, that same potential is assigned to every state with the same position values. Each position will therefore have 49 associated states due to the possible velocity value combinations.

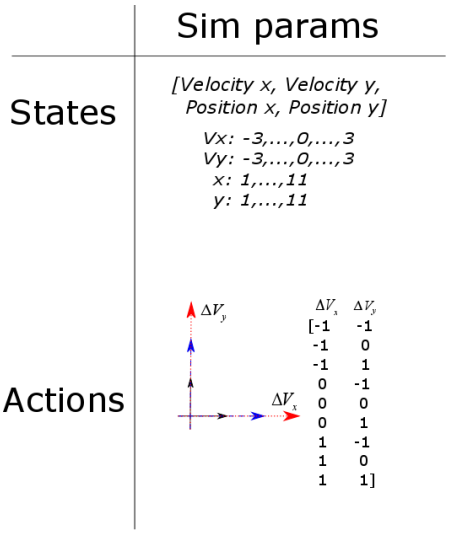


Figure 4. Model parameters

These P-table values are then used to influence the actions of the agent, which can only 'see' 1-step around, but can take actions that take it much further from its current position. This is a design choice and may not be very appropriate as it could lead to situations such as in Fig. 5 where the agent, which is currently at cell (5,2), detects an obstacle and following its potential field reactionary dynamics, is told to take an immediate action to cell (2,5). However, there is an obstacle spanning cells (3,4) to (3,5).

Table 1. Potential values assigned as a function of position relative to obstacle

Agent relative position (to obs)	Potential assigned
Obstacle	-10
1 step	-5
2 steps	-3
>2 steps	0

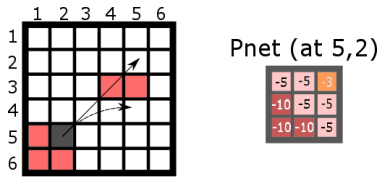


Figure 5. Example of sup-optimal situations with the current method

This is avoided by introducing further inner evaluation loops, i.e., if the agent decides to, based on the potential information and current velocity, take an action that would take it more than one step away, the agent's dynamics restrict it to make multiple action–re-evaluation steps. The agent takes a single step action in the general direction of the initial suggested action and re-evaluates its immediate potential field, which gives it new information. The inner evaluation loops are identical to the outer except for the fact that they do not re-select an action for the agent, unless, of course, there's an obstacle in the way. In that case, the inner loop is broken and the algorithm considers the current pre-obstacle state as the starting point for re-evaluation. The inner loop

actions also do not update the Q-table. This is only updated when the action set is completed or the inner loop is broken.

This brings up the question of how the agent decides where precisely to go. The specific direction is determined using a vector based system, which is best explained with an example. Referring back to Fig. 5, the `Pnet` perceived by the agent when it is at cell (5,2) is as shown to the right of the grid. The values follow from Table 1. It must be noted here that a significant part of `Pnet` is influenced by the potential fields of both obstacle clusters. However, for ease of explanation, the `Pnet` shown in the example is only due to the closest obstacle cluster, i.e., bottom left. Interference of multiple potential fields and how they are dealt with is explained in detail in Section III.C.

The next step is to generate vectors indexed from the agent's current position. Considering the agent in the middle of `Pnet`, the cartesian product of the position indices -1, 0 and 1 is taken as a reference table. The respective vectors are multiplied with their values from `Pnet` and then added together to result in one 'suggested vector' called `P_act`. Using a sign convention of down—positive y and right—positive x, this results in $[y, x] = [-12, 12]$ being the suggested action vector as shown below:

$$\begin{aligned}[-5*(-1,-1)]+[-10*(0,-1)]+[-10*(1,-1)]+\\ [-5*(-1,0)]+[-5*(0,0)]+[-10*(1,0)]+\\ [-3*(-1,1)]+[-5*(0,1)]+[-5*(1,1)] = \textbf{[- 12,12]}\end{aligned}$$

This tells the agent to go top-right, which makes sense since this takes it away from the immediate obstacles, which were on the bottom-left, relative to the agent. The magnitude and direction of this vector is then scaled to result in the action taken by the agent purely due to the potential field.

The P-table provided to the agent is initialised with all zeros. This, of course, implies that actions based on P-values will not be optimal or even correct in the beginning. Therefore, the standard Q-learning update is also implemented as in Eq. 2 and the ϵ-greedy scheme has been adapted to include potential based actions.

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max(Q(s',:)) - Q(s,a)] \tag{2}$$

This is done by programming the agent to carry out actions based on the Q-table (effectively a measure of the 'quality' of each action in each state) if the magnitude of `P_act` is a certain value, κ, implying potential information that is not strong enough to follow. This avoids the agent getting stuck in local minima and helps for initialisation. Furthermore, the Q-table is also minimally perturbed with a small random value for each action in order to move away from the deterministic nature of the problem, and also to ensure that the algorithm can find a maximum Q-value in each state.

The choice of using Q-learning as opposed to other techniques that enable an agent to find an optimal action selection policy, is motivated by ease of use. That being said, the key limitations of RL are expected to be present and consistent across most techniques. Time and space complexity, input generalisation, sensitivity to parameter values and selection of the reinforcement function are all issues that every RL algorithm has to deal with.[12] With Q-learning specifically, since it is a table-based scheme, it induces a high memory requirement, with the 'exponential relationship between the size of the input vector and the size of the state space being a key problem'.[12] In a physical context, pure Q-learning will need to be adapted. This is because it requires visiting all states infinitely many times, which is impractical and would take a prohibitively long time on a physical platform. Approaches such as Chapman-Kaelbling[13] take a first step in tackling this issue by using a statistics based initialisation where only the most relevant states are searched. Despite the aforementioned drawbacks, the simplicity of implementation and the model-free nature of Q-learning motivates the decision for its use as the basis for the test environment. In addition, by its nature, Q-learning learns the values of all actions, rather than just finding the optimal policy and sticking to it. This leads to exploration insensitivity, where 'any action can be carried out at any time and information is gained from this experience'.[14] This is in contrast to on-policy methods such as Actor-Critic learning or SARSA, where actions must follow or nearly follow the current policy. Exploration insensitivity also allows Q-learning to 'learn from other controllers where, even if they are directed toward achieving a different task, they can provide valuable data'.[14] This allows Q-learning to, over time, find a compromise action, having the knowledge from several non-optimal actions.

III.B. Analysis Approach

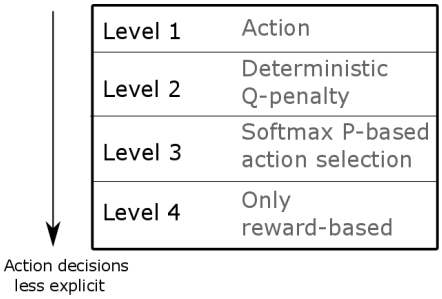


Figure 6. Levels of Analysis

Now that the model is established, it is adapted to investigate combining potential fields. The agent can perceive potential information on varying levels of autonomy. This can range from taking an action that directly follows on from an obstacle's potential field, to letting the agent use this field to only influence its reward function. In the latter case, the agent is expected to, over time, condition its Q-table such that safe 'potential-influenced' actions are taken without being explicitly told to do so.

Four levels of analysis are identified and can be seen, represented from high to low level, in Fig. 6. All these methods are implemented within an 'ϵ-greedy' shell. In order to avoid consistent exploration even after a good policy is found, while still ensuring that the agent does not exploit its established policies excessively in the beginning, ϵ has been designed as the monotonically decreasing function $\varepsilon = a(b^{-n_{ep}})$. Here, ϵ is set as a function of the current episode number n_{ep} and the coefficients, a and b, set empirically and respectively as 0.6 and 1.0015.

1. **Level 1:** Here, the agent is directly given an action in the potential zone, effectively overwriting the Q-table and anything learnt there. One parameter that influences how sensitive the agent is to the potential information is κ. A very high κ implies that the agent will ignore the potential-based action and stick to standard Q-learning (essentially Level 4), while a κ of zero means that the agent will always follow the action suggested by the direction of the potential field. Neither leads to an optimal policy and therefore κ must be carefully tuned, as shown in Section IV.

2. **Level 2:** Here, instead of directly affecting the actions in the potential zone, the agent deterministically shapes the Q-values of the appropriate state-action pair (perhaps even ones in close proximity to it, eg. similar actions or states around the current one). The Q-shaping could be stochastic, or simply deterministic, such as in Eq. 3 where Q_f is the final updated Q and Q_p is the potential zone's contribution to the Q-value, and q is a tune-able gain.

$$Q_f = Q + qQ_p \tag{3}$$

3. **Level 3:** Similar to Level 2, but using a non-deterministic operator such as 'softmax' for action selection. The motivation for using 'softmax' is that it shows positive behaviour in 'settings where it is necessary to maximize utility but also to hedge against problems that arise from putting all of ones weight behind a single maximum utility decision'.[15] Adaptations of the softmax operator for reinforcement learning have been explored with positive results by Asadi and Littman where the motivation for proposing a slightly adapted version called 'mellowmax' was to present an operator that is 'both *non-expansion* (ensuring convergent behaviour in learning and planning) and *differentiable* (making it possible to improve decisions via gradient-descent methods)'.[15]

   Furthermore, this is interesting to explore since the implementation of the potential information within the action-selection procedure is non-trivial. Fig. 7 shows the standard softmax equation and a few suggested modifications to it.

   Convergence to a locally optimal policy has been observed with the first two modifications. However, there are two points to note here. Firstly, softmax includes a 'temperature' parameter, τ which is

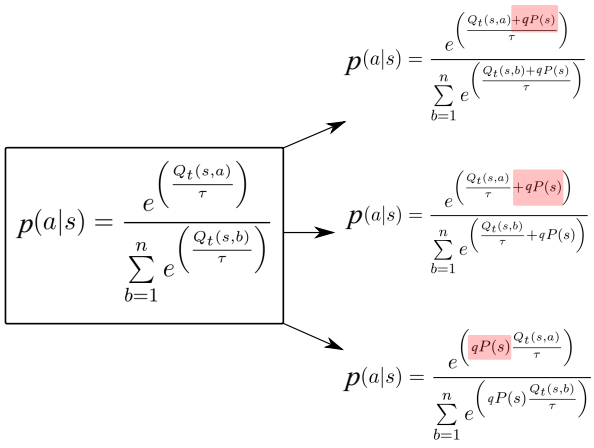


Figure 7. Modification of Softmax to include Potential

not intuitive to tune. Starting from the fact that as τ tends to infinity, all the actions approach equi-probability and as τ tends to zero, there is a greater difference in selection probability for actions that differ in Q-values, a value of $\tau = 50$ has been set. In addition to this, these modifications add another tune-able parameter, q, which is intended to condition the potential strength values (Similar to the factor *'q'* in Level 2). Tuning these parameters requires a more thorough analysis of the system. At the moment, no comparative study has been found on how to set these parameters (at-least on τ).

Secondly, as can be seen from Fig. 7, the 'softmax' equation returns a normalised probability of action selection given that the agent is at a specific state. This is intuitive since the probability depends on a Q-value that itself is dependent on a specific state-action pair. However, the potential information available to the agent is **only state-based**. By including this information in the softmax equation, the implicit assumption is being made that regardless of the action that is being taken at a given state, the effect of the potential is the same. This is a conscious choice made to simplify the problem. Suggestions to deal with this are discussed in Section V.

4. **Level 4:** This is the most 'hands-off' approach since only the rewards are influenced by the potential information. The agent is not actively barred from going into a potential zone. Rather, it is expected that the Q-update should eventually realise that the potential zones result in higher negative rewards and should automatically suggest actions that lead the agent away. This is done by setting $\kappa > 30$, as discussed in 'Level 1' [a].

III.C. Potential Interaction

Potential fields around agents will interact if the obstacles are placed close together. This interaction is shown in Fig. 9 where the raised surfaces are obstacles and the 'depressed' cell represents the potential 'pull' of the goal. In case of potential fields interfering due to obstacles being close to each other, the potential value for those cells are determined by adding the original potential values (due to their respective obstacles) and then multiplying it with a merge-factor, $(0 < m < 1)$, 0.75 in this case[b]. The only requirement while setting this value is to ensure that the inter-obstacle space should not be able to have a higher negative potential than an obstacle itself. This is done to prevent the agent from coming across any situations where it determines that the best option is to go through an obstacle.

$$P_s = m \sum_{i=1}^{N} p_{s,i} \tag{4}$$

[a] **30** is the highest possible magnitude of the vector '`P_act`', based on the potential field values used throughout this paper

[b] The value for m has to be optimised based on the potential strength magnitudes used. If a sequence of doubling $(-10, -5, -2.5, -1.25)$ is followed while setting the potential strength values, then, regardless of what the merge factor used is, there will be no issues with overlapping potential strengths being higher than the obstacles themselves. In this simulation, the sequence $(-10, -5, -3)$ has been used arbitrarily and hence 0.75 has been determined to be an appropriate value for demonstration. Future work will be adapted to stick to the doubling sequence

This is visualised in Fig. 8, where it can be seen that potentials in the 'interfering' states do not exceed the -10 value assigned to the obstacles. Eq. 4 presents a generalised formula for potential strength assignment for each state, where P_s is the potential value at a given state, $p_{s,i}$ is the potential due to obstacle cluster i and m is the aforementioned merge-factor. The total number of obstacle clusters in the simulated world is denoted by N.

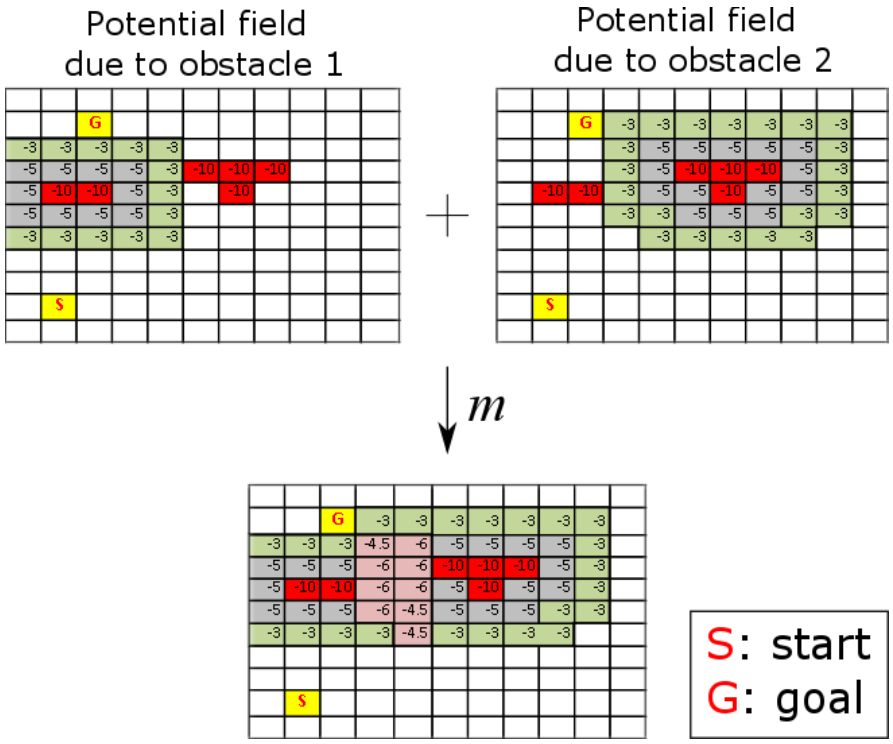


Figure 8. Potential strength for interfering fields

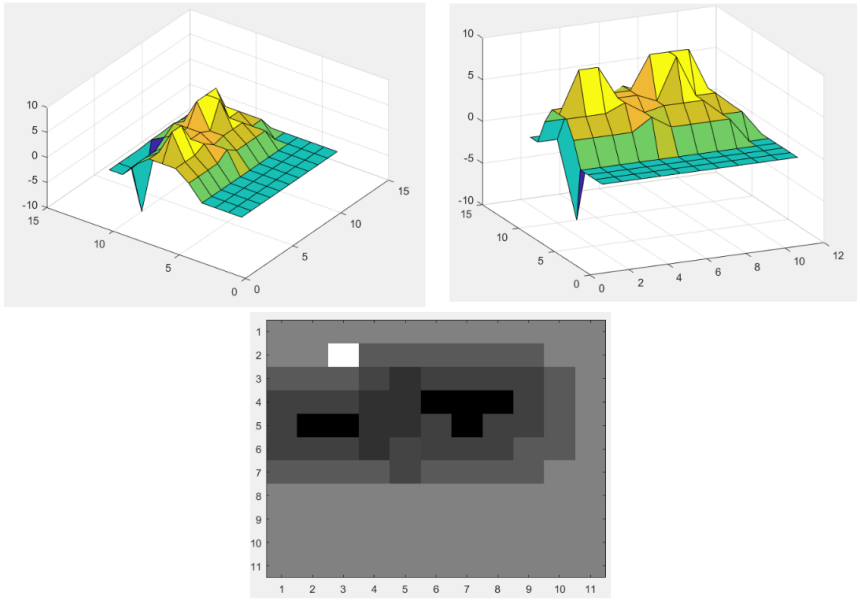


Figure 9. Potential Field render

IV. Results

The model is explored in 2 aspects, one in terms of *framework* and the other, in terms of *setup*. The former refers to the 4 levels of analysis elaborated upon in Section III.B, out of which Level 1 and 4 are taken as comparison baselines and Level 3, referring to the adaptation of softmax to include potential field information, is explored in more detail. Level 2 will not be explored in this paper due to the difficulty of obtaining an insight into how Q-values are actually affected. 'Setup' refers to the obstacle environment itself. It is most interesting to put the agent in a situation where action selection and policy convergence is not trivial. Any algorithms using APFs are naturally susceptible to areas of local minima and 3 different setups have been proposed in Fig. 10 in order to observe how the agent would deal with these situations.

In order to present the reader with a visual representation of the results, these are presented in 2 distinct sections, one for Level 1 and 4 (essentially the same framework but with different κ values) and one for 2 softmax variations (classified as Level 3). Each section contains 2 sets of results. The first set contains a plot of **rewards per episode**, obtained over the whole run of 2500 episodes and a graphic of whether the **goal has been reached per episode**. The second presents results of observed **collisions** and **convergence** in more detail.

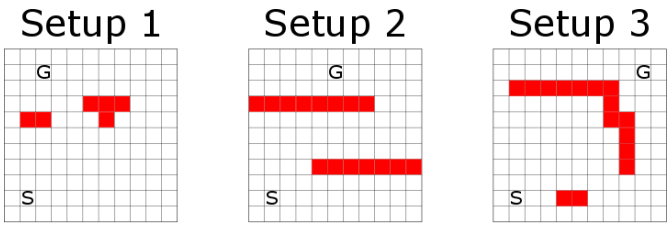


Figure 10. Proposed Simulation Setups to analyse agent adaptability to local minima

Fig. 7 first introduced the proposed softmax variations. The first and second modifications can be considered similar, the only difference being a scaling with the temperature parameter, τ. Therefore, for this analysis, the second and third modifications, shown in Eq. 5 and called Level 3b and Level 3c respectively, are chosen. These are expected to show different behaviour patterns since, in the second case, when the parameter q is set to zero, the exponent contains a pure Q-term and the agent is expected to act greedily. However, in the third modification, the $qP(s)$ term is multiplied, and setting q to zero would result in the whole exponent being zero. This is expected to result in random behaviour on behalf of the agent.

In the first set of results, the second figure shows whether the agent arrives at the goal state during any run in the episode. The second set of figures shows two aspects of the simulation run, convergence and collisions. Convergence can be shown by applying the first distance norm, as shown in Eq. 6, on consecutive Q-tables. Here, **N** and **M** refer to the dimensions of the Q-table, where NM would then be the number of states. In order to make definitive statements about policy convergence and to avoid interference effects from exploratory actions, this norm is applied to the Q-tables obtained after every 100 episodes, where the last episode is run with ϵ set to zero. This ensures a pure greedy policy and shows how the agent is conditioned up to that point.

$$p(a|s) = \frac{e^{\left(\frac{Q_t(s,a)}{\tau} + \mathbf{qP(s)}\right)}}{\sum_{b=1}^{n} e^{\left(\frac{Q_t(s,b)}{\tau} + qP(s)\right)}}$$
$$p(a|s) = \frac{e^{\left(\mathbf{qP(s)}\frac{Q_t(s,a)}{\tau}\right)}}{\sum_{b=1}^{n} e^{\left(qP(s)\frac{Q_t(s,b)}{\tau}\right)}} \tag{5}$$

$$d_1(Q_k, Q_{k+1}) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} |Q_k(i,j) - Q_{k+1}(i,j)|}{NM} \tag{6}$$

Since the velocity values in x and y directions are contained within the state, the magnitude of the resultant vector is taken as the speed value. For situations where the agent takes an action that drives it into a wall or into an obstacle, the average speed is taken as the collision value for that episode. A distinction can be made between the magnitude and number of hits per episode. Furthermore, wall and obstacle hits are differentiated and these are displayed in the last set of figures. For ease of interpretation, instead of displaying the data points for each episode, only one data point is plotted for each 100 episodes. This is obtained by taking the mean of all the data points for each 100 episode set.

This is then followed up by an analysis of the effects of changing key parameters of the L3 framework in Section IV.B. Detailed results for all relevant levels are presented in Section IV.A, the general explanations of which have been introduced in this section. It must be noted that up to this point, these results are

presented only for Setup 1 (refer to Fig. 10). Next, a discussion on velocity and convergence information for the other setups is presented in Section IV.C.

IV.A. Detailed Results

This section presents detailed simulation results for Levels 1, 4 and 3 of the framework introduced previously. The key tunable parameter for L1 and 4 is κ, which essentially regulates the sensitivity of the agent to the potential field. The presented results are for κ values of 0, 15 and > 30. This is because $\kappa > 30$ is equivalent to the agent taking purely reward based actions without taking into account any potential based action suggestions. On the other hand, $\kappa = 0$ only uses potential-based actions. The analysis in this section will tackle the various aspects of the presented graphs individually. The effect of changing κ on the reward profile will be discussed first, followed by a discussion on the collisions. This is then concluded with an analysis of the convergence behaviour.

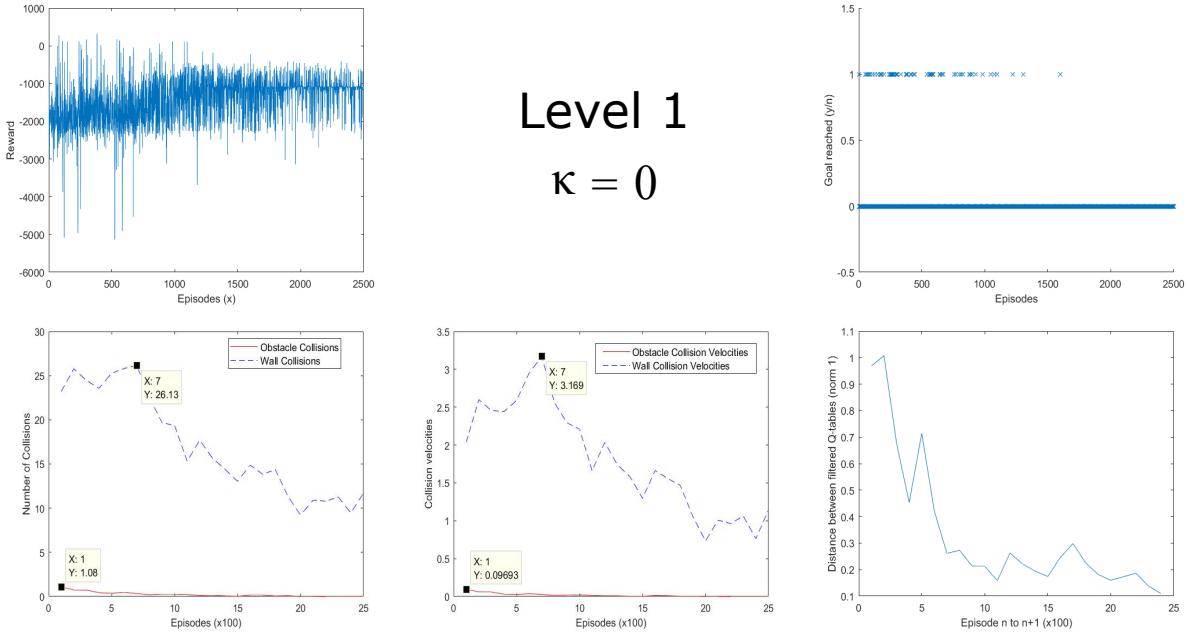


Figure 11. Simulation results for L1 ($\kappa = 0$)

Reward behaviour over the whole training set of 2500 episodes displays significant change as κ is increased. For $\kappa = 0$, referring to Fig. 11, the agent shows reluctant and slow learning. Visually, slight convergence to a final baseline value can be seen. This is the value around which the reward per episode fluctuates for about the last 500 episodes. This is by no means optimal. The decrease in the scale of reward fluctuations over the whole run can be attributed to the decreasing ϵ profile that is used in the simulation. This exploration factor is decreased over the whole run as it is expected that the agent, after learning and conditioning its behaviour over the initial episodes, will not need to explore so much and can instead exploit what it has learnt. Looking at the same plot for increasing κ values in Figs. 12 and 13, it can be seen that the agent converges to an optimal value more quickly and maintains this baseline value for the rest of the run. This occurs earlier for higher values of κ. What this also means is that the reward over the whole run is higher as κ is increased. To put these observations in context, $\kappa = 0$ fully uses the potential information without any regulation. This is counter-productive since the agent avoids any zones which could potentially lead to conflict. It is more content to move around where it started, where it does not observe any potential fields. The goal is therefore effectively never reached despite the actions taken being extremely safe. On the other hand, Level 4 behaviour in Fig. 13 shows favourable performance while having a larger magnitude of fluctuation in the beginning of the run where the agent did not have the potential information as a guide. Another significant observation is that of the **goal-reaching** behaviour. The lower the κ, the less favourable this is. With a κ value of 0, the agent reaches the goal in the beginning due to some lucky exploratory actions, but fails to successfully learn and over time conditions itself to carry out safe but unfavourable (in

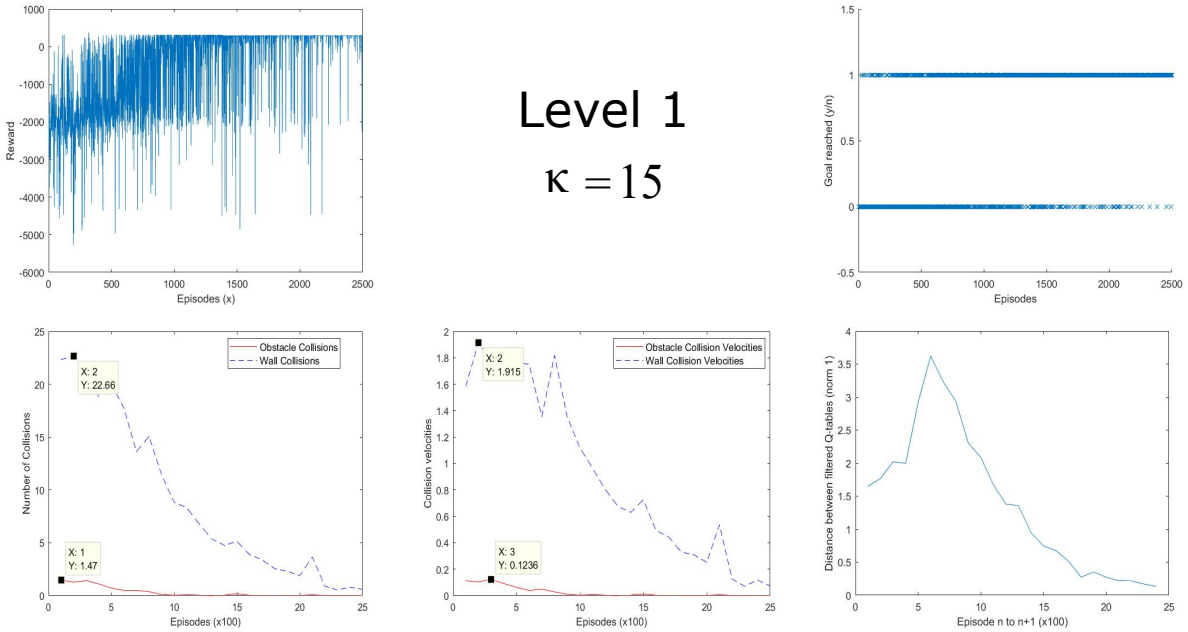


Figure 12. Simulation results for L1 ($\kappa = 15$)

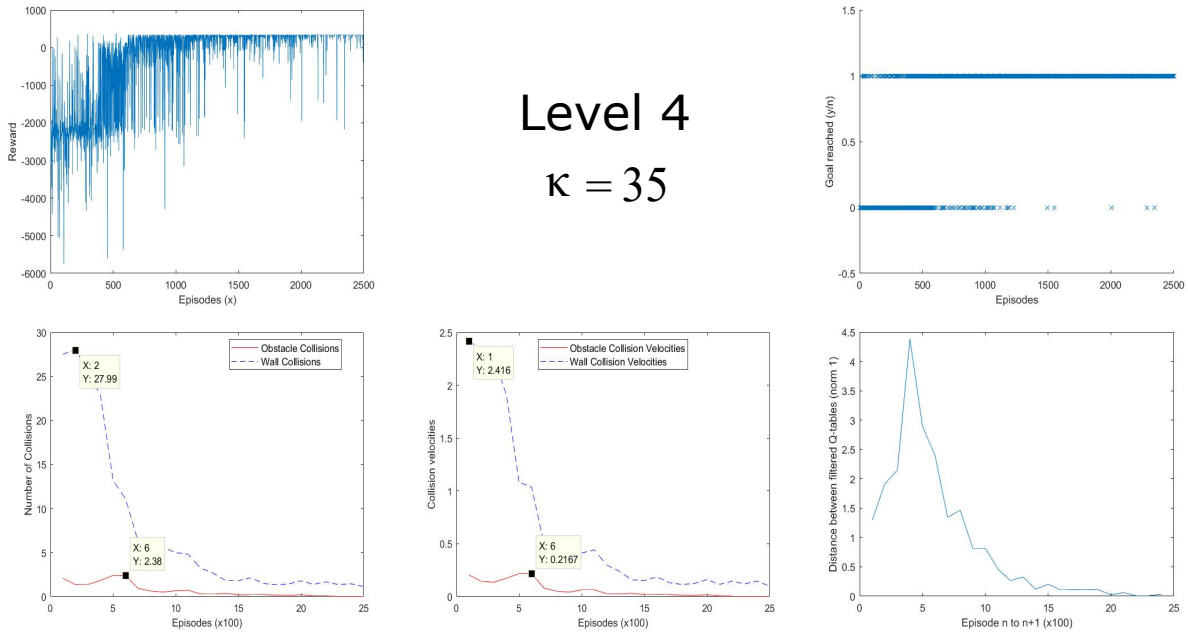


Figure 13. Simulation results for L4 ($\kappa > 30$)

terms of performance) behaviour. In contrast, ignoring the potential field in Level 4 shows behaviour where, for about the final 1000 episodes, the agent consistently reaches the goal. Learning here has therefore been successful. As expected, an intermediate κ setting of 15 shows behaviour that is in between the two discussed cases.

