Learning to Soar: Exploration Strategies in Reinforcement Learning for Resource-Constrained Missions

Jen Jen Chung

A thesis submitted in fulfillment
of the requirements of the degree of
Doctor of Philosophy

Australian Centre for Field Robotics
School of Aerospace, Mechanical and Mechatronic Engineering
The University of Sydney

March 2014

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

Jen Jen Chung

31 March 2014

Abstract

Jen Jen Chung
The University of Sydney

Doctor of Philosophy
March 2014

Learning to Soar: Exploration Strategies in Reinforcement Learning for Resource-Constrained Missions

An unpowered aerial glider learning to soar in a wind field presents a new manifestation of the exploration-exploitation trade-off. The glider agent begins with limited energy and must explore its state-action space to learn how to perform energy-gaining flight trajectories. However, most actions cause the glider to lose energy, thereby reducing its available flight time. This coupling, between what energy gain rewards are observed and the glider's ability to continue to make new observations to improve its policy, presents new challenges for developing effective exploration strategies that can cater for resource limitations. This thesis proposes a directed, adaptive and nonmyopic exploration strategy in a temporal difference reinforcement learning framework for tackling the resource-constrained exploration-exploitation task of this autonomous soaring problem.

The complete learning algorithm is developed in a SARSA(λ) framework, which uses a Gaussian process with a squared exponential covariance function to approximate the value function. The three key contributions of this thesis form the proposed exploration-exploitation strategy. Firstly, a new information measure is derived from the change in the variance volume surrounding the Gaussian process estimate. This measure of information gain is used to define the exploration reward of an observation. An analytical solution to finding the variance volume is also presented and this result can be extended to any choice of covariance function that satisfies some simple

integrability properties. Secondly, a nonmyopic information value is presented that captures both the immediate exploration reward due to taking an action as well as future exploration opportunities that result. A rollout mechanism is introduced to generate the set of reachable state-actions and the discounted information gain of these potential observations are used to compute the information value. Finally, this information value is combined with the state-action value of SARSA(λ) through a dynamic weighting factor to produce an exploration-exploitation management scheme for resource-constrained learning systems. The proposed learning strategy encourages either exploratory or exploitative behaviour depending on the requirements of the learning task and the available resources.

While the motivating problem behind this work is that of autonomous soaring, the presented exploration strategies can be applied across various learning and information gain tasks. The performance of the learning algorithms presented in this thesis is compared against other SARSA(λ) methods on the standard benchmarking problems, puddle world and cart pole, as well as on a battery cycling problem and the specific resource-constrained autonomous soaring problem. Results show that actively directing exploration to regions of the state-action space with high uncertainty improves the rate of learning, while dynamic management of the exploration-exploitation behaviour according to the available resources produces prudent learning behaviour in resource-constrained systems.

Acknowledgements

To my supervisors, Salah and Nick: thank you for all your guidance, thank you for your advice, and thank you for your criticisms. Thank you for taking me on as a student and thank you for your high expectations. I don't know that this is exactly what you had in mind when we first started working on it four years ago—it seemed to pick up momentum all on its own, so thank you for helping me steer it all the way here.

To the aerial robotics group at CATEC, Aníbal, Antidio and Miguel Ángel: thank you for letting me work in your amazing facilities, thank you for trusting me with your very new and very expensive equipment, and thank you for all the time and support you provided. To all the wonderful people at CATEC: thank you for showing me your beautiful city, thank you for teaching me some very useful Spanish, and thank you for making my time in Seville so memorable. You are a great group of people to work and hang out with and hopefully there will be more adventures in the future.

To the members of the LEAF group: thank you for letting me build planes with you all, thank you for all the fun times in the field lab and at Marulan, and I'm really, truly sorry for crashing that Skywalker.

To Zhe and John: thank you for all your technical support, thank you for helping me solve all the dumb little computer problems I had (especially when you really ought to have been working on your own theses), thank you for introducing me to the joys of Linux and cloud computing, and most of all, thank you for your patience. It would have been impossible to generate all the results in this thesis without your help.

To the amazing group of people at ACFR: thank you for creating such an incredible community to work in, thank you sharing coffee and Tim Tams with me, and thank you for teaching me so much more about the world. Whether it be about single malt whisky, aquarium maintenance, the effect of vitamin deficiencies or real-time strategy computer games; it is surprising the amount of information one seems to absorb just by being here. You are all such diverse, talented and engaged people; it has been a joy to work in your company.

To my family: thank you for your unwavering support, thank you for all the packed lunches, and thank you most of all for always reminding me that there is life outside of the thesis bubble.

For my family

Contents

List of Publications

J. J. Chung, N. R. J. Lawrance and S. Sukkarieh. Learning to Soar: Resource-Constrained Exploration in Reinforcement Learning. *International Journal of Robotics Research*, 2014. Submitted.

J. J. Chung, N. R. J. Lawrance and S. Sukkarieh. Gaussian processes for informative exploration in reinforcement learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2618–2624, 2013.

J. J. Chung, N. R. J. Lawrance and S. Sukkarieh. Resource constrained exploration in reinforcement learning. In *Proceedings of the 2013 Robotics Science and Systems Workshop on Robotic Exploration, Monitoring, and Information Collection: Nonparametric Modeling, Information-Based Control, and Planning under Uncertainty*, pages 1–6, 2013.

J. J. Chung, M. Á. Trujillo Soto and S. Sukkarieh. A new utility function for smooth transition between exploration and exploitation of a wind energy field. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, pages 4999–5005, 2012.

J. J. Chung and S. Sukkarieh. High level risk analysis and decision making regarding the prediction of thermal lift locations for an autonomous Mars glider. In *Proceedings of the 10th Australian Space Science Conference*, pages 237–248, 2010.

N. R. J. Lawrance, J. J. Acevedo, J. J. Chung, J. L. Nguyen, D. Wilson and S. Sukkarieh. Long Endurance Autonomous Flight for Unmanned Aerial Vehicles. *Onera Aerospace Lab Journal*, 2014. Accepted for publication.

G. Brooker, J. Randle, M. E. Attia, Z. Xu, T. Abuhashim, A. Kassir, J. J. Chung, S. Sukkarieh and N. Tahir. Strobe based sensor for tracking multiple locusts from a UAV. In *Proceedings of the Progress in Radar Research*, 2012.

G. Brooker, J. Randle, M. E. Attia, Z. Xu, T. Abuhashim, A. Kassir, J. J. Chung, S. Sukkarieh and N. Tahir. First airborne trial of a UAV based optical locust tracker. In *Proceedings of the Australasian Conference on Robotics and Automation*, pages 1–9, 2011.

List of Figures

List of Tables

List of Algorithms

Nomenclature

Acronyms

DAI	Dynamic Allocation Index
DOF	Degrees Of Freedom
DP	Dynamic Programming
GP	Gaussian Process
MC	Monte Carlo
MCQ-L	Modified Connectionist Q-Learning
MDP	Markov Decision Process
RC	Remote Controlled
RL	Reinforcement Learning
SARSA	State-Action-Reward-State-Action
TC	Tile Coding
TD	Temporal Difference
UAV	Unmanned Aerial Vehicle
UCB	Upper Confidence Bound

Greek Symbols

α	Step size
β	Linear independence measure of a GP observation
γ	Reward discount factor

γ_r	Rollout discount factor
γ_{thres}	Rollout discount threshold
δ	Temporal difference
Δt	Time step
ε	Exploration parameter
ϵ	GP training input score
θ	Linear value function approximator parameters
λ	Eligibility trace decay factor
π	Policy
σ_f^2	GP process variance hyperparameter
σ_n^2	GP noise variance hyperparameter
τ_r	Exploration value decay parameter
ϕ	Linear function approximator basis functions
ω_e	Resource-constrained objective function weighting
ω_t	Time-step-dependent objective function weighting
ψ	Heading/bearing
ω	SARSA(λ) backup normalisation factor component

Roman Symbols

$\mathcal{A}$	Set of actions
a	Action
$\mathcal{BV}$	GP basis vector set
D	Drag
d_{therm}	Distance to thermal centre
d_{thres}	Battery cycling discharge reward threshold
E	Energy

e	Eligibility trace
F	Cart pole input action force
g	Gravity
$\mathcal{GP}$	Gaussian process
$\mathcal{H}$	Differential entropy
h	Glider altitude ratio
I	Information gain
J	Action selection objective function
K	GP covariance matrix
k	GP covariance function
k_{therm}	Toroidal thermal elliptical factor
L	Lift
l	GP length scale hyperparameter
l_p	Length of the pole
l_{therm}	Thermal radius
M	Diagonal matrix of squared exponential GP length scales
m	Number of tiles
m_c	Cart mass
m_{glider}	Glider mass
m_p	Pole mass
N	Total number of state-action observations
Q	State-action value function
$\mathcal{R}$	Reward function
R	Return
r	Reward
R_{therm}	Toroidal thermal major radius

$\mathcal{S}$	Set of states
s	State
$\mathcal{T}$	Transition function
V	Air-relative velocity
V_{bound}	GP variance volume
$\hat{w}$	GP observation projection
W	Wind energy
w_{therm}	Thermal core velocity
X	GP observation set
x	State-action observation
y	GP value function estimate
z	Altitude

Subscripts

a	GP variance volume integral lower limit
b	GP variance volume integral upper limit
i	Time step/GP training input index
j	GP training input index
m	GP training input dimension index
n	Total number of GP training input dimensions
t	Time index
u	Total number of state dimensions
v	Total number of action dimensions

Chapter 1

Introduction

1.1 Motivation

An autonomous, unpowered aerial glider learning to soar in a wind energy field presents a new manifestation of the exploration-exploitation problem. Soaring is described as energy-gaining flight whereby particular flight trajectories through a non-uniform wind field allow the glider platform to gain kinetic and/or potential energy. With no other means of energy storage, the glider must expend energy to explore the space for efficient energy-gaining trajectories whilst simultaneously balancing this against the need to exploit known trajectories to maintain sufficient energy such that it can continue to explore. Aside from this cyclic dependency between exploration and exploitation, the problem of learning to soar is particularly challenging due to the additional constraint of maintaining positive platform energy and altitude throughout the task.

1.2 Thesis Problem and Related Fields

The soaring problem can be approached in a number of ways, one method involves learning a model of the wind field and deriving appropriate control actions given

this model, another is to look purely at the energy gains associated with each glider wind-relative state-action to learn profitable policies. The former requires the agent to build a model of the wind field as well as a mapping between the glider actions at each state in the wind field and the expected energy gain. This has been the approach of much of the prior research into autonomous soaring, for example, the work by Allen (2007), Langelaan (2007), Edwards and Silverberg (2010) and Lawrance (2011). The second method bypasses the need for a wind model and attempts to directly solve the policy by observing the energy gains for each glider state-action. Furthermore, it has the additional benefit of being able to adapt to a changing wind field since the state-action space is measured relative to the wind.

By taking the second approach, this problem can be tackled using reinforcement learning (RL) techniques, which are designed to train control policies for complex tasks by observing reward signals due to state-action transitions. Indeed, the exploration-exploitation trade-off is also a factor that must be considered in many RL methods. For example, temporal difference (TD) learning algorithms rely on exploration strategies such as ε-greedy sampling to guarantee accessibility and coverage of the state-action space. In this way, they are able to build value functions that converge as each state-action in the space is observed a sufficient number of times. The value function is used to derive a reward-gaining policy across the entire state space. For on-policy learning, such as SARSA(λ), deciding when to explore the space for more information and when to exploit the current policy to gain reward is a matter of keen interest, particularly when dealing with a resource-constrained problem such as learning to soar in an unpowered glider.

Under such resourced-constrained learning problems, it is desirable to direct exploration to areas where observations will most improve the value function estimate, and to adapt the exploration behaviour according to the available resources. For example, a resource-constrained learning agent should explore more aggressively in unobserved regions of the state-action space when resource levels are high and exploit the learnt value function to replenish the resource when levels are low. Existing exploration strategies have provided exploration rewards based on confidence bounds, such as in

Lai (1987) and Brochu et al. (2010), or other ad-hoc information gain measures like those of Sutton (1990) and Schmidhuber (1991). These measures are myopic in that they only consider the information gain of the next observation; furthermore, they are combined with the state-action value in the action selection objective function in a fixed manner, that is, the influence of the exploration reward does not adapt to the current learning conditions.

1.3 Thesis Contributions

This thesis proposes a directed, adaptive and nonmyopic exploration strategy to tackle the resource-constrained exploration-exploitation task of learning to soar. There are three key contributions of this thesis that are summarised as follows:

Derivation of a new information measure

The learning algorithms presented in this thesis use a Gaussian process (GP) regression model to approximate the SARSA(λ) value function. The sum of the GP variance at each state-action across the problem space represents an uncertainty volume; furthermore, the change in this volume due to taking a new observation can be used as a measure of the information gain of that observation. This thesis derives an analytical solution to finding this variance volume for a GP that uses a squared exponential covariance function. It is also shown that an analytical solution can be found for general covariance functions that satisfy certain integrability conditions.

Development of a nonmyopic information value

The notion of an information *value* introduced in this thesis is consistent with the definition of the state-action value in RL as the discounted sum of future rewards. However, instead of dealing with state-action rewards, the information value derives from the information gain rewards of possible future state-action observations. The

proposed rollout method draws inspiration from the operation of the eligibility trace in SARSA(λ) and computes a nonmyopic information value as the discounted sum of future information gain rewards.

An exploration-exploitation management scheme for resource-constrained learning systems

An unpowered aerial glider learning to soar in a wind field is constrained by the platform energy, which dictates the available flight time of the glider. Exploitation of reward gaining state-action trajectories can increase platform energy and thereby increase flight time, while exploration of the wind field is required to first identify such reward gaining state-actions and also to continue searching for more efficient trajectories. Here the state-action value can be seen as analogous to the value of exploitation and similarly the information value represents the value of exploration. This thesis presents a method for dynamically managing exploration and exploitation behaviour by weighting the exploration value in the action selection objective function according to the current resource level.

1.4 Thesis Structure

The chapters of this thesis build sequentially towards the final exploration-exploitation learning algorithm that is applied to the resource-constrained problem of an unpowered aerial glider learning to soar in a wind field.

Chapter 2 provides a literature review on the three major research fields from which this thesis derives: autonomous soaring, RL, and information-based exploration.

Chapter 3 begins with a description of the mathematical framework for SARSA(λ) RL and outlines techniques for value function approximation. A focused analysis of GP regression modelling for value function approximation is provided along with a description of the squared exponential covariance function, which is used thoughout the thesis. This chapter also provides a discussion on an existing

sparsification method that can be used to bound the computation time of the GP update.

Chapter 4 presents two of the key contributions of this thesis that are required for defining the exploration value. The proposed GP variance volume information measure is introduced in Section 4.2.1 and is compared to other existing information measures. The rollout mechanism used to compute a nonmyopic information value is presented in Section 4.3.1 and an adaptation of this is given in Section 4.4.2 that can reduce the computational complexity of this method. A dynamic exploration weighting based on the elapsed time steps is described in Section 4.4.1 before the full (informative) iGP-SARSA(λ) algorithm is presented in Section 4.5.

Chapter 5 compares the performance of the proposed learning algorithm with existing RL algorithms on several benchmarking problems, namely: puddle world and cart pole from the 2005 NIPS RL Benchmarking workshop, and the battery cycling problem and 2D 3DOF soaring glider problem first presented in Chung et al. (2013). An analysis of the results discusses the various characteristics of the proposed iGP-SARSA(λ) algorithm and the problem instances that it is best suited to tackling.

Chapter 6 addresses the issue of resource-constrained exploration specific to the problem of an unpowered aerial glider learning to soar in a wind field. The dynamic energy and altitude-based exploration strategy presented in this chapter is incorporated into the learning algorithm to produce the (energy-weighted) eGP-SARSA(λ) algorithm for autonomous soaring in Section 6.2. A discussion on the computational complexity of this algorithm is also provided in Section 6.2.1.

Chapter 7 presents simulation results of the proposed iGP- and eGP-SARSA(λ) learning algorithms as applied to the 2D 3DOF soaring glider problem described in Section 5.4, as well as for a full 3D 6DOF glider simulator with a single thermal updraft energy source.

Chapter 8 provides conclusions about the information measure and exploration

strategies proposed in this thesis and suggests directions for future research that can extend the work that has been presented.

Chapter 2

Background

The motivating problem behind this work is the unique exploration-exploitation conundrum faced by an autonomous aerial glider learning to soar in a wind field. Soaring is defined as energy-gaining flight whereby particular flight trajectories through a wind field allow for energy capture from the wind surpassing the energy loss due to drag. Successful energy capture is particularly pertinent in the problem of autonomous soaring since the glider is an unpowered aircraft and is typically unable to store energy in any form other than as gravitational potential energy or kinetic energy. In recent times there have been studies into the feasibility of regenerative soaring with dual-role windprops, see Barnes (2006), however this thesis focuses on soaring without this capability. The soaring task that is addressed in this thesis involves learning a control policy that maps every glider state to a control action. Since there is no prior knowledge of the energy gain rewards for each possible state-action pair, extensive exploration of the state-action space is required to learn a good policy. However, in opposition to an unrestrained style of exploration is the constraint placed on the problem due to the available platform energy, which restricts the available flight time. Thus, in the problem of learning to soar, there is a tight coupling between what energy gain rewards are observed throughout the learning process and the glider's ability to continue to make new observations.

In this thesis, autonomous soaring is viewed as a policy learning problem that can

be tackled using RL techniques. Indeed, many of the early RL problems such as the bandit style problems described in Gittins (1979) and Katehakis and Veinott (1987) recognised the inherent exploration-exploitation trade-off in the problem structure and developed sampling methods that accounted for both the expected reward as well as the information that can be obtained from sampling a particular point. This work grew largely from the field of experimental design, a fine example of this is the work by Robbins (1952), which presented an early forerunner to the bandit problem. Research into exploration methodologies in RL has continued to draw inspiration from the areas of experimental design and information-based exploration.

The following sections provide some background on aerial soaring and the recent interest in autonomous soaring, particularly as an exploration-exploitation problem. Section 2.2 gives a review of how the RL community has handled the exploration-exploitation trade-off; and finally, a discussion on the informative exploration techniques used for path planning under uncertainty is given in Section 2.3.

2.1 Autonomous Soaring

2.1.1 A brief history of soaring research

Gliding and soaring flight was first observed in large birds, which were seen to remain aloft and travel long distances with minimal or no flapping of the wings. In fact, one of the first contributions towards understanding the mechanics of heavier-than-air flight was the identification of the aerodynamic forces: lift, drag and weight, by Cayley (1809, 1810a,b) after observing large birds in gliding flight.

> When large birds, that have considerable extent of wing compared with their weight, have acquired full velocity, it may frequently be observed, that they extend their wings, and without waving them, continue to skim for some time in a horizontal path. Cayley (1809)

Soaring bird flight was first noted by Peal (1880), who observed large birds gaining altitude while flying in large curves of a spiral that headed downwind.

> Firstly they rise by flapping the wings vigorously, and when up some 100 or 200 feet, *if there is a breeze*, begin to soar in large circular sweeps, rising 10 to 20 feet at each lap, the whole bird being otherwise quite motionless, and the wings extended rigidly. Peal (1880)

Although what he actually saw was most likely a form of static soaring in a thermal bubble travelling laterally due to the prevailing wind, Peal's hypothesis of this behaviour was a basic description of the energy transfers involved in dynamic wind shear soaring, albeit without consideration of the horizontal wind profile.

> I take it the explanation is, that in passing round *with* the wind, and slightly falling, great impetus is gained, which is slowed down by turning to meet and rise on the wind *like a kite*. Peal (1880)

Drawing from this, Rayleigh (1883) proposed that soaring can be achieved when either the wind is not horizontal (as in the case for static soaring) or when the horizontal wind is not uniform, that is, when there is wind shear. Expanding on the latter, he described the air relative velocity changes for a bird dropping altitude while flying leeward into a slower horizontal wind shear layer and then turning to rise windward back into the faster wind shear layer above. Figure 2.1 gives a pictorial description of dynamic soaring as described by Rayleigh (1883), this energy gaining flight trajectory was later dubbed the Rayleigh cycle and has remained the basis of our understanding of dynamic soaring.

The lead up to the 20th century saw many aviation pioneers make significant engineering contributions towards unpowered human flight. Between 1891 and 1896, the German brothers Otto and Gustav Lilienthal designed, built and flew their hang-glider-type aircraft, the Derwitzer, in over 2000 successful and well-documented gliding flights, see Lilienthal (1889). In 1896, Octave Chanute designed a biplane hang-glider based on the Lilienthals' work and this in turn inspired the Wright brothers'

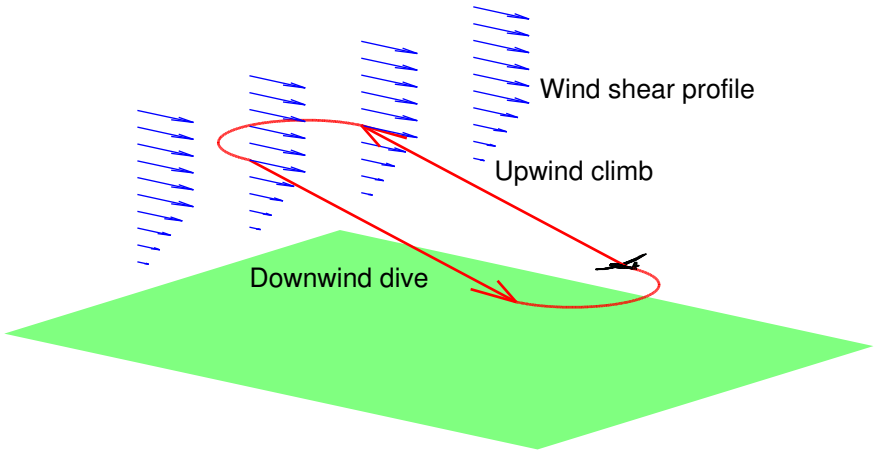


Figure 2.1: Dynamic soaring in a horizontal wind shear as described by Rayleigh (1883), airspeed is gained in both the climb and dive segments while groundspeed is maintained.

design of the Wright Glider in 1900. Three years of experimentation and the addition of a motor and propeller produced the Wright Flyer in which the Wright brothers achieved the first controlled, powered and sustained heavier-than-air human flight.

The advent of World War I saw interest in soaring flight wane in favour of research and development into powered flight; however, after the war, the Treaty of Versailles placed restrictions on the manufacture and use of powered aircraft in Germany, resulting in a renewed interest in soaring that spread throughout Germany, and later, Europe. Indeed, as described in Shenstone and Scott-Hall (1935), German sailplane engineering and soaring technology had made remarkable advances all throughout the 1920s. Most notably, in 1926, the possibility of thermal updraft soaring was discovered, largely by accident, when German sailplane pilot, Max Kegel, was swept up into a thunderstorm that carried him (safely) to a distance of 54 km; the previous distance record had been less than half of that since pilots had relied on updrafts created by local hills to perform ridge soaring and this geographically restricted their flight zones.

The following years of soaring and glider platform research were driven by thermal soaring. Platform designs underwent heavy changes to accommodate the manoeuvrability required in thermal soaring and new on board instruments, such as the variometer, were invented to help detect when the glider was in a thermal. Continued

research into the thermalling behaviour of birds during migration, as in Mackintosh (1949), provided proof-of-concept that even greater distances in cross-country soaring could be achieved via thermal hopping, while the work by Cone (1962) studied the formation and structure of thermal updrafts and began to formalise the mechanics of thermal soaring. Research into dynamic soaring also experienced something of a renaissance during the mid-latter half of the 20th century when the flight of the albatross recaptured the attention of ornithologists and aerodynamicists alike, as shown in the work by Cone (1964), Wood (1972) and Weimerskirch et al. (2000).

Following the development of remote controlled (RC) unmanned gliders and the establishment of RC cross-country soaring as a competitive hobby-sport, recent years have seen the focus turn to robotic unmanned aerial vehicles (UAVs) and the prospect of autonomous soaring for long endurance aerial missions. The following subsections describe the recent research into the methods of autonomous soaring and the unique exploration-exploitation trade-off encountered in this problem.

2.1.2 Methods of autonomous soaring

The idea of autonomous soaring is attributed to Wharington (1998), who applied RL to the design of adaptive control systems for soaring aircraft. The approach taken in this seminal work was to investigate the feasibility of various techniques (dual adaptive heuristic critic, neural networks and Q-learning) for learning the optimal pitch, bank and speed control of a glider. In terms of its optimality measures, this work drew on conventions established by manned and RC soaring communities, which for the above cases were defined as maximum range/endurance, maximum vertical velocity in thermalling, and fastest transit in a sequence of climb-cruise segments using thermals (speed-to-fly, MacCready (1958)), respectively. Approaching autonomous soaring as a control problem was a natural extension of the existing literature on optimal soaring techniques for glider pilots and the major contributions of Wharington (1998, 2004) were in implementing adaptive algorithms to refine the rules-of-thumb given by MacCready (1958) and Metzger and Hedrick (1975) for autonomous UAVs.

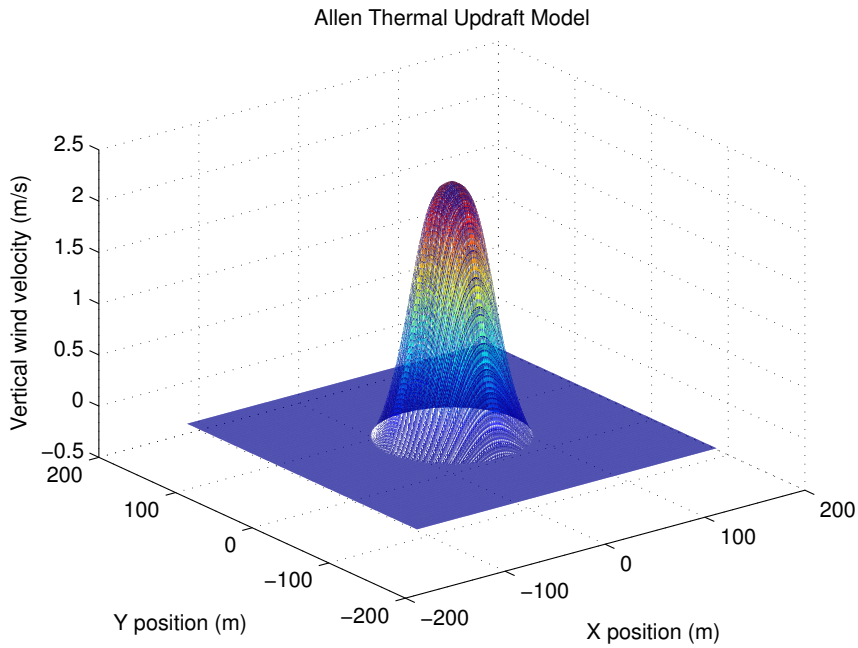


Figure 2.2: The thermal model introduced by Allen (2006) models the vertical velocity of each horizontal cross section of a thermal with a Gaussian function.

In addition to the problem of optimal soaring control for UAVs, researchers became interested in how to better model and estimate wind features such as wind shear and thermal updrafts. Early studies made by Allen (2005) modelled thermal updraft characteristics given surface radiation and weather balloon data taken at Desert Rock, Nevada, and showed the feasibility of thermal soaring for extending flight duration for small UAVs. Furthermore, the Allen (2006) Gaussian model shown in Figure 2.2 became a standard model for simulating the vertical velocity of thermal updrafts in much of the autonomous soaring research that followed.

Real world implementations of autonomous soaring were also achieved during this time. The first successful autonomous soaring flights were a realisation of the work by Allen (2005, 2006) and are reported in Allen (2007). These flights were a demonstration of the guidance algorithms used for thermal detection and the control algorithms used to exploit thermals for altitude gain. As an improvement to the thermal localisation used in Allen (2007), an adaptive grid search method was applied in Edwards (2008) to refine the thermal centre estimation. The algorithm's capabilities were demonstrated in a cross-country soaring competition in which it competed

against RC gliders; the autonomous glider placed 3rd overall, flying a course distance of 63.4 km in 3.5 hrs as reported in Edwards and Silverberg (2010).

Langelaan (2007) continued with the development of complete autonomous soaring systems by combining estimation and prediction of the wind field with trajectory planning, decision making and low-level flight control. The point mass model glider was shown in simulation to successfully utilise ridge soaring to travel from a start point to a goal point that would, without the aid of soaring, be impossible to reach. A similar demonstration was given in Lawrance and Sukkarieh (2009); the strip-method 6DOF flight simulation used in this work was able to model the wind effects on the scale of the aircraft. The algorithm combined wind estimation and control for dynamic soaring and was able to perform "energy-neutral trajectories" to reach specified goal points.

One of the most recent steps in UAV soaring research has been to introduce methods of handling stochasticity in the wind field. Examples include decaying the expected power gain across the estimated wind field according to the short term and long term memory modes defined in Bower et al. (2010), and the use of GP regression in Lawrance and Sukkarieh (2010, 2011b) to model static or dynamic wind fields, producing not only an estimate of the wind but also computing the uncertainty associated with the estimate. At the same time, more sophisticated thermal models have been introduced that model not only the vertical wind profile, but also the lateral wind velocities and disturbances due to turbulence as surveyed in Bencatel et al. (2013). 3D wind turbulence was introduced in the thermal model used by Ákos et al. (2010), while the toroidal thermal model presented by Lawrance (2011) and shown in Figure 2.3 is one such model that describes the full 3D wind field of a thermal bubble.

2.1.3 The exploration-exploitation trade-off

Flight trajectories for wind field sampling and mapping had been suggested as early as Allen (2005), however the exploration-exploitation trade-off in the task of autonomous soaring was only first recognised and addressed in Lawrance and Sukkarieh (2010,

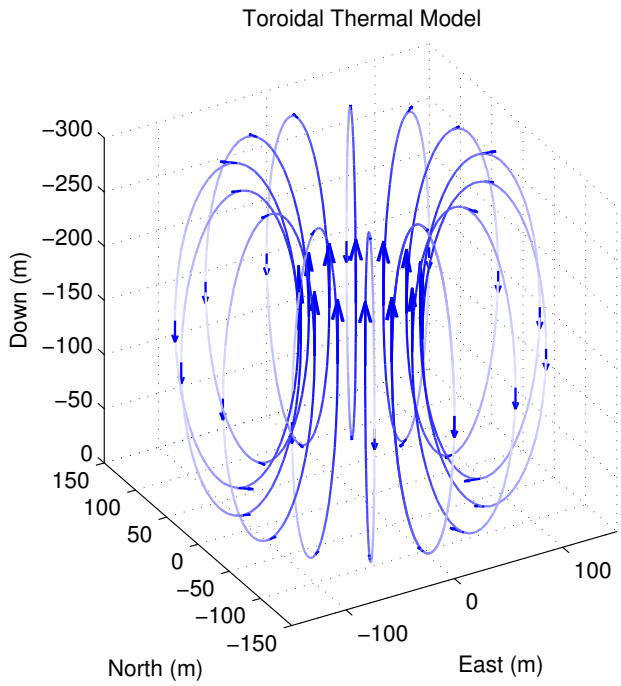


Figure 2.3: The toroidal thermal model introduced by Lawrance (2011) considers both the vertical wind velocity across the thermal as well as the lateral wind velocities, ensuring that the overall volumetric air flow of the thermal bubble is zero. The vertical wind profile at each horizontal cross-section of the thermal is that of a sinusoid.

2011a). The problem was broken down thus: at any decision instance, the autonomous glider had to choose between (a) gathering more wind data to reduce the overall uncertainty of the wind field model or (b) visiting a location of high expected energy gain to boost platform energy and extend its flight duration. The difficulty arises in the fact that in order to continue performing (a), the glider must successfully gain energy from the wind via (b), and to efficiently gain energy from the wind via (b), the glider must perform (a) to improve its map of the wind field.

In Lawrance and Sukkarieh (2011a) the exploration-exploitation trade-off was handled on both the target assignment level and the path planning level. The exploration goal state was defined as the location of maximum map uncertainty, as measured from the GP covariance, and included a minimum energy requirement for the glider at that position. The exploitation goal point was defined as the location of maximum recorded power gain. The exploration goal state was assigned as the global target as long as the glider was determined to have sufficient energy to reach the exploration goal, otherwise

the glider was assigned to the exploitation goal. For travel to the global target, at each decision instance, the path planning algorithm selected a trajectory segment from a pool of nominal paths according to a reward function that weighted the optimistic energy gain, distance to goal, and reduction in map uncertainty expected for each possible path.

The algorithm presented in Lawrance and Sukkarieh (2011a) directs an unpowered aircraft to simultaneously map and use the wind field for soaring with a control strategy that is dependent on the computed reward over a defined time horizon. The time horizon can become a limitation in this approach since trajectories that generate higher rewards in the future may be overlooked if they provide low reward within the planning horizon. If a model of the wind field is only required for computing the expected rewards when selecting the next set of actions, then an alternative method can be to model the utility of each state-action pair directly from the observed energy rewards. This produces a value function that maps the glider states and actions to an expected return, which represents the discounted sum of all future rewards. In this way, the value function describes the policy across the entire state space, and actions chosen according to this policy do not suffer from the myopia that is introduced by a defined planning horizon. The following section investigates how such value functions have been used in RL to deal with the exploration-exploitation trade-off.

2.2 Exploration and Exploitation as a Reinforcement Learning Problem

"Reinforcement learning is about learning from interaction how to behave in order to achieve a goal," Sutton and Barto (1998); it can be defined as a Markov decision process (MDP) consisting of:

- a set of states $\mathcal{S}$
- a set of actions $\mathcal{A}$

- a (potentially unknown) state transition function $\mathcal{T} : \mathcal{S}, \mathcal{A} \longrightarrow \mathcal{S}'$, which can be stochastic, and
- a hidden reward function $\mathcal{R} : \mathcal{S}, \mathcal{A}, \mathcal{S}' \longrightarrow \Re$.

The RL *agent* must interact with its *environment* over a sequence of discrete time steps to observe *reward* signals for each *state* and *action*. The reward function is an individual state-action level representation of the overall goal while the *return* is a function of future rewards, conventionally the discounted sum of future rewards. The reward signals are used as the basis for evaluating the expected return for each state-action and this is stored as a *value function* from which the *policy* is derived. The optimal value function assigns to each state-action the largest expected return achievable by any policy, and an optimal policy is greedy with respect to the optimal value function.

Since the purpose of any RL problem is to learn a policy/the optimal policy for achieving some goal, an integral part of the problem lies in estimating the value function. Each state-action that is explored provides information to aid estimation while the estimated value function itself indicates state-action trajectories that are likely to earn high long term rewards. The exploration-exploitation trade-off is apparent in the decision of whether to proceed along a known trajectory that has reasonable expected returns or explore new state-actions in the hopes of finding more efficient/profitable trajectories. The following subsections are an investigation into the evolution of the exploration-exploitation trade-off from its beginning in the multi-armed bandit problem to our current understanding of its role in RL. Supplementary to the following discussion, readers are directed to the survey of early RL research by Kaelbling et al. (1996), the required reading textbook by Sutton and Barto (1998), the benchmarking efforts of Littman et al. (2005), and a recent survey of RL research in robotics by Kober et al. (2013).

2.2.1 Sequential design and the bandit problem

The sequential design of experiments is an adequately named method of using prior observations for choosing what action to take next, that is, where to take future samples to achieve maximum cumulative reward. As described by Robbins (1952), "the size and composition of the samples are not fixed in advance but are functions of the observations themselves". Robbins (1952) describes the problem of how to draw samples $x_1, x_2, \ldots, x_n$ from two unknown populations such that one obtains the greatest possible expected value of the sum $S_n = x_1 + \ldots + x_n$. The problem described encapsulates the exploration-exploitation problem at the heart of all RL problems; on the one hand, the experimenter/agent can choose to greedily sample from the population that it currently believes to produce the highest values, while on the other, it can choose to sample the population with the highest uncertainty to better characterise the population distribution such that it can make a more informed decision in future samples. This problem was later dubbed the multi-armed bandit problem due to analogies with a gambler choosing the next slot machine (one-armed bandit) to gamble on given its current knowledge of the reward distributions of each machine.