Up until this point, Level 4, where the potential field influence is not felt by the agent, seems to be showing more favourable behaviour. However, the over-arching goal of this paper is to explore safety promoting methods. In order to investigate this, **collision behaviour** must be analysed. An initial observation on all

three settings is that the number and velocity of collisions goes down as the runs progress. This simply shows that the learning is working as expected. However, this is also where the benefits of using potential fields can be seen. Referring to the obstacle collision trend lines on all three figures, both the number and velocity of collisions are consistently higher for Level 4 (no potential) than for either of the Level 1 runs. These values can be seen on Figs. 11-13, 15 and 16 and have also been compiled in Table 2 for ease of comparison. This shows much safer behaviour on behalf of the agent when exposed to the effects of a potential field. It can also be observed over all three frameworks that the number and velocity of wall collisions is much higher than that of the obstacle collisions. This is because the walls are not treated in the simulation as critical and therefore are not considered must-avoid regions. A potential field has also only been imposed on the obstacles and not on the walls.

A thorough analysis has been carried out in order to generalise these observations and the results are presented in Fig. 14. The data for this plot is generated by running the simulation for varying κ values with a Level 1 setting. These values range from 1 to 35, increasing in increments of 5 in order to keep the scale of the simulation manageable. In order to analyse both the performance and the safety of the various settings, two metrics are used, **r_avg** and **safmet**. As can be seen in Eq. 7, the former is essentially the mean return over all the episodes, where i refers to each episode out of 2500 and r_i is the total reward in episode i. This is an indication of performance, as the higher this value, the more reward the agent has accumulated, likely by reaching the goal more times while avoiding too many negative-reward inducing steps. The second metric, **safmet**, reflects safety and is the sum of squares of all the resultant obstacle collision velocity values, over all episodes. Here, V_{t_i} is the resultant velocity value for obstacle collisions in episode i. This, in turn, is calculated by taking the square root of the addition of the squares of the x and y components of the collision velocities. Here, only obstacle collisions are taken into account. The reader is reminded here that the potential field has only been imposed around the obstacles and not on the walls. Collisions into walls, while interesting to observe model behaviour, do not directly reflect the influence of the potential field on safety, and have therefore not been included in the calculation of this metric.

$$r\_avg = \frac{\sum_{i=1}^{2500} r_i}{2500}$$
$$safmet = \frac{\sum_{i=1}^{2500} (V_{t_i})^2}{2500} \tag{7}$$

For each run, the values of r_avg and $safmet^{-1}$ are recorded as indicators of performance and safety behaviour respectively[c]. Fig. 14 shows that there is a tradeoff to be made and an optimal setting where both performance and safety of exploration can benefit. For the current case, this results in a κ value between 10 and 15 but, as discussed previously, this is subject to change, based on the experiment setup. It is also interesting to observe in this plot, as mentioned earlier, that a low κ value, while being extremely safe, suffers from lack of performance. Conversely, a high κ run will potentially learn quickly and reach the goal more times (high performance) at the cost of colliding more often and with higher velocities in the beginning, thereby brining its safety rating down. Neither case is acceptable for an autonomous drone and a trade-off is therefore essential.

The final plot in Figs. 11, 12 and 13 shows convergence behaviour. The norm used to generate these values essentially calculates the distance between subsequent Q-tables, as per Eq. 6. As this value goes down, the interpretation is that the variance in the Q-values decreases. This reflects agent conditioning as it is not exploring and changing the relative weights of its actions as much. Towards the end, as this norm approaches zero, it can be said that, for the majority of states, the agent has picked the actions that it has decided are most favourable. With the exception of $\kappa = 0$ (Fig. 11) where the trend is consistently decreasing, these plots also show an initial spike after which the expected decreasing behaviour resumes. This can be explained by high initial exploration. As this takes place, the Q-table changes drastically every

[c]For consistency, the reciprocal of $safmet$ is taken as the metric for comparison. This way, a higher value is better for both metrics.

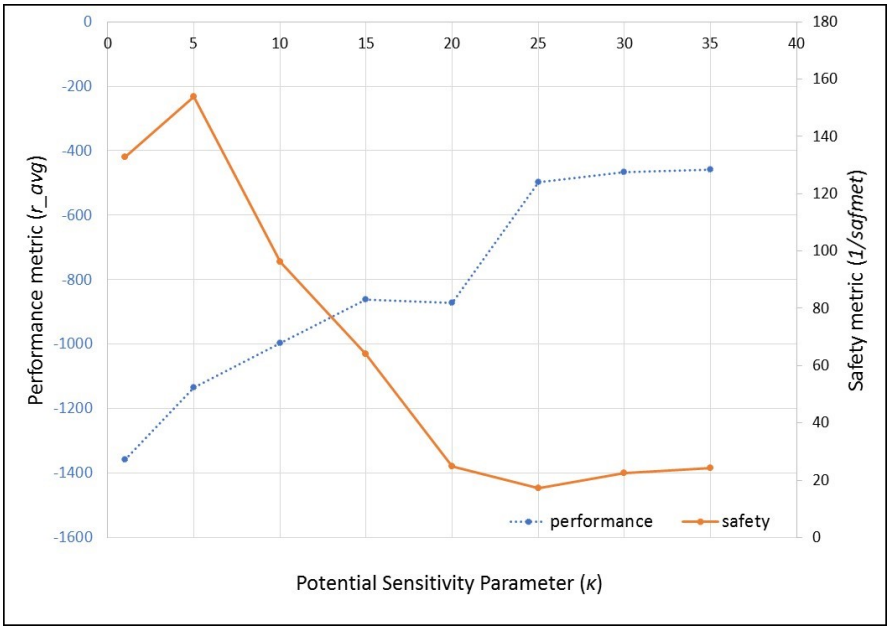


Figure 14. Tradeoff between performance and safety with varying κ

run. Only when the effects of learning can be felt by the agent to some significance and it starts repeating certain actions per state, does the norm start decreasing again. For $\kappa = 0$, the exploration is, from the very beginning, heavily regulated by the restrictions imposed by the potential field. This results in even the initial exploratory actions being conditioned, resulting in relatively less positive change between subsequent Q-tables. Over time, the changes only get smaller as the agent settles into what has been trained.

The discussion so far has focussed on the Level 1 and 4 settings, where κ is the main potential field influencer. However, as discussed previously, the same results can be presented for the softmax based Level 3b and Level 3c modifications, where $q = 40$ and $\tau = 5$. Fig. 16 shows the results for the softmax modification that is not explored in more detail previously and shows precisely why. The fluctuations in the reward plot only marginally improve over the whole run, suggesting minimal learning. Furthermore, it almost never reaches the goal, and in this sense, displays the worst performance seen so far. Furthermore, despite the number and velocity of collisions decreasing over the whole run, the average value is still higher than the rest of the settings, which is undesirable. With regards to the reward plot for Level 3b, shown in Fig. 15, the behaviour here is more favourable and over time, the agent also learns to reach the goal more often as its Q and P-tables get more conditioned. Convergence shows similar behaviour to that of Level 1 and 4, as discussed previously.

Table 2. Detailed collision data (Bold: no potential information used)

	Obstacle Collisions		**Wall Collisions**		
Level	Max Velocity	Max Number	Max Velocity	Max Number	**Mean Reward**
1 (kappa = 0)	0.097	1	3.169	26	-1406
1 (kappa = 15)	0.124	1	1.915	23	-511.7
4 (kappa = 35)	0.217	2	2.416	28	-300
3b	0.248	2	2.332	28	-511.6
3c	0.483	5	2.02	23	-1367

Table 2 summarises the collision data for all the considered frameworks, with Level 4, where no potential information is used, being shown in red. So far, Level 1 and Level 4 have shown the most favourable behaviour. Comparing the number and maximum velocity of wall and obstacle collisions, keeping in mind that it is desired for these numbers to be as low as possible, Level 1 with $\kappa = 15$ displays better performance than Level 3*b*. However, in terms of the whole run, these two levels display very similar mean reward values,

indicating similar performance.

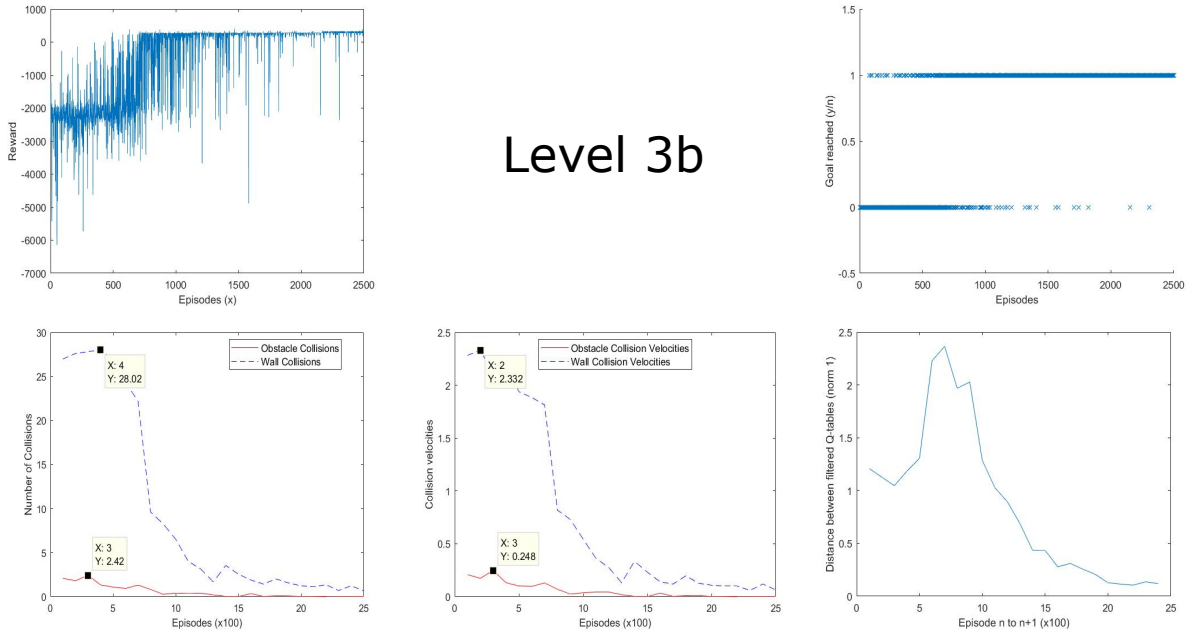


Figure 15. Simulation results for L3b

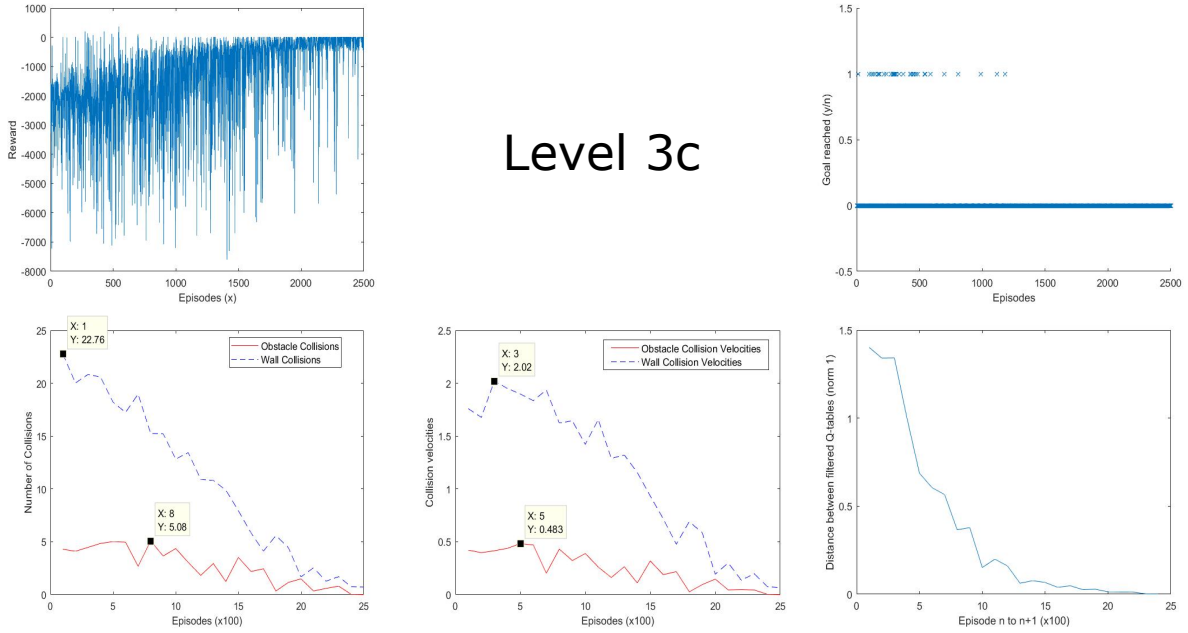


Figure 16. Simulation results for L3c

IV.B. Parameter Analysis

For the softmax framework, based on the disucssion in Section IV.A, Level 3b is chosen for detailed analysis. The experiment can be broken down into the following:

- **Independent Variables:** τ, q

- **Control Variables:** κ, setup, potential strengths (and interpretation method of P-table), reward distribution, number of actions and number of states, maximum number of steps, learning rate (α), discount factor (γ), exploration factor (ϵ) profile
- **Dependent Variables:** performance metric (r_avg) and safety metric ($safmet$)

A standard set of values for τ and q are used to run the simulation, as shown in Table 3. The previously introduced metrics, r_avg and $safmet$ are then recorded in Table 3 for a combination of τ and q values. Each combination is run in the simulation four times and the mean value is then taken for analysis. This is done to mitigate, to some extent, the effects of the stochasticity in the simulation and to achieve result consistency.

Table 3. Performance and Safety Analysis for Level 3 (Softmax based) with varying parameters τ and q

	τ \ q	0	1	20	40	60	100
r_avg	1	*-256.96*	-304.29	-399.79	-265.18	-338.39	-426.31
	5	-373.22	-230.61	***-221.78***	-433.12	-392.28	-360.07
	15	*-372.18*	-479.11	-476.43	-426.93	-473.36	-395.98
	20	-474.63	-540.75	-399.59	-406.15	*-393.8*	-485.45
	30	-540.5	-499.41	*-467.61*	-594.76	-570.47	-508.09

	τ \ q	0	1	20	40	60	100
safmet	1	***0.0383***	0.0472	0.0482	**0.0393**	0.0581	0.0517
	5	0.0560	0.0451	*0.0424*	0.0584	0.0458	0.0565
	15	0.0578	0.0654	0.0591	0.0678	0.0635	*0.0567*
	20	0.0678	0.06	0.0594	*0.0594*	0.0623	0.0610
	30	0.0690	0.0711	*0.0638*	0.0745	0.0719	0.0684

For **r_avg**, it is desirable for the number to be as large as possible as this indicates the highest performance. Conversely, for **safmet**, a lower number indicates safer behaviour for that parameter combination. It must be noted that the table also includes values for simulation runs with $q = 0$. Referring to Fig. 7, this is essentially pure softmax with no potential field influence and has been included for comparison. Furthermore, the italicised numbers signify the best values for each set of τ settings, while the number in bold is the best value over the whole experiment. For the **safmet** table, since the optimum value is for $q = 0$, the next lowest value, which is for $q = 40$ is also brought to the reader's attention and is presented in bold. These results are visualised in Figs. 17 and 18, the former showing the performance analysis and the latter, the analysis for safety. The optimum parameter combinations are also highlighted in these figures.

It must be noted here that these optimal settings are subject to change based on experiment setup and no investigation has been made into the scalability of this algorithm. However, that is outside the scope of this paper. It is interesting to observe that while performance seems to benefit from the addition of the potential scaling factor, q, the trend for safety is harder to generalise. Perhaps counter intuitively, the lowest **safmet** value is for the no potential case ($q = 0$). Looking at Fig. 18, while the expectation is for $safmet$ to show a decreasing trend for increasing q, this is not the case. Some trendlines increase while others show a slight decrease. This is not conclusive and at the moment suggests no safety benefits to introducing the q factor to a Level 3b framework. One reason for this may be the assumption that the potential value is not dependent on the action taken and is simply state based. Both Fig. 18 and Table 3 highlight a value of $q = 40$ as optimal from the ones that do include q. With regards to the temperature parameter, τ, it is used to regulate the probability of action selection per action, over the whole action set. As $\tau \to \inf$, all actions have the same probability, whereas as $\tau \to 0$, the probability of action selection approaches 1 for the action

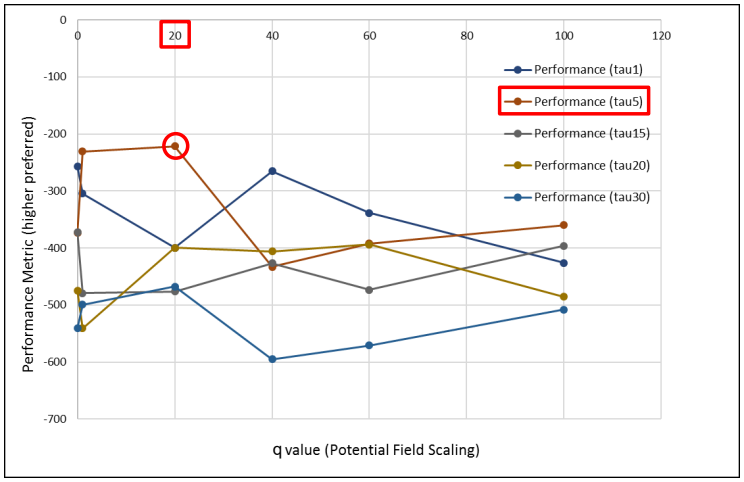


Figure 17. Performance parameter analysis for L3b

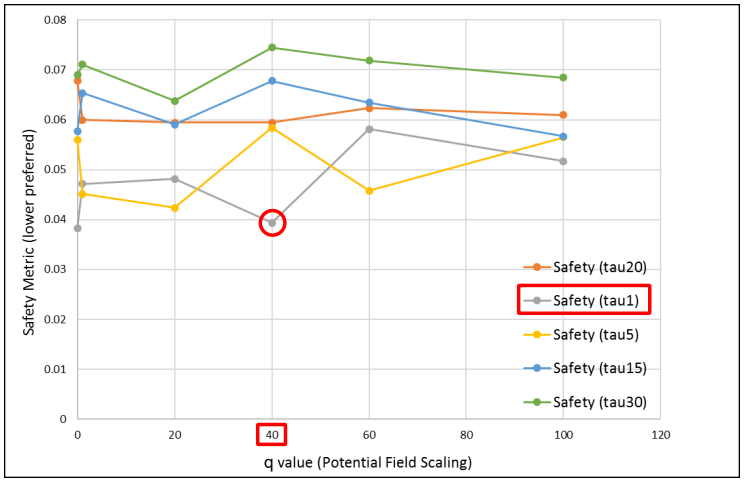


Figure 18. Safety parameter analysis for L3b

with the highest expected reward. Since the potential field information does not directly influence τ in any of the modifications, it is expected to behave and affect action selection in a similar manner. It then follows that a relatively low τ value is seen as optimal. Too high and the effects of learning from the potential field would not be evident (adversely affecting both performance and safety, as can be seen in Figs. 17 and 18) whereas a τ of zero would not be mathematically consistent. A relatively low τ ensures that quick learning takes place, in order to promote safety.

IV.C. Results for Different Simulation Setups

Keeping the previous discussion in mind, a few other simulation setups are explored in more detail here. Specifically, using a Level 1 setting with a κ value of 13 to ensure a favourable balance between performance and safety, this analysis looks at collision behaviour and how the agent adapts to difficult setups. Some examples are proposed in Fig. 10 and are taken as a starting point.

For Setup 2, the trace of the path that the agent takes from start to goal state, after conditioning its Q-table, is presented in Fig. 19. This is expected since the agent gets equally repulsed from both sets of obstacles as it searches for a way out of the 'corridor'. The other hypothesised agent movement would be a 'bouncing' action where it gets repelled by one set of obstacles, only to get repelled back by the other, thereby resulting in an undesirable back and forth action. However, this is avoided by carefully tuning the potential field values.

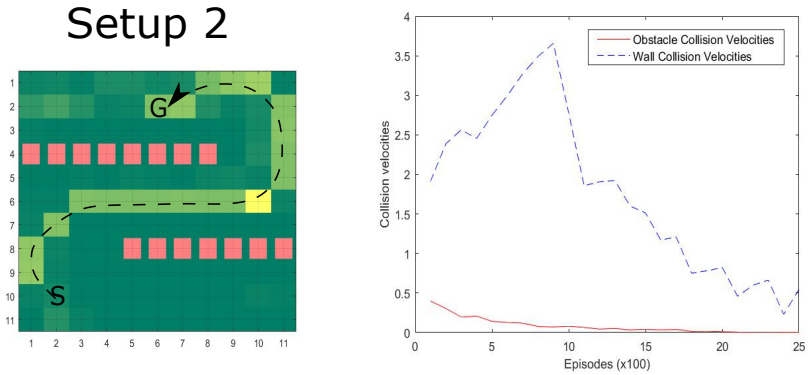


Figure 19. Setup 2: Trace of conditioned path and collision velocities

Fig. 20 shows the trace and collision values for Setup 3. This setup is devised in order to observe how the agent can get out of a highly undesirable potential region and get to the goal which is placed on the other side. As can be seen from the trace, the agent has not totally optimised its behaviour and does not take the shortest path to the goal. However, it has identified that it is more beneficial to go through the gap, despite it having a relatively high negative potential due to interference from both obstacle clusters. Furthermore, the meandering behaviour in the beginning is perhaps necessary to get rid of any excess velocity it may have built up before entering the gap since there is not much room (between the obstacle and the wall) to slow down after the gap.

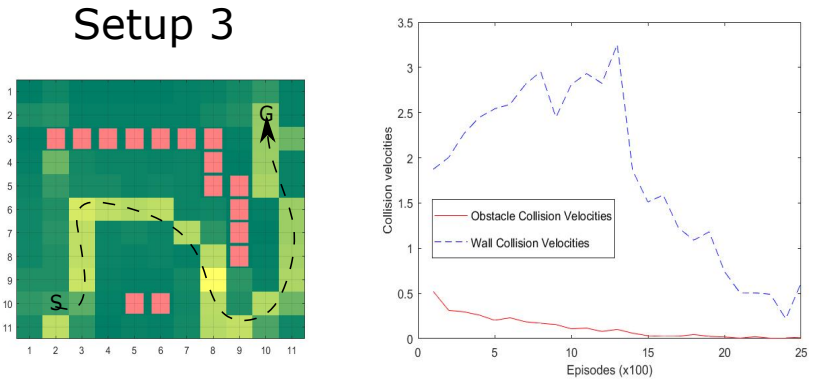


Figure 20. Setup 3: Trace of conditioned path and collision velocities

Both setups, despite being more challenging, show favourable behaviour in terms of collision avoidance. Along with the fact that the collision velocities start at a relatively low value for both, they also decrease as the episodes go by, and for both setups, almost approach zero. This shows the adaptability of using APFs in order to increase safety of exploration. Despite not having a model of the environment in the beginning of the run, the agent quickly learns which states to avoid. This learning process is made safer by adding another layer of information that the agent can extract from its environment, namely the potential field.

V. Conclusion and Recommendations

One of the key issues within RL is the balance between exploration and exploitation. However, before a policy can be considered mature enough to exploit, sufficient exploration needs to take place in order to enable the agent to learn. This process is inherently dangerous. For a software agent, approaching or traversing through a critical state is acceptable and perhaps even desired in the beginning of the episode, since it promotes quicker learning. However, a drone using an RL algorithm to dictate its control laws and overall movement behaviour will need significantly more safeguards to ensure safety. Obstacle collisions, in this case, are considered unacceptable and identified critical states are to be avoided.

Keeping the above context in mind, a Q-learning based test environment is developed, where using nested functions, an RL algorithm is implemented. Various approaches are then proposed where the standard Bellman update equation is modified to include potential values as felt by the agent as it explores its environment. This has an effect on the learning behaviour of the agent and ranges from the potential directly influencing the agent's actions, to it only influencing the reward function. An intermediate approach, Level 2, introduces the concept of a potential based deterministic Q-penalty. This is not explored in detail and can be taken as a recommendation for future work. Level 3, a softmax P-based action selection, is considered in more detail in this paper. An assumption is made that regardless of the action that is being taken at a given state, the effect of the potential is the same.

This research is carried out with a view to demonstrate the applicability of APFs to increasing the safety of autonomous exploration in RL. Despite the need for more research on physical implementation specifics, this paper has shown that safety can, in fact, be improved by interfacing an APF model with a standard RL algorithm. APFs have been shown to be robust and accommodating of the trial-and-error nature of RL and can be applied to uncertain systems in partially known environments. One of the goals of this research that has not been met is the demonstration of a decrease in computational time and complexity while ensuring an increase in safe action selection. This, while being a prompt for future research, can be justified by the exploratory nature of this paper. Novel concepts introduced include the P-table, which stores the agent's perception of the environment as it explores. Furthermore, parameters such as κ, τ and q (the latter two introduced for the softmax based Level 3 approaches) only serve to increase the computational complexity of these methods. However, proof of concept, rather than optimisation, is taken to be the leading motivation of this research, and, in that domain, this paper has met its objectives. The introduction of the velocity states (x and y) have been an attempt in approaching 'realism'. The effects of inertia that inherently lead on from including velocity states complicate this problem and make the results more applicable to a real life situation. That being said, it must be kept in mind that the decision was made early on to use a discrete RL setting in a discrete environment.

In addition, the parameter, κ is introduced, which regulates the sensitivity of the agent to the potential information. This essentially puts the approach under the general classification of the 'Risk-Sensitive Criterion', though not quite entirely. In terms of the Safe RL approaches introduced previously, namely the 'Optimisation Criterion' and the 'Exploration Process', the proposed methods falls in between the two. The fact that the APF is available to the agent means that, through exploring and filling in information about the potential distribution across the states, it is considering 'risk' already in its action selection. Through multiple trials, it is found that there is a tradeoff to be made between safety and performance.

Furthermore, using the artificial potential field (APF) concept itself does not come without drawbacks. Susceptibility to local minima and oscillations due to similar attractive and repulsive forces are a few of the issues faced. In addition, a standard problem here is Goal Non-Reachable with Obstacle Nearby (GNRON). This occurs when the goal is placed close to an obstacle and the agent/robot is repulsed by the obstacle to such an extent that it cannot overcome this force and get to the goal. Most of these issues can be resolved by tuning the actual forces felt by the agent. However, that is not the intention of the research carried out in this paper. For the sake of the simulation, simple values that allow the agent to not encounter these issues have been selected. No attempt has yet been made to generalise at this stage. The fact that minimal previous research has been carried out on the application of APFs to RL further motivates its exploration in this paper.

One of the observations made in this paper was that Level 3b did not perform as expected in terms of safety. The proposed reason is that the potential information is only state based and does not take the action into account. For future work, two suggestions for extension which perhaps make more effective use of the potential information available, are suggested here:

1. The agent also has access to a construct called '`Pnet`' which allows it to observe the potential values in a grid extending one step in every direction from its current state. Using this, one option is to immediately restrict the accessible action-base to those that correspond to potential values over a certain threshold value (this cutoff point can be pre-determined) and then continue with softmax/ϵ-greedy action selection. Using the same `Pnet` as introduced previously, Fig. 21 shows how this works. Here, the cutoff point for no-go actions is a potential value of -10 or lower. Hence, the only available actions are as shown.

   However, there are 2 problems with this. First, it is a brutal approach and perhaps does not belong in this level of autonomy. It is akin to pre-allocating specific potential-based actions for the agent,

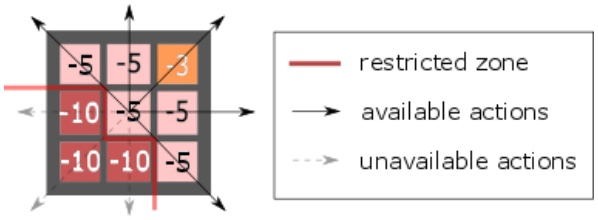


Figure 21. Pnet based action restriction option schematised

similar to Level 1. Secondly, this could lead the agent into getting stuck in local minima as, given a choice between staying in a zone where there is no potential vs going into a negative potential-rich zone, it would choose the former, regardless of the fact that the goal may require it to pass through the potential-rich zone.

2. The other option is to completely rethink what 'potential' means to the agent and to treat it much like the Q-value, where there would be a different 'potential value' for each state-action pair. The risk here is running into problems of dimensionality as now, two rather large data-sets will need to be stored and interpreted. There would then be a need to look at graph-based methods.

The next step would be to adapt these methods to a continuous world. Furthermore, the agent in this simulation moves in two dimensions, x and y. The addition of a third position state would directly make this more applicable to a real life aircraft scenario. The results are expected to be the same, albeit at the cost of computational complexity due to the increase in size of the Q and P tables. Taking these ideas forward with an end goal of implementation on a real drone platform is the next step towards validation.

References

[1]A. Hans, D. Schneegaß, A. M. Schäfer, and S. Udluft, "Safe Exploration for Reinforcement Learning," in *ESANN2008, Proceedings of the 16th European Symposium on Artificial Neural Networks*, no. April, (Bruges, Belgium), pp. 143–148, 2008.

[2]M. Pecka and T. Svoboda, "Safe Exploration Techniques for Reinforcement Learning An Overview," *Lecture Notes in Computer Science*, pp. 1–19, 2014.

[3]J. Garcia and F. Fernandez, "Safe exploration of state and action spaces in reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 45, pp. 515–564, 2012.

[4]T. M. Moldovan and P. Abbeel, "Safe Exploration in Markov Decision Processes," *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[5]M. Heger, "Consideration of Risk in Reinforcement Learning," *Proceedings of the 11th International Conference on Machine Learning (ICML)*, pp. 105–111, 1994.

[6]J. García and F. Fernández, "A Comprehensive Survey on Safe Reinforcement Learning," *Journal of Machine Learning Research*, vol. 16, pp. 1437–1480, 2015.

[7]S. Mannor and J. N. Tsitsiklis, "Mean-Variance Optimization in Markov Decision Processes," in *Proceedings of the 28th International Conference on Machine Learning*, (Bellevue, Washington, USA), pp. 177–184, 2011.

[8]J. de Lope and J. H. Antonio Martín, "Learning Autonomous Helicopter Flight with Evolutionary Reinforcement Learning," in *Computer Aided Systems Theory - EUROCAST 2009* (R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, eds.), no. December, (Las Palmas de Gran Canaria, Spain), pp. 75–82, Springer, 2009.

[9]O. Khatib, "Real Time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[10]L.-j. Xie, G.-r. Xie, H.-w. Chen, and X.-L. Li, "Solution to reinforcement learning problems with artificial potential field," *Journal of Central South University of Technology*, vol. 15, pp. 552–557, 2008.

[11]T. Mannucci, E.-J. van Kampen, C. C. de Visser, and Q. Chu, "SHERPA: A safe exploration algorithm for RL controllers," in *AIAA Guidance, Navigation, and Control Conference*, (Kissimmee, Florida), p. 15, AIAA SciTech, 2015.

[12]M. J. Mataric, "A Comparative Analysis of RL Methods," tech. rep., Massachusetts Institute of Technology, Boston, Massachusetts, 1991.

[13]D. Chapman and L. P. Kaelbling, "Input Generalization in Delayed Reinforcement Learning: An Algorithm And Performance Comparisons," *Proceedings of the Twelfth International Joint Conference on Artificial In- telligence (IJCAI-91)*, pp. 726–731, 1991.

[14]C. Gaskett, D. Wettergreen, and A. Zelinsky, "Q -Learning in Continuous State and Action Spaces," *Australian Joint Conference on Artificial Intelligence*, vol. 1747, pp. 417–428, 1999.

[15]K. Asadi and M. L. Littman, "A New Softmax Operator for Reinforcement Learning," *Cornell University Library*, no. Property 3, 2016.

Part II

Background and Preliminary Analysis

Chapter 2

Reinforcement Learning Basics

The analogy of an exploring infant as an RL agent has been introduced in Ch. 1 in order to give the reader an intuitive understanding of the mode of operation in RL. This chapter will formalise the RL process and introduce some terms and their definitions as will be used throughout the rest of this thesis.

2.1 The RL Model

The RL setting introduces two explicitly separated entities, the *agent* and the *environment*, as can be seen in Fig. 2.1. The agent makes certain decisions where the goal is to accomplish a given task. The environment then represents the 'system in which this task is defined' [4]. The agent interacts with the environment by carrying out *actions* which in turn change the state of the environment. The environment then responds by returning a *reward*, which gives the agent some information about the fitness (how good or bad) of its action in that particular state. This information is important since it allows the agent to adapt its behaviour for subsequent actions. Finally, to complete the cycle, the agent then *observes* the state of the environment and determines what action should be performed next.

This general framework can be applied to various learning problems, such as the one shown in Fig. 2.2. This is a visual representation of an RL agent interacting with its environment and was developed by the author for a previous assignment. The theme of 'dropping' an object such as a camera from an aircraft is taken as the context for the agent. The world is then discretised and modelled as a 20x8 grid, at each end of which there is a start (S) and goal (G) position. This camera could have a high-level goal of following a skydiver. However, the goal is to learn to avoid certain obstacles programmed into this world. Furthermore, a wind profile is included in certain states, which pushes the agent with a certain magnitude in a certain direction if the agent traverses those states. Fig. 2.2 shows the initial schematic of the world as described above, with the grey grids (next to G) being the buildings, and the blue grids (next to S) the wind profile. Additionally, in

Figure 2.1: The Reinforcement Learning Scheme

order to model an object that has reached terminal velocity and is approaching the Earth at a constant speed, the agent is programmed to move 'down' one position regardless of any action that it may take. This is just one example of the versatility of RL and literature has countless others, such as inverted pendulum control[11] or playing classic Atari games[19].

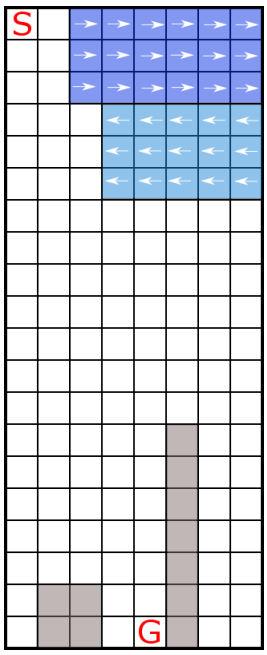


Figure 2.2: Setup of 'skydiving' RL environment with wind profile, agent and S/G positions

2.2 Model Breakdown

This subsection details the various key elements of the standard RL Model, as introduced in Section 2.1 and defines certain conventions that will be carried on throughout the rest of the thesis.

2.2.1 The Environment

The state of the environment is represented by $x(t)$ and varies over time. However, for the purposes of this thesis, most of the discussion will be regarding discrete time RL algorithms. Therefore, the state variable is considered to be observed in discrete time: x_k, for $k = 1, 2...$ It is also to be noted here that most RL algorithms require the state of the environment to be perceived by the agent as *discretised*, taken here to mean that the whole state space is finely divided into individual units. The discretised state is denoted as $s_k \in S$, where S is the set of possible states (state space) and k is the discrete time index. It follows then that the granularity of this discretisation determines how closely the discussion approximates a continuous case.

The environment alters its state based on the action input received by the agent and then provides a reward back to the agent. It is important to distinguish here between *immediate* and *total* reward, the latter conventionally called *return*. The reward received immediately after time step k is denoted by r_k, while the return, perceived by the agent over a period of episodes (or steps in the case of multiple step episodes)[1] is defined as a function of the immediate rewards, $R_k = f(..., r_{k-1}, r_k, r_{k+1}, ...)$. This function could be a simple sum or could use a *discount factor*, γ $(0 \leq \gamma \leq 1)$, as shown in Eq. 2.1 (adapted from [23]). This allows for control over how much weight to assign to current vs. future rewards.

$$R_k = \sum_{k=0}^{\infty} \gamma^k r_{k+1} \tag{2.1}$$

2.2.2 The Agent

The agent needs to carry out an action in order to observe an immediate reward. This action, $a_k \in A$, where A is the set of possible actions, is translated into a physical input to the environment (u_k) in a realistic setting. This u_k can be continuous, while a_k can only take values from the set A. Similarly, the actual physical output of the environment may be denoted as y_k, but the observed state is s_k, as defined before. In this thesis, the discussions will be limited to actions a_k and states y_k.

The agent is tasked with finding an optimal mapping from states to actions $(s_k \rightarrow a_k)$ and this mapping is called the *policy*. A distinction could be made between policies that define a deterministic mapping and ones that give a probability distribution over states and actions. However, for the purposes of this thesis, any policy is denoted as π_k, and may be dependent on only the state or on state-action pairs based on the specific discussion.

2.2.3 Markov Decision Process

The concept of *state*, as used so far, has been taken to mean any observable signal from the environment. It is important to note here that while state signals do include immediately observable measurements based on the sensors available to the agent, the definition

[1] This is schematically represented in Appendix B

is not restricted here. State representations can be 'highly processed versions of original sensations, or they can be complex structures built up over time from the sequence of sensations' [23]. The state is therefore constructed and maintained based on immediate sensory data along with information retained from previously sensed inputs. State signal processing is not discussed in detail in this thesis. The emphasis is rather put on novel decision-making concepts while assuming the availability of appropriately processed sensory data. To put this into context, a simulation of an obstacle avoiding task may take position and velocity data as states, even though the velocity information may not be explicitly available to the agent using it's sensors. It is perhaps calculated instead from the position and time sensors.

A state signal that retains all relevant information is said to have the *Markov property*. The general state-action transition model of the environment that defines the probability of transition to a certain new state s' with a certain immediate reward, r, given the complete sequence of current state, action and all past states and actions, is presented in Eq. 2.2, where $p\{x|y\}$ is the probability of x, given y.

$$p\{s_{k+1}, r_{k+1}|s_k, a_k, r_k, s_{k-1}, a_{k-1}, r_{k-1}, ..., r_1, s_0, a_0\} \tag{2.2}$$

Following the Markov property, Eq. 2.2 may be expressed as in Eq. 2.3.

$$p\{s_{k+1}, r_{k+1}|s_k, a_k\} \tag{2.3}$$

An RL task that satisfies the Markov property can be considered a *Markov Decision Process (MDP)*. Based on the previous definitions, it can therefore be claimed that 'the best policy for choosing actions as a function of a Markov state is just as good as the best policy for choosing actions as a function of complete histories' [23].

2.3 Choice of RL computation method

Having now introduced the components of a typical RL model, a decision must be made on how to put them together and compute the simulation. There are various approaches available in literature, each with their own pros and cons. Fig. 2.3 puts the three main classes on a scale, from ones that need a complete definition of the model, to ones that do not.

Dynamic Programming (DP) methods need a perfect model of the environment, including a full definition of the state and action spaces and a transition model; essentially the whole MDP. In this sense, it is closer to *planning* than to *learning*. Value functions are then used to structure the search for good policies. Starting from an arbitrary policy, π_0, an evaluation and improvement loop is iterated until convergence to the optimal policy, V^*. A key characteristic of DP is that it updates using an estimate of its successor state, referred to as *bootstrapping*.

Monte Carlo (MC) methods, on the other hand, do not need full knowledge of the environment but still carry out a similar policy evaluation and improvement loop. However, these methods only learn when an episode ends, making them infeasible for non-episodic tasks. Furthermore, the expected return starting from state s and following policy π is

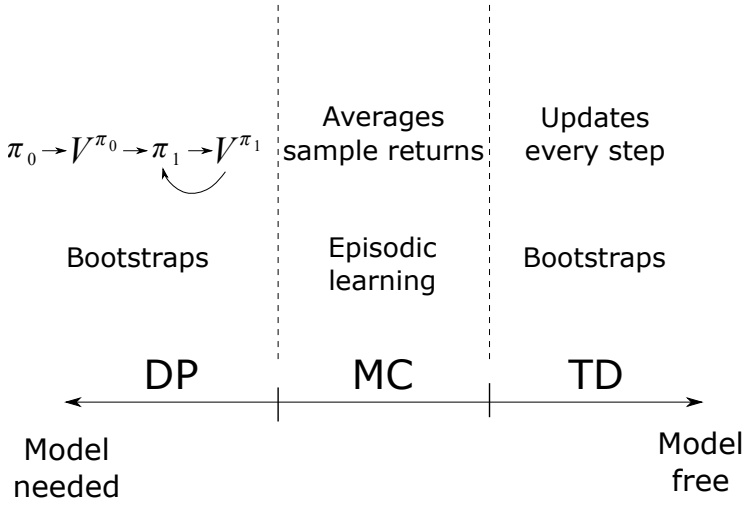


Figure 2.3: Breakdown of RL approaches

computed as the average of observed returns in state s. Being an averaging technique, this also leads to a high variance [23].

Temporal Difference (TD) methods strike a balance between DP and MC. These methods are model-free and can be fully incremental. Unlike MC, learning can take place before knowing the final outcome, i.e. from incomplete sequences. The advantage here is a lower memory and peak computation requirement. Furthermore, much like DP, it bootstraps and 'can be applied on-line, with a minimal amount of computation, and follow from experience generated from interaction with an environment' [23].

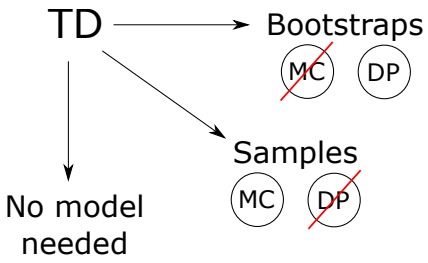


Figure 2.4: Benefits of Temporal Difference methods

Table 2.1: Commonly used TD methods

On-Policy	Off-Policy
SARSA	Q-Learning
Actor-Critic	R-Learning

Fig. 2.4 summarises the advantages offered by TD methods, while Table 2.1 presents a summary of the most common TD methods, divided by those that are *on-policy* and those that are *off-policy*. The difference is the way these methods either use the same policy for all their computations or change it based on the information learnt every step. The on vs off-policy elements in SARSA and Q-Learning are highlighted in their respective

pseudo-codes in Fig. 2.5.

SARSA

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ε-greedy)
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$; $a \leftarrow a'$;
  until s is terminal

Q-Learning

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Take action a, observe r, s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$;
  until s is terminal

Figure 2.5: Comparing SARSA and Q-Learning

Based on this discussion and the relative ease of implementation, Q-Learning is used as the basis of the simulations to be performed in this thesis. It is preferred to use such a TD method, as opposed to Dynamic Programming (DP) in order to generalise the approach and ensure that it does not require a model of the world in order to perform satisfactorily. The Q-update, as proposed by Watkins [25], is shown in Eq. 2.4.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \tag{2.4}$$

Here, α is the learning rate and γ the discount factor. Despite α generally being assigned a single value for the whole run, in order to ensure convergence of the Q-values, it must be a function that decreases based on the number of times a state has been visited. The 'Q-value' is dependent on the agent's current state and the action that it chooses to take. It can therefore be seen as a measure of the estimated utility for each state-action pair. Watkins presents it as 'a simple way for agents to learn how to act optimally in controlled Markovian domains and works by successively improving its evaluations of the quality of particular actions at particular states' [25].

Chapter 3

Safe Reinforcement Learning

A higher level motivation of why safety is important within the RL context has been provided in the previous chapters. The trial and error nature of pure RL methods make them largely infeasible in an aerospace guidance and control task. An autonomous UAV flying from point A to point B cannot afford to learn certain fatal behaviours by trying them out. Hitting an obstacle is generally not an option. This chapter details how safety has been defined in this field so far and places it in the context of autonomous exploration, which is used to guide the methods for safe RL as developed in the rest of this thesis.

3.1 Types of safety

'Safe Exploration' in this context could be intuitively thought of as any 'schematic or algorithm that enables the agent to avoid fatal states and suboptimal lead-to-fatal (LTF) states[17] while rapidly learning about the environment and its dynamics'. LTF states are defined as those that, 'when visited, will lead the agent to end up in the fatal state space with probability one'[17]. Pecka and Svoboda [22] published an overview of safe exploration in RL in 2014, where they collected the various definitions used so far in this non-standardised field. Fig. 3.1 schematises their findings and a short discussion follows.

3.1.1 Labelling

Various authors define safety by labelling certain states and/or actions based on what they consider to be safe. These vary based on the author and the context in which their research takes place.

Hans defines safety as those 'states or transitions that can lead to damage and thus must be avoided' [12]. The research was carried out to optimise the control of industrial plants with a goal to ensure that the exploration of the state-action space does not cause damage to the plant. The major contribution of this paper was the introduction of two components, a *safety function* and a *backup policy*. The former is used to determine a

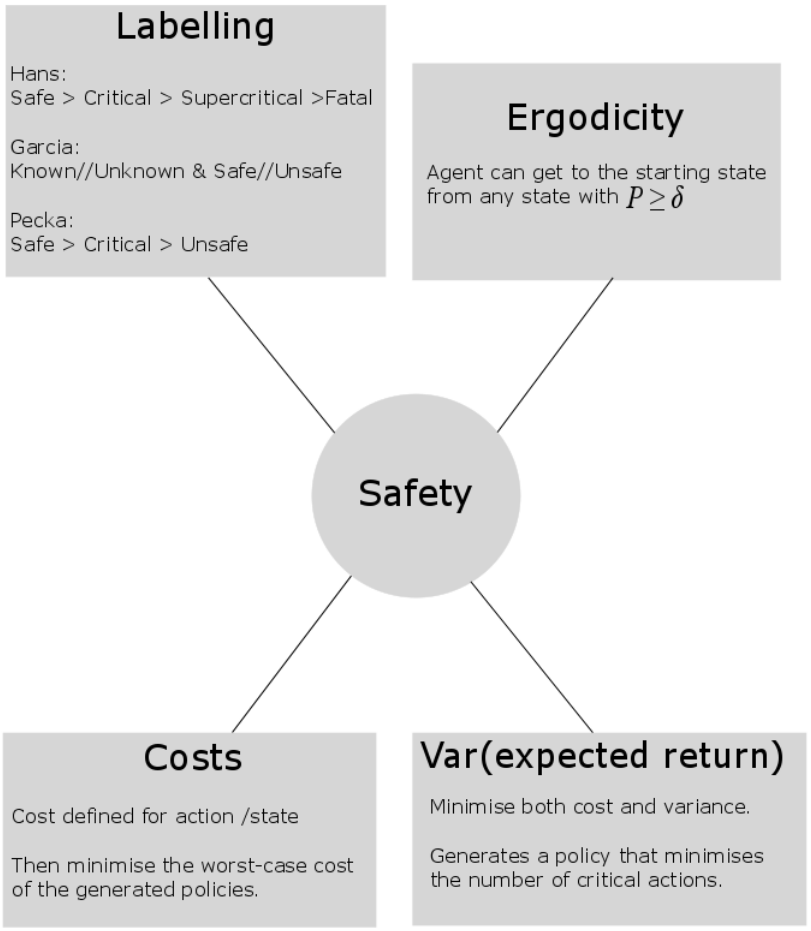


Figure 3.1: Safety summary

state's 'degree of safety' while the latter 'is able to lead the system from a critical state back to a safe one'. In order to take appropriate actions after identifying the degree of safety of a state, it first has to be classified. Referring to a transition from state s to s' while carrying out the action a and receiving reward r with the tuple $< s, a, r, s' >$, Hans defines four broad regions in order of decreasing safety, namely, **'safe'**, **'critical'**, **'supercritical'** and **'fatal'**. A transition is fatal if the r is below a certain threshold, while a safe policy leads only to safe states by using non-fatal transitions. These are logical and follow from our intuitive understanding of safety. However, the definitions of 'critical' and 'supercritical' are more subtle. In a 'supercritical' state, 'there exists no policy that would guarantee no fatal transition for an agent starting from state s [22], i.e., the state that the agent currently is in, s, can, with a very high probability, give rise to a fatal transition. Any action that brings an agent to a supercritical state is itself also called supercritical. Finally, the state from which the agent chose to take a supercritical or fatal action is itself called 'critical'. Taking this one step back, any action that leads to a critical state is itself called 'critical' as well. These definitions can be visualised in a

discretised 2-D world as in Fig. 3.2. It must, however, be noted here that this figure is simply for the reader's understanding and is a simplification which, in some cases, does not strictly adhere to Hans' definitions. For example, an agent in a pink state, called supercritical here due to its proximity to the no-go state (obstacle), could be following a policy that takes it away, i.e., to a critical state, therefore guaranteeing a non-fatal transition.

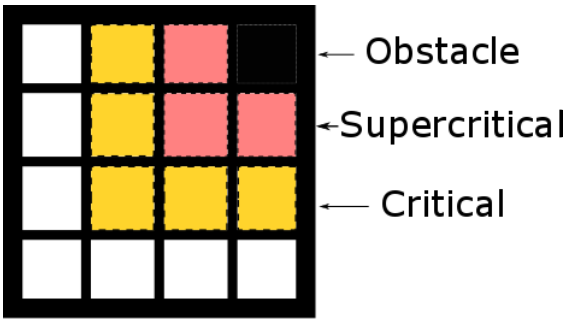


Figure 3.2: Safety definitions (adapted from [12])

Garcia and Fernandes introduce the PI-SRL (*Policy Improvement through Safe Reinforcement Learning*) algorithm which aims to 'improve baseline policies in dangerous domains using RL' [6]. A predefined baseline policy is available and assumed to be suboptimal. Two steps are then taken. The first approximates baseline behaviour using behavioural cloning techniques. This is similar to the famous Stanford RL helicopter [1] project by Abbeel et al where a baseline behaviour is approximated by learning using *Learning from Demonstration (LfD)* techniques. Abbeel calls this 'Apprenticeship Learning' and this is discussed in detail in Section 3.2. In the second step, the algorithm carries out policy improvement by safely exploring the state-action space. It does this by randomly introducing small amounts of Gaussian noise to the greedy actions of the baseline policy.

In order to determine what is safe, four regions are defined, **'Known States'**, **Unknown States**, **Non-Error States** and **Error States**. Fig. 3.3 is reproduced from their paper and graphically represents the relationship between the definitions. The algorithm first

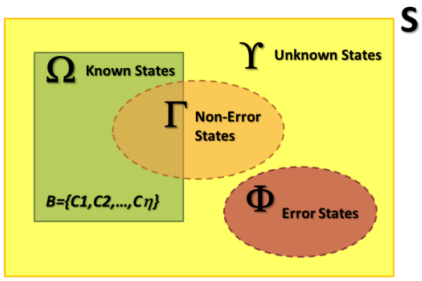


Figure 3.3: Garcia definitions [6]

learns the green 'known-space' from the baseline policy, which is assumed to be safe and suboptimal. The yellow (unknown) and green spaces are then adjusted in order to explore, while avoiding the red (error) area. The exploration is carried out by 'perturbing the state-action trajectories with the addition of Gaussian random noise'. A risk-parameter σ is defined which allows the user to set how much noise (and therefore risk) he/she is

willing to add to the exploration process. This approach applies under the assumptions that **Nearby states have similar optimal actions** and that **Similar actions in similar states tend to produce similar effects**. This may be restrictive in terms of the applicability of this approach to model a nondeterministic physical system, which depends on what is considered to be 'nearby'.

Most other definitions of 'safety' in literature in terms of labels follow from the ones discussed above but the crux of the matter is that there are safe states, fatal states and critical states, the latter being ones which have the potential to lead to a fatal situation.

3.1.2 Ergodicity

Moldovan and Abbeel[20] present an algorithm that 'guarantees safe, but potentially suboptimal exploration' by formulating safety through ergodicity. They define an **ergodic** MDP as one where an agent can ([ref: Fig.3.1] with probability of at-least a certain δ), by following a suitable policy, reach any state, starting from any other state. The physical interpretation here is that any 'mistake' can be reversed. Safe policies are defined as those that 'preserve ergodicity with some well controlled probability'. However, there are drawbacks, as by their own admission, 'ergodicity rarely holds in interesting practical examples and consequently, these methods are rarely applicable for physical systems'.

Ergodicity is not an assumption that will be adopted in this thesis due to its high imposed level of constraint on the agent. A UAV could find itself in a situation where, due to a dynamic environmental change or a system fault, it cannot return back to its original state. This violates ergodicity but does not automatically mean that the UAV is in an unsafe state and therefore perhaps should not be labelled as such.

3.1.3 Costs

Heger[13] presents a discussion on why using policies derived from minimising the expected total discounted cost is not always reliable in his paper on considering risk in RL. Here, a cost is assigned for taking an action/being in a state and the worst-case cost of the generated policies is minimised. Pecka[22] brings up the valid point that using the cost method 'leads only to the *safest possible* policies, which are not necessarily *safe*'. Therefore, any formulation involving cost assignation must be carefully set up.

3.1.4 Variance of expected return

This is somewhat of an extension to the cost minimisation method of ensuring safety, as it recommends minimising both cost and its variance. A safe policy is one that minimizes the number of critical actions, since 'fatal transitions are expected to yield much larger costs than safe transitions, increasing the variance significantly' [22]. This is also expected to be a restrictive approach as there could be situation where the change in returned cost from episode to episode is high, without necessarily putting the agent in a fatal situation.

3.2 Safe RL approaches

A comprehensive survey on safe reinforcement learning techniques has been recently carried out by Garcia and Fernandes[7], where they define 'Safe RL' as 'the process of learning policies that maximise the expectation of the return in problems where it is important to ensure **reasonable system performance** and/or **respect safety constraints** during the learning and deployment process'. They then go on to segment Safe RL algorithms into two branches, ones that influence the *Optimisation criterion* and ones that modify the *Exploration process*. It is noted here that, despite these two branches, a modification to the optimisation criteria will invariably influence the exploration process. However, the first branch considers those methods that include some form of risk in the optimisation criterion while the second branch considers those methods where there is a fixed optimisation criterion, but the exploration process is modified to include risk. Fig. 3.4 is a reproduced version of the overview tree presented in their paper and is used in this section to detail a few of the methods and their appropriateness in relation to this thesis.

3.2.1 Optimisation Criterion

Finding a function that guides which actions to take in which states while optimising a defined criterion is the basis for most RL problems. This function is then also known as an optimal control policy. The criteria are generally based on a cost metric, such as 'minimise time' or 'maximise rewards' and are denoted by multiple terms in literature, such as 'expected return, expected sum of rewards, cumulative reward, cumulative discounted reward or return' [7]. The term 'return' is used throughout this discussion and refers to the (possibly discounted) sum of individual rewards obtained for evaluating an action in a particular state, over the whole process time. Four subcategories are defined in Fig. 3.4 and the most relevant ones are detailed in the sections below.