Apart from defining the problem of sequential design for experiments, two other concepts were raised in Robbins (1952). One was the problem of optional stopping, that is, defining when to stop sampling from a population since some hypothesis of its distribution has become satisfied within some quantifiable bounds, and the other was the notion of *regret* as a measure of asymptotic loss per sample "due to ignorance of the true state of affairs". These ideas are present in the dynamic allocation index (DAI), or Gittins index, proposed by Gittins and Jones (1974) and Gittins (1979), which was used for characterising the expected returns from sampling a population.

> Given a bandit process, the DAI is intuitively that value λ which makes one indifferent between accepting an immediate reward of λ and optimally stopping the bandit process with a residual reward of λ discounted by a^t if stopping occurs at time t. Gittins (1979)

This "discounted sum of future rewards" method is in many ways similar to the sequential analysis used in dynamic programming (DP) for solving stochastic optimal control problems, Bellman (1952).

Bellman (1956) first applied DP to a modified version of the bandit problem where, of the two possible populations to sample, one had known probabilities of return while the other had only *a priori* information on the distribution of the probabilities. It was shown that by making successive approximations to the expected return after each observation, the algorithm converged to the expected returns of the optimal policy. Furthermore, it was suggested that instead of applying value iteration, as was used in this work, it may be more intuitive to apply policy iteration to solving this problem. Indeed the notion of value functions and their application in RL derives mainly from their use in DP.

A point of difference between DP and other forms of RL is that DP assumes full knowledge of the underlying MDP, which allows updates of the value function to be performed in sweeps over the entire state space at each time step. However, because of this, DP suffers from what Bellman (1957) called "the curse of dimensionality", where the computational requirements increase exponentially as the dimensionality of the problem increases. One method used to address this is prioritised sweeping, proposed by Moore and Atkeson (1993), which forms a queue of states to update in the next sweep and the size of the sweep is determined by a computation budget. Priority in the queue is given to those transitions that have the highest Bellman error, which measures the difference between the predicted and actual outcomes. While unlike the explicit exploration of model-free RL, prioritised sweeping is another form of the exploration-exploitation trade-off; the goal is to learn the optimal control strategy, however, given the computation budget, only a limited number of updates can be performed to add to the overall pool of information used to improve the policy. Defining a metric that prioritises the most "unexpected" state-action observations allows the incorporation of the most useful updates in the sweep, thereby improving the rate of learning.

2.2.2 Temporal difference reinforcement learning

In contrast to the previous learning techniques that had assigned credit (reinforcement) according to the difference between predicted and actual outcome, TD learning proposed to assign credit according to differences between temporally successive predictions. The first use of TD methods for machine learning was in the checker-playing programme by Samuel (1959). The programme used a scoring function to evaluate board positions, with each score representing the predicted outcome of the game starting from that position. Thus, the parameters of the scoring function were updated by the difference between the evaluations of each pair of successive positions occurring in a game. In terms of exploration, the program applied a lookahead method of variable depth tree search where search depths were limited by certain board conditions. These conditions were specific to the game of checkers and allowed "greater surveillance of those paths which [offered] better opportunities for gaining or losing an advantage," Samuel (1959).

The first comprehensive study of TD methods was by Sutton (1988), this work showed the convergence and optimality of the linear TD(0) solution and also outlined the generalised TD(λ) method that uses eligibility traces. Convergence proofs for TD(λ) came later in Dayan (1992). These proofs relied on exploration of the state-action space as induced by any distribution of starting probabilities over the entire space *such that there are no inaccessible states*. This requirement meant that very simple exploration strategies, such as ε-greedy, softmax or randomised/biased intial value function estimation, could be applied while still guaranteeing convergence. Although the role of exploration in RL had received considerable attention by Lai (1987), Sutton (1990), Schmidhuber (1991) and Thrun (1992a,b), much of the theoretical research following this was uninterested in investigating more sophisticated exploration strategies. Instead, effort was directed to understanding the effects of value function approximation as in Baird (1995), Boyan and Moore (1995) and Sutton (1996), with convergence proofs for linear function approximators given in Tsitsiklis and Van Roy (1997); there was also considerable analysis on the effects of the step size and discount parameters to the learning rate as by Sutton and Singh (1994) and Singh and Sutton (1996).

A notable implementation of TD(λ) is the Tesauro (1995) TD-gammon programme, which used value function approximation with initially random weights and a simple greedy sampling strategy to learn backgammon strategies from self-play.

Other research at the time also looked at the possibility of using TD methods for learning control policies. Watkins (1989) developed Q-learning as an off-policy strategy for learning optimal control. By extension from the TD(λ) convergence, it was shown that Q-learning is *exploration insensitive*, that is, regardless of the sampling strategy used during learning, the algorithm is guaranteed to converge to the optimal policy as long as each state-action pair is observed often enough. In Q-learning, the TD of the value function update is computed between the value of the previous state-action and the next maximal state-action value,

$$Q_{t+1}\left(s_t, a_t\right) = Q_t\left(s_t, a_t\right) + \alpha\left[r_{t+1} + \gamma \max_{a \in \mathcal{A}(s_{t+1})} Q_t\left(s_{t+1}, a\right) - Q_t\left(s_t, a_t\right)\right]. \tag{2.1}$$

Since Equation (2.1) does not rely on the next actual state-action sampled according to the sampling policy, Q-learning is considered an *off-policy* learning technique. An *on-policy* version of Q-learning called *Modified Connectionist Q-learning* was proposed by Rummery and Niranjan (1994) and was later named SARSA after the algorithm update procedure (State-Action-Reward-State-Action). The TD used in the SARSA update is computed between the value of the previous state-action and the value of the next state-action as determined by the sampling policy,

$$Q_{t+1}\left(s_t, a_t\right) = Q_t\left(s_t, a_t\right) + \alpha\left[r_{t+1} + \gamma Q_t\left(s_{t+1}, a_{t+1}\right) - Q_t\left(s_t, a_t\right)\right]. \tag{2.2}$$

Comparison of Equations (2.1) and (2.2) shows that if the greedy action is taken at each time step, the Q-learning update is identical to the SARSA update.

Conventionally the sampling policy for either TD learning, Q-learning or SARSA derives mainly from the value function estimate with some added randomness to ensure ergodicity, however explicit exploration terms can be incorporated into the sampling policy to generate specific exploration behaviours. The following section reviews existing methods for actively rewarding exploration that reduces the uncertainty of the

value function.

2.2.3 Exploration and active learning

The concept of defining some form of exploration reward as part of the sampling policy is not a new development and has its roots in stochastic control and dual control. Lai (1987) defined a class of upper confidence bounds (UCBs) for adaptive exploration in the multi-armed bandit framework, while Sutton (1990) devised an "exploration bonus" that was based on an ad-hoc "number of steps [since last visit]" measure. Schmidhuber (1991) maintained a separate model of the reliability of the predictions generated by the Q-learning algorithm and defined a curiosity goal as maximising the changes in prediction reliability, performing what he termed "adaptive curiosity".

The turn of the millennia was marked by the popularisation of GPs for classification and regression in the machine learning community as seen in the work by Williams (1998), Neal (1998), Csató and Opper (2002) and Rasmussen and Williams (2005). For RL research in particular, the effects of this trend were largely seen in the area of value function approximation, for TD learning by Engel et al. (2003, 2005) and Engel (2005), as well as DP by Rasmussen and Kuss (2004), Deisenroth et al. (2009) and Deisenroth (2010). Apart from its use as a function approximator, the GP additionally provides an uncertainty measure in the form of a covariance; and as noted by Engel et al. (2003), this uncertainty can be used to direct exploration of the value function space. This sparked a new wave of enthusiasm for investigating efficient exploration strategies in RL.

Dearden et al. (1998) formally defined exploration in RL using the classical description of *value of information*—"the expected improvement in future decision quality that might arise from the information acquired by exploration". This methodology for defining exploration value requires assessment of the agent's uncertainty of the value function; thus, the use of GPs for estimating the value function provided an ideal setup for characterising exploration reward as the uncertainty reduction (information gain) associated with the value function approximation. Recent implementations of

exploration in RL have applied the GP covariance to compute UCBs for bounding regret in Srinivas et al. (2010), or as part of various information measures to form acquisition functions by Brochu et al. (2010). In fact, the latter draws heavily from the field of experimental design, Santner et al. (2003), performing a satisfying loop closure with the exploration-exploitation ideas that originally characterised RL problems.

Some of the latest developments in RL exploration have been concerned with safety considerations, this has been due in part to the increased number of robotic applications of RL in recent years as surveyed in Kober et al. (2013). There have been a number of approaches to devising safe exploration strategies in RL, broadly, these can be grouped under three categories: explicitly adjusting the risk of exploration in the action selection objective function, sampling from a subset of policies that have been deemed "safe", and executing predefined "safe return" policies when necessary. Of the first category, Schneider (1996) proposed an objective function consisting of deterministic, cautionary and probing terms. The deterministic term computed state-action values assuming a perfect model, the cautionary term considered risk associated with uncertainty in the model, while the probing term favoured potentially suboptimal and/or risky controls that improved the model. A similar approach was taken by Gehring and Precup (2013) where the exploration bonus was computed as the negative controllability, or mean absolute TD error, of a state. Related to the second category is the work by Bagnell (2004) that was able to provide robustness guarantees by learning policies that performed well over all possible transition matrices (given some uncertainty over the transition model). In terms of defining "safe" policies, the baseline behaviour of García and Fernández (2012) allowed for transitions to known safe states and also accounted for user-controlled Gaussian noise to the transition model by which the agent performed exploration. Another example is the set of "δ-safe" policies, proposed by Moldovan and Abbeel (2012), which are defined as those policies that have a (greater than or equal to) δ probability of executing a return policy that can safely control the agent back to a predefined home state. The algorithm by Moldovan and Abbeel (2012) overlaps the third type of safe exploration strategy and is similar to the "backup policy" of Hans et al. (2008), which also falls under this

category.

2.3 Information-Based Exploration

Finally, this chapter cannot be concluded without taking a brief look at the methods of information-based exploration that have appeared in the robotic exploration literature. Robotic exploration in its purest form can be considered as a sensor coverage/placement problem, such as that defined by Hoffmann and Tomlin (2010), for generating a map or model of an unknown area or phenomenon; however, the choice of map representation along with imperfections in the sensor model and knowledge of the vehicle pose are all confounding factors that contribute to uncertainties in the observations and the *a posteriori* model that is built. Robotic exploration draws inspiration from well-established information theory to devise objective functions for guiding exploration, and recent efforts have focused on active and adaptive exploration with recognition given to the exploration-exploitation trade-off that also exists in some robotic systems.

2.3.1 Existing information measures

Many robotic mapping/modelling applications have favoured the use of entropy for defining the information gain of taking particular observations, see Stachniss et al. (2005), Low et al. (2009) and Amigoni and Caglioti (2010), while the closely related mutual information has been used in work by Krause and Guestrin (2007), Singh et al. (2009a,b, 2010) and Bender et al. (2013). Another example is the information gain objective function used in Binney et al. (2013), which compares the average reduction in variance of the GP model for potential sampling points. The algorithm computes this as the normalised difference between the trace of the prior and posterior covariance matrices given the potential observations. In fact, using the trace of the covariance matrix draws from the A-optimality criterion in optimal design as described by Fedorov (1972), while the D-optimality criterion, which uses the determinant of

the covariance matrix, is proportional to the entropy. Although other estimation techniques such as particle filters have been used successfully in such modelling problems, GP modelling features prominently in much of this work and its popularity is again attributed to the GP covariance matrix, which provides a convenient basis for measuring the change in uncertainty over the estimated model.

2.3.2 Adaptive exploration

A recent thrust of robotic exploration research has been to analyse the benefits of performing adaptive online planning instead of using pre-planned paths. This work has been driven by the high computational costs associated with adaptive planning and is centred around the submodularity property of some information measures such as mutual information. Submodularity is the property of *diminishing returns* and was first defined by Nemhauser et al. (1978); in terms of informative exploration, it describes the reduction in utility of taking a particular observation as the observed set grows.

Adaptive planning selects the next sample point, or set of sample points, accounting for all other observations taken up to the current decision instance, whereas non-adaptive planning chooses the full set of observations to make and does not update or alter the plan regardless of what new information arrives. Singh et al. (2009b) defines the *adaptivity gap* as "the ratio of the performance of the optimal [adaptive] policy divided by the performance of the best nonadaptive path" according to some utility function. By providing theoretical bounds on the adaptivity gap, Singh et al. (2009b) showed that their nonmyopic adaptive informative path planning algorithm, which can use any near-optimal nonadaptive algorithm as a subroutine, performs within a competitive bound of the optimal adaptive solution. Hollinger et al. (2013) proved a similar result for a constrained exploration problem, quantifying the adaptivity gap for a ship hull inspection problem with a monotone submodular objective function (the predicted variance reduction of a GP) with an additional constraint on the total observation cost. Computationally efficient adaptive path planning for robotic

exploration remains a challenging research problem.

2.3.3 Exploration-exploitation trade-offs in robotics

The exploration-exploitation trade-off also exists in robotic exploration problems, generally deriving from the need to complete a mapping or modelling task whilst also minimising model uncertainties that can arise from various sources. The decision-theoretic framework proposed by Stachniss et al. (2005) simultaneously considers both the uncertainty in the map as well as in the pose of the vehicle, essentially trading off paths that present loop closure opportunities with paths that can traverse more of the unexplored region. Drawing from Bayesian optimisation and experimental design, Marchant and Ramos (2012) proposed a new acquisition function called the "Distance-based UCB" for monitoring abnormalities/extreme values in environmental phenomena such as air pollution. The acquisition function trades off sensing in regions that are likely to have extreme values with regions that have been observed only sparsely, whilst simultaneously weighing this potential information gain (computed by the GP covariance) against the distance to waypoint so as to promote energy efficiency in exploration. In Krause and Guestrin (2007) and Hoang et al. (2014), the exploitation goal is to accurately model some spatio-temporal phenomenon using a GP, while the exploration goal is to decrease the hyperparameter uncertainty of that GP. The problem described in this work involves an interesting coupling of the exploration and exploitation goals: the exploitation goal point is determined from the estimated uncertainty in the current GP model, however poorly chosen GP hyperparameters will produce an inaccurate representation of the uncertainty. This is similar to the exploration-exploitation relationship in the autonomous soaring problem where exploitation of the wind energy field comes at the cost of learning a more accurate mapping between state-action trajectories and their expected energy rewards.

2.4 Summary

This chapter introduced the three main research areas from which this thesis is built: the unique exploration-exploitation relationship in autonomous soaring, the application of RL to solving exploration-exploitation problems, and the information-based exploration mechanisms that can be used for nonmyopic and adaptive exploration of the RL state-action value function. The discussion presented here provided some background on the historical development of the soaring and RL problems, and also looked at the state-of-the-art in all three research areas. The following chapters describe the methodology used to combine these ideas into a new approach for solving the resource-constrained exploration-exploitation problem of autonomous soaring.

Chapter 3

Gaussian Processes in Reinforcement Learning

The algorithms presented in this thesis build upon the basic on-policy TD RL algorithm, SARSA(λ). This chapter describes the SARSA(λ) algorithm framework and compares two value function approximation techniques, tile coding (TC) and GP regression modelling, for extending the learning algorithm to handle continuous state-action spaces. Furthermore, given the computational load of computing the GP approximation, a sparsification method based on Csató (2002) is also described in the following subsections.

RL is systematic trial-and-error learning through interaction with the world, Sutton and Barto (1998) describe it as a combination of *search* and *memory*, indeed, what separates RL from naïve "guess and check" methods is the formal way it selects actions in each situation and the unique way it connects actions to the situations in which they work best. In an RL problem, the learning agent can sense its current state in the world, perform actions to change its state and observe a reward when transitioning between states. The agent gains "experience" by performing state-action transitions and recording the observed rewards in a value function through an update step known as a *backup*. The value function can then be used to predict future transition rewards and consequently can be used to learn reward-gaining policies.

There are three subclasses within RL: Monte Carlo (MC) methods, DP and TD learning, these differ primarily according to when and how the backup operation is performed on the value function. Broadly, MC methods compute the value function as the average of the returns over a set of episodes (updated at the end of each episode), while DP and TD learning are both bootstrapping methods, that is, they perform updates after each step. DP uses the Bellman optimality equation to backup the value function, and TD methods update the value function using the difference between the predicted value at successive steps. TD learning can also be extended to include eligibility traces which manage the sequence of visited state-actions and assign credit from the current transition back along the history. In a way, TD learning with eligibility traces bridges the divide between DP and MC methods by incorporating both the one-step backup of DP and the long term averaging of rewards over entire episodes as in MC methods.

The remainder of this chapter is focused on learning control policies using on-policy TD learning with replacing eligibility traces. The original SARSA(λ) algorithm is described in the next section, followed by an explanation of how GP regression can be used for value function approximation in the algorithm framework.

3.1 SARSA(λ)

SARSA(λ) is an on-policy TD control method first introduced in Rummery and Niranjan (1994) under the name *Modified Connectionist Q-Learning* or MCQ-L. Its current name arises from the algorithm procedure where the learning agent begins in a state s, performs an action a, observes the transition reward r and next state s', and then selects the next action a' according to its learnt policy. The λ refers to the use of eligibility traces to assign discounted credit along the trajectory history as new transition rewards are observed. The state-action value update equation for SARSA(λ) is,

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a), \tag{3.1}$$

where the TD error δ_t is given by the difference between the observed transition reward r_{t+1} and the expected reward $Q_t(s_t, a_t) - \gamma Q_t(s_{t+1}, a_{t+1})$:

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t). \tag{3.2}$$

The eligibility trace is updated by either the method of *accumulating traces* or the method of *replacing traces*. The method of replacing traces was shown by Singh and Sutton (1996) to produce a significant improvement in the learning rate over the method of accumulating traces and is used in the following investigation. The update is defined as:

$$e_t(s, a) = \begin{cases} 1 & \text{if } s = s_t \text{ and } a = a_t; \\ \gamma\lambda e_{t-1}(s, a) & \text{otherwise} \end{cases} \quad \forall s, a. \tag{3.3}$$

The step size, $0 < \alpha < 1$, weights how far in the direction of the TD error to shift the current state-action value estimate and can be varied over the learning steps. For convergence of the value function, the following conditions must hold for the step size sequence:

$$\sum_{k=1}^{\infty} \alpha_k(a) = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \alpha_k^2(a) < \infty, \tag{3.4}$$

where $\alpha_k(a)$ is the step size used after selecting action a for the k-th time.

The discount rate, $0 \leq \gamma \leq 1$, determines the contribution of future rewards to the current return value,

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \ldots . \tag{3.5}$$

The corrected n-step truncated return sums the first $n-1$ discounted rewards and then approximates the rest of the series using the estimated value of the nth state-action,

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{n-1} r_{t+n} + \gamma^n Q_t(s_{t+n}, a_{t+n}). \tag{3.6}$$

Values of γ close to zero promote myopic reward gaining behaviour, while values of γ approaching 1 give future rewards a strong influence. In most non-trivial problems

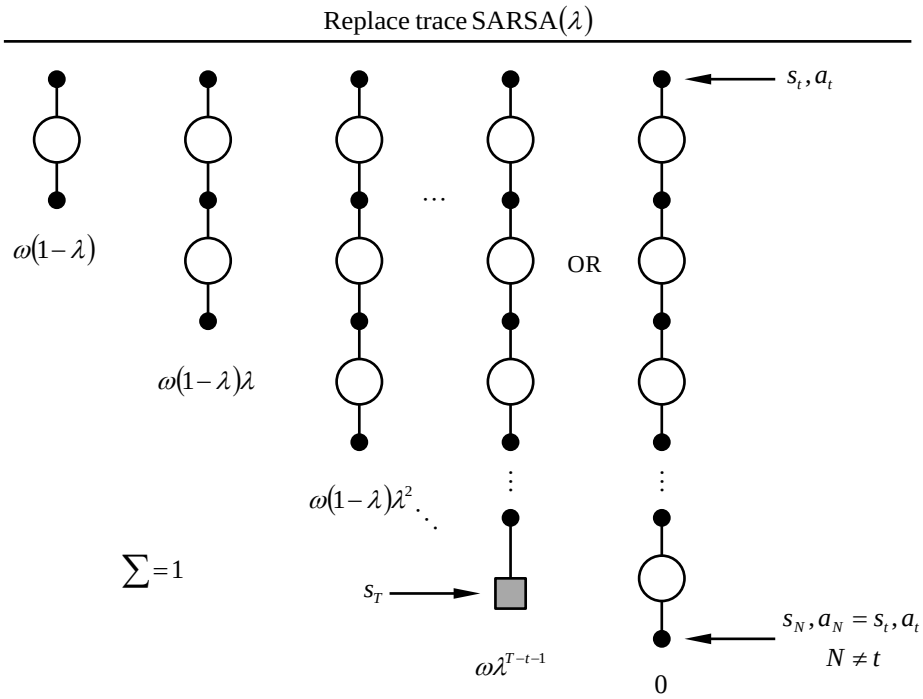


Figure 3.1: The backup diagram for SARSA(λ) with replacing traces. The backup extends from the terminal state of the episode at time $t = T$, represented by the grey square. No additional credit is assigned to revisited states within the episode. The $\omega\,(1-\lambda)$ term ensures that all the weights sum to 1.

there exist multiple local maxima such that performing actions to gain immediate reward can prevent gaining larger rewards in the future. In such situations, larger values of γ should be preferred to encourage nonmyopic behaviour.

The eligibility trace discount factor $0 \leq \lambda \leq 1$ determines how much of the current return to assign back along the state-action history. The backup diagram for SARSA(λ) is shown in Figure 3.1. In contrast to SARSA(λ) with accumulating traces, the weights given to each reward backup include a normalisation factor of $\omega\,(1-\lambda)$ to ensure that they sum to 1. The additional ω is required since the contribution of prior visits to a state-action are zeroed. Given a trajectory of state-actions where revisits occured at times $t = \{N_1, \ldots, N_m\}$, consider the sum of the weights prior to

normalisation:

$$\begin{aligned}\text{sum of weights} &= \sum_{n=1}^{\infty}\left(\lambda^{n-1}\right) - \sum_{n=1}^{m}\left(\lambda^{N_n-t-1}\right) \\ &= \frac{1}{1-\lambda} - \sum_{n=1}^{m}\lambda^{N_n-t-1}. \end{aligned} \tag{3.7}$$

For the weights to sum to 1, this requires

$$\begin{aligned}\text{normalisation factor} &= \frac{1}{\frac{1}{1-\lambda} - \sum_{n=1}^{m}\lambda^{N_n-t-1}} \\ &= \frac{1-\lambda}{1-(1-\lambda)\sum_{n=1}^{m}\lambda^{N_n-t-1}}, \end{aligned} \tag{3.8}$$

such that

$$\omega = \frac{1}{1-(1-\lambda)\sum_{n=1}^{m}\lambda^{N_n-t-1}}. \tag{3.9}$$

This gives rise to the definition of the λ-*return* as:

$$R_t^{\lambda} = \omega\,(1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}R_t^{(n)} - \omega\,(1-\lambda)\sum_{n=1}^{m}\lambda^{N_n-t-1}R_t^{(N_n)}. \tag{3.10}$$

If a terminal state exists, then all returns from that state onwards are equal to R_t and so this can be separated from the sum to give,

$$R_t^{\lambda} = \omega\,(1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}R_t^{(n)} + \omega\lambda^{T-t-1}R_t - \omega\,(1-\lambda)\sum_{n=1}^{m}\lambda^{N_n-t-1}R_t^{(N_n)}. \tag{3.11}$$

From Equation (3.11) it can be seen that when $\lambda = 1$, only the return from the terminal state remains such that all state-actions in the history are updated by this same amount; this update is equivalent to the MC backup. When $\lambda = 0$, only the most recent state-action is updated by the full return value, which is equivalent to one-step SARSA. Values of λ approaching 1 mean that the proportion of the current return passed along the trajectory decays more slowly.

There are noticeable parallels between the functions of the γ and λ discount values. Both are used to tune the "far-sightedness" of the learning algorithm. In the case of

γ discounting, the focus is on future state-action transition *rewards*; λ discounting relates to future *returns*, which in themselves encapsulate the sum of future rewards. Thus, the combination of these two discount factors allows more efficient reward tracking along entire trajectories, which can lead to faster convergence rates as shown in Sutton and Singh (1994) and Singh and Sutton (1996).

The motivation for using a SARSA(λ) framework in this investigation is its on-policy learning characteristic, which allows the trace to develop over the full state-action trajectory. Credit assignment continuity is lost whenever the trace is reset, discarding relevant information and essentially treating one continuous trajectory as two (or more) separate trajectories. In off-policy methods such as Q-learning, developed by Watkins (1989), the trace is reset whenever an exploratory action is taken. Therefore, despite the physical connection, credit received after an exploratory action cannot be assigned back to state-actions visited before that action. Since the number of exploratory actions depends on the sampling policy, in cases where they occur frequently, the effect of the eligibility trace is diminished. In another Q-learning method, developed by Peng and Williams (1996), the trace is not reset; however, regardless of the action taken, the most recent transition reward is taken to be the maximum reward achievable by the available action set. This algorithm uses a hybrid of on- and off-policy updates and as a result converges neither to the state-action values of the current policy, Q^π, or to the optimal policy, Q^*. As opposed to these off-policy methods, SARSA(λ) takes full advantage of the eligibility trace, allowing it to form over the entire trajectory of an episode without needing to reset the trace whenever an exploratory action is taken. This is most relevant in problems where state-actions along a trajectory are highly correlated, that is, they cannot easily be reached from any other state-action, and learning is expected to occur over long trajectories. Other aspects in the choice between on- and off-policy learning also exist and are discussed in Chowdhary et al. (2014).

3.2 Value Function Approximation

The value function, $Q(s,a)$, encapsulates the expected discounted sum of all future rewards leading out from each state-action transition. In SARSA(λ), the value function approximates Q^{π}, the state-action values of the current policy, π. The policy can then be gradually improved via exploratory methods such as ε-greedy sampling, and can be shown to converge (under certain boundary conditions listed in Section 3.1) to the optimal policy as the number of times each state-action is observed goes to infinity.

From the backup shown in Equation (3.1) it can be seen that SARSA(λ), and in fact RL in general, relies on repeat observations of the reward at each state-action location to learn the value function. This may be achievable in discrete (tabular) state-action learning cases where all state-actions are reachable within a finite number of learning steps; however in problems with either continuous states and/or continuous actions, the learning agent will almost surely never revisit any particular state-action and thus cannot take advantage of any experience gained along its trajectory. The algorithm is unable to extrapolate the value of state-action pairs that have not yet been visited; thus even when dealing with purely discrete spaces, a problem can be computationally infeasible if its state-action space is prohibitively large.

To extend RL to problems with continuous state-action spaces, methods for approximating the value function must be employed. While any function approximation technique can be used within the RL framework to model the value function, it is desirable to choose a method that is best able to generalise the available observations to the full state-action space. Linear methods, which are a subclass of gradient descent methods, approximate the value function as a linear function of the parameter vector, $\vec{\theta}_t(i)$:

$$Q_t(s,a) = \sum_{i=1}^{n} \theta_t(i)\,\phi_{s,a}(i), \tag{3.12}$$

where each state-action is represented by a vector of features (basis function set), $\vec{\phi}_{s,a} = (\phi_{s,a}(1), \ldots, \phi_{s,a}(n))$, which has the same number of elements as $\vec{\theta}_t$. The

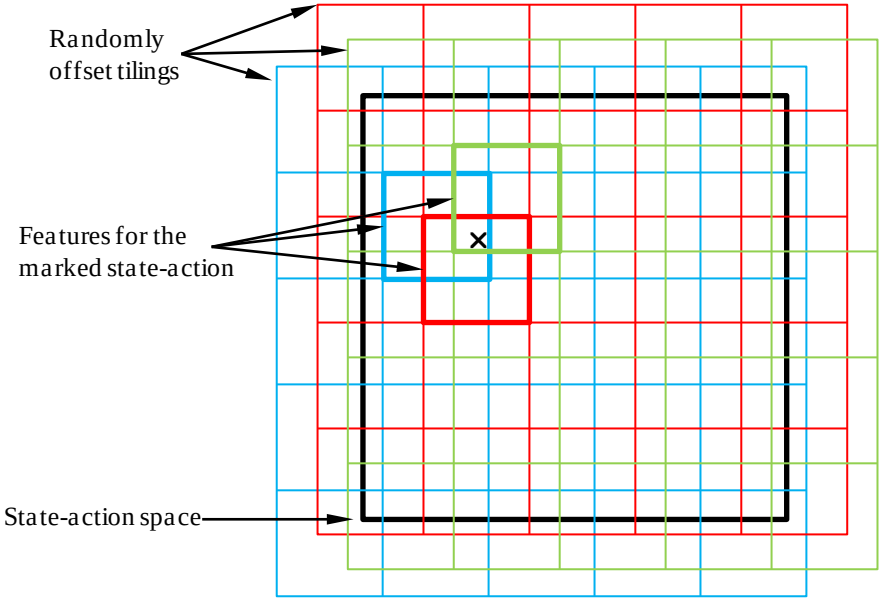


Figure 3.2: A 2D example of TC with 3 randomly offset grid tilings covering the state-action space. The state-action feature set is made up of the relevant overlapping grid cells from each tiling.

features can be constructed in various ways, methods such as TC discretise the entire state-action space into overlapping regions which then become part of the set of features for any state-action they cover. An example of TC features is shown in Figure 3.2.

Linear methods for RL value function approximation are popular since under the same boundary conditions as required previously, the only new requirements for the approximated function to converge to the optimal values is for the basis functions to be linearly independent and not grow too fast (see extended proofs in Tsitsiklis and Van Roy (1997)). However, in high dimensional problems these methods which attempt to discretise the state-action space quickly become impractical to implement. For example, in TC, the resolution of the final approximation is closely linked to the number of tilings and the resolution of the tilings themselves. As the dimensionality and size of the state-action space grows, the number of tilings required to give a "reasonable" approximation also grows. In fact, Sutton and Barto (1998) state that the computational complexity of such methods increases exponentially with the number

of dimensions. For this reason, a more compact function approximation method must be sought when dealing with high dimensional RL problems.

3.3 Gaussian Process Modelling

GP regression is a function approximation method that produces a continuous estimate of the function mean as well as a measure of the estimation uncertainty over the function space in the form of a variance. For the interested reader, Rasmussen and Williams (2005) provides an extensive study into the theory and application of this machine learning method. GP regression has previously been employed to approximate the value function in various RL frameworks, for example, Engel et al. (2005) applied GP regression to learn the continuous state-action value function inside a standard SARSA(λ) procedure, Deisenroth et al. (2009) combined GP regression with DP and demonstrated the ability of the regression technique to estimate the state-action value at locations which are yet unobserved.

The training inputs $X_N = \{\mathbf{x}_i\}_{i=1}^N$ to the GP are the observed state-action pairs,

$$\mathbf{x}_i = [\mathbf{s}_i, \mathbf{a}_i], \tag{3.13}$$

and the training targets $\mathbf{y}_N = \{y_i\}_{i=1}^N$ are the corresponding state-action values which are updated on the fly as the eligibility trace decays,

$$y_i = Q(\mathbf{s}_i, \mathbf{a}_i). \tag{3.14}$$

For a test point $\mathbf{x}_*$, covariance function k, covariance matrix $K = K(X, X)$, and additive white noise drawn from $\mathcal{N}(0, \sigma_n^2)$, the estimated mean value $\bar{Q}_*$ and covariance $\mathrm{cov}(Q_*)$ are

$$\begin{aligned} \bar{Q}_* &= \mathbb{E}[Q(\mathbf{x}_*) | X, \mathbf{y}, \mathbf{x}_*] \\ &= K(\mathbf{x}_*, X)\left(K + \sigma_n^2 I\right)^{-1}\mathbf{y}, \end{aligned} \tag{3.15}$$

$$\operatorname{cov}(Q_*) = K(\mathbf{x}_*, \mathbf{x}_*) - K(\mathbf{x}_*, X)\left(K + \sigma_n^2 I\right)^{-1} K(X, \mathbf{x}_*). \tag{3.16}$$

In fact, Equation (3.15) can be rewritten as a linear combination of N kernel functions:

$$\bar{Q}_* = \sum_{i=1}^{N} \theta_i k(\mathbf{x}_i, \mathbf{x}_*), \tag{3.17}$$

where each kernel/basis function is associated with a state-action observation and the weights,

$$\vec{\theta} = K_{XX}^{-1}\mathbf{y}, \tag{3.18}$$

are a function of the covariance matrix and the training targets, which are updated according to Equation (3.14). Note that in Equation (3.18) $K_{XX} = K + \sigma_n^2 I$.

The representation of the GP in Equation (3.17) bears similarity to the general form of the linear function approximator shown in Equation (3.12), however, there are a couple of significant differences between the two approximation methods. In standard linear function approximation, there is a finite and constant number of basis functions, whereas the GP model grows its basis function set with each new state-action observation. Furthermore, the weighting parameters of the GP model are not only updated according to the value function backup, as per standard linear function approximation, but are also reshaped by the inverse covariance matrix, K_{XX}^{-1}. These differences imply that the convergence proofs of Tsitsiklis and Van Roy (1997) may not apply to GP value function approximation unless some bound is placed on the number of basis functions used in the approximation. Whether sparsification methods that limit the number of basis functions of the GP provide similar convergence guarantees is an interesting and open research question.

The values in the covariance matrix depend upon the choice of covariance function and the respective hyperparameters. The hyperparameters of the GP model can be trained on the data by minimising the negative log marginal likelihood,

$$-\log p(\mathbf{y}|X) = \frac{1}{2}\mathbf{y}^\top K_{XX}^{-1}\mathbf{y} + \frac{1}{2}\log|K_{XX}| + \frac{N}{2}\log 2\pi. \tag{3.19}$$

In this way, the value function approximation weights and GP model are primarily data driven.

The underlying assumption captured by the GP approximation is that state-action pairs close to one another in the covariance function space will have similar associated Q-values. This assumption is not limited to problems where the state and/or action spaces are continuous; it can also be applied to problems where the states and actions are discrete but the transition function implies some sense of continuity, such as in grid search problems. To some extent this continuity is captured by the eligibility trace, however the trace is only able to assign credit to state-action pairs that have been visited whereas the GP approximation is able to estimate the value of state-action locations that are yet to be visited. For this reason, the GP approximation has particular applications to RL problems where the state-action space is continuous or where the problem has a discrete state-action space that is too large to explore exhaustively.

3.3.1 Squared exponential covariance function

While the choice of GP covariance function is not restrictive, prior assumptions regarding the properties of the value function surface can be incorporated via judicious design of this function. For example, there is an assumption of stationarity across the state-action value space of the following experiments, thus the stationary squared exponential covariance function was applied in the GP model used to approximate the value function:

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_f^2 \exp\left[-\frac{1}{2}\left(\mathbf{x} - \mathbf{x}'\right)^T M \left(\mathbf{x} - \mathbf{x}'\right)\right], \tag{3.20}$$

where M is a diagonal matrix with positive elements equal to $\mathbf{l}^{-2}$, and $\mathbf{l} = [l_1, l_2, \ldots, l_n]$ are the length scales in each dimension of the training input vector. The hyperparameters of the covariance function in Equation (3.20) are the length scales, $\mathbf{l}$, and the process variance, σ_f^2. The noise variance, σ_n^2, due to the additive white noise shown in Equation (3.16) is the only other hyperparameter of the GP model.