Worst-Case Criterion

Here, the objective is to *'compute a control policy that maximises the expectation of the return with respect to the worst case scenario incurred in the learning process'* [7]. Maximising the worst case return can be formally defined with the 'minimax' equation in Eq. 3.1, where r_t is the immediate reward for action selection, $\gamma \in [0, 1]$ and is used to discount future rewards, Ω^π is a set of state-action combinations that occur while following policy π and $E_{\pi,w}(.)$ is the expectation with respect to policy π and trajectory w.

$$\max_{\pi \in \Pi} \min_{w \in \Omega^\pi} E_{\pi,w}(R) = \max_{\pi \in \Pi} \min_{w \in \Omega^\pi} E_{\pi,w}(\sum_{t=0}^{\infty} \gamma^t r_t) \tag{3.1}$$

The motivation for using this approach is to mitigate variability due to system stochasticity and/or parameter uncertainty. This is an attempt at decreasing risk and selection of actions that lead to undesirable states.

However, update methods developed based on minimax, such as $\hat{Q}$-learning [13] display

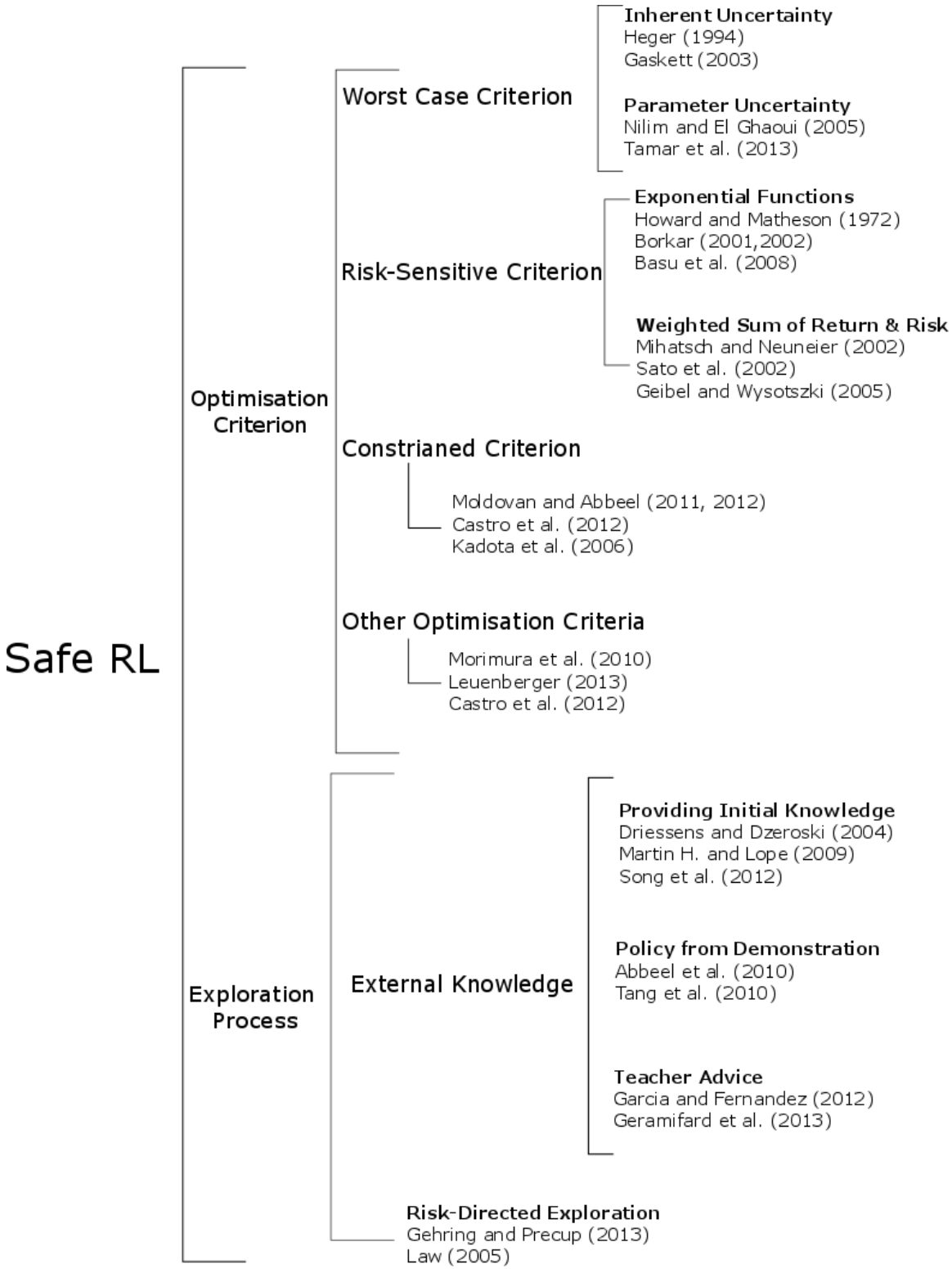


Figure 3.4: Safety summary (adapted from [7])

some behaviour which make them suboptimal for the purposes of this thesis. The approach assumes that actions with the highest known worst-values will be expected after performing them once, i.e., that the worst possible state transitions will occur. This is inherently pessimistic and leads to highly conservative policies. Gaskett[8] analyses this problem in a cliff-world setting and concludes that the minimum operator is to blame here and that 'since the action-values can only move downward in value, they must be initialised optimistically'. A similar concern is voiced by Neuneier and Mihatsch[21] where they claim that this approach is 'too restrictive in real world applications because it takes very rare events fully into account, leading to policies with a low average performance'.

There have been attempts at improving the minimax criterion, such as β-pessimistic Q-learning [8] where $\beta \in [0, 1]$ and effectively switches the equation between standard Q-learning (optimism) and pure minimax (pessimism). In a cliff-walking setting and with a properly tuned β, this does indeed show better behaviour.

Risk-Sensitive Criterion

Any approach that attempts to include a parameter that allows the '*sensitivity* to the risk to be controlled' [7] is classified under this section. In most literature, this parameter is denoted as β, where $\beta > 0$ implies risk aversion, $\beta < 0$ implies risk-seeking behaviour and $\beta = 0$ implies risk neutrality. There are methods based on exponential functions and on the weighted sum of return and risk. In the former, instead of maximising the expected value of the return, the objective is presented as in Eq. 3.2 [14].

$$\max_{\pi \in \Pi} \beta^{-1} \log E_\pi(e^{\beta R}) = \max_{\pi \in \Pi} \beta^{-1} \log E_\pi(e^{\beta \sum_{t=0}^{\infty} \gamma^t r_t}) \tag{3.2}$$

An alternate formulation is when the objective function is expressed as the weighted sum of return and risk, as in Eq. 3.3 [7]. Here, the ω parameter can take different forms to represent risk. Markowitz[18] replaces ω with the variance of the return, while Neuneier and Mihatsch[21] use the temporal difference errors during learning as their ω. Finally, Geibel and Wysotzki[9] replace the ω with the probability with which a state sequence terminates in an error state.

$$\max_{\pi \in \Pi} (E_\pi(R) - \beta\omega) \tag{3.3}$$

This approach attempts to avoid catastrophic situations even if their probability of occurrence is very small. However, typical behaviours here include the underestimation of risk due to the ignorance of improbable but severe events. Furthermore, 'mean-variance' optimisation 'can directly lead to counterintuitive policies' (cited from [7], and inferred from a study on mean-variance optimisation in finite horizon MDPs [16]). Neuneier and Mihatsch[21] also conclude in their study that the limiting behaviours of using exponential utility functions are also seen here. Finally, these methods generally 'require error states to be visited repeatedly in order to approximate the risk function and, subsequently, to avoid dangerous situations' [7]. This has implications for an aircraft guidance and control task as visiting error states, e.g., UAV crash, is simply not an option.

Constrained Criterion

The objective here is to maximise the return while keeping other types of expected measures within some specified bounds. Formally, this could be considered a constrained MDP, defined by the tuple $< S, A, R, T, C >$, where T is the transition from state S to state S', while taking action A and receiving return R, and C is a set of constraints applied to the policy. The constraint(s) serve to effectively reduce the total policy space Π to a smaller subspace of allowable policies, Γ, resulting in the objective as shown in Eq. 3.4. The space, Γ itself may be constrained based on a minimum threshold for the expectation of the return, a maximum threshold for the variance of the return, or to enforce ergodicity.

$$\max_{\pi \in \Gamma} E_{\pi}(R) \tag{3.4}$$

However, as discussed in Section 3.1, the difficulty here lies in the tuning of the threshold value. Taking ergodicity as a constraining criterion heavily restricts the domain and is not very representative of physical processes. Setting too high a threshold value does not ensure safety to an acceptable degree, while not setting it high enough makes the selected policies too restrictive. Furthermore, these approaches 'do not prevent fatal consequences in the short term' [7].

3.2.2 Exploration Process

Methods classified here include the consideration of 'risk' within the exploration process while keeping the optimisation criterion unchanged. The motivation for doing this as opposed to classic RL exploration strategies that rely on some randomness, e.g., $\epsilon-greedy$ is that these random policies require the agent to explore and learn everything from scratch, which will almost certainly lead to catastrophic actions being taken. Furthermore, random exploration is time and resource consuming as irrelevant regions of the state-action space are explored. The bulk of the methods here lead on from the conjecture that 'it is impossible to completely avoid undesirable situations in high-risk environments without a certain amount of external knowledge' [7].

Three methods of using external knowledge are identified here:

- **Provide initial knowledge**, effectively prompting the agents' exploration and value-function approximation in the right direction.
- **Derive a policy from a training set** where the time needed for exploration and discovery is decreased.
- **Guide exploration through teacher advice**, where a 'teacher' is used to provide information when it is considered necessary.

These methods assume some prior knowledge of the system. Generally, a finite set of demonstrations are recorded from a teacher and later interpreted (for example using regression) to generate a policy that learns from these demonstrations. A partial Q-function

is built which then guides further exploration. This helps the agent spend less time with random inefficient actions. A particularly successful implementation of this approach was by Martin and de Lope[5] for their RL competition helicopter where several teachers provided the initial training set. Their algorithm restricted crossovers and mutations, meaning that only slight changes were allowed to the teachers' policies and convergence to near-optimal was achieved rapidly. This is an example of *Learning from Demonstration (LfD)*, or apprenticeship learning, as referred to in Section 3.1. These generally consist of the following steps:

- Teacher demonstrates task to be learnt and state-action trajectories of this demonstration are recorded.
- All state-action trajectories are used to generate a model of system dynamics.
- A near-optimal policy is found using any RL algorithm.
- Policy is tested by running on the real system.

The issue that comes up with LfD, however, is that performance is heavily based on the training set. If the initialisation does not provide information for all important states, the agent will end up with a suboptimal and possibly fatal policy. The resulting exploration process may result in the agent visiting states for which there is no previous information. This is unsafe and unacceptable in situations where the costs of fatal transitions are high.

A modification of this approach is to consider a separate 'teacher' entity, which can be thought of as an external agent that guides the main agents' tasks. This could be through advisory actions that the main agent carries out for that step, a complete sequence of actions, or a reward provided by the teacher and used to judge (and by extension, influence) the agents' behaviour. The teacher senses a state from the environment, just like the agent does, and, based on the algorithm, decides when it should step in and advise. The problem with applying these methods to an autonomous aircraft is the assumption that an all-knowing 'teacher' is always present to guide the agents' actions. This compromises autonomy and increases reliance on constant monitoring. While this approach may be discussed in future extensions of the thesis due to its successful applications in recent work, it is not considered as the starting point.

Chapter 4

Potential Field Methods

The Artificial Potential Field (APF) concept has traditionally been used to develop obstacle avoidance strategies for manipulators and mobile robots. The APF itself is described by Khatib[15] as a 'field of forces' where the desired goal position is an 'attractive pole' for the manipulator and obstacles are surrounded by 'repulsive surfaces'. A defined potential function typically represents the generalised shape of this field and is made up of two distinct 'attractive' and 'repulsive' definitions. The former can be thought of as an energy well in a contoured surface which, by virtue of its shape, drives the agent to the bottom (point of lowest potential). The 'contours' in this surface come about due to the presence of repulsive potential barriers placed around obstacles (or fatal regions). This chapter first addresses how these attractive and repulsive potentials are formally defined, after which the motivation for their use within safe RL are elaborated upon.

4.1 Potential Functions

An often-used example of a repulsive APF is the FIRAS function (Eq. 4.1), developed to be non-negative, continuous and differentiable with values that tend to infinity as the distance to the obstacle's surface gets smaller [15]. Here, k is a gain and ρ_0 is the maximum distance where the agent feels this repulsive potential. Outside this region, the repulsive force felt is equal to zero. $\rho(x)$ then represents the shortest distance from the agent to the obstacle.

$$U_{repulsive}(x) = \begin{cases} \frac{1}{2}k_r(\frac{1}{\rho(x)} - \frac{1}{\rho_0})^2, & \text{if } \rho(x) \leq \rho_0 \\ 0, & \text{if } \rho(x) > \rho_0 \end{cases} \tag{4.1}$$

Conversely, Eq. 4.2 details an attractive potential function, designed to increase as the agent moves away from the goal point, analogous to the increase in potential energy as an object moves away from the Earth's surface [26]. The concept here is that the agent will seek to settle at the zone of lowest potential based on a gradient descent search method.

This equation combines parabolic and conic wells and has been proposed by Andrews [2].

$$U_{attractive}(x) = \begin{cases} \frac{1}{2} k_a \rho_{goal}^2, & \text{if } \rho_{goal}(x) \leq d \\ d k_a \rho_{goal}(x), & \text{if } \rho_{goal}(x) > d \end{cases} \tag{4.2}$$

Here, d is the radius of a quadratic range, k is a function gain and $\rho_{goal}(x)$ is the Euclidean distance from the agent's current position to that of the goal.

Both the attractive and repulsive potential functions then combine to make up the global potential definition of the APF, $U(x)$, as shown in Eq. 4.3. This formulation assumes one goal (attractive) point and multiple obstacles (repulsive), hence the sum.

$$U(x) = U_{attractive} + \Sigma U_{repuslive} \tag{4.3}$$

APF methods have been developed as an on-line collision avoidance method to be used when the agent does not have a prior model of the environment or the obstacle. It can only 'sense' the obstacle as it moves closer to it. This can result in the agent getting stuck in local minima, illustrated by the cases in Fig. 4.1.

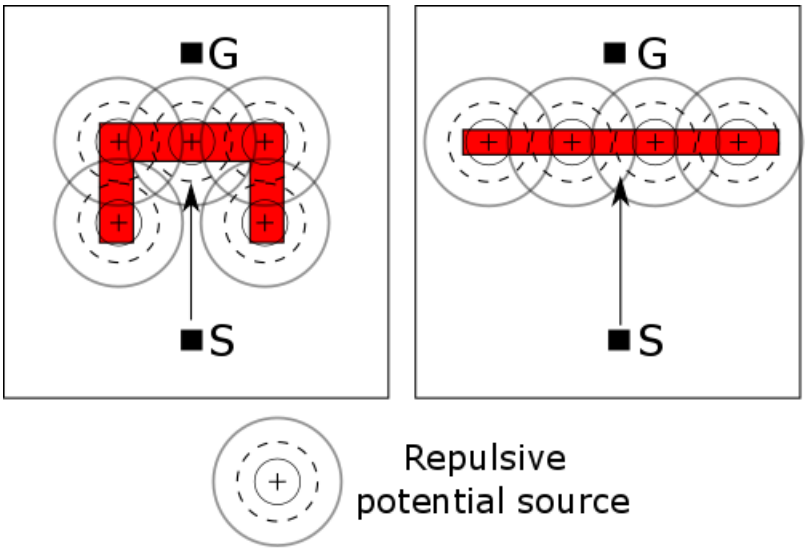


Figure 4.1: Examples of minima using APF (agent goes from start (S) to goal (G)

In order to mitigate this problem, the methods proposed in literature follow one or more of the following strategies.

- Following of the same path out of the local minimum (backtracking) and then using another strategy.
- Random movements in order to 'explore' itself out of the minimum.
- Using a procedural planner.
- Designing more complex APFs that are minima free (harmonic PF) or adaptive based on whether the minimum is local/global.

In addition, a standard problem here is Goal Non-Reachable with Obstacle Nearby (GNRON). This occurs when the goal is placed close to an obstacle and the agent/robot is repulsed by the obstacle to such an extent that it cannot overcome this force and get to the goal.

It has to be noted that these equations have been proposed with robotic manipulators and physical obstacles in mind, which is only a subset of the cases being considered in the case of a UAV, where instead of direct physical obstacles, certain areas of its state-space are being restricted in the interest of safety. That being said, the scope of this thesis does not cover a detailed design of the potential functions themselves and the ones presented are therefore being used as a starting point for discussion.

4.2 Within Safe RL

Despite the conduciveness of RL methods to autonomous exploration, they still stand to benefit from the direction provided by APFs. Combining the two would allow for safer exploration and has the potential to offer a powerful approach to model dynamical systems. Combining APFs and RL is not a very mature field of research. However, an interesting approach, called the 'virtual water flow' method, has been proposed by Xie et al [26] to solve the local minimum problem, which any RL agent subject to APF forces is expected to face.

The RL framework is first directly applied by defining a set of attractive and repulsive sources as ones where the reward received by the agent is either higher or lower than a design threshold respectively. Following on from Eq. 4.2, a potential value at the current state, $U_{att}(s)$, can be obtained by setting the function gain k to the reward obtained at the goal state and ρ_{goal}, as before, to the Euclidean distance from the current state to the goal. Similarly, the repulsive potential follows on from Eq. 4.1 and only applies within the region of influence around the obstacle, set by ρ_0.

The agent first determines whether the current state is at a local minimum by checking if its global potential value (sum of attractive and repulsive at the current state), $U(s)$ is less than $\min U(s')$, interpreted as the minimum of the potential values in the 'neighbour set'. This set consists of states (s') reachable from the current state of the agent.

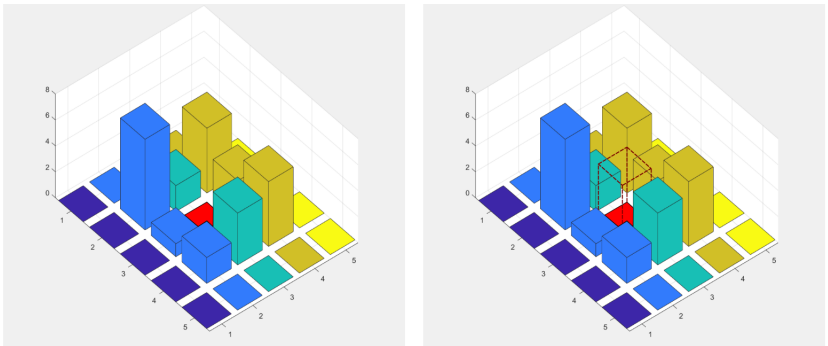


Figure 4.2: Virtual Water Flow visualised (adapted from [26])

If the agent is at a local minimum, the potential at this state is increased and regulated

by a 'speed of filling factor'. This new adjusted potential value is retained by the agent for further action selection where the agent eventually does find itself able to reach a state of lower potential, effectively releasing itself from the local minimum. This can be seen in Fig. 4.2 where the agent is currently at state (3,3), denoted by the red block. The potentials of the neighbour set are visualised as bars protruding around the agent. The potential is then increased as seen in the second half of the figure. Here, the 'neighbour set' consists of those for which the potential 'bars' have been visualised.

Xie et al[26] tested this framework within a grid-world where the agent has two goal states to be reached one after the other, in series. The grid world is set up as a 8x8 maze and therefore lends itself to multiple local minima. The reward function simply assigns a high reward (10) if the agent is at the goal state, -1 if it encounters obstacles and 0 otherwise. The agent takes about 45 trials with this problem set-up to converge to an optimum policy, meeting all its goals while avoiding the obstacles using the APF definitions and getting out of local minima using the 'water flow method'.

It must be noted here that while this method seems promising, the one constraint is that the agent must know the exact 'potential' values of its neighbouring (reachable) states to a high degree of confidence. However, with the assumption of certain sensors, this can be taken to be the case for most practical applications.

Chapter 5

Preliminary Results

The two key concepts in this work, namely **safe exploration** and **artificial potential fields**, have now been discussed in detail and put into context. In order to get a better understanding of how these ideas can be combined and whether their application is feasible, a MATLAB simulation environment is set up.

This takes the form of a simple 2D grid-world such as in Fig. 5.1 in which the agent, represented by a black cell, has the task of learning about its environment and, over time, maturing a policy that allows it to avoid certain obstacles and reach its goal cell. In this case, the so-called 'states' can be seen as the x and y positions. However, since this research is being done with an autonomous UAV in mind, it is expected that the next phase will include more states and ones that are relevant to flight in a stochastic real-life environment. Ch. 7 details the extensions made to this model in order to generate the final results. The purpose of setting up the simulation as such is to test out novel ideas of incorporating APF's within an RL framework. A discussion on the next steps and research method to be carried forward is presented in Ch. 6.

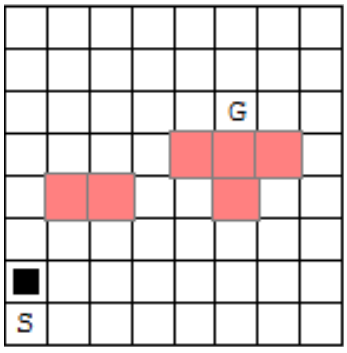


Figure 5.1: Gridworld testbed

The problem of incorporating an APF within the gridworld and then allowing the agent to interpret this in order to promote safe exploration has been divided into three main branches. This can be seen in Fig. 5.2 where the following is proposed:

- A potential field could warrant a change either in the Value/Q function and, by extension, in the way that the reward system is set up.
- The APF could affect the policy directly by stochastically influencing the likelihood of certain actions when exposed to a 'potential'.
- The dynamics of the agent could be influenced by the APF either heuristically or adaptively.

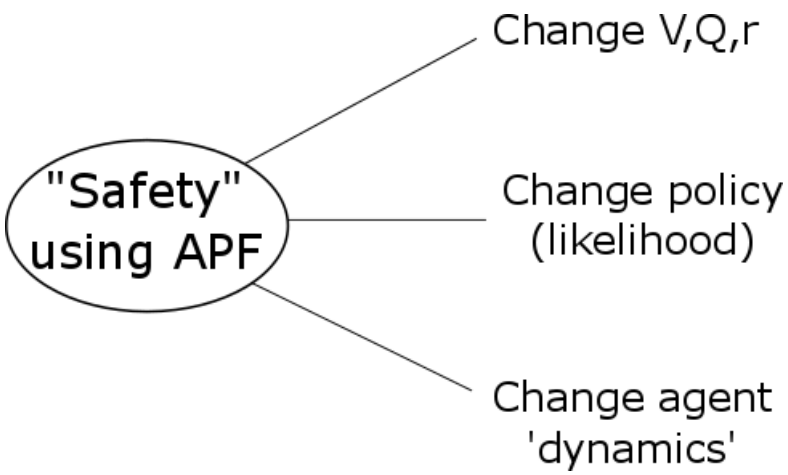


Figure 5.2: Interpreting 'Potential' in the grid

The setup of the code used to generate the simulation environment used in this thesis is elaborated upon in Appendix A in order to provide the reader with an intuitive understanding.

This chapter first details the idea of 'potential' in Section 5.1 and is then concluded in Section 5.2 with preliminary results based on a detailed analytical framework. The choices made for the rest of the thesis work are also discussed and motivated.

5.1 Developing the idea of 'potential'

As soon as any implementation of an APF is looked at, it is realised that the concept itself needs some work. A significant part of the research carried out so far defines a potential field, be it discrete or continuous, as a distance based function whose output values are then mapped to a value that influences the action taken by the agent.

There are two main questions to be tackled here; how to store the observed 'potential' information, and what to do with it. Extending the Q-learning concept of assigning what is essentially an 'estimated utility' value to each state, a **P-table** is devised to store representations of potential for each state. Fig. 5.3 shows the two ways this can be visualised.

The top branch shows the states put on a 2/3D grid where each cell would then be filled as the potential values become more available over many episodes. The other option is similar to the Q-table where all n states are listed in a 'n x 5' table where there are 5

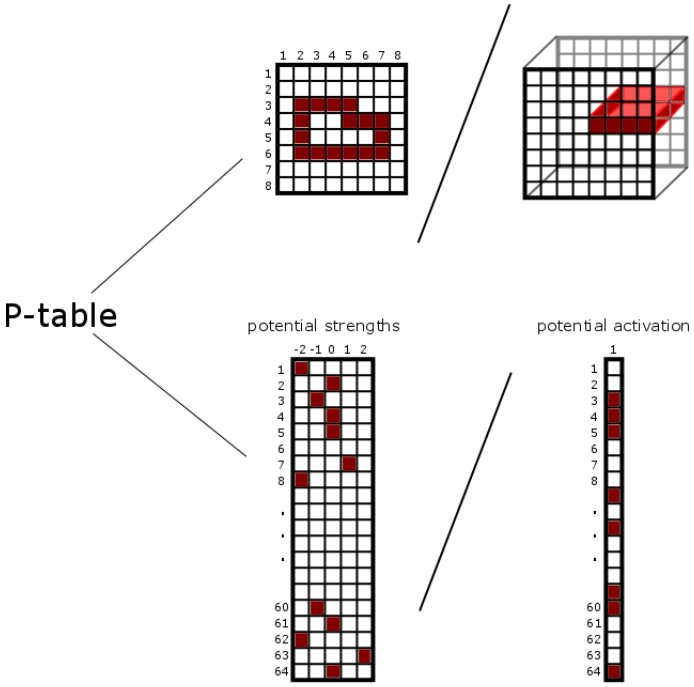


Figure 5.3: 'Potential' visualisation options

defined potential strengths [1], ranging from -2 to 2. Each state would then get an assigned strength over time. The other construct here would be to record whether there is a potential field or not as seen in the bottom right option. Potential 'strengths' can still be taken into account by filling in appropriate values in the appropriately indexed state cells.

The second approach has been adopted since it does not impose any size-based restrictions on the strength values, i.e., any number can be filled in. It is assumed that the agent has sensors which allow it to see 1 step around its current position, much like any proximity sensor (vision based or otherwise) would work on a UAV. Consecutively, the code is written such that when the agent is next to an obstacle, it records its current position as a high potential area, the position slightly further away as a slightly lower potential area, and so on. It must be clarified here that the agent can only assign potential values for the state that it is currently in and does so based on distance to objects of interest. (obstacles, walls, goal etc.) The specific potential values that are assigned based on agent proximity to any obstacle (obs), are detailed in Table 5.1. This is set arbitrarily following the only requirement that the magnitude of the potential should decrease as the distance to the obstacle increases. The agent 'perceives' the 1-step away (in all directions) potentials in a construct known as `Pnet` [2]. Over time, this fills up the potential table based on how the environment is built. `Pnet` is visualised for clarity in Fig. 5.6.

The next question is how to use this potential to influence the agent's actions. The following subsections go through some of the proposals which follow on from the branches

[1] The number '5' is chosen arbitrarily and may be adapted to fit the situation at hand.

[2] This can be found under the section *Generation of 'Potential Field' net* in the `e_greedy_selection.m` function contained within `Episode.m`. For details about these sections, refer to Appendix A.

Table 5.1: Potential values assigned as a function of position relative to obstacle

Agent relative position (to obs)	Potential assigned
Obstacle	-10
1 step	-5
2 steps	-3
>2 steps	0

in Fig. 5.2.

5.1.1 Changing V,Q,r

The first branch suggests an adaptation of the Q-learning update in Eq. 5.1 to include the generated potential information. There are two suggested options.

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max(Q(s',:)) - Q(s,a)] \tag{5.1}$$

One is within the *max* operator. Q-learning is a fairly optimistic update law because it simply takes the maximum value in its calculations. There may be other, less brutal operators that select based on measures that already process the Q-values before selection (eg. based on frequency of occurrence, distribution of values etc.).

The second option is to look at the discount factor, γ. This is used to condition the effect of the delayed reward in relation to the immediate reward. The P-table could eventually lead to a potential factor, $0 < \gamma_{pot} \leq 1$ which, when multiplied with the discount factor as already defined, would give the new conditioning factor. This suggestion is made in order to make action selection a decision that is significantly influenced by the potential information available. An example would be choosing a γ_{pot} that leads to an increase of the γ value if the potential value is large and consistently obtained over multiple runs. This means that the algorithm will give more weight to the immediate reward as opposed to the delayed ones. In a case where the potential value is not-available, or is inconsistent over multiple runs (large variance), it would give the immediate reward less 'importance' and the delayed rewards would be discounted less. Yoshida et al [27] carried out a study where they introduced a state-dependent discount factor to a conventional Q-learning setup. In order to find this discount factor, a discount function is introduced and then optimised using an evolutionary algorithm.

5.1.2 Changing Policy

This method suggests influencing the likelihood of certain actions stochastically, perhaps by the introduction of some randomisation factor, subject to their exposure to a strong potential. The idea behind this is to identify when the agent is receiving a strong impulse from the potential field (P-table) and to treat that as a special case. The final follow-on action would be randomly picked from a set of suggested actions. This also has the added advantage of avoiding getting stuck in minima.

Fig. 5.4 illustrates this method with a Q-table. Here, the agent is at state 1 and has the possibility to choose from 3 actions. Action a_3 has the highest Q-value and therefore would be chosen following standard Q-learning. However, using information from the P-table, a probability distribution could be imposed where each action now has a chance of being selected. This distribution would be designed based on how safe the potential field around the agent is and which actions are therefore more appropriate. Of course, over time, as the Q and P-tables get properly conditioned, they should both suggest the same optimum action.

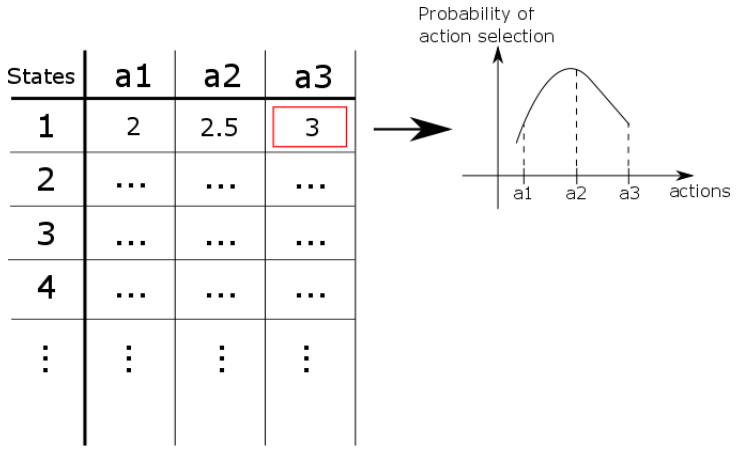


Figure 5.4: Visual explanation of the 'policy changing' approach

5.1.3 Changing Agent Dynamics

In the current iteration of the code, the agent has been given 3 so-called 'levels' of mobility, as can be seen in Fig. 5.5. This simulates various speeds of movement and can be chosen by the agent based on its immediate requirements, eg., if it finds itself within one cell of an obstacle, it can choose to apply a strong (Level 3) impulse in the opposite direction. A disclaimer here is that this method does not take any inertia effects into account. However, this has been included in the final iteration, used to generate the final results in Part III of this thesis and is explained in detail in those chapters.

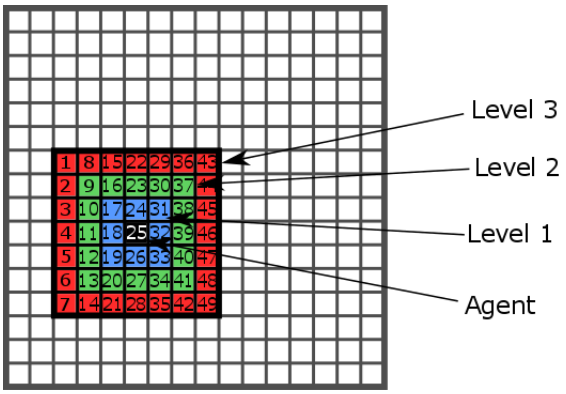


Figure 5.5: Agent action levels

It must be noted that the agent can only see 1-step around, but can take actions up to 3-steps around its current position. This is a design choice and may not be very appropriate as it could lead to situations such as in Fig. 5.6 where the agent, which is currently at cell (5,2) detects an obstacle and following its potential field reactionary dynamics, is told to take an immediate Level 3 action. This would result in it moving (in the next step) to cell (2,5). However, there is an obstacle spanning cells (3,4) to (3,5).

This is avoided by introducing further inner evaluation loops, i.e., if the agent decides to, based on the potential information, take a Level 3 action, the agent's dynamics restrict it to make 2 action–re-evaluation steps. The agent takes a single step action in the general direction of the initial Level 3 action and re-evaluates its immediate potential field, which gives it new information. The inner evaluation loops are identical to the outer except for the fact that they do not re-select an action for the agent, unless, of course, there's an obstacle in the way. In that case, the inner loop is broken and the algorithm considers the current state as the starting point for re-evaluation. The inner loop actions also do not update the Q-table. This is only updated when the action set is completed or the inner loop is broken.

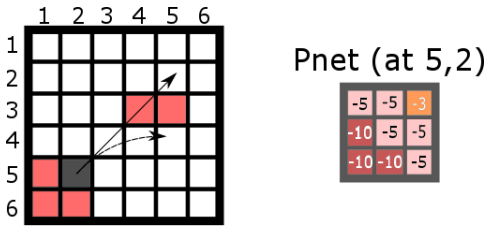


Figure 5.6: Example of sup-optimal situations with the current method

This brings up the question of how the agent decides where precisely to go. So far, all the agent does is act on Potential strengths to determine which level of action it should take. The specific direction is determined using a vector based system, which is best explained with an example. Referring back to Fig. 5.6, the `Pnet` perceived by the agent when it is at cell (5,2) is as shown to the right of the grid. The values follow from Table 5.1. It must be noted here that a significant part of `Pnet` is influenced by the potential fields of both obstacle clusters. However, for ease of explanation, the `Pnet` shown in the example is only due to the closest obstacle cluster, i.e., bottom left. There is a more nuanced method used to deal with interfering potential fields, and this is elaborated upon in Section 5.2.2.

The next step is to generate vectors indexed from the agent's current position. Considering the agent in the middle of `Pnet`, it can move -1, 0 or 1 steps and the cartesian product of these values is taken as a reference table. The respective vectors are multiplied with their values from `Pnet` and then added together to result in one 'suggested vector' called `P_act`. Using the sign convention as shown in Fig. 5.7, this results in $[-12, 12]$ being the suggested action vector as shown below:

$$\begin{aligned}[-5 * (-1,-1)] + [-10 * (0,-1)] + [-10 * (1,-1)] + \\ [-5 * (-1,0)] + [-5 * (0,0)] + [-10 * (1,0)] + \\ [-3 * (-1,1)] + [-5 * (0,1)] + [-5 * (1,1)] = \textbf{[- 12,12]}\end{aligned}$$

This tells the agent to go top-right, which makes sense since this takes it away from the immediate obstacles, which were on the bottom-left, relative to the agent. As can be seen in Fig. 5.7, the angle of `P_act` is also taken into consideration. In this discrete case, the inclusion of the angle does not offer much of an advantage as the directions could have been hard-coded by cell, but this now allows a smoother transition when the method is made more continuous. At the moment, the agent is assigned a sector based on its α and the end cell is chosen accordingly.

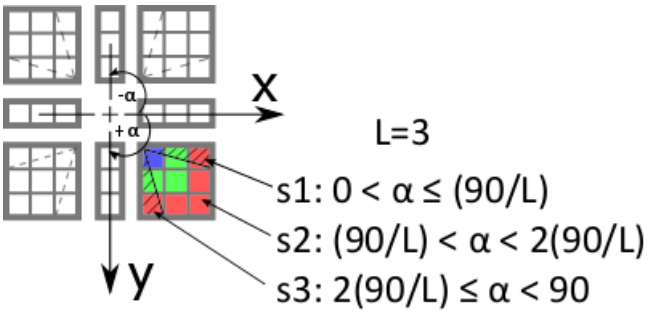


Figure 5.7: Sign convention, including 3 sectors and 3 levels

It must be noted that the agent is not provided with any information about its environment in the beginning and must build up its P-table over multiple runs from scratch. The reader may recall that the P-table provided to the agent was initialised with all zeros. This, of course, implies that actions based on P-values will not be optimal or even correct in the beginning. Therefore, standard Q-learning is implemented as in Eq. 5.1 and the ϵ-greedy scheme has been adapted to include potential based actions. This is done by programming the agent to carry out actions based on the Q-table (effectively a measure of the 'quality' of each action in each state) if the magnitude of `P_act` is a certain value, κ, implying potential information that is not strong enough to follow (refer to the discussion on κ in Section 5.2.1). This avoids the agent getting stuck in local minima and helps for initialisation. Furthermore, the Q-table is also minimally perturbed with a small random value for each action in order to move away from the deterministic nature of the problem, and also to ensure that the algorithm can find a maximum Q-value in each state.

5.2 Analysis

Section 5.1 broadly presented some approaches to incorporating potential information within a standard RL setup. However, in order to make an informed decision on the most effective research path to follow, this section will present simulation results in a bit more detail, culminating in a recommendation for the direction of the final thesis work, the results of which are presented in Part III.

5.2.1 Framework

The agent can perceive potential information on varying levels of autonomy. This can range from taking an action that directly follows on from an obstacle or associated potential field, to letting the agent use the potential field to only influence its reward function.

In the latter case, the agent is expected to, over time, condition its Q-table such that safe 'potential-influenced' actions are taken without being explicitly told to do so.

Four levels of analysis have been identified and can be seen, represented from high to low level, in Fig. 5.8 [3]. The reader may, at this point, draw parallels between these levels and the three branches of Fig. 5.2. 'Level 1' here essentially corresponds to *Changing agent 'dynamics'* while 'Level 4' encompasses, among others, a purely reward-based approach.

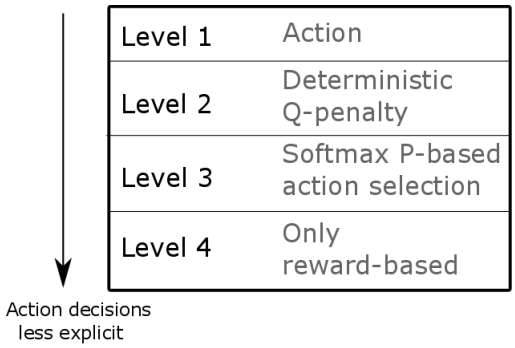


Figure 5.8: Levels of Analysis

1. **Level 1:** Here, the agent is directly given an action in the potential zone, effectively overwriting the Q-table and anything learnt there. This is done using the methods described in Section 5.1.3. One parameter that influences how sensitive the agent is to the potential information is κ. A very high κ implies that the agent will ignore the potential-based action and stick to standard Watkins' Q-learning (essentially Level 4), while a κ of zero means that the agent will always follow the action suggested by the direction of the potential field. Neither leads to an optimal policy and therefore κ must be carefully tuned.

2. **Level 2:** Here, instead of directly affecting the actions in the potential zone, the agent deterministically shapes the Q-values of the appropriate state-action pair (perhaps even ones in close proximity to it, eg. similar actions or states around the current one). The Q-shaping could be stochastic, or simply deterministic, such as in Eq. 5.2 where Q_f is the final updated Q and Q_p is the potential zone's contribution to the Q-value, and q is a tune-able gain.

$$Q_f = Q + qQ_p \tag{5.2}$$

   The original implementation included this within the Q-learning, effectively modifying the Q-**update** as in Eq. 5.3, where α is the learning rate and T_{err} is the temporal difference error. However, what this does is essentially introduce another 'reward', making it a Level 4 approach. Furthermore, since this is included in the update itself, every step of every episode that the agent is in a potential zone introduces this factor, meaning that the penalty on the agent is compounded each

[3] All these methods are implemented within an 'ϵ-greedy' shell, implying that some exploration always takes place, based on the decaying ϵ value.

time.

$$Q_f = Q + \alpha T_{err} + qQ_p \tag{5.3}$$

3. **Level 3:** Similar to Level 2, but using a non-deterministic operator such as 'softmax' for action selection. The motivation for using 'softmax' is that it shows positive behaviour in 'settings where it is necessary to maximize utility but also to hedge against problems that arise from putting all of ones weight behind a single maximum utility decision' [3]. Adaptations of the softmax operator for reinforcement learning have been explored with positive results by Asadi et al. where their motivation for proposing a slightly adapted version called 'mellowmax' was to present an operator that is 'both *non-expansion* (ensuring convergent behaviour in learning and planning) and *differentiable* (making it possible to improve decisions via gradient-descent methods)' [3].

   Furthermore, this is interesting to explore since the implementation of the potential information within the action-selection procedure is non-trivial. Fig. 5.9 shows the standard softmax equation and a few suggested modifications to it.

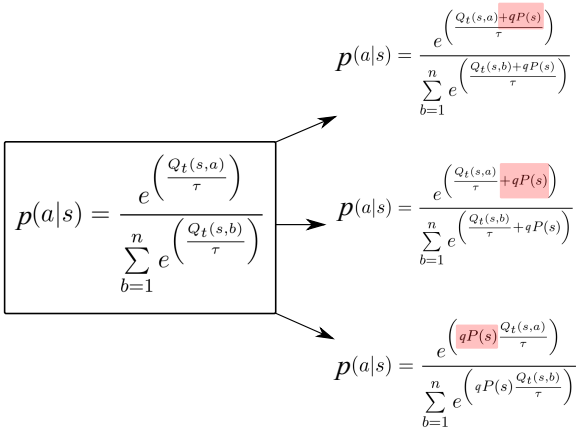


Figure 5.9: Modification of Softmax to include Potential

Convergence to a locally optimal policy has been observed with the first two modifications. However, there are two points to note here. Firstly, softmax includes a 'temperature' parameter, τ which is not intuitive to tune. Starting from the fact that as τ tends to infinity, all the actions approach equi-probability and as τ tends to zero, there is a greater difference in selection probability for actions that differ in Q-values, a value of $\tau = 50$ has been set. In addition to this, these modifications add another tune-able parameter, q, which is intended to condition the potential strength values (Similar to the factor 'k' in Level 2). Tuning these parameters requires a more thorough analysis of the system. At the moment, no comparative study has been found on how to set these parameters (at-least on τ).

Secondly, as can be seen from Fig. 5.9, the 'softmax' equation returns a normalised probability of action selection given that the agent is at a specific state. In the original form, this is intuitive since the probability depends on a Q-value that itself is dependent on a specific state-action pair. However, the potential information

available to the agent is **only state-based**. By including this information in the softmax equation, the implicit assumption is being made that regardless of the action that is being taken at a given state, the effect of the potential is the same. This is perhaps not the most effective use of the information available. Two suggestions to deal with this are discussed here:

(a) The agent also has access to a construct called 'Pnet' (used up until this point, only for Level 1) which allows it to observe the potential values in a grid extending one step in every direction from its current state. Using this, one option would be to immediately restrict the accessible action-base to those that correspond to potential values over a certain threshold value (this cutoff point can be pre-determined) and then continue with softmax/epsilon-greedy action selection. Using the same Pnet as introduced previously, Fig. 5.10 shows how this would work. Here, the cutoff point for no-go actions is a potential value of -10 or lower. Hence, the only available actions are as shown.

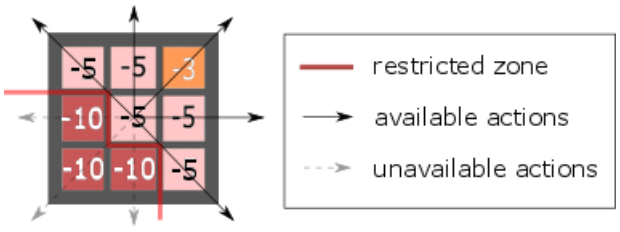


Figure 5.10: Pnet based action restriction option schematised

However, there are 2 problems with this. First, it is a brutal approach and perhaps does not belong in this level of autonomy. It is akin to pre-allocating specific potential-based actions for the agent, similar to Level 1. Secondly, this could lead the agent into getting stuck in local minima as, given a choice between staying in a zone where there is no potential vs going into a negative potential-rich zone, it would choose the former, regardless of the fact that the goal may require it to pass through the potential-rich zone.

Another option is to check which 'sector' the actions fall into and then to index these sections with Pnet. If that sector in Pnet has a low potential, the probability of selecting that action should be restricted. This is more in line with the level of autonomy desired in Level 3. However, more analysis must be made on what the best way is, to influence this probability. One option is to use the previously introduced parameter, q. The agent, while assigning probabilities to the action-space, first checks which sector (top-left, up, top-right etc.) the action falls in. This sector is then indexed with Pnet. If the corresponding potential value for that sector in Pnet is undesirable, a higher k could be set and vice versa. This allows for the influence of the action to also be taken into account.

(b) The other option is to completely rethink what 'potential' means to the agent and to treat it much like the Q-value, where there would be a different 'potential value' for each state-action pair. The risk here is running into problems of

dimensionality as now, two rather large data-sets will need to be stored and interpreted.

4. **Level 4:** This is the most 'hands-off' approach since only the rewards are influenced by the potential information. The agent is not actively barred from going into a potential zone. Rather, it is expected that the Q-update should eventually realise that the potential zones result in higher negative rewards and should automatically suggest actions that lead the agent away. This is done by setting $\kappa > 30$, as discussed in 'Level 1' [4].

5.2.2 Results and Conclusions

This subsection presents the results of simulation runs carried out following the four levels as described in Section 5.2.1. In order to avoid consistent exploration even after a good policy had been found while still ensuring that the agent does not exploit its established policies excessively in the beginning, ϵ has been designed as the monotonically decreasing function in Eq. 5.4. Here, ϵ is set as a function of the current episode number n_{ep} and the coefficients, a and b are set empirically and respectively as 0.6 and 1.0015.

$$\varepsilon = a(b^{-n_{ep}}) + c \tag{5.4}$$

Despite this, it could be that the final policy is still sometimes found to be suboptimal. The epsilon rapidly decreases, so the agent learns a policy and then over time, does not have enough 'residual' left in the epsilon equation to look for a more optimal one. This could be solved by temporarily increasing the epsilon after it has already decayed to its minimum once, in order to promote finding a better policy through exploration. This is then again allowed to decay to the residual. The Q-table after the first decay is then compared to the table after the second decay in order to see if the agent does indeed converge on a more optimal policy. This can be seen schematically in Fig. 5.11.

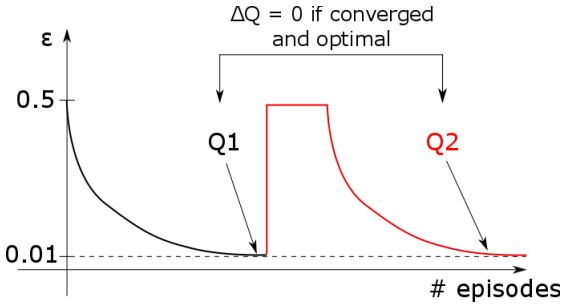


Figure 5.11: Exploration strategy

The simulation world is set up as in Fig. 5.12 where the agent is, for the first 500 episodes, initialised in a random location at the bottom left of the grid, after which the start position is fixed.

[4] **30** is the highest possible magnitude of the vector '`P_act`', based on the potential field values used throughout this paper

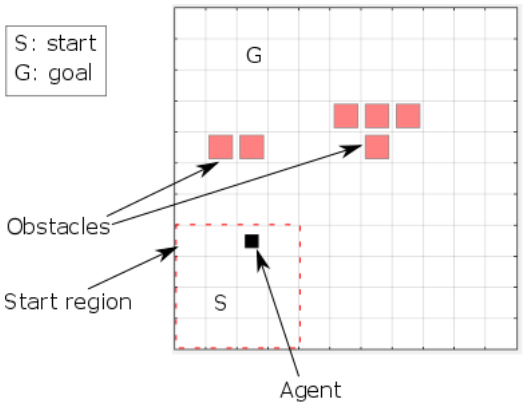


Figure 5.12: Simulation grid

The goal is placed such that the agent will have to interact with the potential field generated by the obstacles. This is shown in Fig. 5.14 where the raised surfaces are obstacles and the 'depressed' cell represents the potential 'pull' of the goal. In case of potential fields interfering due to obstacles being close to each other, the potential value for those cells are determined by adding the original potential values (due to their respective obstacles) and then multiplying it with a merge-factor, $(0 < m < 1)$, 0.75 in this case[5]. The only requirement while setting this value is to ensure that the inter-obstacle space should not be able to have a higher negative potential than an obstacle itself. This is done to prevent the agent from coming across any situations where it determines that the best option is to go through an obstacle. This has been visualised in Fig. 5.13 where it can be seen that the potential values in the 'interfering' states do not exceed the -10 value assigned to the obstacles. Eq. 5.5 presents a generalised formula for potential strength assignment for each state, where P_s is the potential value at a given state, $p_{s,i}$ is the potential due to obstacle cluster i and m is the aforementioned merge-factor. The total number of obstacle clusters in the world is denoted by N.

$$P_s = m \sum_{i=1}^{N} p_{s,i} \tag{5.5}$$

A summary of the reward assignments in this simulation can be seen in Table 5.2 where the rewards are based both on the state that the agent is in ('position') and the action that it chooses to take ('action'). The latter could be a single action or a Level 2/3 action ('within the loop').

Fig. 5.16 shows the simulation results for Level 1 and Level 4. The first column shows the reward value return per episode over the whole run. While simulations 2 and 3 seem

[5]The value for m has to be optimised based on the potential strength magnitudes used. If a sequence of doubling $(-10, -5, -2.5, -1.25)$ is followed while setting the potential strength values, then, regardless of what the merge factor used is, there will be no issues with overlapping potential strengths being higher than the obstacles themselves. In this simulation, the sequence $(-10, -5, -3)$ has been used arbitrarily and hence 0.75 has been determined to be an appropriate value for demonstration. Future work will be adapted to stick to the doubling sequence

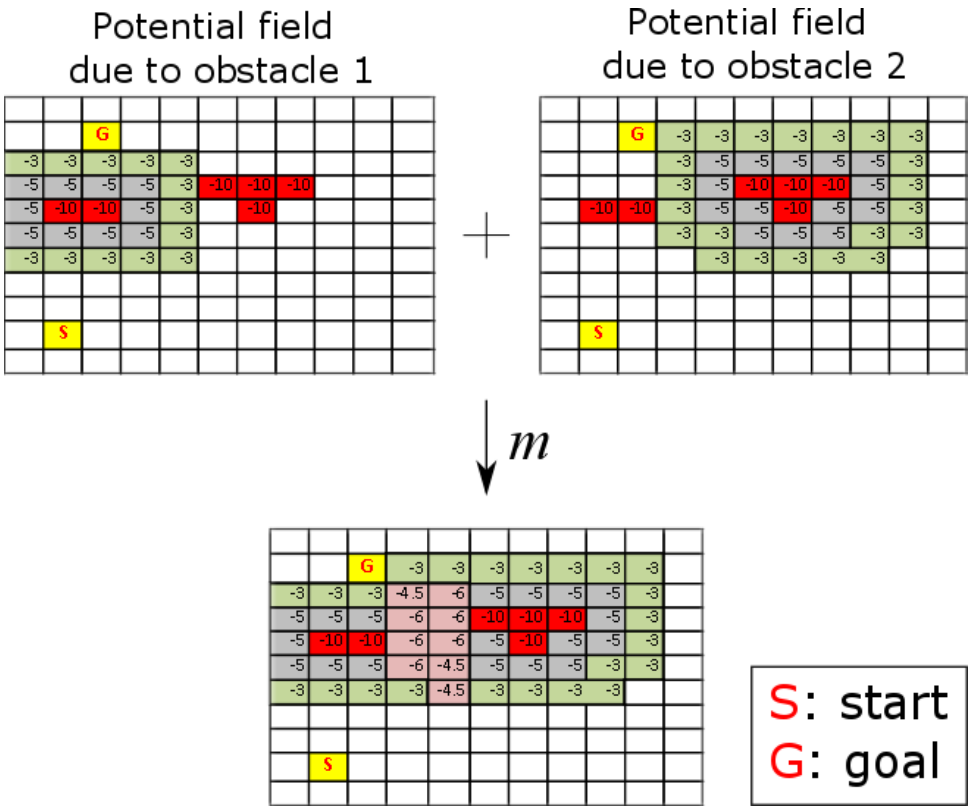


Figure 5.13: Potential strength for interfering fields

to increase the return per episode over the entire run, simulation 1 barely shows any improvement. Column 2 shows the most visited states, which can be interpreted as a visualisation of the final policy of the agent. Here, the goal state is marked with a black circle. The third column shows, over the whole run, whether the agent has reached the goal. A mark is placed on 'one' if, in any given episode, the agent reaches the goal, and 'zero' otherwise. It can be seen that for the second and third simulations, the agent eventually adopts a policy where it reaches the goal every episode. This is not the case for the first simulation. Finally, the fourth column uses the first distance norm, as shown in Eq. 7.2, on consecutive Q-tables, to show convergence. Here, **N** and **M** refer to the dimensions of the grid, where NM would be the number of states.

$$d_1(Q_k, Q_{k+1}) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} |Q_k(i,j) - Q_{k+1}(i,j)|}{NM} \tag{5.6}$$

Two settings for Level 1 are included to highlight the effect of the κ variable. In the case where this is set to zero, the agent only follows the suggested action by the potential field. As can be seen, this leads to it exploring aimlessly and getting stuck since the goal is beyond a sizeable potential heavy zone. Every time the agent approaches this zone with the aim of getting to the other side, the suggested potential-based action is to go back. When κ is set to larger than 30 (the maximum possible 'potential' magnitude), the potential information is totally ignored and the agent carries out standard Q-learning as

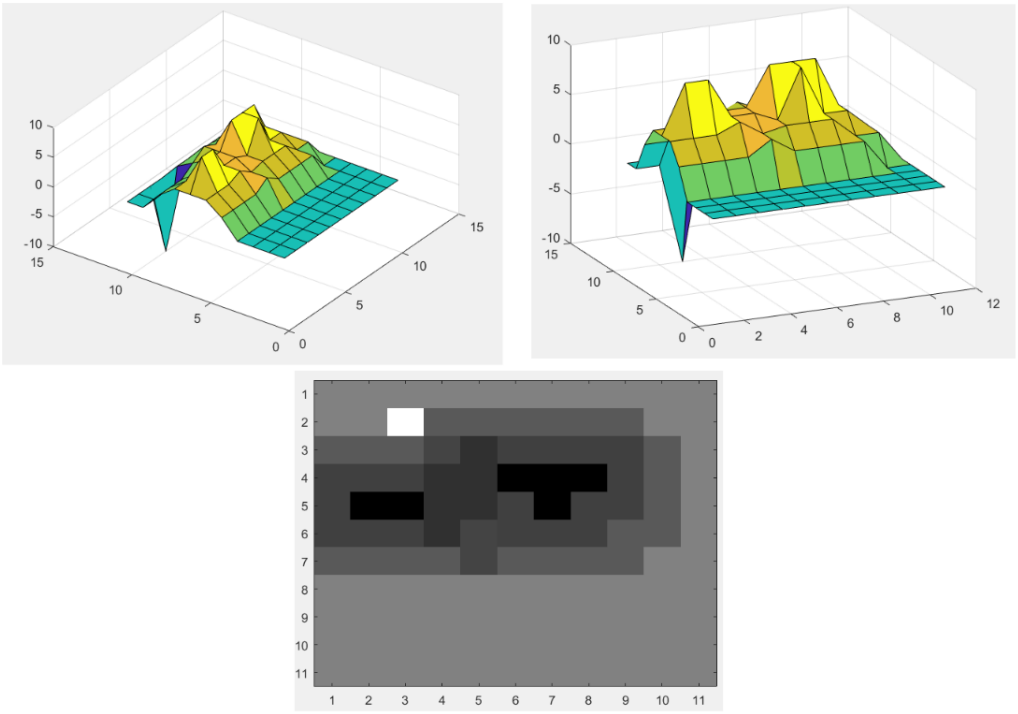


Figure 5.14: Potential Field render

can be seen in the third set of results (Level 4). Setting $\kappa = 15$ aims to strike a balance where the potential information is used if it is 'strong enough'. Otherwise, the agent continues with standard Q-learning.

These two levels will have to be included in every comparative study as a baseline. However, the decision to be made here is whether to investigate Level 2 or Level 3 in more detail. Fig. 5.17 shows the simulation results for these levels. Level 2 involves finding a way to shape the Q-value, as discussed previously. The problem here is one of compatibility. The potential information cannot be simply added to the Q-value, as the latter includes a different type of information, based on both state and action. Furthermore, as it stands, the agent is penalised once over each episode based on whether it finds itself in a potential-rich zone. This may not be very effective and would certainly require setup-based tuning in order to ensure that these effects are still felt by the agent, regardless of the maximum number of steps per episode. Coming up with a generalised algorithm for this approach will require more research than is perhaps appropriate for this thesis.