The covariance matrix, $K(\cdot,\cdot)$, is the covariance function evaluated between each pair of points in the input sets, which for the squared exponential covariance function requires the computation of the squared distance in the exponent. Angular dimensions in the state-action training inputs will invoke wrap-around conditions that must be accounted for when evaluating Equation (3.20) to avoid exaggerating the distance between two angles. For example, if heading is a dimension of the training inputs, then no two headings should be separated by more than π radians. This consideration should also be used to bound the hyperparameters of angular dimensions.

In the experiments presented in this thesis, the hyperparameters were trained offline on a set of simulation data gathered using a preliminary set of estimated hyperparameters. While it is possible to train the hyperparameters online, in practice, the initial stages of learning tended to generate large changes in the value function causing an inflation in the process variance and noise variance hyperparameters, σ_f^2 and σ_n^2, respectively. The Matlab gradient descent function, `fminunc`, was used for hyperparameter learning in the following experiments and was unable to overcome the initial dominance of these two terms during online hyperparameter training, thus it could not learn meaningful hyperparameters.

3.3.2 Sparsification

One of the main drawbacks of using GP regression for value function approximation is the $\mathcal{O}(N^3)$ inversion of the covariance matrix at each update step in Equations (3.15) and (3.16), where N is the number of training inputs to the GP. By updating a single observation at a time, it is possible to take advantage of the matrix inversion lemma to reduce this to an $\mathcal{O}(N^2)$ operation; however, this is still prohibitive over the long training sequences that are a feature of RL. To bound the computation time, a budget can be placed on the number of inputs used to train the GP. Furthermore, the sparsification method described by Csató and Opper (2002) can be applied such that all observations, whether retained in the training set or not, contribute to the final GP model.

Consider the GP training set as a set of basis vectors, $\mathcal{BV} = X_N$, the (noise-free) variance at an observation point $\mathbf{x}_{i+1}$ gives a measure of its linear independence, or "novelty", with respect to the current set of basis vectors,

$$\beta_{\mathbf{x}_{i+1}} = k\left(\mathbf{x}_{i+1}, \mathbf{x}_{i+1}\right) - K\left(\mathbf{x}_{i+1}, X_N\right) K^{-1} K\left(X_N, \mathbf{x}_{i+1}\right). \tag{3.21}$$

As described in Csató and Opper (2002), points with β less than some specified tolerance value β_{tol} are not included into $\mathcal{BV}$. To retain some of the information from these points in the GP model, the relevant covariance matrices can be updated according to the projection of these observations onto the basis vectors,

$$\hat{w}_{i+1} = K^{-1} k\left(X_N, \mathbf{x}_{i+1}\right), \tag{3.22}$$

$$K_{XX}^{-1} = K_{XX}^{-1} - \frac{\left[K_{XX}^{-1} k\left(X_N, \mathbf{x}_{i+1}\right) - \hat{w}_{i+1}\right]\left[K_{XX}^{-1} k\left(X_N, \mathbf{x}_{i+1}\right) - \hat{w}_{i+1}\right]^{\top}}{\sigma_n^2 + \operatorname{cov}\left(\mathbf{x}_{i+1}\right)}. \tag{3.23}$$

Equation (3.23) is derived from the matrix inversion lemma and detailed derivations can be found in Csató (2002).

If the number of points in $\mathcal{BV}$ exceeds the budget, then the observation with the lowest score as defined by,

$$\epsilon_i = \frac{\left(K_{XX}^{-1} y\right)_i}{K_{ii}^{-1} - K_{XX_{ii}}^{-1}}, \tag{3.24}$$

is removed. In Equation (3.24), K_{ii}^{-1} is the i-th diagonal element of K^{-1} and similarly for $K_{XX_{ii}}^{-1}$. The score is an approximation to the Kullback-Leibler divergence between the GP models generated with and without observation $\mathbf{x}_i$. For further details on the derivation of the score, the reader is referred to Csató (2002).

By keeping track of the relevant inverse matrices and applying the matrix inversion lemma at each update/downdate step, it is possible to bound the GP computation time to $\mathcal{O}\left(N_{max}^2\right)$, where N_{max} is the maximum number of training inputs as defined by the computation budget.

3.4 Summary

SARSA(λ) with GP function approximation has a number of desirable properties that makes it particularly suitable for solving control problems such as the autonomous soaring problem. The on-policy learning characteristic of SARSA(λ) allows credit assignment to occur over entire state-action trajectories regardless of the sampling policy, while extension to problems with continuous state-action spaces is granted by applying GP value function approximation. The GP model can provide estimates of the value function at any location in the state-action space without the requirement of having previously observed that state-action. Furthermore, it produces a measure of the uncertainty associated with the estimate in the form of a covariance. The following chapter investigates how this uncertainty measure can be used to direct exploration for more efficient and informative sampling of the state-action space.

Chapter 4

Informative Exploration

The role of exploration in RL is to encourage repeat observations of all state-action transitions to improve the estimate of the value function across the entire space. In this thesis, exploration is treated as an information gathering task where the goal is to reduce the overall uncertainty of the value function estimate. Informative exploration in this context is related to the reduction of uncertainty in the search space, whether for target search and track, such as in Levine et al. (2010), or other sensor coverage tasks such as monitoring spatio-temporal fields as performed in Singh et al. (2010). In these examples, the information utility of performing an action is measured as the reduction in uncertainty of the modelled phenomenon. The same methodology can be applied to induce informative exploration behaviour in RL problems where the modelled phenomenon is the value function.

This chapter introduces a method for directed, adaptive and nonmyopic informative exploration in RL based on a GP model of the state-action value function. The following sections begin with a discussion on the existing exploration methods in RL and the informative sampling literature. A new information measure based on the GP variance volume is then introduced in Section 4.2, with a nonmyopic information value presented in Section 4.3. The chapter concludes with a discussion on how to combine the information value with the state-action value to manipulate exploration behaviour with the (informative) iGP-SARSA(λ) algorithm given in Section 4.5.

4.1 Existing Exploration Strategies

There are a number of methods traditionally used in RL to induce exploratory actions. One of the most common methods is ε-greedy sampling where the greedy (maximal state-action value) action is performed with probability $1 - \varepsilon$ while a random action is performed with probability ε. For such a simple concept, ε-greedy sampling is able to reduce the value function convergence times for some problems. Furthermore, it provides guarantees that all state-actions have a non-zero probability of being visited, thereby satisfying the requirements to prove convergence of the value function for some TD learning algorithms. However, since it is completely undirected and non-adaptive, this method relies entirely on the available actions to "bounce" the learning agent out of local minima and fails in situations where longer sequences of actions are required to move the agent away from learnt trajectories that perform poorly.

Although not strictly used in RL, but worth mentioning here, is the less well-known but quaintly named "run and twiddle" control method proposed by Selfridge (1984). It is inspired by the behaviour of bacteria and acts as an adaptive form of ε-greedy sampling in that it takes the greedy action (run) so long as it is gaining reward, and when a loss is incurred, a random action (twiddle) then follows. Like in ε-greedy sampling, this method does not direct exploration in any way, and simply introduces randomness into the action selection in hopes of discovering a better trajectory. Furthermore, it can also struggle in cases where movement penalties are applied and rewards are only gained at the successful completion of each episode since a significant amount of twiddling would be required before learning to run.

In discrete (tabular) RL problems, a popular method for directing exploration is to bias the initial estimate of the value function. With sufficient prior knowledge of the problem, it is possible to initialise the value function to bias exploration towards areas of high expected reward, however it is often the case that the agent begins the task with little or no knowledge of the field. A uniformly high expected value applied across the entire value function space will promote exploration to areas that have not yet been visited since the value function will gradually be driven down in locations

where the agent has repeat observations of lower than expected rewards. This method can equivalently be thought of as applying a constant information gain reward at each state-action at the start of exploration whose value decreases as observations are made at those locations. As discussed in Section 3.2 however, in a continuous state-action space, the agent will almost surely never revisit any single location, and so this method of promoting exploration would only serve to add a constant bias over the value function approximation.

Nevertheless, the task of exploring a value function space can be equated to an information gathering task where observations at different locations in the space reduce the uncertainty of the value estimate at, and possibly around, the observation location depending on the estimation algorithm. One issue that must then be addressed is how to quantify the uncertainty reduction, or information gain, of an exploratory action.

4.2 Information Measure

The informative sampling literature outside of the RL community is rich with suggestions of how to measure and incorporate the notion of uncertainty reduction when choosing sampling locations. The alphabet optimality criteria are derived from maximising properties of the information matrix such as maximising the minimum eigenvalue in E-optimality, or minimising various properties of the covariance matrix, such as the trace and determinant for A- and D-optimality, respectively, as applied in Binney et al. (2013) and Kollar and Roy (2008). Mutual information and entropy share direct links with A- and D-optimality measures and are also common information measures that have been investigated in many informative path planning applications such as in Singh et al. (2009a) and Hollinger et al. (2013). These metrics are popular since for most problem formulations they maintain their submodularity property which provides performance guarantees on the associated greedy policy, see Nemhauser et al. (1978).

The idea of using information gain rewards to direct exploration in RL has also

been gaining momentum. Engel et al. (2005) suggested using confidence intervals derived from the GP covariance to expand the repertoire of exploration strategies used in RL, while more recently, Still and Precup (2012) used the Kullback-Liebler divergence between successive estimates of the value function to quantify exploration utility. These information measures are restricted by the condition that they can only be computed at discrete sample locations in the space. The following section introduces a new information measure based on the GP variance volume which is capable of measuring the uncertainty reduction across the entire estimation space.

4.2.1 Gaussian process variance volume

Given a GP framework for modelling the value function, it is intuitive to use the GP variance to quantify and compare the information gain of possible state-action observations for directing exploration. Furthermore, since the ultimate goal of accumulating reward is inherently tied to accurately modelling the value function, using the GP variance to measure the information gain maintains a desirable consistency across the value function approximation and information reward. This thesis proposes to use the change in the GP variance volume over the entire state-action space as the measure of uncertainty reduction due to a new set of observations; this information measure was first presented in Chung et al. (2013).

The GP variance over the state-action space represents a bounding volume around the estimated value function surface, shown by the translucent grey surface bounding the coloured mean estimation surface from above in Figure 4.1. Each consecutive observation results in a reduction in this volume and we define this as the uncertainty reduction, or information gain, of the corresponding state-action observation,

$$V_{bound_N} = \int_{x_{n_a}}^{x_{n_b}} \cdots \int_{x_{1_a}}^{x_{1_b}} \operatorname{cov}\left([x_1, \ldots, x_n] \,|X_N\right) \mathrm{d}x_1 \ldots \mathrm{d}x_n, \tag{4.1}$$

$$I_{gain} = V_{bound_N} - V_{bound_{N+1}}. \tag{4.2}$$

The training set $X_N = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ consists of N observations of n-dimensional

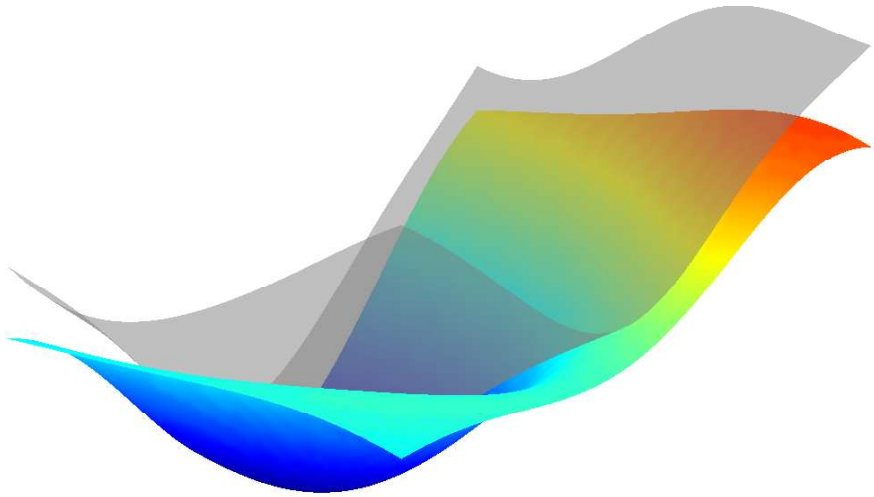


Figure 4.1: The GP variance volume is bounded above by the translucent grey surface and below by the GP mean estimate, shown as the coloured surface.

state-action pairs, with dimensions $[x_1, \ldots, x_n] = [s_1, \ldots, s_u, a_1, \ldots, a_v]$; furthermore, $X_{N+1} = X_N \bigcup \mathbf{x}_{N+1}$.

Since the squared exponential covariance function given in Equation (3.20) is an integrable function, an analytical solution to Equation (4.1) can be found,

$$\begin{aligned} V_{bound} = & \int_{x_{n_a}}^{x_{n_b}} \cdots \int_{x_{1_a}}^{x_{1_b}} k\left([x_1, \ldots, x_n], [x_1, \ldots, x_n]\right) - \\ & k\left([x_1, \ldots, x_n], X_N\right) K_{XX}^{-1} k\left(X_N, [x_1, \ldots, x_n]\right) \mathrm{d}x_1 \ldots \mathrm{d}x_n \\ = & \sigma_f^2 \prod_{m=1}^{n} \left(x_{m_b} - x_{m_a}\right) - \sigma_f^4 \left(\frac{\sqrt{\pi}}{2}\right)^n \prod_{m=1}^{n} \left(l_m\right) \\ & \times \sum_{i=1}^{N} \sum_{j=1}^{N} \left\{ \left[K_{XX}^{-1}\right]_{ij} \exp\left[-\sum_{m=1}^{n} \left(\frac{x_{i_m} - x_{j_m}}{2 l_m}\right)^2\right] \right. \\ & \times \left. \prod_{m=1}^{n} \left[\operatorname{erf}\left(\frac{x_{m_b} - \frac{x_{i_m} + x_{j_m}}{2}}{l_m}\right) - \operatorname{erf}\left(\frac{x_{m_a} - \frac{x_{i_m} + x_{j_m}}{2}}{l_m}\right)\right] \right\}, \end{aligned} \tag{4.3}$$

where k is the squared exponential covariance function and $K_{XX}^{-1} = \left[K\left(X, X\right) + \sigma_n^2 I\right]^{-1}$ is the inverse of the covariance matrix of all the observed locations including additive white noise drawn from $\mathcal{N}\left(0, \sigma_n^2\right)$. The full derivation of V_{bound} is given in Appendix B.

Indeed, the requirement for deriving an analytical solution to the variance volume for

a GP with any covariance function comes down to the integrability of the expression:

$$\int_{\mathbf{x}_a}^{\mathbf{x}_b} k\left(\mathbf{x}, \mathbf{x}_i\right) k\left(\mathbf{x}_j, \mathbf{x}\right). \tag{4.4}$$

If this integral can be formulated as a closed-form expression then an analytical solution to the variance volume can be found for the respective covariance function. Equation (4.3) is not strictly a closed-form expression due to the requirement for numerical approximation of the error function terms. However, most modern programming languages such as Matlab have an inbuilt `erf` function that uses elementary functions to compute fast and accurate approximations, thus providing tractability.

The limits of integration $\{\mathbf{x}_a, \mathbf{x}_b\}$ can be chosen to incorporate the entire space or only a local portion of it, for example, a reachable set within a finite time horizon. Given that the computational cost is the same for either calculation it seems natural to take the integral over the complete space. However, depending on the design of the GP covariance function, there may be situations where it is desirable to restrict the integration limits such that the information gain measure only represents the uncertainty reduction in a region of interest. This may be implemented to prevent large changes in the variance volume in highly correlated but spatially distant regions from biasing exploration away from state-actions that can improve the value function estimate in more "interesting" or more immediately reachable areas.

4.2.2 Comparison to other information measures

Other common measures used to compute the uncertainty associated with the GP estimate include the trace and the entropy, which operate on the associated covariance matrix derived from discrete samples of the input space. The covariance matrix for a sequence of sample points $X_* = [\mathbf{x}_{*_1}, \ldots \mathbf{x}_{*_c}]$ given training inputs X_N is

$$K\left(X_*, X_* | X_N\right) = K\left(X_*, X_*\right) - K\left(X_*, X_N\right) K_{XX}^{-1} K\left(X_N, X_*\right), \tag{4.5}$$

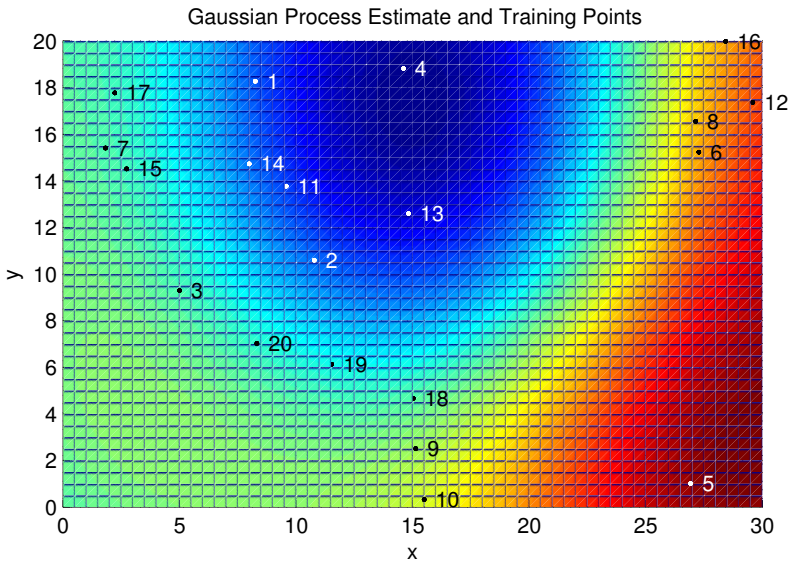


Figure 4.2: The GP estimate used in the information measure comparison study. The sequence of 20 randomly selected training inputs are numbered and overlaid on top of the final GP estimate of the underlying function surface.

which derives from Equation (3.16). The trace of this matrix is simply computed as the sum of the diagonal terms, while the differential entropy is defined by Guestrin et al. (2005) to be,

$$\mathcal{H}\left(X_*|X_N\right) = \frac{1}{2}\log\left(2\pi\exp\left(K\left(X_*, X_*|X_N\right)\right)\right). \tag{4.6}$$

A comparison of the uncertainty and information gain calculated using the GP variance volume and these measures is conducted on the test case shown in Figure 4.2. The uncertainty of the GP estimate is computed as each consecutive random sample point is added to the training input set. In the variance volume information measure, the integration limits were taken to be the (x, y) limits of the space. For the trace and entropy measures, the GP covariance was predicted at the grid of points generated along $x = [0, 1, \ldots, 30]$ and $y = [0, 1, \ldots, 20]$, giving a total of $31 \times 21 = 651$ test points.

The computed uncertainty as each new training input was added to the set is shown in Figure 4.3. All three measures record a similar trend in the uncertainty reduction as new training points are introduced, with differences more pronounced in the variance

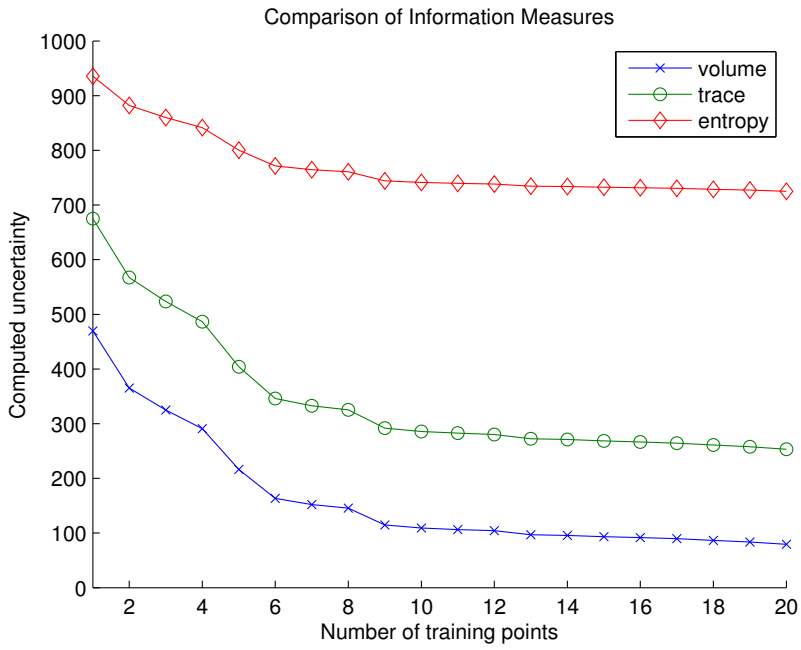
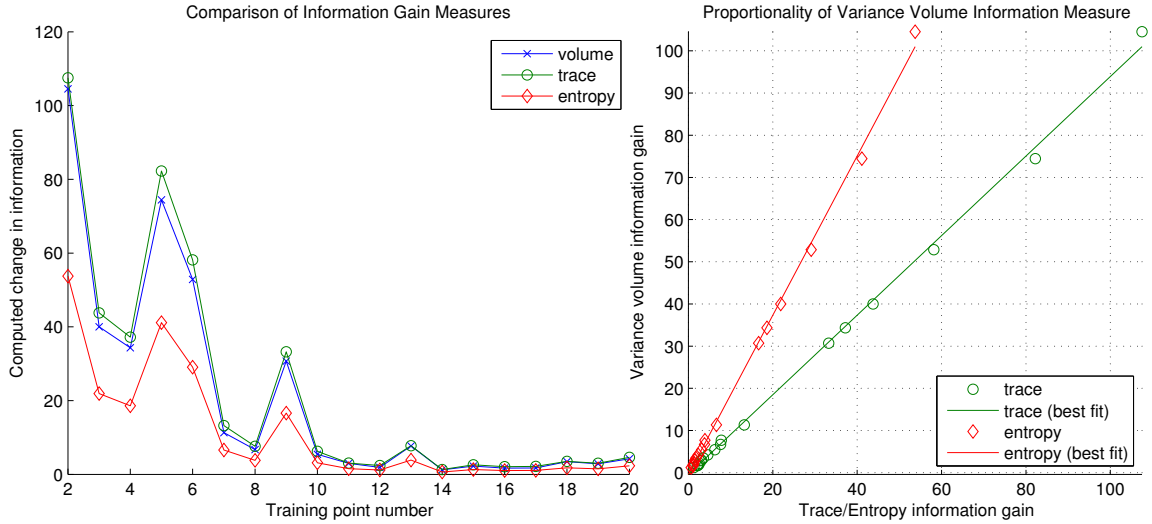


Figure 4.3: A comparison of the computed uncertainty of the GP estimate using the GP variance volume information measure, the trace of the covariance matrix and the entropy.

volume and trace measures. The reduction in uncertainty (that is, the information gain) over consecutive measurements is shown in Figure 4.4a. There is a clear consistency between the variance volume information gain measurements and the trace and entropy measurements. Figure 4.4b plots the variance volume measurements against those of the trace and entropy, showing a distinct direct proportionality relationship between the measures. This is to be expected since the diagonals of the covariance matrix represent the predicted variance at those points while the differential entropy is a monotonic function of the variance.

It is important to note, however, that the trace and entropy are dependent on the number of sampled points since the total uncertainty is taken as the sum of the uncertainty at all the sampled points. This has implications on the choice of the test point set as well as on the computation time. Often the sample points are chosen according to some external criteria, for example, expected target locations or potential sensor placement positions; without such restrictions, the choice of sample points can only be guided by coverage. The test point set in this study was chosen uniformly across the input space, however, it tends to be the case that the uncertainty is not



(a) Incremental information gain.

(b) Proportionality relationship.

Figure 4.4: Figure 4.4a gives a comparison of the computed information gain of including the n-th training point to the GP estimate across the three different information measures. The GP variance volume information measure is consistent with the both of the other information measures. A close look at the proportionality relationship between the variance volume measure and the other information measures is given in Figure 4.4b; it can be seen that the relationship to the trace and entropy measures is close to linear.

distributed equally and so differences in the sample set will lead to differences in the computed uncertainty and information gain. To reduce this effect, a denser set of sample points can be taken, this leads to another consideration in the increased computational requirement of a larger set.

A comparison of the computation times for each information measure is given in Table 4.1. All three information measures require the K_{XX}^{-1} matrix and the time taken to compute this matrix is dependent on the number of training inputs as described in Section 3.3.2. However, since it is constant across the three cases, it is not included in the values shown in Table 4.1. The times shown here are for a Matlab program running on a dual core 3 GHz computer. The variance volume provides an analytical measure across the entire input space and so is not dependent on the number of sample points. In comparison, the computation times for the trace and entropy measures increase proportionally as this set grows.

Table 4.1: Information Gain Computation Times (20 training inputs)

Number of test points	Variance volume	Trace	Entropy
176	0.0017 s	0.0023 s	0.0303 s
651	0.0017 s	0.0383 s	0.1398 s
2501	0.0017 s	0.4799 s	0.8766 s

4.3 Information Value

The information *reward* of taking an observation at a particular state-action is simply computed as Equation (4.2), however, approaching this from an RL perspective, it is more meaningful to consider the nonmyopic information *value*, that is, the total future information gain possible due to performing a particular action. There is one key difference to note between the RL state-action value and this information value: the RL reward for any state-action is constant whereas the information reward for reobserving a particular state-action decreases over the number of observations. The main consequence of this is that the information reward cannot be directly included in the computation of Q, and similarly, information gain credit cannot be meaningfully assigned back along the eligibility trace. Instead, the approach proposed in this thesis is to look at the set of possible future state-action observations leading out from a particular action via a forward propagating rollout, the discounted sum of those expected future information gain rewards are then used to compute the information value of that initial action.

4.3.1 Rollout

The information gain rollout technique is illustrated in Figure 4.5. The information value sums the information gained from the next proposed transition, $s', a' \rightarrow s''$ where $a' \in \mathcal{A}(s')$ and $\mathcal{A}(s')$ is the set of available actions from state s', with the discounted information gained from all possible future state-actions rolled out from s'' up to a threshold discount factor. Given a discount parameter $\gamma_r < 1$, the total

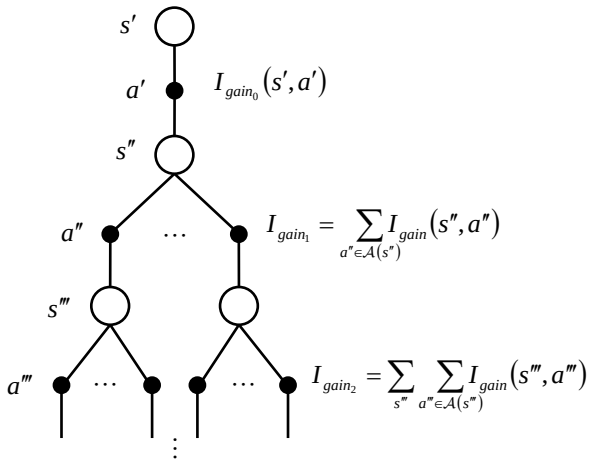


Figure 4.5: The rollout method introduced in Chung et al. (2013). The information gain of each rollout level considers all the reachable state-actions at that level. The total information value is a discounted sum of the information gain of each level.

information gain of an action a is computed as,

$$I_{a_{total}} = I_{gain_0} + \gamma_r I_{gain_1} + \gamma_r^2 I_{gain_2} + \ldots + \gamma_r^p I_{gain_p}, \tag{4.7}$$

where p is the highest integer for which γ_r^p is greater than the discount threshold γ_{thres}.

The breadth of the expansion tree is determined by the number of available actions at each depth, therefore, large action sets and a long rollout depth will generate a heavy computational burden when computing the rollout. A greedy rollout method is later presented in Section 4.4.2 that can reduce the exponential branching of the information gain tree to a linear computation which is dependent only on the search depth.

Rollout requires knowledge of the state transition model for forward state propagation, however, stochasticity in the form of external disturbances or an imperfect model will affect the quality of the future state estimates. Because of this, the information value is more accurately thought of as an expectation, that is, the information value is the discounted sum of expected future information gain in the same way that the

state-action value is the discounted sum of expected future rewards.

4.4 Objective Function Trade-off

The state-action value from the GP value function approximation and the nonmyopic information rollout value quantify, respectively, the exploitation and exploration utilities of performing an action at a particular state. The way in which these two utilities are combined in the action selection objective function will produce different learning behaviours in the agent. For example, heavily favouring the information value will encourage the agent to select potentially risky actions that take it to areas of the state-action space that have high uncertainty, while favouring the state-action value will promote more risk averse actions that move the agent to areas of high expected values. Furthermore, it is often desirable for a learning system to dynamically adapt its exploration-exploitation behaviour according to some overarching specifications such as the total available learning time. Under such circumstances, a dynamic weighting factor, such as the attention parameter proposed by Thrun and Möller (1992) or the exploration-exploitation utility function presented in Chung et al. (2012), can be introduced to balance the information value against the state-action value.

When combining the two values, some thought must also be given to the difference in magnitude that may exist between the state-action value and the information value. The magnitude of the state-action value is largely determined by the reward function, discount factors and the step-size parameter. If the step-size does not satisfy Equation (3.4) then there is no guarantee of value function convergence and the state-action value can potentially grow unbounded if the reward function and discount factors are poorly chosen. On the other hand, the information value as computed using the GP variance volume, tends to shrink as more state-action observations are made.

The ranking of the available actions according to the objective function ultimately determines which action is executed. Since this is the case, it is useful to normalise

the values when combining them as this removes the problem of their differences in magnitude. That is, let

$$\hat{Q}_i = \frac{\bar{Q}_i}{\max|\bar{Q}_i|}, \tag{4.8}$$

$$\hat{I}_i = \frac{I_{i_{total}}}{\max|I_{i_{total}}|}, \tag{4.9}$$

where i is the current time step. Note that the value estimate component lies within $[-1, 1]$ while the information gain component lies within $(0, 1]$. The action selection objective function can now be written as,

$$J_i = \hat{Q}_i + \omega_i \hat{I}_i, \tag{4.10}$$

where the exploration weighting factor ω_i can be a function of the learning system state. Note that normalisation of the values in the action selection objective function removes the need to handle the magnitude differences between the raw measurements in the design of the exploration-exploitation decision making function ω. Other value combination methods may also be used to control learning behaviour. For example, the actual magnitude of information value provides some notion of the expected information gain rewards and may be useful for determining when exploration is no longer necessary, that is, when exploration provides little improvement to the uncertainty reduction of the value function estimate. This and other methods for combining $\bar{Q}$ and I_{total} are possible within the framework of the learning algorithm. The weighting functions presented in this thesis all have values drawn from $[0, 1]$; this follows the "optimism in the face of uncertainty" methodology which applies exploration bonuses on top of the expected return. Other weighting functions that draw from a larger range such as $[-1, 1]$ can also be used to penalise exploration. Alternatively, a counter-weight can be applied to the state-action value to shift from pure exploitation to pure exploration.

Two weighting functions are developed in this thesis, the following subsection describes a time-step-dependent weighting function while a resource-level-dependent weighting function is presented later on in Section 6.1.

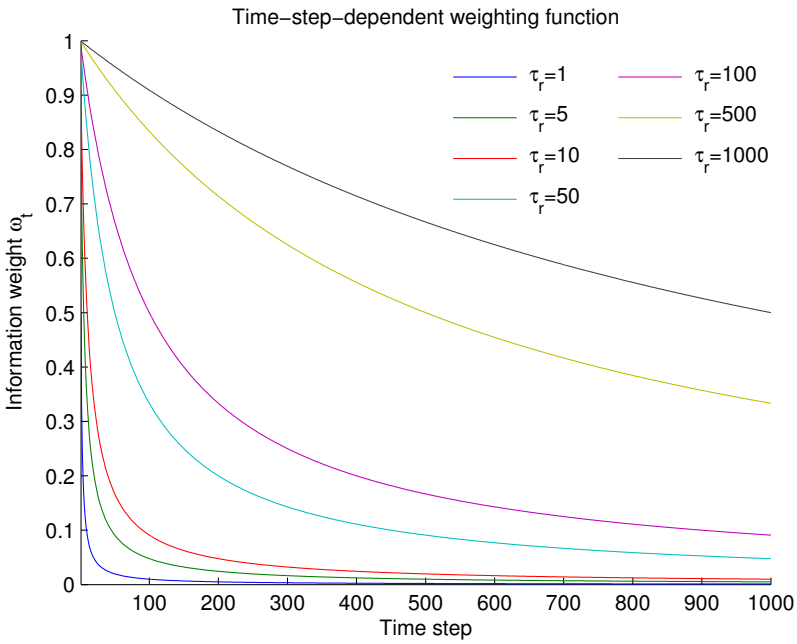


Figure 4.6: The time-step-dependent information weighting function evalutated for a range of τ_r values.

4.4.1 Time-step-dependent information weighting

When using the change in the GP covariance volume as the information measure, a consequence of the normalisation in Equation (4.9) is that the natural decline in information value as more observations are taken is also eliminated. Generally it is desirable to maintain this decline in the final action selection objective function to encourage the learning agent to explore more at the start and less later on when the number of observations increases.

The decline in information value can be reintroduced as a decreasing weighting factor on the normalised information value. The time-step-dependent information weighting function is defined as,

$$\omega_t = \frac{\tau_r}{\tau_r + i}, \tag{4.11}$$

where i is the current time step. For problems with a continuous state and/or action space, i is equivalent to the number of unique state-action observations, however, this is not necessarily the case for discrete problems where a single state-action can be reobserved multiple times. Depending on the requirements of the problem, a possible

modification of Equation (4.11) would be to directly use the number of unique state-action observations in place of i. The overall effect of this is to scale the agent's behaviour from exploratory to exploitative as learning progresses, Figure 4.6 plots ω_t for various τ_r values. The weighting factor in Equation (4.11) does not rely on a fixed mission length by which to decay the information value weight, instead the decay rate is determined by the τ_r parameter. The weighting decay profile has an inverse relationship with the current time step where τ_r represents the "half-life" of the information value, that is, the information value of the τ_r-th step is given a weighting of $\frac{1}{2}$. Small values of τ_r drive the agent towards exploitative behaviour early on, while large values encourage exploration for a longer proportion of the mission.

4.4.2 Greedy rollout

Having introduced the weighted objective function, it is now possible to propose a directed rollout method to reduce the computational load of the information value calculation. Recall from Section 4.3.1 that the information gain rollout tree branches exponentially in the number of available actions. To prevent this exponential branching, a directed sampling method can be applied to restrict the rollout to only include the expected future state-actions. That is, only consider the information gain from future state-actions where the actions are selected from an approximation of the current policy,

$$a = \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}} \tilde{J}_a = \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}} \hat{Q}_a + \omega_i \frac{I_{a_{gain_0}}}{\max \left| I_{a_{gain_0}} \right|}, \tag{4.12}$$

when computing the information value. The rollout diagram for this *greedy rollout* method is similar to the backup diagram for Q-learning and is shown in Figure 4.7.

Enforcing this restriction on the state-actions considered in the information gain rollout ensures that the computation of the information value is linear in the rollout depth. It is expected that when the current policy is in a state of flux, the greedy rollout information value will poorly represent the actual future information gains since the state-action value function, and consequently the policy, are likely to change. In contrast, this issue does not exist when the full information rollout is taken since

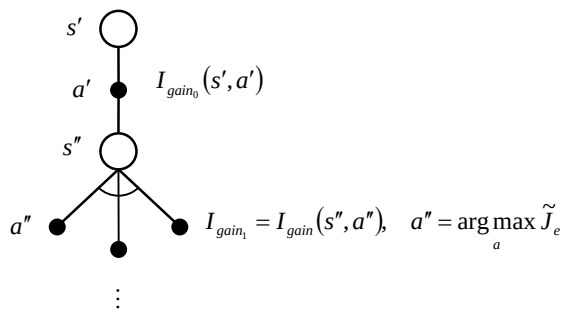


Figure 4.7: Rollout diagram for the greedy rollout method. Only the information gain from the expected state-actions of the current policy are considered in the calculation of the information value.

all possible actions are considered and contribute to the information value.