Two Level 3 results have been included here; 3*a* refers to the first softmax modification as shown in Fig. 5.9, while 3*b* refers to the second modification. In order to mitigate the problems with this 'Level' as discussed in Section 5.2.1, it is assumed that the agent has access to the `Pnet` construct. This is a valid assumption as the information to generate `Pnet` would be available to the UAV anyway if it has the one-step-ahead sensors on board and also the P-table in its memory. The suggested action due to this is then taken into consideration in the following way, visualised in Fig. 5.15. The modified softmax suggests an action, say 'A', while `Pnet` suggests action 'B'. Each action is associated with a probability according to softmax, with action 'A' having the highest. However, instead

Table 5.2: Reward summary

Category	Detail	Reward
Position	At goal	500
	At obs-1	-50
	Being at -10 $\leq$ P $<$ -5	-20
	Being at -5 $\leq$ P $<$ -3	-10
Actions	Into obstacle	-50
	Into wall	-50
	Single step	-10
Within loop	(3 steps)	-10-10γ-10γ^2
	(2 steps)	-10-10γ

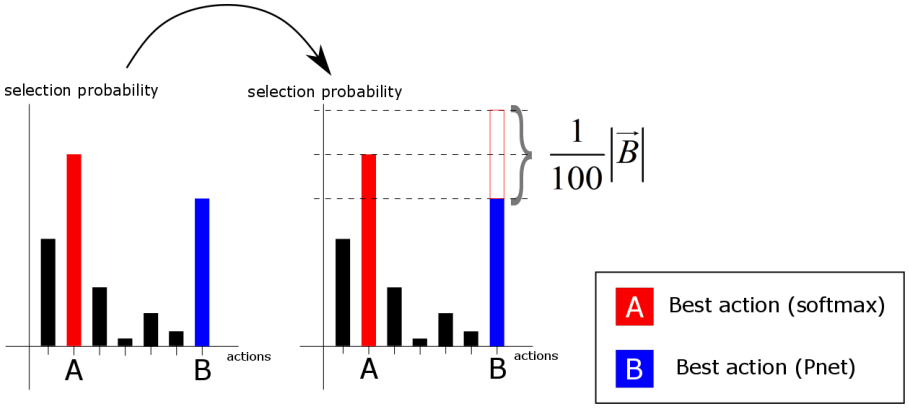


Figure 5.15: Level 3 stochastic action selection

of simply choosing this action, the potential information is further utilised by increasing the probability of, in this case, action 'B' by a factor[6] of the magnitude of the action vector suggested by `Pnet`, namely |`P_act`| (introduced previously in Section 5.1.3). After normalising this new probability set for all the actions to ensure that they add up to 1, it is then used to pick a corresponding action. All of this is, of course, carried out within an epsilon greedy shell where a random action is taken with a probability ϵ.

Based on the above discussion, Level 3 seems like the most promising option to explore in more detail. Level 3 includes tune-able factors such as the 'temperature', τ for softmax and q, which can be interpreted as the 'influence' scaling for the potential information. Furthermore, the modification of the softmax equation requires more analysis and research but seems promising so far in terms of convergence and obstacle avoidance. This is as opposed to Level 2 which has fundamental issues in terms of obtaining an insight into how Q-values are actually affected. This makes it inappropriate for the final thesis work. Another key advantage of using a Level 3 approach that has been demonstrated is in terms of safety. This is shown in Fig. 5.18 as the number of obstacle collisions per episode, over the whole run. It must be noted here that 'collisions' in this case refers to the 'intent of colliding' on behalf of the agent. These are situations where the agent has determined

[6]This 'factor' is 1/100 and is chosen based on the probability magnitudes and the range of values generated by `P_act`. This ensures that the probability increase is not excessive and stays between 0 and 1.

that the best action to take from its current state would be to go through an obstacle. Now, due to the aforementioned 'inner-loops' in the problem setup and the lack of inertia effects, the agent always stops itself before actually hitting an obstacle. However, the selection of an action that takes it through one, is recorded and plotted. It can be seen here that the Level 3 plots show a marked decrease in collisions, both in an absolute sense (scale shows smaller numbers than for Level 1 plots) and also as the run progresses and the agent's behaviour matures.

Following on from the findings so far, Ch.6 concludes the preliminary thesis part and details the research method applied in the main thesis section.

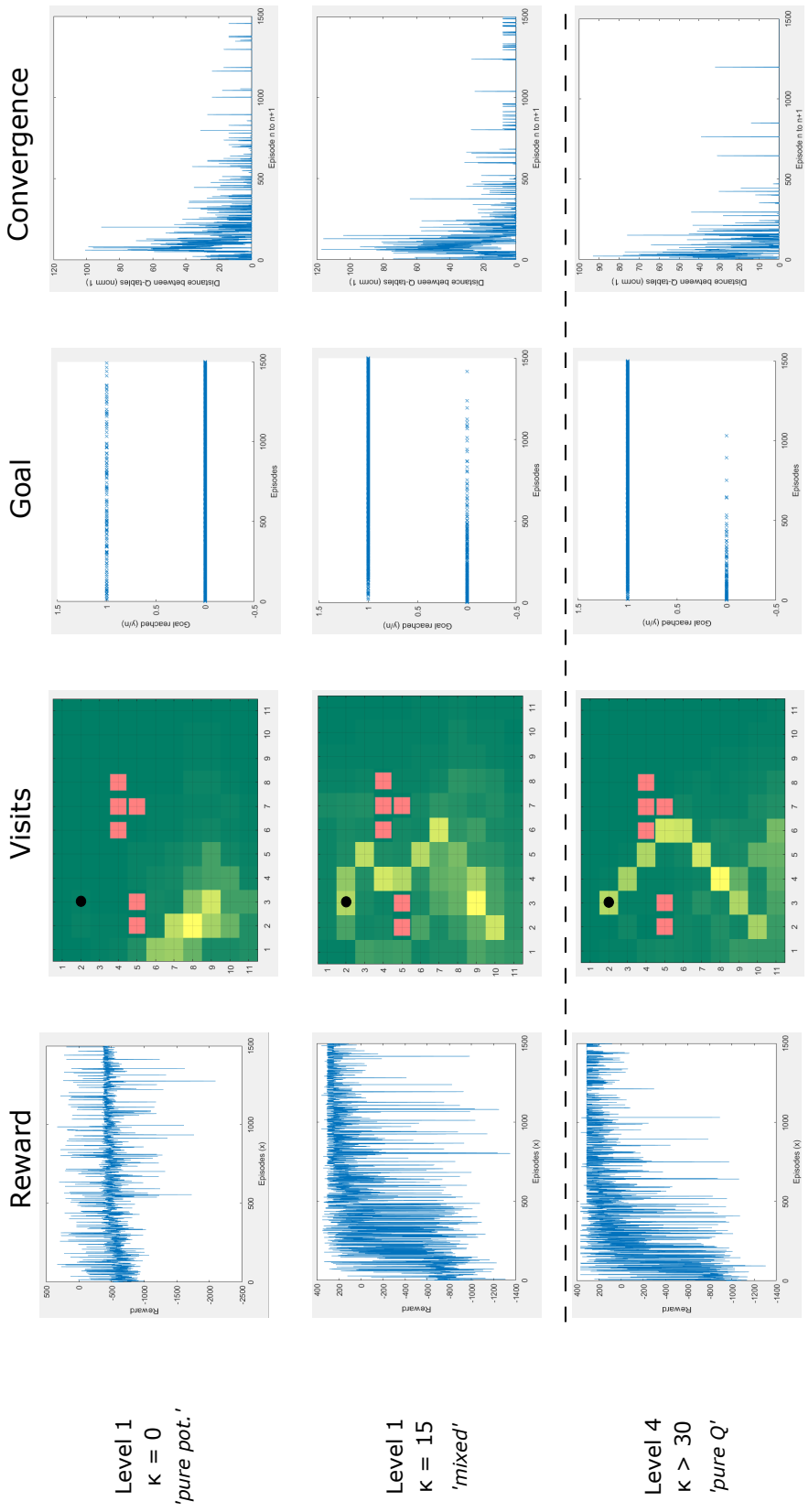


Figure 5.16: Results for Levels 1 and 4

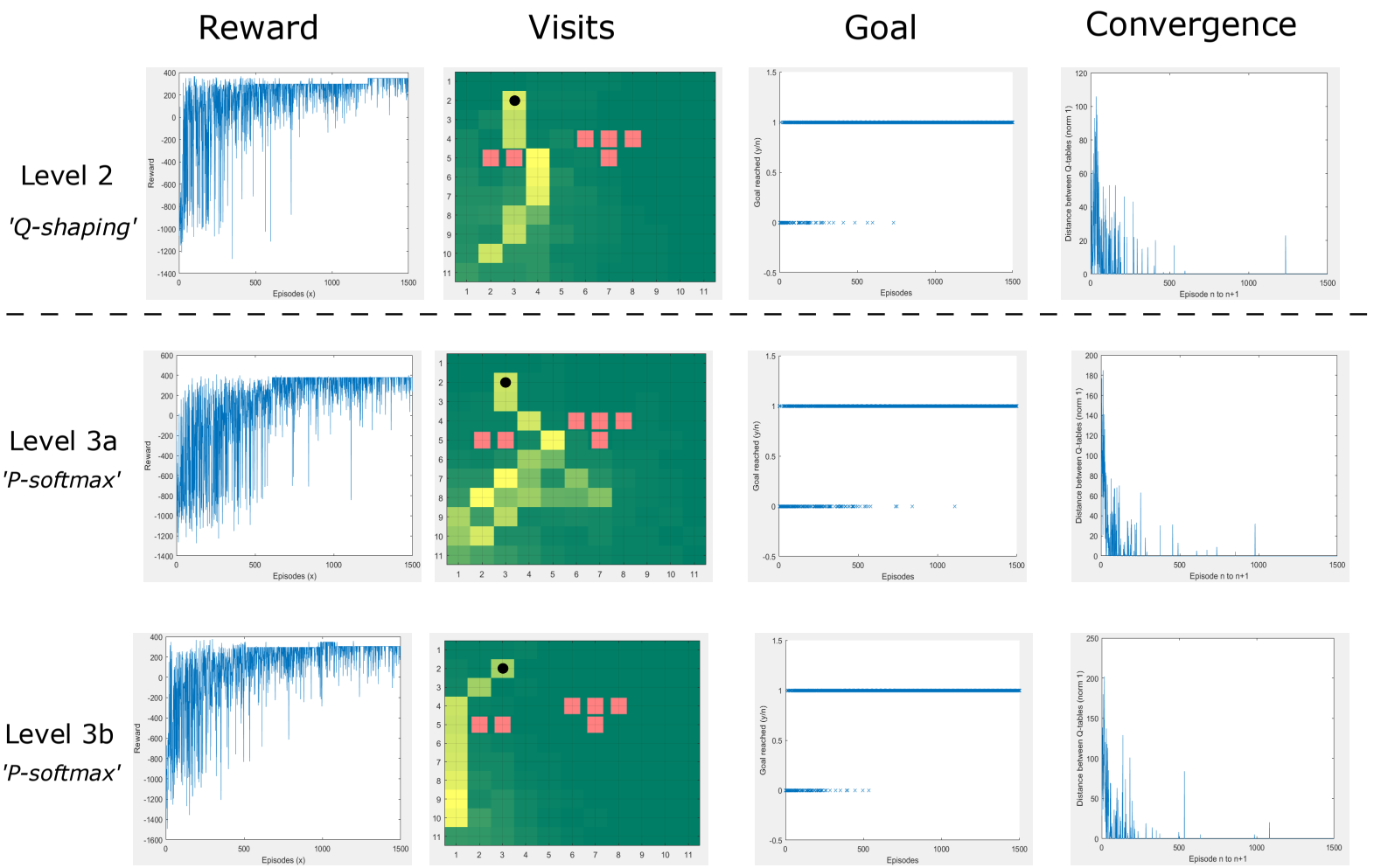


Figure 5.17: Results for Levels 2 and 3

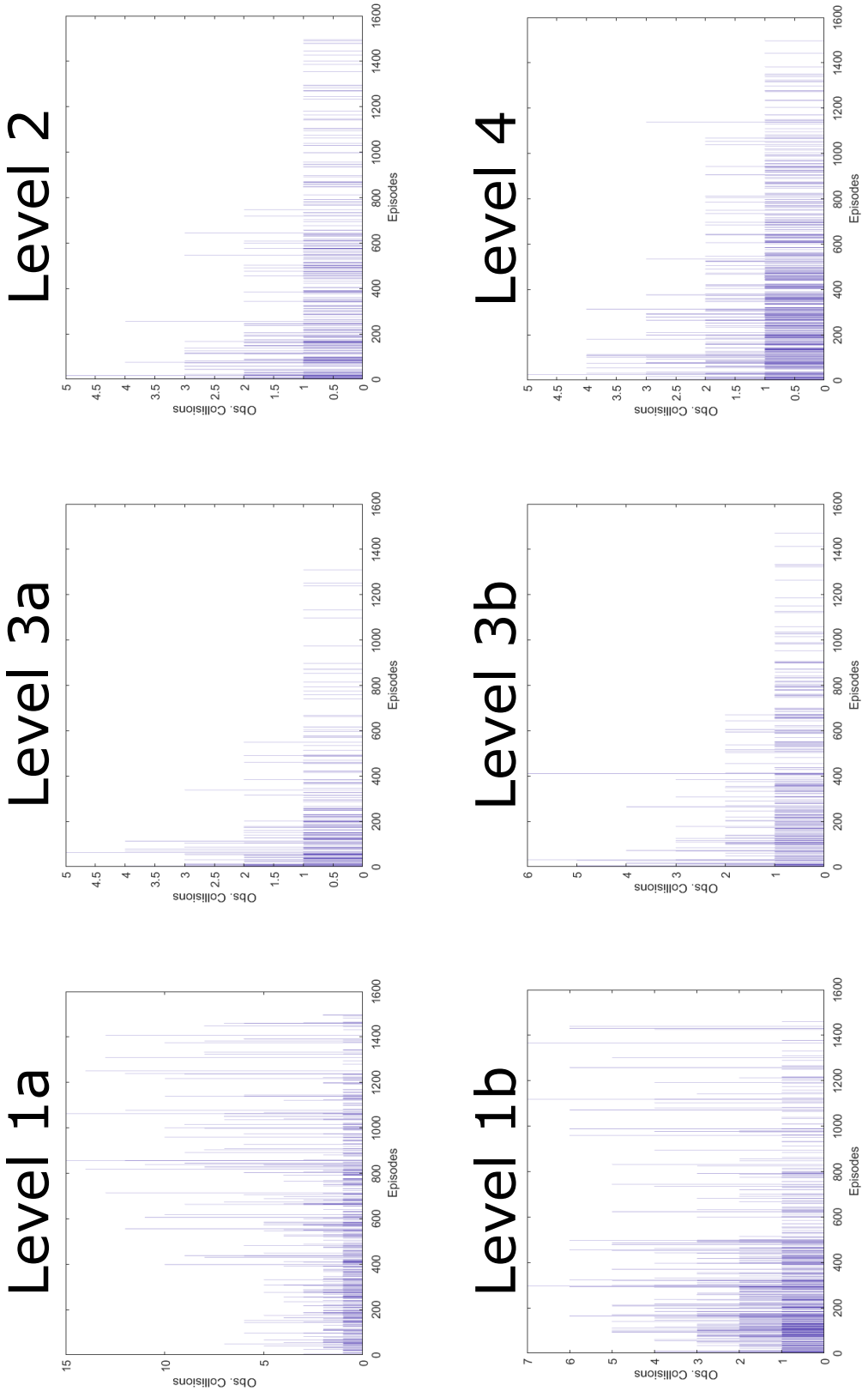


Figure 5.18: Collisions over the whole run

Chapter 6

Research Method

The use of Artificial Potential Field driven Reinforcement Learning has the capacity to make exploration safer. The methods and approaches discussed so far are promising and must now be rigorously tested and adapted to a more realistic scenario, with a view to bring the theory closer to application on an aircraft. Namely, the problem must be scaled up.

After refining the implementation of the 'agent dynamics' based approach as detailed in Ch. 5, the grid-world simulation will be subjected to some changes within the action-selection process, namely by analysing the Level 3 approach in more detail. Although most research in safe exploration starts off with defining and including a parameter to represent 'risk' as seen in Ch. 3, here, risk is assessed through information obtained using a separate source, namely the potential field. Approaches which directly affect the action-selection policy of the agent will then be tested. This should result in a comprehensive survey of how APFs can be applied within the RL framework.

The next steps are detailed in Fig. 6.1 where Level 1 proposes 3 complementary directions this research can go, namely adding more dynamism to the environment (perhaps making it continuous or atleast more refined), carrying out a more thorough analysis on the 'P-softmax' action selection (started in Ch.5) and finally, analysing and suggesting solutions to local minima problems which arise due to the use of APFs. The first branch also considers adding inertia effects to the agent, essentially preventing it from taking unrealistic actions within its state-space. Level 2 then takes the simulation closer to reality by adding more states and validating the approach with data from a UAV model. It is not expected that Level 2 will be reached in the duration of this thesis project, but should be considered for future work.

Scaling the problem up is expected to present further challenges. The data sets are going to be larger and since RL is not known for its ability to deal with higher order problems on-line, other solutions will have to be proposed. One such idea is the storage of large data sets in adaptive state graphs. Regions of the state-space are forgotten and re-learnt as and when the agent needs it. This will still be computationally expensive for on-line, in-flight usage, but will save on memory as compared to traditional methods (e.g.,

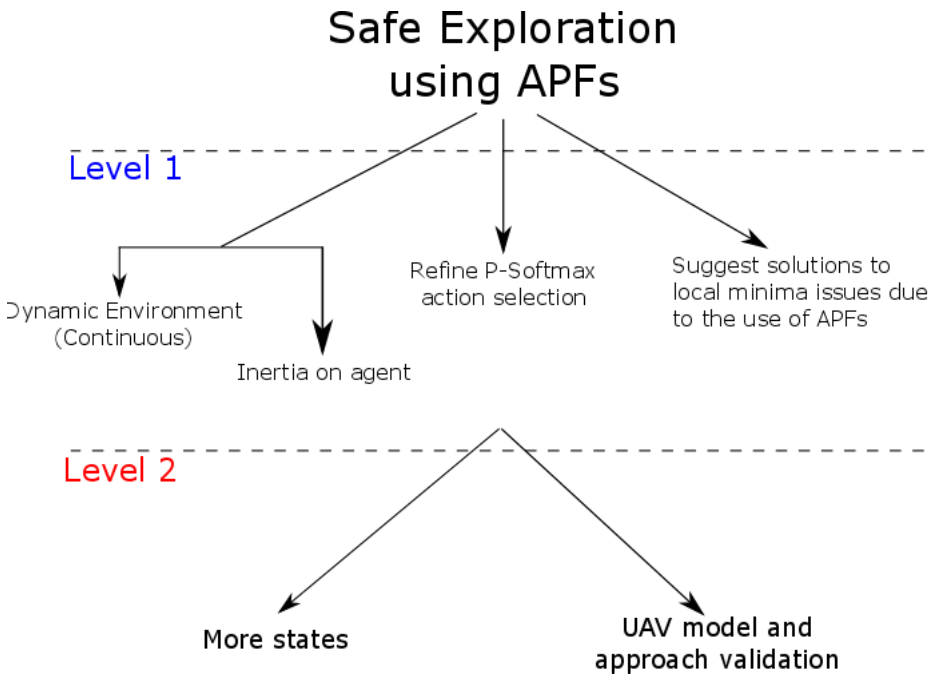


Figure 6.1: Next steps for the thesis

storing the multi-dimensional Q-table for the whole state-space). Another issue will be the interpretation of the generalised forces of the potential field to make them suitable for the UAV. In a real-life scenario, it is not just a simulated 'agent' that can immediately react to a repulsive/attractive 'force', but an actual aircraft that has to translate this information to an appropriate command signal to its actuators.

Part III

Extended Results

Chapter 7

Detailed Results and Analysis

The gridworld simulation proposed in Part II is now extended to include four states, namely velocities and positions in x and y directions **Vx, Vy, x** and **y**. The environment is an 11x11 grid world, where some of the grid-squares are considered obstacles. The agent, represented by a black cell in Fig. 7.1 has the task of learning about its environment, and over time, maturing a policy that allows it to avoid the obstacles and reach its goal cell, G, with no overshoot, beginning from the start cell, S.

The goal state, **G** may be represented by Eq. 7.1. If the agent passes over the goal position in its run, it is said to have overshot the goal. This is elaborated upon in Fig. 7.2. Here, a represents the change in velocities taken as action by the agent. On the left, the agent already has velocity values of 3 in both x and y directions. The action taken is $(0, 0)$, namely no velocity change. This takes it through a path that goes over the goal state. The agent carries out this movement one step at a time, where, at the second step, the agent will be at the goal position. However, due to the fact that it still has residual speed, it will end up overshooting. The situation on the right is of a valid goal state. While there is no overshoot, it must be noted here that both scenarios have residual inertia. It is not imposed that the agent must reach the goal position with zero velocities, as discussed previously.

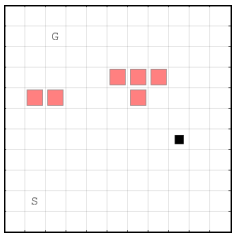


Figure 7.1: Gridworld with obstacles, start (S) and goal (G) positions

The goal state is made up of a user chosen position and velocity. For the latter, perhaps the most realistic case would be to impose a zero-velocity condition on the agent, i.e. it should arrive at the goal location with no residual speed. However, this is also the most restrictive and therefore difficult, condition. In order to ensure that the simulation results can be interpreted and compared, a less restrictive goal condition is set, namely that the agent should learn and condition its algorithm such that it arrives at the goal position with no overshoot instead of no velocity. In order to clarify this, it is imperative

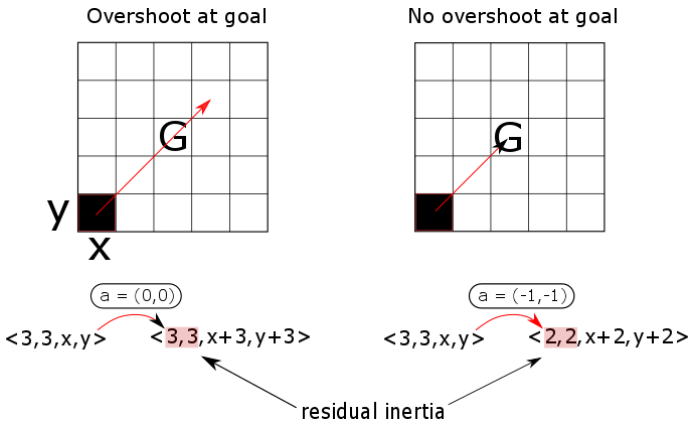


Figure 7.2: Goal Overshoot scenario

to explain the actions that are available to the agent. Every step, the agent has the option to increase or decrease its change in horizontal and vertical velocity, namely ΔV_x and ΔV_y. Once this is set, the agent can move one block at a time, where the number of blocks moved (and the direction) in that step is determined by the current position, velocity and change in velocity at that state.

$$\begin{aligned} G_{ideal} &\rightarrow < 0, 0, x_{goal}, y_{goal} > \\ G_{sim} &\rightarrow < V_x, V_y, x_{goal}, y_{goal} > \end{aligned} \tag{7.1}$$

For completeness, the range of possible values for this simulation can be seen in Fig. 7.3. There are 49 combinations of velocities that are possible in this simulation (7 from V_x and V_y each, the values of which both go from -3 to 3). As far as actions go, each step, the agent can choose from decreasing or increasing its velocity in x and y direction by 1, or not changing it at all. For clarity, the whole set is written out in Fig. 7.3. These values are chosen based on an informal tradeoff between model size (and therefore computational complexity and time), and return in terms of effect observability. What is meant by this is that the chosen size still allows for the observation of potential based interaction effects. A bigger state-space by magnitude would not add much.

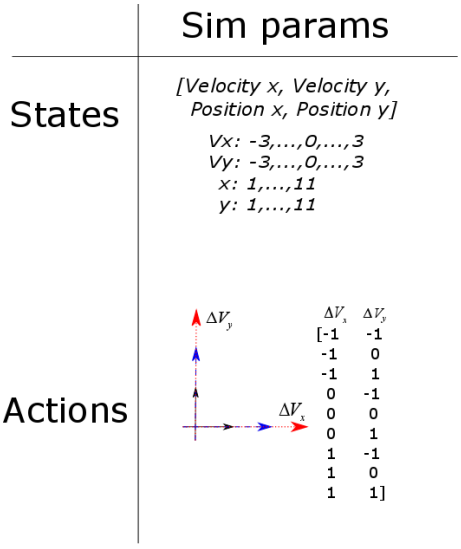


Figure 7.3: Model parameters

Regarding the potential field, the P-table, introduced in Section 5.1 of Part II, is an [nx1] table, where n is the total number of states (in this case, 5929 [7x7x11x11]), where the presence of potential in each state is recorded in the appropriate cell. The magnitude of

this number represents the strength. It must also be noted here that, since the potential 'field' is only distance based, if a potential is detected at a certain state, that same potential is assigned to every state with the same position values. Each position will therefore have 49 associated states due to the possible velocity value combinations.

7.1 Elaboration of Analysis Methods

In order to perform a thorough analysis, the model is explored in 2 aspects, one in terms of *framework* and the other, in terms of *setup*. The former refers to the 4 levels of analysis elaborated upon in Section 5.2.1 of Part II, out of which Level 1 and 4 are taken as comparison baselines and Level 3, referring to the adaptation of softmax to include potential field information, is explored in more detail. Level 2 will not be explored in this paper due to the difficulty of obtaining an insight into how Q-values are actually affected. 'Setup' refers to the obstacle environment itself. It is most interesting to put the agent in a situation where action selection and policy convergence is not trivial. Any algorithms using APFs are naturally susceptible to areas of local minima and 3 different setups have been proposed in Fig. 7.4 in order to observe how the agent would deal with these situations.

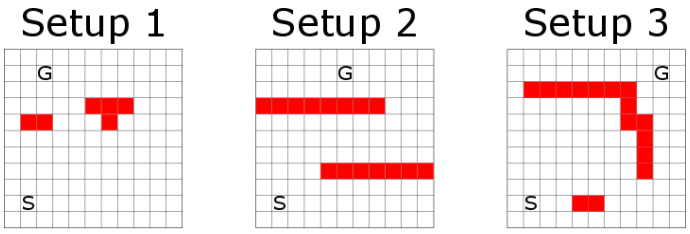


Figure 7.4: Proposed Simulation Setups to analyse agent adaptability to local minima

In order to present the reader with a visual representation of the results, these are presented in 2 distinct sections, one for Level 1 and 4 (essentially the same framework but with different κ values) and one for 2 softmax variations (classified as Level 3). Each section contains 2 sets of results. The first set contains a plot of **rewards per episode**, obtained over the whole run of 2500 episodes and a graphic of whether the **goal has been reached per episode**. The second presents results of observed **collisions** and **convergence** in more detail.

Fig. 5.9 first introduced the proposed softmax variations. The first and second modifications can be considered similar, the only difference being a scaling with the temperature parameter, τ. Therefore, for this analysis, the second and third modifications, shown in Eq. 7.3 and called Level 3b and Level 3c respectively, are chosen. These are expected to show different behaviour patterns since, in the second case, when the parameter q is set to zero, the exponent contains a pure Q-term and the agent is expected to act greedily. However, in the third modification, the $qP(s)$ term is multiplied, and setting q to zero would result in the whole exponent being zero. This is expected to result in random behaviour on behalf of the agent.

In the first set of results, the second figure shows whether the agent arrives at the goal state during any run in the episode. The second set of figures shows two aspects of the

simulation run, convergence and collisions. Convergence can be shown by applying the first distance norm, as shown in Eq. 7.2, on consecutive Q-tables. Here, **N** and **M** refer to the dimensions of the Q-table, where NM would then be the number of states. This was also previously introduced in Part II. In order to make definitive statements about policy convergence and to avoid interference effects from exploratory actions, this norm is applied to the Q-tables obtained after every 100 episodes, where the last episode is run with ϵ set to zero. This ensures a pure greedy policy and shows how the agent is conditioned up to that point.

$$d_1(Q_k, Q_{k+1}) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} |Q_k(i,j) - Q_{k+1}(i,j)|}{NM} \tag{7.2}$$

Since the velocity values in x and y directions are contained within the state, the magnitude of the resultant vector is taken as the speed value. For situations where the agent takes an action that drives it into a wall or into an obstacle, the average speed is taken as the collision value for that episode. A distinction can be made between the magnitude and number of hits per episode. Furthermore, wall and obstacle hits are differentiated and these are displayed in the last set of figures. For ease of interpretation, instead of displaying the data points for each episode, only one data point is plotted per 100 episodes. This is obtained by taking the mean of all the data points for each 100 episode set.

$$p(a|s) = \frac{e^{\left(\frac{Q_t(s,a)}{\tau} + \mathbf{qP(s)}\right)}}{\sum_{b=1}^{n} e^{\left(\frac{Q_t(s,b)}{\tau} + qP(s)\right)}}$$
$$p(a|s) = \frac{e^{\left(\mathbf{qP(s)}\frac{Q_t(s,a)}{\tau}\right)}}{\sum_{b=1}^{n} e^{\left(qP(s)\frac{Q_t(s,b)}{\tau}\right)}} \tag{7.3}$$

An analysis of the effects of changing key parameters of the L3 framework is presented in Section 7.3. Detailed results for all relevant levels are presented in Section 7.2, the general explanations of which have been introduced in this section. It must be noted that up to this point, these results are presented only for Setup 1 (refer to Fig. 7.4). A discussion on the other setups is presented in Ch 8.