4.5 iGP-SARSA(λ)

The complete learning algorithm using an informative exploration strategy is dubbed iGP-SARSA(λ) and is shown in Algorithm 1. It combines GP value function approximation with informative exploration, via the GP variance volume information measure from Equation (4.2) and the action selection objective function given by Equation (4.11), in a replace trace SARSA(λ) RL framework. Both the full rollout and greedy rollout sampling methods can be used within this algorithm by adjusting the information value computation on line 11.

iGP-SARSA(λ) in Algorithm 1 differs from a standard GP-SARSA(λ) implementation by the information value and objective function calculation in lines 12 to 17. In GP-SARSA(λ), this would be replaced with an ε-greedy selection of the next action to execute.

4.6 Summary

This chapter presented the mechanisms from which an informative exploration strategy was developed for a SARSA(λ) RL framework that uses a GP for value function

Algorithm 1 iGP-SARSA(λ)

```
1: Q̄_a ← 0                                                    ▷ Initial value estimate
2: Initialise θ                                                ▷ GP hyperparameters
3: for each episode do
4:     e ← 0                                                   ▷ Initialise trace
5:     s, a ← initial state and action
6:     for each step i do
7:         e(s, a) ← 1                                         ▷ Replacing traces
8:         Take action a, observe reward, r, and next state, s'
9:         δ ← r − Q̄_a
10:        for all a* ∈ A(s') do
11:            Q̄_{a*} ∼ GP_Q                                   ▷ GP approximation for Q
12:            I_{a*_total} ← I_{gain_0} + γ_r I_{gain_1} + ... + γ_r^p I_{gain_p}
13:        end for
14:        ω_t ← τ_r / (τ_r + i)                               ▷ Time-step-dependent information weighting
15:        Q̂_{a*} ← Q̄_{a*} / max|Q̄_{a*}|                       ▷ Normalised state-action value
16:        Î_{a*} ← I_{a*_total} / max|I_{a*_total}|           ▷ Normalised information value
17:        J_{a*_t} ← Q̂_{a*} + ω_t Î_{a*}
18:        a' ← arg max_{a*} J_{a*_t}
19:        δ ← δ + γQ̄_{a'}
20:        if (s, a) is a new state then
21:            Append (s, a) to GP_Q training inputs X
22:        end if
23:        y ← y + αδe                                         ▷ Update GP_Q training targets
24:        if retrain hyperparameters then
25:            θ ← arg min_θ [− log p(y|X, θ)]                 ▷ Minimise the
                                                               negative log marginal likelihood
26:        end if
27:        e ← γλe
28:        s ← s'
29:        a ← a'
30:    end for
31: end for
```

approximation. An information gain metric based on the change in the GP variance was proposed and an analytical solution to finding the variance volume of a GP with a squared exponential covariance function was provided in Equation (4.3). The information gain rollout method is able to compute a nonmyopic information value from the discounted sum of future information gain and this value was combined with the

state-action value according to a time-step-dependent weighting function.

The presented algorithm, iGP-SARSA(λ), uses a sampling strategy that is able to direct exploration to areas of the state-action space where there is potential for high information gain. Furthermore, this strategy adapts according to the state-actions that have already been observed since the information gain in those areas are driven down in the GP variance. Finally, the weighting function shown in Equation (4.11) is designed to promote exploration in the early stages of learning and gradually increase exploitative behaviour that attempts to maximise the state-action value as learning progresses. The following chapter investigates the performance of iGP-SARSA(λ) as compared to existing RL methods.

Chapter 5

Benchmarking Experiments

This chapter presents a series of benchmarking experiments to compare the iGP-SARSA(λ) algorithm against existing SARSA(λ) methods. Four learning scenarios are investigated, the first two are based on the puddle world and cart pole problems as defined in the 2005 NIPS RL Benchmarking workshop, and the final two are the battery cycling problem and 2D 3DOF soaring glider problem that were first presented in Chung et al. (2013). Puddle world, cart pole and the battery cycling problem all have continuous states with discrete actions, while the 2D soaring glider problem has both discrete states and actions. In the cases where the state space is continuous, TC (as described in Section 3.2) is used for value function approximation.

The learning algorithms compared in the following experiments are listed below along with their associated plot markers:

1. SARSA(λ) with TC for value function approximation (if value function approximation is required) and ε-greedy sampling (blue diamonds)
2. SARSA(λ) with GPs for value function approximation and ε-greedy sampling (green squares)
3. iGP-SARSA(λ) with greedy rollout informative exploration (red circles)
4. iGP-SARSA(λ) with full rollout informative exploration (cyan triangles).

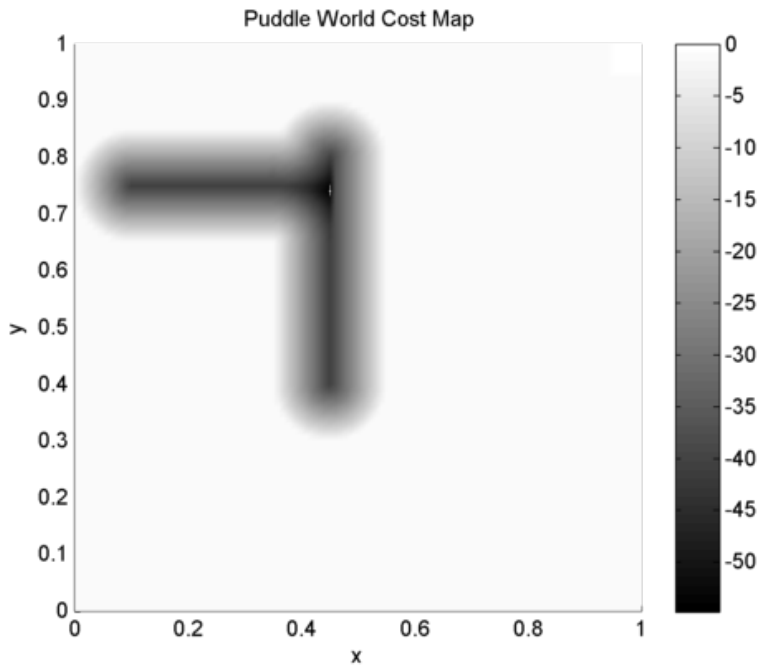


Figure 5.1: Puddle world cost map. Terminal states lie in the region $x \geq 0.95 \bigcap y \geq 0.95$ and all transitions incur a cost of $-1 - 400 \times$ *distance inside puddle.*

5.1 Puddle World

5.1.1 Simulation setup

The puddle world simulation is based on the problem described in Sutton (1996). The agent state is defined by its (x, y) coordinates and at each step the agent can choose one of four actions: {up, down, right, left}, which moves the agent in the chosen direction by a distance drawn from $\mathcal{N}(0.05, 0.01^2)$. The goal region is defined as $x \geq 0.95 \bigcap y \geq 0.95$ and transition into this area terminates the episode. All other state-action transitions incur a cost of $-1 - 400 \times$ *distance inside puddle.* The puddle world cost map is shown in Figure 5.1. The two puddles both have radii of 0.1 and are located between centre points $\{(0.1, 0.75), (0.45, 0.75)\}$ and $\{(0.45, 0.4), (0.45, 0.8)\}$.

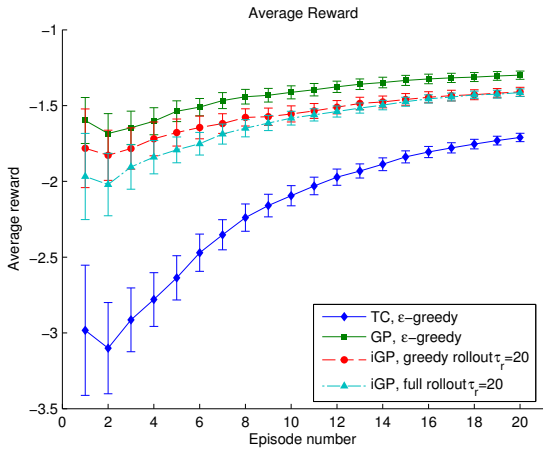
Simulations for each algorithm were repeated over 100 trials, each consisting of 20 episodes. Each episode was run for a maximum of 100 steps with the agent beginning in the same set of random locations for each algorithm test set. Since both dimensions of the problem state are continuous, value function approximation in the form of TC

and GP regression modelling was required. For the TC experiments, $m = 10$ tilings with 10 partitions for each state dimension over 4 actions were used to discretise the state-action space, the partition size was chosen with consideration of the puddle dimensions and the required resolution to adequately map the cost variations across the puddle world. The step-size was chosen as, $\alpha = \frac{0.5}{m}$, with the discount parameters $\gamma = 0.9$, $\lambda = 0.9$, and random exploration probability, $\varepsilon = 0.01$.

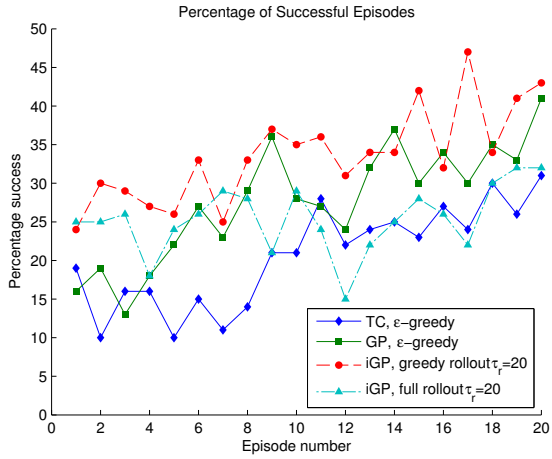
For the GP value function approximation experiments, the same discount factors were used, while the step-size was chosen to be $\alpha = 0.5$. In the GP-SARSA(λ) experiments, the same ε value was used for the sampling strategy. The informative exploration trials used greedy and full rollout with a rollout discount value of $\gamma_r = 0.4$ to a threshold parameter $\gamma_{thres} = 0.1$, providing a rollout depth of 3. In the action selection objective function, the step number count was reset for each new episode, and the information value half-life was chosen to be $\tau_r = 20$, which represents one-fifth of the maximum allowable number of time steps per episode. GP hyperparameter training was performed offline with a data set generated using a nominal set of hyperparameter values. The final hyperparameters were chosen to be $\left\{0.1, 0.1, \frac{\pi}{3}\right\}$ for the length scales in the $\{x, y, action\}$ dimensions, respectively, with process variance $\sigma_f^2 = 0.1$ and noise variance $\sigma_n^2 = 0.1$. A summary of the learning parameters used in this set of trials is given in Table A.1 of Appendix A.1.

5.1.2 Results

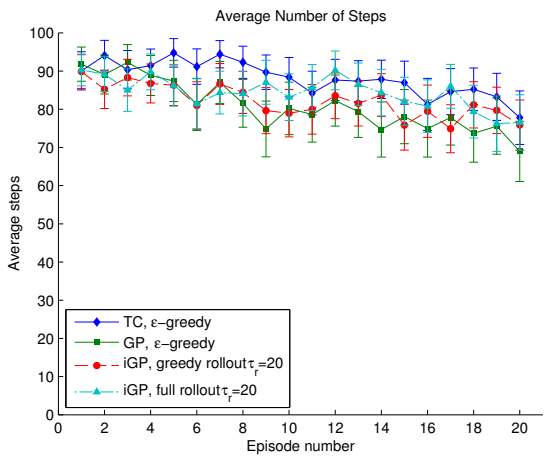
The averaged results over each set of 100 simulation trials are shown in Figure 5.2. The greatest difference in performance between the four tested algorithms is seen in the plot of the average rewards in Figure 5.2a. The three GP value function approximation algorithms are each able to achieve an average reward above -1.5 whereas the TC simulations produced a much lower average reward over all the episodes culminating in an average reward of -1.7 in the final episode. This difference in performance is particularly interesting since the percentage of episodes that ended in the terminal region for the TC experiments and the iGP-SARSA(λ) experiments with full rollout



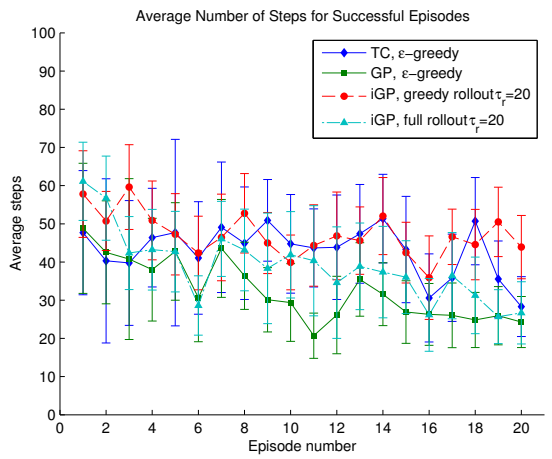
(a) Average reward over all simulations.



(b) Percentage of episodes that ended in the terminal region.



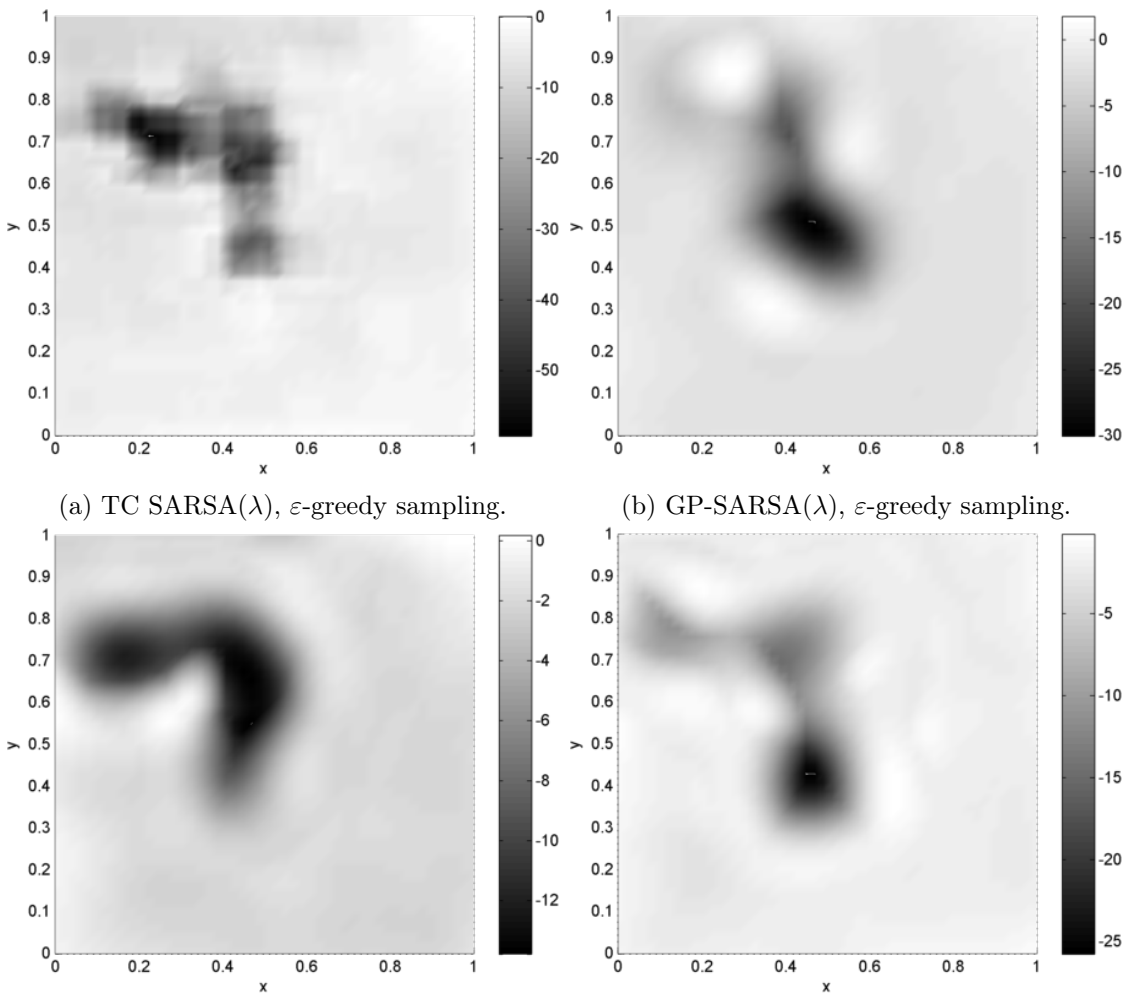
(c) Average number of steps taken over all simulations for all episodes.



(d) Average number of steps taken over all simulations for episodes that ended in the terminal region.

Figure 5.2: Results for the set of puddle world simulations. Each algorithm was tested on 100 simulations of 20 episodes, each episode was allowed a maximum number of 100 steps. Episodes terminated if the goal region was not reached within the maximum number of steps. Averages over all 100 simulations for each algorithm set are shown with 95% confidence intervals where applicable.

are very close, especially towards the latter half of the episodes. This suggests that the iGP-SARSA(λ) algorithm was able to find lower cost paths to the terminal region than TC SARSA(λ). In this set of experiments, the GP-SARSA(λ) algorithm produced the highest final average reward of -1.3, however, iGP-SARSA(λ) with greedy rollout for informative exploration produced the highest number of episodes that terminated in the goal region, with a final average reward coinciding with that of the full rollout



(a) TC SARSA(λ), ε-greedy sampling.

(b) GP-SARSA(λ), ε-greedy sampling.

(c) iGP-SARSA(λ), greedy rollout sampling.

(d) iGP-SARSA(λ), full rollout sampling.

Figure 5.3: Final estimated value functions of one trial for each learning algorithm. Although the estimates vary in precision, the darker high cost regions are all roughly estimated to be around the puddle regions shown in Figure 5.1.

simulations.

The effect of the informative exploration strategies can be seen in the average number of steps taken shown in Figure 5.2d. The greedy and full rollout algorithms visited more state-actions in the episodes that terminated in the goal region than GP-SARSA(λ) with ε-greedy sampling. The TC experiments also required more steps to reach the goal region; however, since this set of experiments also applied an ε-greedy sampling strategy, and given its poor reward-gain performance, this behaviour is more likely attributed to learning an inferior policy.

The final estimated value function (for the maximal actions) of one group of trials is shown for each learning algorithm in Figure 5.3. For each of the tested algorithms, the darker high cost regions all roughly align to the puddle regions in the cost map shown in Figure 5.1. There is a notable difference between the smoothness of the estimations from the TC approximation and the GP approximation; while the smoothness of the GP model is influenced by the choice of covariance function, TC relies on the resolution of the partitionings and the number of tilings used. Increasing the number of partitions can increase the resolution of the value function approximation, however, in this case, more state-action observations are required to visit all the partitions and cover the space. Instead, it may be beneficial to limit the partitioning resolution in favour of increasing the number of tilings to improve learning performance.

One issue with using GP regression for value function approximation is that unlike in TC, the GP estimate is not bounded by the observations. Therefore, even though the entire puddle world cost map has values that are less than or equal to 0, it is possible for the GP approximation to assign positive values to some state-action pairs; this can be seen in Figures 5.3b and 5.3c. These artifacts are due to the GP attempting to smooth the model according to the covariance function and the respective hyperparameters. They can be reduced by taking more observations at the locations where they occur.

Both of the informative exploration algorithms generated value functions that appear more similar in shape to the actual puddle world cost map when compared to GP-SARSA(λ). The iGP-SARSA(λ) algorithms were able to pick up both the horizontal and vertical puddle components while GP-SARSA(λ) only picked out the vertical puddle. Considering the similar average reward profiles shown in Figure 5.2a, and the percentage of episodes that reached the goal region shown in Figure 5.2b, it can be seen that in the puddle world problem informative exploration can produce a better approximation of the value function while also performing mission tasks to a degree comparable to that of existing strategies such as ε-greedy sampling.

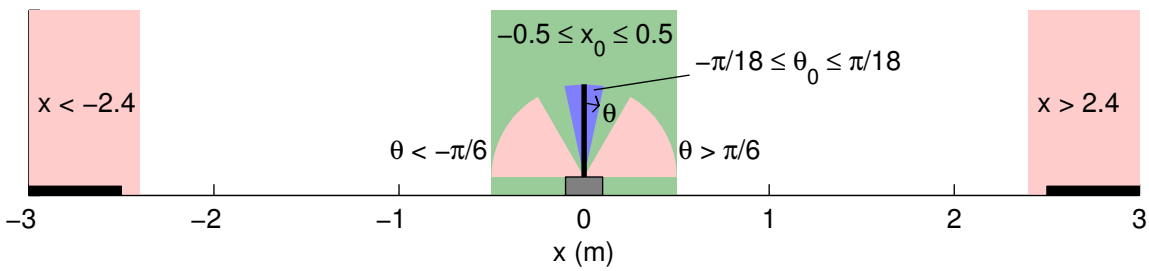


Figure 5.4: The cart pole problem setup: the initial conditions of the cart position and pole position for each run are shown in the shaded green and shaded blue regions, respectively. Failure conditions are shown by the shaded red regions. The cart is able to slide freely between $-2.4 \leq x \leq 2.4$ and the pole is able to rotate freely about its hinge.

5.2 Cart Pole

5.2.1 Simulation setup

The cart pole problem as described in Littman et al. (2005) is "to apply forces to a cart moving along a track so as to keep a pole hinged to the cart from falling over. A failure is said to occur if the pole falls past a given angle from vertical or if the cart runs off the track." The four continuous state variables in this problem are the cart position x (m), cart velocity v (m/s), pole angle θ (rad from vertical), and pole angular velocity ω (rad/s). The initial conditions of each episode are chosen from the following region: $-0.5 \leq x_0 \leq 0.5$, $\frac{-\pi}{18} \leq \theta_0 \leq \frac{\pi}{18}$, $v_0 = 0$, $\omega_0 = 0$, as shown in Figure 5.4, while the failure conditions are defined as $|x| > 2.4$ or $|\theta| > \frac{\pi}{6}$.

The cart pole dynamics are defined by the following equations:

$$
\begin{aligned}
\dot{\theta} &= \omega, \\
\dot{x} &= v, \\
\dot{\omega} &= \frac{g \sin\theta + \cos\theta \frac{-F - m_p l_p \omega^2 \sin\theta}{m_c + m_p}}{l_p \left(\frac{4}{3} - \frac{m_p \cos\theta}{m_c + m_p} \right)}, \\
\dot{v} &= \frac{F + m_p l_p \left(\omega^2 \sin\theta - \dot{\omega} \cos\theta \right)}{m_c + m_p},
\end{aligned}
\tag{5.1}
$$

where F is the input action force applied to the cart and is drawn from the action

set, $\mathcal{A} = F \in \{-10, -9, \ldots, 9, 10\}$ N, and the constants in Equation (5.1) are given in Table A.2. The Matlab simulation implemented a fourth order integration routine via the inbuilt `ode45` function using a time step of $\Delta t = 0.02\,\text{s}$.

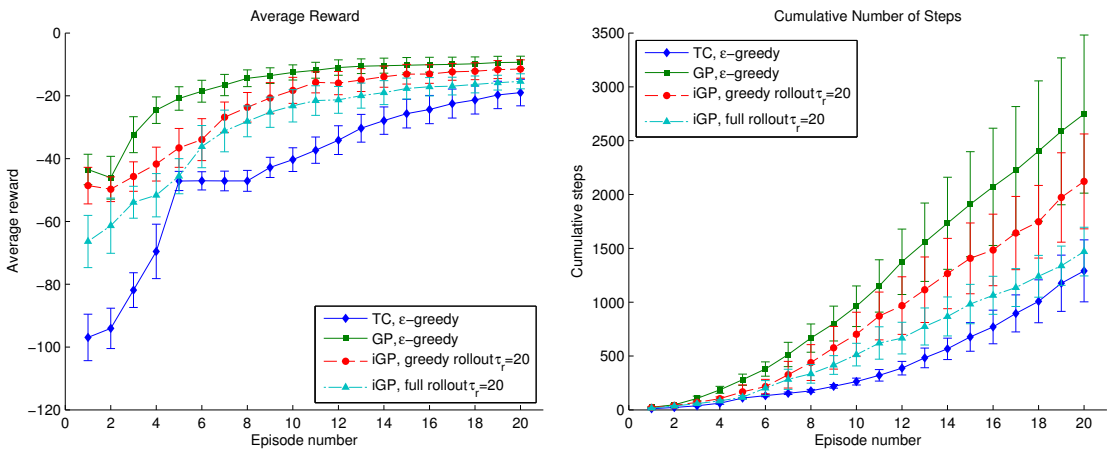
The reward function for the cart pole problem is given as:

$$r = \begin{cases} 0 & \text{if } |\theta| \leq \frac{\pi}{60} \text{ and } |x| \leq 0.05 \text{ (balancing)}, \\ -1000 & \text{if } |\theta| \geq \frac{\pi}{6} \text{ or } |x| \geq 2.4 \text{ (failure)}, \\ -1 & \text{otherwise.} \end{cases} \tag{5.2}$$

Episodes were also terminated if a failure state was encountered, that is, if the cart ran off the tracks or if the pole angle was greater than $\frac{\pi}{6}$ rad from vertical.

Each algorithm was run for 15 trials, each consisting of 20 episodes; the starting states were consistent across the four tested algorithms. Each episode was run for a maximum of 500 steps, which is equivalent to 10 s at a time step of $\Delta t = 0.02\,\text{s}$. For the TC experiments, $m = 100$ tilings with 5 partitions for each state dimension over 21 actions (a total of $5^4 \times 21 \times 100 = 1312500$ features) were used to discretise the state-action space. The step-size was chosen as, $\alpha = \frac{0.5}{m}$, with the discount parameters $\gamma = 0.9$, $\lambda = 0.9$, and random exploration probability, $\varepsilon = 0.01$.

For the GP value function approximation experiments, the same discount factors were used, while the step-size was chosen to be $\alpha = 0.5$. As with the puddle world trials, the GP-SARSA(λ) experiments used the same ε value as the TC experiments for the sampling strategy. The informative exploration trials used greedy and full rollout with a rollout discount value of $\gamma_r = 0.4$ to a threshold parameter $\gamma_{thres} = 0.1$, providing a rollout depth of 3; in the action selection objective function, the information value half-life was chosen to be $\tau_r = 20$. GP hyperparameter training was performed offline with a data set generated using a nominal set of hyperparameter values. The final hyperparameters were chosen to be $\left\{\frac{\pi}{18}, \frac{\pi}{18}, 1.5, 1, 1.5\right\}$ for the length scales in the $\{\theta, \omega, x, v, action\}$ dimensions, respectively, with process variance $\sigma_f^2 = 0.1$ and noise variance $\sigma_n^2 = 0.1$. A summary of the learning parameters used in this set of trials is given in Table A.3 of Appendix A.2.



(a) Average reward over all simulations.

(b) Cumulative number of steps across the episodes and averaged over all simulations.

Figure 5.5: Results for the set of cart pole simulations. Each algorithm was tested on 15 simulations of 20 episodes, each episode is allowed a maximum number of 500 steps. Episodes terminate if the pole falls beyond a defined angle from vertical or if the cart attempts to move beyond the rails. Averages over all 15 simulations for each algorithm set are shown with 95% confidence intervals.

5.2.2 Results

The average reward and cumulative number of steps for the cart pole simulation sets are shown in Figure 5.5. The GP-SARSA(λ) algorithm converges to the highest average reward while also achieving the highest number of cumulative steps. The TC simulations display the poorest performance despite using the same ε-greedy exploration method, achieving the lowest average reward and lowest number of cumulative steps overall. This difference in performance is attributed to the ability of the GP function approximation to infer values across the entire state-action space given only a finite number of observations. In comparison, TC requires visitation of all state-action partitions to garner a similar understanding of the value function surface. Furthermore, since the value function is initially assumed to be 0 everywhere and given the reward scheme in Equation (5.2), it is likely that at the start of learning, all unobserved state-actions will appear to be of greater value than those already visited, generating greater exploration to state-actions that are difficult to recover from. This issue is far less prevalent in GP function approximation since the entire value function estimate is updated with each observation.

The number of episodes for which the pole was able to balance for the entire duration of the episode is shown in Table 5.1.

Table 5.1: Number of Completed Episodes

Algorithm	**Number of completed episodes**
TC SARSA(λ)	3
GP-SARSA(λ)	16
iGP-SARSA(λ) with greedy rollout	21
iGP-SARSA(λ) with full rollout	6

It is interesting to note that although iGP-SARSA(λ) with greedy rollout was able to balance the pole for more episodes, the cumulative number of steps of GP-SARSA(λ) was still greater than either of the informative exploration algorithms. The proposed explanation for these results is related to the particular reward setup for the cart pole problem and the exploration strategy applied by iGP-SARSA(λ).

iGP-SARSA(λ) encourages early exploration to decay as the number of time steps in each episode increases, however, many of the state-actions of the cart pole problem lead to termination conditions, that is, the pole falling beyond a certain angle from vertical or the cart driving off the rails. Before observation, state-action value estimates along these "failure" trajectories have a higher associated variance and are more profitable in an exploration sense, the difficulty is that once in these states, there become far fewer actions that return the pole to a balanced position than actions that cause termination. iGP-SARSA(λ) tries to explore these state-action regions at the start of each episode but this often leads to early termination and consequently fewer observations to improve the value function approximation. Furthermore, the maximum reward region is very narrow and so only a small number of observations is required to drive down the variance in this region. With the exploration value of the maximum reward state-actions quickly reduced, the exploration phase at the start of each episode becomes dominated by state-actions that are difficult to recover from and are much more likely to lead to termination.

From this analysis, it is perhaps fair to conclude that the particular exploration strategy used in iGP-SARSA(λ) is not suitable for this cart pole balancing problem. The problem setup has parallels to the soaring glider problem, which also exhibits similar critical states. Because of this, a more conservative exploration mechanism that can account for the risk of failure seems to be required for this task. For example, an exploration strategy that can take into account the pole angle when deciding whether to explore or exploit may produce longer trajectories and consequently improve learning performance over the current iGP-SARSA(λ) exploration scheme.

On the other hand, a modification to the problem setup to allow swing up may result in more favourable learning conditions for iGP-SARSA(λ). The most common cause for episode termination is the pole falling beyond the threshold vertical angle, if this termination condition is removed, iGP-SARSA(λ) will have greater opportunity to recover and learn from costly states encountered during its early exploration phase.

5.3 Battery Cycling

For both of the prior benchmarking experiments, puddle world and cart pole, there is only one way to gain reward in each setup. In puddle world there is only one termination region and in cart pole there is only one region where balancing the pole incurs no cost. The design of these narrow reward regions results in no real benefit for performing exploration once the agent has found a trajectory that is successful. Local exploration by an ε-greedy sampling strategy allows for small refinements to a successful trajectory, however iGP-SARSA(λ) actively directs the agent away from visited locations, which in these experiments is in fact distracting the agent from the goal of the learning problem and typically induces a heavy penalty as well.

The problem of learning to soar differs from the previous benchmarking problems in that there are multiple trajectories that the agent can execute to can gain reward, however some are more efficient than others. In these situations exploration is necessary to traverse the state-action space and discover the most profitable paths.

The following two benchmarking experiments are designed with multiple reward regions to compare the exploration ability of the iGP-SARSA(λ) algorithm to ε-greedy sampling.

5.3.1 Simulation setup

Consider the task of providing energy above a particular rate from a battery with the charging and discharging profiles shown in Figure 5.6. A reward of 0.01 is received when the discharge rate of the battery is greater than or equal to the reward threshold of $d_{thres} = 0.045$, while a cost of -0.01 is incurred any time the battery is charging or whenever it is discharging below the reward threshold. The two profiles generate a non-convex reward surface where local optima exist between $0.4 \leq energy \leq 1$. At each step, two discrete actions are available: the battery can either charge up or discharge. The energy state of the battery is continuous between 0 and 1, i.e. empty and full, respectively.

Each algorithm was tested over 100 trials of 500 s simulations with a time step of $\Delta t = 0.5$ s, giving a total of 1000 time steps per trial. For the TC experiments, $m = 100$ tilings with 50 state partitions over 2 actions were used to discretise the state-action space. The step-size was chosen as, $\alpha = \frac{0.5}{m}$, with discount parameters, $\gamma = 0.9$, $\lambda = 0.7$, and random exploration probability, $\varepsilon = 0.01$. In the GP- and iGP-SARSA(λ) experiments, the same discount parameters were used, while the step size was set to $\alpha = 0.5$. The hyperparameters of the GP model were trained offline, they were found to be $\{0.3, 0.18\}$ for the lengths scales in the $\{energy, action\}$ dimensions, with a process variance $\sigma_f^2 = 0.14$ and noise variance $\sigma_n^2 = 0.013$. In the greedy and full rollout iGP-SARSA(λ) experiments, a rollout discount value of $\gamma_r = 0.4$ was used with a threshold parameter $\gamma_{thres} = 0.1$, providing a rollout depth of 3; in the action selection objective function, the information value half-life was chosen to be $\tau_r = 500$, that is, half the maximum number of time steps for each simulation. For each experiment, the battery began at a random energy state; the set of random starting states was consistent across the full and greedy rollout iGP-, GP- and TC

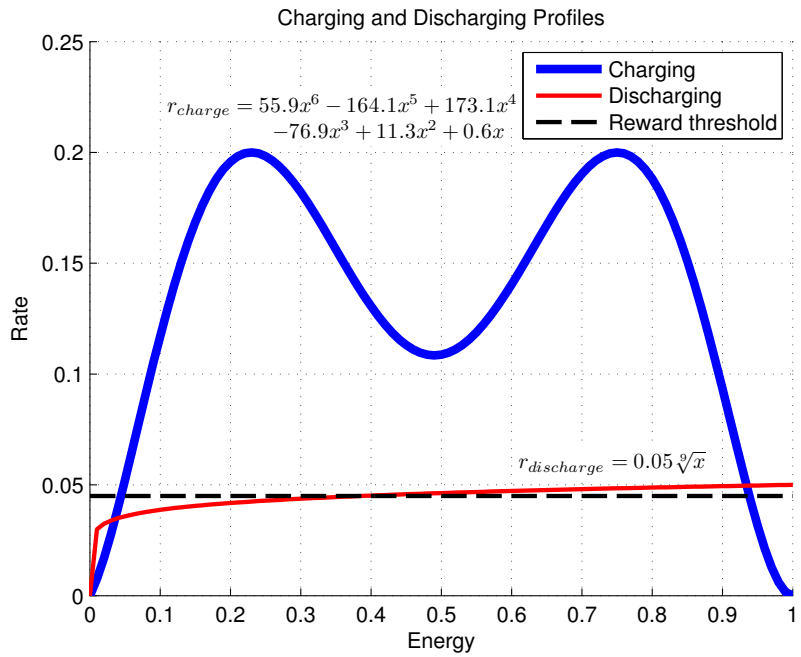


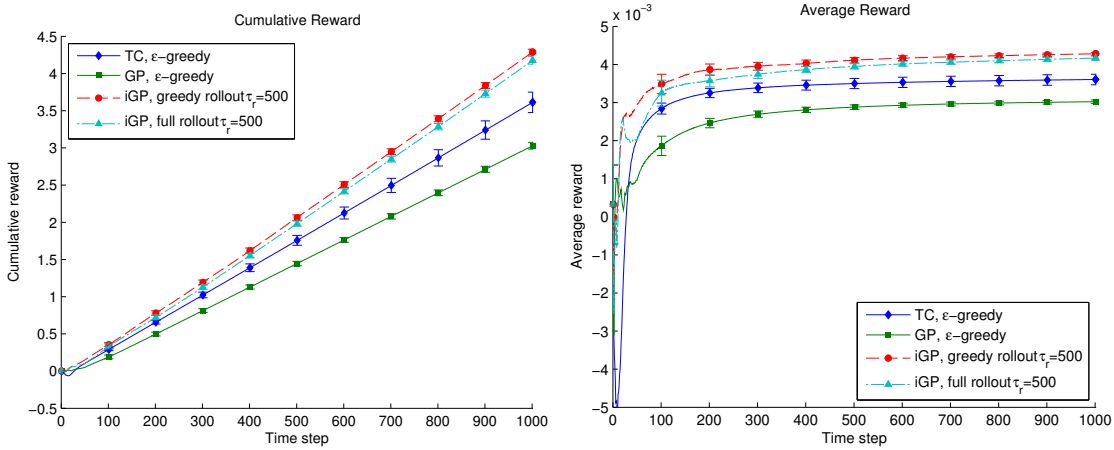
Figure 5.6: Battery charging and discharging profiles as a function of the energy state, equations for each profile are also given where $x = energy$. The discharge reward threshold is shown as the dashed line.