7.2 Detailed Results

This section presents detailed simulation results for Levels 1, 4 and 3 of the framework introduced previously. The key tunable parameter for L1 and 4 is κ, which essentially regulates the sensitivity of the agent to the potential field. The presented results are for κ values of 0, 15 and > 30. This is because $\kappa > 30$ is equivalent to the agent taking purely reward based actions without taking into account any potential based action suggestions. On the other hand, $\kappa = 0$ only uses potential-based actions. The analysis in this section will tackle the various aspects of the presented graphs individually. The effect of changing κ on the reward profile will be discussed first, followed by a discussion on the collisions. This is then concluded with an analysis of the convergence behaviour.

Reward behaviour over the whole training set of 2500 episodes displays significant change as κ is increased. For $\kappa = 0$, referring to Fig. 7.5, the agent shows reluctant and

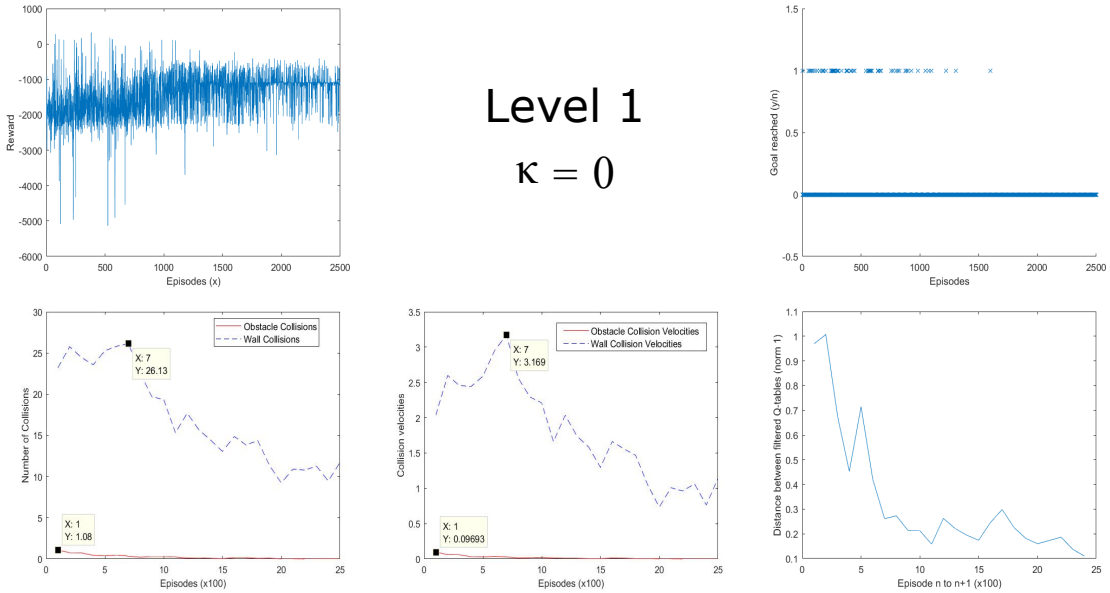


Figure 7.5: Detailed simulation results for L1 ($\kappa = 0$)

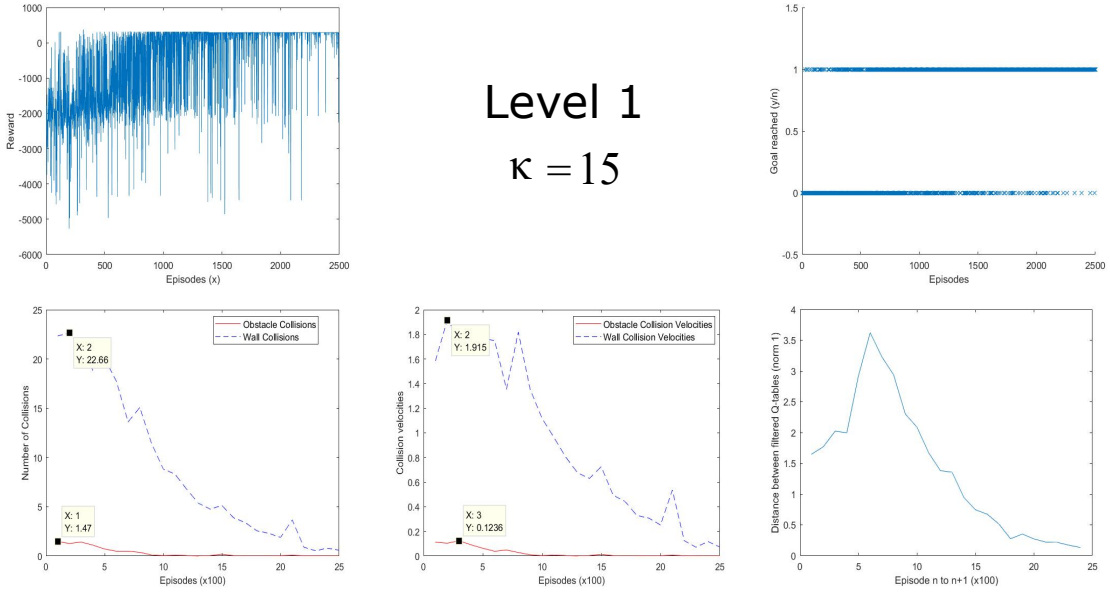


Figure 7.6: Detailed simulation results for L1 ($\kappa = 15$)

slow learning. Visually, slight convergence to a final baseline value can be seen. This is the value around which the reward per episode fluctuates for about the last 500 episodes. This is by no means optimal. The decrease in the scale of reward fluctuations over the whole run can be attributed to the decreasing ϵ profile that is used in the simulation. This exploration factor is decreased over the whole run as it is expected that the agent, after learning and conditioning its behaviour over the initial episodes, will not need to explore so much and can instead exploit what it has learnt. Looking at the same plot for increasing κ values in Figs. 7.6 and 7.7, it can be seen that the agent converges to an

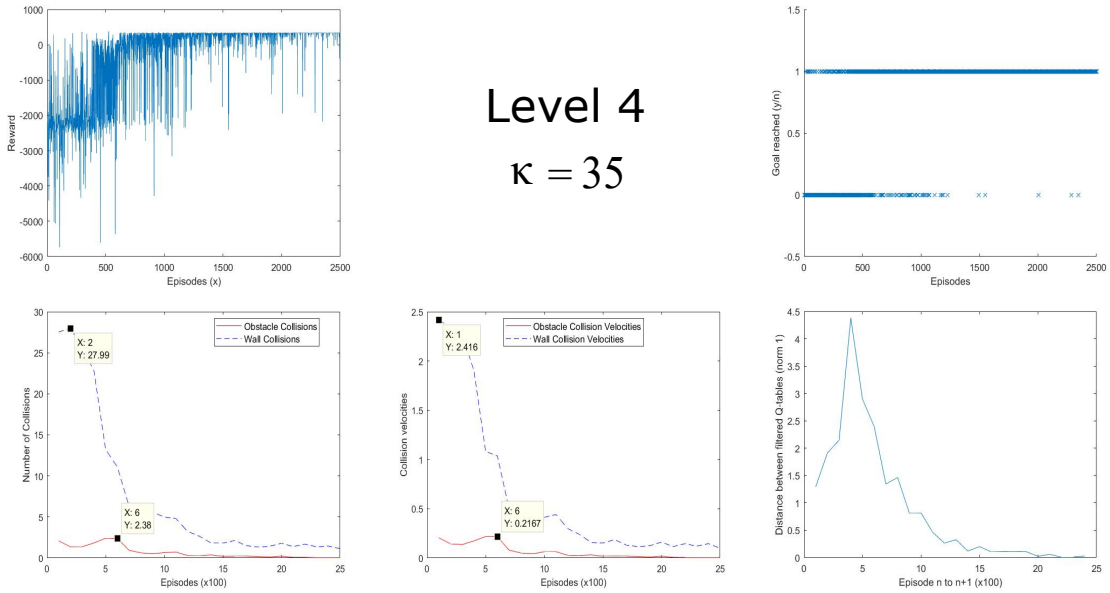


Figure 7.7: Detailed simulation results for L4 ($\kappa > 30$)

optimal value more quickly and maintains this baseline value for the rest of the run. This occurs earlier for higher values of κ. What this also means is that the reward over the whole run is higher as κ is increased. To put these observations in context, $\kappa = 0$ fully uses the potential information without any regulation. This is counter-productive since the agent avoids any zones which could potentially lead to conflict. It is more content to move around where it started, where it does not observe any potential fields. The goal is therefore effectively never reached despite the actions taken being extremely safe. On the other hand, Level 4 behaviour in Fig. 7.7 shows favourable performance while having a larger magnitude of fluctuation in the beginning of the run where the agent did not have the potential information as a guide. Another significant observation is that of the **goal-reaching** behaviour. The lower the κ, the less favourable this is. With a κ value of 0, the agent reaches the goal in the beginning due to some lucky exploratory actions, but fails to successfully learn and over time conditions itself to carry out safe but unfavourable (in terms of performance) behaviour. In contrast, ignoring the potential field in Level 4 shows behaviour where, for about the final 1000 episodes, the agent consistently reaches the goal. Learning here has therefore been successful. As expected, an intermediate κ setting of 15 shows behaviour that is in between the two discussed cases.

Up until this point, Level 4, where the potential field influence is not felt by the agent, seems to be showing more favourable behaviour. However, the over-arching goal of this paper is to explore safety promoting methods. In order to investigate this, **collision behaviour** must be analysed. An initial observation on all three settings is that the number and velocity of collisions goes down as the runs progress. This simply shows that the learning is working as expected. However, this is also where the benefits of using potential fields can be seen. Referring to the obstacle collision trend lines on all three figures, both the number and velocity of collisions are consistently higher for Level 4 (no potential) than for either of the Level 1 runs. These values can be seen on Figs. 7.5-7.7, 7.9 and 7.10 and have also been compiled in Table 7.1 in the Appendix, for ease of

$$r\_avg = \frac{\sum_{i=1}^{2500} r_i}{2500}$$
$$safmet = \frac{\sum_{i=1}^{2500} (V_{t_i})^2}{2500} \tag{7.4}$$

comparison. This shows much safer behaviour on behalf of the agent when exposed to the effects of a potential field. It can also be observed over all three frameworks that the number and velocity of wall collisions is much higher than that of the obstacle collisions. This is because the walls are not treated in the simulation as critical and therefore are not considered must-avoid regions. A potential field has also only been imposed on the obstacles and not on the walls.

A thorough analysis has been carried out in order to generalise these observations and the results are presented in Fig. 7.8. The data for this plot is generated by running the simulation for varying κ values with a Level 1 setting. These values range from 1 to 35, increasing in increments of 5 in order to keep the scale of the simulation manageable. In order to analyse both the performance and the safety of the various settings, two metrics are used, **r_avg** and **safmet**. As can be seen in Eq. 7.4, the former is essentially the mean return over all the episodes, where i refers to each episode out of 2500 and r_i is the total reward in episode i. This is an indication of performance, as the higher this value, the more reward the agent has accumulated, likely by reaching the goal more times while avoiding too many negative-reward inducing steps. The second metric, **safmet**, reflects safety and is the sum of squares of all the resultant obstacle collision velocity values, over all episodes. Here, V_{t_i} is the resultant velocity value for obstacle collisions in episode i. This, in turn, is calculated by taking the square root of the addition of the squares of the x and y components of the collision velocities. Here, only obstacle collisions are taken into account. The reader is reminded here that the potential field has only been imposed around the obstacles and not on the walls. Collisions into walls, while interesting to observe model behaviour, do not directly reflect the influence of the potential field on safety, and have therefore not been included in the calculation of this metric.

For each run, the values of r_avg and $safmet^{-1}$ are recorded as indicators of performance and safety behaviour respectively. Fig. 7.8 shows that there is a tradeoff to be made and an optimal setting where both performance and safety of exploration can benefit. For the current case, this results in a κ value between 10 and 15 but, as discussed previously, this is subject to change, based on the experiment setup. It is also interesting to observe in this plot, as mentioned earlier, that a low κ value, while being extremely safe, suffers from lack of performance. Conversely, a high κ run will potentially learn quickly and reach the goal more times (high performance) at the cost of colliding more often and with higher velocities in the beginning, thereby brining its safety rating down. Neither case is acceptable for an autonomous drone and a trade-off is therefore essential.

The final plot in Figs. 7.5, 7.6 and 7.7 shows convergence behaviour. The norm used

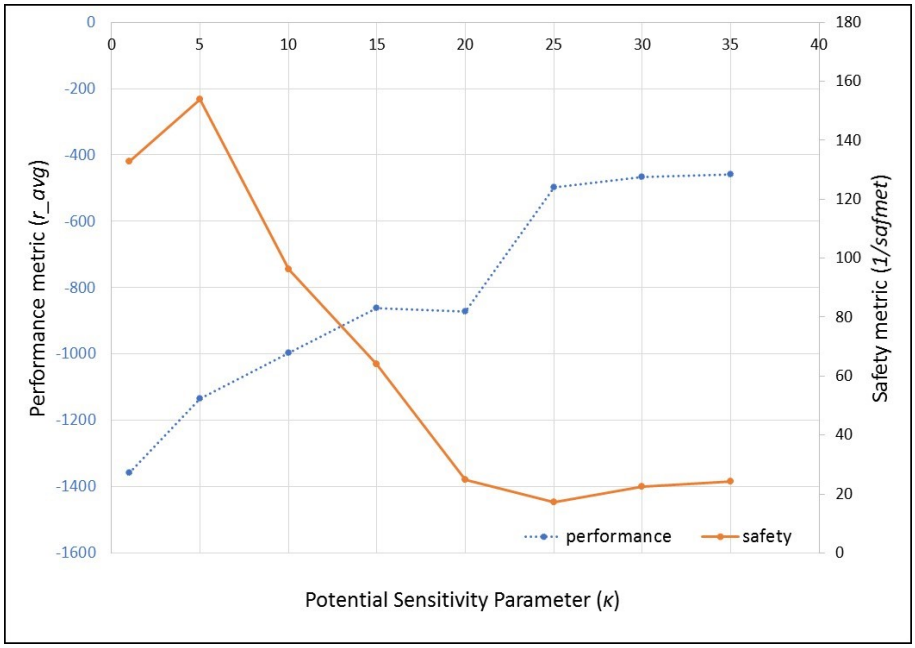


Figure 7.8: Tradeoff between performance and safety with varying κ

to generate these values essentially calculates the distance between subsequent Q-tables, as per Eq. 7.2. As this value goes down, the interpretation is that the variance in the Q-values decreases. This reflects agent conditioning as it is not exploring and changing the relative weights of its actions as much. Towards the end, as this norm approaches zero, it can be said that, for the majority of states, the agent has picked the actions that it has decided are most favourable. With the exception of $\kappa = 0$ (Fig. 7.5) where the trend is consistently decreasing, these plots also show an initial spike after which the expected decreasing behaviour resumes. This can be explained by high initial exploration. As this takes place, the Q-table changes drastically every run. Only when the effects of learning can be felt by the agent to some significance and it starts repeating certain actions per state, does the norm start decreasing again. For $\kappa = 0$, the exploration is, from the very beginning, heavily regulated by the restrictions imposed by the potential field. This results in even the initial exploratory actions being conditioned, resulting in relatively less positive change between subsequent Q-tables. Over time, the changes only get smaller as the agent settles into what has been trained.

The discussion so far has focussed on the Level 1 and 4 settings, where κ is the main potential field influencer. However, as discussed previously, the same results can be presented for the softmax based Level 3b and Level 3c modifications, where $q = 40$ and $\tau = 5$. Fig. 7.10 shows the results for the softmax modification that is not explored in more detail previously and shows precisely why. The fluctuations in the reward plot only marginally improve over the whole run, suggesting minimal learning. Furthermore, it almost never reaches the goal, and in this sense, displays the worst performance seen so far. Furthermore, despite the number and velocity of collisions decreasing over the whole run, the average value is still higher than the rest of the settings, which is undesirable. With regards to the reward plot for Level 3b, shown in Fig. 7.9, the behaviour here is more favourable and over time, the agent also learns to reach the goal more often as its

Q and P-tables get more conditioned. Convergence shows similar behaviour to that of Level 1 and 4, as discussed previously.

Table 7.1: Detailed collision data (Red: no potential information used)

	Obstacle Collisions		Wall Collisions		
Level	Max Velocity	Max Number	Max Velocity	Max Number	**Mean Reward**
1 (kappa = 0)	0.097	1	3.169	26	-1406
1 (kappa = 15)	0.124	1	1.915	23	-511.7
4 (kappa = 35)	0.217	2	2.416	28	-300
3b	0.248	2	2.332	28	-511.6
3c	0.483	5	2.02	23	-1367

Table 7.1 summarises the collision data for all the considered frameworks, with Level 4, where no potential information is used, being shown in red. So far, Level 1 and Level 4 have shown the most favourable behaviour. Comparing the number and maximum velocity of wall and obstacle collisions, keeping in mind that it is desired for these numbers to be as low as possible, Level 1 with $\kappa = 15$ displays better performance than Level 3*b*. However, in terms of the whole run, these two levels display very similar mean reward values, indicating similar performance.

In order to explain the extremely unfavourable behaviour displayed by Level 3c, it is worth looking at the formula in more detail. Referring to the second expression in eq. 7.3, Level 3c includes a scaled version of the potential value multiplied by the Q-value in the exponential. However, due to the setup of the rewards in this simulation, the majority of the relevant Q-values are negative by default. Furthermore, all potential field values are negative, with a more negative value signifying severe potential (eg. closer to an obstacle). Multiplying these two negatives results in a positive value in the exponential when the agent is exposed to an unfavourable state-action pair. What this leads to is the agent conditioning itself to pick the worst actions in this case (eg. speeding up and heading towards an obstacle) instead of a safer option which would ideally eliminate these cases completely. As a result, it hardly ever reaches the goal and doesn't receive significantly higher rewards over time. Perhaps multiplying by $-P(s)$ instead of $P(s)$ in this modification would be one step towards a working solution. This has not been explored in detail since the other options display more favourable performance already and modification of Level 3c is considered to be outside the scope.

7.3 Parameter Analysis

For the softmax framework, based on the disucssion in Section 7.2, Level 3b is chosen for detailed analysis. The experiment can be broken down into the following:

- **Independent Variables:** τ, q
- **Control Variables:** κ, setup, potential strengths (and interpretation method of P-table), reward distribution, number of actions and number of states, maximum number of steps, learning rate (α), discount factor (γ), exploration factor (ϵ) profile

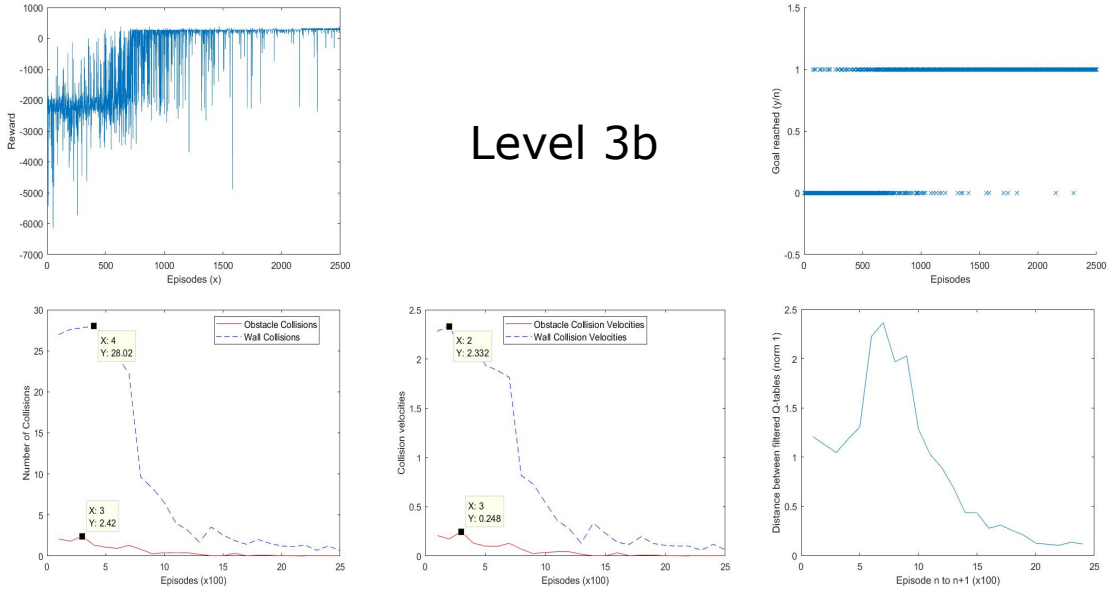


Figure 7.9: Detailed simulation results for L3b

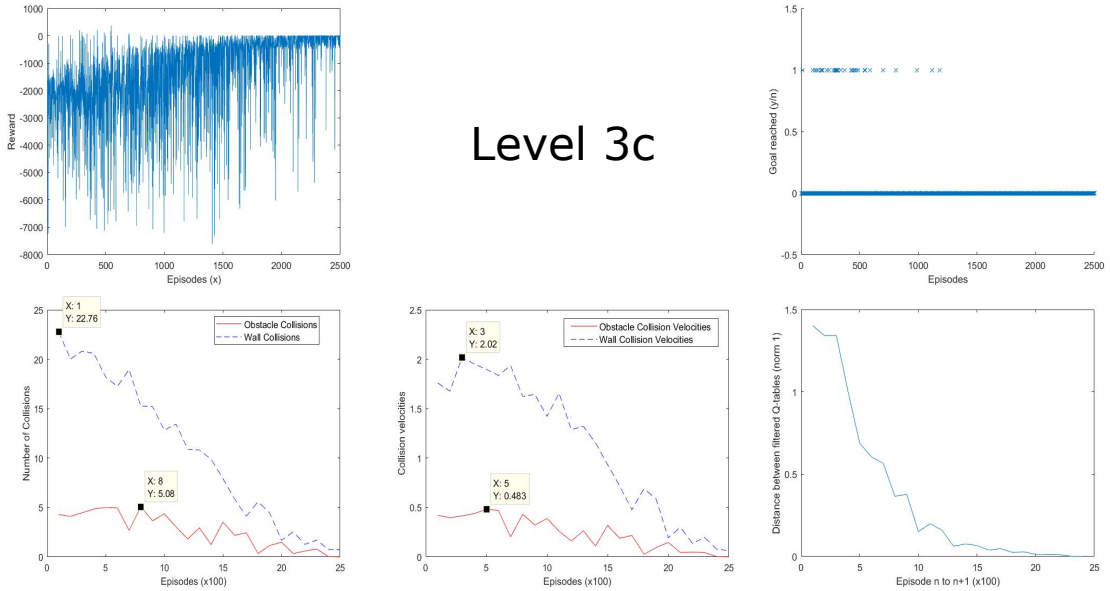


Figure 7.10: Detailed simulation results for L3c

- **Dependent Variables:** performance metric (r_avg) and safety metric ($safmet$)

A standard set of values for τ and q are used to run the simulation, as shown in Table 7.2. The previously introduced metrics, r_avg and $safmet$ are then recorded in Table 7.2 for a combination of τ and q values. Each combination is run in the simulation four times and the mean value is then taken for analysis. This is done to mitigate, to some extent, the effects of the stochasticity in the simulation and to achieve result consistency.

For **r_avg**, it is desirable for the number to be as large as possible as this indicates the

Table 7.2: Parameter Analysis for Level 3 (Softmax based)

	τ \ q	0	1	20	40	60	100
r_avg	1	-256.9556	-304.29	-399.79	-265.18	-338.39	-426.31
	5	-373.2208	-230.61	**-221.78**	-433.12	-392.28	-360.07
	15	-372.1809	-479.11	-476.43	-426.93	-473.36	-395.98
	20	-474.6333	-540.75	-399.59	-406.15	-393.8	-485.45
	30	-540.4954	-499.41	-467.61	-594.76	-570.47	-508.09

	τ \ q	0	1	20	40	60	100
safmet	1	0.0383	0.0472	0.04818	**0.03933**	0.05813	0.0517
	5	0.056025	0.04513	0.04235	0.05843	0.0458	0.0565
	15	0.05775	0.06535	0.05908	0.0678	0.06348	0.05668
	20	0.067775	0.06	0.05945	0.05943	0.06233	0.06095
	30	0.06895	0.07105	0.0638	0.07448	0.07188	0.06843

highest performance. Conversely, for **safmet**, a lower number indicates safer behaviour for that parameter combination. It must be noted that the table also includes values for simulation runs with $q = 0$. Referring to Fig. 5.9, this is essentially pure softmax with no potential field influence and has been included for comparison. Furthermore, the red cells highlight the best values for each set of τ settings, while the green cell shows the best value over the whole experiment. For the **safmet** table, since the optimum value is for $q = 0$, the next lowest value , which is for $q = 40$ is brought to the reader's attention and is presented in bold. These results are visualised in Figs. 7.11 and 7.12, the former showing the performance analysis and the latter, the analysis for safety. The optimum parameter combinations are also highlighted in these figures.

It must be noted here that these optimal settings are subject to change based on experiment setup and no investigation has been made into the scalability of this algorithm. However, that is outside the scope of this paper. It is interesting to observe that while performance seems to benefit from the addition of the potential scaling factor, q, the trend for safety is harder to generalise. Perhaps counter intuitively, the lowest **safmet** value is for the no potential case ($q = 0$). Looking at Fig. 7.12, while the expectation is for $safmet$ to show a decreasing trend for increasing q, this is not the case. Some trendlines increase while others show a slight decrease. This is not conclusive and at the moment suggests no safety benefits to introducing the q factor to a Level 3b framework. One reason for this may be the assumption that the potential value is not dependent on the action taken and is simply state based. Both Fig. 7.12 and Table 7.2 highlight a value of $q = 40$ as optimal from the ones that do include q. With regards to the temperature parameter, τ, it is used to regulate the probability of action selection per action, over the whole action set. As $\tau \to$ inf, all actions have the same probability, whereas as $\tau \to 0$,

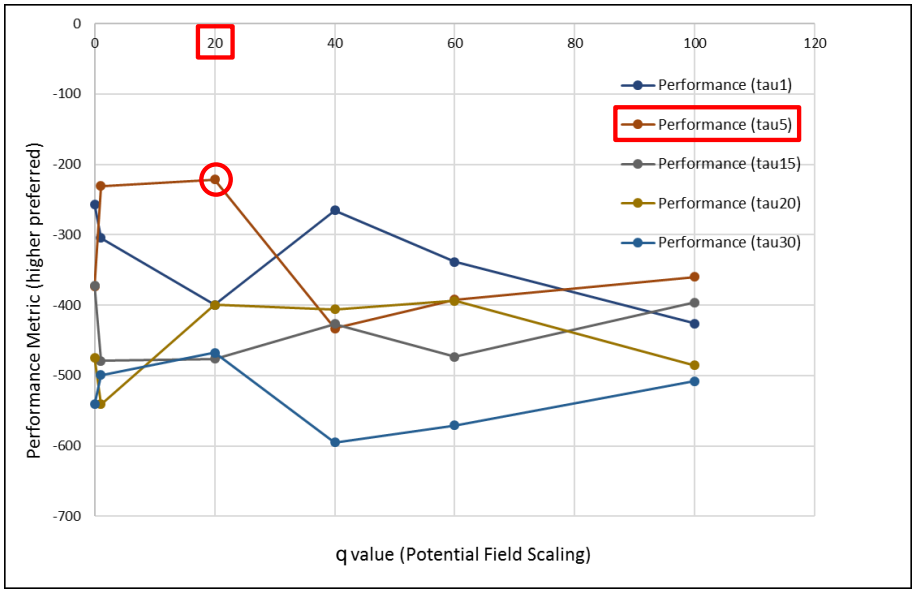


Figure 7.11: Performance parameter analysis for L3b

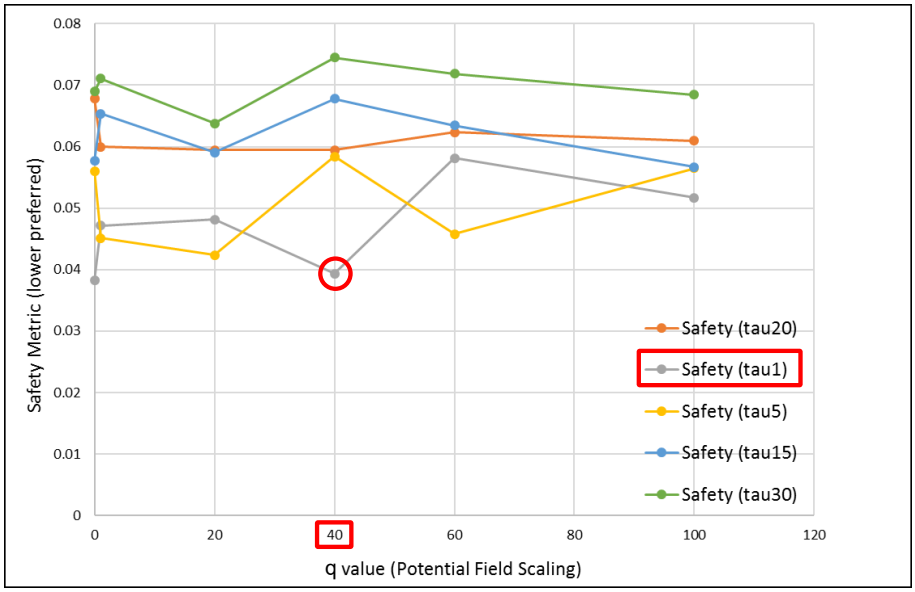


Figure 7.12: Safety parameter analysis for L3b

the probability of action selection approaches 1 for the action with the highest expected reward. Since the potential field information does not directly influence τ in any of the modifications, it is expected to behave and affect action selection in a similar manner. It then follows that a relatively low τ value is seen as optimal. Too high and the effects of learning from the potential field would not be evident (adversely affecting both performance and safety, as can be seen in Figs. 7.11 and 7.12) whereas a τ of zero would not be mathematically consistent. A relatively low τ ensures that quick learning takes place, in order to promote safety.