SARSA(λ) experiments. A summary of the learning parameters used in this set of trials is given in Table A.4 of Appendix A.3.

5.3.2 Results

Figure 5.7 shows the averaged results for the cumulative and average rewards over the 100 simulation trials for each algorithm. Both of the iGP-SARSA(λ) algorithms are seen to perform better than the algorithms that relied on ε-greedy for exploration. The average rewards for the informative exploration algorithms converge to 4.17×10^{-3} for the full rollout case and 4.28×10^{-3} for the greedy rollout simulations, while the TC simulations converged to an average reward of 3.61×10^{-3} and the GP-SARSA(λ) average rewards converged to 3.02×10^{-3}.

It is perhaps unexpected that GP-SARSA(λ) performed so poorly compared to the other three algorithms especially when considering its performance in the puddle world and cart pole problems shown in the previous subsections. The main differ-

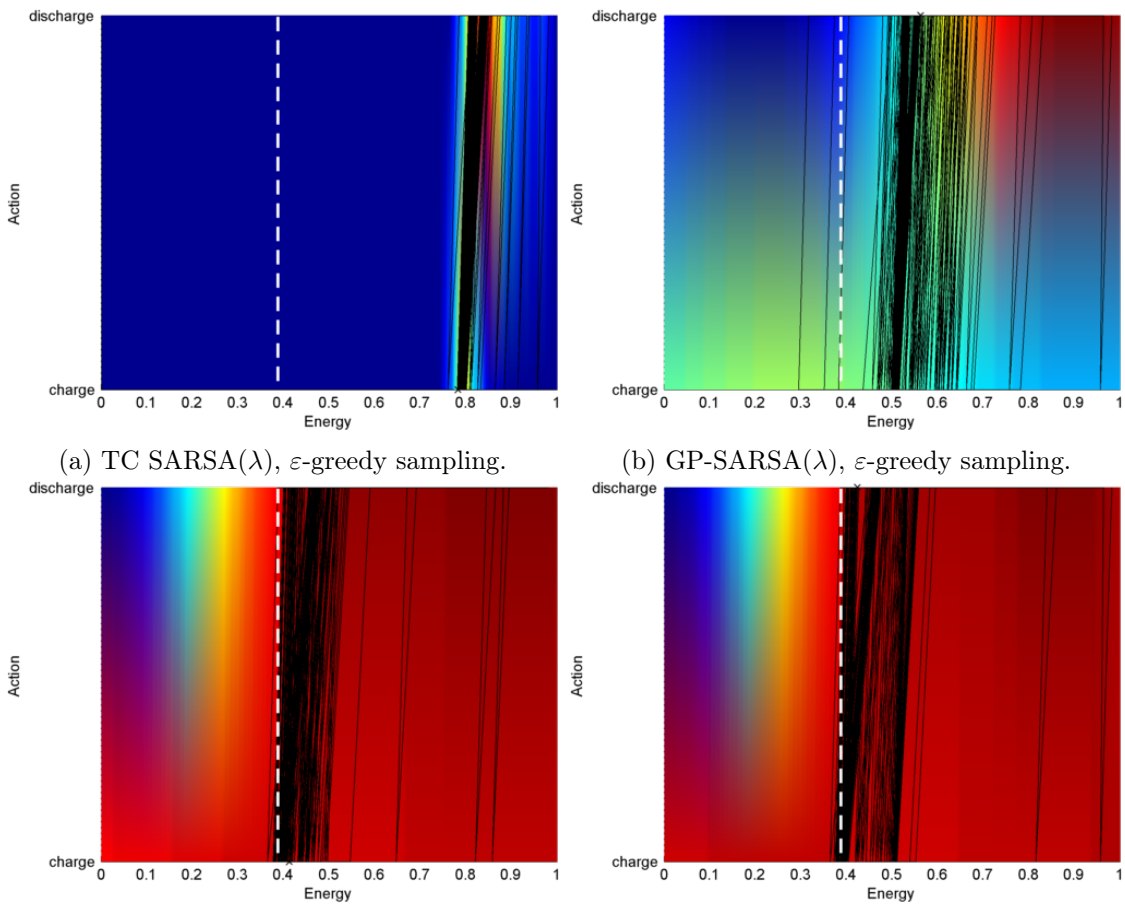


(a) Cumulative reward over all trials. (b) Progression of average reward over all trials.

Figure 5.7: Comparison of the mean cumulative reward over all 100 trials for TC SARSA(λ) and GP-SARSA(λ) against iGP-SARSA(λ), 95% confidence intervals are given at 100 time step intervals. The averaged cumulative reward is shown in Figure 5.7a and the average reward received is given in Figure 5.7b.

ence between the battery cycling problem and the previous two problems is that the state-actions that generate positive reinforcement in the battery cycling problem form trajectory loops that exist in local maxima across the state-action space, this is shown by the different trajectories taken by each of the learning algorithms in Figure 5.8. Each of these trajectories have converged to a locally optimal reward-gaining cycle, however, without adequate nonmyopic exploration, it becomes difficult to search for other cycles that may produce a higher average reward. These results show that the informative exploration strategy is able to explore beyond locally optimal loops to find a higher reward-gaining cycle than the ε-greedy exploration strategy.

The state-action trajectory and approximated state-action value function of one trial is shown for each of the tested algorithms in Figure 5.8. The state-action trajectory is given by the solid black line, which is overlaid above the learnt state-action value function represented by the coloured surface plot beneath. Since TC SARSA(λ) with ε-greedy sampling was unable to explore beyond the first locally optimal cycle it discovered, the state-action value function is only approximated at and around those locations that were visited. GP-SARSA(λ) with ε-greedy sampling provides better coverage of the state-action value function in its approximation since the GP is able



(a) TC SARSA(λ), ε-greedy sampling.

(b) GP-SARSA(λ), ε-greedy sampling.

(c) iGP-SARSA(λ), greedy rollout sampling.

(d) iGP-SARSA(λ), full rollout sampling.

Figure 5.8: Battery charge/discharge paths for one set of trials overlaid on top of the approximated value function. The dashed white line shows the energy at which the discharge rate is above the reward threshold. A cost is incurred if the battery attempts to discharge anywhere to the left of the dashed line. The SARSA(λ) trial coverges early on to the nearest locally optimal solution, around $energy = 0.8$, while the iGP-SARSA(λ) algorithm is able to find the globally optimal cycle between $0.4 \leq energy \leq 0.5$. The final state-action location is indicated by the cross.

to infer the value at locations that have not been visited. As discussed previously, the random sampling method used in this algorithm is unable to drive exploration beyond locally optimal trajectory cycles and so the learnt value function does not detect the higher reward-gaining loop near the discharge rate cut-off (shown by the dashed white line in Figure 5.8), instead it expects the optimal loop to exist around $energy = 0.5$. In comparison, the two informative exploration trials both discover the globally optimal trajectory cycle around $energy = 0.4$; in particular, the ability of the exploration mechanism is demonstrated in Figures 5.8c and 5.8d where it can

be seen that the algorithm initially discovers the loop around $energy = 0.5$ but is able to explore beyond it to converge to the cycle with the higher reward gain.

5.4 2D 3DOF Soaring Glider

5.4.1 Simulation setup

In the soaring glider problem, the search space is discretised into an hexagonal grid world with an underlying wind energy field as shown in Figure 5.9. The glider state is defined by (x, y, ψ), the location and heading of the platform. The headings are discretised according to the six faces of the hexgrid cells, however, since this state is angular, the wraparound condition must be enforced when computing the squared distances for the GP covariance function. The glider is only able to travel within the defined search space and receives a penalty for choosing actions that attempt to move it beyond the boundaries.

At each state, the glider has the choice of three actions: turn left, go straight or turn right. The state transition for each action is shown in Figure 5.10. Headings are defined as 0° north, then 60°, 120°, etc., travelling in an anti-clockwise direction from north. Since the state-action space is discrete, TC was not necessary for these experiments and standard SARSA(λ) was applied.

The reward functions for the hexgrid example are based on a simple glider aircraft model. During flight in no wind the aircraft is assumed to lose energy at a constant rate while travelling forwards with an additional penalty when turning. This is equivalent to a glider travelling at a constant airspeed that loses energy in proportion to the lift to drag ratio $\frac{L}{D}$, the vehicle mass m_{glider} and the acceleration due to gravity g. For an electric vehicle these can be considered constant values during flight. An additional penalty is applied to turns due to the drag from increased wing loading. With the fixed turn angles of 60° in the hexgrid example, the additional turn penalty is approximated as 1.3 times the steady-level energy loss over the same distance. The

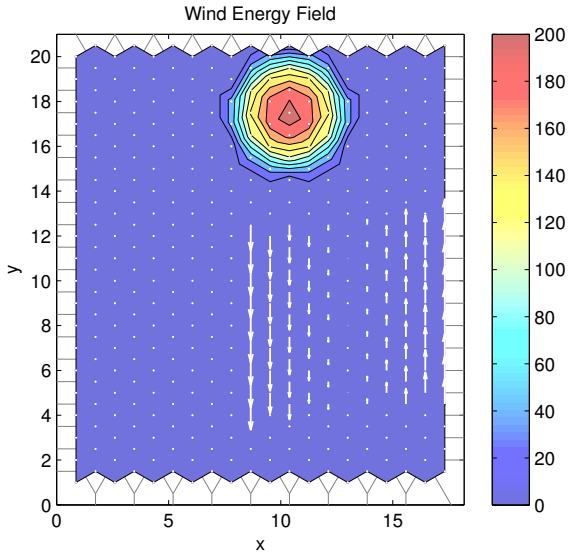


Figure 5.9: Wind energy field for the soaring glider simulation. A thermal is centred at $(12, 17)$ and a wind shear field is present between the limits of $10 \leq x \leq 20$ and $5 \leq y \leq 15$. The filled contour plot shows the thermal energy in Joules.

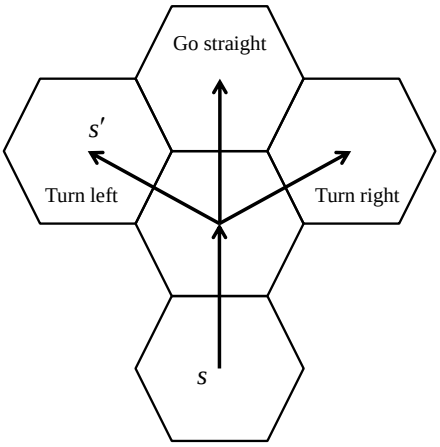


Figure 5.10: State transition for each available action. Each action involves an initial forward step, the following step is either into the cell on a bearing of 60° to the left or right, or into the cell directly ahead.

resulting cost function for a movement action between cells spaced distance d apart is

$$r_{move} = -\frac{m_{glider} g d}{\frac{L}{D}} \times \begin{cases} 1 & \text{if action} = \text{go straight} \\ 1.3 & \text{otherwise.} \end{cases} \tag{5.3}$$

The energy sources in the field are either rising air (static soaring) or wind gradient

(dynamic soaring) sources. A rising air energy source is placed at $(12, 17)$ with a defined core wind speed w_{therm} and radius l_{therm}. Energy is gained from rising air simply by flying through it (independent of heading) and is proportional to the strength of the wind. At a distance d_{therm} from the thermal centre the reward is

$$r_{therm} = m_{glider} g \Delta t w_{therm} \times \begin{cases} 1 - \left(\frac{d_{therm}}{l_{therm}}\right)^2 & \text{if } d_{therm} < l_{therm} \\ 0 & \text{if } d_{therm} \geq l_{therm}, \end{cases} \tag{5.4}$$

where Δt is the time step.

Dynamic soaring is the process of collecting energy by moving through a spatial wind gradient, which effectively increases the airspeed of the aircraft. By flying cyclic patterns through wind shear it is possible to continuously increase airspeed. Typically, dynamic soaring is performed in a vertical wind gradient (with respect to altitude) rather than the planar wind gradient used here; whilst an in-depth discussion of dynamic soaring is beyond the scope of this thesis, suitable descriptions can be found in Wood (1972), Weimerskirch et al. (2000), Lawrance (2011). The dynamic soaring sources are modelled as a planar linear shear gradient, as noted by the wind vectors in Figure 5.9. Energy capture from a wind gradient is due to the increased air-relative kinetic energy gained through the increase in airspeed from the wind gradient. For a linear wind gradient $\frac{\partial W_x}{\partial y}$, the kinetic energy and power are

$$E_{kinetic} = \tfrac{1}{2} m V^2, \tag{5.5}$$

$$\frac{dE_{kinetic}}{dt} = \tfrac{1}{2} m \left(2V \frac{dV}{dt}\right) \tag{5.6}$$

$$= \tfrac{1}{2} m \left(\tfrac{\partial W_x}{\partial y} V^2 \cos\psi \sin\psi\right). \tag{5.7}$$

This is effectively the projection of the wind gradient in the airspeed direction, for this simulation the resulting reward function for travelling distance d through a linear

gradient $\frac{\partial W_x}{\partial y}$ at airspeed V and heading ψ is

$$r_{shear} = \tfrac{1}{2} m d \sin\psi \cos\psi \tfrac{\partial W_x}{\partial y} \left(2V + d \sin\psi \cos\psi \tfrac{\partial W_x}{\partial y} \right). \tag{5.8}$$

Thus positive reward is obtained by heading into a positive gradient, while negative reward is received for heading into a negative gradient.

Finally, the reward function includes a penalty term for flight outside the specified area. The edge penalty is twice the magnitude of the strongest wind energy source,

$$r_{edge} = \begin{cases} -2 \max W & \text{if outside flight area} \\ 0 & \text{otherwise.} \end{cases} \tag{5.9}$$

The resulting reward function is the sum of these four components,

$$r = r_{move} + r_{therm} + r_{shear} + r_{edge}. \tag{5.10}$$

In addition to the reward function, the four energy gain/loss components are used to compute the remaining platform energy. In the experiments that follow, the agent is restricted to an upper energy bound defined by the maximum energy and a lower bound of zero which represents a critical failure. The glider began each experiment at maximum energy, this was chosen to be the amount of energy required to fly straight and level for half of the total simulation time. Considering the additional energy penalties for turning and attempting to exit the field, it is not expected that a random policy will be able to maintain positive platform energy levels over the duration of the simulation.

The 2D 3DOF glider experiment was repeated over 15 trials of 10^4 s simulations (timestep $\Delta t = 1$ s) for each algorithm. In each of the 15 trials, the glider began in a random starting location that was consistent across the four tested algorithms. The discount factors used in this set of simulations are $\gamma = 0.9$, $\lambda = 0.7$, with step size $\alpha = 0.5$. For the random sampling exploration policies, $\varepsilon = 0.01$, while the rollout discount value was again chosen to be $\gamma_r = 0.4$, with discount threshold,

$\gamma_{thres} = 0.1$, giving a total of 3 rollout levels. To observe the effect of the information half-life parameter, τ_r was selected to be $\{2000, 5000, 7000\}$. As with the previous experiments, the hyperparameters of the GP model used to approximate the value function were trained offline; the length scales were chosen to be $\left\{3, 3.5, \frac{\pi}{4}, 0.5\right\}$ for the $\{x, y, \psi, action\}$ dimensions, respectively, with process variance $\sigma_f^2 = 10$ and noise variance $\sigma_n^2 = 10$. A summary of the learning parameters used in this set of trials is given in Table A.5 of Appendix A.4.

5.4.2 Results

Sample flight paths for each algorithm are shown in Figure 5.11. Figure 5.12 shows the averaged results over all 20 trials with 95% confidence intervals given at 1000 time step intervals; the following series of plots show only the $\tau_r = 5000$ results. The progression of the average reward is shown in Figure 5.12a where it can be seen that GP-SARSA(λ) with ε-greedy sampling achieves the highest final average reward across the simulations. Both of the informative exploration algorithms have a similar average reward profile to that of GP-SARSA(λ), while SARSA(λ) exhibits several initial dips before rising and overtaking the informative exploration strategies in the latter half of the simulations. The steep drops in reward-gain performance early on for the SARSA(λ) simulations suggests that the agent continues to attempt actions that send it outside of the flight area. The GP approximation cases are able to avoid this behaviour because the GP can efficiently propagate the large negative reward triggered by these actions to nearby state-actions thereby preventing the agent from attempting similar actions in those states.

Since the state-action space is discrete in this problem, it is possible to directly compare the exploration behaviour of each algorithm by examining the number of observed state-action pairs, this is shown in Figure 5.12b. The two algorithms that use ε-greedy sampling produce the highest and lowest number of observed state-actions. This is perhaps not that surprising given the average reward profiles of Figure 5.12a; GP-SARSA(λ) is able to discover and maintain reward-gaining state-actions early

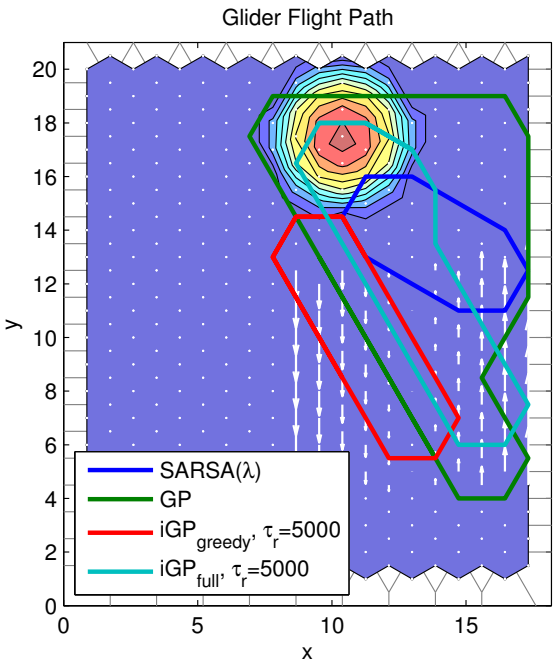
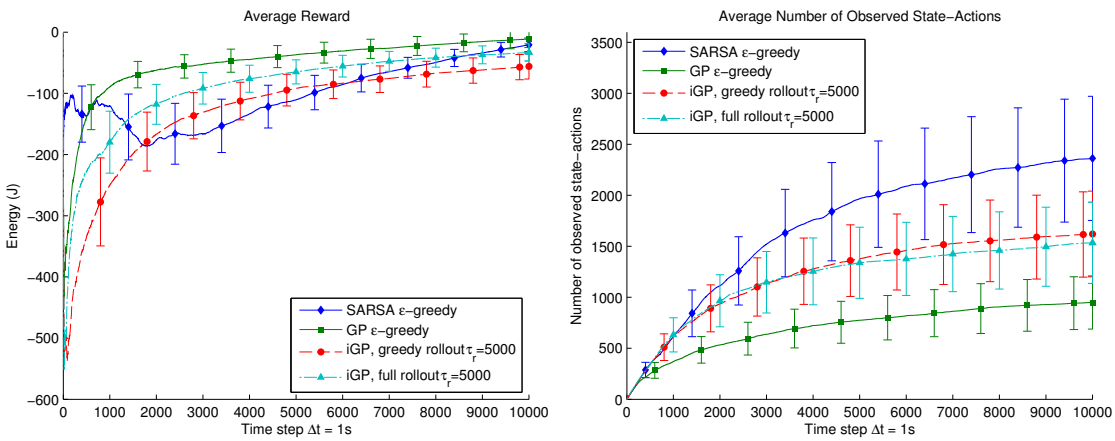


Figure 5.11: Flight trajectories learnt by each of the algorithms. Most of the flight time is concentrated around areas of potential wind energy gain, and it is apparent that paths can vary in their energy gain efficiency.

on, as a result, the random actions induced by ε-greedy sampling are largely damped out by the learnt value function. In comparison, SARSA(λ) cannot infer the value of unobserved state-actions and so requires many more time steps to identify reward-gaining state-actions. Furthermore, the ε-greedy sampling strategy is not the only force driving exploration at the start. The initial value function is assumed to be zero everywhere, however for most state-actions, the reward from Equation (5.10) is negative, this means that during the start of learning, unobserved state-actions appear more rewarding than most of those that have been observed and this would contribute to the exploration of new state-actions until positive reward-gaining state-actions are observed.

The informative exploration strategies reach a middle ground between the two extremes displayed by the ε-greedy sampling strategies. The objective function given by Equation (4.10) places higher weighting on the exploration value early on and so the number of observed state-actions follows closely with the SARSA(λ) exploration profile, particularly in the first 1000 time steps. As learning progresses, the explo-



(a) Average rewards. (b) Observed state-actions.

Figure 5.12: Averaged rewards and observed state-actions for each tested algorithm over each of the 15 trials, 95% confidence intervals are also given at 1000 time step intervals offset by 200 time steps for each set of trials.

ration weight declines and the algorithm favours the learnt reward-gaining trajectories identified in the approximated value function.

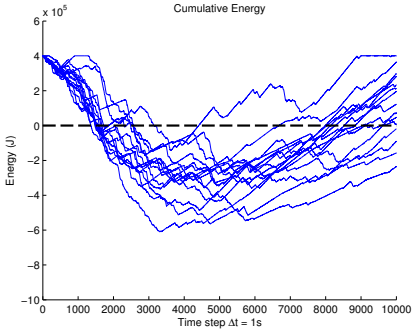
It is also worth discussing the differences in the informative exploration strategy due to the state-action space being either continuous or discrete. The previous three experiments all included continuous states, while the 2D soaring glider operates in a discrete state-action space; this means that each state-action observation of puddle world, cart pole and the battery cycling problem is unique and almost surely irreproducible, whereas samples in the 2D soaring glider problem can be reobserved any number of times. Thus state-actions that have been observed previously have an information gain reward due only to the noise variance hyperparameter and derive no information gain reward from the squared exponential component of the covariance function since the distance to an existing training input is exactly 0. The length-scale hyperparameters continue to dictate the information gain reward for all other previously unobserved state-actions, however due to the discretisation, there is now a step change between the information gain of neighbouring state-actions. This makes reobservations far less appealing than new observations especially during the predominant exploration phase at the start of learning when using the objective function of Equation (4.10). This is advantageous for achieving greater exploration of the state-action

space, however, it does somewhat inhibit revisitation of profitable states while the exploration weight is high, leading to potentially slower learning rates as a trade-off for greater exploration of the state-action space.

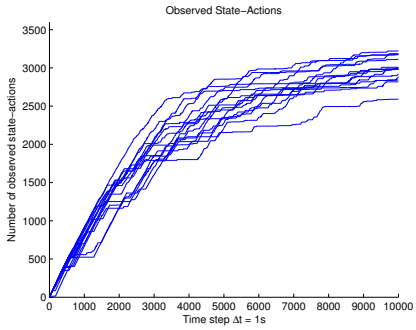
Consider the plots of the cumulative energy and observed states for each trial shown in Figure 5.13. In the plots of the left column, the dashed black lines show the 0 energy threshold. The SARSA(λ) plots in Figures 5.13a and 5.13b show a distinct inverse relationship between the energy and the rate of observing new states. In fact, all SARSA(λ) trials initially drop below the energy threshold before the algorithm discovers a region of energy gaining state-action loops, after which the cumulative energy is replenished and the rate of new state-action observations drops. On the other hand, the GP-SARSA(λ) trials show early discovery and maintenance of energy-positive flight paths with the cumulative energy reaching maximum capacity for many of the trials. Unfortunately, this comes at the cost of poor exploration performance, which is apparent when comparing against the number of observed state-actions of all the other algorithms.

One of the most interesting features of the informative exploration energy profiles is the early peaks in energy gain seen in both the greedy rollout and full rollout cases. These indicate that although the algorithm has discovered energy gaining trajectories, it abandons them in order to explore more of the state-action space. Consequently, the cumulative energy drops and may not recover if the agent is unable to return to these regions later on. This appears to be common in the greedy rollout case, however most of the full rollout trials retain or regain positive cumulative energy throughout the simulation.

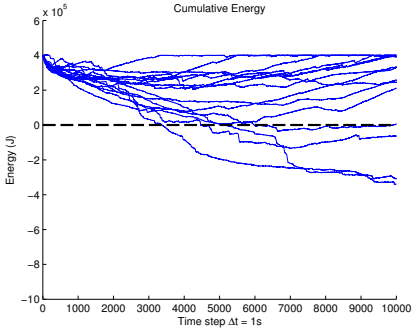
The question that should now be asked is: how can learning be conducted to retain positive energy without compromising the exploration performance of the learning agent? This question will be tackled in the following chapter.



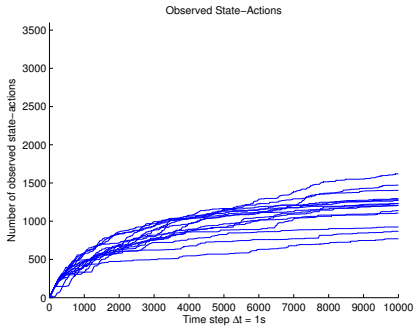
(a) SARSA(λ) cumulative energy.



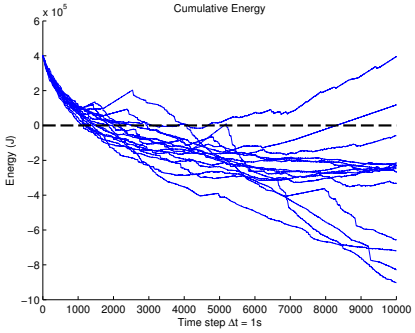
(b) SARSA(λ) observed state-actions.



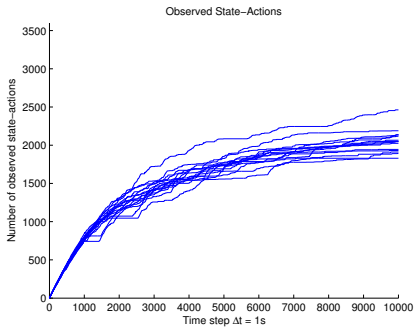
(c) GP-SARSA(λ) cumulative energy.



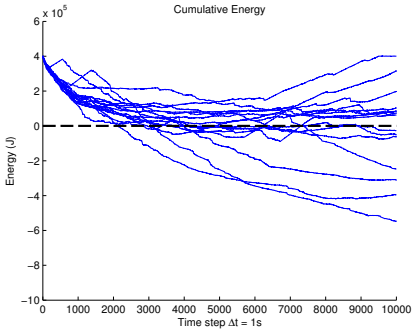
(d) GP-SARSA(λ) observed state-actions.



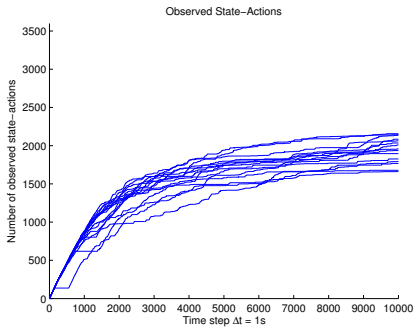
(e) iGP-SARSA(λ) with greedy roll-out cumulative energy.



(f) iGP-SARSA(λ) with greedy roll-out observed states-actions.

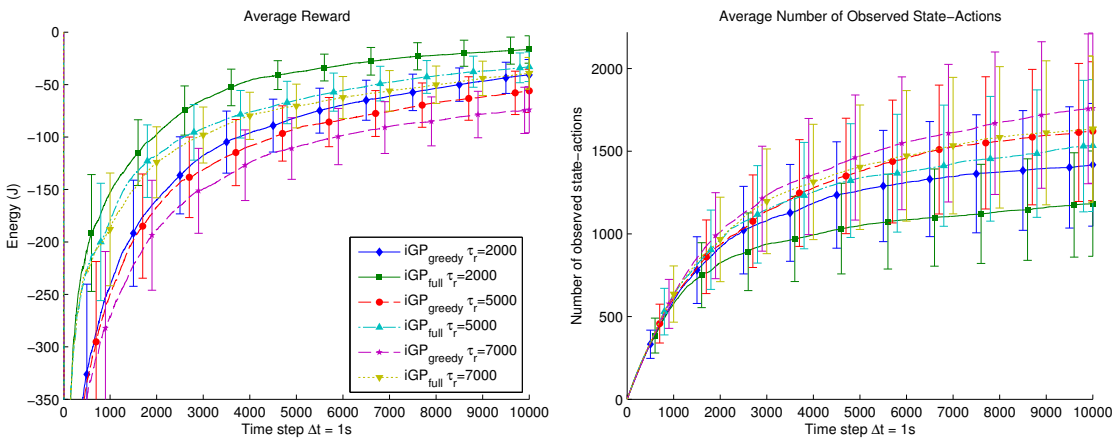


(g) iGP-SARSA(λ) with full rollout cumulative energy.



(h) iGP-SARSA(λ) with full rollout observed state-actions.

Figure 5.13: Cumulative energy and observed state-actions over the course of the simulation for each of the 15 trials. Energies below the threshold value of 0, shown by the black dashed line, represent a critical failure.



(a) Average rewards. (b) Observed state-actions.

Figure 5.14: Averaged rewards and observed state-actions for varying τ_r values over each of the 15 trials, 95% confidence intervals are also given at 1000 time step intervals, offset by 100 time steps for each set of trials.

The effect of τ_r

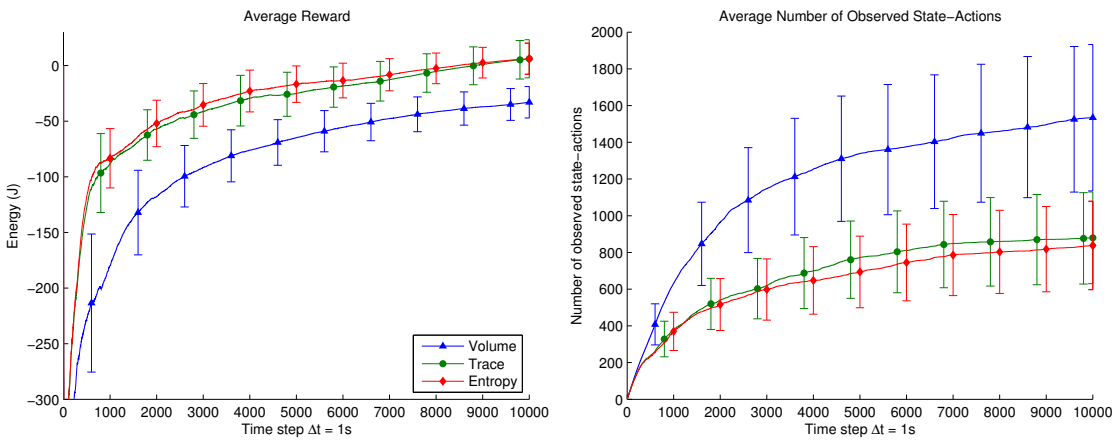
The effect of the information value half-life can be seen in Figure 5.14, these plots compare the progression of the average reward and the number of observed state-actions for varying τ_r values. The exploration performance of all the trials is very similar within the first 1000 time steps, beyond this, the full rollout trials show considerable divergence from one another with the higher τ_r value trials observing new state-actions at a greater rate. The greedy rollout trials of $\tau_r = 7000$ also diverges to achieve the highest average number of state-action observations, while the greedy rollout $\tau_r = \{2000, 5000\}$ trials continue to match in exploration performance until after 2500 steps when the rate of new state-action observations of the lower τ_r value trials declines. In terms of reward-gain performance, trials with lower τ_r values appear to reach higher average rewards faster than those with a higher τ_r value. Full rollout simulations also appear to consistently achieve better reward-gain performance than informative exploration strategies that use greedy rollout, particularly during the early stages of learning.

The trends demonstrated in Figure 5.14 are to be expected since the τ_r value controls the desire to explore through the exploration weighting in the action selection objec-

tive function. Lower τ_r values cause the learning agent to favour exploiting the learnt value function earlier, resulting in higher returns earlier on, while higher τ_r encourages a longer exploration period at the start of learning. In this 2D glider problem, the results suggest that a τ_r value less than 2000 may induce sufficient exploration of the state-action space to discover reward-gaining trajectories. However, for a problem such as the battery cycling problem, where many locally optimal loops exist, more exploration may be required to discover globally optimal solutions. Ultimately, the τ_r value should be chosen to reflect the number of state-action observations required to provide sufficient coverage of the state-action space, where sufficiency is guided by the ability of the learning agent to sample across the space and the proportional size of any reward-gaining regions.

Comparison of information gain measures

The iGP-SARSA(λ) algorithm with $\tau_r = 5000$ was also tested using the covariance matrix trace and differential entropy information measures described in Section 4.2.2. The average reward and number of observed state-actions for each information gain measure are shown in Figure 5.15. Although full information rollout was applied with each of the information measures, the trace and entropy measures produced exploration and exploitation behaviour similar to the GP-SARSA(λ) algorithm with ε-greedy sampling. The average number of observed state-actions for the trace and entropy measures are roughly half that of the GP variance volume simulations. The primary reason for the lag in exploration performance is that these two information values are computed as the discounted sum of the trace or entropy at each of the state-actions in the rollout; this is different to the GP variance volume measure, which computes the information gain across the entire state-action space and not at discrete locations. By limiting the information gain calculation to discrete locations, an element of the nonmyopic exploration behaviour is lost since exploratory actions now only consider the information gain from the next sequence of state-action locations, and not across the entire value function space.



(a) Average rewards. (b) Observed state-actions.

Figure 5.15: Averaged rewards and observed state-actions for different information gain measures over each of the 15 trials, 95% confidence intervals are also given at 1000 time step intervals, offset by 200 time steps for each set of trials.

5.5 Summary

This chapter compared the performance of the iGP-SARSA(λ) algorithm, which has an adaptive, directed and nonmyopic exploration strategy, to that of standard SARSA(λ) learning with ε-greedy exploration. The algorithms were tested on four experiments: puddle world, cart pole, a battery cycling problem and a 2D soaring glider problem. The results from each of these experiments suggest that using GPs for value function approximation can improve reward-gain performance over the TC function approximation method. The GP function approximation updates the value estimate at every state-action whenever a new observation is made, whereas the TC method only updates those features that were observed. Thus, the GP approximation does not suffer from bias due to the initial value estimate, whereas this can be an issue for approximation methods such as TC, which depend on the initial value estimate before the necessary observations are made.

GP-SARSA(λ) with ε-greedy sampling demonstrated good reward-gain performance in both the puddle world and cart pole experiments, however the battery cycling and 2D soaring problem highlighted the need for a directed and nonmyopic exploration scheme to improve coverage of the state-action space. In the 2D soaring problem, the

ε-greedy strategy discovered a reward-gaining loop early on in the learning process and did not explore many other state-actions afterwards. This was also the case for the battery cycling simulations, however, because of the multiple locally optimal loops that exist in this problem, the learning agent was unable to discover the globally optimal trajectory via random exploration. In comparison, informative exploration performed particularly well in the battery cycling problem, iGP-SARSA(λ) using either the greedy rollout or full rollout method was capable of directing exploration beyond these local optima to discover more efficient reward-gaining trajectories.

In general, designing the action selection objective function to actively control exploration and exploitation behaviour requires some knowledge and consideration of the learning agent and learning system. For example, the size and complexity of the state and the ability of the agent to traverse and observe state-actions should be a factor when determining information value decay rates. Traditional exploration methods such as ε-greedy do not require, and therefore do not consider, these meta-properties of the learning problem, simplifying the associated function approximation and sampling computation somewhat. However, as shown in the battery cycling results, some problems require more sophisticated exploration strategies to provide sufficient coverage of the state-action space.

The results presented in this chapter have also shown cases where exploration can jeopardise the ability to continue learning. In the cart pole problem, some actions can trigger transitions into states that are difficult to recover from, causing failures with severe penalties to occur. Since the iGP-SARSA(λ) policy promotes exploration during the early stages of each episode, early termination due to costly failure conditions were common. This was evidenced by the fewer number of cumulative steps for both informative exploration algorithms as compared to GP-SARSA(λ) with ε-greedy sampling. This in turn limited the total number of observed state-actions that were available for value function approximation, degrading the overall performance even further.

The challenge is to develop a learning algorithm that can actively decide when it is safe to explore and when exploitation is required. For this, some measure of "safety"

must be defined and accordingly used to adapt the exploratory or exploitative nature of the action selection. The following chapter addresses this problem in the context of an unpowered aerial glider learning to soar in a wind energy field.

Chapter 6

Resource-Constrained Learning for Autonomous Soaring

This chapter presents an extension of the informative exploration strategy to problems that exhibit resource constraints, specifically, the autonomous soaring problem described in Section 2.1. As was discussed in Chapter 4, the goal of the sampling policy (that is, the action selection function) in the RL problem is to consolidate the two competing objectives of *exploration* and *exploitation* to improve the overall value function estimate and consequently improve the policy. However, given a system subject to resource constraints, such as limited platform energy, the desire to explore or exploit should adapt according to the current available resources. This is of particular importance when the resource of interest constitutes the reward signal and also determines the continuation or (critical) termination of the learning process. Exploration can come at the cost of mission success if accessibility to successful state-actions is very narrow (that is, small deviations may severely affect the final reward) and the agent is only able to take a limited number of steps during learning due to these resource constraints. On the other hand, the agent should continue to explore the state-action space to discover ever more efficient reward-gaining trajectories.

The following subsections propose a weighting function to adaptively preference exploration or exploitation according to the currently available resources. This dy-

namic weighting function replaces the time-step-dependent weighting function of Section 4.4.1 and is used to formulate the sampling policy used in the eGP-SARSA(λ) algorithm described in Section 6.2.

6.1 Resource Limitations

RL research has typically dealt with problems where the single goal is to efficiently find an optimal state-action trajectory for an agent as its number of observations increases to infinity. Limitations on available resources such as platform energy have largely been neglected from such problems in existing research. However, when the RL problem is a resource-seeking mission, the reward and the ability to continue collecting rewards become tightly coupled and such limitations can no longer be ignored.

Resource limitations can be seen as a risk management issue when characterising exploration-exploitation behaviour. Safety considerations for reinforcement learning have been investigated in the work by Schneider (1996), García and Fernández (2012), Moldovan and Abbeel (2012) and others, and these methods consider risk due to uncertainty in the transition model, particularly with regard to actions that may cause the learning agent to experience catastrophic failures. The proposed methods typically require prior knowledge of some "safe" state-action regions and rely on heuristic policies that can drive the agent back into these areas when risky state-actions are encountered. In the autonomous soaring problem described in this thesis, the level of risk is considered to be inversely proportional to the available platform energy, however, to perform flight trajectories to regain platform energy is exactly the learning task of the system. For simplicity, the remainder of this discussion will focus solely on platform energy as the resource and reward of interest.

A soaring glider learning to gather energy from a wind field must expend energy to explore the space for profitable flight trajectories. With no auxiliary energy storage, the glider must store wind energy by increasing its altitude and/or velocity. The instantaneous energy of the glider can be computed as the sum of its kinetic and

potential energies,

$$E_i = \frac{1}{2} m_{glider} V_i^2 + m_{glider} g z_i, \tag{6.1}$$

where m_{glider} is the mass of the glider, g is the acceleration due to gravity, V_i is its current velocity and z_i is its current altitude.

Given the structure of the general objective function in Equation (4.10), the weighting function ω can be thought to describe how optimistic the agent can afford to be in its assumption of uncertain values. For the glider agent, a possible measure of this is its available flight time. The maximum gliding flight time is directly linked to the available energy and is also dependent on the condition that $z > 0$ throughout the flight. Thus, a suitable dynamic weighting function to scale exploratory behaviour according to these factors is,

$$\omega_e = \frac{2}{\pi} \arctan(h_i) \times \max\left(0, \min\left(1, \frac{E_i}{E_{max}}\right)\right)^1, \tag{6.2}$$

where, E_i is the available platform energy at time step i, E_{max} is the maximum amount of energy that can be stored on the platform (for example, corresponding to a maximum speed at a maximum altitude), and

$$h_i = \max\left(0, \min\left(1, \frac{z_i}{z_{max}}\right)\right) \times 100\%. \tag{6.3}$$

The purpose of including h_i is to explicitly restrict exploration in states at low altitudes, regardless of available platform energy (which also includes the altitude in the potential energy component). The altitude penalty is squeezed into an arctan profile between $[0, 1)$ so that for the most part it does not greatly affect ω_e, however at critically low altitudes, it drives ω_e to zero, ultimately restricting the exploration of state-action trajectories that may potentially send the platform into the ground at high speeds. Other weighting functions may also be designed that use different methods to combine these or other factors significant to the specific learning problem,

[1] The max min terms are required for simulation analysis only since it is not expected for energy to drop below 0 or rise above E_{max} in a physical system. This similarly applies to the computation of h_i in Equation (6.3).

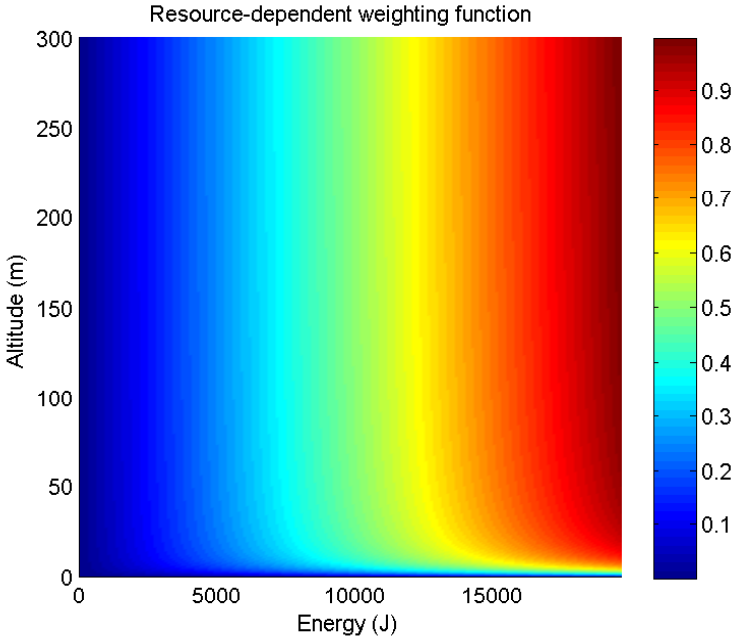


Figure 6.1: The resource-dependent information weighting function from Equation (6.2) evaluated for a range of energy and altitude values. E_{max} and z_{max} are, respectively, the maximum energy and maximum altitude allowable according to the 3D 6DOF soaring glider simulation described in the following chapter. The surface primarily gradates linearly between $0 \rightarrow E_{max}$ for most altitude values but is driven towards 0 for critically low altitudes.

however, the general concept of representing exploration affordability in the context of resource constraints should be maintained.

The weighting factor in Equation (6.2) attempts to consolidate all the principal dimensions of the physical glider state to represent the affordability of exploration on a scale of 0 to 1. This requires some simplifications, which in this case has been to define affordability according to the physical relationship between the glider's altitude and energy and its ability to do work. Certainly, this objective function does not represent the complete picture of the interactions that occur during soaring, but it is sufficient in describing the principal elements in this complex task.

The weighting function is shown in Figure 6.1 and applies the same simulation constants to compute maximum altitude and energy as will be used in the 3D 6DOF soaring glider simulation that will be presented in the following chapter. The surface plot shows the predominantly linear gradation of ω_e across the range of platform en-

ergies, while critically low altitudes have ω_e values shifted towards 0 by the additional arctan scaling component.

The weighting factor in Equation (6.2) is combined with Equations (4.8) and (4.9) to produce the resource-constrained exploration-exploitation objective function:

$$J_{i_e} = \hat{Q}_i + \omega_e \hat{I}_i, \tag{6.4}$$

The effect of this objective function is to increase the influence of the information value when platform energy and altitude is high so that the agent tends to explore areas of the state-action space that have high uncertainty. When platform energy and/or altitude is low, the state-action value dominates the action selection objective function so that the agent tends to exploit areas of the state-action space that are believed to produce high energy reward, thereby replenishing the resource.

6.2 eGP-SARSA(λ)

The energy-weighted action selection objective function is applied directly into the SARSA(λ) algorithm with GP value function approximation. The full algorithm, eGP-SARSA(λ), is shown in Algorithm 2. At each decision instance, the agent must query the available platform energy and current altitude to compute the resource-limited exploration weight using Equation (6.2), this is shown on line 14 and replaces the time-step-dependent weight on line 14 of Algorithm 1.

6.2.1 Computational complexity

For each action selection, Algorithm 2 requires simulation of the motion model and estimation of the resulting information gain across a tree with branching factor in the number of possible actions and search depth imposed by the limit γ_{thres}. It is possible, at each depth, to compute the information gain components of each branch in parallel, reducing the exponential complexity of traversing the tree to a constant

Algorithm 2 eGP-SARSA(λ)

1: $\bar{Q}_a \leftarrow 0$ ▷ Initial value estimate
2: Initialise θ ▷ GP hyperparameters
3: **for** each episode **do**
4: $\mathbf{e} \leftarrow \mathbf{0}$ ▷ Initialise trace
5: $s, a \leftarrow$ initial state and action
6: **for** each step i **do**
7: $e(s,a) \leftarrow 1$ ▷ Replacing traces
8: Take action a, observe reward, r, next state, s', and current energy E_i
9: $\delta \leftarrow r - \bar{Q}_a$
10: **for** all $a^* \in \mathcal{A}(s')$ **do**
11: $\bar{Q}_{a^*} \sim \mathcal{GP}_Q$ ▷ GP approximation for Q
12: $I_{a^*_{total}} \leftarrow I_{gain_0} + \gamma_r I_{gain_1} + \ldots + \gamma_r^p I_{gain_p}$ ▷ Information value
13: **end for**
14: $\omega_e \leftarrow \frac{2}{\pi} \arctan(h_i) \times \max\left(0, \min\left(1, \frac{E_i}{E_{max}}\right)\right)$ ▷ Resource-limited exploration weight
15: $\hat{Q}_{a^*} \leftarrow \frac{\bar{Q}_{a^*}}{\max|\bar{Q}_{a^*}|}$ ▷ Normalised state-action value
16: $\hat{I}_{a^*} \leftarrow \frac{I_{a^*_{total}}}{\max|I_{a^*_{total}}|}$ ▷ Normalised information value
17: $J_{a^*_e} \leftarrow \hat{Q}_{a^*} + \omega_e \hat{I}_{a^*}$
18: $a' \leftarrow \arg\max_{a^*} J_{a^*_e}$
19: $\delta \leftarrow \delta + \gamma \bar{Q}_{a'}$
20: **if** sparsify **then**
21: **if** $\beta_i > \beta_{tol}$ **then** ▷ Linear independence test
22: Append (s,a) to $\mathcal{GP}_Q$ training inputs X
23: **if** $|\mathcal{BV}| >$budget **then**
24: Delete training input with lowest score from Equation (3.24)
25: **end if**
26: **end if**
27: **else**
28: **if** (s,a) is a new state **then**
29: Append (s,a) to $\mathcal{GP}_Q$ training inputs X
30: **end if**
31: **end if**
32: $\mathbf{y} \leftarrow \mathbf{y} + \alpha\delta\mathbf{e}$ ▷ Update $\mathcal{GP}_Q$ training targets
33: **if** retrain hyperparameters **then**
34: $\theta \leftarrow \arg\min_\theta [-\log p(\mathbf{y}|X,\theta)]$ ▷ Minimise the negative log marginal likelihood
35: **end if**
36: $\mathbf{e} \leftarrow \gamma\lambda\mathbf{e}$
37: $s, a \leftarrow s', a'$
38: **end for**
39: **end for**

multiple of the search depth. However, practically speaking, it would be a challenge to mount the number of processors required to maintain this computational speed.

As stated in Section 4.4.2, applying the greedy rollout method on the state-actions considered in the information gain rollout allows the computation of the information value to be linear in the rollout depth. That is, the cost of computing the information value becomes $\mathcal{O}(N^2 tM)$, where the N^2 term arises from the evaluation of the GP covariance, there are t layers in the search tree, and the motion model computation has complexity $\mathcal{O}(M)$. To restrict the complexity even further, it is possible to apply the sparsification method outlined in Section 3.3.2 to the GP approximation of the value function, limiting the complexity to a constant $\mathcal{O}(N_{max}^2 tM)$ where N_{max} is the fixed size of the $\mathcal{BV}$ set used to train the GP.

6.3 Summary

This chapter introduced the concept of resource-constrained exploration in RL. In learning problems where a replenishable resource is required to enable action execution or to avoid undesirable terminal states, exploration should be conducted with consideration of the available resources. The trade-off between exploration and exploitation becomes much more coupled in these situations since exploration enables discovery of new state-actions that can replenish the resource more efficiently, while exploitation of known reward-gaining state-actions builds up resource levels and allows greater exploration of the state-action space.

The motivating problem studied in this chapter was that of an unpowered aerial glider learning to soar in a wind energy field. In this problem, the resource of interest was the glider platform energy, which is depleted during exploratory flight and replenished when learnt energy-gaining flight trajectories are performed. A dynamic exploration weighting function was introduced that can promote exploration when platform energy and altitude is high, and preferences exploitative behaviour when low energy or altitude is detected.

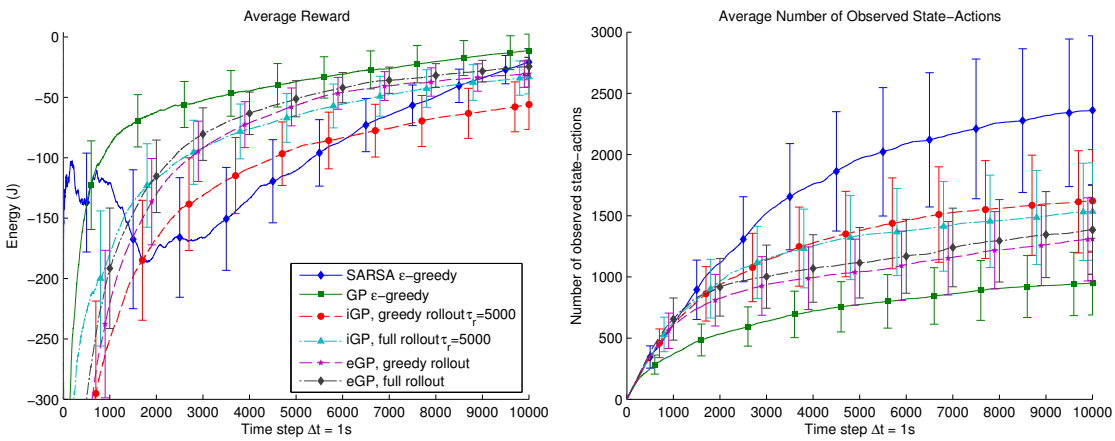
The discussion in this chapter was focused on the autonomous soaring problem, however, this learning approach can be applied to any problem where the reward (or part of the reward) constitutes an expendable resource that is depleted throughout the learning process. For example, the original multi-armed bandit problem can be formulated to consider the actual "monetary" reserves of the agent as an indicator of how many more samples it can take. A robot or a swarm of robots learning to move in an unknown environment to search and track multiple targets could consider time as the resource if the reward is inversely related to the uncertainty of the target locations, which grows over time when targets are unobserved. Alternatively, this can be thought of as a mapping problem where the agents have limited platform energy and the environment contains recharging stations.

Chapter 7

Soaring Simulation Experiments

This chapter revisits the 2D 3DOF soaring glider problem outlined in Section 5.4 and introduces a full 3D 6DOF glider simulation to compare the performance of eGP-SARSA(λ) in Algorithm 2 against the previously tested set of algorithms. The purpose of these experiments is to investigate the ability of the resource-constrained learning algorithm to handle RL problems where the reward and the ability to seek reward are tightly coupled. The setup of the soaring problem includes a reward function that is expressive of the performance of the agent. For both the 2D and 3D experiments, the reward function includes the observed energy gain/loss of the state-action transition, along with discrete penalties for observing certain state-actions. The agent begins each experiment with insufficient energy to explore the entire state-action space for the duration of the episode and must exploit the wind energy field in which it is operating to maintain positive energy. Unlike in the 2D soaring simulations, the full 3D 6DOF simulations terminate the episode if the agent flies outside the defined soaring region or if the glider expends all of its platform energy.

The goal of both learning problems is for the agent to maintain positive platform energy without degrading its exploration performance. As mentioned in Section 5.4.2, the exploration performance of the discrete 2D soaring problem can be measured as the rate of newly observed state-actions. In the 3D soaring scenario, the state-action space is continuous, and so every state-action observation is almost surely unique,



(a) Average rewards. (b) Observed state-actions.

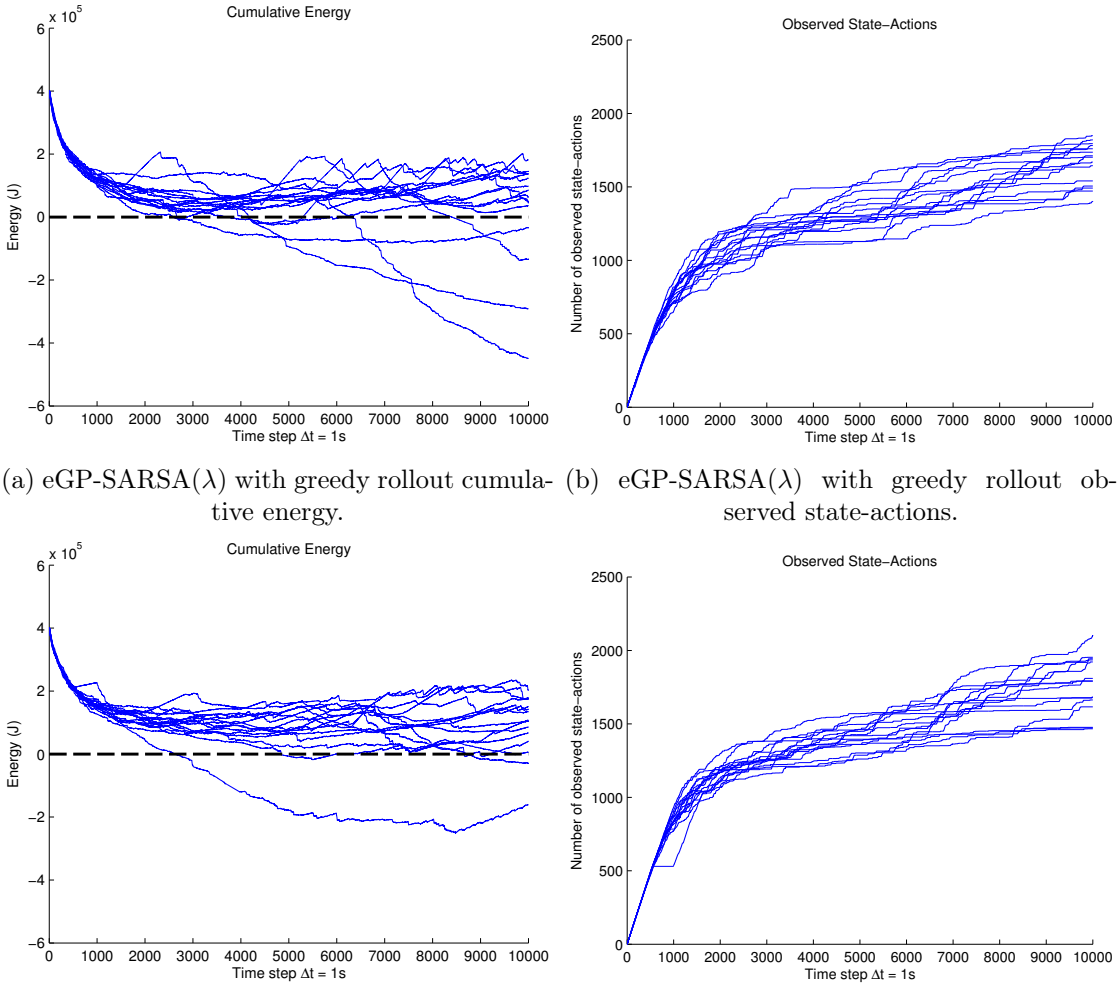
Figure 7.1: Averaged rewards and observed state-actions for eGP-SARSA(λ) over each of the 15 trials as compared to the previously tested learning algorithms, 95% confidence intervals are also given at 1000 time step intervals. eGP-SARSA(λ) has a stronger reward-gaining performance than iGP-SARSA(λ); the number of newly observed state-actions also appears to increase at a constant rate for eGP-SARSA(λ) whereas this value decays for all other algorithms as the number of time steps increases.

thus the cumulative number of steps over all the episodes is used to measure the exploration performance in the 3D simulations.

7.1 2D 3DOF Soaring Glider

7.1.1 Simulation setup

The setup for this set of trials is identical to that of Section 5.4 with the only difference being in the learning algorithm applied to the problem. eGP-SARSA(λ) with both greedy and full rollout was tested, however, with no measure for glider altitude in this 2D problem, the arctan factor was dropped from the resource-constrained exploration weighting in Equation (6.2), leaving a linear scaling of the exploration value according to the current platform energy.



(a) eGP-SARSA(λ) with greedy rollout cumulative energy.

(b) eGP-SARSA(λ) with greedy rollout observed state-actions.

(c) eGP-SARSA(λ) with full rollout cumulative energy.

(d) eGP-SARSA(λ) with full rollout observed state-actions.

Figure 7.2: Cumulative energy and observed state-actions for eGP-SARSA(λ) over the course of the simulation for each of the 15 trials. Energies below the threshold value of 0, shown by the dashed black line, represent a critical failure. The number of observed state-actions exhibits a distinctly different profile from the other tested algorithms shown in Figure 5.13. After an initially strong exploration period of approximately 1000 time steps, the rate of new observations decreases but appears to remain a positive constant as shown by the linear upward trend.

7.1.2 Results

The progression of the average reward and the number of observed state actions for the eGP-SARSA(λ) trials are compared to the results of the previously tested algorithms in Figure 7.1. Both the greedy and full rollout versions of eGP-SARSA(λ) have a stronger reward-gaining performance than iGP-SARSA(λ). Furthermore, the

number of newly observed state-actions also appears to increase at a constant rate for both eGP-SARSA(λ) trial sets whereas this value decays for all other algorithms as the number of time steps increases.

The individual trial results are shown in Figure 7.2. The cumulative energy plots along the left column show periodic energy gain and loss cycles as the learning agent balances the need to explore the state-action space whilst maintaining positive platform energy. Unlike the cumulative energy plots for GP-SARSA(λ) in Figure 5.13c, eGP-SARSA(λ) does not saturate the platform energy and instead is able to use the harvested energy to explore new state-actions. From the spread of the cumulative energy plots, it seems that for the majority of time steps of each trial, the exploration weighting lay below $\frac{1}{2}$ and rarely strayed above this threshold after the initial exploration phase. This suggests that some equilibrium exists around this point above which such exploration is generated that tends to result in energy loss and below which exploitation and energy gain is preferred.

For both the greedy and full rollout information values, eGP-SARSA(λ) is capable of maintaining positive platform energy for a majority of the trials. A comparison of all the algorithms in this regard is given in Table 7.1.

Table 7.1: Number of Trials for which Platform Energy Remained Positive (out of 15 trials)

Algorithm	**Number of successful trials**
SARSA(λ)	0
GP-SARSA(λ)	11
iGP-SARSA(λ), max rollout $\tau_r = 5000$	0
iGP-SARSA(λ), full rollout $\tau_r = 5000$	6
eGP-SARSA(λ), greedy rollout	9
eGP-SARSA(λ), full rollout	11

The number of observed state-actions is given in the right column of Figure 7.2 and exhibits a distinctly different profile from the other tested algorithms shown in Figure 5.13. After an initially strong exploration period of approximately 1000 time

steps, the rate of new observations decreases but appears to remain a positive constant as shown by the linear upward trend. These results show that the resource-constrained learning algorithm, eGP-SARSA(λ), is able to manage available platform energy by exploring new state-actions when energy levels are high and exploiting known energy-gaining trajectories when energy levels are low.

7.2 3D 6DOF Soaring Glider

7.2.1 Simulation setup

The resource-constrained eGP-SARSA(λ) algorithm was also tested on a full 3D 6DOF soaring glider simulation under a static (thermal updraft) soaring scenario. As with the 2D soaring scenario, the reward is the energy gained by the glider agent flying particular trajectories in the wind field; to learn these trajectories, the agent must expend energy to explore the state-action space. The goal of these experiments was to show that given a wind energy field consisting of a single thermal, the proposed resource-constrained learning algorithm is able to effectively explore the state-action space to generate energy-positive trajectories. Unlike the 2D simulation, this full 3D soaring simulation makes no provisions for forcing the glider to remain in the flight boundary or above an altitude of 0 m. Thus, in the early stages of learning, it was expected that many of the trajectories would result in failure or with the glider exiting the field, and so the problem was formulated as an episodic learning task where each episode was terminated either due to the glider crashing or exiting the designated field.

Thermal wind field

The toroidal thermal model is a realistic 3D updraft model that is commonly used in the autonomous soaring literature, see (Bencatel et al., 2013). The toroidal thermal model developed in Lawrance (2011) and shown in Figure 7.3 was the 3D wind field

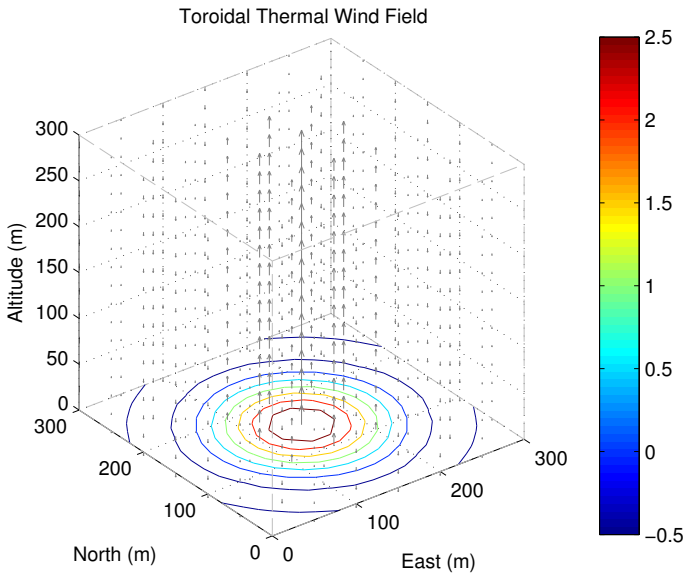


Figure 7.3: Toroidal thermal vector field of a thermal updraft based on the Lawrance (2011) model shown in Figure 2.3. The model is dominated by the central column of rising air, which is offset by the thinner surrounding ring of sinking air. Small lateral wind components also exist outside the flight boundary at the upper and lower extremities of the thermal to maintain the toroidal flow of the air.

used in the following simulations. The thermal model is conservative in its flow, meaning that there is no net horizontal or vertical flow. This is achieved via symmetry in the horizontal flow at the upper and lower extremities of the thermal, and balanced rising and sinking volumetric flow in the vertical plane. In this way, the thermal core of strong rising air is offset by a surrounding column of slow sinking air, while the lateral transitions from rising to sinking air at the top of the thermal are equal and opposite to the transitions from sinking to rising air at the bottom of the thermal.

The shape of the thermal model is determined by three main terms: the core vertical wind speed, $w_{therm} = 3\,\mathrm{m/s}$; the major radius, $R_{therm} = 100\,\mathrm{m}$; and the elliptical factor, $k_{therm} = 100$. The core vertical wind speed is scaled against a sinusoidal function of the thermal radius to form a smooth vertical wind profile. The elliptical factor determines the contribution of the lateral wind components. By choosing a high value for k_{therm}, the thermal model is dominated by the vertical wind flow.

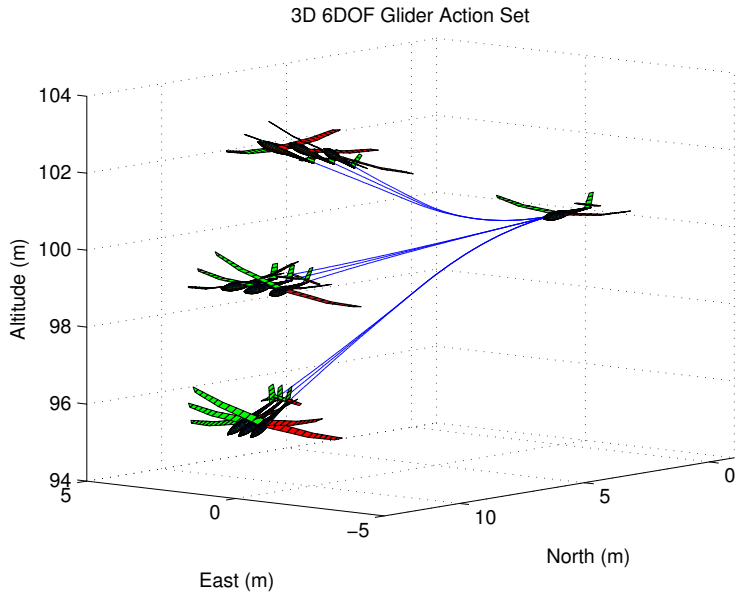


Figure 7.4: Roll and pitch rate action set for a glider travelling straight and level in still air at $V = 12\,\mathrm{m/s}$.

Glider platform

The dynamic model for the glider is a non-linear aerodynamic point mass model based on the RnR SBXC 4.32 m wingspan scale model glider (see Table A.6 in Appendix A.5 for platform specifications). The applied forces are the aerodynamic force (decomposed into lift and drag) and the weight force. Lift is limited by the maximum lift coefficient and load factor constraints. Body force due to sideslip is not considered. A flat Earth model is assumed due to the relatively small scale of the aircraft and flight paths, so the weight force is directed down. Since this is a glider, modelling the effect of wind is an important consideration. This model includes effects for the changes in lift and drag due to the wind and locally linear spatial wind gradients but does not account for moments imparted by differential wind across the lifting surfaces. The model equations can be found in detail in Lawrance (2011). The simulation is performed using numerical integration of the non-linear equations of motion.

Control is modelled using commanded inputs of roll and pitch rates. At each control time step there are three discrete commands available for both roll and pitch, such

that a single action,

$$\mathbf{a} = \left[a_{\dot{\phi}}, a_{\dot{\theta}}\right]. \tag{7.1}$$

This gives rise to a total of nine available actions in the action set shown in Figure 7.4. Intuitively, the roll rate commands result in banking further to the left or right, or maintaining the current bank angle, and similarly for the pitch rate commands which control the aircraft climb angle. Maximum and minimum load factors along with a maximum lift coefficient are defined to restrict the available actions; this simulates an onboard controller that prevents the glider from taking actions that will cause stall conditions. The actions are also limited such that they will not generate commands for bank angles over $\pm 45°$ or pitch angles that would result in a dive steeper than $50°$ so that all actions in the action set should always be achievable.

In the state-action value function approximation, the pitch and roll commands were regarded as separate dimensions in the training input space since they generated responses in orthogonal axes of the aircraft's body frame. As such, no meaningful length scale or metric could be defined to measure the "difference" between banking and pitching, and so it was left to the GP to model the interactions between these commands and the expected value.

Reward function

The reward function used was based on the specific energy gained by the platform during each state-action transition, penalties were also applied when the glider stalled (that is, if airspeed fell below stall speed) and if the glider dropped below an altitude of 0 m. While the former condition was recoverable, the latter indicated a critical failure, thus the reward function was designed to reflect the severity of these states. The penalty incurred for stalling was computed as 25% of the specific kinetic energy at stall velocity,

$$r_{stall} = -25\% \times \frac{1}{2} V_{stall}^2, \tag{7.2}$$

while the penalty for crashing (altitude $\leq 0\,\mathrm{m}$) was defined as twice the maximum allowable specific energy in the flight envelope,

$$r_{crash} = -\frac{2E_{max}}{m_{glider}}, \tag{7.3}$$

where E_{max} corresponds to the platform energy when flying at the maximum allowable speed and altitude, m_{glider} is the platform mass and V_{stall} is the stall velocity of the platform. The complete reward function was computed as,

$$r_i = \frac{E_i}{m_{glider}} + stall \times r_{stall} + crash \times r_{crash}, \tag{7.4}$$

where *stall* and *crash* are boolean indicators for the two respective conditions.

Relative states

To use the full glider state space would require learning over 13 state dimensions in addition to the 2 action dimensions, rendering the learning task infeasible given the prohibitively large number of points required to generate a reasonable GP model of the entire state-action space. The problem can be simplified if it is assumed that the thermal centre is known. The states of interest, that is, the primary contributors to the reward function by virtue of wind field geometry, can therefore be defined as the distance to the thermal centre, the bearing to the thermal centre, and the airspeed:

$$\mathbf{s} = [d_{therm}, \psi_{therm}, V]\,. \tag{7.5}$$

The value function then becomes a function of the states and actions presented in Equations (7.1) and (7.5).

The full glider state is still maintained since it is required in the rollout (where the state transition is performed under the assumption of no wind) and the reduced relative states are not Markovian in this regard. In fact, without full information of the wind field, the full glider state is also not strictly Markovian, however this algorithm relies on the smoothness in the GP model to handle variability in the

states introduced by the unknown thermal wind field.

It is noted that although the r_{crash} component of the reward is a function of the altitude, this variable is not included as a dimension in the relative learning state. In fact, the event of crashing is linked to the climb angle and airspeed of the aircraft, and these two variables are themselves linked through the glider dynamics shown in Lawrance (2011). In the results given in the next section, it is shown that the RL algorithm learns to avoid high airspeeds since there is a strong correlation between this and receiving a strong negative reward due to r_{crash}.

7.2.2 Results

The following five learning algorithms were tested:

- eGP-SARSA(λ) with full rollout information value,
- iGP-SARSA(λ) with full rollout information value, $\tau_r = 100$,
- eGP-SARSA(λ) with greedy rollout information value,
- sparse eGP-SARSA(λ) with full rollout information value, 1000 $\mathcal{BV}$, $\beta_{tol} = \sigma_n^2$ (outlined in Section 3.3.2), and
- GP-SARSA(λ) with ε-greedy exploration, $\varepsilon = 0.01$.

Each algorithm was run over 10 trials, each with a learning period of 50 episodes. The learning agent began each episode in a random location in the field with a bearing of 0° to the thermal centre, these starting locations were consistent across all the tested learning algorithms.