Chapter 8

Different Simulation Setups

Keeping the previous discussion in mind, a few other simulation setups are explored in more detail here. Specifically, keeping the results of the tradeoff between performance and safety in Ch. 7 in mind, a Level 1 setting with κ values between 10 and 15 is used. This analysis will look at collision behaviour and how the agent adapts to difficult setups. Some examples have been proposed in Fig. 7.4 and are taken as a starting point.

Setup 2

For Setup 2, the trace of the path that the agent takes from start to goal state, after conditioning its Q-table, is presented in Fig. 8.2. This is expected since the agent gets equally repulsed from both sets of obstacles as it searches for a way out of the 'corridor'. Behaviour, of course, depends on the κ value that is set. In this case, a value of 12 displays such movement. If this κ value is set very low, such as 2, the potential field strength is too strong and the agent displays behaviour as in Fig. 8.3 where the potential field prohibits it from reaching the goal. On the other hand, a higher κ of 20 results in Fig. 8.4. Here, the agent is less repulsed by the obstacles and starts to display a sort of 'bouncing' action. This is where it gets repelled by one set of obstacles, only to get repelled back by the other, thereby resulting in an undesirable back and forth action. However, this is avoided by carefully tuning the potential field values and in this case, the agent recovers very quickly midway through the 'corridor'.

The potential field, as perceived by the agent at the end of the run, can be seen in Fig. 8.1 with the white cell signifying the goal position. This is essentially a P-table visualisation and shows that regardless of strategy chosen by the agent to solve this problem, it will have to interact with a potential field in the corridor in order to reach its goal.

Furthermore, following the norm as detailed previously, favourable convergence behaviour can be seen in Fig. 8.5 while the goal profile can be seen in Fig. 8.6 and indicates that, over time, the agent does indeed adapt and reach the goal consistently, despite the setup.

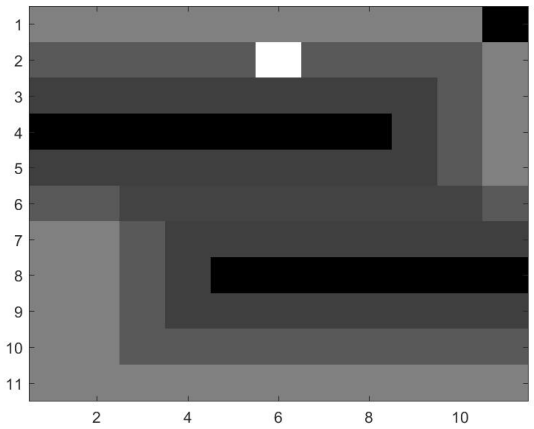


Figure 8.1: Setup 2: Potential field, as seen by the agent at the end of the run

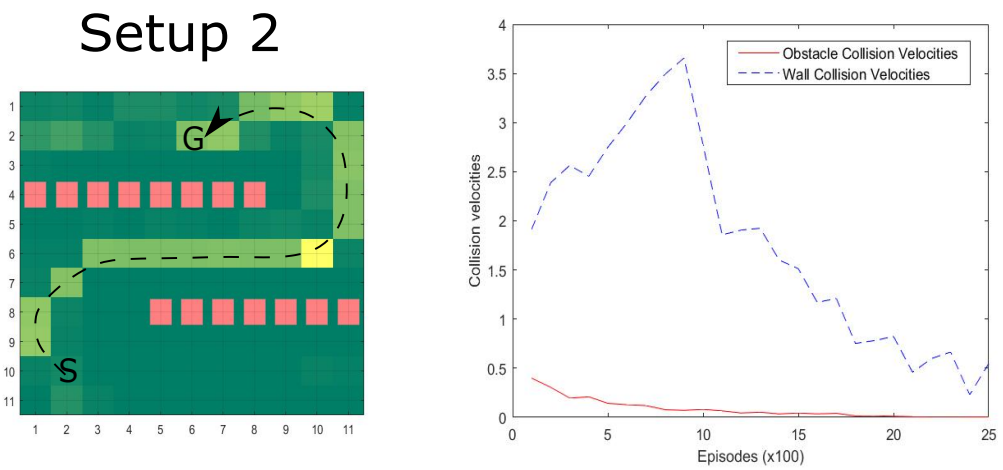


Figure 8.2: Setup 2: Trace of conditioned path and collision velocities

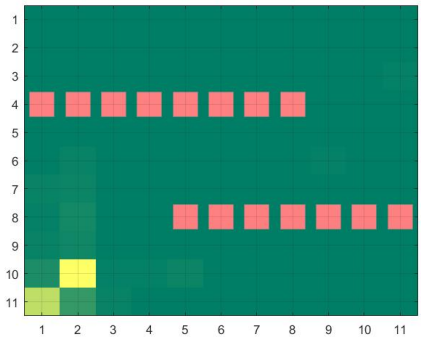


Figure 8.3: Setup 2: Trace of conditioned path with $\kappa = 2$

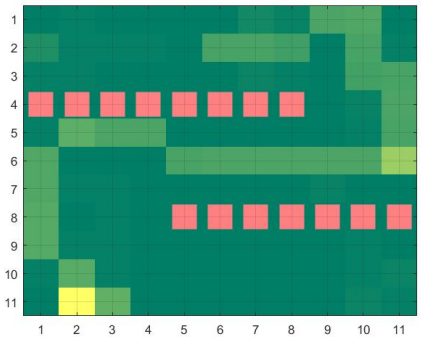


Figure 8.4: Setup 2: Trace of conditioned path with $\kappa = 20$

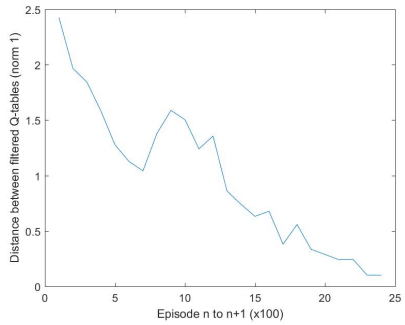


Figure 8.5: Setup 2: Convergence behaviour with $\kappa = 12$

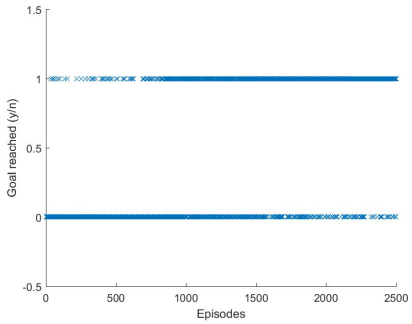


Figure 8.6: Setup 2: Goal profile with $\kappa = 12$

Setup 3

Fig. 8.7 shows the trace and collision values of the agent interacting with Setup 3. This setup is devised in order to observe how the agent can get out of a highly undesirable potential region and get to the goal which is placed on the other side. As can be seen from the trace, the agent has not totally optimised its behaviour and does not take the shortest path to the goal. However, it has identified that it is more beneficial to go through the gap, despite it having a relatively high negative potential due to interference from both obstacle clusters. Furthermore, the meandering behaviour in the beginning is perhaps necessary to get rid of any excess velocity it may have built up before entering the gap since there is not much room (between the obstacle and the wall) to slow down after the gap.

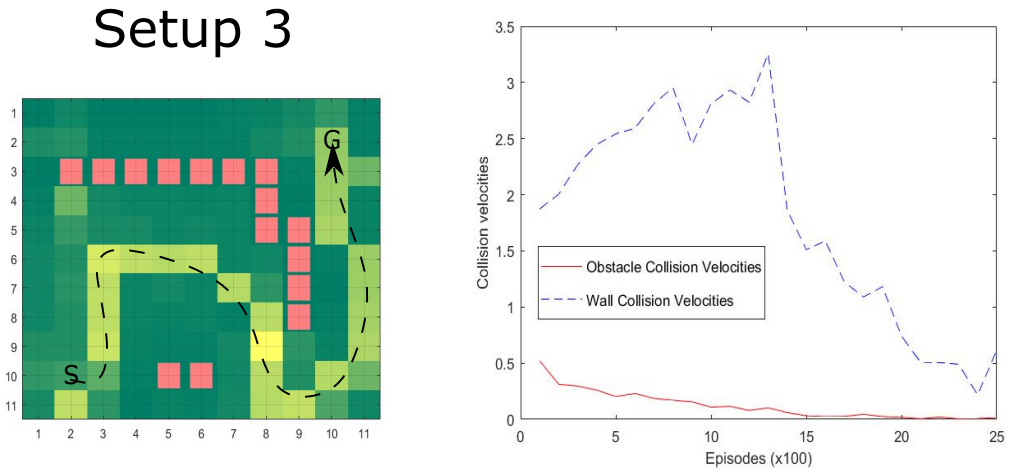


Figure 8.7: Setup 3: Trace of conditioned path and collision velocities

Fig. 8.8 shows a similar potential map, as in the previous section,that the agent can observe. Designing an environment in a way that the agent will have to learn to go through high potential zones in order to reach its goal, was the intention here. This demonstrates the robustness of the algorithm and the nuanced autonomous decision making that can be achieved here. Fig. 8.7 shows the agent deliberately choosing to go through a high

potential zone. Fig. 8.9 shows the path taken by an agent in the same environment, but with a higher κ value. Here, it can be seen that the decision taken is different but equally effective.

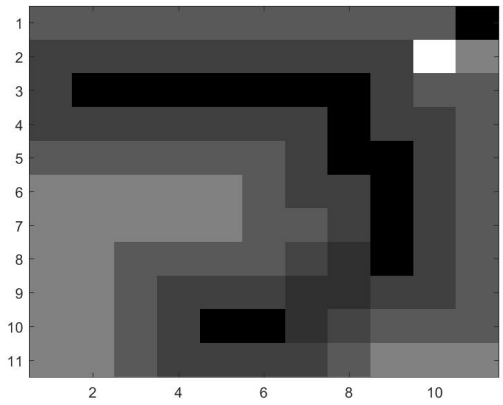


Figure 8.8: Setup 3: Potential field, as seen by the agent at the end of the run

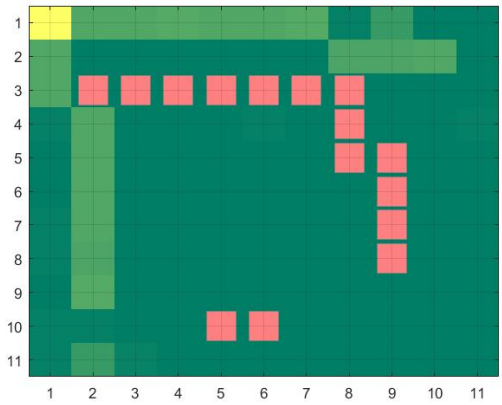


Figure 8.9: Setup 3: A different optimum path, influenced by a higher κ

Convergence and goal reaching behaviour can also be seen in Figs. 8.10 and 8.11 and are similarly favourable.

Both setups, despite being more challenging, show favourable behaviour in terms of collision avoidance. Along with the fact that the collision velocities start at a relatively low value for both, they also drastically decrease as the runs go by, and for both runs, almost approach zero by the final few episodes. This shows the adaptability of using potential fields in order to increase safety of exploration. Despite not having a model of the environment in the beginning of the run, the agent quickly learns which states to avoid. This learning process is made safer by adding another layer of information that the agent can extract from its environment, namely the potential field. A detailed parameter analysis for each setup, using the $safmet$ safety metric and r_avg performance measure, would be the next step in understanding agent behaviour for these setups.

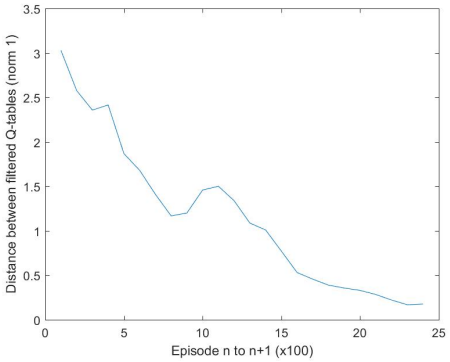


Figure 8.10: Setup 3: Convergence behaviour with $\kappa = 14$

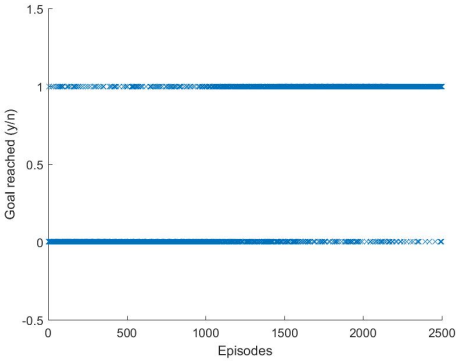


Figure 8.11: Setup 3: Goal profile with $\kappa = 14$

Chapter 9

Conclusions and Recommendations

One of the key issues within RL is the balance between exploration and exploitation. However, before a policy can be considered mature enough to exploit, sufficient exploration needs to take place in order to enable the agent to learn. This process is inherently dangerous. For a software agent, approaching or traversing through a critical state is acceptable and perhaps even desired in the beginning of the episode, since it promotes quicker learning. However, a drone using an RL algorithm to dictate its control laws and overall movement behaviour will need significantly more safeguards to ensure safety. Obstacle collisions, in this case, are considered unacceptable and identified critical states are to be avoided.

Keeping the above context in mind, a Q-learning based test environment is developed in MATLAB where, using nested functions, an RL algorithm is implemented. Various approaches are then proposed where the standard Bellman update equation is modified to include potential values as felt by the agent as it explores its environment. This has an effect on the learning behaviour of the agent and ranges from the potential directly influencing the agent's actions, to it only influencing the reward function. An intermediate approach, Level 2, introduces the concept of a potential based deterministic Q-penalty. This is not explored in detail and can be taken as a recommendation for future work. Level 3, a softmax P-based action selection, is considered in more detail in this thesis. Two modifications to softmax action selection, designed in order to include potential field information, are proposed and tested. Here, no marked safety benefit of using a potential field is seen. The proposal for future work is to increase the size of the experiment in order to make significant statements about effectiveness and also to redesign the Level 3 modification equations, keeping in mind the effect of the P-values on the Q-table.

This research is carried out with a view to demonstrate the applicability of APFs to increasing the safety of autonomous exploration in RL. Despite the need for more research on physical implementation specifics, this thesis has shown that safety can, in fact, be improved by interfacing an APF model with a standard RL algorithm. This is done using metrics based on performance and safety, which are defined using reward and collision data respectively. APFs have been shown to be robust and accommodating of the trial-and-error nature of RL and can be applied in partially known environments. The adaptability

of these methods to multiple setups has been shown, including the fact that convergence and goal seeking behaviour are maintained by the agent, despite drastic changes in its environment. One of the goals of this research that has not been met is the demonstration of a decrease in computational time and complexity while ensuring an increase in safe action selection. This, while being a prompt for future research, can be justified by the exploratory nature of this work. Novel concepts introduced include the P-table, which stores the agent's perception of the environment as it explores. Furthermore, parameters such as κ, τ and q (the latter two introduced for the softmax based Level 3 approaches) only serve to increase the computational complexity of these methods. However, proof of concept, rather than optimisation, is taken to be the leading motivation of this research, and, in that domain, this thesis has met its objectives. The introduction of the velocity states (x and y) have been an attempt in approaching 'realism'. The effects of inertia that inherently lead on from including velocity states complicate this problem and make the results more applicable to a real life situation. That being said, it must be kept in mind that the decision was made early on to use a discrete RL setting in a discrete environment.

The parameter, κ is introduced with the Level 1 framework, and regulates the sensitivity of the agent to the potential information. This essentially puts the approach under the general classification of the 'Risk-Sensitive Criterion', though not quite entirely. In terms of the Safe RL approaches introduced previously, namely the 'Optimisation Criterion' and the 'Exploration Process', the proposed method falls in between the two. The fact that the APF is available to the agent means that, through exploring and filling in information about the potential distribution across the states, it is considering 'risk' already in its action selection. Through multiple trials, it is found that there is a tradeoff to be made between safety and performance. Furthermore, using the APF concept itself does not come without drawbacks. Susceptibility to local minima, oscillations due to similar attractive and repulsive forces and Goal Non-Reachable with Obstacle Nearby (GNRON) are a few of the issues faced. Most of these can be resolved by tuning the actual forces felt by the agent. However, that is not the intention of the research carried out in this thesis. For the sake of the simulation, simple values that allow the agent to not encounter these issues have been selected. No attempt has yet been made to generalise at this stage. The fact that minimal prior research has been carried out on the application of APFs to RL further motivates its exploration in this thesis.

The next step would be to adapt these methods to a continuous world. Furthermore, the agent in this simulation moves in two dimensions, x and y. The addition of a third position state would directly make this more applicable to a real life aircraft scenario. The results are expected to be the same, albeit at the cost of computational complexity due to the increase in size of the Q and P tables. Taking these ideas forward with an end goal of implementation on a real drone platform would be the next step towards validation.

References

[1] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y Ng. An application of reinforcement learning to aerobatic helicopter flight. *Education*, 19:1, 2007.

[2] J.R. Andrews and N. Hogan. Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator. *Control of Manufacturing Processes and Robotic Systems, ASME*, pages 243–251, 1983.

[3] Kavosh Asadi and Michael L. Littman. A New Softmax Operator for Reinforcement Learning. *Cornell University Library*, (Property 3), 2016.

[4] Robert Babuska. *Knowledge Based Control Systems*. Delft Center for Systems and Control, Delft, The Netherlands, 1 edition, 2010.

[5] Javier de Lope and José H Antonio Martín. Learning Autonomous Helicopter Flight with Evolutionary Reinforcement Learning. In Roberto Moreno-Díaz, Franz Pichler, and Alexis Quesada-Arencibia, editors, *Computer Aided Systems Theory - EUROCAST 2009*, number December, pages 75–82, Las Palmas de Gran Canaria, Spain, 2009. Springer.

[6] Javier Garcia and Fernando Fernandez. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45:515–564, 2012.

[7] Javier García and Fernando Fernández. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research*, 16:1437–1480, 2015.

[8] Chris Gaskett. Reinforcement learning under circumstances beyond its control. *International Conference on Computational Intelligence for Modelling Control and Automation*, 2003.

[9] P Geibel. Reinforcement learning with bounded risk. *Icml*, 9(D):162–169, 2001.

[10] Alborz Geramifard. *Practical reinforcement learning using representation learning and safe exploration for large scale Markov decision processes*. PhD thesis, Massachusetts Institute of Technology, 2012.

[11] Fredrik Gustafsson. Control of Inverted Double Pendulum using Reinforcement Learning. Technical report, Stanford University, Stanford, California, USA, 2016.

[12] Alexander Hans, Daniel Schneegaß, Anton Maximilian Schäfer, and Steffen Udluft. Safe Exploration for Reinforcement Learning. In *ESANN2008, Proceedings of the 16th European Symposium on Artificial Neural Networks*, number April, pages 143–148, Bruges, Belgium, 2008.

[13] Matthias Heger. Consideration of Risk in Reinforcement Learning. *Proceedings of the 11th International Conference on Machine Learning (ICML)*, pages 105–111, 1994.

[14] Ronald A Howard and James E Matheson. Risk-Sensitive Markov Decision Processes. *Management Science*, 18(2):356–369, 1972.

[15] Oussama Khatib. Real Time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, 1986.

[16] Shie Mannor and John N Tsitsiklis. Mean-Variance Optimization in Markov Decision Processes. In *Proceedings of the 28th International Conference on Machine Learning*, pages 177–184, Bellevue, Washington, USA, 2011.

[17] Tommaso Mannucci, Erik-Jan van Kampen, Coen C. de Visser, and Qiping Chu. SHERPA: A safe exploration algorithm for RL controllers. In *AIAA Guidance, Navigation, and Control Conference*, page 15, Kissimmee, Florida, 2015. AIAA SciTech.

[18] Harry Markowitz. Portfolio Selection. *The Journal of Finance*, 7(1):77–91, 1952.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, and Martin Wierstra, DaanRiedmiller. Playing Atari with Deep Reinforcement Learning. In *NIPS Deep Learning Workshop 2013*, Lake Tahoe, USA, 2013. DeepMind Technologies.

[20] Teodor Mihai Moldovan and Pieter Abbeel. Safe Exploration in Markov Decision Processes. *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[21] Ralph Neuneier and Oliver Mihatsch. Risk Sensitive Reinforcement Learning. *Machine Learning*, pages 1–7, 2002.

[22] Martin Pecka and Tomas Svoboda. Safe Exploration Techniques for Reinforcement Learning An Overview. *Lecture Notes in Computer Science*, pages 1–19, 2014.

[23] R S Sutton and A G Barto. Reinforcement learning : an introduction. *Neural Networks IEEE Transactions on*, 9(5):1054, 2013.

[24] Charles W Warren. Global Path Planning Using Artificial Potential Fields. *IEEE*, pages 316–321, 1989.

[25] Christopher J C H Watkins and Peter Dayan. Technical Note: Q-Learning. *Machine Learning*, 8(3):279–292, 1992.

[26] Li-juan Xie, Guang-rong Xie, Huan-wen Chen, and Xiao-Li Li. Solution to rein-

forcement learning problems with artificial potential field. *Journal of Central South University of Technology*, 15:552–557, 2008.

[27] Naoto Yoshida, Eiji Uchibe, and Kenji Doya. Reinforcement learning with state-dependent discount factor. *2013 IEEE 3rd Joint International Conference on Development and Learning and Epigenetic Robotics, ICDL 2013 - Electronic Conference Proceedings*, pages 1–6, 2013.

Appendix A

Code Layout

The baseline code structure is briefly introduced in this appendix and is used throughout the thesis for comparison purposes. Fig. A.1 shows a schematic of how the Matlab code is organised in layers. The main file runs `MazeDemo.m` and also generates some post-processing plots (Q-table, Q-table visualised, P-table etc).[1] `MazeDemo.m` takes as the only input, the maximum number of episodes that the user would like the simulation to run for. The function itself is divided into two blocks, namely *Setup* and *Run*.

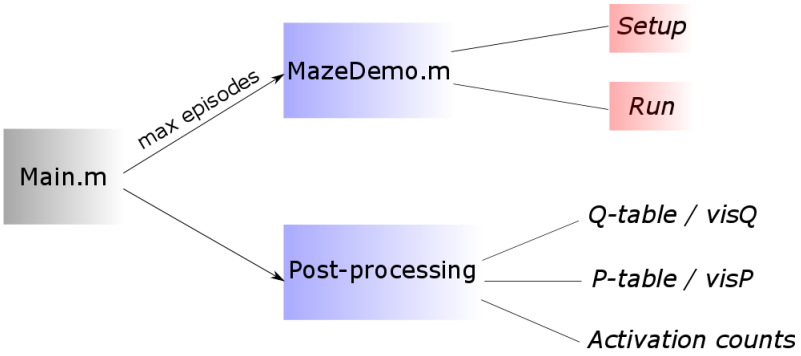


Figure A.1: First Layer of code

A.1 Setup

The purpose of this block is not to run any simulation specific functions, but to initialise some of the required elements. After setting the start (S) and goal (G) positions, the grid itself is created by running `CreateMaze.m`. This is the function where the size of the gridworld and the position of the obstacles are defined. Next, three similar functions are run which return state, action and potential lists. `BuildStateList.m` takes the row and column size of the gridworld as inputs and returns a table containing their cartesian

[1] These terms are elaborated upon in Ch. 5

product. This is the first step to ensuring that there is a way to index every potential cell ('state') in this world. `BuildActionList.m` returns an array of all possible actions that the agent can carry out. `BuildPotentialList.m` at the moment simply returns the number 1, which is then used later to build the **P-table**. This is one possible approach and a short discussion of alternates is presented in Sec. 5.1. The number of states and actions follow from the previous definitions and are used to build a **Q-table**, initialised with uniformly distributed random numbers between 0.0001 and 0.001. These values have been chosen as they are small enough so as not to significantly influence learning, while still enabling the simulation to start off with a sensible policy. Standard RL parameters such as the learning rate α, the discount factor γ, the probability of random action selection ϵ and maximum allowed steps per episode are then set.

A.2 Run

In this block, the function `Episode.m` is run and iterated over the desired number of episodes as previously set by the user. Without going into too much detail about code specifics, Fig. A.2 schematises the general functioning of this block.

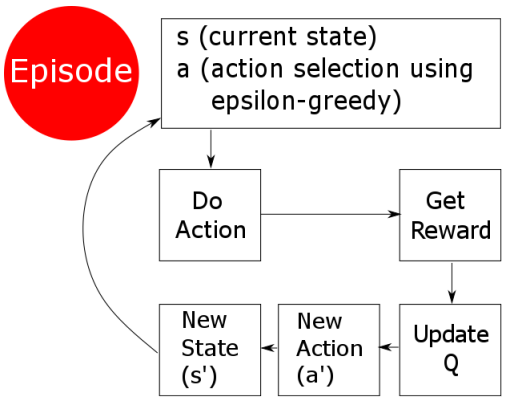


Figure A.2: 'Episode' block

Given a current position of the agent (state s) and action (a, selected using ϵ-greedy [23]), the code carries out the action, gets a reward, cumulates it to the total reward, and then selects the next state (s') and action (a'). The Q-table is then updated, after which the whole process is repeated again.

Appendix B

RL Simulation Levels

Fig. B.1 shows the hierarchy of a typical RL simulation of the type considered in this thesis. Over the whole run, which is what is carried out every time the user initiates a simulation, there may be multiple episodes. All the simulations carried out in the main thesis section consist of 2500 episodes. Each episode then consists of multiple steps. The agent in each episode is allowed to take up to 50 steps, barring any incidences during those 50 steps where it may reach a termination condition. These termination conditions may be designed by the user to be obstacle collisions, wall collisions, overspeed, getting stuck in local minima etc. However, in the case of the simulation carried out in this thesis, the only imposed termination condition is if the agent reaches the goal. Obstacle and wall collisions are penalised but not taken to be situations where the agent must start afresh. This is done to aid learning. Upon termination of an episode, consisting of up to 50 steps, the agent starts the next episode, until the 2500th one is complete.

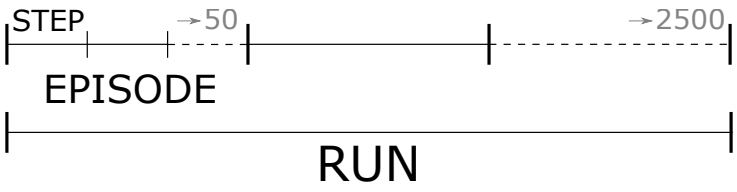


Figure B.1: RL Simulation Hierarchy