Reward and energy gain performance

The progression of the average reward across the episodes for each tested algorithm is shown in Figure 7.5; the average of each set of 10 trials is plotted along with 95%

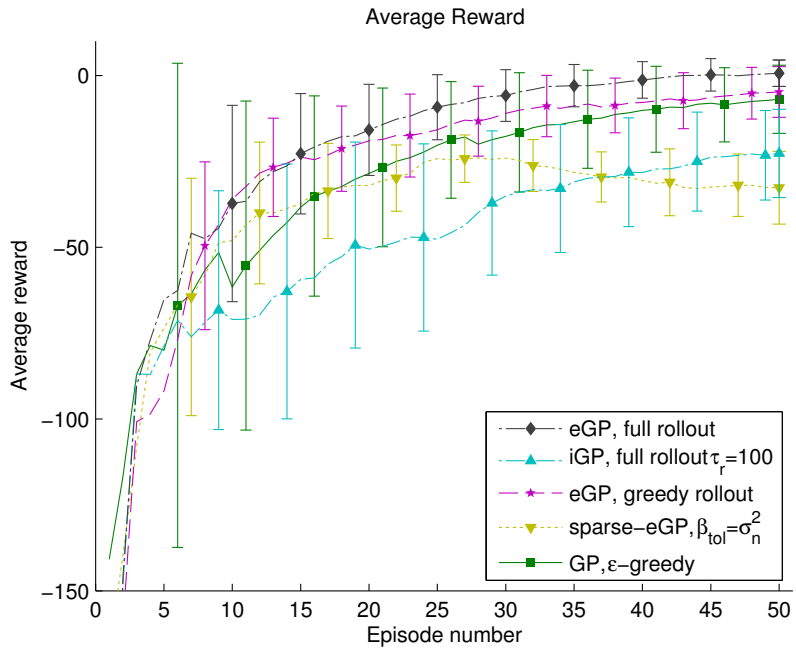


Figure 7.5: Progression of the average reward across the episodes. The plots compare the performance of eGP-SARSA(λ) with full rollout, iGP-SARSA(λ) with full rollout, eGP-SARSA(λ) with greedy rollout, sparse eGP-SARSA(λ) with $\beta_{tol} = \sigma_n^2$, and GP-SARSA(λ) with ε-greedy sampling. The average of each set of 10 trials is plotted along with 95% confidence intervals at 5 episode intervals, the episodes plotted for each algorithm are offset by 1 episode each to improve clarity, however the data at episode 50 are shown for all the tested algorithms.

confidence intervals at 5 episode intervals, the episodes plotted for each algorithm are offset by 1 episode each to improve clarity, however the data at episode 50 are shown for all the tested algorithms. The graph shows that both the greedy and full rollout eGP-SARSA(λ) algorithms promptly converge to a higher average reward and are followed closely by GP-SARSA(λ) with ε-greedy sampling, however the latter has a larger confidence interval than either of the eGP-SARSA(λ) cases. The iGP-SARSA(λ) trials produce the lowest initial average rewards and also display a high level of variation between the rewards gained in each simulation. The graph also shows the departure of the sparse-eGP approach in terms of reward-gaining performance as the training input set is reduced during the early episodes. Indeed, the average reward achieved begins to decrease during the later episodes and even drops below that of iGP-SARSA(λ) after episode 37. This suggests that the size of $\mathcal{BV}$ may be insufficient in representing the full state-action space or that the linear independence metrics used

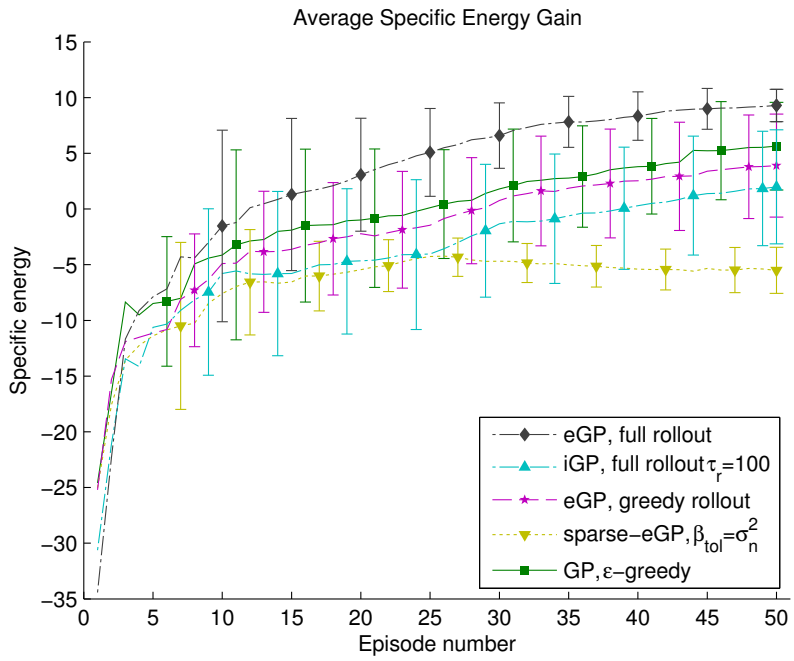


Figure 7.6: Progression of the average specific energy gain across the episodes and averaged for each simulation case, 95% confidence bounds are given at 5 episode intervals.

to reject and discard observations from $\mathcal{BV}$ are inadequate for this learning task.

The progression of the average specific energy gain across the episodes is shown in Figure 7.6 and the total specific energy gain for each episode is shown in Figure 7.7. The average specific energy profiles are similar to the average reward profiles, since a large portion of the reward is derived from the energy gain, however, the reward also includes the r_{stall} and r_{crash} components. Looking purely in terms of energy gain, it can be seen that the sparse-eGP case performs the worst of all the tested algorithms, with iGP-SARSA(λ) over-taking its energy-gaining performance much earlier in episode 5. The average specific energy gain per step for the sparse-eGP case dropped to $-5.5\,\mathrm{J/kg}$ by the final episode; to place this value in context, it is approximately the specific drag induced by the glider flying straight and level at a speed of $15.0\,\mathrm{m/s}$ over one time step. Furthermore, the slight decline in energy gain performance from episode 26 onwards cannot entirely account for the much steeper decline in reward gain performance seen in Figure 7.5; this suggests that the agent experienced more *stall* and/or *crash* events during the later episodes.

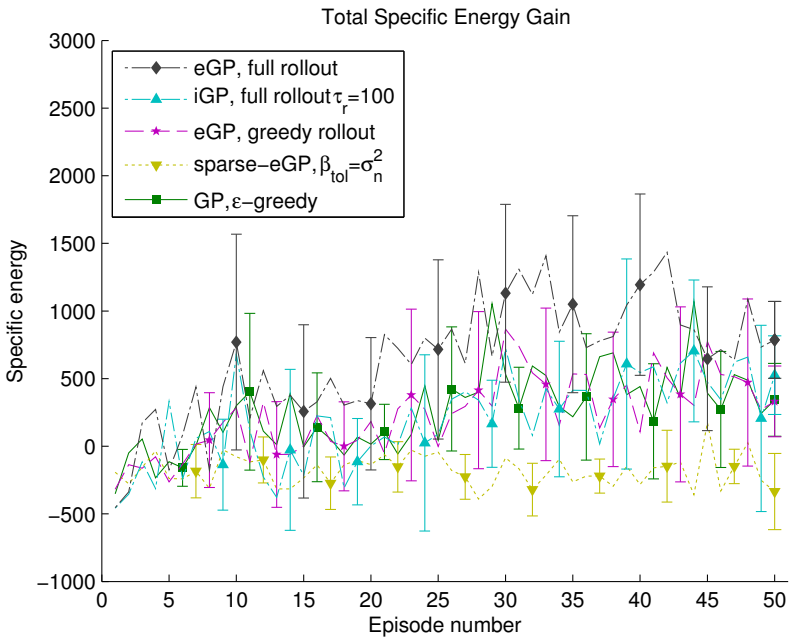


Figure 7.7: The total specific energy gain averaged for each simulation case at each episode, 95% confidence bounds are given at 5 episode intervals. In the eGP-SARSA(λ) simulation with full rollout information gain, once the learning agent observes an episode of large energy gain, it is able to continue executing energy-gaining trajectories with relative consistency.

Aside from sparse-eGP, all other learning algorithms were able to achieve a positive average specific energy gain by episode 50. Figure 7.6 shows that GP-SARSA(λ) with ε-greedy sampling achieves a higher average specific energy gain than eGP-SARSA(λ) with greedy rollout despite the latter achieving a greater average reward in Figure 7.5. Both algorithms have much lower average energy gain profiles than eGP-SARSA(λ) with full rollout information value. Furthermore, the eGP-SARSA(λ) with full rollout has a more consistent energy gain performance as shown by the significantly smaller confidence intervals.

The total specific energy gain averaged over all the trials also provides an interesting perspective on the relative performances of each algorithm. The total energy gain of the first 15 episodes is very similar for all five test cases, however, the results diverge into three groups beyond this point. On average, sparse-eGP generated trials with overall energy loss; the GP-SARSA(λ), eGP-SARSA(λ) with greedy rollout and iGP-SARSA(λ) with full rollout trials were able to produce positive energy gain trajectories; however, eGP-SARSA(λ) with full rollout information value consis-

tently achieved a higher total energy gain for each episode over all the other tested algorithms.

Exploration management

The preceding three plots in Figures 7.5 to 7.7 have summarised the energy- and reward-gain performance of the tested algorithms. The results show a clear performance advantage of eGP-SARSA(λ) with the full rollout information value in terms of efficient reward- and energy-gain for this soaring glider problem. GP-, iGP- and eGP-SARSA(λ) with greedy rollout have shown comparable performances, however the greedy rollout algorithm is able to retain a higher average reward despite achieving a lower average energy gain.

The reason for the disparities between Figures 7.5 and 7.6 comes down to the number of *crash* incidents each algorithm experiences. While *stall* events also lower the average reward, it is the heavy cost of crashing that has the greatest influence over the average reward. The termination statistics for all 10 trials of each algorithm are shown in Table 7.2. A successful termination is defined as one in which the glider gains enough energy to soar out through the upper flight boundary. Although eGP-SARSA(λ) with greedy rollout and GP-SARSA(λ) have similar termination statistics, the 4 fewer *crash* incidents of eGP-SARSA(λ) with greedy rollout boost its reward-gaining performance above that of ε-greedy, despite having slightly fewer successful terminations. The iGP-SARSA(λ) terminations help to explain its poor reward-gaining performance. Although iGP-SARSA(λ) has the second highest number of successful terminations, 141, it also has the highest number of *crash* terminations at 77. These termination statistics show that eGP-SARSA(λ) with full rollout performs the best overall in terms of successful flights. Over half of all the episodes for full rollout eGP-SARSA(λ) terminated with the glider gaining enough energy to soar out through the upper flight boundary. Although it also experienced 33 crash terminations, this figure only represents 12.9% of its successful terminations, as compared to 54.6% for iGP-SARSA(λ), 23.1% for eGP-SARSA(λ) with greedy rollout, 725% for sparse-eGP, and 26.8% for GP-SARSA(λ) with ε-greedy sampling.

Table 7.2: Episode Termination Statistics

Algorithm	**Successes**	**Crash terminations**	**Lateral boundary termination**
eGP-SARSA(λ), full rollout	256	33	211
iGP-SARSA(λ), full rollout	141	77	282
eGP-SARSA(λ), greedy rollout	78	18	404
sparse-eGP, $\beta_{tol} = \sigma_n^2$	8	58	434
GP-SARSA(λ), ε-greedy	82	22	396

The poor performance of sparse-eGP is attributed to insufficient or inappropriate training input points that cannot effectively approximate the value function. Table 7.3 presents the average number of observations and the final size of the training input set for each simulation case. On average, sparse-eGP observes far fewer state-actions over a single trial and tends to discard approximately 34% of these observations according to the linear independence test of Equation (3.21). The linear independence test used to reject observations from $\mathcal{BV}$ uses the squared exponential covariance function and therefore favours observations taken in isolated state-action locations. However, for the problem of learning to soar in a static thermal, energy-gaining flight can only occur in a local region around the thermal centre where it is necessary to have a more refined policy. In regions further away from the thermal centre, a reasonably coarse policy can be applied to direct the glider towards the energy-gain regions. Therefore, a uniform density of training inputs over the entire state-action space may not provide as good a model for learning the policy as compared to a distribution of inputs that concentrate observations in the regions of interest (for example in state-action areas that lead to *crash* or *stall* conditions, or near state-actions that generate positive reward). In addition to this, the low number of observations has a cumulative effect since fewer observations leads to poorer estimation of the value function and thus a poorer sampling policy that is more likely to lead to early termination. This is supported by the statistics in Table 7.2 that show sparse-eGP with the lowest number of successful terminations.

Table 7.3: Average Number of Observations and Training Inputs

Algorithm	**Average number of observations**	**Average $\mathcal{BV}$ size**
eGP-SARSA(λ), full rollout	3448	3448
iGP-SARSA(λ), full rollout	2702	2702
eGP-SARSA(λ), greedy rollout	2118	2118
sparse-eGP, $\beta_{tol} = \sigma_n^2$	1264	829
GP-SARSA(λ), ε-greedy	2072	2072

In comparison, all the other tested learning algorithms are, on average, able to achieve over 2000 observations during each trial. Moreover, they retain all observations for use in training the GP approximation of the value function. eGP-SARSA(λ) with the full rollout information value has the highest average number of observations in each trial. This demonstrates that the resource-constrained exploration strategy is able to manage platform energy throughout the flight to enable consistently longer trajectories in each episode, as shown in Figure 7.8. Consequently, the agent visits more state-action locations and is able to learn a better model of the value function.

The curves plotted in Figure 7.8 are consistent with the successful termination statistics given in Table 7.2, however it is interesting to observe the progression of the sparse-eGP curve, which initially follows the iGP-SARSA(λ) plot but diverges rapidly after episode 25. Greedy rollout eGP-SARSA(λ) and GP-SARSA(λ) with ε-greedy sampling have very similar profiles that, from the very first episode, are slower than full rollout eGP-SARSA(λ) and iGP-SARSA(λ) to accumulate steps. The reason for the initial divergence between the full and greedy rollout can be attributed to the poorly modelled value function at the start of learning. While the full rollout accounts for the information gain of all the possible future state-actions, the greedy rollout only considers the information gain of those state-actions it expects to visit according to the current value function estimate. Since the value function is poorly modelled at the start of learning, the greedy rollout information value consequently computes a poor estimate of the information value.

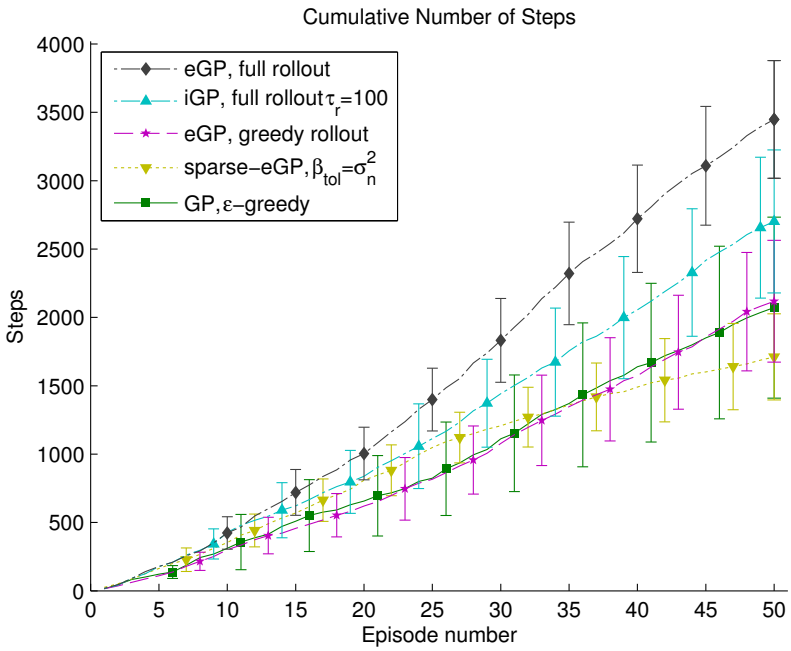


Figure 7.8: Comparison of the cumulative number of steps over the episodes and averaged for each simulation case, 95% confidence bounds are given at 5 episode intervals.

Learnt value function

The learnt value function for one trial of full rollout eGP-SARSA(λ) at 5 m/s velocity intervals from stall velocity up to the maximum allowable airspeed is shown in Figure 7.9. The plots show the estimated state-action value for the best set of turning actions, $[a_{\dot{\phi}}, a_{\dot{\theta}}]$, at each state. As expected, higher values are predicted for states close to the thermal centre; at states further away, bearings closer to 0° produced higher values. The wraparound condition enforced by the GP distance measure can also be observed along the bearing axis.

There is a distinct change from positive values to negative values as the airspeed increases, showing the agent learned that the event of crashing, which generates a strong negative reward from r_{crash}, tended to occur at high velocities. In this way, the two negative reward components, r_{crash} and r_{stall}, effectively placed an upper and lower bound on the glider airspeed.

It was expected that the value function would be symmetrical in the bearing axis since, if all else was equal, circling clockwise or anticlockwise around the thermal

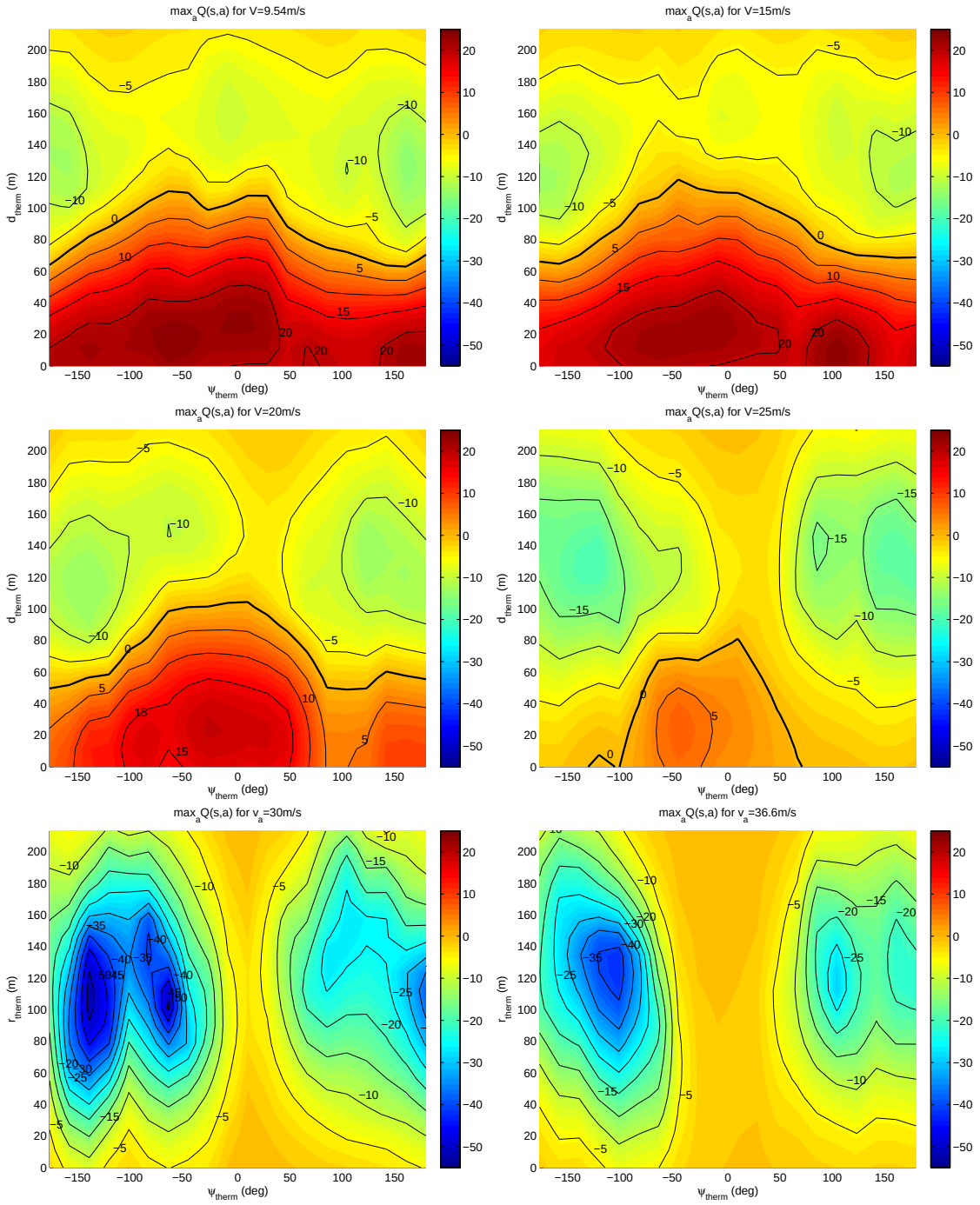


Figure 7.9: The full eGP-SARSA(λ) value function for one trial evaluated at the best action for each state. There is a distinct transition into negative values at $V = 30\,\mathrm{m/s}$, this is because the slowest airspeed for which the agent experienced a *crash* event was $V = 30.9\,\mathrm{m/s}$. The strong negative reward signal observed at this state-action caused a steep reduction in the values around this airspeed. The values plotted for $V = 36.6\,\mathrm{m/s}$ appear to improve upon those at $V = 30\,\mathrm{m/s}$, however, this is due mostly to the fewer observations made at and around the higher airspeed.

should give equivalent energy gain profiles. Figure 7.9 gives a hint of this symmetry, however, it appears that during this particular learning trial, the agent favoured making anticlockwise loops around the thermal centre. This may be due to the fact that the agent travelled in an anticlockwise trajectory during the first episode in which it successfully gained enough energy to soar out through the upper boundary (see Figure 7.10), thereby creating an initial bias for later trajectories.

Flight trajectories

Successive flight trajectories from one trial of the full rollout eGP-SARSA(λ) simulations are shown in Figure 7.10. These plots show the steady progression of the glider agent learning to gain energy from the thermal wind field. In the first episode, the actions executed by the agent are unsuccessful in gaining energy and the episode ends with the glider exiting the lateral boundary of the field. This is to be expected since very few observations of the state-action have been collected at this point and so the GP estimate of the value function is still highly uncertain. In episode 14, the glider performs a single loop that traverses near the centre of the thermal where energy gain potential is high before once again exiting the lateral flight boundary.

Episode 20 is the first instance in which the agent consistently performs loops around the thermal centre and gains enough energy throughout the flight trajectory to allow the glider to soar out through the upper boundary of the field. The trajectory in this episode is not a smooth rise, nor does it have a smooth turning motion, and this would result in higher energy losses due to increased drag. The trajectories of episodes 27 through to 50 show the glider moving efficiently towards the thermal centre at the start of flight when energy is lowest and rapidly gain altitude via thermal soaring. The flight paths show a general trend towards smoother and more regular circling, representing more efficient energy gain paths.

The effect of the resource-constrained informative exploration strategy can also be seen in the flight trajectories that occupy the upper regions of the flight boundary. At higher altitudes, the agent has gained significant potential energy from the thermal

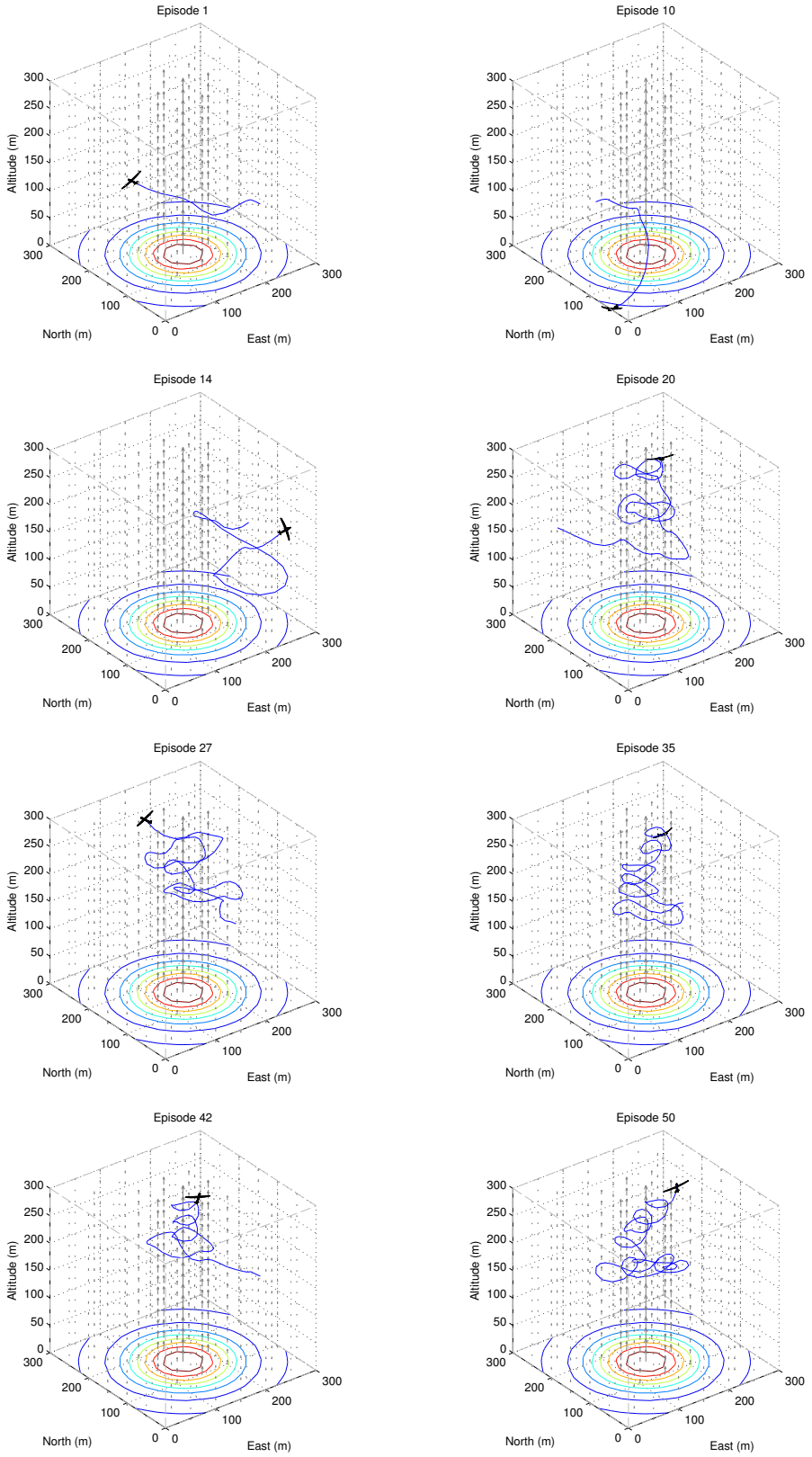


Figure 7.10: Evolution of the learnt flight path for one trial of the eGP-SARSA(λ) case. The glider operates in a flight boundary defined by a $300 \times 300 \times 300\,\mathrm{m}^3$ volume, in these figures, the glider is enlarged by a factor of 10 so that it can be seen more easily. During the first episode, the glider performs largely random actions; in episode 20 the agent successfully gains enough energy to exit the field via the upper boundary; the flight path of episode 50 shows a consistent circling trajectory.

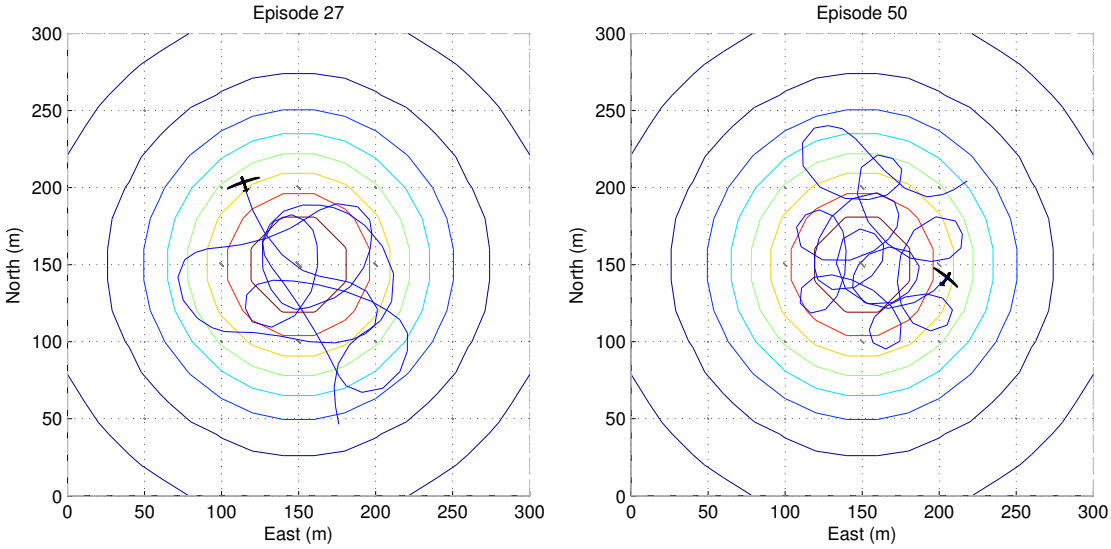


Figure 7.11: Aerial view of episodes 27 and 50, the glider has been enlarged by a factor of 5 so that it can be seen more easily. These paths illustrate the exploration behaviour generated by eGP-SARSA(λ). In episode 27, the agent explores across the range of radial distances to the thermal centre, while in episode 50 the aerial view shows that the agent flight path resembles a superhelix that samples over a range of bearings and radial distances.

and can afford to explore more of the state-action space. It does so by morphing the trajectory helix, which allows it to sample new regions of the state-action space while maintaining proximity to known energy gain transitions. The aerial view of the flight trajectories generated in episodes 27 and 50 are shown in Figure 7.11. In episode 27, the glider traverses a range of radial distances as it gains more platform energy, and in episode 50, the flight path resembles a superhelix, which is able to rapidly sample a range of radial distances and bearings.

7.3 Summary

This chapter analysed the performance of the resource-constrained informative exploration strategy of eGP-SARSA(λ) when applied to the problem of an unpowered aerial glider learning to soar in a wind energy field. The algorithm was first tested on the 2D 3DOF glider simulation from Section 5.4 and then extended to a full 3D 6DOF simulation with a single thermal updraft energy source. The proposed algorithm was

compared against the basic GP-SARSA(λ) with ε-greedy exploration algorithm and the iGP-SARSA(λ) algorithm presented in Section 4.5. Both the greedy and full rollout versions of the resource-constrained informative exploration strategy were tested as well as a sparse-eGP implementation of the value function approximation (using the full rollout information value).

The 2D 3DOF simulation results demonstrated the ability of the resource-constrained exploration strategy of eGP-SARSA(λ) to maintain positive platform energy while continually exploring new state-actions. The exploration management scheme was able to exploit wind energy sources when platform energy was low and explore unobserved state-actions when platform energy was high.

The 3D 6DOF simulation results presented in this chapter show that eGP-SARSA(λ) using the full rollout exploration value achieved the highest average reward and highest average energy gain over all other tested algorithms. It produced the greatest number of successful episodes where the glider was able to capture enough energy from the wind to soar out through the upper boundary of the wind field. The dynamic resource-constrained exploration weighting was able to manage exploration and exploitation behaviour to generate longer learning trajectories, which explore more state-action transitions without compromising overall reward- and energy-gain performance. Successive flight trajectories of a single trial were provided and showed the evolution of the learnt policy. The learnt flight paths were dominated by a helix occupying the region close to the thermal centre, but also included evidence of morphing in the upper regions of the flight boundary when the resource-constrained informative exploration strategy increased exploration priority.

Chapter 8

Conclusion and Future Research

The problem of an unpowered aerial glider learning to soar in a wind energy field encapsulates an aspect of the exploration-exploitation trade-off that has not previously been addressed in the RL literature. In this soaring task, the learning agent must expend energy to explore for more efficient energy-gaining trajectories whilst simultaneously managing the need to exploit known flight paths to maintain altitude and airspeed. Unlike in other typical RL applications, the learning agent in this resource-constrained problem cannot afford to explore the value function space exhaustively and must manage its learning behaviour to accommodate for these constraints.

This thesis presented an informative exploration strategy for use in RL problems, such as the soaring glider problem, where the trade-off between exploration and exploitation is driven by resource constraints. The proposed algorithms, iGP- and eGP-SARSA(λ), actively direct exploration to regions of the state-action space where uncertainty of the value function estimate is high. The exploration strategy developed here is nonmyopic in that the effect of future observations is considered when assessing the exploration utility of potential actions. Furthermore, the algorithms adaptively manage the exploration-exploitation behaviour of the learning agent according to the available resources, whether it be learning time or a physical resource such as platform energy. Section 8.1 summarises the contributions of this thesis and Section 8.2 gives suggestions for future research directions.

8.1 Summary of Contributions

Derivation of a new information measure

A new information measure was derived based on the change in the GP variance volume due to a new training input. An analytical solution to the variance volume of a GP with a squared exponential covariance function was presented. Furthermore, the required covariance function integrability condition that guarantees an analytical solution was also defined. This information measure was compared against existing measures, namely the differential entropy and the trace of the covariance matrix, and was shown to be consistent with both.

Unlike the differential entropy or trace measures, which can only be applied to a discrete set of sample points, the variance volume information measure can capture the reduction in uncertainty over the entire continuous GP query space. Furthermore, the differential entropy and trace measures induce computational overheads due to the additional matrix computations required for solving $K\left(X_*, X_* | X_N\right)$, these are proportional to the number of discrete query points. On the other hand, the analytical solution to the variance volume information measure is not susceptible to sampling resolution and involves no such additional overheads.

Development of a nonmyopic information value

A state-action rollout mechanism was introduced to compute the nonmyopic information value as the discounted sum of potential future information gain. The information *reward* is the expected uncertainty reduction over the value function estimate due to a single observation, thus the rollout ensures that the information *value* not only accounts for immediate information gain rewards but also represents the potential for future information gain. The rollout method was inspired by the definition of the RL state-action value and the eligibility trace of TD(λ). Two types of rollouts were presented: the full rollout information value sums the discounted information gain of the complete set of state-actions that are reachable within a rollout threshold, whereas the

greedy rollout information value collects only the expected state-action observations according to the current sampling policy. The benchmarking results of this thesis showed that using the nonmyopic information value to direct exploration improved learning rates over an ε-greedy exploration strategy, particularly when dealing with state-action value functions with multiple local minima.

The information value can be used to direct exploration in problems for which choosing one action over another changes the forward reachable set of state-actions in the n-step future, that is, problems that deal inherently with state-action *trajectories*. Thus, computing the nonmyopic information value requires a state transition model to determine the state-actions reachable at each rollout depth. For a setup such as the multi-armed bandit problem where each action returns the agent to the original state with the same set of available actions, the nonmyopic information value will not provide any benefit over using the myopic information gain reward. This is because the entire state-action space is reachable at each step regardless of the action taken. The benefit of the nonmyopic information value lies in being able to discriminate the future exploration opportunities of the immediately available actions.

Experiments in this thesis have used the nonmyopic information value to direct exploration in an RL context. However, this method can be applied more generally to informative path planning problems where the goal is to take observations along a trajectory to reduce the uncertainty of a model.

An exploration-exploitation management scheme for resource-constrained learning systems

A dynamic action selection objective function for adaptive exploration-exploitation management was presented in this thesis. The proposed objective function computes a dynamic resource-constrained exploration weighting that represents how optimistic the agent can afford to be regarding uncertain value estimates given the current resource levels. In this way, the policy preferences actions that have large exploration values when resource levels are high, and reverts to exploitative actions according to

the current value function estimate when resources are low. This management scheme was tested on a preliminary 2D glider simulation and a full 3D 6DOF resource-limited soaring glider mission and was shown to be capable of maintaining required resource levels without compromising the exploration performance.

While this thesis focused on the autonomous soaring problem as an example of a resource-constrained learning system, the proposed learning approach is applicable to any problem where the reward can be formulated as an expendable resource that is depleted throughout the course of learning. For example, a literal implementation of the multi-armed bandit problem can be considered a resource-constrained learning system, where each action induces a base cost and the total monetary return for all actions determines the ability of the agent to continue sampling. A robot or a team of robots performing search and track over multiple targets can consider time as the resource if the reward is inversely related to the uncertainty of the target locations, which grows over time when targets are unobserved. Other derivations of energy-constrained exploration problems also exist, such as robots with limited platform energy performing a mapping task in a large environment containing recharging stations. The exploration-exploitation management scheme presented in this thesis can also be applied to enable resource-constrained action selection for these problems.

8.2 Future Research

Convergence analysis of GP-SARSA(λ)

This thesis used a GP to approximate the RL value function with the assumptions that the GP model provides a good estimate of the value function and that the estimate improves as more observations are used to update and train the GP. An analysis of the convergence properties of GP value function approximation would give greater insight and perhaps provide guarantees on the quality of this approximation method. While GPs have a representation as a linear function approximator, there are a number of key differences between it and other standard linear function approximators that

violate the assumptions required for the value function convergence proofs given by Tsitsiklis and Van Roy (1997). For example, Lemma 7 of Tsitsiklis and Van Roy (1997) assumes that the number of basis functions in the linear function approximator is a finite constant, however in a GP model, a new basis function is added for each new training input included in the $\mathcal{BV}$ set, and the size of this set can increase to infinity (albeit only when the number of training inputs increases to infinity). If a sparsification method, such as the one by Csató (2002) that is applied in this thesis, is used to limit the size of $\mathcal{BV}$, then futher investigation must be sought over the effect of its update method on the overall value function convergence properties.

Sparsification methods for online GP-SARSA(λ)

The GP sparsification method used in this thesis applied a novelty threshold for determining which observations to include in the $\mathcal{BV}$ set and which to reject, however results showed that this measure may be inadequate for the learning problems that were considered in this thesis. To reiterate Section 8.2, further research is warranted to analyse the effects of the sparse update on the value function approximation properties. Other sparsification methods that can maintain a conservative approximation of the value function may also be investigated to reduce the overall computational complexity of the approximation and enable online on-platform applications.

Safety guarantees for aerial soaring

The presented learning algorithm is able to incorporate resource constraint considerations into the action selection, however it provides no strong guarantees for remaining within the resource budget when performing exploration, as evidenced by the early trajectories when the glider exited the field or crashed. It is perhaps an understatement to say that episode termination due to a crash is undesirable in a flight trial, and so safety guarantees must be provided to ensure safe learning conditions. Future research will involve generating robust policies with known safe routes to avoid breaching hard resource constraints.

Soaring in multi-feature wind fields

This work aims towards developing practical solutions for guidance of a gliding aircraft in large unknown wind fields consisting of multiple types of wind features. Soaring flight in unknown wind fields is a high-dimensional problem, and the solution presented in the current work addressed this issue by identifying a lower dimensional feature set to represent a known wind feature. The work presented here does not attempt to estimate the parameters of the wind field and has not addressed any robustness issues associated with imperfect knowledge of the thermal parameters. Future work will examine hierarchical methods to autonomously identify lower dimensional feature spaces and consider how multiple policies could be learnt and then integrated with connecting approach and exit trajectories, such as those in Woodbury et al. (2014), to achieve an adaptive global policy for exploration of unknown spaces with resource constraints.

Bibliography

Zsuzsa Ákos, Máté Nagy, Severin Leven, and Tamás Vicsek. Thermal soaring flight of birds and unmanned aerial vehicles. *Bioinspiration & Biomimetics*, 5(4), 2010.

Michael J. Allen. Autonomous soaring for improved endurance of small uninhabited air vehicle. In *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005.

Michael J. Allen. Updraft model for development of autonomous soaring uninhabited air vehicles. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, 2006.

Michael J Allen. Guidance and control of an autonomous soaring UAV. Technical report, NASA Dryden Flight Research Center, February 2007. URL `http://ntrs.nasa.gov/search.jsp?R=20070005019`.

Francesco Amigoni and Vincenzo Caglioti. An information-based exploration strategy for environment mapping with mobile robots. *Robotics and Autonomous Systems*, 58(5):684–699, 2010.

J. Andrew Bagnell. *Learning Decisions: Robustness, Uncertainty, and Approximation*. PhD thesis, Robotics Institute, Carnegie Mellon University, August 2004.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning*, pages 30–37. Morgan Kaufmann Publisher, Inc., 1995.

J Philip Barnes. Flight Without Fuel–Regenerative Soaring Feasibility Study. Technical report, SAE Technical Paper, 2006.

Richard E. Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716–719, 1952.

Richard E. Bellman. A problem in the sequential design of experiments. *Sankhya: The Indian Journal of Statistics (1933-1960)*, 16(3/4):221–229, 1956. ISSN 00364452.

Richard E. Bellman. *Dynamic Programming.* Princeton University Press, 1957. ISBN 9780691079516.

Ricardo Bencatel, João Tasso de Sousa, and Anouck Girard. Atmospheric flow field models applicable for aircraft endurance extension. *Progress in Aerospace Sciences*, 61:1–25, 2013.

Asher Bender, Stefan B. Williams, and Oscar Pizarro. Autonomous methods for environmental modeling and exploration. In *Proceedings of the 2013 Robotic Science and Systems Workshop on Robotic Exploration, Monitoring, and Information Collection: Nonparametric Modeling, Information-based Control, and Planning under Uncertainty*, 2013.

Jonathan Binney, Andreas Krause, and Gaurav S. Sukhatme. Optimizing waypoints for monitoring spatiotemporal phenomena. *The International Journal of Robotics Research*, 32(8):873–888, 2013.

Geoffrey. C. Bower, Tristan C. Flanzer, Alexander D. Naiman, and Suman Saripalli. Dynamic environment mapping for autonomous thermal soaring. In *AIAA Guidance, Navigation and Control Conference*, 2010.

Justin Boyan and Andrew W Moore. Generalization in reinforcement learning: Safely approximating the value function. *Advances in Neural Information Processing Systems*, pages 369–376, 1995.

Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Technical report, University of British Columbia, 2010. URL `http://arxiv.org/abs/1012.2599`.

George Cayley. On aerial navigation. *A Journal of Natural Philosophy, Chemistry and the Arts*, 24:164–174, 1809.

George Cayley. On aerial navigation. *A Journal of Natural Philosophy, Chemistry and the Arts*, 25:81–87, 1810a.

George Cayley. On aerial navigation. *A Journal of Natural Philosophy, Chemistry and the Arts*, 25:161–173, 1810b.

Girish Chowdhary, Miao Liu, Robert Grande, Thomas Walsh, Jonathan How, and Lawrence Carin. Off-Policy Reinforcement Learning with Gaussian Processes. *Acta Automatica Sinica, to appear*, 2014.

Jen Jen Chung, Miguel Ángel Trujillo, and Salah Sukkarieh. A new utility function for smooth transition between exploration and exploitation of a wind energy field. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4999–5005, 2012.

Jen Jen Chung, Nicholas R. J. Lawrance, and Salah Sukkarieh. Gaussian processes for informative exploration in reinforcement learning. In *2013 IEEE International Conference on Robotics and Automation*, pages 2633–2639, 2013.

Clarence D. Cone. Thermal soaring of birds. *American Scientist*, 50(1):180–209, 1962.

Clarence D. Cone. *A Mathematical Analysis of the Dynamic Soaring Flight of the Albatross: With Ecological Interpretations*. Virginia Institute of Marine Science, 1964.

Lehel Csató. *Gaussian Processes: Iterative Sparse Approximations*. PhD thesis, Aston University, 2002.

Lehel Csató and Manfred Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

Peter Dayan. The convergence of TD(λ) for general λ. *Machine Learning*, 8(3-4): 341–362, 1992.

Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian Q-learning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 761–768, 1998.

Marc P. Deisenroth. *Efficient Reinforcement Learning using Gaussian Processes*. PhD thesis, Karlsruhe Institute of Technology, 2010.

Marc P. Deisenroth, Carl Edward Rasmussen, and Jan Peters. Gaussian process dynamic programming. *Neurocomputing*, 72(7-9):1508–1524, 2009.

Daniel J. Edwards. Implementation details and flight test results of an autonomous soaring controller. 2008.

Daniel J. Edwards and Larry M. Silverberg. Autonomous soaring: The Montague cross-country challenge. *Journal of Aircraft*, 47(5):1763–1769, 2010.

Yaakov Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, The Hebrew University of Jerusalem, 2005.

Yaakov Engel, Shie Mannor, and Ron Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 154–161, 2003.

Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement learning with Gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 201–208, 2005.

Valerii Vadimovich Fedorov. *Theory of Optimal Experiments*. Academic Press, Inc., 1972.

Javier García and Fernando Fernández. Safe exploration of state and action spaces in reinforcement learning. *Journal of Artificial Intelligence Research*, 45(1): 515–564, 2012. ISSN 1076-9757.

Clement Gehring and Doina Precup. Smart exploration in reinforcement learning using absolute temporal difference errors. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 1037–1044, 2013. ISBN 978-1-4503-1993-5.

John C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2):148–177, 1979.

John C. Gittins and D. M. Jones. A dynamic allocation index for the sequential allocation of experiments. *Progress in Statistics*, pages 241–66, 1974.

Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in Gaussian processes. In *Proceedings of the 22nd international Conference on Machine Learning*, pages 265–272. ACM, 2005.

Alexander Hans, Daniel Schneegaß, Anton Maximilian Schäfer, and Steffen Udluft. Safe exploration for reinforcement learning. In *Proceedings of 16th European Symposium on Artificial Neural Networks*, pages 143–148, 2008.

Trong Nghia Hoang, Kian Hsiang Low, Patrick Jaillet, and Mohan Kankanhalli. Nonmyopic ϵ-Bayes-Optimal Active Learning of Gaussian Processes. In *Proceedings of the 31st International Conference on Machine Learning*, pages 739–747, 2014.

Gabriel M. Hoffmann and Claire J. Tomlin. Mobile sensor network control using mutual information methods and particle filters. *IEEE Transactions on Automatic Control*, 55(1):32–47, 2010. ISSN 0018-9286.

Geoffrey A. Hollinger, Brendan Englot, Franz S. Hover, Urbashi Mitra, and Gaurav S. Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *The International Journal of Robotics Research*, 32(1):3–18, 2013.

Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

Michael N. Katehakis and Arthur F. Veinott. The multi-armed bandit problem: Decomposition and computation. *Mathematics of Operations Research*, 12(2): 262–268, 1987.

Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Thomas Kollar and Nicholas Roy. Trajectory optimization using reinforcement learning for map exploration. *The International Journal of Robotics Research*, 27 (2):175–196, 2008.

Andreas Krause and Carlos Guestrin. Nonmyopic active learning of Gaussian processes: An exploration-exploitation approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 449–456. ACM, 2007.

Tze Leung Lai. Adaptive treatment allocation and the multi-armed bandit problem. *The Annals of Statistics*, 15(3):1091–1114, 1987.

Jack W. Langelaan. Long distance/duration trajectory optimization for small UAVs. In *2007 AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007.

Nicholas R. J. Lawrance. *Autonomous Soaring Flight for Unmanned Aerial Vehicles*. PhD thesis, The University of Sydney, 2011.

Nicholas R. J. Lawrance and Salah Sukkarieh. A guidance and control strategy for dynamic soaring with a gliding UAV. In *2009 IEEE International Conference on Robotics and Automation*, pages 3632–3637, 2009.

Nicholas R. J. Lawrance and Salah Sukkarieh. Simultaneous exploration and exploitation of a wind field for a small gliding UAV. In *AIAA Guidance, Navigation and Control Conference*, volume 8032, 2010.

Nicholas R. J. Lawrance and Salah Sukkarieh. Autonomous exploration of a wind field with a gliding aircraft. *Journal of Guidance, Control, and Dynamics*, 34(3), 2011a.

Nicholas R. J. Lawrance and Salah Sukkarieh. Path planning for autonomous soaring flight in dynamic wind fields. In *2011 IEEE International Conference on Robotics and Automation*, pages 2499–2505. IEEE, 2011b.

Daniel Levine, Brandon Luders, and Jonathan P. How. Information-rich path planning with general constraints using rapidly-exploring random trees. In *AIAA Infotech@ Aerospace Conference, Atlanta, GA*, 2010.

Otto Lilienthal. *Der Vogelflug als Grundlage der Fliegekunst : ein Beitrag zur Systematik der Flugtechnik / auf Grund zahlreicher von O. und G. Lilienthal ausgeführter Versuche ; bearbeitet von Otto Lilienthal.* R. Gaertner, 1889.

Michael Littman, Alain Dutech, Tim Edmunds, Jelle Kok, Michail Lagoudakis, Martin Riedmiller, Brian Russell, Bruno Scherrer, Rich Sutton, Stephan Timmer, Nikos Vlassis, Adam White, and Shimon Whiteson. NIPS Workshop: Reinforcement Learning Benchmarks and Bake-offs II. 2005.

Kian Hsiang Low, John M. Dolan, and Pradeep Khosla. Information-theoretic multi-robot adaptive exploration and mapping of environmental hotspot fields. In *ESSA 2009: Workshop on Sensor Networks for Earth and Space Sciences Applications*, 2009.

Paul B. MacCready. Optimum airspeed selector. *Soaring (January–February)*, pages 10–11, 1958.

D. R. Mackintosh. The use of thermal currents by birds on migration. *Ibis*, 91(1): 55–59, 1949. ISSN 1474-919X.

Roman Marchant and Fabio Ramos. Bayesian optimisation for intelligent environmental monitoring. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2242–2249, 2012.

Darryl E. Metzger and J. Karl Hedrick. Optimal flight paths for soaring flight. *Journal of Aircraft*, 12(11):867–871, 1975.

Teodor Mihai Moldovan and Pieter Abbeel. Safe exploration in Markov decision processes. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1): 103–130, 1993.

R. M. Neal. Regression and classification using Gaussian process priors. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Baysian Statistics 6*, pages 475–501. Oxford University Press, 1998.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978.

S. E. Peal. Soaring of birds. *Nature*, 23(575):10–11, 1880.

Jing Peng and Ronald J. Williams. Incremental multi-step Q-learning. *Machine Learning*, 22(1-3):283–290, 1996.

Carl Edward Rasmussen and Malte Kuss. Gaussian processes in reinforcement learning. *Advances in Neural Information Processing Systems 16*, 16:751–759, 2004.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

Lord Rayleigh. The soaring of birds. *Nature*, 27(701):534–535, 1883.

Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report, Cambridge University Engineering Department, 1994. URL `http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.2539`.

Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3:535–554, 1959.

Thomas J. Santner, Brian J. Williams, and William Notz. *The Design and Analysis of Computer Experiments*. Springer, 2003.

Jürgen Schmidhuber. Curious model-building control systems. In *1991 IEEE International Joint Conference on Neural Networks*, pages 1458–1463. IEEE, 1991.

Jeff G. Schneider. Exploiting model uncertainty estimates for safe dynamic control learning. In *Neural Information Processing Systems 9*, pages 1047–1053. The MIT Press, 1996.

Oliver G. Selfridge. Some themes and primitives in ill-defined systems. In *Adaptive Control of Ill-defined Systems*, pages 21–26. Springer, 1984.

B. S. Shenstone and S. Scott-Hall. Glider development in Germany: A technical survey of progress in design in Germany since 1922. *Aircraft Engineering and Aerospace Technology*, 7(10):249–258, 1935.

Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William J. Kaiser. Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research*, 34(2):707–755, 2009a.

Amarjeet Singh, Andreas Krause, and William J. Kaiser. Nonmyopic adaptive informative path planning for multiple robots. In *Proceedings of the 21st International Joint Conference on Artifical intelligence*, pages 1843–1850. Morgan Kaufmann Publishers Inc., 2009b.

Amarjeet Singh, Fabio Ramos, Hugh Durrant Whyte, and William J. Kaiser. Modeling and decision making in spatio-temporal processes for environmental surveillance. In *IEEE International Conference on Robotics and Automation*, pages 5490–5497. IEEE, 2010.

Satinder P. Singh and Richard S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1-3):123–158, 1996.

Niranjan Srinivas, Andreas Kraus, Sham M. Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the International Conference on Machine Learning*, 2010.

Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using Rao-Blackwellized particle filters. In *Proceedings of Robotics: Science and Systems*, 2005.

Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning*, pages 216–224, 1990.

Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044. MIT Press, 1996.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Richard S. Sutton and Satinder P. Singh. On step-size and bias in temporal-difference learning. In *The Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems*, pages 91–96, 1994.

Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Sebastian Thrun and Knut Möller. Active exploration in dynamic environments. In *Advances in Neural Information Processing Systems 4*, pages 531–538, 1992.

Sebastian B. Thrun. The role of exploration in learning control. In David A. White and Donald A. Sofge, editors, *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, 1992a.

Sebastian B. Thrun. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, School of Computer Science, Carnegie-Mellon University, January 1992b. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.4011`.

John N. Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, 1989.

H. Weimerskirch, T. Guionnet, J. Martin, S. A. Shaffer, and D. P. Costa. Fast and fuel efficient? Optimal use of wind by flying albatrosses. *Proceedings of the Royal Society of London - Biological Sciences*, 267(1455):1869–1874, 2000.

John M. Wharington. *Autonomous Control of a Soaring Aircraft by Reinforcement Learning*. PhD thesis, Department of Aerospace Engineering, Royal Melbourne Institute of Technology, 1998.

John M. Wharington. Heuristic control of dynamic soaring. In *Control Conference, 2004. 5th Asian*, volume 2, pages 714–722, 2004.

Christopher K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*, pages 599–621. Kluwer Academic Press, 1998.

C. J. Wood. The flight of albatrosses (A computer simulation). *Ibis*, 115(2): 244–256, 1972.

Tim Woodbury, Caroline Dunn, and John Valasek. Autonomous Soaring Using Reinforcement Learning for Trajectory Generation. In *5nd Aerospace Sciences Meeting*, pages 1–11, 2014.

Appendix A

Simulation Specifications

A.1 Puddle World

Learning parameters for the puddle world simulation set are shown in Table A.1.

Table A.1: Puddle World Learning Parameters

Description	**Symbol**	**Value**
Time discretisation	Δt	1 s
Max steps		100
Exploration parameter	ε	0.01
Exploration value decay parameter	τ_r	20
Reward discount factor	γ	0.9
Trace discount factor	λ	0.9
Rollout discount factor	γ_r	0.4
Step size	α	0.5
State	s	$[x, y]$
Action	a	$\{up, down, left, right\}$
		$\sim \mathcal{N}(0.05, 0.01^2)$
Tilings	m	10
Partitions		10

A.2 Cart Pole

The constants used to compute the dynamics of the cart pole problem are given in Table A.2, while the learning parameters used across the simulation set are shown in Table A.3.

Table A.2: Cart Pole Problem Constants

Description	**Symbol**	**Value**	**Unit**
Gravity	g	9.81	$\mathrm{m/s^2}$
Cart mass	m_c	1	kg
Pole mass	m_p	0.1	kg
Pole length	l_p	0.5	m

Table A.3: Cart Pole Learning Parameters

Description	**Symbol**	**Value**
Time discretisation	Δt	0.02 s
Max steps		500
Exploration parameter	ε	0.01
Exploration value decay parameter	τ_r	20
Reward discount factor	γ	0.9
Trace discount factor	λ	0.9
Rollout discount factor	γ_r	0.4
Step size	α	0.5
State	s	$[x, v, \theta, \omega]$
Action	a	$\in \{-10, -9, \ldots, 10\}$ N
Tilings	m	100
Partitions		5

A.3 Battery Cycling

The learning parameters used across the battery cycling simulation set are shown in Table A.4.

Table A.4: Battery Cycling Learning Parameters

Description	**Symbol**	**Value**
Time discretisation	Δt	0.5 s
Max steps		1000
Exploration parameter	ε	0.01
Exploration value decay parameter	τ_r	500
Reward discount factor	γ	0.9
Trace discount factor	λ	0.7
Rollout discount factor	γ_r	0.4
Step size	α	0.5
State	s	$[energy]$
Action	a	$\in \{charge, discharge\}$
Tilings	m	100
Partitions		50

A.4 2D 3DOF Soaring Glider

The learning parameters used across the 2D 3DOF soaring glider simulation set are shown in Table A.5.

Table A.5: 2D 3DOF Soaring Glider Learning Parameters

Description	**Symbol**	**Value**
Time discretisation	Δt	1 s
Max steps		10000
Exploration parameter	ε	0.01
Exploration value decay parameter	τ_r	5000
Reward discount factor	γ	0.9
Trace discount factor	λ	0.7
Rollout discount factor	γ_r	0.4
Step size	α	0.5
State	s	$[x, y, \psi]$
Action	a	$\in \{left, straight, right\}$
Maximum energy	E_{max}	4.002×10^5 J

A.5 3D 6DOF Soaring Glider

The glider platform simulated in the experiments was modelled on the RnR SBXC glider. The aircraft parameters used in the simulations are given in Table A.6. The learning parameters used across the 3D 6DOF soaring glider simulation set are shown in Table A.7.

Table A.6: Glider Aircraft Parameter Values

Aircraft Parameter	**Value**
Parasitic drag coefficient	0.012
Wing reference area	$0.95677\,\mathrm{m}^2$
Wing aspect ratio	19.54
Oswald's efficiency factor	0.85
Vehicle mass	5.44 kg
Maximum positive load factor	2.0
Maximum negative load factor	0
Maximum lift coefficient	1.2
Maximum roll rate	30° /s
Maximum roll angle	60°
Maximum air relative climb angle	50°
Approximate glide ratio	30
Stall speed	9.54 m/s
Maximum allowable airspeed	36.6 m/s

Table A.7: 3D 6DOF Soaring Glider Learning Parameters

Description	**Symbol**	**Value**
Time discretisation	Δt	1 s
Max steps		500
Exploration parameter	ε	0.01
Exploration value decay parameter	τ_r	100
Reward discount factor	γ	0.9
Trace discount factor	λ	0.7
Rollout discount factor	γ_r	0.4
Step size	α	0.5
State	s	$[r, v, \psi]$
Action	a	$\left[\dot{\phi}, \dot{\theta}\right]$
		$\dot{\phi} \in \{left, straight, right\}$
		$\dot{\theta} \in \{up, straight, down\}$
Maximum energy	E_{max}	1.965×10^4 J

Appendix B

Derivation of the GP Variance Volume

Given the squared exponential covariance function, shown in Equation (3.20) and repeated here,

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_f^2 \exp\left[-\frac{1}{2}\left(\mathbf{x}-\mathbf{x}'\right)^T M \left(\mathbf{x}-\mathbf{x}'\right)\right], \tag{B.1}$$

the variance volume for a set of n-dimensional observations X_N can be derived as follows:

$$\begin{aligned}
V_{bound} &= \int_{\mathbf{x}_a}^{\mathbf{x}_b} \operatorname{cov}\left(\mathbf{x}\right) \mathrm{d}\mathbf{x} \\
&= \int_{x_{n_a}}^{x_{n_b}} \cdots \int_{x_{1_a}}^{x_{1_b}} \operatorname{cov}\left(\left[x_1, \ldots x_n\right]\right) \mathrm{d}x_1 \ldots \mathrm{d}x_n \\
&= \int_{x_{n_a}}^{x_{n_b}} \cdots \int_{x_{1_a}}^{x_{1_b}} k\left(\left[x_1, \ldots, x_n\right], \left[x_1, \ldots, x_n\right]\right) \\
&\quad - k\left(\left[x_1, \ldots, x_n\right], X_N\right) K_{XX}^{-1} k\left(X_N, \left[x_1, \ldots, x_n\right]\right) \mathrm{d}x_1 \ldots \mathrm{d}x_n.
\end{aligned} \tag{B.2}$$

In the interest of space and clarity, the following equations will refer to the vector form of $\mathbf{x} = \left[x_1, \ldots x_n\right]$ in the covariance function terms and the integral limits.

Equation (B.2) can be split into two terms, with the covariance function Equation (B.1), the first term gives:

$$\begin{aligned}\int_{\mathbf{x}_a}^{\mathbf{x}_b} k\left(\mathbf{x},\mathbf{x}\right)\mathrm{d}\mathbf{x} &= \int_{\mathbf{x}_a}^{\mathbf{x}_b} \sigma_f^2 \mathrm{d}\mathbf{x} \\ &= \sigma_f^2 \times \left(x_{1_b} - x_{1_a}\right) \times \ldots \times \left(x_{n_b} - x_{n_a}\right) \\ &= \sigma_f^2 \prod_{m=1}^{n} \left(x_{m_b} - x_{m_a}\right). \end{aligned} \tag{B.3}$$

To compute the general form of the second integral term, consider

$$\begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} c & d \\ e & f \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = aca + bea + adb + bfb,$$

which leads to the equation,

$$k\left(\mathbf{x}, X_N\right) K_{XX}^{-1} k\left(X_N, \mathbf{x}\right) = \sum_i \sum_j k\left(\mathbf{x}, \mathbf{x}_i\right) \left[K_{XX}^{-1}\right]_{ij} k\left(\mathbf{x}_j, \mathbf{x}\right). \tag{B.4}$$

Since $\left[K_{XX}^{-1}\right]$ is not a function of $\mathbf{x}$, then the second integral term in Equation (B.2) can be written as,

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \left[K_{XX}^{-1}\right]_{ij} \int_{\mathbf{x}_a}^{\mathbf{x}_b} k\left(\mathbf{x}, \mathbf{x}_i\right) k\left(\mathbf{x}_j, \mathbf{x}\right) \mathrm{d}\mathbf{x}, \tag{B.5}$$

where in a slight abuse of notation, $\mathbf{x}_i$ refers to the i-th observation in the set of observations X_N. The components of this integral can be expanded to aid clarification, giving the vectors

$$\begin{aligned} k\left(\mathbf{x}, X_N\right) &= \begin{bmatrix} k\left(\mathbf{x}, \mathbf{x}_1\right) & \ldots & k\left(\mathbf{x}, \mathbf{x}_N\right) \end{bmatrix}, \\ k\left(X_N, \mathbf{x}\right) &= \begin{bmatrix} k\left(\mathbf{x}, \mathbf{x}_1\right) \\ \vdots \\ k\left(\mathbf{x}, \mathbf{x}_N\right) \end{bmatrix}, \end{aligned}$$

since the covariance matrix is symmetrical. These can be used to give a matrix representation of the integral in Equation (B.5),

$$C = \begin{bmatrix} \int_{\mathbf{x}_a}^{\mathbf{x}_b} k\left(\mathbf{x}, \mathbf{x}_1\right) k\left(\mathbf{x}_1, \mathbf{x}\right) \mathrm{d}\mathbf{x} & \dots & \int_{\mathbf{x}_a}^{\mathbf{x}_b} k\left(\mathbf{x}, \mathbf{x}_1\right) k\left(\mathbf{x}_N, \mathbf{x}\right) \mathrm{d}\mathbf{x} \\ \vdots & \ddots & \vdots \\ \int_{\mathbf{x}_a}^{\mathbf{x}_b} k\left(\mathbf{x}, \mathbf{x}_N\right) k\left(\mathbf{x}_1, \mathbf{x}\right) \mathrm{d}\mathbf{x} & \dots & \int_{\mathbf{x}_a}^{\mathbf{x}_b} k\left(\mathbf{x}, \mathbf{x}_N\right) k\left(\mathbf{x}_N, \mathbf{x}\right) \mathrm{d}\mathbf{x} \end{bmatrix}, \tag{B.6}$$

such that the second integral term in Equation (B.2) becomes,

$$\int_{\mathbf{x}_a}^{\mathbf{x}_b} k\left(\mathbf{x}, X_N\right) K_{XX}^{-1} k\left(X_N, \mathbf{x}\right) \mathrm{d}\mathbf{x} = \sum_{i=1}^{N} \sum_{j=1}^{N} \left[K_{XX}^{-1} \cdot C\right]_{ij}. \tag{B.7}$$

The individual elements in C can be solved for by investigating the relevant covariance function, in this case, the squared exponential function shown in Equation (B.1). For the i, j-th integral element in C,

$$\begin{aligned} & k\left(\mathbf{x}, \mathbf{x}_i\right) k\left(\mathbf{x}_j, \mathbf{x}\right) \\ & = \sigma_f^2 \exp\left[-\frac{1}{2}\left(\mathbf{x} - \mathbf{x}_i\right)^T M \left(\mathbf{x} - \mathbf{x}_i\right)\right] \sigma_f^2 \exp\left[-\frac{1}{2}\left(\mathbf{x}_j - \mathbf{x}\right)^T M \left(\mathbf{x}_j - \mathbf{x}\right)\right] \\ & = \sigma_f^2 \exp\left\{-\frac{1}{2}\left[\left(\frac{x_1 - x_{i_1}}{l_1}\right)^2 + \ldots + \left(\frac{x_n - x_{i_n}}{l_n}\right)^2\right]\right\} \\ & \quad \times \sigma_f^2 \exp\left\{-\frac{1}{2}\left[\left(\frac{x_{j_1} - x_1}{l_1}\right)^2 + \ldots + \left(\frac{x_{j_n} - x_n}{l_n}\right)^2\right]\right\} \\ & = \sigma_f^4 \exp\left\{-\left[\frac{\left(x_1 - x_{i_1}\right)^2 + \left(x_{j_1} - x_1\right)^2}{2 l_1^2} + \ldots + \frac{\left(x_n - x_{i_n}\right)^2 + \left(x_{j_n} - x_n\right)^2}{2 l_n^2}\right]\right\}, \end{aligned} \tag{B.8}$$

remembering that M is the diagonal matrix of squared length scales $\mathrm{diag}\left(M\right) = \mathbf{l}^{-2} = \left[l_1^{-2}, \ldots, l_n^{-2}\right]$. This expression can be simplified by completing the square for each element in the exponent (the dimensional subscript is removed here for ease of

reading):

$$\frac{(x-x_i)^2+(x_j-x)^2}{2l^2}=\left(\frac{x-\frac{x_i+x_j}{2}}{l}\right)^2+\left(\frac{x_i-x_j}{2l}\right)^2, \tag{B.9}$$

and recognising now that σ_f^4 and the second term in Equation (B.9) are not functions of $\mathbf{x}$ and so can be taken out of the integral, giving,

$$\int_{\mathbf{x}_a}^{\mathbf{x}_b} k(\mathbf{x},\mathbf{x}_i)\,k(\mathbf{x}_j,\mathbf{x})\,\mathrm{d}\mathbf{x}=\sigma_f^4\exp\left\{-\left[\left(\frac{x_{i_1}-x_{j_1}}{2l_1}\right)^2+\ldots+\left(\frac{x_{i_n}-x_{j_n}}{2l_m}\right)^2\right]\right\}$$
$$\times\int_{x_{n_a}}^{x_{n_b}}\cdots\int_{x_{1_a}}^{x_{1_b}}\exp\left\{-\left[\left(\frac{x_1-\frac{x_{i_1}+x_{j_1}}{2}}{l_1}\right)^2+\ldots+\left(\frac{x_n-\frac{x_{i_n}+x_{j_n}}{2}}{l_n}\right)^2\right]\right\}\mathrm{d}x_1\ldots\mathrm{d}x_n. \tag{B.10}$$

The integral in Equation (B.10) can now be separated into the constituent dimensions,

$$\int_{\mathbf{x}_a}^{\mathbf{x}_b} k(\mathbf{x},\mathbf{x}_i)\,k(\mathbf{x}_j,\mathbf{x})\,\mathrm{d}\mathbf{x}=\sigma_f^4\exp\left[-\sum_{m=1}^{n}\left(\frac{x_{i_m}-x_{j_m}}{2l_m}\right)^2\right]$$
$$\times\int_{x_{n_a}}^{x_{n_b}}\exp\left[-\left(\frac{x_n-\frac{x_{i_n}+x_{j_n}}{2}}{l_n}\right)^2\right]\mathrm{d}x_n\times\cdots\times\int_{x_{1_a}}^{x_{1_b}}\exp\left[-\left(\frac{x_1-\frac{x_{i_1}+x_{j_1}}{2}}{l_1}\right)^2\right]\mathrm{d}x_1. \tag{B.11}$$

From here (dropping the dimensional subscript again for clarity) performing the following substitution,

$$t=\frac{x-\frac{x_i+x_j}{2}}{l}, \tag{B.12}$$

such that,

$$\mathrm{d}t=\frac{\mathrm{d}x}{l},$$

and when

$$x = x_a, \quad t = \frac{x_a - \frac{x_i + x_j}{2}}{l}, \qquad x = x_b, \quad t = \frac{x_b - \frac{x_i + x_j}{2}}{l},$$

allows the integration:

$$\int_{x_a}^{x_b} \exp\left[-\left(\frac{x - \frac{x_i+x_j}{2}}{l}\right)^2\right] \mathrm{d}x = l\left(\frac{\sqrt{\pi}}{2}\right)\left[\mathrm{erf}\left(\frac{x_b - \frac{x_i+x_j}{2}}{l}\right) - \mathrm{erf}\left(\frac{x_a - \frac{x_i+x_j}{2}}{l}\right)\right]. \tag{B.13}$$

Substituting back into Equations (B.11), (B.6) and (B.7) gives the solution to the second integral term of Equation (B.2):

$$\begin{aligned}
&\int_{\mathbf{x}_a}^{\mathbf{x}_b} k\left(\mathbf{x}, X_N\right) K_{XX}^{-1} k\left(X_N, \mathbf{x}\right) \mathrm{d}\mathbf{x} \\
&= \sigma_f^4 \left(\frac{\sqrt{\pi}}{2}\right)^n \left(l_1 \times \ldots \times l_n\right) \times \sum_{i=1}^{N}\sum_{j=1}^{N} \left\{ \left[K_{XX}^{-1}\right]_{ij} \exp\left[-\sum_{m=1}^{n}\left(\frac{x_{i_m} - x_{j_m}}{2l_m}\right)^2\right] \right. \\
&\quad \times \left[\mathrm{erf}\left(\frac{x_{1_b} - \frac{x_{i_1}+x_{j_1}}{2}}{l_1}\right) - \mathrm{erf}\left(\frac{x_{1_a} - \frac{x_{i_1}+x_{j_1}}{2}}{l_1}\right)\right] \times \ldots \\
&\quad \left. \times \left[\mathrm{erf}\left(\frac{x_{n_b} - \frac{x_{i_n}+x_{j_n}}{2}}{l_n}\right) - \mathrm{erf}\left(\frac{x_{n_a} - \frac{x_{i_n}+x_{j_n}}{2}}{l_n}\right)\right]\right\}.
\end{aligned} \tag{B.14}$$

Combining Equation (B.14) with Equation (B.3) gives the final result:

$$\begin{aligned}
V_{bound} = \sigma_f^2 \prod_{m=1}^{n}\left(x_{m_b} - x_{m_a}\right) - \sigma_f^4 \left(\frac{\sqrt{\pi}}{2}\right)^n \prod_{m=1}^{n}\left(l_m\right) \\
\times \sum_{i=1}^{N}\sum_{j=1}^{N} \left\{ \left[K_{XX}^{-1}\right]_{ij} \exp\left[-\sum_{m=1}^{n}\left(\frac{x_{i_m} - x_{j_m}}{2l_m}\right)^2\right] \right. \\
\left. \times \prod_{m=1}^{n}\left[\mathrm{erf}\left(\frac{x_{m_b} - \frac{x_{i_m}+x_{j_m}}{2}}{l_m}\right) - \mathrm{erf}\left(\frac{x_{m_a} - \frac{x_{i_m}+x_{j_m}}{2}}{l_m}\right)\right]\right\}.
\end{aligned} \tag{B.15}$$

Note that the only requirement to finding an analytical solution to Equation (B.2) is that the covariance function multiplication $k\left(\mathbf{x}, \mathbf{x}_i\right) k\left(\mathbf{x}_j, \mathbf{x}\right)$ be integrable. Significant

computational speed-up can also be achieved by keeping track of the matrices K_{XX}^{-1} and C. Since the GP approximation of the value function already uses the block update form to maintain the inverse covariance matrix, all that remains is to append the new row and column of C for each new observation; furthermore, the computation of these elements is linear in the number of observations since the the matrix is symmetrical.